UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"SISTEMA IOT PARA EL CONTROL DE ALIMENTACIÓN DE COTURNIX JAPONICA (CODORNIZ JAPONESA)"

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

Línea de investigación: Producción industrial y tecnología sostenible

AUTOR:
Edison Steev Quilca Serrano

DIRECTOR:
Msc. Jaime Roberto Michilena Calderón

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1.IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004592968		
APELLIDOS Y NOMBRES:	Quilca Serrano Edison Steev		
DIRECCIÓN:	Sector Chiricorral, avenida Atahualpa vía San Clemente		
EMAIL:	esquilcas@utn.edu.ec		
TELÉFONO FIJO:	No registra	**TELF. MOVIL**	0939321676

DATOS DE LA OBRA	
TÍTULO:	Sistema IoT para el control de alimentación de coturnix japonica (codorniz japonesa)
AUTOR (ES):	Quilca Serrano Edison Steev
FECHA: DD/MM/AAAA	28/01/2026
SOLO PARA TRABAJOS DE GRADO	
CARRERA/PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Telecomunicaciones
DIRECTOR:	Msc. Jaime Roberto Michilena Calderón
ASESOR:	Msc. Carlos Alberto Vásquez Ayala

2.CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de enero del 2026

EL AUTOR:

Quilca Serrano Edison Steev

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 28 de enero del 2026

MSC. JAIME ROBERTO MICHILENA CALDERÓN

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Firmado electrónicamente por:
JAIME ROBERTO MICHILENA CALDERON
Validar únicamente con FirmaEC

(f) ……

Msc. Jaime Roberto Michilena Calderón

C.C.:1002198438

DEDICATORIA

Este logro quiero dedicar a todas las personas que fueron y son parte fundamental en mi vida, principalmente a Dios, por darme la vida, la fuerza en los momentos difíciles, la sabiduría ya que, sin su presencia constante, este camino no habría sido posible.

Dedico este logro con todo mi amor y con mucho cariño a mis más grandes tesoros a mi madre Inés Serrano y a mi padre Carlos Quilca, quienes han sido mi mayor ejemplo de amor, esfuerzo, sacrificio y superación, gracias por su apoyo incondicional, mamita gracias por no dejarme caer en este camino, ya que sus palabras y su amor no me dejaron desmayar y me motivaron a seguir adelante, papito igual gracias por su apoyo y amor incondicional, por todo el esfuerzo que ha realizado por mí, por eso y por mucho más este logro va dedicado a ustedes, les quiero mucho.

A mis hermanas Katherine, Lizbeth, Joselyn, Anshely y a mi hermanito Carlitos, quienes en mi corazón seguirán siendo mis hermanitos pequeños, quiero que sepan que gracias a su apoyo pude cumplir esta meta, además, quiero que sepan que los quiero mucho.

A mi novia Saigua, por ser aquella persona que supo apoyarme desde el día que la conocí, quiero agradecerte por tu amor, paciencia y comprensión, por cada gesto que me impulsó a continuar incluso cuando ni yo me creía capaz, te amo mucho, también a tu familia, por su acogida y apoyo muchas gracias.

A los perritos Zeus, Negro, Tayson y gatito Nené, en especial a mis perritos Viejito y Balú por ser aquellos compañeros que siempre me esperaban alegres al llegar a casa luego de noches de trabajo y estudio, a ti Viejito, aunque ya nos estas aquí, te llevaré por siempre en mi corazón, gracias.

Edison Steev Quilca Serrano

AGRADECIMIENTO

Agradezco a Dios por permitir que mis padres y hermanas/no estén a mi lado en este logro tan importante en mi vida, por ser mi guía en los momentos donde el camino se tornaba difícil y por brindarme la fortaleza para vencer las barreras que se presentaban durante este viaje.

Agradezco a mí mismo por nunca rendirme, por ser perseverante y por superar etapas donde ni tu creías poder superar, por todo eso te admiro mucho y sé que en futuro lograras cosas grandes.

Al Ingeniero Jaime Michilena, tutor docente de este trabajo, gracias por confiar en mí durante este proceso, su guía fue clave para concretar cada parte del trabajo, gracias por la calidad de ser humano y por motivarme a seguir adelante.

Al docente asesor, el Ingeniero Carlos Vásquez, por su compromiso, y por su apoyo durante el desarrollo del proyecto, por brindarme siempre el apoyo necesario para culminar con éxito este trabajo.

A los docentes de la carrera de Telecomunicaciones, por su dedicación y por formar una base sólida en mi preparación profesional, cada clase impartida dejó una huella en mi formación, quiero agradecer especialmente a la Ingeniera Ana Umaquinga y a la Ingeniera Pamela Godoy, quienes siempre confiaron en mí y me impulsaron a superar mis miedos y obstáculos que se presentaban en el camino.

Finalmente quiero agradecer a todas las personas que influyeron y conocí durante toda mi etapa universitaria, en especial a mis grandes amigos Magali, Melanie, Andrés, Marlon y Henry, con quienes compartí no solo aprendizajes, sino también risas, desvelos y momentos que quedarán marcados en esta etapa tan importante de mi vida.

Edison Steev Quilca Serrano

RESUMEN EJECUTIVO

El presente trabajo de integración curricular aborda el problema de la falta de control y atención en la alimentación adecuada de las codornices japonesas, trayendo consigo una baja producción, además de pérdidas económicas debido a un inadecuado manejo de la alimentación, por lo tanto, con el fin de tener un mejor control de la alimentación y mejorar la producción de huevos se plantea como objetivo principal implementar un sistema IoT para el control de alimento de codornices, para lograrlo se hizo uso de la metodología en cascada la cual abarca distintas fases empezando por el análisis de requerimientos hasta llegar a la de mantenimiento.

El sistema se basa en dos subsistemas uno enfocado a la parte de dispensación de alimento y el otro a la dispensación de agua, además del conteo de huevos lo que permite registrar de forma automática la producción diaria. Para ello se hace el uso de la integración de componentes electrónicos, además de plataformas virtuales los cuales permiten enviar notificaciones de acuerdo a estados críticos del sistema y el monitoreo constante de los niveles de alimento, agua y el conteo real de huevos producidos.

De acuerdo a las pruebas realizadas y resultados obtenidos se pudo constatar que el sistema mide correctamente los niveles de alimento y agua, proporciona la cantidad de alimento adecuada de acuerdo a un horario específico, además, permite la visualización de datos y envío de notificaciones en tiempo real, además mediante las pruebas realizadas se evidenció que mediante el método tradicional se obtuvieron 570 huevos, mientras que con la implementación del sistema IoT de control de alimento se registró un total de 852 lo que evidencia una mayor producción con la implementación del sistema desarrollado.

Finalmente, el aumento de la productividad no se evidenció únicamente a través del aumento en la cantidad de huevos producidos, sino también por mejoras en aspectos como en la higiene del criadero, el ahorro del tiempo por la disminución de la intervención manual dedicadas al manejo de actividades en el criadero y organización de los procesos. Con esto no solo se mejoró la producción de huevos sino también la productividad general siendo de mucho beneficio para el propietario de las codornices.

Palabras clave: codornices japonesas, producción, huevos, dispensación, monitoreo, nivel de agua, nivel de comida.

ABSTRACT

The present curricular integration project addresses the problem of the lack of control and attention in the appropiate feeding of Japanese quails, resulting in low production, in addition to economic losses due to inadequate feeding management, therefore in order to have better control of feeding and improve egg production, the main objective is to implement an IoT system for quail feed control. To achieve this objective, the waterfall methodology was used, which covers different phases, starting from requirements analysis and ending with the maintenace phase.

The system is composed of two subsystems. One is focused on feed dispensing, while the other is responsible for water dispensing. Additionally, an egg counting mechanism is included, allowing the automatic recording of daily production. For this purpose, the integration of electronic components is employed, in addition to virtual platforms that allow notifications to be sent based on critical system states and enable continuous monitoring of feed levels, water levels, and the actual number of eggs produced.

According to the tests carried out and the results obtained, it was confirmed that the system correctly measures feed and water levels, supplies the appropriate amount of feed according to a specific schedule, and allows data visualization and real-time notification delivery. In addition, the tests showed that using the traditional method, a total of 570 eggs were obtained, whereas with the implementation of the IoT-based feed control system, a total of 852 eggs were recorded, which demonstrates a higher level of production achieved through the implementation of the developed system.

Finally, the increase in productivity was not only evidenced by the growth in the number of eggs produced, but also through improvements in aspects such as farm hygiene, time savings due to the reduction of manual intervention in daily management activities, and better organization of production processes. As a result, not only was egg production improved, but overall productivity was also enhanced, providing significant benefits for the quail farm owner.

Keywords: Japanese quails, production, eggs, dispensing, monitoring, water level, feed level, eggs.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo I: Antecedentes

1.1 Tema

SISTEMA IOT PARA EL CONTROL DE ALIMENTACIÓN DE COTURNIX JAPONICA (CODORNIZ JAPONESA).

1.2 Problema

En la parroquia rural de La Esperanza, situada en el cantón Ibarra de la provincia de Imbabura, se lleva a cabo la crianza de codornices por varias familias locales. Esta actividad no solo constituye una fuente de ingresos adicional, sino también un pilar fundamental para la economía doméstica, gracias a las ventas de los huevos que generan, sin embargo, debido a otras actividades principales en el hogar, los miembros de la familia no pueden dedicar el tiempo y la atención necesarios para mantener la alimentación adecuada de las codornices, trayendo consigo una baja producción y mala calidad de huevos, además de pérdidas económicas debido a un inadecuado manejo de la alimentación, según (Martínez Zambrano, 2013) la mala alimentación trae consigo diversos problemas como la desnutrición o malnutrición resultante de la falta de nutrientes esenciales lo que compromete en la salud y el desarrollo adecuado de las aves, llevando a una disminución en la producción de huevos al no proporcionar los nutrientes necesarios para mantener una alta tasa de puesta.

El proceso de formación es complejo y comprende desde la ovulación hasta la puesta del huevo; se demora aproximadamente 21 a 23 horas. Para llegar a una adecuada formación el huevo, debe cumplir con ciertas técnicas que se debe llevar diariamente y así cumplir con las normas de bienestar animal como: tener un alimento de buena calidad con altos nutrientes, mantenerlas en un ambiente optimo y tranquilo libre de estrés. (Satan Chuim, 2020, como se cita en (Valle Muñoz, Bustamante Castro, Rodríguez, Vivas, & Guillet, Manual crianza y

manejo de codornices, 2015). Siendo la alimentación una de las técnicas fundamentales para poder obtener huevos de calidad.

Según (Martínez Zambrano, 2013) menciona que las codornices requieren siempre de un alimento rico en proteínas además de una dieta balanceada que puede estar integrada por maíz, alfalfa además de requerir suplementos vitamínicos y minerales como calcio, fósforo, sodio, cloro.

Según un estudio realizado por (Quiroz Martínez, 2017) menciona que algunas empresas comercializadoras de alimentos concentrados fabrican balanceados especiales para codornices, como lo es BIOMENTOS que tiene sucursales a nivel nacional.

Cada codorniz consume 25g de concentrado tanto para las codornices de engorde como para las de postura aquí en la ciudad de Ibarra, puede variar en otras ciudades de acuerdo al clima, si las aves están demasiado pesadas se debe proporcionar una reducción en la ración alimenticia del 10% al 15% esto ocasionara que bajara el peso corporal, por el contrario las aves estuvieran muy delgadas se debe aumentar 10% en la ración alimenticia, hay que tener en cuenta los desperdicios de comida para lo cual se debe suministrar el alimento tres veces al día (Quiroz Martínez, 2017).

Según un estudio realizado por (Agudelo, Víctor Libardo, & Torres Novoa, 2021), menciona que se puede llegar a una incorporación de alimentos no convencionales en las dietas de codornices mismo que se realiza con el propósito de reducir los costos de producción, mejorar la rentabilidad, sin afectar la calidad del huevo, en su estudio llegaron a la conclusión que el arroz partido puede sustituir en 50% al maíz, en la fabricación de raciones para codornices japonesas en fase inicial de postura sin afectar la producción de huevos además que el afrecho de yuca, las harinas de arroz y de plátano en sustitución al 50% del maíz, proporcionan bajos índices de desempeño productivo.

Como se mencionó anteriormente la alimentación adecuada es primordial durante la etapa de puesta para obtener huevos de calidad, es por ello que con ayuda de las tecnologías emergentes como el internet de las cosas se puede dar un control adaptado para la realidad de las familias que se dedican a esta actividad , con ello se optimizará el tiempo, se aumentará la calidad y producción de huevos además de ayudar en la economía del hogar al evitar desperdicio de comida por la alimentación de forma tradicional.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un sistema IoT para el control de alimento de codornices para mejorar la producción de huevos.

1.3.2 Objetivos Específicos

- Analizar datos bibliográficos enfocados en la alimentación de las codornices y la etapa de puesta, así como también el estado del arte de sistemas IoT aplicados al sector avícola.
- Diseñar un sistema IoT integrado con módulos, sensores, componentes electrónicos y plataformas virtuales que pueda recolectar datos en tiempo real y transmitirlos de manera eficiente a un dispositivo final para el control de alimentación por parte del usuario.
- Determinar los requerimientos necesarios para la implementación del sistema IoT en la jaula de las codornices el cual cumpla con el control de alimento de acuerdo a las porciones diarias necesarias durante la etapa de puesta.
- Realizar pruebas que permitan verificar la validez del sistema desarrollado.

1.4 Alcance

El presente trabajo de integración curricular tiene como finalidad la creación de un sistema IoT que permita el control de alimentación de las codornices en jaulas, para ello se procede con la recopilación de datos bibliográficos sobre la etapa de puesta de las codornices y su alimentación durante esta etapa, además de analizar qué tipo de alimentación será la más óptima para obtener huevos de mejor calidad y una mayor producción, el sistema integrará varios componentes como módulos, sensores, componentes electrónicos y plataformas virtuales con el fin de que el usuario pueda monitorear el abastecimiento de comida en el contenedor y la alimentación automática a las codornices con las porciones correctas.

Este sistema enviará una notificación por medio de una plataforma de mensajería al teléfono celular del usuario cuando el nivel de alimento y agua de los contenedores llegue a un punto inferior para abastecer nuevamente de forma manual el contenedor con alimento y agua, además se realizará un sistema con el uso de componentes electrónicos que permita dispensar los gramos de alimentación de manera automática necesarios de acuerdo al número de codornices presentes en la jaula, con esto se podrá optimizar tiempo, ahorros económicos debido a un mejor control de la alimentación. Además, como punto final se añadirá una jaula que permita que los huevos se almacenen de manera que el usuario pueda recolectarlos en un horario específico.

Para el presente trabajo de integración curricular se optará por la metodología en cascada. En este modelo los procesos que se llevan a cabo sugieren un orden secuencial, para el desarrollo de un sistema, en donde se comienza con el análisis de requerimientos, al terminar cada fase se continúa con la siguiente de manera sucesiva hasta llegar a la de mantenimiento. (Garduño Aparicio, 2015)

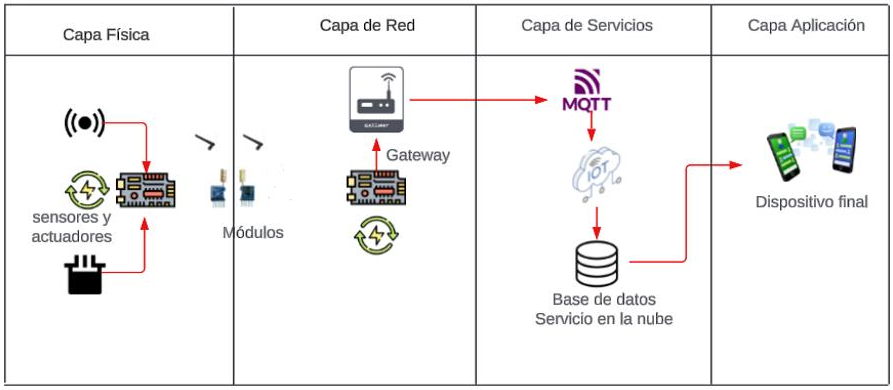
I. Análisis de requerimientos

En esta fase se dará inicio con el análisis de fuentes bibliográficas que hagan énfasis en la coturnicultura, donde se analizarán aspectos importantes sobre la correcta alimentación de las codornices, además de mencionar que tipo de alimento es el adecuado y cuáles son las porciones correctas durante la etapa de puesta, de igual modo se analizará distintos sistemas de alimentación IoT enfocados al área avícola que permitan tener una mejor visión sobre los componentes que la integran los cuales serán útiles para implementar en el sistema propuesto.

II. Diseño del sistema

Figura 1

Diagrama de arquitectura del sistema IoT



Luego de tener los conceptos bibliográficos claros podemos observar en la Figura 1. La arquitectura que se implementará en el sistema IoT mismo que se lo realizará mediante el uso módulos, sensores, componentes electrónicos y plataformas virtuales mismos que serán colocados en un punto estratégico de la jaula de las codornices, además se debe mencionar que se utilizará una arquitectura de 4 capas (capa física, capa de red, capa de servicios, capa aplicación).

En la capa física estarán todos los dispositivos y componentes electrónicos necesarios para poder obtener los datos recopilados por los sensores implementados mismos que estarán conectados en una placa de desarrollo, además del sistema electrónico necesario para que las porciones de comida y agua se lo realice de manera automática de acuerdo a las porciones

requeridas de acuerdo al número de codornices en la jaula, en la capa de red tendremos los dispositivos de comunicación necesarios para que se realice la transmisión de los datos que se han recopilado anteriormente por los sensores mismos que través de un Gateway se transmitirán los datos para ser procesados y almacenados, posteriormente la capa de servicios comprende la parte del alojamiento de los datos recopilados estos mediante el uso de un servicio virtualizado en la nube donde se registrará el nivel del alimento y agua presente en el contenedor, por último tenemos la parte de capa aplicación donde el usuario recibirá una notificación por medio de una plataforma de mensajería de teléfono celular el cual notifique cuando sea necesario llenar de forma manual el contenedor cuando el alimento y el agua lleguen a un punto inferior del contenedor, esto con el fin de recargarlos nuevamente de forma manual.

III. Implementación

Una vez que se haya diseñado el sistema IoT procedemos a la implementación del sistema para una de las familias de la parroquia la Esperanza del cantón Ibarra de la provincia de Imbabura quienes actualmente se dedican a esta actividad para ello se elegirá un lugar estratégico en la jaula de las codornices, aquí se realizarán todas las conexiones de los componentes que conforman el sistema.

IV. Pruebas

Una vez que se ha realizado la implementación del sistema se procede con la realización de las pruebas que satisfagan con la validez del sistema. Para ello se realizará una planificación de pruebas mediante un análisis de la situación actual por parte del coturnicultor de una de las familias de la parroquia de la Esperanza que se dedican a esta actividad entre los datos que se recopilaran para ver la efectividad del sistema propuesto será el tipo de comida, la cantidad proporcionada y el número de puesta de huevos producidos de acuerdo a un cronograma con el fin de observar la cantidad de huevos que las codornices producen aplicando una

alimentación de la forma tradicional y aplicando la implementación del sistema, con estos datos obtenidos se podrá validar la mejora de producción de huevos de codorniz con esto se cumplirá con éxito el objetivo desarrollado, además como último punto se sugerirá al cornicultor que para un buen funcionamiento del sistema deberá dar el mantenimiento necesario como limpieza necesaria de los contenedores, la jaula y revisión de componentes electrónicos en caso de presentar fallos.

1.5 Justificación

De palabras del señor Luis Enrique Pupiales Brusil habitante de la parroquia de la Esperanza del cantón Ibarra de la provincia de Imbabura propietario de 30 codornices menciona que desde la adquisición de estos animales la alimentación se la ha realizado de forma tradicional mismo que ha sido un problema el suministrar el alimento adecuado así como la cantidad de veces que se deben realizar al día, esto debido a que se dedica a otras actividades primordiales, además menciona que el alimento proporcionado en la dieta de las codornices consta en su mayoría por granos como trigo, maíz molido y un poco de balanceado de la marca BIOMENTOS por ello tiene la necesidad de implementar un sistema que le ayude con el control de alimentación de las codornices que le facilite controlar la alimentación adecuada y dar las porciones necesarias de acuerdo al número de codornices que posee, además menciona que al ser una actividad secundaria espera que con la implementación de este sistema poder optimizar el tiempo, obtener huevos de una mejor calidad y que mejore la producción de huevos ya que los recolecta y vende en la tienda de su sector obteniendo ingresos económicos extras para su hogar. Como punto final menciona que en un futuro piensa adquirir más ejemplares de este tipo de aves.

La coturnicultura es una actividad productiva alternativa, especial para desarrollar en escala de pequeñas y medianas empresas, y de tipo familiar, misma que trata de una explotación accesible para un micro emprendimiento familiar, con una inversión inicial baja tanto en animales como en instalaciones. (Terán Caicedo, 2020)

Para producir huevos para consumo, las hembras pueden alojarse en grupos de 30 a 40 en cada piso de la batería (módulo), y esta debe tener el piso inclinado a su frente libre en la parte inferior, para permitir que los huevos salgan al exterior y caigan en el retén que tiene en el fondo de la jaula, donde serán recogidos con facilidad. (Revelo & Terán , 2012)

La recolección de los huevos se debe hacer dos veces al día en la mañana, y por la tarde, ya que los animales no ponen a la misma hora, además una vez recogidos, se deben eliminar los que presentan roturas o estén sucios y los demás almacenarlos en un sitio fresco hasta el momento de su venta, las hembras para postura no deben tenerse más de dos años, (lógicamente que en el segundo año la postura baja considerablemente) al cabo de este tiempo deberán ser eliminadas y vendidas para el consumo. (Revelo & Terán , 2012)

Este trabajo de integración curricular en base al objetivo 9 del marco de los Objetivos de Desarrollo Sostenible (ODS) promueve una industrialización inclusiva y sostenible, además busca aumentar significativamente la contribución de la industria al empleo y al producto interno bruto, de acuerdo con las circunstancias nacionales, y duplicar esa contribución en los países menos adelantados. (Naciones Unidas Ecuador, 2024)

También tomando como referencia el objetivo 11 se busca modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales en las familias que se dedican a la producción de huevos de codorniz en la parroquia La Esperanza. (Naciones Unidas Ecuador, 2024)

Según el (Ministerio de Agricultura y Ganadería, 2023) La producción de proteína animal (carne, huevos de mesa) tiene una relevancia económica, productiva y social para el país. Es parte de una cadena, donde se incluye la producción de maíz duro, elaboración de alimento balanceado y producción avícola.

Actualmente el uso del internet forma parte de la vida cotidiana de las personas y contar con sistemas de automatización que interactúen con el ser humano, en busca de mejorar el estilo de vida que hagan más fácil el día a día, aumentar el confort de las personas, facilitar tareas y procesos en diversas áreas no parece ser algo complicado. Haciendo uso de los avances e innovaciones tecnológicas puede lograrse, tecnologías como IoT nos permite todo lo anterior manteniendo la interconectividad entre dispositivos electrónicos. (Salinas Anaya, 2022)

Capítulo II: Marco Teórico

En primera instancia, se abordará el tema de la coturnicultura, sus orígenes y su distribución geográfica en el Ecuador. Se pondrá especial énfasis en la codorniz japonesa, tratando aspectos como la taxonomía, la alimentación y su importancia en la crianza de estas aves. También se explicarán las condiciones ambientales óptimas para su explotación doméstica. Además, se explorarán aspectos relacionados con el huevo de codorniz, abarcando su etapa de puesta, producción, recolección y el diseño de las jaulas óptimas para su crianza. Esto con el fin de obtener huevos de mejor calidad y una mayor producción.

Finalmente, se plasmarán conceptos básicos sobre el internet de las cosas y la arquitectura IoT, además se dará a conocer sobre la aplicación de sistemas IoT (Internet de las Cosas) en la avicultura, destacando el uso de tecnologías automatizadas y los beneficios que estas aportan a la gestión y optimización en la avicultura. También, se hablará sobre cómo la automatización puede incrementar la eficiencia, disminuir los costos y potenciar la productividad en el sector avícola, además de mostrar el tipo de componentes más utilizados para este tipo de sistemas de automatizados.

2.1 Coturnicultura

La coturnicultura es una práctica dentro de la avicultura que se centra en la cría y mejora de codornices mismo que tiene como objetivo principal aumentar y optimizar la producción de codornices para obtener diversos productos, entre ellos productos como los huevos, que son valorados por su alto contenido nutricional y sabor delicado, además de la carne, conocida por ser tierna y saludable.

De acuerdo a (Lindao Vera, 2023), en Ecuador, la coturnicultura se lleva a cabo en varias regiones del país. La producción y venta de huevos de codorniz han logrado establecerse en el mercado, llegando incluso a ser distribuidos en supermercados a nivel nacional.

Según (El Productor, 2017), la cría de codornices se presenta como una actividad económica prometedora, debido a la fácil adaptación a distintas regiones y espacios, y la alta capacidad de producción de huevos. Sin embargo, para que la coturnicultura alcance su máximo potencial, es indispensable que los productores dominen técnicas especializadas y posean conocimientos adecuados sobre: una correcta alimentación, dieta equilibrada, manejo eficiente de las instalaciones y correcta selección genética.

2.1.1 Origen de la codorniz japonesa y distribución geográfica en el Ecuador

La codorniz japonesa es originaria de Asia Oriental, China, Indonesia, Japón y fue domesticado por primera vez en Japón en el año de 1595, posteriormente fueron introducidas a Europa y América entre 1930 y 1950, durante el siglo XVII, la codorniz japonesa fue reconocido como animal de producción debido a su carne y huevos, mismos que se involucraron en la dieta oriental y la medicina tradicional, por ello desde entonces, la codorniz japonesa ha sido introducida exitosamente en numerosas regiones alrededor del mundo.

En Ecuador las condiciones climáticas son favorables para la actividad de la coturnicultura, la cría de codornices es una actividad relativamente reciente, que comenzó hace aproximadamente 25 años. Sin embargo, en la última década, esta práctica ha experimentado un notable crecimiento, consolidándose como una actividad comercial de alto rendimiento.

Según (Sagñay Sagñay , 2021) en el Ecuador las provincias con mayor crecimiento en la producción coturnícola son: Quito, Guayaquil, Santo Domingo, Cañar y Tungurahua.

Según (INEC, 2024) en la última encuesta agropecuaria realizada en el Ecuador existen 91.000 codornices en planteles avícolas, en cuanto a la región que mayor aporta con la producción de huevos en general es la Sierra.

2.1.2 Codorniz Japonesa

Según (Flores Rivera, 2019) menciona que la codorniz japonesa anida en el archipiélago de japón y la isla de Sakhaline, emigra a Indochina, a Siam y a Taiwán, además esta ave fue

introducida a Estados Unidos con fines decorativos e investigativos en el siglo XIX misma que con el paso del tiempo fue consolidándose en la industria avícola.

Esta ave tiene un tamaño promedio de 15 a 20 cm de longitud y un peso de 100 a 150 g.

La codorniz japonesa, “Es la especie más comercial para la producción de huevos y usada con fines de alimentación humana con la mejor conversión de alimento en huevos de un 85-95%” (Agrotendencia, 2020).

Como podemos observar en la Figura 2, la codorniz japonesa se diferencia entre machos y hembras, debido a que los machos tienen un color de cuello y barbilla más grande con un plumaje mucho más colorido con tonos marrones y rojizos en el pecho con coloración más intensa alrededor del pico y el canto es característico en este sexo, mientras que la hembra tiene las plumas más discreto y uniforme con tonos marrones y grises además de ser lanceoladas y manchadas de negro, por lo general son más grandes que los machos.

Figura 2

Codorniz Japonesa (macho y hembra)

Fuente: (El Sitio Avícola, 2014)

2.1.2.1 Clasificación taxonómica

Tabla 1

Clasificación de la taxonomía de la codorniz japonesa

Taxonomía	
Nombre científico	Coturnix coturnix japónica o coturnix japónica
Nombre común	Codorniz japonesa
Reino	Animal
Tipo	Vertebrado
Clase	Aves
Subclase	Carinados
Orden	Galliniforme
Familia	Phaisanidae
Subfamilia	Eurasiana
Género	Coturnix
Especies Similares	Coturnix chinensis, Coturnix delegorguei, Coturnix, coturnix

Nota. Información taxonómica de la codorniz japonesa, obtenida de (Oficina Estatal de Información para el Desarrollo Rural Sustentable , 2009)

En la **Tabla *1***, podemos observar la clasificación taxonómica de la codorniz japonesa, científicamente llamada coturnix coturnix japónica o coturnix japónica, se clasifica mediante un sistema taxonómico jerárquico.

Este sistema organiza la especie en diversas categorías, facilitando su identificación y estudio comparativo con otros organismos, destacando sus características distintivas y su lugar en la evolución de las aves.

2.1.2.2 Alimentación

Es importante para las codornices de todas las edades principalmente para las ponedoras, pues ellas necesitan de una buena alimentación, no solo para mantener su vida y compensar los desgastes orgánicos (ración de mantenimiento), y también para formación de los huevos que exige una alimentación abundante y rica en nutrientes que irá a componer cada una de las partes del huevo sean o no fértiles. (Cumpa Gavidia, CRIANZA Y MANEJO DE CODORNICES, 2009)

Según (Cumpa Gavidia, Nutrición y alimentación de las codornices japonesas (Parte III)., 2022), el consumo diario de alimento para una codorniz ponedora varía entre 22 y 25 gramos por ave, ya que estas aves tienen necesidades nutricionales esenciales que incluyen agua, carbohidratos, proteínas, grasas, vitaminas y minerales, lo cual es crucial para su desarrollo adecuado. A causa de su rápido crecimiento y su temprana capacidad para poner huevos, las codornices requieren una dieta con altos niveles de proteínas y energía. Además, de acuerdo a (Mendieta Suárez, 2015), se necesitan de manera constante agua limpia debido a que cada codorniz consume entre 40 a 60 ml de agua diariamente.

Es fundamental que los nutrientes se encuentren en las proporciones adecuadas para asegurar no solo su salud y bienestar, sino también su máxima productividad en la producción de huevos. Por lo tanto, la alimentación de las codornices debe ser cuidadosamente formulada y balanceada para satisfacer estas demandas nutricionales específicas, contribuyendo así a su desarrollo óptimo y a una producción eficiente de huevos, lo cual es esencial para los productores avícolas que buscan mantener altos estándares de calidad y rendimiento en sus granjas.

Según (Sagñay Sagñay , 2021), menciona que el éxito para un buen desarrollo y una buena producción es el alimento, debido a que un buen alimento ayuda a que el animal se desarrolle de mejor manera, es por ello que empresas como Nutril, Biolimentar y Exibal

ofrecen alimentos balanceados específicamente formulados para codornices en producción de huevos.

2.1.2.3 Condiciones ambientales

Según (Angelfire, 2001), la codorniz japonesa es una especie que se adapta bien a diversas condiciones ambientales, aunque para su explotación doméstica se obtienen mejores resultados en zonas con climas entre 18 y 30ºC y ambientes secos. Además, menciona que estas aves son muy sensibles al frío, por lo que no se recomienda su cría en lugares con temperaturas bajas por lo que es crucial que las jaulas estén en sitios protegidos y sin corrientes de aire, y que reciban algo de luz por la mañana temprano. Por otra parte, para climas cálidos, recomienda el uso de ventiladores eléctricos ubicados en la parte alta de las paredes para evitar corrientes de aire directas sobre las codornices. Además, el uso de cortinas puede ayudar a proporcionar un entorno óptimo para su desarrollo.

2.1.2.4 Huevos

Según (Sánchez Arias, 2023) ,los huevos de codorniz son más pequeños que los huevos de gallina y presentan cáscaras moteadas que pueden ser blancas o de tonos marrón moteado, su sabor es similar. En términos de valor nutricional, destacan por ser más calóricos y por su alto contenido en minerales esenciales.

- ***Características***

De acuerdo con (Tapia Garófalo , 2010), el huevo de codorniz japonesa, distinguido por su pequeño tamaño y forma ovoide, es adecuado a su origen, cuentan con un peso promedio de 10 gramos, estos huevos presentan un diámetro longitudinal de 3.14 cm y un diámetro transversal de 2.41 cm. Aunque poseen un cascarón relativamente resistente, son vulnerables a golpes y amontonamientos.

La cáscara, dura y blanca, está salpicada de manchas negras que pueden variar en color. La resistencia del huevo no se debe solo a la cáscara, sino también a la membrana interna que

la recubre y a la dieta de las ponedoras, que debe ser rica en calcio, fósforo y vitaminas. Además, los huevos de codorniz se destacan por su alto contenido proteico.

De acuerdo con (Imbaquingo Nazate , 2019), menciona que la calidad exterior de los huevos se evalúa a través de varios parámetros, entre los cuales se incluyen un índice de forma efectivo que varía entre el 80 y el 90%, un grosor de cáscara que oscila entre 0.20 y 0.22 mm, y una pigmentación uniforme que puede ir desde un color blanco cremoso hasta un marrón moteado. Respecto a la calidad interna de los huevos de codorniz, se consideran aspectos como la calidad de la albúmina, la coloración de la yema, y los porcentajes de cáscara, yema y albúmina. Además, algunos autores destacan la importancia de evaluar la composición nutricional del huevo.

- ***Formación del Huevo***

La formación del huevo de codorniz japonesa, es un proceso complejo que abarca desde la ovulación hasta la puesta, implica múltiples etapas en el tracto reproductivo de la hembra. Este proceso de formación inicia en el ovario donde maduran los folículos, cada uno alojando una yema que se libera durante la ovulación y es capturada por el infundíbulo del oviducto. Posteriormente, la yema avanza al magnum, donde se añaden las capas de albúmina, después el huevo avanza hacia el istmo, aquí es donde se forman las membranas internas y externas posteriormente en la glándula de la cáscara, se deposita el carbonato de calcio para la formación de la cáscara del huevo. Finalmente, el huevo completamente formado es expulsado a través de la cloaca.

La calidad del huevo, incluyendo la fortaleza de su cáscara, está significativamente influenciada por la dieta de la codorniz, la cual debe ser abundante en calcio, fósforo y vitaminas. Además, la integridad de la membrana interna del huevo también es fundamental para su resistencia.

De acuerdo con (Valle Muñoz, Bustamante Castro, Rodríguez, Guillet, & Vivas, 2015), menciona que para asegurar que los huevos de codorniz cumplan con los estándares de calidad, es crucial que todos los componentes que los conforman sean sintetizados de manera correcta y se dispongan en la secuencia, cantidad y orientación adecuadas. Este éxito en la formación del huevo depende fundamentalmente de que las codornices sean alimentadas con nutrientes de alta calidad y se mantengan en un entorno que les brinde confort ambiental y un óptimo estado sanitario. Estas condiciones permiten que el proceso biológico de formación del huevo se realice de manera eficiente, garantizando así la calidad y seguridad del producto final.

- **Irregularidades en la formación del huevo**

Según (Rodas Z., 2004), las irregularidades en la formación del huevo de codorniz pueden originarse por diversos factores que afectan tanto la calidad del huevo como la frecuencia de la puesta.

Dichos factores son de origen genético, defectos en el oviducto, falta de cantidad de calcio y proteína necesaria en el ave, estrés, alargamiento del ciclo de producción (codorniz vieja), enfermedades, condiciones de manejo inapropiadas, incluyendo la falta de espacio adecuado y malas prácticas de higiene, contribuyen a la aparición de irregularidades en la formación del huevo las cuales traen consigo consecuencias como huevos con cáscaras débiles, sin manchas, y pequeños.

2.1.2.5 Producción de huevos

La producción de huevos en codornices da inicio cuando llegan a la madurez sexual y empiezan a poner huevos, por ende, es primordial que en este periodo se proporcione a estas aves un ambiente adecuado con una dieta equilibrada con el fin de mantener una producción de huevos estable y contante.

- ***Etapa de producción***

En la codorniz japonesa, según (Otálora, 2017), la fase de producción de huevos se inicia a las seis semanas de vida y se extiende hasta las 30 semanas de edad o más, abarcando al menos 22 a 24 semanas de producción continua. Durante este período, una codorniz japonesa puede producir aproximadamente de 200 a 300 huevos, manteniendo un ciclo de postura regular con intervalos de 22 horas entre cada huevo. Este ciclo productivo tiene una duración de un año, mismo que se destaca por su eficiencia y regularidad.

De acuerdo a Durante el periodo de producción de huevos, las reproductoras son alojadas en grupos de 1 macho con 3 hembras con el objetivo de que los huevos producidos puedan ser destinados a incubación, caso contrario si el objetivo la producción comercial, el consumo o venta de huevos es recomendable tener 20 hembras sin macho en una jaula o 1000 hembras con 4 machos, pero no en el mismo sitio que las hembras sino en otras jaulas pero dentro del mismo galpón, debido a que con el canto estimula en la postura, además un huevo infértil es muy importante para una mejor conservación del huevo.

- ***Curva de postura***

La codorniz japonesa inicia su período de postura a una edad temprana, típicamente entre los 35 y 45 días de vida. Durante esta etapa inicial, los huevos producidos varían significativamente en tamaño y peso, oscilando entre 1 y 24 gramos. conforme pasa el tiempo y va creciendo la codorniz, su capacidad de producción de huevos aumenta progresivamente, alcanzando un incremento de hasta el 90%. Aproximadamente entre los dos meses y medio y los tres meses, la codorniz llega a su pico de postura, el cual es el nivel máximo de producción. En esta fase, la codorniz puede poner de uno a dos huevos por día, manteniendo este nivel de alta producción por un periodo de 4 a 6 semanas. La duración y estabilidad del pico de postura son indicadores clave de la productividad anual del ave. Un pico de postura elevado se traduce en una disminución gradual de la producción a lo largo del año, mientras que un pico bajo

resulta en una rápida disminución de la postura, finalizando el año con menos del 40% de producción y una mayor incidencia de cáscaras frágiles, lo que afecta negativamente la calidad de los huevos.

2.1.2.6 Recolección de huevos

Debido a que las aves no ponen huevos a la misma hora, es posible realizar hasta tres recolecciones diarias. Sin embargo, se recomienda efectuar solo dos recolecciones, una en la mañana y otra por la tarde. Los huevos deben ser colocados en bandejas con el extremo agudo hacia abajo, y deben ser clasificados siguiendo parámetros como color, forma, tamaño y peso.

Según (VILLACIS VIVAR & VIZHCO MINCHALA , 2016) , menciona que la limpieza de los huevos debe realizarse con sumo cuidado para evitar la eliminación de la cutícula protectora, la cual actúa como una barrera contra la contaminación. No se recomienda lavar los huevos con agua, ya que esto puede eliminar esta capa protectora y aumentar el riesgo de infecciones. En su lugar, se aconseja limpiarlos suavemente con un paño seco o ligeramente húmedo si es necesario, para mantener la integridad de la cutícula y asegurar la seguridad del huevo.

De acuerdo a (Zosimo, 2023) menciona que la vida útil de los huevos de codorniz depende principalmente de las condiciones de almacenamiento y de si se han lavado antes del almacenamiento, los huevos de codorniz que no han sido lavados y se mantienen refrigerados pueden conservarse en buen estado hasta por tres semanas. En contraste, los huevos que han sido lavados tienen una vida útil significativamente más corta, generalmente entre 7 y 10 días, debido a la eliminación de la cutícula protectora que los recubre.

Para monitorear la postura adecuadamente, es crucial calcular la cantidad diaria de huevos recolectados, que puede oscilar entre el 70% y el 90% de la producción total de las aves en postura. Este porcentaje puede fluctuar dependiendo de la edad de los animales involucrados.

2.1.2.7 Jaula para codornices japonesas

Según (Sembralia, 2021), la estructura destinada al alojamiento de codornices debe ser planificada de manera que asegure el bienestar de las aves, facilite el acceso a alimentos y agua, y permita una desinfección eficiente. Respecto a las jaulas, estas pueden ser instaladas en diversos entornos como patios, granjas, espacios reducidos o corrales donde llegue la luz ya que en la etapa de producción estas necesitan por lo menos 15 horas de luz por día.

(Sembralia, 2021) también recomienda que al momento de elegir una jaula se deben tener en cuenta características como poseer un techo de metal resistente para reducir el riesgo de lesiones en la cabeza de las codornices, proporcionar al menos 145 cm^2 de espacio en el suelo por ave para una mejor producción, el espacio en una jaula destinada a dos codornices, se recomienda que las dimensiones sean de 13 × 20 cm y por cada codorniz debe contar con al menos 0.6 cm de espacio en el bebedero y un espacio entre 1.25 y 2.5 cm por cada codorniz en el comedero, además de disposición de agua fresca y limpia con una buena iluminación en las jaulas.

2.2 Internet de las cosas (IoT)

El término IoT, o Internet de las cosas, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos, gracias a la llegada de los chips de ordenador de bajo coste y a las telecomunicaciones de gran ancho de banda, ahora tenemos miles de millones de dispositivos conectados a Internet, con esto dispositivos de uso diario, las aspiradoras, los coches y las máquinas, pueden utilizar sensores para recopilar datos y responder de forma inteligente a los usuarios. (Amazon Web Services, 2023)

Las posibilidades de aplicación del IoT son extensas y diversas, y su influencia se está haciendo sentir en numerosos sectores industriales como la manufactura, el transporte, la atención médica y la agricultura. A lo largo del tiempo, el IoT ha evolucionado y expandido su

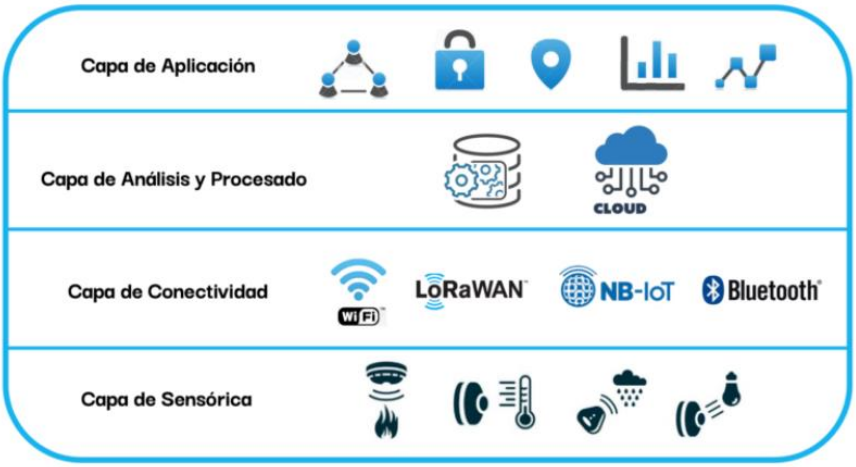
alcance apoyado en tecnologías fundamentales como sensores, protocolos de red, computación en la nube, aprendizaje automático e inteligencia artificial (IA).

2.2.1 Arquitectura IoT

Según (Unir, 2023), la arquitectura del Internet de las Cosas (IoT) se refiere a un sistema donde la información se desplaza a través de la red. Este proceso comienza con la recolección de datos mediante sensores distribuidos en diversos entornos, una vez que los datos son digitalizados son transmitidos a través de la red hacia un centro de datos o una plataforma en la nube. Una vez allí, la información es procesada y almacenada para su análisis y uso posterior.

Figura 3

Capas de arquitectura IoT



Fuente: (Istec, 2022)

Como se puede observar en la Figura 3. Esta arquitectura consta de 4 capas:

- **Capa de sensórica:** En esta capa están presentes los dispositivos capaces de medir magnitudes físicas y la presencia de sensores para ejecutar acciones e interactuar.
- **Capa de conectividad:** Esta capa establece canales entre estos dispositivos y plataformas, ya sea en infraestructuras concentradas o en la nube, mediante tecnologías inalámbricas como Wi-Fi, NB-IoT, Zigbee, Bluetooth y LPWAN.
- **Capa de análisis y procesado:** Analiza los datos recopilados por los sensores.

- **Capa de aplicación:** Las aplicaciones o interfaces de usuario que facilitan la gestión remota de dispositivos, integración, monitoreo, mantenimiento de la conectividad y seguridad a lo largo del ciclo de vida del sistema.

2.2.1.1 Capa Sensórica

De acuerdo a (Jahnke, 2020), esta capa es la primera, responsable de recolectar y convertir datos del entorno físico en información digital la cual comprende la parte de sensores, actuadores etc.

- ***Componentes***

  - **Sensores**

Los sensores son dispositivos diseñados para detectar y medir cambios en el entorno físico, como temperatura, humedad, presión, luz, movimiento, entre otros. Estos dispositivos convierten las magnitudes físicas que capturan en señales eléctricas que pueden ser procesadas y utilizadas por sistemas electrónicos, incluyendo sistemas IoT.

  - **Actuadores**

Según (Especificar, 2021), un actuador es un dispositivo que transforma la energía de una fuente, como aire, líquido o electricidad, en movimiento. Existen dos tipos principales de actuadores: lineales, que generan movimientos rectilíneos para empujar o tirar, y rotativos, que producen movimientos giratorios, como en válvulas de mariposa o de bola. Además, algunos actuadores pueden realizar movimientos oscilatorios, son utilizados en motores, interruptores válvulas y bombas. Cada tipo de actuador está disponible en diversas versiones adaptadas a diferentes tamaños, estilos y modos de operación específicos para cada aplicación.

Existen diferentes tipos de actuadores como: actuadores neumáticos, hidráulicos, eléctricos y térmicos.

- **Microcontroladores**

De acuerdo a (Dusun, 2023), en el contexto del Internet de las Cosas (IoT), los microcontroladores son dispositivos electrónicos compactos y eficientes en consumo energético. Estos funcionan como el núcleo operativo de los sistemas IoT, encargados de la gestión y control de dispositivos y sensores conectados. Su rol es esencial, permitiendo la recolección de datos del entorno, su procesamiento local y la conexión con otros dispositivos o sistemas IoT a través de la red, en la Tabla 2, podemos observar algunos microcontroladores con sus características más importantes con el fin de ver sus características y elegir el más óptimo de acuerdo a las necesidades del proyecto.

Tabla 2

Microcontroladores

Microcontrolador	Conectividad	Poder de procesamiento	Interfaces IO	Consumo de energía	Entorno de desarrollo	Costo
Arduino	Varía según tablero	Moderado	GPIO, ADC, UART, I2C, SPI, etc.	Variable	Arduino IDE, extensa biblioteca	Bajo a moderado
ESP32	Wi-Fi, Bluetooth, Zigbee, Celular	Moderado a alto	GPIO, ADC, UART,	Bajo a moderado	Arduino IDE, ESP-IDF,	Bajo a moderado

			I2C, SPI, etc. GPIO, USB, HDMI, I2C, SPI, UART			
Frambuesa Pi	WiFi, Ethernet, Bluetooth	Alta		Moderado a alto	Raspberry Pi OS, ecosistema extenso	Moderado

Fuente: (Dusun, 2023)

- **Fuente de energía**

De acuerdo a (Lima Pambaquishpe , 2023), menciona que este aspecto es crucial para el funcionamiento eléctrico del sensor. Se refiere a cómo se proporciona energía a los circuitos internos del sensor, ya sea mediante baterías, paneles solares u otras fuentes de alimentación eléctrica disponibles.

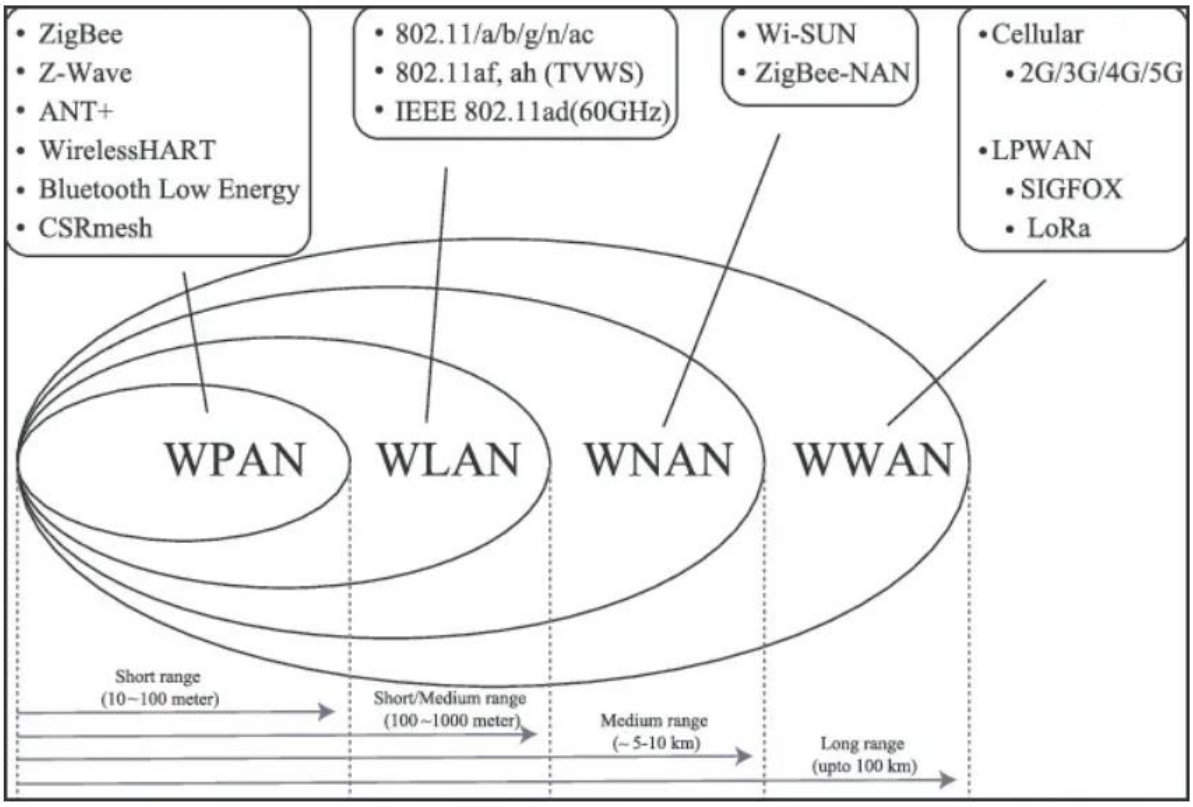
2.2.1.2 Capa de conectividad

La capa de conectividad en la arquitectura IoT se encarga de establecer los medios físicos y protocolos de comunicación necesarios para que los dispositivos IoT puedan intercambiar datos entre sí y con sistemas externos, como plataformas en la nube o aplicaciones de gestión.

- ***Tecnologías inalámbricas***

Figura 4

Tipos de tecnologías inalámbricas



Fuente: (mokolora, 2021)

Como podemos observar en la Figura 4. Tenemos las tecnologías inalámbricas las cuales abarcan una amplia gama de estándares y protocolos que permiten la comunicación sin necesidad de cables físicos, cada una adaptada a diferentes alcances y aplicaciones específicas:

- WPAN: Permite la comunicación entre dispositivos dentro de un área personal cercana, típicamente unos pocos metros.
- WLAN: Proporciona conectividad inalámbrica a redes locales y permite el acceso a internet en áreas limitadas como hogares, oficinas, campus universitarios y espacios públicos.

- WMAN: Diseñada para abarcar áreas metropolitanas, WMAN permite la conectividad inalámbrica a través de distancias más grandes que WLAN.
- WWAN: Facilita la conectividad inalámbrica a través de áreas extensas, a menudo utilizando redes celulares como 3G, 4G LTE y 5G. Estas tecnologías permiten la comunicación móvil de alta velocidad y son fundamentales para dispositivos como teléfonos inteligentes, tablets y dispositivos IoT que requieren acceso a internet desde ubicaciones remotas o en movimiento.

De acuerdo con (Lima Pambaquishpe , 2023), cuando se desarrolla un proyecto o sistema IoT, es fundamental entender que cada diseño tiene requisitos de red específicos. Aspectos como el alcance, la calidad del servicio, la seguridad, el consumo de energía y la gestión de la red son cruciales al momento de elegir la tecnología más adecuada para cada situación, además en la Tabla 3, se presenta la tecnología más usada de acuerdo al área de aplicación.

Tabla 3

Tecnologías de acuerdo al área de aplicación

Área de aplicación	LPWAN	ZigBee	Wi-Fi	RFID
IoT Industrial	✓			
Edificios inteligentes	✓			
Hogar inteligente		✓	✓	
Avicultura/Agricultura	✓		✓	✓

Fuente: (Lima Pambaquishpe , 2023)

2.2.1.3 Capa de servicios

La capa de servicios en la arquitectura IoT proporciona la infraestructura necesaria para gestionar eficientemente los dispositivos, los datos y las aplicaciones IoT, asegurando una operación robusta, segura y escalable del sistema completo.

- ***Plataformas de Computación en la nube***

Según (Grapsas, 2018), las plataformas de computación en la nube son estructuras que proporcionan acceso a recursos como capacidad de procesamiento, almacenamiento, redes y servicios variados a través de internet. Esto permite a organizaciones y usuarios utilizar estos recursos de forma flexible, escalable y según demanda, sin requerir inversión en infraestructura física propia. Entre los principales proveedores de estas plataformas se encuentran Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud y Oracle Cloud, entre otros destacados en el mercado.

2.2.1.4 Capa Aplicación

La capa de aplicación constituye el nivel más alto del sistema, donde se encuentran las aplicaciones específicas que interactúan directamente con los usuarios finales o con otros sistemas externos. Esta capa se encarga de procesar los datos recogidos por los dispositivos IoT a través de las capas inferiores, como la de conectividad y la de servicios, transformándolos en información útil y funcionalidades concretas.

- **Protocolos de comunicación**

Según (GEEKFLARE, 2024) los protocolos de comunicación posibilitan la transferencia bidireccional de datos entre dispositivos, actuando como mediadores que aseguran la interoperabilidad universal entre dispositivos de diferentes especificaciones, siendo esencial porque establecen la conexión esencial entre los dispositivos IoT y los sistemas encargados de manejar y analizar los datos que producen.

- MQTT: Se trata de una implementación simplificada y liviana de un protocolo de comunicación diseñado para redes con ancho de banda limitado y alta latencia.
- CoAP: Es un protocolo de transferencia especializado diseñado para operar en nodos y redes de baja potencia con altos niveles de pérdidas.
- XMPP: Este protocolo de mensajería, basado en XML, ha evolucionado para incluir extensiones a HTML, facilitando la estructuración de datos y la escalabilidad en redes IoT.

- ***Plataformas de mensajería***

Según (Zendesk, 2024) , las aplicaciones de mensajería instantánea permiten que empresas y personas se comuniquen de manera rápida y efectiva usando mensajes de texto, voz y video. Son muy prácticas para brindar soporte al cliente, responder preguntas comunes y solucionar problemas de manera rápida y personalizada.

Las plataformas de mensajería como WhatsApp y Telegram son aplicaciones diseñadas principalmente para la comunicación entre usuarios a través de dispositivos móviles y computadoras.

Ambas plataformas son ampliamente utilizadas a nivel mundial debido a su facilidad de uso, disponibilidad multiplataforma y funcionalidades avanzadas que incluyen la creación de grupos, canales de difusión, bots automatizados y soporte para desarrolladores de terceros mediante APIs abiertas.

2.3 Control manual y automático de alimentación enfocados a la avicultura

Según (González Montaño , 2023), menciona que, en la actualidad, los sistemas manuales y en cadena han disminuido su uso comparado con épocas anteriores debido a que los comederos manuales generan estrés en los pollitos debido a la interacción humana, mientras que los comederos de cadena, aunque automáticos, son susceptibles a averías y pueden

desperdiciar entre 0.5 y 1.0 gramos de alimento por ave diariamente. Por esta razón, la adopción de sistemas automáticos en comederos y bebederos presenta diversas ventajas en la avicultura ya que estos dispositivos, optimizan el proceso en las granjas avícolas, perfilándose como una solución óptima en este sector al mejorar la eficiencia económica, promover el bienestar animal y reducir el impacto ambiental, Además en su trabajo concluye que los comederos y bebederos automáticos son soluciones óptimas para disminuir los costos de producción y favorecer el rendimiento de los animales.

2.3.1 Beneficios de un sistema automático

La implementación de sistemas automatizados tiene varios beneficios y ventajas como estimular el consumo con fácil acceso al alimento, se elimina el desperdicio y se mejora la tasa de conversión. Además, se asegura un alto nivel de bioseguridad y bienestar para las aves. La activación del alimentador mediante sensores elimina la necesidad de control manual, reduciendo los costos laborales y aumentando la productividad. Esto, a su vez, mejora la eficiencia en la gestión del agua, incrementando la rentabilidad en la producción avícola. Además, "los sistemas automáticos de alimentación y de agua son una buena alternativa para aumentar los niveles de producción en el sector avícola, reduciendo costos en diferentes áreas" (BM Editores , 2022).

2.4 Uso de IoT en la automatización para el sector avícola

Según (García Vaca & Mora Cruz, 2021), las tecnologías del Internet de las Cosas (IoT) aplicadas a la avicultura ofrecerán a los pequeños avicultores la oportunidad de implementar sistemas avanzados de monitoreo y gestión, optimizando así la eficiencia de la producción y mejorando las condiciones de crianza de las aves.

De acuerdo con (Echavarria & Acero Castillo, 2023), el Internet de las Cosas (IoT) convierte una avícola normal en una moderna avícola e inteligente, con un gran impacto en la producción, Además, menciona que la incorporación del internet de las cosas (IoT) puede

proporcionar soluciones innovadoras las cuales pueden ayudar a obtener eficiencia de la productividad en las pequeñas granjas avícolas para ello es necesario, contar con el apoyo técnico necesario hacia el sector avícola en la implementación de estas nuevas tecnologías.

2.4.1 IoT aplicado a la ventilación de gallineros para la cría inteligente

Según (Generación IoT, 2022), la aplicación del Internet de las Cosas (IoT) en la ventilación de gallineros para la cría inteligente representa un avance significativo en la gestión eficiente de ambientes agrícolas ya que, mediante el uso de sensores de temperatura, humedad y calidad del aire, junto con actuadores que regulan ventiladores y compuertas, este enfoque optimiza el ambiente para mejorar el bienestar de las aves y aumentar la eficiencia productiva.

Además, (Generación IoT, 2022), menciona que los datos recopilados son enviados a plataformas en la nube, donde se analizan en tiempo real para ajustar los parámetros de ventilación según las necesidades específicas, con ello no solo mejora la salud y el rendimiento de las aves, sino que también reduce los costos operativos al optimizar el uso de recursos como la energía y el agua.

2.4.2 IoT aplicado al seguimiento de activos en tiempo real mediante la tecnología GPS

De acuerdo con (Generación IoT, 2022) , el seguimiento por GPS, conocido por su capacidad para controlar equipos en la agricultura inteligente, también es eficaz en la supervisión de la producción y el ciclo de vida de las aves de corral. Esta tecnología permite a los granjeros monitorear todos los aspectos del ciclo de vida de las aves, desde la cría hasta el procesamiento, asegurando una experiencia auténtica de la granja a la mesa. Los datos recopilados optimizan la producción avícola, ayudan a identificar causas de brotes virales y localizan contaminaciones específicas en los corrales de alimentación, contribuyendo así a mantener la seguridad de la manada.

2.4.3 IoT aplicado al monitoreo de la temperatura y la humedad relativa

Según (Generación IoT, 2022), la temperatura y la humedad del entorno de cría son factores cruciales que afectan la productividad de las aves, ya que las condiciones de alta temperatura y humedad pueden inducir estrés en las aves, perjudicando su crecimiento y capacidad para producir huevos, también un ambiente excesivamente cálido puede resultar en gallinas de menor tamaño y en huevos con cáscaras más delgadas, entre otros problemas potenciales.

Además, (Generación IoT, 2022), menciona que aquí es donde los sensores ambientales controlados por inteligencia artificial (IA) pueden ser de gran ayuda, mitigando riesgos al comparar datos ambientales y de rendimiento. Estas soluciones son efectivas para regular diversos factores ambientales dentro de un recinto de cría, automatizando tareas de mantenimiento rutinarias y repetitivas necesarias en la avicultura.

2.4.4 Componentes más utilizados en IoT enfocados a la avicultura

En la avicultura, la implementación de IoT ha revolucionado la manera en que se gestionan y optimizan las granjas avícolas.

En cuanto a componentes mayormente utilizados para este tipo de sistemas son: sensores, actuadores, sistemas de ventilación automatizados, dispositivos de seguimiento por GPS, drones, microcontroladores como ESP32 y arduino, plataformas en la nube como Amazon Web Services, Azure, Google cloud.

2.4.5 Beneficios

Según el trabajo realizado por (Pérez Clemente, 2023), menciona que, en el monitoreo inteligente diseñado en base a un tipo de crianza intensiva enfocado a la recopilación de datos en tiempo real, y en altos volúmenes, requiere de ciertos puntos para su rendimiento óptimo.

Además, menciona que este tipo de sistemas tienen potencial a futuro debido a que estos sistemas pueden convertir una granja tradicional en una granja inteligente a un menor costo, también, producir mayores ingresos en base a la eficiencia económica debido al rendimiento productivo.

2.4.6 Desafíos

Según (Pérez Clemente, 2023), algunos de los desafíos son:

- Acceso a la tecnología.
- Inversión elevada para pequeños productores.
- Garantizar una buena comunicación continua durante él envió de datos a la nube, y recepción de datos en tiempo real.
- Requerimiento de una fuente de alimentación eléctrica de 110V o 220V en corriente continua para el funcionamiento de los dispositivos.

Capítulo III: Diseño del Sistema

Para el desarrollo de este capítulo se empleará la metodología en cascada, la cual permite abordar de manera secuencial cada una de las fases del proyecto. En primer lugar, se analizará la situación actual del criadero de codornices, poniendo especial énfasis en el tipo de alimentación que se suministra en la actualidad y la cantidad promedio de huevos recolectados diariamente.

Posteriormente, se llevará a cabo una fase de experimentación con diferentes tipos de balanceados disponibles en el mercado destinados a la etapa de postura, así como con alimentos tradicionales comúnmente utilizados por el propietario, se dará a las codornices las porciones óptimas durante la etapa de puesta. El objetivo de esta experimentación es determinar cuál de estas alternativas alimenticias ofrece mejores resultados en cuanto a productividad.

Además, con base en los datos obtenidos, se elaborará una recomendación fundamentada para el coturnicultor, indicando cuál es la opción de alimentación más eficiente y que contribuye de manera significativa a mejorar la producción de huevos mismo que se usará en el sistema de control de alimentación.

Finalmente, se detallarán los requerimientos del sistema propuesto, así como su diseño e implementación, con el fin de desarrollar una solución tecnológica integral que permita automatizar y optimizar el control de la alimentación en el criadero.

3.1 Análisis de la Situación Actual

El presente análisis tiene como finalidad evaluar el estado actual del criadero de codornices perteneciente al Sr. Luis Enrique Pupiales Brusil, un pequeño productor local que lleva desarrollando esta actividad de forma artesanal desde hace aproximadamente tres años.

Los datos obtenidos para este apartado fueron gracias a una entrevista dirigida al al coturnicultor como se puede evidenciar en el **Anexo A**.

Su labor se enfoca principalmente en la producción y recolección de huevos de codorniz, los cuales comercializa dentro de su comunidad y en sectores cercanos. El criadero se encuentra ubicado en la comunidad de San Clemente, parroquia La Esperanza, cantón Ibarra, provincia de Imbabura como se muestra en la **Figura 5** . En este contexto, el diseño del sistema propuesto busca dar respuesta a las necesidades identificadas en el manejo actual del criadero, brindando una solución que permita optimizar los procesos de alimentación y monitoreo de la producción de huevos.

Figura 5

Ubicación del sitio del criadero de codornices en el mapa geográfico

Fuente: Autoría propia

3.1.1 Ubicación

De acuerdo a (Teca Méndez, 2017), La comunidad de San Clemente está localizada en la parroquia La Esperanza perteneciente al cantón Ibarra en la provincia de Imbabura y se ubica aproximadamente a 7 kilómetros al sur de la ciudad de Ibarra , además menciona que cuenta con una altitud que varía entre los 2600 metros sobre el nivel del mar en las zonas bajas y los 2800 metros en las áreas más elevadas asentándose a los pies del volcán Imbabura y presentando una topografía irregular característica de los paisajes andinos del Ecuador

3.1.2 Clima

La comunidad de San Clemente en sí no cuenta con registros meteorológicos propios sin embargo al ser parte del Cantón Ibarra podemos caracterizar su clima por medio de los registros anuarios meteorológicos históricos realizados en el plan de desarrollo y ordenamiento territorial 2023-2027 del cantón Ibarra (Gobierno Autónomo Descentralizado Municipal San Miguel de Ibarra, 2025), menciona que el cantón Ibarra se caracteriza por una gran diversidad de microclimas que van desde el frío andino hasta el clima cálido húmedo, por lo que la temperatura media del cantón es de 15.90° C, oscilando entre los 20 y 25° C en su máxima temperatura, mientras que la mínima temperatura oscila entre los 7 y 11°C.

3.1.3 Infraestructura

Con el propósito de identificar las condiciones actuales en las que se desarrolla esta actividad y establecer un punto de partida para el diseño del sistema se realiza un diagnóstico del criadero actual donde se pudo constatar que el criadero cuenta con 1 jaula de dimensiones de 1.92 cm de largo x 0.71 cm ancho y 0.50 de alto, como se puede observar las dimensiones no cumplen con las características optimas recomendadas para la producción eficiente de huevos para el número de ejemplares , debido a que el espacio disponible resulta ineficiente para garantizar condiciones adecuadas para el bienestar de estas aves además de evidenciar que en su mayoría está compuesta de tablas de madera y mallas, además no existe un sistema que permita garantizar la higiene tanto de los animales como de los huevos por lo que podemos mencionar que estas deficiencias estructurales influyen de manera directa en el rendimiento productivo del criadero, las condiciones descritas pueden observarse en la **Figura 6**, donde se aprecia las condiciones actuales del criadero.

Figura 6

Situación actual de la Jaula de criadero de codornices

Fuente: Autoría propia

3.1.4 Alimentación

En relación con la alimentación de las codornices, el propietario menciona que las actividades asociadas al manejo de las aves, consta de suministro del alimento y provisión de agua, mencionando que esto se realiza de manera completamente manual y sin un horario definido lo que genera variaciones en la rutina de alimentación.

Indica también que el alimento utilizado corresponde a un balanceado comercial de la marca Biomentos, el cual es mezclado con maíz o morochillo triturado con el objetivo de reducir los costos de alimentación. Dicha mezcla es suministrada una o dos veces al día dependiendo de la disponibilidad del tiempo del propietario, quien además desempeña labores en otro lugar, lo que limita la regularidad en el suministro.

En cuanto al suministro de agua, mencionó que el recambio se lo realiza cada tres días; sin embargo, durante ese periodo, el agua tiende a ensuciarse, adquiriendo una coloración amarilla, situación que puede afectar la calidad del recurso hídrico, y en consecuencia la salud y el desempeño productivo de las aves.

Respecto a los recipientes utilizados para suministrar alimento y agua, se observó el uso de botellas plásticas reutilizadas como comederos y bebederos. Si bien esta práctica representa una alternativa de bajo costo, dificulta el control de consumo, la dosificación adecuada del alimento, además de limitar la posibilidad de monitoreo continuo del proceso tal como se evidencia en la **Figura 7**.

Las condiciones descritas reflejan la ausencia de un sistema estructurado de manejo, lo cual incide directamente en la eficiencia productiva del criadero y pone en evidencia la necesidad de implementar soluciones técnicas que optimicen los procesos de alimentación y suministro de agua.

Figura 7

Estado actual de los comederos y bebederos en el criadero.

Fuente: Autoría propia

3.1.5 Limpieza de Jaulas

En este apartado el propietario menciona que la limpieza de jaulas se lo realiza como parte de las actividades dentro del manejo del criadero, mencionando que la limpieza es de manera irregular y sin un horario definido, por lo que, durante este tiempo entre limpiezas se evidencia una acumulación progresiva de residuos orgánicos, tales como excrementos, restos de alimento y plumas, influyendo de manera negativa el área donde se alojan las aves como se puede apreciar en la **Figura 8**.

Además, esta irregularidad con lo que respecta a la limpieza tiene un impacto directo en la higiene, sanidad y calidad de ambiente productivo debido a que la presencia de estos desechos los huevos salen sucios en su gran mayoría además de que el hecho de que la base de la jaula este construida con tablas de madera de forma irregular hace difícil la correcta limpieza de estos desechos ya que no se puede limpiar de manera óptima la jaula,

Por otro lado, el propietario menciona desde una perspectiva positiva que estos desechos recolectados de la jaula son aprovechados como abono y le resulta muy beneficioso para su terrero agrícola, mencionando que esta práctica no solo contribuye al aprovechamiento de residuos, sino que además representa un ingreso adicional, dado que otra de las actividades a las que se dedica el propietario es la agricultura.

Figura 8

Limpieza de desechos de la jaula de codornices de manera tradicional

Fuente: Autoría propia

3.1.6 Producción de huevos

En cuanto a la producción de huevos el propietario menciona que la recolección de huevos se realiza manualmente por lo general en horas de la noche unas 4 o 5 veces por semana por lo que comenta que algunos de los huevos recolectados están rotos o sucios por los desechos producidos por las aves como podemos observar en la **Figura 9** , debido a esto, para poder venderlos el propietario tiene que escoger los que estén en buen estado por lo que menciona que demanda tiempo para realizar este proceso.

Figura 9

Huevos producidos de forma tradicional

Fuente: Autoría propia

3.2 Evaluación comparativa de dietas tradicionales para codornices en postura

Con el objetivo de identificar el alimento balanceado o combinación que mejor se ajusta a las necesidades productivas del criadero de codornices, se realizará un análisis comparativo entre dos marcas comerciales ampliamente utilizadas en establecimientos agropecuarios de la ciudad de Ibarra: Exibal y Biomentos. Además, se evaluará un tercer grupo experimental alimentado con una combinación de Exibal y morochillo triturado, con el fin de observar posibles variaciones al modificar el contenido nutricional.

3.2.1 Balanceado de postura

En la ciudad de Ibarra mediante observación directa en establecimientos comerciales agropecuarios se constató que actualmente se comercializan dos tipos de balanceados destinados a codornices en etapa de postura, correspondientes a las marcas Biomentos con un valor de 0.45ctvs la libra y Exibal cuyo valor es de 0.40ctvs la libra, cada uno con diferentes concentraciones de proteínas y nutrientes. Estos balanceados de postura están formulados específicamente para cubrir los requerimientos nutricionales de las aves ponedoras, favoreciendo una producción óptima de huevos en cuanto a cantidad, calidad y frecuencia.

3.2.2 Caracterización nutricional del balanceado comercial y del morochillo en la alimentación de codornices en postura

La caracterización nutricional del balanceado comercial y del morochillo es esencial para entender su influencia en la alimentación de codornices ya que permite identificar si estos insumos aportan niveles adecuados de proteínas, grasas, fibras, vitaminas y minerales necesarios para cubrir los requerimientos nutricionales de las aves un adecuado equilibrio de estos componentes favorece el mantenimiento de la salud, el metabolismo y la producción eficiente de huevos, tanto en cantidad como en calidad.

3.2.2.1 Biomentos

De acuerdo a la ficha técnica comercial obtenida de un repositorio digital, Biomentos aves la cual es una marca de balanceados perteneciente a la empresa Bioalimentar, especializada en la producción de balanceados de alta calidad, destinados a la nutrición de distintos tipos de animales, como aves, cerdos y bovinos.

Los cuales están formulados con el fin de aportar nutrientes esenciales para distintas etapas productivas, con ello favorecer al crecimiento, salud y productividad (Biomentos Aves, 2025).

Adicional a esto, la empresa promueve prácticas orientadas a la calidad y sostenibilidad de sus productos, como podemos observar en la **Figura 10**, tenemos la representación comercial del bulto de 40Kg del Balanceado Biomentos disponible en el mercado actual de la ciudad de Ibarra.

Figura 10

Presentación comercial de balanceado Biomentos

Nota. Imagen tomada de un establecimiento comercial agropecuario. Autoría propia

- ***Composición nutricional***

En la **Tabla *4***, podemos observar según la ficha técnica de biomentos aves composición nutricional del balanceado Biomentos en cuanto a su composición nutricional que cuenta con un 22% (mín) de proteína cruda, esencial para el crecimiento y la producción de huevos, un 5%(mín) de grasa, lo que proporciona la energía necesaria para las aves, un 4% (máx) de fibra cruda esencial para el equilibrio de la microbiota intestinal y una buena digestión , un 12%(máx) de humedad mismos que son óptimos para una conservación, estabilidad y calidad del producto, un 10%(máx) de cenizas los cuales son necesarios para una correcta formación de la cáscara del huevo, Estos parámetros hacen que el balanceado Biomentos Postura sea una opción ideal para maximizar la productividad y salud de las aves.

Tabla 4

Composición nutricional del balanceado Biomentos para codornices

Parámetro	Contenido%
Fibra cruda	4 (MAX)
Proteína cruda	22 (MIN)
Humedad	12 (MAX)
Cenizas	10 (MAX)
Grasas	5 (MIN)

Nota. Información correspondiente a la composición nutricional del balanceado para codornices en etapa de puesta presente en la ficha técnica de balanceado biomentos aves, obtenida de (Biomentos Aves, 2025)

3.2.2.2 Exibal

De acuerdo al sitio oficial de Exibal, es una empresa especializada en el ámbito nutricional cuya actividad principal se centra en la producción, procesamiento y comercialización de alimentos de alta calidad, su propósito es contribuir a mejorar la nutrición humana y animal, además es una empresa especializada en la producción de balanceados de alta calidad, diseñados para la nutrición óptima de aves, entre ellos codornices, garantizando una alimentación equilibrada que favorece el crecimiento, la salud y la productividad de las aves (Exibal), como se puede apreciar en la **Figura *11***, tenemos la representación comercial de un bulto de 40 Kg del Balanceado Exibal disponible en el mercado actual de la ciudad de Ibarra.

Figura 11

Presentación comercial del balanceado exibal

Nota. Imagen tomada de un establecimiento comercial agropecuario. Autoría propia

- ***Composición nutricional***

En la **Tabla *5***, podemos observar que el balanceado Exibal está formulado con una composición nutricional diseñada específicamente para maximizar la productividad de las codornices en etapa de postura. Contiene 22% (mín) de proteína cruda, que favorece el desarrollo muscular y la producción de huevos, en cuanto a fibra cruda este contine 7% (máx) el cuál es esencial para el equilibrio de la microbiota intestinal y una buena digestión, seguido de una humedad de 13% (máx) mismos que son óptimos para una conservación, estabilidad y calidad del producto, y grasas de 7% (mín) para una buena energía de las aves.

Tabla 5

Composición nutricional del balanceado Exibal para codornices

Parámetro	Contenido%
Fibra cruda	7 (MAX)
Proteína cruda	22 (MIN)
Humedad	13 (MAX)
Grasas	7 (MIN)

Nota. Información correspondiente a la composición nutricional del balanceado para codornices presente en la página oficial de Exibal, obtenida de (Exibal)

3.2.2.3 Morochillo

De acuerdo a el Instituto Nacional de Investigaciones Agropecuarias menciona que el morochillo también conocido como maíz duro amarillo es uno de los productos agrícolas más relevantes de la economía ecuatoriana debido a que es una materia fundamental para la elaboración de balanceados para el sector pecuario, además desde el punto de vista nutricional se lo emplea como fuente de energía debido al alto valor de contenido de almidón que posee (Instituto Nacional de Investigaciones Agropecuarias).

Como podemos observar en la **Figura *12***, podemos ver la representación del quintal de morochillo se comercializa en los abastos de la ciudad de Ibarra.

Figura 12

Representación del quintal de morochillo

Nota. Imagen tomada en uno de los abastos de la ciudad de Ibarra, Autoría propia

- ***Composición nutricional***

En cuanto a la composición nutricional del morochillo podemos observar en la **Tabla 6**, que en un 60-65% este grano se compone de almidón mismo que confirma una fuente de energía predominante, en cuanto al contenido de proteína tiene un porcentaje de 6.5-8-5% siendo un nivel bajo como fuente de proteínas es por esto que por lo general se lo combina con balanceados procesados con el fin de cumplir con las necesidades nutricionales de las aves debido a que la proteína es fundamental en cuanto a la formación del huevo, por otro parámetro tenemos la humedad la cual representa un valor de 14%(MAX), de igual modo el contenido de grasas oscila entre 3 -4% mientras que el valor de cenizas es de 1.5% (MAX).

Tabla 6

Composición nutricional del morochillo

Parámetro	Contenido%
Almidón	60-65%
Proteína	6.5-8.5%
Humedad	14.5% (MAX)
Grasas	3-4%
Cenizas	1.5% (MAX)

Nota. Información correspondiente a la composición nutricional del morochillo, obtenida de *(italcol)*

3.2.3 Prueba de alimentación de forma tradicional con distintos tipos de alimentos (balanceados y morochillo triturado).

En esta sección con el objetivo de determinar el tipo de alimento que permita una mayor productividad en cuanto a la producción de huevos, se pondrán a prueba tres tipos de alimentación mismos que serán suministrados de manera tradicional, Estos alimentos corresponden en primer lugar al balanceado de la marca Biomentos, como segundo lugar al balanceado de la marca Exibal, y como tercer lugar el alimento que el propietario de las codornices les suministra de manera tradicional la cual consiste en la mezcla de morochillo triturado y balanceado de la marca Biomentos , los cuales están presentes en el mercado actual.

Una vez realizadas las pruebas se podrá sustentar y dar una recomendación adecuada sobre el alimento más eficiente al propietario de estas aves, con ello se dará paso a la automatización con un sistema de alimentación óptimo.

Antes de iniciar con la prueba experimental, es importante mencionar que al momento de iniciar con estas pruebas experimentales el propietario cuenta con 30 codornices las cuales

fueron adquiridas tienen una edad de meses por lo que las aves ya se encontraban en etapa de postura.

Una vez identificada la situación actual del criadero se da inicio con la prueba experimental, para lo cual se establecieron dos periodos continuos, el primer periodo con una duración de 14 días, este tiempo fue considerado previo a la fase de análisis productivo ya que en el trabajo realizado por (Ludke, y otros, 2018) evaluaron estudios experimentales en codornices en cuanto a dietas en ciclos de 21 días , es por ello que se ha escogido este tiempo destinado a la adaptación de las codornices a su nuevo espacio, además de la adaptación al nuevo régimen alimenticio ya que se da paso al suministro de alimento y agua de acuerdo a cada grupo experimental, con este periodo de adaptación evitaremos posibles efectos del estrés lo cual podría derivar en alteración fisiológica y cambios de comportamiento provocando sesgos en los resultados por reducción en la producción de huevos, Posteriormente, se dará paso con el segundo periodo el cual consta de 28 días tiempo considerado adecuado para observar variaciones en la producción de huevos asociadas al tipo de alimento suministrado.

Durante este periodo se da paso al análisis productivo donde se realizará el seguimiento de la producción, lo que permitirá analizar de manera más precisa el impacto de cada alternativa alimenticia, por último, se realizará la suma total del total de huevos producidos durante todo el periodo de pruebas, incluyendo la primera etapa.

Cada grupo recibirá un tipo de alimento específico de acuerdo a la cantidad de comida por codorniz que debe recibir en la etapa de postura, 25gramos sería los óptimos por cada codorniz, con este dato se procede a suministrar 250 gramos por grupo mismos que fueron racionadas en 85gramos con el fin de proporcionar alimento 3 veces al día como se puede observar en la **Figura *13***, las raciones se las obtuvo con ayuda de una balanza.

Finalmente, durante todo el periodo experimental se llevará a cabo un registro diario de la producción de huevos, con ello contabilizaremos la cantidad de huevos obtenidos por cada

grupo, lo que permitirá analizar el comportamiento productivo de las codornices en función del alimento consumido.

Figura 13

Racionamiento de 3 tipos distintos de alimento en porciones de 85 gramos

Figura 1a Figura 1b Figura 1C

Nota. En la **Figura 1a**, podemos observar el racionamiento de 85 gramos del alimento tradicional en cual consiste en la mezcla de balanceado biomentos con morochillo triturado, en la **Figura 1b**, podemos ver el racionamiento de 85 gramos del balanceado biomentos, mientras que en la **Figura 1c,** podemos ver el racionamiento de 85 gramos del balanceado exibal, Autoría propia

3.2.3.1 Registro semanal de la producción de huevos

Como se mencionó anteriormente, en este apartado se llevará a cabo el registro semanal de la producción de huevos con el fin de llevar un control sistemático de acuerdo al desempeño productivo de las aves durante el periodo experimental, este registro también nos permitirá obtener información ordenada y continua sobre la cantidad de huevos producidos lo cual nos facilitará a tomar una decisión a la hora de dar una recomendación al propietario de las aves

sobre el mejor alimento de acuerdo a las variaciones asociadas debido al tipo de alimento suministrado a cada grupo, como podemos apreciar en la , además mencionar que los nombres del tipo de alimento se basarán en un acrónimo los cuales podemos observar en la **Tabla 7**,y por último se procederá con un análisis general de los datos obtenidos.

Tabla 7

Acrónimo de tipo de alimentos

Tipo de alimento	Abreviatura
Biomentos	B
Exibal	E
Morochillo+Biomentos	MB

Fuente: Autoría propia

- ***Semana 1***

Como se puede observar en la **Tabla 8** tenemos la primera semana de pruebas la cual consta desde el día 17 de junio del 2025 hasta el 23 de junio del 2025, está primera semana de pruebas se encuentra en el primer periodo ya que es un periodo donde las aves pasan a un nuevo espacio y a una nueva dieta alimenticia, lo que permitirá su adaptación y evitar el estrés en las codornices.

Tabla 8

Registro semanal de prueba experimental semana 1

Fecha	Día N°	Grupo N°	Tipo de Alimento	8:00(am) (g)	12:00(pm) (g)	16:00(pm) (g)	N° Huevos
		1	*B*	*85*	*85*	*85*	*3*
17/06/2025	*1*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*2*
		1	*B*	*85*	*85*	*85*	*3*
18/06/2025	*2*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*2*
		1	*B*	*85*	*85*	*85*	*2*
19/06/2025	*3*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*2*
20/06/2025	*4*	*2*	*E*	*85*	*85*	*85*	*2*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*3*
21/06/2025	*5*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*3*
22/06/2025	*6*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*4*
23/06/2025	*7*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*4*
		Huevos producidos					
Biomentos							*20*
Exibal							*23*
Morochillo+Biomentos							*20*
Total							*63*

Nota. Autoría propia

Observación general: Durante esta semana se registró la producción de huevos conforme a la alimentación suministrada a cada grupo, el grupo alimentado con Biomentos presentó una producción total de 20 huevos, mientras que el alimento Exibal alcanzó un total de 23 huevos, por otro lado, el grupo que recibió la mezcla de Morochillo + Biomentos registro un total de 20 huevos, dando como resultado una producción total de 63 huevos.

A partir de estos resultados se observa que el alimento Exibal presentó mayor rendimiento productivo durante esta semana, en contraste los alimentos Biomentos y Morochillo+Biomentos los cuales mostraron un comportamiento productivo similar con una producción total de 20 huevos, en sí la baja producción en esta semana puede deberse a factores propios del periodo de adaptación de las codornices a su nuevo entorno, al estrés generado por el cambio repentino de alimentación, al cambio de régimen de horario de comida parámetros a considerar en esta etapa inicial del experimento.

- ***Semana 2***

Como se puede observar en la **Tabla 9** , se tiene el registro correspondiente a la segunda semana de pruebas la cual consta desde el día 24 de junio del 2025 hasta el 30 de junio del 2025, esta segunda semana de pruebas se encuentra en el primer periodo siendo la última semana que pertenecerá al periodo de adaptación.

Tabla 9

Registro semanal de prueba experimental semana 2

Fecha	Día N°	Grupo N°	Tipo de Alimento	8:00(am) (g)	12:00(pm) (g)	16:00(pm) (g)	N° Huevos
		1	*B*	*85*	*85*	*85*	*4*
24/06/2025	*1*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*4*
25/06/2025	*2*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*3*
26/06/2025	*3*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*3*
27/06/2025	*4*	*2*	*E*	*85*	*85*	*85*	*2*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*4*
28/06/2025	*5*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*4*
29/06/2025	*6*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*4*
30/06/2025	*7*	*2*	*E*	*85*	*85*	*85*	*3*
		3	*MB*	*85*	*85*	*85*	*2*
Huevos producidos							
Biomentos							*26*
Exibal							*22*
Morochillo+Biomentos							*22*
Total							*70*

Fuente: Autoría propia

Observación general: Durante esta semana se registró la producción de huevos conforme a la alimentación suministrada a cada grupo, el grupo alimentado con Biomentos presentó una producción total de 26 huevos, mientras que el alimento Exibal alcanzó un total de 22 huevos, por otro lado, el grupo que recibió la mezcla de Morochillo + Biomentos registro un total de 22 huevos, dando como resultado una producción total de 70 huevos.

A partir de estos resultados se observa que el alimento Biomentos presentó mayor rendimiento productivo durante esta semana, en contraste los alimentos Exibal y Morochillo + Biomentos los cuales mostraron un comportamiento productivo similar con un total de 22 huevos.

Durante esta semana se pudo observar un incremento de 7 huevos en la producción total.

- ***Semana 3***

Como se puede observar en la **Tabla 10** , se tiene el registro correspondiente a la tercera semana de pruebas la cual consta desde el día 01 de julio del 2025 hasta el 07 de julio del 2025, esta tercera semana de pruebas da inicio con el segundo periodo de pronto donde ya se podrá ver cambios notables con relación al tipo de alimento suministrado.

Tabla 10

Registro semanal de prueba experimental semana 3

Fecha	Día N°	Grupo N°	Tipo de Alimento	8:00(am) (g)	12:00(pm) (g)	16:00(pm) (g)	N° Huevos
		1	*B*	*85*	*85*	*85*	*4*
01/07/2025	*1*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*4*
02/07/2025	*2*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*5*
03/07/2025	*3*	*2*	*E*	*85*	*85*	*85*	*5*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*5*
04/07/2025	*4*	*2*	*E*	*85*	*85*	*85*	*5*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*4*
05/07/2025	*5*	*2*	*E*	*85*	*85*	*85*	*4*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*5*
06/07/2025	*6*	*2*	*E*	*85*	*85*	*85*	*5*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*5*
07/07/2025	*7*	*2*	*E*	*85*	*85*	*85*	*5*
		3	*MB*	*85*	*85*	*85*	*4*
		Huevos producidos					
Biomentos							*32*
Exibal							*32*
Morochillo+Biomentos							*24*
Total							*88*

Fuente: Autoría propia

Observación general: Durante esta semana se registró la producción de huevos conforme a la alimentación suministrada a cada grupo, el grupo alimentado con Biomentos y Exibal obtuvieron una producción total de 32 huevos, mientras que la mezcla de Morochillo + Biomentos registro un total de 24 huevos, dando como resultado una producción total de 88 huevos.

A partir de estos resultados se observa que el alimento Biomentos y Exibal presentan un mayor rendimiento productivo durante esta semana, mientras que el Morochillo + Biomentos presento un comportamiento productivo bajo con un total de 24 huevos.

Durante esta semana se pudo observar un incremento de 18 huevos en relación a la semana 2 con respecto la producción total, comenzando a ver cambios en la postura de las aves.

- ***Semana 4***

Como se puede observar en la **Tabla 11**, se tiene el registro correspondiente a la cuarta semana de pruebas la cual consta desde el día 08 de julio del 2025 hasta el 14 de julio del 2025.

Tabla 11

Registro semanal de prueba experimental semana 4

Fecha	Día N°	Grupo N°	Tipo de Alimento	8:00(am) (g)	12:00(pm) (g)	16:00(pm) (g)	N° Huevos
		1	*B*	*85*	*85*	*85*	*5*
08/07/2025	*1*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
09/07/2025	*2*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
10/07/2025	*3*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*5*
11/07/2025	*4*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
12/07/2025	*5*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*6*
13/07/2025	*6*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*5*
		1	*B*	*85*	*85*	*85*	*6*
14/07/2025	*7*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*5*
		Huevos producidos					
Biomentos							*40*
Exibal							*44*
Morochillo+Biomentos							*28*
Total							*112*

Fuente: Autoría propia

Observación general: Durante esta semana se registró la producción de huevos conforme a la alimentación suministrada a cada grupo, el grupo alimentado con Biomentos obtuvo una producción total de 40 huevos, el grupo alimentado con Exibal obtuvo una producción total de 44 huevos, mientras que la mezcla de Morochillo + Biomentos registro un total de 28 huevos, dando como resultado una producción total de 112 huevos.

A partir de estos resultados se observa que el alimento Exibal presentó un mayor rendimiento productivo durante esta semana, mientras que el Morochillo + Biomentos presento un comportamiento productivo bajo.

Durante esta semana se pudo observar un incremento de 24 huevos en relación a la semana 3 con respecto la producción total, cabe mencionar que en esta semana ya se ve un cambio y una variación en cuanto a la postura de los huevos ya que el número varía de manera considerable.

- ***Semana 5***

Como se puede observar en la **Tabla 12**, se tiene el registro correspondiente a la quinta semana de pruebas la cual consta desde el día 15 de julio del 2025 hasta el 21 de julio del 2025, En esta quinta semana los cambios son evidentes.

Tabla 12

Registro semanal de prueba experimental semana 5

Fecha	Día N°	Grupo N°	Tipo de Alimento	8:00(am) (g)	12:00(pm) (g)	16:00(pm) (g)	N° Huevos
		1	*B*	*85*	*85*	*85*	*6*
15/07/2025	*1*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
16/07/2025	*2*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*5*
		1	*B*	*85*	*85*	*85*	*6*
17/07/2025	*3*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*7*
18/07/2025	*4*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*7*
19/07/2025	*5*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
20/07/2025	*6*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*3*
		1	*B*	*85*	*85*	*85*	*6*
21/07/2025	*7*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*5*
				Huevos producidos			
Biomentos							*44*
Exibal							*46*
Morochillo+Biomentos							*29*
Total							*119*

Fuente: Autoría propia

Observación general: Durante esta semana se registró la producción de huevos conforme a la alimentación suministrada a cada grupo, el grupo alimentado con Biomentos obtuvo una producción total de 44 huevos, el grupo alimentado con Exibal obtuvo una producción total de 46 huevos, mientras que la mezcla de Morochillo + Biomentos registro un total de 29 huevos, dando como resultado una producción total de 119 huevos.

A partir de estos resultados se observa que el alimento Exibal presentó un mayor rendimiento productivo durante esta semana, mientras que el Morochillo + Biomentos presento un comportamiento productivo bajo.

Durante esta semana se pudo observar un incremento de 7 huevos en relación a la semana 4 con respecto a la producción total.

- ***Semana 6***

Como se puede observar en la **Tabla 13**, se tiene el último registro correspondiente a la sexta semana de pruebas la cual consta desde el día 22 de julio del 2025 hasta el 28 de julio del 2025, En esta sexta semana se observa lo cambios muy evidentes y con esto se finaliza para poder recomendar el mejor alimento según las pruebas obtenidas.

Tabla 13

Registro semanal de prueba experimental semana 6

Fecha	Día N°	Grupo N°	Tipo de Alimento	8:00(am) (g)	12:00(pm) (g)	16:00(pm) (g)	N° Huevos
		1	*B*	*85*	*85*	*85*	*6*
22/07/2025	*1*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
23/07/2025	*2*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
24/07/2025	*3*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*7*
25/07/2025	*4*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*5*
		1	*B*	*85*	*85*	*85*	*6*
26/07/2025	*5*	*2*	*E*	*85*	*85*	*85*	*6*
		3	*MB*	*85*	*85*	*85*	*5*
		1	*B*	*85*	*85*	*85*	*7*
27/07/2025	*6*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*4*
		1	*B*	*85*	*85*	*85*	*6*
28/07/2025	*7*	*2*	*E*	*85*	*85*	*85*	*7*
		3	*MB*	*85*	*85*	*85*	*3*
		Huevos producidos					
Biomentos							*44*
Exibal							*45*
Morochillo+Biomentos							*29*
Total							*118*

Fuente: Autoría propia

Observación general: Durante esta semana se registró la producción de huevos conforme a la alimentación suministrada a cada grupo, el grupo alimentado con Biomentos obtuvo una producción total de 44 huevos, el grupo alimentado con Exibal obtuvo una producción total de 45 huevos, mientras que la mezcla de Morochillo + Biomentos registro un total de 29 huevos, dando como resultado una producción total de 118 huevos.

A partir de estos resultados se observa que el alimento Exibal presentó un mayor rendimiento productivo durante esta semana, mientras que el Morochillo + Biomentos presento un comportamiento productivo bajo.

Durante esta semana se pudo observar un decrecimiento de 1 huevo en relación a la semana 5 con respecto a la producción total.

3.2.4 Conclusiones del experimento

En base a los resultados obtenidos en el registro semanal de la producción de huevos, se presenta la **Tabla 14**, en la que se resumen los resultados comparativos del experimento.

Tabla 14

Tabla comparativa de los resultados comparativos del experimento

Tipo de Alimento	N° Semanas/ Huevos						Total de huevos /tipo de alimento
	S1	S2	S3	S4	S5	S6	
B	20	26	32	40	44	44	206
E	23	22	32	44	46	45	212
MB	20	22	24	28	29	29	152
Total	63	70	88	112	119	118	570

Fuente: Autoría propia

Al Analizar la evolución semanal de la producción se pudo observar que las primeras semanas las diferencias entre alimentos fueron poco notorias lo cual puede atribuirse al proceso

de adaptación, sin embargo, a partir de la tercera semana se comenzaron a notar cambios y a un aumento en la producción en los grupos donde solo se les suministró el balanceado tanto Exibal como Biomentos, pero a partir de la cuarta semana el cambio fue de mayor estabilidad aunque el alimento que consistía de la mezcla de morochillo molido y Biomentos la producción se mantuvo constante.

Sin embargo tomando en cuenta la **Tabla 14**, podemos mencionar que en las seis semanas de experimentación se pueden observar diferencias notorias en la producción de huevos entre los diferentes tipos de alimento suministrados, El alimento Exibal(E) presenta la mayor producción acumulada con 212 huevos , seguido de Biomentos(B)con 206 huevos, mientras que la mezcla de morochillo + Biomentos(MB) alcanzó una producción acumulada baja de 152 huevos, con estos resultados podemos evidenciar que el tipo de alimento si influye de manera directa en el rendimiento productivo de las codornices, siendo el balanceado Exibal el mejor alimento ya que presenta mejor desempeño durante el periodo experimental.

Además, desde el punto de vista económico también podemos mencionar que el costo por libra del alimento Exibal es de 0.40 CTVS, el de Biomentos es de 0.45 CTVS por libra, mientras que el morochillo tiene un valor de 0.40 CTVS. A pesar de que el morochillo tiene un valor igual al balanceado exibal y ligeramente menor al balaceado Biomentos, su baja producción de huevos reduce su eficiencia económica, por otro lado, el balanceado Biomentos pese a ser el alimento con mayor costo tiene similares resultados con el balanceado Exibal.

En función del análisis productivo y económico realizado se opta por recomendar el uso del alimento Exibal para el desarrollo propuesto ya que es el que más producción presentó durante estas seis semanas de pruebas, además este balanceado permitirá una mayor productividad, estabilidad y adecuada relación costo beneficio.

3.3 Requisitos del sistema

En esta sección es importante establecer una metodología organizada y adecuada sobre los requisitos del proyecto. En primer lugar, los actores involucrados empezando por los requisitos de usuarios o stakeholders los cuales reflejaran las expectativas y necesidades del coturnicultor y demás interesados, luego se incluyen los requisitos del sistema donde se describe las características que el sistema debe cumplir, por último tenemos los requisitos de arquitectura mismo que estable la estructura del sistema y limitaciones a considerar, bajo este enfoque estos requisitos permitirán construir un marco consistente que respalde el desarrollo del proyecto, como punto final se presenta el conjunto de requisitos con sus respectivas abreviaturas como se pude apreciar en la **Tabla *15***, con el fin de una mejor organización y seguimiento.

Tabla 15

Acrónimo de los requerimientos del sistema

Requerimiento	Abreviatura
R. Usuarios/Stakeholders	SR
R. Sistema	SyR
R. Arquitectura	AR

Nota. Autoría propia

3.3.1 Stakeholders

En esta sección identificaremos los requisitos de los stakeholders ya que mediante esto se podrá comprender las expectativas, necesidades y restricciones de las partes involucradas, además de facilitar la alineación entre los objetivos del proyecto, así como las necesidades de coturnicultor., En la **Tabla *16***, podemos observar la lista de los involucrados en el proyecto, contando como involucrado principal al señor Enrique Pupiales (propietario de las aves).

Tabla 16

Lista de involucrados en el proyecto

Involucrados	**Función**	**Participación**	
		Directa	***Indirecta***
Enrique Pupiales	Propietario de las aves	X	
Msc Jaime Michilena	Director del T.I.C.	X	
Msc. Carlos Vásquez	Asesor del T.I.C.	X	
Edison Quilca	Autor del T.I.C.	X	

Fuente: Autoría propia

3.3.1.1 Requerimientos de stakeholders

Los requerimientos se establecieron mediante una entrevista realizada al propietario de las aves, Como se puede evidenciar en la parte de **Anexo A**, en esta entrevista se abordaron temas como situación actual del criadero, visualización de los datos de monitoreo de comedero y tanque de agua, tiempos de dispensación, tipo de interfaz de visualización y notificaciones, costos, eficiencia y ventajas sistema etc. Como se puede observar en la **Tabla 17**, podemos ver los requerimientos del usuario y operacionales mismos que nos servirán como base para el diseño e implementación del sistema.

Tabla 17

Requerimientos de stakeholders y operacionales (SR)

Número	**Requerimientos**	**Prioridad**		
SR	**Usuarios o Stakeholders**	**Alta**	**Media**	**Baja**
SR1	El usuario requiere reducir la intervención manual para la	**X**		

	dispensación de alimento y agua.	
SR2	El usuario requiere que el sistema sea de fácil supervisión y manejo.	**X**
SR3	El usuario busca recudir, y proporcionar la cantidad adecuada de alimento evitando el desperdicio de alimento.	**X**
SR4	El usuario necesita visualizar el nivel de estado tanto en el contenedor de comida como en el tanque de agua.	**X**
SR5	El usuario requiere que el sistema abastezca de agua limpia y constante a las aves.	**X**
SR6	El usuario requiere optimizar el tiempo para realizar otras actividades con este sistema, sin preocuparse de	**X**

	suministrar alimento y agua.	
SR7	En cuanto a abastecimiento de contenedores tanto de alimento como de agua, el usuario requiere que se le notifique cuando el nivel sea inferior o llegue a un punto bajo/crítico.	**X**
SR8	Al momento de abastecer nuevamente de comida o agua en el contenedor este deberá emitir una alerta sonora, con el fin de alertar al usuario y evitar desbordamientos.	**X**
SR9	El usuario necesita saber la producción diaria de huevos.	**X**
SR10	El usuario requiere observar y recibir las notificaciones en un teléfono celular y que en su mayoría tengan	**X**

	gráficas por mayor facilidad de control.			
SR11	El usuario requiere una gráfica que muestre la producción de huevos de acuerdo al día en que se encuentre.	**X**		
SR12	El usuario requiere que la recolección de huevos sea mucho más fácil e higiénico, además de realizarlo solo cuando se llegue a cierta cantidad de huevos para recolectar todo.	**X**		

SR1	**Operacionales**	**Alta**	**Media**	**Baja**
SR13	El usuario requiere que el sistema funcione de manera constante sin interrupciones.	**X**		
SR14	El usuario requiere que la jaula se adapte a un espacio de 5m de largo, 2 m de ancho y 2m de alto.	**X**		

SR15	El usuario requiere que la limpieza se lo realice sin desmontar nada.	**X**
SR16	El usuario requiere se pueda realizar el mantenimiento correspondiente sin la necesidad de desmontar todo el sistema.	**X**
SR17	El usuario requiere que en caso de dañarse algún componente electrónico como motores o bombas poder reemplazarlo de manera fácil.	**X**
SR18	El usuario requiere que el sistema funcione y tenga un respaldo de energía en caso de presentar percances con el servicio eléctrico.	**X**

Fuente: Autoría propia

3.3.1.2 Requerimientos del sistema

En este apartado se definen los requerimientos del sistema los cuales definen las condiciones técnicas y funcionales necesarias para el desarrollo de la solución automatizada,

permitiendo que el sistema opere de manera eficiente y siempre acorde a las necesidades reales del criadero, es por ello que en base a estos lineamientos se establecen a partir del análisis del entorno del trabajo y de los requerimientos del coturnicultor considerando aspectos como desempeño, funcionalidad, confiabilidad y restricciones físicas, es por este motivo que en la **Tabla *18***, podemos observar los requerimientos del sistema definidos, mismos que será de ayuda y nos orientarán con el diseño e implementación del sistema de alimentación en el criadero.

Tabla 18

Requerimientos del Sistema (SyR)

Número	**Requerimientos del Sistema**	**Prioridad**		
SyR	**Interfaz**	**Alta**	**Media**	**Baja**
SyR1	El sistema debe mostrar el nivel de alimento y agua en un entorno gráfico, de acuerdo al estado de nivel.	**X**		
SyR2	El sistema debe mostrar el conteo de huevos en tiempo real, además de la visualización del progreso de producción.	**X**		
SyR3	El sistema debe permitir el acceso hacia el dashboard	**X**		

	desde cualquier dispositivo móvil.			
SyR4	El sistema debe presentar la información sobre el registro de la producción de huevos de manera ordenada y clara.	**X**		
SyR5	El sistema debe contar con un calendario de producción donde se vea la cantidad de huevos producidos en el día.	**X**		

SyR	**Uso**	**Alta**	**Media**	**Baja**
SyR6	El sistema debe permitir el acceso a la información en todo momento, siempre y cuando tenga acceso a internet.	**X**		
SyR7	El sistema facilitará a monitorización del estado de contenedores.	**X**		
SyR8	El sistema debe permitir la revisión diaria del conteo de huevos.	**X**		

SyR9	El sistema debe mantenerse en funcionamiento de manera ininterrumpida.	**X**		
SyR10	El sistema debe enviar alertas en tiempo real de acuerdo al estado de los contenedores tanto de alimento como de agua.	**X**		

SyR	**Estados**	**Alta**	**Media**	**Baja**
SyR11	El sistema debe identificar los niveles como lleno, medio, vacío.	**X**		
SyR12	El sistema debe tener la capacidad de enviar alertas y notificaciones cuando se detecte un nivel crítico mediante umbrales.	**X**		
SyR13	El sistema debe evitar alertas y notificaciones cuando se mantenga en un solo estado.	**X**		

SyR14	El sistema debe transmitir los datos recopilados en tiempo real.	**X**		

SyR	Físicos	Alta	Media	Baja
SyR15	El sistema debe ser instalado dentro de la jaula del criadero.	**X**		
SyR16	El sistema debe tener las condiciones adecuadas y la distribución necesaria dentro de la jaula tanto para el dispensador de alimento como para el dispensador de agua.	**X**		
SyR17	El sistema deberá contar con una banda transportadora para la distribución uniforme del alimento, así como un bebedero adecuado apto para el rellenado de agua.	**X**		
SyR18	El sistema debe proteger los componentes electrónicos del polvo y agua.	**X**		

SyR19	El sistema debe integrarse sin afectar el bienestar de las codornices.	**X**		

Fuente: Autoría propia

3.3.1.3 Requerimientos de la arquitectura

En este apartado se da a conocer los requisitos de la arquitectura los cuales establecen condiciones técnicas y funcionales necesarias para un buen desarrollo, operación y adaptación en torno al criadero, lo lineamientos que contemplan este parámetro se organizan en categorías como software, hardware, criterios lógicos y de diseño, así como la parte eléctrica, estos requerimientos priorizan según su impacto en el funcionamiento integral del sistema.

En la **Tabla *19***, podemos observar estos requerimientos de arquitectura.

Tabla 19

Requerimientos de Arquitectura (AR)

Número	**Requerimientos de Arquitectura**	**Prioridad**		
AR	**Hardware**	**Alta**	**Media**	**Baja**
AR1	El sistema requiere un microcontrolador como unidad de control principal.	**X**		
AR2	El sistema integrará componentes electrónicos como bombas de agua, motores eléctricos tanto	**X**		

	para el dispensado alimento como para el suministro de agua al bebedero.			
AR3	El sistema deberá integrar sensores óptimos para el conteo de huevos y medición del nivel de los contenedores.	**X**		
AR4	El sistema integrará componentes electrónicos que satisfagan con las alertas sonoras y la activación o desactivación de bombas y motores.	**X**		
AR5	El sistema debe contar con un calendario de producción donde se vea la cantidad de huevos producidos en el día.	**X**		

AR	**Software**	**Alta**	**Media**	**Baja**
AR6	El sistema debe garantizar una IDE de programación de código abierto que contenga todas la	**X**		

	bibliotecas y librerías necesarias.			
AR7	El sistema debe contar con una base de datos en tiempo real.	**X**		
AR8	El software debe permitir el registro histórico de datos.	**X**		
AR9	El software debe ser compatible con los dispositivos móviles.	**X**		
AR10	Plataforma en la nube para almacenar y visualizar los datos generados por los sensores.	**X**		
AR	**Lógicos y diseño**	**Alta**	**Media**	**Baja**
AR11	El sistema debe separar las funciones de control y medición.	**X**		
AR12	El sistema debe tener coherencia en el flujo de datos.	**X**		
AR13	El sistema debe adaptarse al diseño de la jaula, además de proteger los	**X**		

	componentes electrónicos.			
AR14	El diseño debe garantizar la durabilidad del sistema y minimizar el impacto en las codornices.	**X**		
AR	**Eléctricos**	**Alta**	**Media**	**Baja**
AR15	El sistema debe ser suministrado con el voltaje adecuado de acuerdo a los requerimientos de los componentes.	**X**		
AR16	El sistema debe permitir la desconexión segura para mantenimiento.	**X**		
AR17	El sistema debe contar con un respaldo de energía y que garantice el funcionamiento constante.	**X**		

Fuente: Autoría propia

3.4 Selección de componentes (Hardware/Software)

A continuación tenemos la parte de selección de componentes tanto como de hardware como software los cuales nos permitirá establecer los componentes y elementos necesarios para

la construcción y funcionamiento del sistema que se realizará, Cabe mencionar que para la elección del componente o elementos apropiados se llevará a cabo un proceso de análisis donde se consideran aspectos importante como facilidad de adquisición, costo, compatibilidad, rendimiento y facilidad de integración con otros elementos del sistema, gracias a esto se podrá escoger el mejor componente el cual cumplirá de mejor manera a los requerimientos técnicos y operativos del proyecto antes establecidos. A continuación, se muestran los componentes seleccionados mismos que están establecidos mediante etapas.

3.4.1 Etapa física del sistema

Esta etapa abarca elementos de hardware los cuales serán los responsables de la interacción con el entorno y ejecución de acciones ,por lo que en esta fase se seleccionan componentes que aseguren la correcta integración y funcionamiento en el sistema automatizado, dicho esto , en esta etapa se evalúan y se seleccionan distintos tipos de sensores tanto de nivel como de detección de objetos para lo cual se consideran características como accesibilidad, voltaje de operación, torque, velocidad/caudal, corriente, el método y rango de medición, costos etc. Con el objetivo de garantizar la selección de los componentes más eficiente para el diseño y construcción del sistema de alimentación, adicional a esto, también se incorporan motores y servomotores mismos que nos ayudarán a dispensar el alimento de manera correcta, así mismo tenemos las bombas de agua los cuales nos ayudarán con la dispensación constante del agua.

Además, en este apartado tendremos la selección de un microcontrolador mismo que actúa como núcleo del sistema el cual será el encargado de recibir datos de los sensores, ejecutar rutinas de control previamente programadas y el envío de información a través de conexión inalámbrica.

3.4.1.1 Sensor de medición y detección

En este apartado se analizará las características de cada tipo de sensor los cuales nos servirán para la medición del nivel del estado de nuestros contenedores tanto de alimento como de agua, por otro lado también se escogerá el más óptimo para poder detectar huevos, para ello es muy importante y fundamental analizar las características técnicas de cada uno de los sensores de medición, de acuerdo a esto escoger el más óptimo garantizando la precisión la compatibilidad y la eficiencia con el resto de componentes del sistema, para la selección del mejor componente se pone en comparativa 4 tipos de sensores tanto para la parte de sensores de medición de distancia como 4 de sensores de proximidad como se puede observar en la **Tabla *20***, de los cuales se evidenciará características como accesibilidad, voltaje de operación, corriente, método de medición, rango de medición, dimensión/tamaño y el costo de cada uno de los componentes.

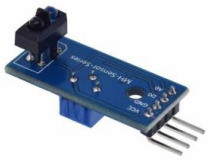
Tabla 20

Tabla comparativa de sensores de medición de distancia y detección de objetos

Sensores de medición de distancia				
Nombre comercial	**Sensor de nivel de agua ZPC1**	**HC-SR04**	**VL53LOC**	**MB7076**
Imagen				
Accesibilidad	Alta	Alta	Media	Media
Voltaje de Operación	Max 220V/AC	5V	3.3V - 5V DC	3V - 5.5V DC
Corriente	Max:3ª	15mA	20mA	3.4mA
Método de medición	Mediante un interruptor de flotador	Mediante ultrasonido(pulso)	Mediante tecnología Time of Flight (pulso)	Mediante ultrasonido, utilizando frecuencia de 42kHz

Rango de medición	Nivel puntual, no continuo	Min 2cm Max 4m	Min 500mm Max 1200mm	Min 1 cm Max 10.68m
Dimensión/tamaño	Compacto, flotador integrado con longitud de cable de 37cm	45*20*15mm	4.8*2.9*1.2mm	43.8,50.7mm y rosca estándar 3/4
Compatibilidad	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry
Costo	$6	$2.70	$8	$127

Sensores de detección de objetos

Nombre comercial	**Sensor infrarrojo FC51**	**Sensor infrarrojo TCRT5000**	**Módulo sensor de movimiento PIR**	**Sensor Grove-AK9753**
Imagen				
Accesibilidad	Alta	Alta	Alta	Media
Voltaje de Operación	3.3V-5V DC	3.3V-5V DC	4.5V-12V DC	3.3V-5V DC
Corriente	Min 23mA Max 43mA	15mA	<50uA	0.2mA
Método de medición	Medición por reflexión infrarroja	Medición por reflexión infrarroja	Medición por detección de movimiento por infrarrojo pasivo (PIR)	Medición por infrarrojo pasivo (IR)
Rango de medición	Min 2 cm Max 30 cm	Min 0.5mm Max 1.5mm	Min 3m Max 7m	Max 3m
Dimensión/tamaño	31*14mm	31*13*10mm	32.7*29mm	40*20mm
Compatibilidad	Arduino Esp32 Raspberry	Arduino Esp32	Arduino Esp32	Arduino Esp32

		Raspberry	Raspberry	Raspberry
Costo	$2.00	$2.50	$3	$18

Nota. En la siguiente tabla se muestran sensores tanto de medición de distancia como detección de objetos presentes en el mercado actual, obtenida de (ElectroStore, 2025) & (ElecFreaks, 2025) & (JA-bots, 2025) & (MaxBotix, 2025)& (AGElectrónica, 2024)& (UnitElectronics, 2025)& (Novatronic, 2020)& (Kamami, 2025)

- ***Componente seleccionado (Medición de distancia)***

De acuerdo a análisis realizado a partir de las características técnicas de los diferentes componentes presentes en el mercado cuya función es la medición de distancia, se determinó que el que más se adapta para el desarrollo del sistema es el **sensor de distancia ultrasónico HC-SR04**, ya que a diferencia con los otros sensores(sensor de nivel de agua, VL53LOC y MB7076) tiene una alta accesibilidad en el mercado lo que nos facilita para su adquisición y reemplazo en caso de requerir, además opera con un voltaje de 5v, valor necesario para el funcionamiento del sistema, además de presentar un bajo consumo energético.

Otra característica importante para la elección de este componente es el rango de medición, e cual se ajusta a las necesidades del sistema, ya que para el sistema se requiere monitorear el nivel de alimento en un contenedor una altura máxima de 50 cm, así como el nivel de agua en un tanque de agua con una altura aproximada de 40 cm, siendo este sensor el más idóneos para cubrir estas distancias.

Adicionalmente su principio de funcionamiento basado en la emisión de ondas sonoras de alta frecuencia y la medición del tiempo de retorno del eco en regresar tras rebotar en un objeto ya sea sólido o líquido lo que lo hace ideal para determinar con precisión el nivel de alimento y agua dentro de los contenedores.

Además, se consideraron aspectos físicos y económicos para su elección, el tamaño compacto por sus dimensiones la convierte en el más ideal ya que su adaptación dentro de los

contenedores será de fácil instalación además el costo en comparación con los otros 3 sensores es el más económico lo que lo posiciona como la alternativa más eficiente.

Finalmente mencionar que otro punto importante a considerar es que cumple con los requerimientos de usuario SR1, SR7, SR16, SR17 presentes en la **Tabla *17***, también los requerimientos de sistema SyR7, SyR10, SyR11, SyR12 presentes en la **Tabla *18***, y por último los requerimientos de arquitectura AR2 y AR3 presentes en la **Tabla *19***.

- ***Componente seleccionado (Detección de objetos)***

De acuerdo a análisis realizado a partir de las características técnicas de los diferentes componentes presentes en el mercado cuya función es la detección de objetos, se determinó que el que más se adapta para el desarrollo del sistema es el **sensor infrarrojo FC51** , ya que a diferencia con los otros sensores(Sensor infrarrojo TCRT5000, Módulo sensor de movimiento PIR y Sensor Grove-AK9753) tiene una alta accesibilidad en el mercado lo que nos facilita para su adquisición y reemplazo en caso de requerir, además opera con un voltaje de 3.3.V-5v, valor necesario para el funcionamiento del sistema, además de presentar un bajo consumo energético ya que está en los rango de Min 23mA y Max 43mA .

Otra característica importante para la elección de este componente es el rango de medición, el cual se ajusta a las necesidades del sistema, ya que para el sistema se requiere detectar los huevos a una altura máxima de 7cm y el sensor esta entre el rango de 2cm a 30cm, debido a que los huevos de la codorniz presentan un diámetro longitudinal de 3.14 cm y un diámetro transversal de 2.41 cm, como se muestra en a página 35, con esto garantizar una distancia considerable entre el sensor de detección y el huevo esto con el fin de realizar el correcto conteo de los huevos.

Adicionalmente su principio de funcionamiento basado en la emisión de luz infrarroja y la detección de la señal reflejada por un objeto cercano, permitiéndonos identificar la presencia o

ausencia al momento de pasar un objeto sin la necesidad de contacto físico, en el caso del sistema será muy útil para el conteo al pasar el huevo.

Además, se consideraron aspectos físicos y económicos para su elección, el tamaño compacto por sus dimensiones de 31*14mm la convierte en el más ideal para su adaptación dentro del canal por donde se desplazarán los huevos, además el costo en comparación con los otros 3 sensores es el más económico con un valor de $2 4 siendo una alternativa más rentable en cuanto a costos.

Finalmente mencionar que otro punto importante a considerar es que cumple con los requerimientos de usuario SR9, SR12 y SR17 presentes en la **Tabla *17***, también los requerimientos de sistema SyR2, SyR5 y SyR8 presentes en la **Tabla *18***, y por último los requerimientos de arquitectura AR3 presentes en la **Tabla *19***.

3.4.1.2 Servomotor

En este apartado el servomotor será útil para el accionamiento del dispensador de alimento es por ello que se seleccionará un servomotor cuya función permita abrir y cerrar el mecanismo de dispensación de comida de manera precisa, para la elección de este componente se analizarán tres tipos de servomotores como se puede observar en la **Tabla *21*** , estos servomotores están presentes en el mercado actual , dicho esto se analizaran sus características técnicas como accesibilidad, voltaje de operación, consumo de corriente, tipo de movimiento, tipo de control, torque, dimensiones/tamaño, costo, de acuerdo a estas características se elegirá al idóneo que satisfaga con los requerimientos para la construcción del sistema de dispensación de alimento.

Tabla 21

Tabla comparativa de servomotores

Servomotores

Nombre Comercial	MG995 Tower Pro	SG90 9g 180°	Micro FS90R
Imagen			
Accesibilidad	Alta	Alta	Alta
Voltaje de Operación	4.8V-7.2V DC	4V-7.2V DC	4.8V-6V DC
Corriente	Min 100mA Max 2 A	Min 100 mA Max 700 mA	Min 100 mA Max 800 mA
Tipo de movimiento	Angular Rotación continua	Angular	Rotación continua
Tipo de control	PWM	PWM	PWM
Ángulo de rotación	0° - 180° 360°	0° - 180°	360°
Torque	Engranajes metálicos (360°): 8.5 kg*cm a 4.8 V 10 kg*cm a 6 V Engranajes plásticos (180°): 6.8 kg*cm a 4.8 V 8 kg*cm a 6 V	1.2Kg*cm a 4.8V	1.3 kg*cm a 4.8 V 1.5 kg*cm a 6 V
Dimensión/tamaño	40.7*19.7*-42.9mm	22*11.5*27mm	23.2*12.5*22mm
Compatibilidad	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry
Costo	$12	$4	$10

Nota. En la siguiente tabla podemos observar tres tipos de servomotores presentes en el mercado actual cada uno muestra sus características técnicas, obtenida de (Tecmikro, 2025) & (Novatronic, 2020) & (Dualtronica, 2025)

- ***Componente seleccionado***

De acuerdo a análisis realizado a partir de las características técnicas de los diferentes servomotores presentes en el mercado actual, se determinó que el que más se adapta para el accionamiento del dispensador de alimento es el **servomotor SG90 9g 180° ,** ya que a diferencia con los otros servomotores(MG995 Tower Pro y Micro FS90R) tiene una alta accesibilidad en el mercado lo que nos facilita para su adquisición y reemplazo en caso de requerir, además opera con un voltaje de 4V-7.2V DC, valor necesario para el funcionamiento del sistema, además de presentar un bajo consumo energético ya que está en los rango de Min 100mA y Max 700mA.

Otra característica importante para la elección de este componente es su tipo de movimiento angular con un ángulo de 0° a 180° controlado por una señal PWM, el cual se ajusta a las necesidades del sistema, ya que para el sistema se requiere abrir o cerrar el dispensador de alimento, en comparación del servomotor FS90R el cual realiza rotación continua de 360° que esta diseñado para rodillos donde no se requiere una posición fija, así mismo, el otro servomotor MG995 el cual tiene la peculiaridad de que tiene un torque muy elevado por lo que no es muy óptimo para el sistema ya que demandaría mayor consumo de energía por el elevado torque que este posee es por ello también que la elección se hizo por el servomotor SG90 9g 180°.el cual a más de satisfacer con el sistema ,tendrá un menor consumo de energía.

Además, se consideraron aspectos físicos y económicos para su elección, el tamaño compacto por sus dimensiones de 22*11.5*27mm la convierte en el más ideal para su adaptación en la boca del embudo por donde se dispensará el alimento, además el costo en comparación con los otros 2 servomotores es el más económico con un valor de $4 siendo una alternativa más rentable en cuanto a costos.

Finalmente mencionar que otro punto importante a considerar es que cumple con los requerimientos de usuario SR1, SR3 y SR17 presentes en la **Tabla 17**, también los

requerimientos de sistema SyR5 y SyR16 presentes en la **Tabla 18**, y por último el requerimiento de arquitectura AR2 presente en la **Tabla 19**.

3.4.1.3 Motor

En este apartado la elección de los motores será útil para la creación de una banda transportadora de alimento con el fin de que el alimento se distribuya de manera uniforme para el fácil acceso de las codornices y tengan una mejor forma de alimentación, es por ello que se seleccionará un motor cuya función permita mover el mecanismo de la banda transportadora de manera precisa, para la elección de este componente se analizarán tres tipos de motores como se puede observar en la **Tabla *22*** , estos motores están presentes en el mercado actual , dicho esto se analizaran sus características técnicas como voltaje de operación, consumo de corriente, torque, velocidad de respuesta, tamaño, accesibilidad en el mercado y costo, de acuerdo a estas características se elegirá al idóneo que satisfaga con los requerimientos para la construcción del sistema de dispensación de alimento.

Tabla 22

Tabla comparativa de motores

Motores			
Nombre Comercial	Motor JGY370	Mini motor DC	Motor DC reductora amarilla
Imagen			
Accesibilidad	Media	Alta	Alta
Voltaje de Operación	12V DC	3V-6V DC	3V-9V DC
Corriente	Min 60mA Max 0.6A	Min 0.35 A Max 0.4 A	Min 200 mA

			Max 600 mA
Torque	22.5 kg*cm	Bajo	2-3 kg*cm
Tipo de control	PWM mediante driver	PWM mediante driver	PWM mediante driver
Velocidad de rotación	10RPM	8000RPM	70-150RPM
Dimensión/tamaño	Tamaño de motor:76*32mm Diámetro de eje:6mm Longitud de eje:14mm	Tamaño del motor:25*15*20mm Diámetro de eje:2mm Longitud de eje:9mm	Tamaño del motor:70*22*18mm Doble eje
Compatibilidad	Arduino Esp32 Raspberry (mediante uso de driver de motor)	Arduino Esp32 Raspberry (mediante uso de driver de motor)	Arduino Esp32 Raspberry (mediante uso de driver de motor)
Costo	$25	$2	$2.50

Nota. En la siguiente tabla podemos observar tres tipos de motores presentes en el mercado actual cada uno muestra sus características técnicas, obtenida de (Novatronic, 2020) & (AV Electronics, 2025) & (Megatronica, 2025)

- ***Componente seleccionado***

De acuerdo a análisis realizado a partir de las características técnicas de los diferentes motores presentes en el mercado actual, se determinó que el que más se adapta para realizar la banda transportadora de alimento el **Motor DC reductora amarillo ,** ya que a diferencia con los otros motores(Motor JGY370 y Mini motor DC) tiene una alta accesibilidad en el mercado lo que nos facilita para su adquisición y reemplazo en caso de requerir, además opera con un voltaje de 3V-9V DC, valor necesario para el funcionamiento del sistema.

Además a pesar de que el motor DC reductora amarillo presenta un consumo de corriente menor que el Motor JGY370 y ligeramente mayor al mini motor DC, ofrece un equilibrio adecuado entre consumo energético, torque y velocidad de rotación ya que el Motor DC

reductora amarillo trabaja en un rango aproximado de 70-150 RPM, lo que permitirá un desplazamiento estable de la banda transportadora de alimento ya que debido a su torque de 2-3 kg*cm a será suficiente para mover el alimento a lo largo de la banda transportadora, a comparación del motor JGY370 que su torque es de 22.5 kg*cm y una velocidad apenas de 10 RPM lo que causaría un mayor consumo de energía por lo que sería algo innecesario para la banda transportadora de comida, mientras que el mini motor DC no es apta para cargar cosas pesadas ya que su torque es bajo e impide mover cargas y sería inútil para la banda transportadora.

Además, se consideraron aspectos físicos y económicos para su elección, el tamaño compacto por sus dimensiones de motor:70*22*18mm y tener doble eje la convierte en el más ideal para su adaptación en la banda transportadora de alimento alimento, además el costo en comparación con los otros 2 servomotores es el más económico con un valor de $2.50 siendo una alternativa mucho mejor, aunque el costo sea ligeramente superior al mini motor DC.

Finalmente mencionar que la elección de este componente cumple con el requerimiento de usuario SR17 presente en la **Tabla 17**, también los requerimientos de sistema SyR17 y SyR19 presentes en la **Tabla 18**, y por último el requerimiento de arquitectura AR2 presente en la **Tabla 19**.

3.4.1.4 Bomba de agua

En este apartado la elección de las bombas de agua será útil para la dispensación de agua del contenedor hacia el bebedero, con el objetivo de que siempre permanezcan con agua las codornices, es por ello que se seleccionará una bomba cuya función permita la dispensación correcta del agua, para la elección de este componente se analizarán tres tipos de bombas como se puede observar en la **Tabla *23*** , estas bombas están presentes en el mercado actual , dicho esto se analizaran sus características técnicas como accesibilidad en el mercado voltaje de operación, consumo de corriente, caudal, tamaño, costo. De acuerdo a estas características se

elegirá al idóneo que satisfaga con los requerimientos para la construcción del sistema de dispensación de agua.

Tabla 23

Tabla comparativa de bombas de agua

Bombas de agua			
Nombre Comercial	Mini Bomba de agua sumergible 5V	Bomba de agua de diafragma	Mini Bomba de agua sumergible 5Vsin escobillas
Imagen			
Accesibilidad	Alta	Media	Media
Voltaje de Operación	2.5V-6V DC	12V DC	5V-12V DC
Corriente	Min 130mA Max 220mA	6 A	Max 0.38 mA
Flujo	80-120 L/h	6 L/m	4L/min
Elevación máxima de bombeo de forma vertical	40cm Presión máx: 3-5(PSI)	80 a 90 m Presión max:120-130 (PSI)	3m Presión max:4.3 (PSI)
Condición de uso	Sumergido	No sumergible	Sumergible y no sumergible
Diámetro de salida	7.5mm	10mm	8mm
Dimensión/tamaño	Tamaño de la bomba:43*23mm Largo cable:20cm	Tamaño de la bomba: 16.5*10*6.2cm	Tamaño de la bomba:51*34*42.7mm

Compatibilidad	Arduino Esp32 Raspberry (mediante uso de relé o transistor)	Arduino Esp32 Raspberry (mediante uso de relé o módulo de potencia)	Arduino Esp32 Raspberry (mediante uso de driver, relé o módulo de potencia)
Costo	$3.50	$30-45	$18-25

Nota. En la siguiente tabla podemos observar tres tipos de bombas de agua presentes en el mercado actual cada uno muestra sus características técnicas, obtenida de (ElectroStore, 2025) & (SHENZHEN Zhongke Century Technology Co, 2025) & (Novatronic, 2020)

- ***Componente seleccionado***

De acuerdo a análisis realizado a partir de las características técnicas de las diferentes bombas presentes en el mercado actual, se determinó que el que más se adapta para realizar la dispensación de agua de forma correcta es la **Mini Bomba de agua sumergible 5V ,** ya que a diferencia con las otras bombas (Bomba de agua de diafragma y Mini Bomba de agua sumergible 5Vsin escobillas) tiene una alta accesibilidad en el mercado lo que nos facilita para su adquisición y reemplazo en caso de requerir, además opera con un voltaje de 2.5V-6V DC, valor necesario para el funcionamiento del sistema.

Además en cuanto al consumo de corriente tiene un valor Min 130mA y Max 220mA lo que hace que esta bomba sea un componente de bajo consumo energético, otro punto a considerar es la elevación de bombeo máxima en forma vertical, si bien, las bombas de agua (Bomba de agua de diafragma y Mini Bomba de agua sumergible 5Vsin escobillas) presentan una elevación de bombeo superior a los 3 metros, el contenedor donde lo implementaremos tiene una altura máxima de 40cm, también el flujo de caudal es demasiado para el sistema a desarrollar ya que comprenden valores de 6L/m y 4L/m, por lo que la bomba seleccionada es óptima ya que también nos ayudará a un ahorro energético.

Otro punto importante a considerar fueron los aspectos físicos y económicos para su elección, el tamaño compacto por sus dimensiones de la bomba de 43*23mm, cable de 20cm

y un diámetro de salida de 7.5mm suficientes para el correcto dispensado de agua, además de ser sumergible, adecuado para ponerlo dentro del tanque de agua.

Por último, el costo en comparación con las otras 2 bombas de agua es el más económico con un valor de $3.50 siendo una alternativa mucho mejor.

Finalmente mencionar que la elección de este componente cumple con el requerimiento de usuario SR5 presente en la **Tabla 17**, también los requerimientos de sistema SyR17 y SyR19 presentes en la **Tabla 18**, y por último el requerimiento de arquitectura AR2 presente en la **Tabla 19**.

3.4.1.5 Microcontrolador

En este apartado la elección de un buen microcontrolador es fundamental para un correcto funcionamiento del sistema a desarrollar ya que este componente actúa como cerebro del sistema porque será el encargado de recibir datos de los sensores, el control, monitoreo y procesamiento de la información a través de conexiones inalámbricas, además tener en cuenta que el microcontrolador que se escoja será el encargado de gestionar la medición de los niveles en los contenedores, la activación de servomotores y motores, bombas de agua, además de la comunicación con plataformas externas y la toma de decisiones en tiempo real.

Por lo tanto para una buena elección del mejor microcontrolador para el sistema de control de alimentación y elegir el más idóneo, se realizará un análisis comparativo de seis microcontroladores disponibles en el mercado actual mismos que se pueden evidenciar en la **Tabla 24**, entre los criterios evaluados se consideran la accesibilidad en el mercado, frecuencia, voltaje de operación , corriente, CPU, Pines I/O digitales y Pines I/O análogas, los cuales nos servirán para la conexión de (sensores, motores, servomotores etc), memoria RAM, memoria FLASH, Dimensión/tamaño, capacidad de comunicación, periféricos integrados y costo de cada uno de los microcontroladores.

Por último, en base al análisis se determinará el microcontrolador que mejor se adapte a los requerimientos del sistema y los objetivos del sistema.

Tabla 24

Tabla comparativa de microcontroladores

Nombre Comercial	Arduino mega 2560	ESP8266	ESP32-DevKit V1	Raspberry pi 4
Imagen				
Accesibilidad	Alta	Alta	Alta	Alta
Frecuencia	16MHz	80MHz	240 MHz	133MHz
Voltaje de Operación	Funcionamiento: 5V DC Entrada:7V-12V DC	Funcionamiento: 5V DC Entrada:4.5V -18V DC	Funcionamiento: 5V DC Entrada:4.5V DC	3.3V-5VDC
Corriente	Max 50mA	Max 200mA	Max 80-260mA	Max 150mA
CPU	ATmega2560, 8bits	Esp8266, 32 bits	Tensilica Xtensa LX6 de 32 bits, doble núcleo, hasta 600 DMIPS	Dual core Arm Cortex-M0
Pines digitales	54 15(PWM)	17	Físicos:30 GPIO:24	26
Pines analógicos	16	1	2 ADC tipo SAR de 12 bits	3 canales ADC de 12 bits
Memoria RAM	8KB	64KB	520KB	264KB

Memoria Flash	256KB	4MB	4MB	2MB
Dimensión/tamaño	101.52*53mm	46*26 mm	52*28.5*15mm	51mm*21mm
Capacidad de comunicación	Comunicación Serial	WiFi	WiFi Bluetooth Antena	WiFi
Periféricos integrados	ADC, PWM, UART, SPI, I²C, temporizadores	UART, SPI, I²C, SDIO PWM, ADC y GPIO	ADC, PWM, UART, SPI, I²C	2 * SPI 3 * ADC 12 bits 2 * I²C 2 *UART 16 * canales PWM
IDE de desarrollo	Arduino IDE Lenguaje: C/C++	Arduino IDE Lenguaje: C/C++	Arduino IDE, Platform IO, ESP-IDF Lenguaje: C/C++	Arduino IDE,Thonny,SDK Raspberry pi Lenguaje: C/C++, MicroPython
Costo	$25	$10	$15	$15

Nota. En la siguiente tabla podemos observar cuatro tipos de mirocontroladores presentes en el mercado actual cada uno muestra sus características técnicas, obtenida de (Unit Electronics, 2025) & (Novatronic, 2020) & (Megatronica, 2025) & (ElectroStore, 2025)

- ***Componente seleccionado***

De acuerdo al análisis comparativo realizado entre cuatro microcontroladores como son Arduino Mega 2560, ESP8266, ESP32 DevKit V1 y Raspberry Pi Pico los cuales son los más usados para proyectos IoT se hace la comparativa entre sus características técnicas sin embargo , no todos cumplen con los requerimientos necesarios para el sistema propuesto ya que el sistema propuesto demanda de conectividad inalámbrica, capacidad alta de procesamiento, además debe contar con presencia de múltiples pines ya que se necesita conectar sensores, motores, bombas de agua etc. Con ello cumplir con los objetivos del trabajo.

Es por ello que el que mejor se adapta a los requerimientos del sistema es el microcontrolador **ESP32 DevKit V1** ya que este componente a diferencia del Arduino Mega este ofrece conectividad WiFi y Bluetooth, otro punto importante es que a diferencia del ESP8266 presenta una mayor cantidad de pines GPIO además de un procesador de doble núcleo, otro punto a considerar es la capacidad de memoria el cual es superior a diferencia del ESP8266 y Raspberry Pi Pico.

Otro punto que se consideró para elegirlo como el ideal es que ESP32 DevKit V1 posee múltiples entornos de desarrollo como Arduino IDE, Platform y ESP-IDF, además de su accesibilidad alta en el mercado en caso de presentar fallos poder reemplazarlo de una manera fácil, de igual modo, considerando su costo es el más económico en cuanto a microcontroladores con la capacidad de comunicación por WiFi, aunque el ESP8266 es mucho más económico no cumple con las prestaciones que el sistema a desarrollar requiere, por ultimo su tamaño es ideal para poder implementarlo en un lugar estratégico en la jaula.

Finalmente mencionar que la elección de este componente cumple con los requerimientos de usuario SR1, SR13, SR17 presente en la **Tabla *17***, también los requerimientos de sistema SyR6 y SyR16 presentes en la **Tabla *18***, y por último los requerimientos de arquitectura AR2 y AR6 presente en la **Tabla *19***.

3.4.1.6 Dispositivos de control y actuación

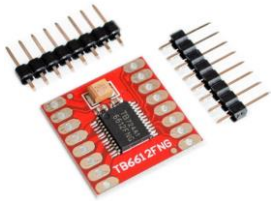
En este apartado se muestran dispositivos de control y actuación ya que serán de mucha utilidad para poder adaptarlo al sistema ya que permitirán la interacción directa con las decisiones tomadas por el microcontrolador, estas decisiones van desde la activación de motores los cuales estarán presentes en la banda transportadora, la activación de bombas de agua los cuales estarán dentro del contenedor de agua, así mismo se elegirá el mejor buzzer de acuerdo a los requerimientos del sistema.

- ***Driver de control para motor (puente H)***

La elección del mejor driver de control para motor será esencial para poder crear la banda transportadora de alimento, ya que este será el encargado de manejar el sentido de giro y control de movimiento de los motores debido a que actuará como el puente entre el microcontrolador y los motores, por esta razón a continuación en la **Tabla *25***, se muestran 2 tipos de drivers para el control de motores presentes en el mercado actual donde se escogerá el que más se integre al sistema a desarrollar.

Tabla 25

Tabla comparativa de drivers de control para motor

Driver de control para motor		
Nombre Comercial	L298N	TB6612FNG
Imagen		
Accesibilidad	Alta	Alta
Voltaje de operación	5V-35V DC	4.5V-13.5V DC
Voltaje lógico	5V DC	2.7V-5.5V DC
Capacidad de corriente	Max 2 A x motor	Max 3 A por canal
Corriente de control	0-36mA	5-25uA
Cantidad de motores	2 motores DC o 1 Motor paso a paso	2 motores DC o 1 Motor paso a paso
Dimensión/tamaño	43*43*27mm	32*32mm
Compatibilidad	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry

Costo	$3	$7

Nota. En la siguiente tabla podemos observar dos tipos de drivers para el control de motores presentes en el mercado actual, cada uno muestra sus características técnicas, obtenida de (Novatronic, 2020) & (Megatronica, 2025)

- **Componente seleccionado**

Después de un análisis comparativo de sus características técnicas entre el driver controlador L298N y TB6612FNG se seleccionó como el más idóneo al driver de control **L298N** ya que presenta un rango de voltaje mayor al TB6612FNG lo que permitiría a futuro en caso de requerir el reemplazo de nuevos motores con un capacidad ya que se lo podría hacer sin ningún problema además en cuanto a su accesibilidad es muy alta en el mercado actual por lo que sería fácil de reemplazar en caso de presentar daños.

Otra razón es la gran cantidad de documentación donde se evidencia el uso de este componente en proyectos prácticos de automatización, finalmente cabe mencionar que otra característica importante es la adaptabilidad con microcontroladores como Arduino, esp32 y Raspberry, siendo la mejor opción para el desarrollo del sistema propuesto.

- ***Dispositivos de conmutación eléctrica***

En este apartado será fundamental elegir un dispositivo de conmutación eléctrica para tener un control adecuado de las bombas de agua ya que este dispositivo servirá como un puente entre el microcontrolador y las bombas de agua evitando así daños, gracias a esto se podrá dispensar de manera correcta el agua y mantener de manera constante agua limpia en el bebedero.

Para su elección se analizarán diferentes características técnicas de cada uno de los dispositivos presentes en el mercado actual como se puede observar en la **Tabla *26***, finalmente se elegirá el que mejor se adapte a los requerimientos para el desarrollo del sistema.

Tabla 26

Tabla comparativa de dispositivos de conmutación eléctrica

Dispositivos de conmutación eléctrica			
Nombre Comercial	Módulo de relé 5V	Módulo Relé electromecánico Jqc-3FF-S-Z-05V	Mini Bomba de agua sumergible 5Vsin escobillas
Imagen			
Accesibilidad	Alta	Alta	Media
Voltaje de Operación	5V DC	3.75V-6V DC	3V-32V DC
Corriente de activación	20mA	70mA	7.5 mA a 12V DC
Voltaje MAX que permite en los contactos	250V AC o 30 V DC	250V AC o 30 V DC	24 a 380 VAC
Corriente MAX que permite en los contactos	10 A	10 A	24 A
Tipo de contacto	1 polo, 2 tiros (SPDT)	1 polo, 2 tiros (SPDT)	Control CC a CA sin contacto mecánico
Dimensión/tamaño	46.5*12*18.5mm	16.5*10*6.2cm	62*45*23mm
Compatibilidad	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry PLC
Costo	$2.50	$3	$6-12

Nota. En la siguiente tabla podemos observar tres tipos de dispositivos de conmutación eléctrica presentes en el mercado actual, cada uno muestra sus características técnicas, obtenida de (Novatronic, 2020) & (ElectroStore, 2025) & (Components101, 2023)

- **Componente seleccionado**

El cuanto al componente seleccionado la mejor opción y el elemento seleccionado es el **módulo relé de 5V** debido a que de acuerdo a sus características este tiene un voltaje de operación de 5V DC también que para su funcionamiento tiene un consumo bajo de 20mA, también este componente presenta compatibilidad con distintos microcontroladores presentes en el mercado, otro punto importante es su fácil accesibilidad y bajo costo ya que es el más económico en comparación a los otros dos dispositivos de conmutación.

Además, este componente ofrece un aislamiento eléctrico mediante un optoacoplador lo cual es una ventaja para proteger el sistema frente a picos de voltaje, también un punto más importante a considerar es el voltaje máximo que permite en los contactos es de 250 V AC o 30 V DC y un amperaje de 10 A los cuales resultan más que suficientes para accionar las bombas de agua, es por esto que este componente se presenta como la mejor opción para su integración en el sistema a desarrollar.

- ***Dispositivo de alerta sonora***

Para la alerta sonora el cual estará ubicado en el contenedor tanto de comida como de agua será de mucha importancia escoger el más idóneo ya que este dispositivo nos permitirá alertar al cotornicultor hasta qué punto se debe de abastecer el contenedor, con ello evitar desbordamientos, a continuación como se puede observar en la **Tabla *27***, tenemos dos tipos de dispositivos de alerta sonora cada uno con sus propias características propias, finalmente se elegirá al que mejor se adapte para el sistema a desarrollar.

Tabla 27

Tabla comparativa de dispositivo de alerta sonora

Driver de control para motor		
Nombre Comercial	Buzzer activo	Buzzer pasivo
Imagen		
Accesibilidad	Alta	Alta
Voltaje de operación	4V -8V DC	4V -8V DC
Consumo de corriente	30mA	30mA
Nivel de presión sonora	85dB a 10cm	85dB a 10cm
Oscilador interno	SI	NO
Tipo de señal	Digital	PWM
Frecuencia resonante	2300+-300Hz	2300+-300Hz
Control de frecuencia	NO	SI
Dimensión/tamaño	12mm	12 mm
Compatibilidad	Arduino Esp32 Raspberry	Arduino Esp32 Raspberry
Uso típico	Alarmas simples	Sistemas programables
Costo	$1	$1

Nota. En la siguiente tabla podemos observar dos tipos de dispositivos de alerta sonora presentes en el mercado actual, cada uno muestra sus características técnicas, obtenida de (Novatronic, 2020) & (Novatronic, 2020)

- **Componente seleccionado**

En base al análisis de cada una de las características de cada dispositivo, el dispositivo elegido es el **buzzer activo** ya que a pesar de que ambos cumplen con similares características y tienen el mismo costo, el buzzer activo cuenta con un oscilador interno el cual genera automáticamente una señal, este tipo de señal es digital es decir ON/OFF, a comparación del otro buzzer pasivo el cual puede variar el tono de la señal , por todas estas características el que mejor se adapta correctamente al sistema es el buzzer activo, además otro punto importante a considerar es que la única función de este dispositivo será alertar al cotornicultor durante el llenado de los contenedores por lo que no es necesario variar el tono de la alarma sonora.

Finalmente mencionar que cumple con los requerimientos de usuario SR8 presente en la **Tabla *17***, también los requerimientos de sistema SyR12 presentes en la **Tabla *18***, y por último los requerimientos de arquitectura AR4 presente en la **Tabla *19***.

- ***Consumo eléctrico del sistema***

La fuente de alimentación es de mucha importancia para el correcto funcionamiento del sistema, por lo que una selección inadecuada puede provocar reinicios del microcontrolador o causar daños al microcontrolador o a los componentes conectados al sistema, es por esta razón que para su elección lo primero que se hará es realizar un análisis previo de acuerdo a todos los componentes que conformaran el sistema considerando su voltaje de operación y el consumo de corriente máximo de cada componente, cabe mencionar que el sistema se divide en dos : el sistema de alimentación donde estará el contenedor de alimento y la banda transportadora, la otra parte estará conformada por el contenedor de agua y bebedero además del contador de huevos, esto nos permitirá un mejor control del consumo energético, además de garantizar un funcionamiento estable y prolongar la vida útil de los componentes y por ende del sistema.

- **Subsistema de dispensador de alimento**

Antes de seleccionar la fuente de alimentación ideal para nuestro sistema del comedero vamos a realizar un análisis de voltaje de operación y corriente máxima de los componentes seleccionados anteriormente, una vez que realicemos esto podremos ver el consumo eléctrico total y elegir una fuente de alimentación que satisfaga con la alimentación del sistema, esto lo podemos observar en la **Tabla *28***.

Tabla 28

Consumo eléctrico de componentes que integran el sistema del comedero

Cantidad	Nombre del dispositivo	Voltaje de operación	Corriente máxima
1	Microcontrolador ESP32 DevKit V1	5V DC	260mA
1	Sensor de distancia ultrasónico HC-SR04	5V DC	15mA
1	Buzzer activo	4V -8V DC	30mA
1	Servo motor SG90 9g 180°	4V-7.2V DC	700mA
2	Motores DC	3V-9V DC	600mA
1	Puente H L298N	5V	0.36mA

Fuente: Autoría propia

De acuerdo a la tabla y sus valores de voltaje de operación y corriente máxima podemos definir la corriente máxima de sistema debido a que cada componente estará conectado en paralelo el voltaje de alimentación es el mismo para todos los dispositivos, mientras que la corriente total que se necesita resulta de la suma de todas las corrientes de cada dispositivo, para lo cual nos basamos en el principio básico de circuitos eléctricos.

Corriente total = 260mA+15mA+30mA+700mA+2(600mA) +0.36mA

Corriente total=2205mA o 2.205A

Como podemos observar tenemos una corriente total de 2205mA lo que equivale a 2.205 Amperios cuando todos los componentes del sistema trabajen simultáneamente bajo condiciones de máxima carga.

De acuerdo a estos resultados obtenidos podemos mencionar el uso de una sola única fuente de alimentación podría ser inadecuado debido a la presencia de motores y servomotor lo que podría afectar al funcionamiento del sistema, es por ello que se opta por hacer uso de tres adaptadores de corriente independientes, el primer adaptador será de 5V a 1 A con la capacidad suficiente para alimentar el microcontrolador ESP32, Sensor de distancia ultrasónico HC-SR04 y buzzer , el segundo adaptador alimentará de forma directa al puente h L298N y por ende será el encargado de alimentar los motores de la banda transportadora para ello usaremos un adaptador de corriente de 5V a 2 A de igual forma de acuerdo al consumo de corriente este adaptador tendrá la capacidad suficiente para alimentar estos componentes, por último se usará otro adaptador de 5 voltios a 2 A para la alimentación del servomotor, con esta distribución aseguraremos un funcionamiento estable y evitaremos problemas a la hora del funcionamiento.

- **Subsistema de dispensador de agua**

Antes de seleccionar la fuente de alimentación ideal para nuestro subsistema de dispensador de agua vamos a realizar un análisis de voltaje de operación y corriente máxima de los componentes seleccionados anteriormente, una vez que realicemos esto podremos ver el consumo eléctrico total y elegir una fuente de alimentación que satisfaga con la alimentación del sistema, esto lo podemos observar en la **Tabla *28***.

Tabla 29

Consumo eléctrico de componentes que integran el sistema del comedero

Cantidad	Nombre del dispositivo	Voltaje de operación	Corriente máxima
1	Microcontrolador ESP32 DevKit V1	5V DC	260mA

2	Sensor de distancia ultrasónico HC-SR04	5V DC	15mA
1	Buzzer activo	4V -8V DC	30mA
2	Módulo de relé 5V	5V DC	20mA
2	Mini Bomba de agua sumergible 5V	2.5V-6V DC	220mA

Fuente: Autoría propia

De acuerdo a la tabla y sus valores de voltaje de operación y corriente máxima podemos definir la corriente máxima de sistema debido a que cada componente estará conectado en paralelo el voltaje de alimentación es el mismo para todos los dispositivos, mientras que la corriente total que se necesita resulta de la suma de todas las corrientes de cada dispositivo, para lo cual nos basamos en el principio básico de circuitos eléctricos.

Corriente total = 260mA+2(15mA)+30mA+2(20mA)+2(220mA)

Corriente total=800mA o 0.8A

Como podemos observar tenemos una corriente total de 800mA lo que equivale a 0.8 Amperios cuando todos los componentes del sistema trabajen simultáneamente bajo condiciones de máxima carga.

De acuerdo a estos resultados obtenidos podemos mencionar el uso de una sola única fuente de alimentación podría funcionar sin problemas pero debido a una mejor distribución se opta por hacer uso de dos adaptadores de corriente independientes, el primer adaptador será de 5V a 1 A con la capacidad suficiente para alimentar el microcontrolador ESP32, los dos sensores de distancia ultrasónico HC-SR04 y buzzer , el segundo adaptador alimentará los relé y por ende las bombas de agua ya que estas bombas para su funcionamiento tendrán una fuente externa mediante un adaptador de corriente de 5V a 2 A, con esta distribución aseguraremos un funcionamiento estable y evitaremos problemas a la hora del funcionamiento.

- **Selección de batería de respaldo**

Para elegir una batería adecuada de respaldo se realizó el siguiente cálculo:

Tomamos la corriente máxima del subsistema de dispensador de alimento y la corriente máxima obtenida del subsistema del dispensador de agua el cual es el siguiente:

- Corriente Máxima subsistema de dispensador de alimento: 2205mA
- Corriente Máxima subsistema de dispensador de agua: 800mA

Por lo que la corriente total máxima del sistema corresponde a la suma de ambas corrientes, el cual da como resultado: 3.005 A.

La potencia de la batería se lo obtiene con la fórmula: P= V x I = 5x3.005=15.025W.

Con esto podemos evidenciar que la batería de 500 VA, satisface con el funcionamiento del sistema, ya que el UPS seleccionado cuenta con una potencia real máxima de 300 W, la cual supera ampliamente la potencia requerida por el sistema que es de 15.025W.

Además de acuerdo a su referencia de batería de 12V/7Ah y su eficiencia del 65% se realiza el siguiente cálculo:

Tiempo de respaldo (UPS) = (Capacidad de batería*voltaje de batería *Eficiencia) /Consumo total(W)

Reemplazando sus valores tenemos:

Tiempo de respaldo (UPS) = (9x12x0.65) /15.025

Tiempo de respaldo (UPS) =4.67 horas

Con este resultado tenemos que el tiempo de respaldo aproximado es de 4 a 5 horas aproximadamente con el sistema funcionando a su máxima capacidad, pero como la parte de los motores y bombas se activan por unos pocos segundos durante el día el respaldo se extenderá, además pen caso de requerir más tiempo de respaldo se debe adaptar una batería con mayor capacidad.

3.4.2 Etapa de red y comunicación

Esta etapa cumple un papel muy importante dentro de nuestro sistema de control de alimento debido a que nos permite la transmisión de la información recopilada por los diferentes sensores que estarán ubicados en diferentes puntos dentro de la jaula, cabe mencionar que el encargado de realizar todo este proceso será el microcontrolador ESP32 DevKit V1 seleccionado anteriormente, mismo que cumplirá funciones como establecer la conexión WiFi mediante un router como nodo Gateway quien actuará como punto de interconexión entre la red local inalámbrica e internet, además facilitará el acceso a servicios en la nube y la transmisión de los datos hacia una plataforma de visualización,

3.4.3 Etapa de servicios y almacenamiento de datos

Esta etapa nos permitirá el almacenamiento de datos generados por los sensores implementados en el sistema como son los sensores de distancia ultrasónico HC-SR04 mismos que estarán en el interior del tanque tanto de alimento como de agua para la medición del estado de nivel, además del sensor infrarrojo FC51 el cual nos ayudará con el conteo de los huevos, gracias a esta etapa se podrá consultar y visualizar estos datos de manera remota.

Para ello los datos deberán ser almacenados en algún servicio en la nube por lo que es necesario contar con uno que cuente con una base de datos para que se registren variables como nivel de alimento, nivel de agua, conteo de huevos etc. También esta etapa nos permitirá mantener un historial de los datos, siendo muy importante para la siguiente etapa la cual hacer referencia a la etapa de aplicación y visualización.

Finamente se elegirá el más idóneo para integrarlo en el sistema y a su vez escoger el que nos permita realizar todo lo mencionado anteriormente, a continuación, podemos ver en la, **Tabla *30*** ,3 tipos de plataformas virtuales en la nube más utilizados en la actualidad como son Firebase, Microsoft Azure IoT Hub y AWS IoT Core, finalmente elegiremos el que mejor se adapte al sistema a desarrollar.

Tabla 30

Tabla comparativa de plataforma virtuales en la nube

Plataformas virtuales en la nube			
Nombre Comercial	Firebase	AWS IoT Core	Microsoft Azure IoT Hub
Imagen			
Tipo	BaaS (Backend as a service)	PaaS(Platform as a Service)	PaaS(Platform as a Service)
Protocolos soportados	HTTP, HTTPS, WebSockets	MQTT, MQTT over WebSockets, HTTPS	MQTT, AMQP, HTTPS
Datos en tiempo real	SI	SI	SI
Almacenamiento de datos	NoSQL(Realtime Database, Firestore)	Integracion con BD NoSQL/SQL externas	Integración con BD NoSQL/SQL externas
Hosting /aplicaciones web	Si, (Firebase hosting)	No, requiere servicios adicionales	No, requiere servicios adicionales
Seguridad	Reglas de acceso y autenticación	Certificados X509, políticas	Autenticación y control empresarial
Acceso multiplataforma	Si	Si	Si
Costo	Sin costo dentro del plan gratuito, mientras no se superen los límites	Plan gratuito limitado	Plan gratuito limitado

Nota. En la siguiente tabla podemos observar tres tipos de plataformas virtuales en la nube, cada uno muestra sus características, obtenida de (Amazon Web Services, 2025) & (Firebase, 2025) & (Microsoft, 2025)

- ***Plataforma seleccionada***

En base al análisis realizado a las características de las plataformas virtuales en la nube se elige la plataforma **Firebase** debido a que cumple con los requerimientos necesarios para el desarrollo del sistema, esta plataforma es de tipo BaaS siendo una ventaja ya que la plataforma es la que se encarga de gestionar los servidores y la infraestructura, mientras que en AWS IoT Core y Azufe IoT Hub son de tipo PaaS por lo que requieren configuración y es más compleja, otro punto importante que se considero es que firebase cuenta con un soporte de base de datos en tiempo real lo que permite la sincronización en tiempo real de los datos que generen los sensores implementados en el sistema de control de alimento, a diferencia de los otras dos plataformas que requieren integrar servicios adicionales, otro punto importante que se consideró es que firebase cuenta con Firebase Hosting el cual será de mucha importancia ya que nos ayudará para la creación del dashboard, siendo un punto a favor ya que a comparación de las otras plataformas no ofrecen esto y se requiere el uso de componentes externos para crear el dashboard, además en cuanto al costo esta plataforma ofrece un plan gratuito mientras no se supere el límite de uso mientras que las otras plataformas tienen un plan limitado.

Finalmente mencionar que cumple con los requerimientos de usuario SR4,SR10 y SR11 presente en la **Tabla *17***, también los requerimientos de sistema SyR1, SyR2, SyR3, SyR4, SyR5, SyR6, SyR7, SyR8, SyR11 y SyR14 presentes en la **Tabla *18***, y por último los requerimientos de arquitectura AR5, AR8, AR10 y AR14 presente en la **Tabla *19***.

3.4.4 Etapa de aplicación y visualización

En esta etapa podemos ver que su objetivo principal es convertir los obtenidos por los sensores presentes en lugares estratégicos en la jaula en una fácil interpretación mediante el uso de interfaces gráficas desde cualquier dispositivo o punto que cuente con acceso a internet, a través ello el coturnicultor podrá interactuar con el sistema ya que podrá visualizar el estado

de los contenderos de alimento y agua, además de tener un control adecuado en cuanto a la producción de huevos, esto mediante los datos previamente almacenado en la base de datos de la nube, por lo que se hará uso de **Firebase hosting** una característica de la plataforma Firebase ya que como se puede apreciar en la **Tabla *30***, este nos permitirá alojar y publicar un dashboard web facilitando el acceso remoto desde cualquier navegador.

Además, en esta etapa se incorpora la parte de servicios de mensajería el cual será de mucha importancia para recibir notificaciones ante situaciones relevantes durante el funcionamiento del sistema de control de alimentación estas situaciones comprenden como recibir notificaciones ante estados críticos referentes al estado de llenado de contenedor, o en la dispensación del alimento o agua.

Es por ello que para la elección de la mejor plataforma de mensajería se hará una comparación entre las distintas plataformas que existen actualmente como se puede observar en la **Tabla *31*** , de acuerdo a esto elegirá al más idóneo para su integración en el sistema de control de alimentación, con esto garantizaremos una alerta mediante notificación en tiempo real sin depender solo de la interfaz de gráfica.

Tabla 31

Tabla comparativa de plataformas de mensajería

Plataformas de mensajería		
Nombre Comercial	WhatsApp	Telegram
Imagen		
Tipo de plataforma	Servicio de mensajería	Servicio de mensajería
Comunicación en tiempo real	Sí	Sí
Chats grupales	Sí, con limite	Sí, grupos grandes

Automatización mediante bots	No nativo	Sí, soporte nativo
Acceso multiplataforma	Si	Sí
Requisitos para su uso	Conexión a internet Número telefónico	Conexión a internet Número telefónico
Requisitos para integración con sistemas	Integración limitada, requiere servicios externos	API para bots y servicios de automatización
Costo	Gratuito	Gratuito

Nota. En la siguiente tabla podemos observar dos tipos de plataformas de mensajería, cada uno muestra sus características, obtenida de (GoDaddy, 2024) & (RD Station Marketing, 2022)

- ***Plataforma seleccionada***

Una vez realizada la comparación entre ambas plataformas de mensajería se opta por la plataforma de Telegram ya que esta tiene soporte nativo para la automatización mediante bots lo que nos ayuda a la implementación de notificaciones automáticas sin necesidad de servicios externos. Además de que en cuanto a los requisitos para la integración con sistemas este cuenta con API para bots y servicios de automatización, aunque ambos son gratuitos para usuarios finales esta plataforma cumple con los requisitos requeridos para nuestro sistema de notificaciones por medio de una plataforma de mensajería.

Finalmente mencionar que cumple con los requerimientos de usuario SR10 y SR12 presente en la **Tabla *17***, también los requerimientos de sistema SyR10 y SyR12 presentes en la **Tabla *18***, y por último el requerimiento de arquitectura AR9 presente en la **Tabla *19***.

3.5 Diseño del Sistema

Como segunda fase de la metodología en cascada propuesta para cumplir con los objetivos del desarrollo del sistema tenemos la parte de diseño del sistema donde mediante los componentes seleccionados en la fase uno, se mostrará un diagrama de bloques lo que nos permitirá conocer de una forma más fácil las relaciones funcionales y su interacción entre sus

principales componentes pertenecientes al sistema, seguido de esto podremos ver la arquitectura del sistema propuesto, donde mostrará la forma de interconexión de los componentes seleccionados entre sí, es decir mostrará la estructura general del sistema.

Finalmente tendremos la parte del flujograma donde podremos ver el comportamiento lógico del sistema de una manera secuencial de acuerdo a las decisiones y acciones realizadas en función a las lecturas de los sensores y las condiciones establecidas, cabe mencionar que este flujograma nos servirá como una guía a la hora de programar el sistema.

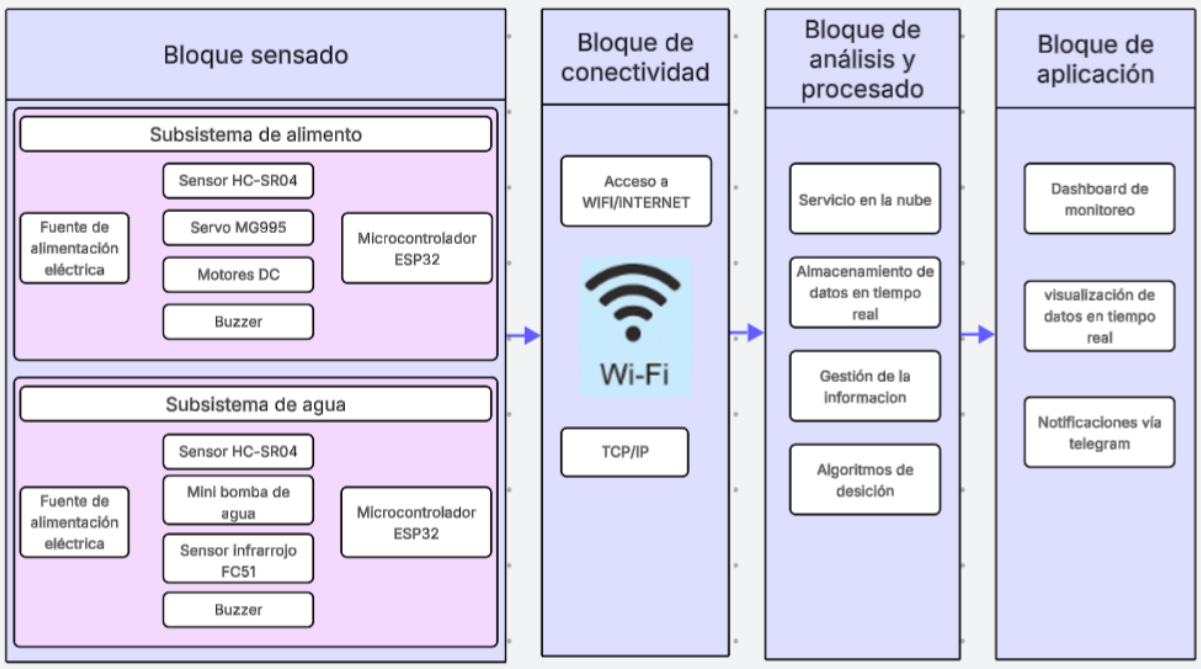
3.5.1 Diagrama de Bloques

En este apartado en la **Figura *14***, podemos observar el diagrama de bloques el cual nos da una visión clara de cómo interactúan las diferentes etapas que lo conforman, El desarrollo de este diagrama nos permite ver el funcionamiento del sistema de control de alimento para codornices desde la captura de los datos por parte de los sensores hasta la visualización final de los datos por parte del coturnicultor.

Este sistema en sí, se basa en una arquitectura IoT de 4 capas, abarca desde la capa de sensado, seguido de la capa de conectividad, luego por la capa de análisis y procesado, finalmente la capa de aplicación.

Figura 14

Diagrama de bloques basado en la arquitectura IoT de 4 capas



Fuente: Autoría propia

A continuación, tenemos los bloques que conforman el sistema:

3.5.1.1 Bloque sensado

Como primera etapa tenemos el bloque de sensado el cual integra componentes encargados de los procesos de toma de datos de los sensores y actuación involucradas tanto en el proceso de alimentación como en el suministro de agua y conteo de huevos, En esta etapa se hace uso de sensores de distancia los cuales nos servirán para la medición del nivel tanto de alimento como de agua, además de un sensor infrarrojo que nos ayudará con el conteo de huevos, todos los datos recopilados serán enviadas al microcontrolador ESP32 Dev KitV1 el cual está establecido como unidad central y será el encargado del procesamiento lógico de los datos y ejecutar acciones automáticas del sistema.

También mencionar que este bloque abarca la parte de actuación donde se integra componentes como motores, mini bombas de agua, buzzer, los cuales permitirán ejecutar acciones automáticas de acuerdo a los datos obtenidos por los sensores.

Finalmente, este bloque será el encargado mediante el microcontrolador de la transmisión de información hacia los siguientes bloques del sistema.

3.5.1.2 Bloque de conectividad

En este bloque tenemos el establecimiento de comunicación inalámbrica de nuestro sistema donde tenemos la transmisión de los datos obtenidos por los sensores presentes en el bloque de sensado, esta conexión se realiza mediante el módulo WiFi el cual viene integrado en el microcontrolador escogido, el cual se enlaza a la red inalámbrica local del lugar donde se implementará el sistema de control de alimento. Además, el router cumplirá como Gateway ya que facilita la interconexión entre la red local del sistema y la red externa con acceso a internet, de esta manera podemos pasar al siguiente bloque del sistema.

3.5.1.3 Bloque de análisis y procesado

En este bloque tenemos la parte de almacenamiento de datos donde usaremos la base de datos en tiempo real Firebase realtime database donde se almacenan los datos provenientes del bloque de sensado, este bloque facilita el manejo de los datos los cuales están relacionados como nivel de alimento y agua, conteo de huevos con el fin de que siempre estén disponibles, es decir este bloque actuará como un puente entre la adquisición de datos de los sensores y la presentación final al coturnicutor.

3.5.1.4 Bloque de aplicación

Este bloque cuenta con un dashboard web el cual permitirá al coturnicultor monitorear el estado de los contenedores de alimento y agua mediante el uso de los datos almacenados en la base de datos, además de ver la cantidad de huevos disponibles en tiempo real, este dashboard

está desplegada mediante firebase hosting, lo que facilita al coturnicultor al acceso a la página desde cualquier sitio con conexión a internet.

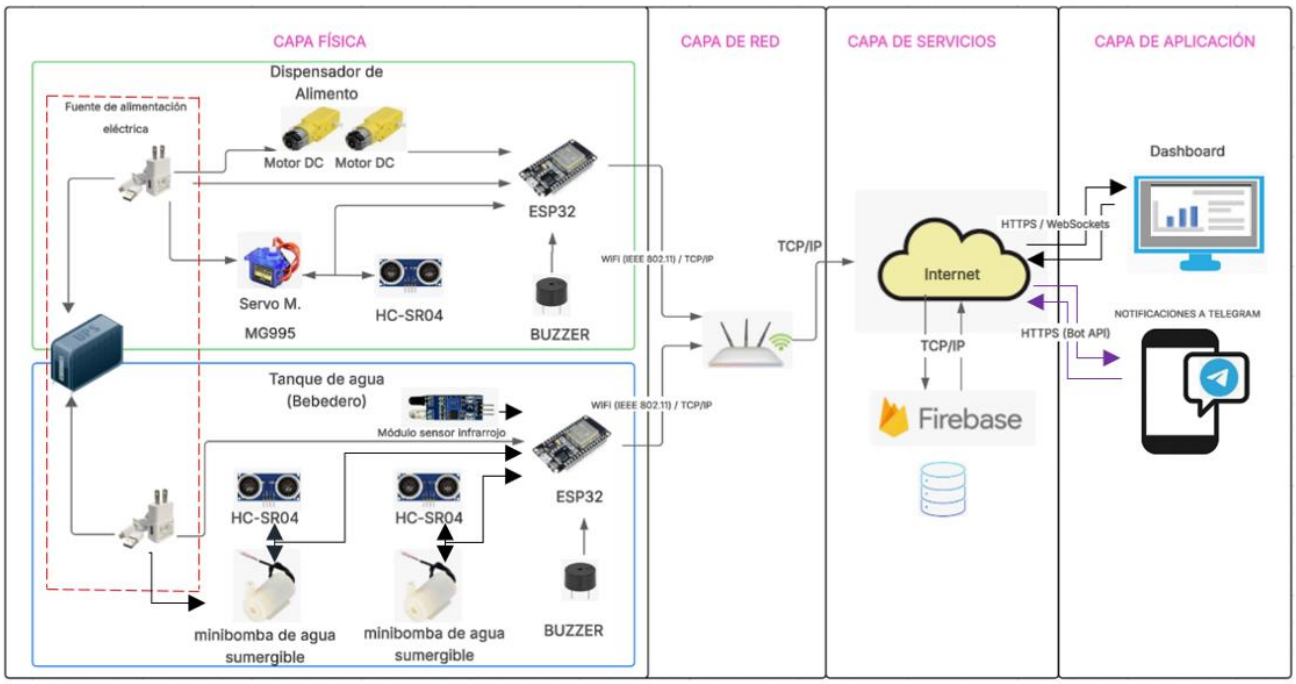
Finalmente mencionar que en este bloque tenemos la parte de notificaciones a través de Telegram, con el objetivo de notificar al coturnicultor sobre situaciones importantes durante la ejecución del sistema.

3.5.2 Arquitectura

En este apartado tenemos la parte de la arquitectura de sistema, esta arquitectura se basa en cuatro capas la cuales son capa física, capa de red, capa de servicios y capa aplicación, gracias a esto tenemos una mejor organización y funcionamiento de los diferentes componentes que conforman el sistema, esta arquitectura facilitará la comprensión de la estructura del sistema, esto lo podemos observar en la **Figura *15***.

Figura 15

Arquitectura del sistema



Fuente: Autoría propia

Como primer capa tenemos la **capa física**, aquí tenemos la integración de dos subsistemas , el primer subsistema será referente al dispensador de alimento en donde el sensor de distancia HC-SR04 ubicado en el interior del tanque de alimento el cual medirá de manera continua el nivel de alimento presente en el contenedor, además en la parte de actuación tenemos un servomotor MG955 a la salida como control de la dosificación de alimento de acuerdo a los requerimientos establecidos, además, los motores DC que serán parte de la banda transportadora para una distribución uniforme del alimento y por último el buzzer el cual será el encargado de las alertas sonoras para el llenado del contenedor de alimento lo cual permitirá una intervención oportuna por parte del coturnicultor. Para el subsistema del dispensador de agua tenemos la presencia de 2 sensores de distancia HC-SR04, donde uno estará ubicado en el interior del tanque de agua, mientras que el otro estará ubicado en el bebedero, estos sensores serán los encargados de medir de manera continua el nivel de agua presente en el tanque de agua y el bebedero, sumado a esto tendremos la implementación de un sensor infrarrojo FC51 el cual será el encargado del conteo de huevos mismo que estará ubicado en el canal de deslizamientos de huevos, además en la parte de actuación tenemos 2 mini bombas de agua sumergibles las cuales serán esenciales para una buenas suministración de agua, estas mini bombas estarán ubicadas tanto en el tanque de agua como en el bebedero, por último el buzzer el cual será el encargado de las alertas sonoras para el llenado del contenedor de agua lo cual permitirá una intervención oportuna por parte del coturnicultor, todos estos componentes presentes en el subsistema de alimentación y de agua ,como sensores y actuadores irán conectados al esp32 Dev Kit V1 para cada subsistema los cuales además de recolectar datos ayudará con el procesamiento lógico y será el encargado de la transmisión de la información hacia la siguiente capa.

Por último tenemos la parte de alimentación eléctrica, para lo cual para el subsistema de alimentación se requieren 2 adaptadores de 5v a 2 A , para el suministro de energía tanto

para el servomotor como para los motores, y otro adaptador de 5v a 1 A , esencial para suministrar de energía al microcontrolador, por otro lado para el subsistema de agua se requiere el uso de 1 adaptador de 5V a 2 A, el cual será el encargado de alimentar las bombas de agua, y otro adaptador de 5V a 1 A el cual será el encargado de alimentar el microcontrolador,

Adicionalmente el uso de una fuente de respaldo nos permitirá mantener la operación del sistema ante posibles interrupciones del suministro eléctrico, asegurando una operación ininterrumpida del sistema.

La **capa de red** será la encargada de la comunicación entre los dispositivos físicos y servicios en la nube, para ello el microcontrolador ESP32 DevKitV1 establece una conexión inalámbrica mediante WiFi local bajo el estándar IEEE 802.11, por medio del uso del protocolo TCP/IP como base para la transmisión de los datos hacia el internet.

A continuación, tenemos la **capa de servicios** donde podemos ver la parte de procesamiento y almacenamiento de los datos, para ello se hace uso de la plataforma Firebase la cual nos permite hacer uso de realtime database una base de datos en tiempo real, aquí se da la comunicación entre el esp32 Dev Kit V1 y Firebase se da a través del protocolo HTTPS.

Finalmente tenemos la **capa de aplicación** donde tenemos la parte de la interacción con el usuario mediante la visualización de un dashboard web en tiempo real, donde el coturnicultor podrá visualizar el estado de los contendores y las cantidad de huevos, así como un calendario de producción, para esta interacción se emplean los protocolos HTTPS y WebSockets, Además el sistema tiene alertas en tiempo real gracias al uso de la plataforma de telegram, mediante el uso de API, el cual opera bajo el protocolo HTTPS.

3.5.3 Diagrama de interconexión del sistema

Una vez determinados los requerimientos necesarios para el desarrollo del sistema se procede con la representación gráfica donde se muestran las conexiones tanto físicas y eléctricas del sistema, cabe mencionar que el sistema se divide en dos, un subsistema para el dispensador de alimento, y el otro para el dispensador de agua, aquí podremos ver como los componentes como los sensores, motores, bombas y demás dispositivos se interconectan con el microcontrolador que ha sido seleccionado previamente, una vez que tengamos el diagrama de interconexión nos servirá como montaje del sistema y nos permitirá replicar de una manera más confiable con los componentes reales.

3.5.3.1 Dispensador de alimento

En este apartado tenemos la parte de dispensador de alimento el cual será el encargado de dosificar la cantidad correcta de alimento de acuerdo al número de aves presentes en la jaula, además de distribuir el alimento de manera uniforme para todas ellas, además este subsistema contará con una alarma sonora para la precaución del coturnicultor a la hora de abastecimiento del contenedor de alimento y evitar desbordamientos de alimento.

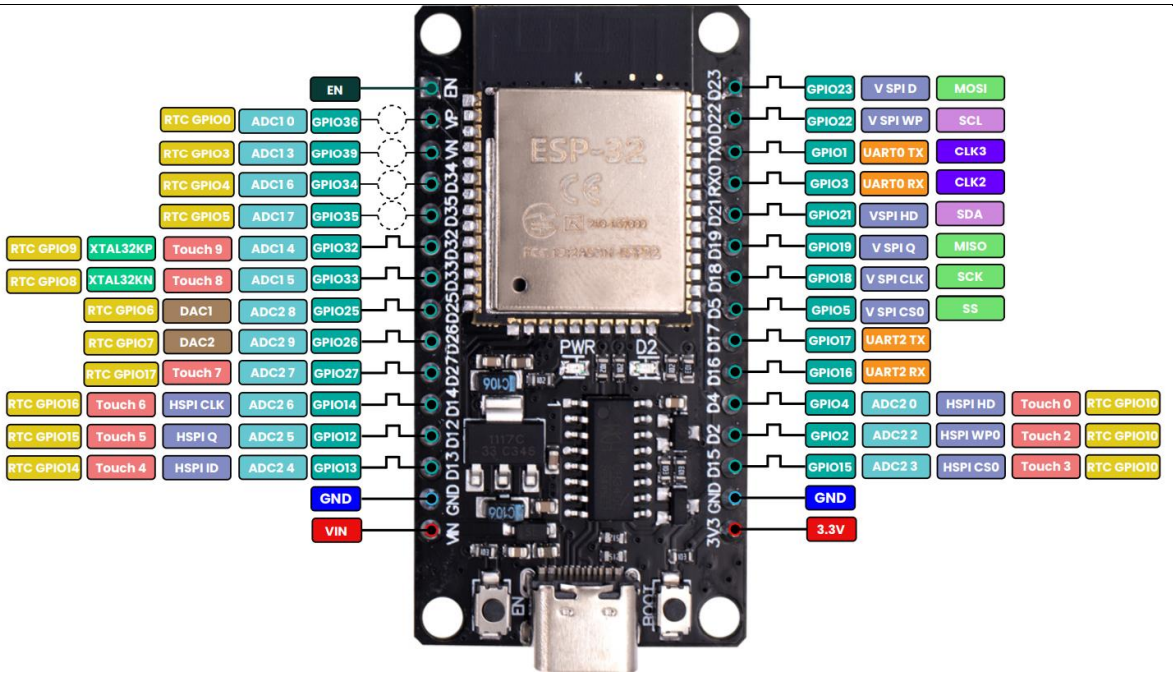
Dicho esto, en la **Figura *16***, podemos ver la interconexión de todos los sensores y actuadores que se integran al microcontrolador, para ello nos basaremos en la

Tabla 32, donde podemos ver los pines disponibles de acuerdo al datasheet de cada uno de los componentes usados en el diseño de este subsistema, para ver de una forma más detalla las características técnicas de cada componente podemos observar en el **Anexo B**.

Tabla 32

Identificación de pines de componentes que conforman el dispensador de alimento de acuerdo a su datasheet

Datasheet
Microcontrolador ESP32 DEVKIT V1

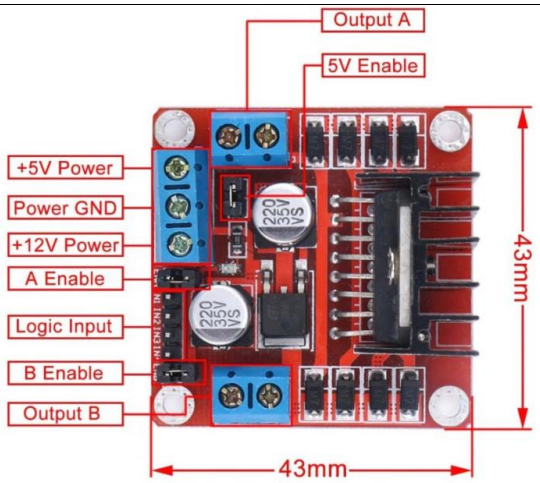
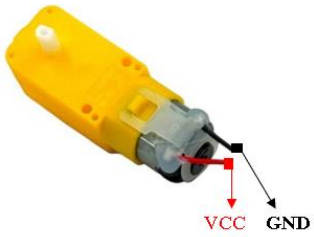


Sensor de distancia HC-SR04

VCC-Alimentación 5V
TRIG-Señal de entrada
ECHO-Señal de salida
GND-Conexión a tierra

Servomotor SG90 180°

VCC- alimentación 5V
SEÑAL-PWM
GND-Conexión a tierra

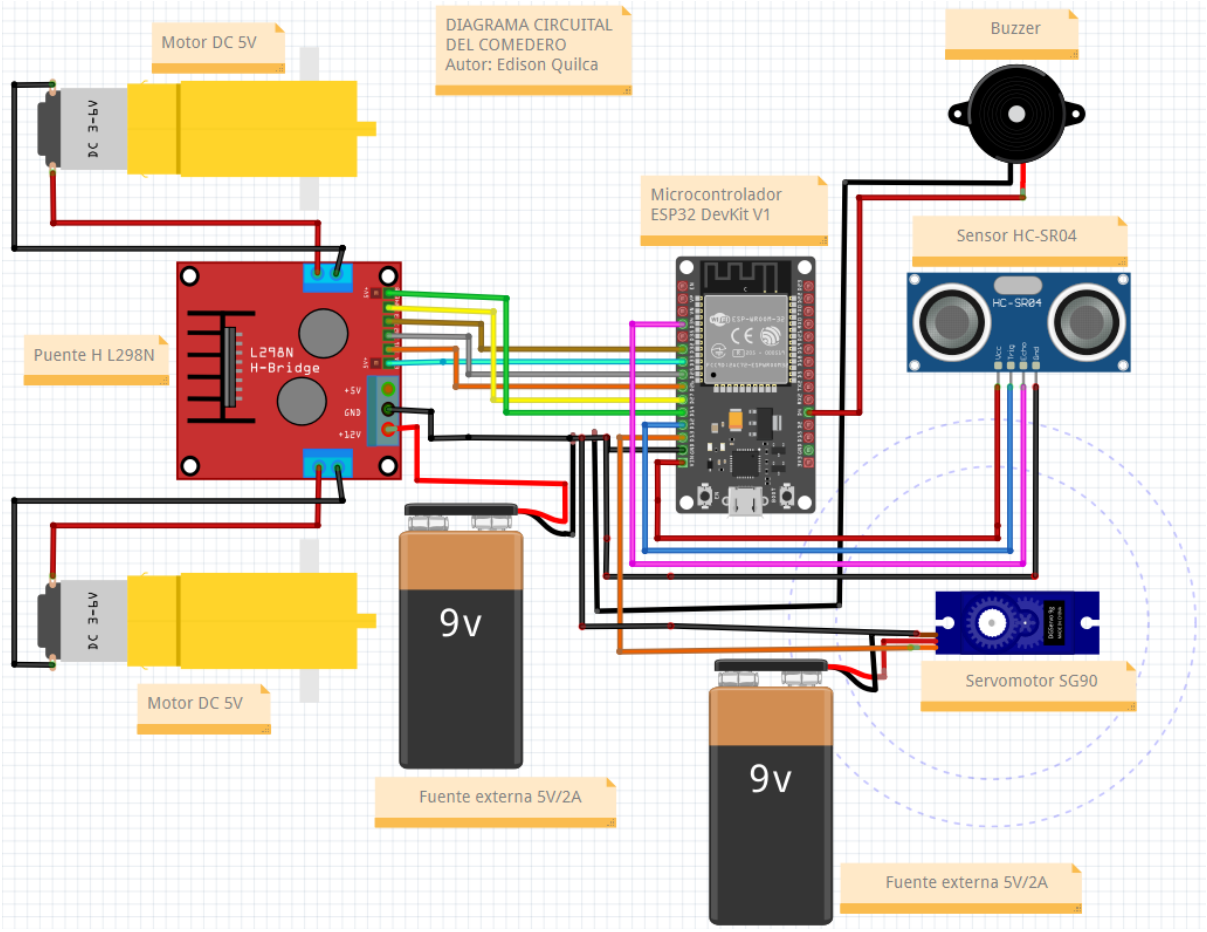
Buzzer activo
 VCC-alimentación 5V GND-Conexión a tierra
Driver controlador de motor L298N

Motor DC amarillo


Nota. *En la siguiente tabla podemos observar los pines de todos los componentes que conforman el subsistema del dispensador de alimento, obtenida de (uElectronics, 2023)* &

(SparkFun Electronics, 2015) & (FriendlyWires, 2023) & (BolanosDJ, 2018) & (Components101, 2021)

Figura 16

Diagrama de interconexión del dispensador de alimento



Nota: Autoría propia

A continuación, podemos observar la interconexión de los componentes del subsistema de alimento con el microcontrolador ESP32 DevKit V1 y alimentación eléctrica:

a) Interconexión de sensor de distancia HC-SR04

-El pin VCC del sensor HC-SR04 conecta al pin 5V del microcontrolador.

-El pin TRIG del sensor HC-SR04 conecta al pin GPIO D12del microcontrolador.

-El pin ECHO del sensor HC-SR04 conecta al pin GPIO D34 del microcontrolador.

-El pin GND del sensor HC-SR04 conecta al pin GND del microcontrolador

b) Interconexión del servomotor SG90 180°

-El pin VCC del servomotor conecta al VCC de la fuente externa (Adaptador de corriente 5V/2 A).

-El pin de la señal del servomotor conecta al pin GPIO D13 del microcontrolador.

-El pin GND del servomotor conecta al GND en común entre pin GND del microcontrolador y GND de la fuente externa.

c) Interconexión de buzzer activo

-El VCC del buzzer conecta al pin GPIO D4 del microcontrolador.

-El GND del buzzer conecta al GND en común entre pin GND del microcontrolador y GND de la fuente externa.

d) Interconexión de puente H L298N y motores DC

-El VCC del puente H conecta al VCC de la fuente externa (Adaptador 5V/2 A).

-El GND del puente H conecta al GND en común entre pin GND del microcontrolador y GND de la fuente externa.

-El ENA del puente H conecta al pin GPIO D33 del microcontrolador.

-El ENB del puente H conecta al pin GPIO D14 del microcontrolador.

-El IN 1 del puente H conecta al pin GPIO D26 del microcontrolador.

-El IN 2 del puente H conecta al pin GPIO D25 del microcontrolador.

-El IN 3 del puente H conecta al pin GPIO D32 del microcontrolador.

-El IN 4 del puente H conecta al pin GPIO D27 del microcontrolador.

-El VCC del motor DC conecta a OUT A1 del puente H.

-El GND del motor DC conecta a OUT A2 del puente H.

-El VCC del motor DC conecta a OUT B1 del puente H.

-El GND del motor DC conecta a OUT B2 del puente H.

- ***Diagrama de flujo***

En este apartado podemos ver en la **Figura 17**, la representación lógica y secuencial del subsistema de dispensación de alimento, en este diagrama podemos observar las etapas que ejecuta el sistema desde el inicio hasta la comunicación con la base de datos Firebase Realtime Database, facilitando la comprensión de la lógica de funcionamiento del subsistema.

Como primer punto tenemos la parte de inicialización seguido de la lectura el sensor ultrasónico HC-SR04 el cual estará implementado dentro del contenedor de alimento, el cual nos ayudará con la medición y el cálculo del nivel real de comida dentro del contenedor de alimento, con ello determinar el estado del mismo con estado: lleno, medio, bajo, crítico o vacío.

Una vez realizado este proceso se verifica la disponibilidad de la conexión a WiFi y del servicio Firebase Realtime Database, si ambos servicios se encuentran disponibles los datos obtenidos por el sensor ultrasónico de distancia HC-SR04 se almacenarán en la base de datos RTDB, si en caso de que uno de los dos servicios no está disponible entonces el sistema sigue operando en modo local con el fin de no interrumpir las funciones principales, las cuales son la dispensación de alimento.

De acuerdo al proceso anterior, el sistema ejecutará acciones específicas como, dependiendo del nivel alcanzado dará estados como críticos, vació o lleno, el sistema enviará notificaciones por telegram, evitando el envío de mensajes repetitivos. También cuando el estado del comedero alcance un nivel lleno se activará una alerta sonora mediante a activación del buzzer activo, el cual sonará por 5 segundos, volverá a rearmarse cuando el nivel descienda y vuelva a llenarse.

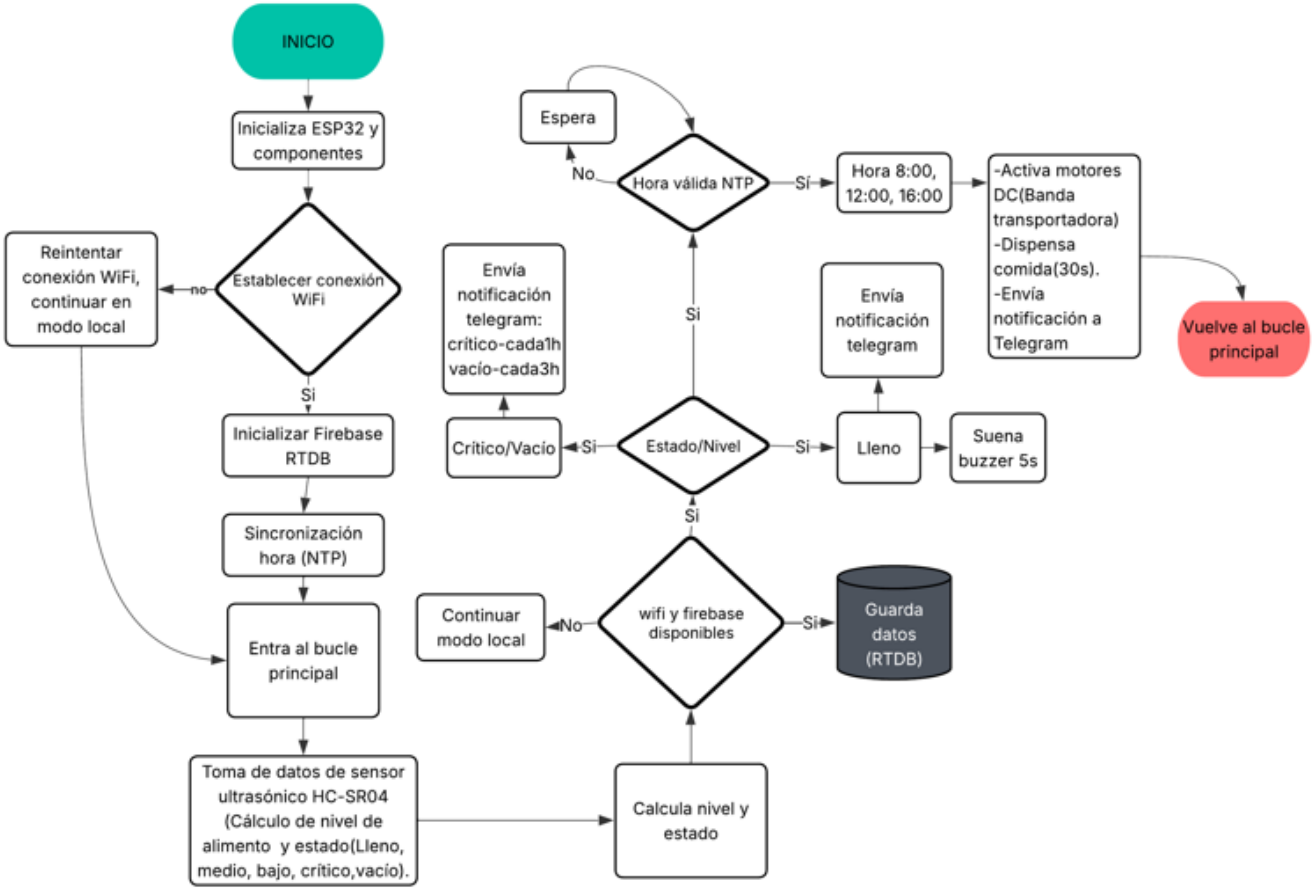
Posteriormente para la parte de la dispensación de alimento se evalúa continuamente la hora obtenida a través del protocolo NTP, ya que si la hora de programación de dispensación corresponde a las 8:00, 12:00 y 16:00 entonces se activará la dispensación de alimento lo que

a su vez controla el servomotor y los motores implementados en la banda transportadora durante un tiempo de 30 segundos, una vez que la dispensación inicia y finaliza enviará una notificación por telegram notificando este proceso.

Como punto final el flujo retorna al bucle principal donde repite el proceso de acuerdo al ciclo de operación propuesto.

Figura 17

Diagrama de flujo del subsistema de dispensador de alimento



Fuente: Autoría propia

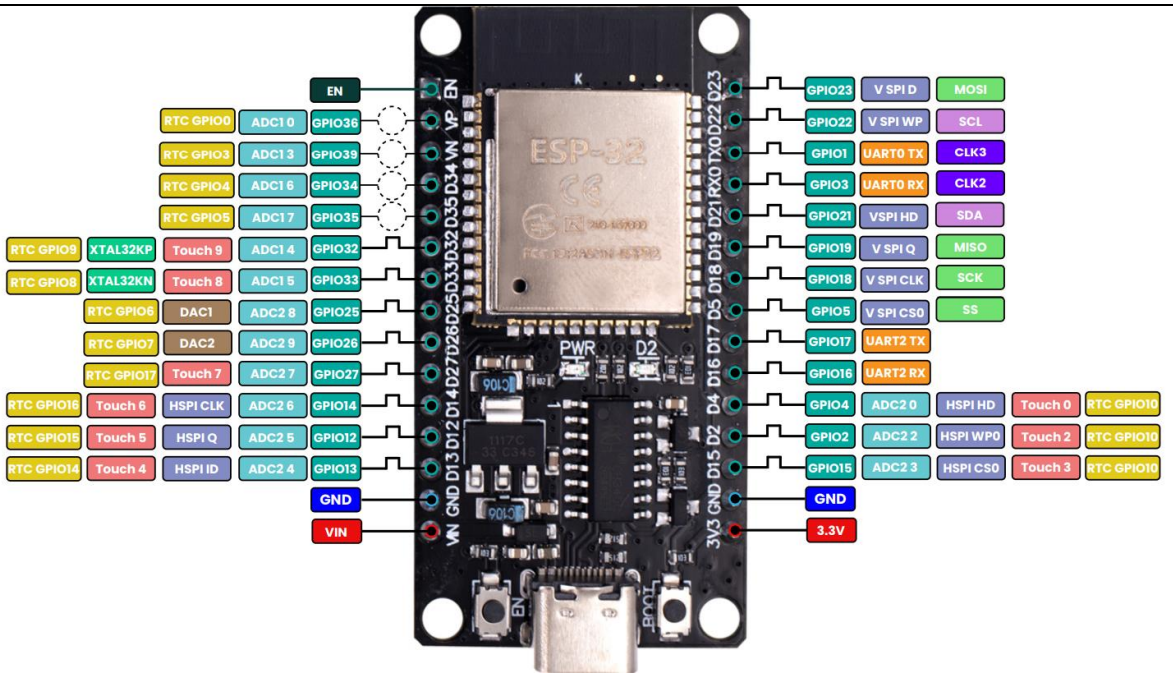
3.5.3.2 Dispensador de agua

En este apartado tenemos la parte del subsistema del dispensador de agua el cual será el encargado de suministrar el agua de manera permanente mientras el tanque de agua contenga agua, este subsistema contará con una alarma sonora para la precaución del coturnicultor a la hora de abastecimiento del contenedor de agua, evitando desbordamientos y posibles daños.

Dicho esto, en la **Figura *18***, podemos ver la interconexión de todos los sensores y actuadores que se integran al microcontrolador, para ello nos basaremos en la **Tabla *33*** , donde podemos ver los pines disponibles de acuerdo al datasheet de cada uno de los componentes usados en el diseño de este subsistema, para ver de una forma más detalla las características técnicas de cada componente podemos observar en el **Anexo B**.

Tabla 33

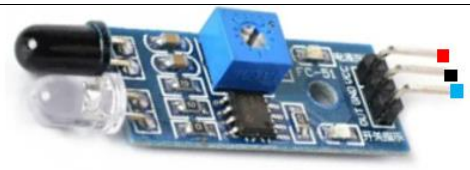
Identificación de pines de componentes que conforman el subsistema del dispensador de agua de acuerdo a su datasheet

Datasheet
Microcontrolador ESP32 DEVKIT V1


Sensor de distancia HC-SR04

VCC-Alimentación 5V
TRIG-Señal de entrada
ECHO-Señal de salida
GND-Conexión a tierra

Sensor infrarrojo FC51

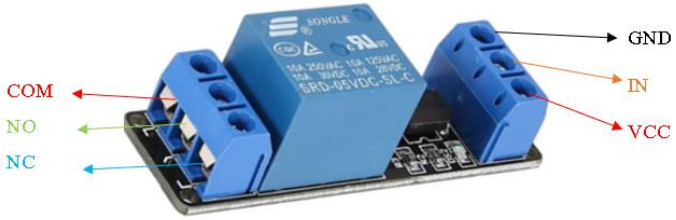


VCC-Alimentación 5V
GND-Conexión a tierra
OUT-Señal digital

Buzzer activo

VCC-alimentación 5V
GND-Conexión a tierra

Módulo relé de 5V

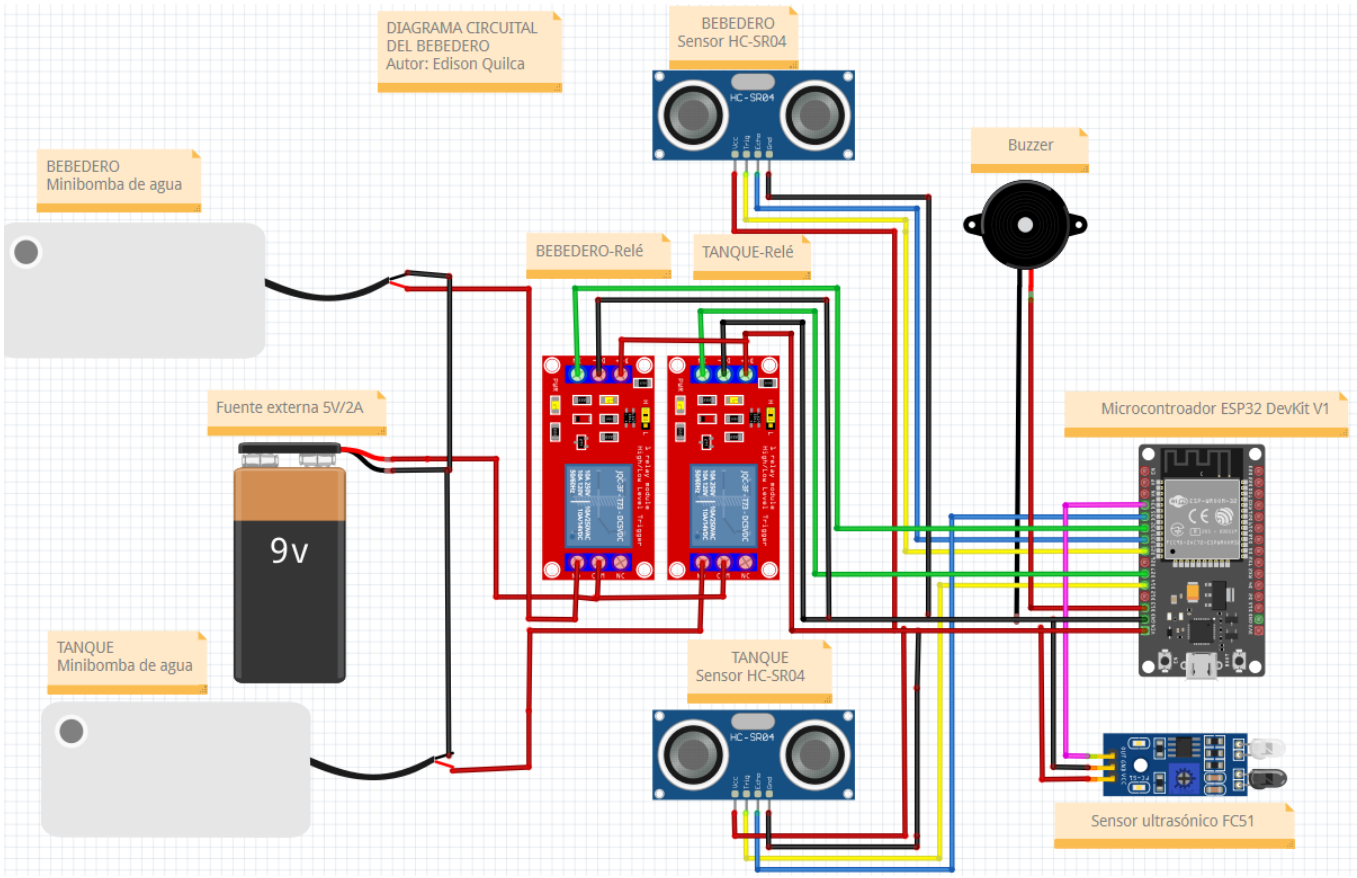


Minibomba de agua sumergible

Nota. *En la siguiente tabla podemos observar los pines de todos los componentes que conforman el subsistema del dispensador de agua, obtenida de (uElectronics, 2023) & (SparkFun Electronics, 2015) & (BolanosDJ, 2018) & (uElectronics, 2018)*

Figura 18

Diagrama de interconexión del subsistema de dispensador de agua



Fuente: Autoría propia

A continuación, podemos observar la interconexión de los componentes del subsistema del dispensador de agua con el microcontrolador ESP32 DevKit V1 y alimentación eléctrica:

a) Interconexión de sensor de distancia HC-SR04 (tanque)

-El pin VCC del sensor HC-SR04 conecta al pin 5V del microcontrolador.

-El pin TRIG del sensor HC-SR04 conecta al pin GPIO D14 del microcontrolador.

-El pin ECHO del sensor HC-SR04 conecta al pin GPIO D35 del microcontrolador.

-El pin GND del sensor HC-SR04 conecta al pin GND del microcontrolador

b) Interconexión de sensor de distancia HC-SR04 (bebedero)

-El pin VCC del sensor HC-SR04 conecta al pin 5V del microcontrolador.

-El pin TRIG del sensor HC-SR04 conecta al pin GPIO D25 del microcontrolador.

-El pin ECHO del sensor HC-SR04 conecta al pin GPIO D33 del microcontrolador.

-El pin GND del sensor HC-SR04 conecta al pin GND del microcontrolador

c) Interconexión de buzzer activo

-El pin VCC del buzzer conecta al pin GPIO D13 del microcontrolador.

-El pin GND del buzzer conecta al pin GND en común entre pin GND del microcontrolador y GND de la fuente externa.

d) Interconexión de módulo relé y mini bomba de agua (tanque)

-El pin VCC de módulo relé del tanque conecta al pin VCC del microcontrolador.

-El pin GND de módulo relé del tanque conecta al pin GND del microcontrolador.

-El pin IN del módulo relé del tanque conecta al pin GPIO D27 del microcontrolador.

-El contacto COM del módulo relé conecta a el cable VCC de la fuente externa de 5V/2 A.

-El contacto NO del módulo relé conecta al cable VCC de la mini bomba de agua sumergible.

-El cable GND de la mini bomba sumergible conecta al cable GND de la fuente externa de 5V/2 A.

e) Interconexión de módulo relé y mini bomba de agua (bebedero)

-El pin VCC de módulo relé del bebedero conecta al pin VCC del microcontrolador.

-El pin GND de módulo relé del bebedero conecta al pin GND del microcontrolador.

-El pin IN del módulo relé del bebedero conecta al pin GPIO D32 del microcontrolador.

-El contacto COM del módulo relé conecta a VCC de la fuente externa de 5V/2 A.

-El contacto NO del módulo relé conecta a VCC de la mini bomba de agua sumergible.

-El cable GND de la mini bomba sumergible conecta a GND de la fuente externa de 5V/2 A.

f) Interconexión de sensor infrarrojo FC51

-El pin VCC del sensor infrarrojo conecta al pin VCC del microcontrolador.

-El pin GND del sensor infrarrojo conecta al pin GND del microcontrolador.

-El pin OUT del sensor infrarrojo conecta al pin GPIO D34 del microcontrolador.

- ***Diagrama de flujo***

A continuación, en la **Figura 19**, tenemos el diagrama de flujo donde podemos ver el funcionamiento lógico del subsistema del dispensador de agua, desde la parte de inicialización de componentes hasta el almacenamiento en la nube.

El proceso da inicio con la inicialización del esp32 y los componentes presentes en el subsistema, una vez inicializado el sistema busca establecer la conexión a WiFi y a la base de datos Realtime Database, de igual modo la sincronización de la hora a través del protocolo NTP, en caso de no lograrse la conectividad el sistema continúa operando en modo local.

Una vez que se completa la fase de inicialización se da paso a la inicialización del bucle principal donde da paso a la toma de datos por parte de los sensores ultrasónicos HC-SR04 los cuales permiten determinar el nivel de agua en el tanque y en el bebedero, de igual modo a la

detección del paso de huevos mediante el uso del sensor infrarrojo FC51 mediante interrupciones por flanco, con el fin de tener una detección precisa.

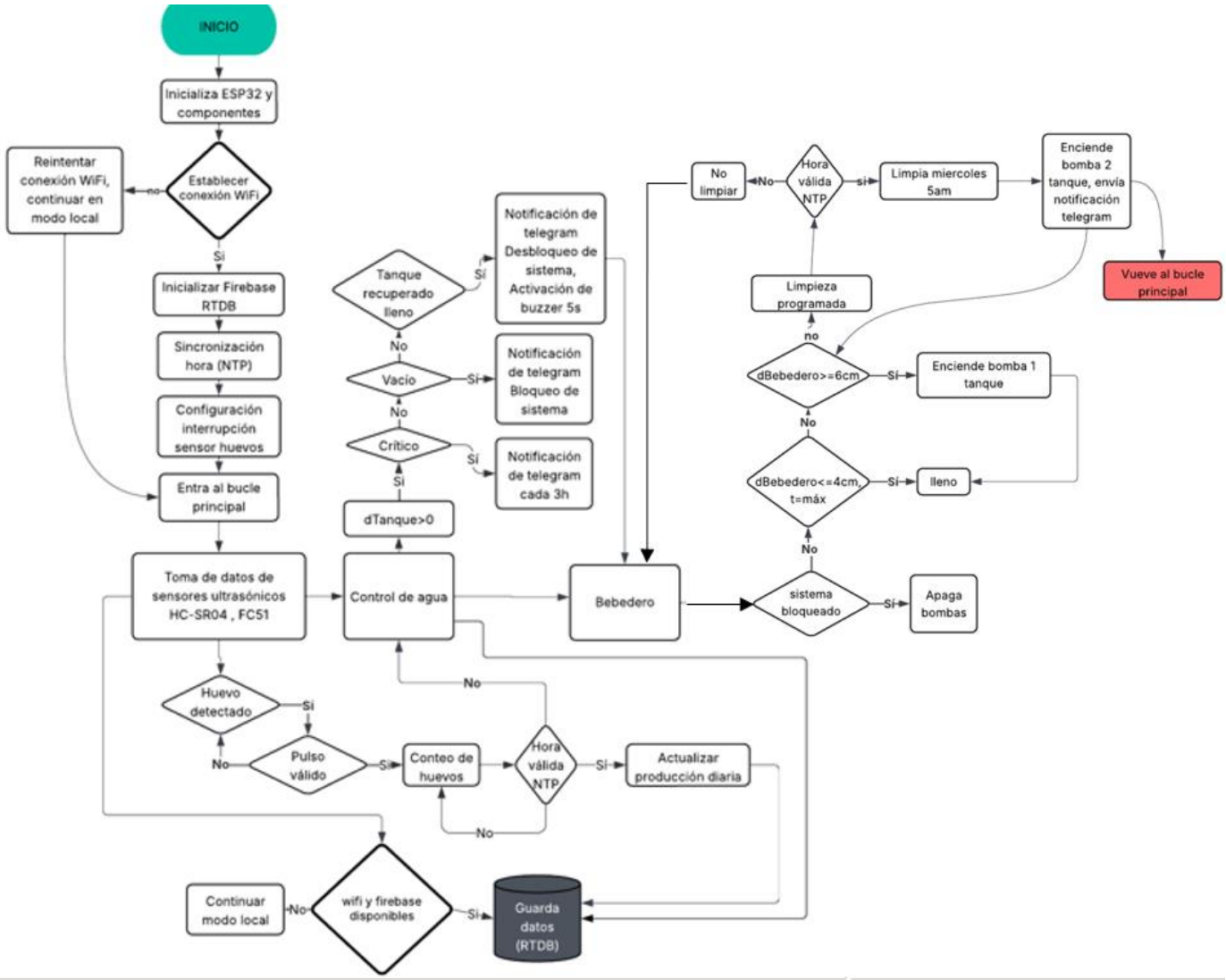
Una vez que se tengan los niveles de agua el sistema ejecuta la lógica de control entre activar o desactivar las bombas presentes tanto en el tanque de agua como en el bebedero para la respectiva limpieza y drenaje, Además se añade una lógica de bloqueo de bombas y notificación a Telegram, si el sistema presenta niveles críticos, esto con el fin de prevenir daños en las bombas de agua

Siguiendo la lógica, cuando el tanque alcance un nivel de lleno, se activa una alerta sonora por medio de un buzzer, además el sistema enviará una notificación del estado de nivel presentado por Telegram.

Por último, el sistema realiza el conteo de huevos y el cálculo de la producción diaria, validando la hora mediante el protocolo NPT, los datos obtenidos por el sensor infrarrojo FC51 se almacenarán en el almacenamiento de datos de Firebase cuando la conectividad esté disponible, caso contrario, el sistema seguirá operando en modo local hasta que la conexión esté disponible.

Figura 19

Diagrama de flujo del subsistema de dispensador de agua



Fuente: Autoría propia

3.6 Desarrollo de software

En esta sección tenemos la parte del desarrollo del software con el fin de realizar una integración entre los dispositivos físicos, servicio en la nube, y plataformas que nos permitirán la visualización y notificación de acuerdo a situaciones críticas del sistema y para alertar al usuario, con el desarrollo de software permitiremos que cada componente cumpla una función específica dentro del funcionamiento general del sistema de control de comida.

3.6.1 Configuración de Firebase

En este apartado se describe la implementación y configuración de la base de datos en la nube, destinada al almacenamiento de los datos generados por los nodos del sistema los cuales se basan en dos microcontroladores, uno para cada subsistema, el primer nodo destinado a la dispensación de alimento y el otro al suministro de agua, para ello se da paso a la creación de una base de datos haciendo uso del servicio Realtime Database de la plataforma Firebase. Además, para el despliegue de estos datos y visualizarlos a través de un dashboard se configura el servicio de firebase hosting, el cual permite alojar la interfaz web del sistema y acceder a los datos obtenidos de forma remota desde cualquier lugar donde tenga acceso a internet.

3.6.1.1 Creación de Base de Datos (RTDB)

Para la creación de la base de datos como primer punto se ingresa a la página oficial de firebase donde con ayuda de un correo electrónico crearemos una cuenta, una vez que tengamos la cuenta ingresamos en consola y creamos un nuevo proyecto al que se le ha denominado *criaderocj* como se puede observar en la **Figura *20***.

Figura 20

Creación de nuevo proyecto criaderocj

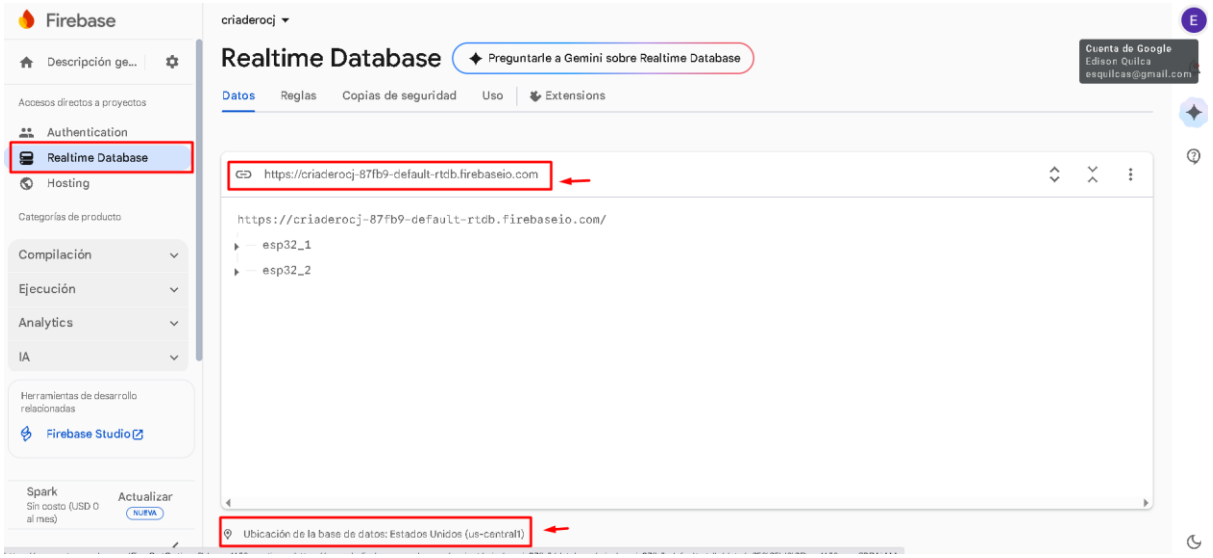
Fuente: Autoría propia

Una vez que se ha creado correctamente el proyecto, accedemos al panel principal donde seleccionaremos el servicio Firebase Realtime Database, esto se puede apreciar en la

Figura *21* , donde crearemos la base de datos, lo que nos permitirá almacenar y sincronizar los datos generados por los nodos del sistema tanto del subsistema de dispensación de alimento como el subsistema de suministro de agua, para su creación escogemos el servidor que más cercano de acuerdo a nuestra posición, en este caso se ha escogido la opción Estados Unidos(us-central1), una vez realizado esto, nos genera el siguiente url de base de datos : *https://criaderocj-87fb9-default-rtdb.firebaseio.com*.

Figura 21

Creación de base de datos

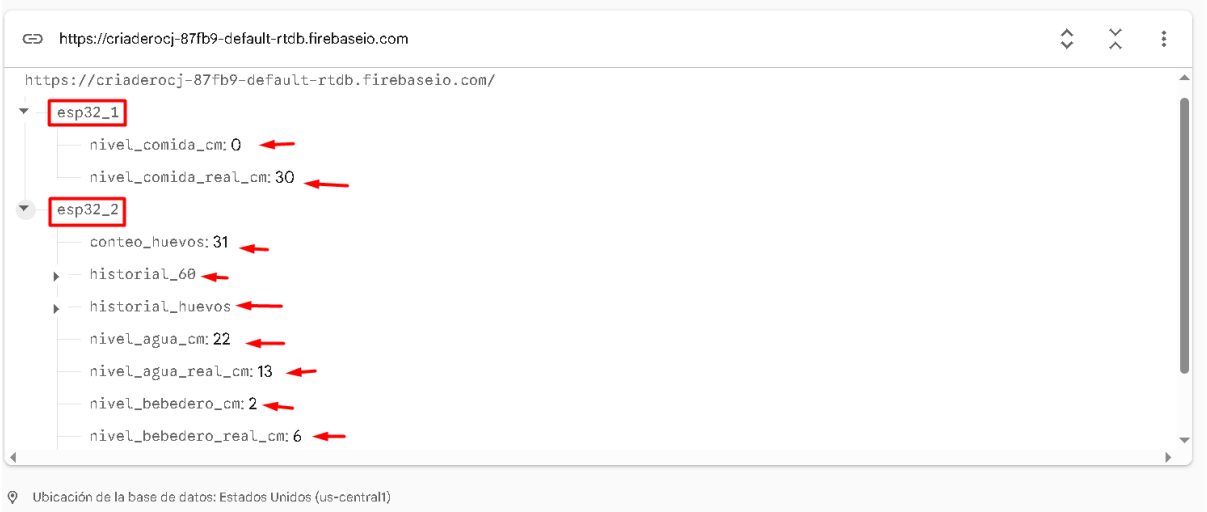


Fuente: Autoría propia

A continuación, tenemos la base de datos el cual se organiza en 2 nodos principales los cuales son el nodo esp32_1 y el nodo esp32_2, los cuales corresponden a cada subsistema tanto de dispensación de alimentación y dispensación de agua, dentro de cada nodo tendremos la presencia de subnodos los cuales representaran un campo de datos los cuales almacenaran valores generados por cada uno de los sensores.

Figura 22

Estructura de nodos en Firebase Realtime Database

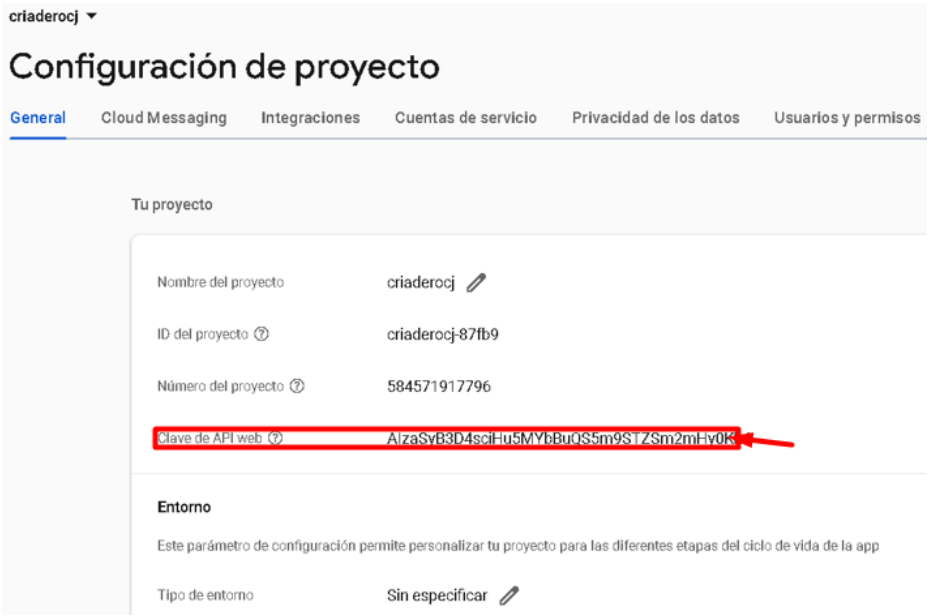


Fuente: Autoría propia

La url de la base de datos obtenida es un elemento muy importante para la programación del microcontrolador ESP32 ya que permite establecer la comunicación con Firebase Realtime Database, Además es importante disponer de la clave API la cual es proporcionada por la plataforma, es por ello que para visualizar estos parámetros nos dirigimos a configuración del proyecto, donde podremos observar la clave API de nuestro proyecto como podemos observar en la **Figura *23***.

Figura 23

Visualización de clave API del proyecto



Fuente: Autoría propia

3.6.1.2 Configuración de Firebase Hosting

En este apartado tenemos la configuración del servicio firebase hosting el cual nos permitirá visualizar en tiempo real los datos almacenados en la base de datos (RTDB), para ello ingresaremos al proyecto previamente creado, una vez dentro nos dirigimos a WEB como se puede observar en la **Figura *24***, una vez que presionemos en web nos muestra una pestaña **"Agrega Firebase a tu app web"**, aquí agregamos el nombre *criaderocodorniz* y seleccionamos el casillero que dice configurar firebase hosting para esta app.

Figura 24

Configuración de firebase hosting

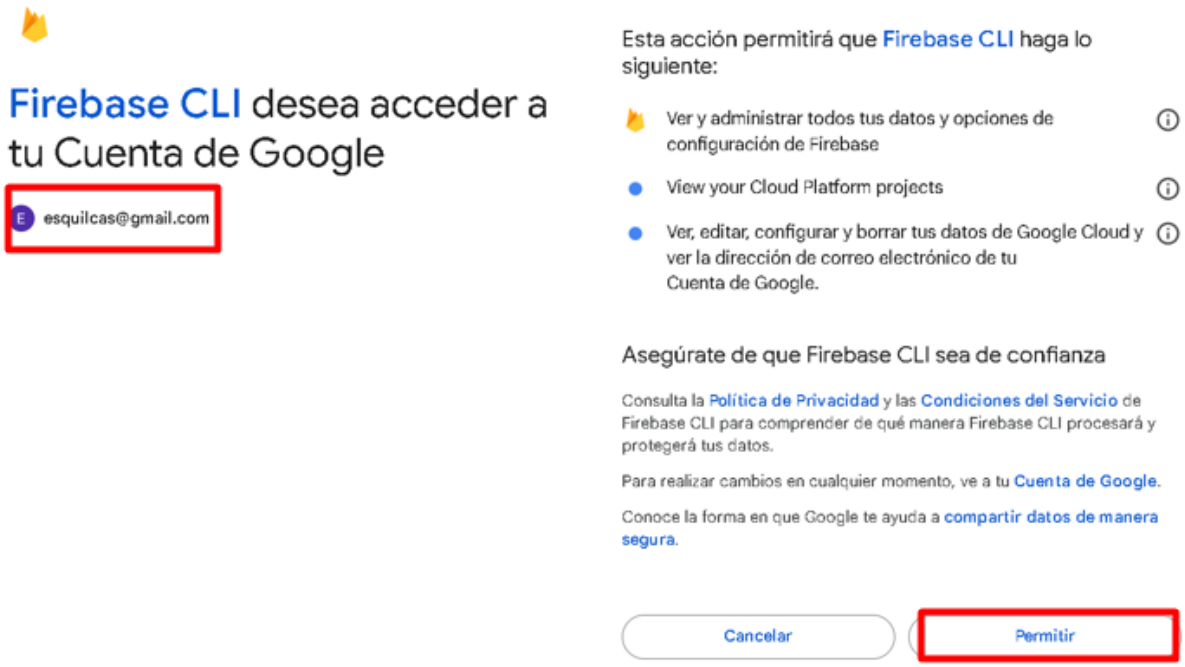
Fuente: Autoría propia

Una vez que realizado el registro, se dio paso a la creación de un directorio de trabajo en el equipo local C:\Users\ASUS\FIREBASE, el cual servirá de entorno para el desarrollo del dashboard, luego se da paso a la instalación de herramientas necesarias para el desarrollo como Visualcode studio y Node.js, los cuales son importantes para editar el código fuente y ejecutar utilidades desde la línea de comandos, en el siguiente paso se incorpora sdk de Firebase con la ejecución del comando comando *npm install firebase*, este sdk se utiliza dentro de los archivos html y js del panel de control, lo que nos permite ver en tiempo real los datos almacenados en la base de datos en tiempo real y visualización de estos datos sin necesidad de recargar la página.

Siguiendo con la configuración tenemos la parte de la instalación de Firebase CLI haciendo uso del comando npm install -g firebase-tools, es por ello que durante el proceso de su configuración se necesita autorizar su acceso a la cuenta de Google que está asociada al proyecto creado, como se puede observar en la **Figura *25***, esto permitirá interactuar con los servicios de Google Cloud y Firebase.

Figura 25

Solicitud de autorización de Firebase CLI para acceder a la cuenta de Google



Fuente: Autoría propia

3.6.2 Configuración Telegram

Para el desarrollo del sistema de control de alimento, el servicio de mensajería de telegram es muy importante ya que será el medio de comunicación entre el sistema y el coturnicultor debido a que gracias a su implementación el coturnicultor podrá recibir alertas por medio de notificaciones automáticas en tiempo real sobre eventos importantes presentes durante el funcionamiento del sistema. Además, con esto el coturnicultor tendrá una herramienta adicional para supervisar el correcto funcionamiento del sistema.

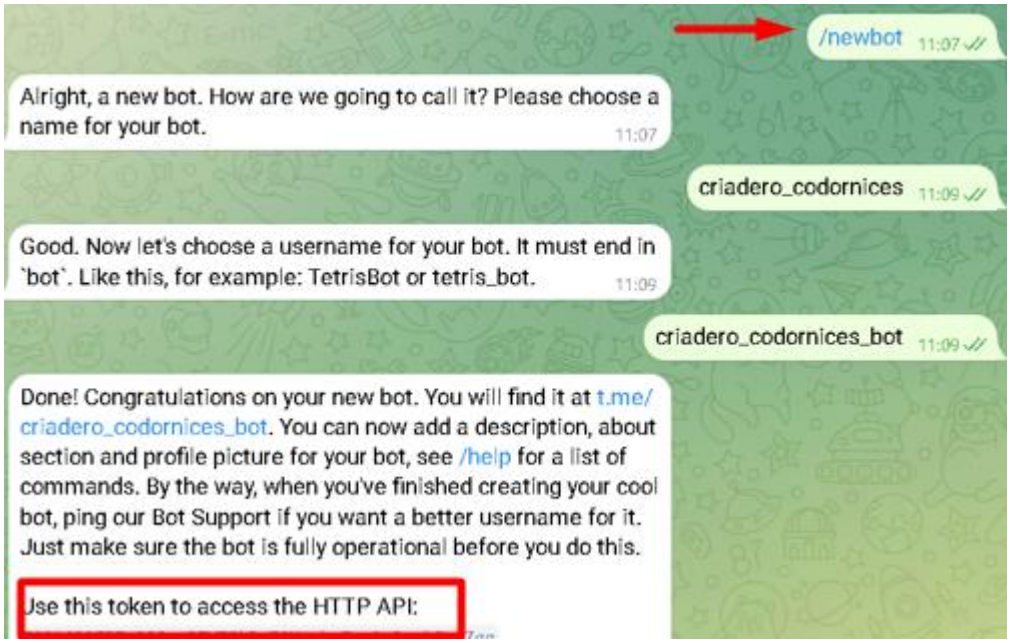
Para llevar a cabo la configuración de Telegram dentro del sistema propuesto, es necesario disponer de dos elementos esenciales como son el token del bot y el ID del chat. A continuación, se presenta la obtención de estos dos elementos.

- ***Creación del Bot de Telegram***

Para la creación del bot de Telegram lo que debemos hacer es ingresar a Telegram y buscar el chat BotFather, una vez que lo encontremos ingresamos y enviamos */newbot*, seguido ponemos un nombre al bot, para el caso del sistema lo hemos denominado criadero_codornices, una vez hecho Telegram genera automáticamente un token de acceso, como podemos observar en la ***Figura 26***.

Figura 26

Creación del bot de telegram



Fuente: *Autoría propia*

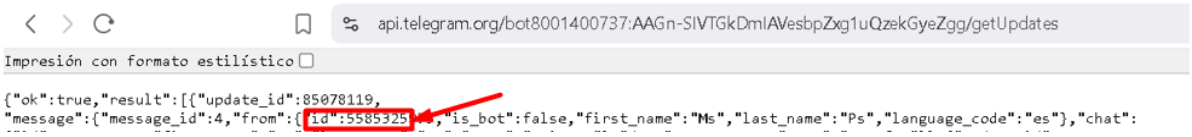
- ***Obtención del ID de chat***

Para la obtención del ID de chat lo primero que debemos realizar es enviar un mensaje al bot de telegram y consultando la API de Telegram ingresando en un navegador web la siguiente dirección y reemplazando la parte de token por el token del bot creado https://api.telegram.org/bot<reemplazarmiTOKEN>/getUpdates , una vez que realizamos esto, recargamos la página y telegram nos devuelve una respuesta en formato JSON donde podemos

observar los datos del mensaje enviado, entre estos datos podemos observar el ID de chat, el cual corresponde al identificador numérico único del chat, como podemos observar en la **Figura *27*** .

Figura 27

Obtención del ID de chat Telegram



Fuente: Autoría propia

3.6.3 Programación ESP32

En este apartado tenemos la programación de los microcontroladores ESP32 tanto para el subsistema del comedero como el subsistema de dispensador de agua ya que es lo que permitirá ejecutar la lógica de control requerida para el funcionamiento de los procesos del sistema de control de alimentación de las codornices, este proceso se lo llevará a cabo a través del firmware desarrollado, en el cual el microcontrolador será capaz de adquirir datos de los sensores y procesarlos de acuerdo a las condiciones establecidas además de activar actuadores en la cual se encuentran las bombas, servomotor, motores y buzzer.

Además, para dar el paso al desarrollo del código se utilizó Arduino IDE debido a que el microcontrolador elegido presentó compatibilidad con este IDE de desarrollo, otro punto que se consideró fue la disponibilidad de librerías y la facilidad para compilar y cargar el programa mediante conexión USB.

Como punto final mencionar que los códigos completos están en la parte de **Anexos C** y **Anexo D**.

3.6.3.1 Subsistema dispensador de alimento

Como se puede apreciar en la **Figura *28***, podemos observar la sección inicial del código del sistema implementado, comienza con la incorporación de las librerías necesarias para el funcionamiento de microcontrolador ESP32, así mismo en esta parte del programa se establece la base sobre la cual opera todo el sistema, la cual es la habilitación de la comunicación inalámbrica mediante WiFi , así mismo se integra la librería de Firebase y la librería para comunicaciones seguras mediante HTTPS el cual nos servirá para enviar notificaciones por telegram, además incluimos la librería encargada del control del servomotor junto con la librería del tiempo el cual nos ayudará a la hora de trabajar con horarios definidos mediante la sincronización con servidores NTP, además se incluye la librería Watchdog el será importante para prevenir bloqueos del sistema ,como último punto tenemos la definición de los parámetros de la red WiFi como son el nombre de la red WiFi: INNO-FIBER- FLIA PUPIALES y la contraseña: 1704Pupiales#, con esto permitiremos la conexión automática y un funcionamiento continuo.

Figura 28

Inclusión de librerías y credenciales de conexión WiFi, (dispensador de alimento)

```
//*************************Librerías*************************
#include <Arduino.h>              // Funciones base
#include <WiFi.h>                 // Conexión WiFi
#include <WiFiClientSecure.h>   // Cliente HTTPS (TLS) para Telegram
#include <Firebase_ESP_Client.h>  // Firebase RTDB
#include <ESP32Servo.h>            // Control de servo
#include <time.h>                 // Hora NTP
#include "esp_task_wdt.h"         // Watchdog

//******************Credenciales red WiFi*****************

#define WIFI_SSID     "INNO-FIBER- FLIA PUPIALES" // Nombre de la red WiFi
#define WIFI_PASSWORD "1704Pupiales#"             // Contraseña de la red WiFi

```

Fuente: Autoría propia

Como podemos observar en la **Figura *29***, tenemos la asignación de pines físicos del microcontrolador ESP32 donde se definen los pines TRIG GIO12 y ECHO GIO34 correspondientes al sensor ultrasónico HC-SR04 el cual será el encargado de medir el nivel de alimento, el pin GIO13 para el servomotor el cual será el encargado de la dispensación controlada del alimento, además se presentan los pines GIO26,25,33,32,27 y 14 asociados al driver de motores los cuales nos permitirán el control para la banda transportadora de comida, además el pin GIO4 destinado al buzzer el cual cumple con la función de emitir la alerta sonora, siguiendo el código tenemos la parte donde incorporamos las credenciales necesarias para la conexión con la base de datos , aquí hacemos uso del API key Firebase y URL de RTDB, con esto posibilitamos el envío y recepción de datos generados por el sistema, por último, se incorporan las credenciales del bot de Telegram como es el token de acceso y el ID del chat, los cuales permiten establecer la comunicación para el envío de notificaciones al usuario.

Figura 29

Asignación de pines y credenciales de Firebase/Telegram, (dispensador de alimento)

```
//******************Asignación de pines ******************
#define TRIG_PIN   12   // TRIG del sensor ultrasónico
#define ECHO_PIN   34   // ECHO del ultrasónico
#define SERVO_PIN  13   // Pin del servo
#define IN1        26   // Control motor
#define IN2        25   // Control motor
#define ENA        33   // PWM del motor A
#define IN3        32   // Control motor B
#define IN4        27   // Control motor B
#define ENB        14   // PWM del motor B
#define BUZZER_PIN 4    // Pin del buzzer

//****************** Credenciales Firebase para conectar a la base de datos******************
#define API_KEY      "AIzaSyB3D4sciHu5MYbBuQS5m9STZSm2mHy0Kh4" // API Key de Firebase
#define DATABASE_URL "https://criaderocj-87fb9-default-rtdb.firebaseio.com/" // URL RTDB

FirebaseData fbdo;        // Objeto para operaciones en RTDB
FirebaseAuth auth;        // Objeto de autenticación
FirebaseConfig config;    // Configuración de Firebase
bool signupOK      = false; // Marca si el signup fue correcto
bool firebaseReady = false; // Marca si Firebase quedó inicializado

//****************** Parámetros de autenticaicón con telegram******************

const char* TELEGRAM_BOT_TOKEN = "8001400737:AAGn-SIVTGkDmIAVesbpZxg1uQzekGyeZgg"; // Token del bot
const char* TELEGRAM_CHAT_ID   = "5585325975";                                     // ID del chat destino
WiFiClientSecure telegramClient; // Cliente seguro para conectar a api.telegram.org
```

Fuente: Autoría propia

Como se puede apreciar en la **Figura *30***, en primer lugar tenemos la función leerDistanciaCrudaCM() el cual se encarga de realizar la lectura básica del sensor mediante la generación del pulso TRIG y la medición del tiempo de retorno del eco, convirtiendo este valor a centímetros y descartando lecturas inválidas o fuera del rango físico permitido. Seguido a esto tenemos la función medirEstableInt() la cual toma un conjunto de 10 lecturas consecutivas y almacena las válidas, con el objetivo de disponer de una muestra representativa del estado real del nivel del alimento, sobre estos datos obtenidos aplicamos el algoritmo quickSelect(), el cual permite calcular de la mediana de los datos obtenidos este valor actúa de referencia y, a partir de ella, se realiza un promedio recortado, considerando únicamente las lecturas que se encuentran dentro de una banda de ±10 % , gracias a este proceso se podrá eliminar picos aislados que no representan cambios reales en el nivel del comedero. Como punto final tenemos, el suavizado del resultado mediante EMA, una vez que se haya aplicado todos estos procesos tendremos una medición estable la cual nos servirá para tomar decisiones posteriormente con lo que respecta al sistema.

Figura 30

Funciones y filtrado de lectura de distancia por el sensor ultrasónico HC-SR04, (dispensador de alimento)

```
float leerDistanciaCrudaCM(uint8_t trigPin, uint8_t echoPin) { // Lectura directa del HC-SR04
  digitalWrite(trigPin, LOW);            // Asegura TRIG en bajo
  delayMicroseconds(3);                 // Pequeña espera
  digitalWrite(trigPin, HIGH);          // Pulso HIGH
  delayMicroseconds(10);                // 10us de disparo
  digitalWrite(trigPin, LOW);           // Termina pulso

  long duracion = pulseIn(echoPin, HIGH, 25000); // Tiempo del eco (máx 25ms)
  if (duracion <= 0) return 0.0f;                // Si no hubo eco, inválido

  float d = duracion * 0.0343f / 2.0f;           // Convierte a cm (velocidad sonido)
  if (d < 1.0f || d > 400.0f) return 0.0f;       // Filtra rangos raros
  return d;                                      // Retorna distancia válida
}

int medirEstableInt(uint8_t trigPin, uint8_t echoPin, float &emaRef) { // Lectura filtrada
  const int N = 10;                 // Número de muestras
  float vals[N];                    // Arreglo de lecturas
  int cnt = 0;                      // Contador de lecturas válidas

  for (int i = 0; i < N; i++) {                    // Toma N lecturas
    float v = leerDistanciaCrudaCM(trigPin, echoPin); // Lee distancia cruda
    if (v > 0) vals[cnt++] = v;                    // Guarda solo si es válida
    delay(2);                                      // Pequeña pausa
  }

  if (cnt == 0) {                                  // Si no hubo lecturas válidas
    if (isnan(emaRef)) return 0;                   // Si ni EMA existe, devuelve 0
    return (int)roundf(emaRef);                    // Si existe, usa EMA anterior
  }
```

Fuente: Autoría propia

A continuación como se puede apreciar en la **Figura *31***, se muestra el conjunto de funciones desarrolladas para permitir el envío de notificaciones automáticas a través de la plataforma de Telegram, inicialmente se adapta el contenido del mensaje a un formato compatible con el envío por internet, transformando caracteres especiales y espacios para evitar errores durante la transmisión, Seguido a esto, el sistema verifica la disponibilidad de la conexión WiFi antes de intentar el envío, con esto se evita procesos innecesarios cuando no existe conectividad.

Después de esto si se confirma la conexión, el ESP32 establecerá comunicación segura con los servidores de Telegram mediante el uso de peticiones HTTPS, como punto final el

sistema analiza la respuesta del servidor para confirmar si el mensaje fue enviado correctamente.

Figura 31

Notificaciones a través de telegram, (dispensador de alimento)

```
196  float leerDistanciaCrudaCM(uint8_t trigPin, uint8_t echoPin) { // Lectura directa del HC-SR04
197    digitalWrite(trigPin, LOW);            // Asegura TRIG en bajo
198    delayMicroseconds(3);                  // Pequeña espera
199    digitalWrite(trigPin, HIGH);           // Pulso HIGH
200    delayMicroseconds(10);                 // 10us de disparo
201    digitalWrite(trigPin, LOW);            // Termina pulso
202
203    long duracion = pulseIn(echoPin, HIGH, 25000); // Tiempo del eco (máx 25ms)
204    if (duracion <= 0) return 0.0f;                // Si no hubo eco, inválido
205
206    float d = duracion * 0.0343f / 2.0f;           // Convierte a cm (velocidad sonido)
207    if (d < 1.0f || d > 400.0f) return 0.0f;       // Filtra rangos raros
208    return d;                                      // Retorna distancia válida
209  }
210
211  int medirEstableInt(uint8_t trigPin, uint8_t echoPin, float &emaRef) { // Lectura filtrada
212    const int N = 10;                  // Número de muestras
213    float vals[N];                     // Arreglo de lecturas
214    int cnt = 0;                       // Contador de lecturas válidas
215
216    for (int i = 0; i < N; i++) {                    // Toma N lecturas
217      float v = leerDistanciaCrudaCM(trigPin, echoPin); // Lee distancia cruda
218      if (v > 0) vals[cnt++] = v;                    // Guarda solo si es válida
219      delay(2);                                      // Pequeña pausa
220    }
221
222    if (cnt == 0) {                                  // Si no hubo lecturas válidas
223      if (isnan(emaRef)) return 0;                   // Si ni EMA existe, devuelve 0
224      return (int)roundf(emaRef);                    // Si existe, usa EMA anterior
225    }
226
227    float tmp[10];                                   // Arreglo temporal para mediana
```

Fuente: Autoría propia

En este apartado se puede apreciar la **Figura *32***,en el cual tenemos la función encargada de ejecutar la dispensación automática del alimento, la cual se activa cuando es el momento programado para realizar la dispensación del alimento, como punto inicial tenemos la notificación al usuario mediante Telegram donde se envía un mensaje diciendo “Inicio de dispensación de comida”, Seguido a esto, se acciona el servomotor encargado de abrir la compuerta para dispensar el alimento por un tiempo de 30 segundos, de igual modo los motores de la banda transportadora se ponen en funcionamiento para asegurar la correcta salida del

alimento, una vez transcurrido el tiempo de dispensación, los motores se detienen y la compuerta se cierra, a su vez una vez finalizado este proceso el sistema envía una notificación final confirmando la correcta dispensación con un mensaje “250 gramos de comida exibal dispensados correctamente”.

Figura 32

Función de control para la dispensación de alimento, (dispensador de alimento)

```
// ********************** DISPENSADOR ************************************
void dispensar() {
  enviarMensajeTelegram("🔄🔃Inicio de dispensación de comida"); // Envía notificacion de inicio de dispensación a Telegram
  dispensador.write(180);              // Mueve servo a 180º (abre)
  delay(300);                          // Espera para que el servo llegu
  digitalWrite(IN1, HIGH);             // Activa sentido motor A
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);             // Activa sentido motor B
  digitalWrite(IN4, LOW);
  delay(100);                          // Pequeña espera antes del PWM

  ledcWrite(4, 255);                   // PWM máximo en canal 4 (ENA)
  ledcWrite(5, 255);                   // PWM máximo en canal 5 (ENB)
  unsigned long start = millis();      // Marca inicio del tiempo de trabajo
  while (millis() - start < 30000) {   // Mantiene el proceso 30s
    esp_task_wdt_reset();              // Alimenta watchdog para no reiniciar
    delay(100);                        // Pausa corta (evita bloquear totalmente)
  }
  ledcWrite(4, 0);                     // Apaga PWM motor A
  digitalWrite(IN1, LOW);              // Apaga señales motor A
  digitalWrite(IN2, LOW);
  ledcWrite(5, 0);                     // Apaga PWM motor B
  digitalWrite(IN3, LOW);              // Apaga señales motor B
  digitalWrite(IN4, LOW);
  dispensador.write(0);                // Regresa servo a 0º (cierra)
  delay(300);                          // Espera final
  enviarMensajeTelegram("✅🔃250 gramos de comida exibal dispensados correctamente"); // Envía notificacion de fin dedispensación a Telegram
  Serial.println("[DISPENSADOR] FIN"); // Log al monitor serial
}
```

Fuente: Autoría propia

A continuación, en la **Figura *33***, se presenta la conexión del ESP32 a la red WiFi, inicia la conexión utilizando las credenciales definidas, el programa establece un tiempo máximo de espera de 15000 ms, durante el cual se verifica repetidamente el estado de conexión, mostrando un punto de avance aproximadamente cada 300 ms, lo que permite observar claramente si el enlace está progresando o no. Si después del tiempo de 15 segundos el ESP32 no logra conectarse continúa en modo OFFLINE es decir se mantiene bajo operación local del dispositivo, pero si la conexión se completa correctamente, muestra la dirección IP asignada, indicando que el sistema queda listo para continuar con sus funciones.

Figura 33

Conexión WiFi del sistema, (dispensador de alimento)

```
// ********************** WIFI**********************
void conectarWiFiRobusto() {
  WiFi.mode(WIFI_STA);                        // Modo estación (cliente)
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);       // Inicia conexión WiFi
  unsigned long inicio = millis();            // Marca inicio
  Serial.print("[WiFi] Conectando");          // Log

  while (WiFi.status() != WL_CONNECTED && millis() - inicio < 15000) { // Espera máx 15s
    delay(300);                               // Espera
    Serial.print(".");                        // Progreso
  }
  Serial.println();                           // Salto de línea

  if (WiFi.status() != WL_CONNECTED) {        // Si no conectó
    Serial.println("[WiFi] No se pudo conectar en 15s. Sigo OFFLINE (solo local).");
  } else {                                    // Si conectó
    Serial.print("[WiFi] Conectado. IP: ");
    Serial.println(WiFi.localIP());           // Muestra IP
  }
}
```

Fuente: Autoría propia

En la **Figura *34***, podemos ver el código realizado para la parte de inicialización de firebase, reintentos no bloqueantes, y control de alertas, aquí como inicio tenemos la función iniciarFirebase() el cual verifica si el dispositivo está conectado a WiFi, pero en caso de no existir conexión el sistema evita intentar enlazarse a la nube y marca a Firebase como “no listo”, lo que nos permite que el sistema siga funcionando con normalidad sin interrupciones pero sin subir nada a firebase. Pero en caso de que si exista el servicio de internet se asignan los parámetros como API Key y URL del RTDB realizando un registro anónimo (signup).

Seguido, si ya se inicializó Firebase, se activa el reconectado automático de WiFi y se confirma que el servicio queda disponible Además se puede ver un mecanismo de reintentarFirebaseNoBloqueante() el cual vuelve a intentar la inicialización únicamente cuando sea necesario y con una espera mínima de 60 segundos, con esto evita que el sistema se quede atrapado en bucles de reconexión. Como último punto en este bloque tenemos la parte donde se define variables y temporizadores para controlar la repetición de notificaciones, en este caso

el aviso de “vacío” se repite cada 3 horas, mientras que un estado “crítico” se repite cada 1 hora, con esto evitamos el exceso de mensajes.

Figura 34

Inicialización de firebase, reintentos no bloqueantes y control de alertas, (dispensador de alimento)

```
// ****************************FIREBASE**********************************
void iniciarFirebase() {
  if (WiFi.status() != WL_CONNECTED) {       // Si no hay WiFi, no intenta Firebase
    Serial.println("[Firebase] No inicio: no hay WiFi.");
    firebaseReady = false;                   // Marca no listo
    return;                                  // Sale
  }
  Serial.println("[Firebase] Inicializando..."); // Log
  config.api_key      = API_KEY;             // Asigna API key
  config.database_url = DATABASE_URL;        // Asigna URL del RTDB
  if (Firebase.signUp(&config, &auth, "", "")) { // Signup anónimo
    signupOK = true;                         // Marca OK
    Serial.println("[Firebase] SignUp OK");
  } else {                                   // Si falló signup
    signupOK      = false;                   // Marca error
    firebaseReady = false;                   // Firebase no listo
    Serial.print("[Firebase] SignUp ERROR: ");
    Serial.println(config.signer.signupError.message.c_str()); // Mensaje de error
    return;                                  // Sale
  }
  Firebase.begin(&config, &auth);        // Inicia Firebase
  Firebase.reconnectWiFi(true);          // Reintenta WiFi automáticamente si se cae
  firebaseReady = true;                 // Marca listo
  Serial.println("[Firebase] Listo (modo mejor esfuerzo).");
}
void reintentarFirebaseNoBloqueante(unsigned long ahora) {
  if (firebaseReady) return;                 // Si ya está listo, no reintenta
  if (ahora - ultimoIntentoFbMs < 60000) return; // Reintenta cada 60s
  Serial.println("[Firebase] Reintentando iniciar..."); // Log
  iniciarFirebase();                                    // Intenta iniciar otra vez
  ultimoIntentoFbMs = ahora;                            // Guarda momento del intento
}
```

Fuente: Autoría propia

En la siguiente captura podemos ver la **Figura *35***, donde podemos visualizar la ejecución de la rutina de arranque del ESP32, la cual se activa una sola vez al encender, como segundo punto se habilita la comunicación serial a 115200 baudios, además se configuran dos salidas PWM en los canales 4 y 5, trabajando a 1 kHz con resolución de 8 bits, y se asocian a los pines ENA y ENB para controlar la potencia de los motores, iniciando con valor 0 para

evitar que el dispensador se active accidentalmente. Una vez que se realiza esto sigue con la definición de los pines del sensor ultrasónico, en el caso del pin TRIG como salida y el pin ECHO como entrada, así como los pines del driver del motor IN1, IN2, IN3, IN4 para garantizar que el control de giro quede correctamente establecido.

Además, se configura el buzzer como salida y se lo mantiene apagado desde el inicio, evitando alertas innecesarias cuando arranque a funcionar el sistema. También, se adjunta el servomotor al pin correspondiente y se lo coloca en posición inicial 0°, asegurando que la compuerta comience cerrada.

Una vez realizado esto, el sistema intentará conectarse a la red WiFi mediante el método de conexión robusta y procede a iniciar Firebase para habilitar el monitoreo remoto. Además, se configura la hora mediante NTP usando la zona horaria UTC-5, lo que permitirá que las acciones programadas funcionen con precisión de acuerdo a la hora local. Algo adicional que se agregó fue la activación de el watchdog con un tiempo de supervisión de 10 segundos, esto se lo hizo como una medida de seguridad, en caso de reiniciar automáticamente el sistema si llegara a bloquearse. Por último, tenemos el envío de un mensaje por Telegram indicando que el comedero ha sido iniciado, confirmando al usuario que el sistema quedó operativo.

Figura 35

Rutina de inicialización del sistema y configuración de periféricos (setup) , (dispensador de alimento)

```
// ***************SETUP ************************************
void setup() {
  Serial.begin(115200);                 // Inicia comunicación serial

  ledcSetup(4, 1000, 8);                 // Canal PWM 4, 1kHz, 8 bits
  ledcAttachPin(ENA, 4);                // ENA queda asociado al canal 4
  ledcWrite(4, 0);                      // Arranca con PWM = 0

  ledcSetup(5, 1000, 8);                // Canal PWM 5, 1kHz, 8 bits
  ledcAttachPin(ENB, 5);               // ENB asociado al canal 5
  ledcWrite(5, 0);                      // Arranca con PWM = 0

  pinMode(TRIG_PIN, OUTPUT);            // TRIG como salida
  pinMode(ECHO_PIN, INPUT);               // ECHO como entrada
  pinMode(IN1, OUTPUT);                 // Pines del driver como salida
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);          // Buzzer como salida
  digitalWrite(BUZZER_PIN, LOW);        // Buzzer apagado al inicio

  dispensador.attach(SERVO_PIN);          // Asocia servo al pin indicado
  dispensador.write(0);                 // Posición inicial del servo

  conectarWiFiRobusto();                // Intenta conectar WiFi
  iniciarFirebase();                    // Intenta iniciar Firebase

  configTime(-5 * 3600, 0, "pool.ntp.org", "time.nist.gov"); // Config hora NTP (UTC-5)

  esp_task_wdt_init(10, true);          // Config watchdog (10s, reset si se cuelga)
  esp_task_wdt_add(NULL);               // Agrega tarea actual al watchdog
```

Fuente: Autoría propia

Siguiendo con las líneas de código podemos observar en la **Figura *36*** , el ciclo principal donde se da la lectura de manera periódica del sensor ultrasónico HC-SR04, esta lectura se realiza cada 400 ms, con esto evita lecturas innecesarias que puedan generar ruido, una vez que se obtienen los datos referentes a la distancia pasa por un proceso de filtrado y comparación con la última lectura estable registrada, mediante la aplicación de un criterio de histéresis donde a la primera medición válida se lo toma como referencia, después de esto el sistema compara el nuevo valor filtrado con la última lectura estable, por lo que si la diferencia entre ambos

supera el umbral establecido, se considera que el cambio es real y se acepta la nueva medición, de lo contrario, se mantiene el valor anterior, una vez que realiza esto, el sistema conserva una medición mucho más confiable la cual será utilizada luego para que el sistema pueda tomar decisiones.

Figura 36

Lectura de sensor ultrasónico con estabilidad y filtrado, (dispensador de alimento)

```
//  ***************************** LECTURA ULTRASONIDO CADA 400 ms + HISTERESIS *************
static int distFiltrada = 0;          // Mantiene el valor filtrado entre iteraciones

if (ahora - ultimoTiempoLectura >= intervaloLectura) {   // Si ya pasó el intervalo
  distFiltrada = medirEstableInt(TRIG_PIN, ECHO_PIN, emaComedero); // Lee filtrado
  ultimoTiempoLectura = ahora;                            // Actualiza tiempo de lectura
}

if (ultimaLecturaEstable < 0) {     // Primera vez que entra
  ultimaLecturaEstable = distFiltrada; // Toma lectura inicial
} else {
  if (abs(distFiltrada - ultimaLecturaEstable) > HISTERESIS_CM) { // Si el cambio es real
    ultimaLecturaEstable = distFiltrada; // Acepta el nuevo valor
  }
}

int distEntera = ultimaLecturaEstable; // Distancia final usada en decisiones
float distancia = (float)distEntera;   // Versión float para cálculos

```

Fuente: Autoría propia

A continuación, como se puede observar en la ***Figura 37***, tenemos el procesamiento para determinar la altura real del alimento dentro del contenedor, para ello gracias a la distancia que se obtenga se la resta de la altura total del contenedor, una vez que hace este cálculo se obtiene un valor con el cual el programa obtiene el porcentaje de llenado ya que este porcentaje se lo expresó de 0% a 100%, este valor se ajusta automáticamente con el fin de que no se exceda los límites permitidos. Un punto importante que toca mencionar es que el sistema asegura que la altura nunca sea negativa porque si supera el nivel permitido este entra en la condición donde if (alturaComida < 0) alturaComida = 0, con esto evitamos valores negativos.

Una vez que se obtiene la distancia se clasifica el estado del comedero, si la distancia es menor o igual a 12 cm, el contenedor se considera lleno y el porcentaje se fija en 100 %. Si

la distancia se encuentra entre 13 y 22 cm, el estado se define como medio, si la distancia esta entre 23 y 31 cm, se clasifica como bajo y por último si la distancia esta entre 32 y 39 cm, se considera crítico. Cuando la distancia supera los 39 cm, el comedero está en estado vacío, y se establece el porcentaje en 0 % y la altura del alimento en 0 cm, en cambio para el porcentaje entre el 0% y e 100% se hace uso de la distancia y se usa la siguiente fórmula: porcentaje = (alturaComida / ALTURA_MAX_CM) * 100

Además, si el sistema cuenta con conexión a internet y si la inicialización de Firebase esta activa el ESP32 registra y almacena los datos del sensor ultrasónico en la ruta /esp32_1/nivel_comida_cm, de igual modo el nivel real calculado del alimento en /esp32_1/nivel_comida_real_cm, gracias a esto el coturnicultor podrá monitorear en tiempo real mediante un dashbaord el estado del contenedor de comida.

Figura 37

Cálculo y clasificación del alimento dentro de contenedor, (dispensador de alimento)

```
411   //  ****************************CÁLCULO DE ALTURA Y PORCENTAJE CONTENEDOR DE COMIDA***************
412   float alturaComida = ALTURA_TOTAL_COMEDERO_CM - distancia; // Convierte distancia a “altura”
413   if (alturaComida < 0) alturaComida = 0;                    // Evita negativos
414  float alturaMax = ALTURA_TOTAL_COMEDERO_CM - ALTURA_LLENO_CM; // Altura útil máxima
415   if (alturaComida > alturaMax) alturaComida = alturaMax;       // Limita al máximo
416   int alturaEntera = (int)roundf(alturaComida);  // Redondea altura a entero
417   float porcentajeF = (alturaComida / alturaMax) * 100.0f; // Calcula porcentaje
418   if (porcentajeF < 0) porcentajeF = 0;                     // Límite inferior
419   if (porcentajeF > 100) porcentajeF = 100;                   // Límite superior
420   int porcentaje = (int)roundf(porcentajeF);                // Porcentaje entero
421   String nuevoEstado; // Variable donde se guarda el estado textual
422   if (distEntera <= 12) {      // Si la distancia es pequeña, está lleno
423     nuevoEstado = "lleno";    // Estado lleno
424     porcentaje = 100;        // Fuerza 100%
425   }
426   else if (distEntera <= 22) nuevoEstado = "medio";   // Rango medio
427   else if (distEntera <= 31) nuevoEstado = "bajo";    // Rango bajo
428   else if (distEntera <= 39) nuevoEstado = "critico"; // Rango crítico
429   else {                                              // Si supera 39 cm
430     nuevoEstado = "vacio";      // Estado vacío
431     porcentaje = 0;           // 0%
432     alturaEntera = 0;           // Nivel 0 cm
433   }
434   Serial.printf("[COMEDERO] dist=%d cm | altura=%d cm | %d %% | estado=%s\n",
435                 distEntera, alturaEntera, porcentaje, nuevoEstado.c_str()); // Debug en Serial
436
437   //**************************** ENVÍO A FIREBASE ****************************
438   if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED) { // Verifica condiciones
439     Firebase.RTDB.setInt(&fbdo, "/esp32_1/nivel_comida_cm", distEntera);      // Guarda distancia
440     Firebase.RTDB.setInt(&fbdo, "/esp32_1/nivel_comida_real_cm", alturaEntera); // Guarda nivel real
441   }
```

Fuente: Autoría propia

Siguiendo las líneas de código en la ***Figura 38*** , podemos observar el código implementado para las notificaciones del estado lleno cuando se haga la recarga del contenedor, como punto inicial el sistema al encender realiza la primera lectura y la guarda y desactiva el indicador de primera lectura, para que las posteriores comparaciones puedan basarse con un dato válido inicial, seguido comprueba si el nivel de alimento se ha separado del estado de lleno más allá de un valor de referencia. Cuando esto sucede, se habilita nuevamente el envío de notificaciones, de modo que el sistema pueda alertar otra vez si el comedero vuelve a llenarse.

Siguiendo las líneas d código se comprueba que la notificación de lleno no haya sido enviada previamente, que el comedero no se encontrara en ese estado en la lectura anterior y que la medición actual indique que ha ingresado al nivel de lleno, si se cumplen estas condiciones se genera la notificación el cual indica que el comedero se encuentra lleno al 100 %, además muestra la altura real en centímetros. Por otro lado, si el envío de la notificación por Telegram se envía correctamente el sistema actualiza el estado como notificado, pero si en caso el mensaje por Telegram no se envía, se conserva el estado anterior, lo que nos permitirá que el aviso pueda intentarse nuevamente, al final, la distancia registrada se utiliza como punto de comparación para detectar cambios en el estado del comedero en las siguientes mediciones.

Figura 38

Notificación de estado lleno del contenedor de alimento, (dispensador de alimento)

```
// ********************************NOTIFICACIÓN ESTDADO LLENO*********************************
if (primeraLecturaComedero) {           // Solo la primera vez
  ultimaDistanciaComedero = distEntera; // Inicializa distancia anterior
  primeraLecturaComedero  = false;      // Marca que ya pasó la primera lectura
}

if (distEntera > UMBRAL_LLENO_CM + REARME_LLENO_DELTA) { // Si se alejó del lleno
  comederoLlenoNotificado = false;                        // Rearma notificación
}

if (!comederoLlenoNotificado &&                         // Si aún no notificó
    ultimaDistanciaComedero > UMBRAL_LLENO_CM &&        // Antes estaba “no lleno”
    distEntera <= UMBRAL_LLENO_CM) {                    // Ahora entró a “lleno”

  String msg = "✅🍽️ Comedero LLENO (100%) · Nivel: " + String(alturaEntera) + " cm"; // Mensaje
  if (enviarMensajeTelegram(msg.c_str())) {             // Si se envió bien
    comederoLlenoNotificado = true;                     // Marca como notificado
  } else {
    Serial.println("[Telegram] LLENO no enviado, no marco notificado"); // Log de fallo
  }
}

ultimaDistanciaComedero = distEntera; // Actualiza distancia anterior
```

Fuente: Autoría propia

A continuación, en la ***Figura 39***, podemos ver la programación realizada para el control de notificaciones por Telegram, vamos a empezar por el estado vació donde como primer punto se verifica que el contenedor se encuentre sin alimento, aquí si el contenedor está vacío se controla el tiempo transcurrido desde el último mensaje enviado, de esta manera la notificación se envía solo si no ha sido enviada anteriormente, por otro lado si es la primera vez a este estado, el sistema enviará una notificación indicando el que el nivel de alimento es de 0 cm, si este estado continua, se genera recordatorios cada 3 horas, para este proceso de recordatorios el temporizador se actualizará cuando el mensaje se envíe correctamente, Ahora siguiendo con el estado de crítico la lógica es similar con la diferencia que ahora la notificación se enviará cada hora, mencionando que debe realizar la recarga lo antes posible.

Figura 39

Notificaciones en estado crítico y vacío del contenedor, (dispensador de alimento)

```
// ***************************NOTIFICACIONES A TELEGRAM CUANDO ESTADO SEA VACÍO Y CRÍTICO*********************
if (nuevoEstado == "vacio") { // Si está vacío
  if (lastVacioMsgTime == 0 || (ahora - lastVacioMsgTime) >= INTERVALO_VACIO_MS) { // Control de tiempo
    String msg; // Mensaje a enviar
    if (estadoNotificado != "vacio") { // Si recién cambió a vacío
      msg = "⛔🍽️ Comedero VACÍO · Nivel: 0 cm · Sin alimento disponible. Recargar con urgencia.";
    } else { // Si ya estaba vacío (recordatorio)
      msg = "⛔🍽️ Recordatorio: Comedero continúa VACÍO · Sin alimento disponible.";
    }
    if (enviarMensajeTelegram(msg.c_str())) { // Si se envió bien
      lastVacioMsgTime = ahora;               // Actualiza timer
    } else {
      Serial.println("[Telegram] VACIO no enviado, no actualizo timer"); // Log de error
    }
  }
}
if (nuevoEstado == "critico") { // Si está crítico
  if (lastCriticoMsgTime == 0 || (ahora - lastCriticoMsgTime) >= INTERVALO_CRITICO_MS) { // Control de tiempo
    String msg; // Mensaje a enviar
    if (estadoNotificado != "critico") { // Cambio reciente a crítico
      msg = "⚠️🍽️ Comedero en nivel CRÍTICO (" + String(porcentaje) +
            "%) · Nivel: " + String(alturaEntera) + " cm · Recargar lo antes posible.";
    } else { // Recordatorio si sigue crítico
      msg = "⚠️🍽️ Recordatorio: Comedero continúa en nivel CRÍTICO (" +
            String(porcentaje) + "%) · Nivel: " + String(alturaEntera) + " cm.";
    }
    if (enviarMensajeTelegram(msg.c_str())) { // Si se envió
      lastCriticoMsgTime = ahora;             // Actualiza timer
    } else {
      Serial.println("[Telegram] CRITICO no enviado, no actualizo timer");
    }
  }
}
```

Fuente: Autoría propia

Siguiendo las líneas de código entramos a la implementación de la lógica para la activación del buzzer como podemos ver en la **Figura *40***, aquí cuando el sistema detecta que el contenedor está en estado lleno, el buzzer se encuentra habilitado, se activa por 5 segundos y después lo deshabilita para evitar que vuelva a sonar continuamente.

Cuando el sistema detecta que el comedero deja de estar lleno, comienza a contar el tiempo que permanece en ese estado y si continúa sin estar lleno durante al menos 30 segundos, el buzzer se vuelve a habilitar, quedando listo para emitir una nueva alerta sonora cuando el comedero se llene nuevamente.

Figura 40

Lógica para la activación de buzzer, (dispensador de alimento)

```
// ************************ LÓGICA BUZZER *********************************
if (nuevoEstado == "lleno" && buzzerArmado && !buzzerActivo) { // Si está lleno y aún no sonó
  digitalWrite(BUZZER_PIN, HIGH);   // Enciende buzzer
  buzzerActivo   = true;            // Marca activo
  buzzerInicioMs = millis();        // Guarda inicio
  buzzerArmado   = false;           // Desarma para que no repita
  Serial.println("[BUZZER] Comedero LLENO -> buzzer ON 5s");
}
if (buzzerActivo && (millis() - buzzerInicioMs >= BUZZER_DURACION_MS)) { // Si ya cumplió 5s
  digitalWrite(BUZZER_PIN, LOW); // Apaga buzzer
  buzzerActivo = false;          // Marca inactivo
  Serial.println("[BUZZER] Tiempo cumplido -> buzzer OFF");
}

if (nuevoEstado != "lleno") { // Si NO está lleno
  if (!enEstadoNoLleno) {     // Si recién entra a “no lleno”
    enEstadoNoLleno = true;   // Marca estado “no lleno”
    inicioNoLlenoMs = millis(); // Guarda inicio del conteo
  } else {                    // Si ya estaba en “no lleno”
    if (!buzzerArmado && (millis() - inicioNoLlenoMs >= TIEMPO_MIN_NO_LLENO_MS)) { // Si pasó 30s
      buzzerArmado = true;    // Rearma el buzzer
      Serial.println("[BUZZER] Rearmado tras tiempo en nivel no lleno");
    }
  }
} else { // Si volvió a lleno
  enEstadoNoLleno = false; // Reinicia seguimiento del estado no lleno
}
```

Fuente: Autoría propia

Dando continuación a la programación llegamos a la parte de a lógica de dispensación de comida como podemos observar en la **Figura *41***, para ello lo primero que se hace es la sincronización con la hora local mediante el protocolo NTP, una vez que se verifica que la hora, minutos y segundos sean los actuales, se definen los horarios de dispensación del alimento, como podemos ver los horarios que se establecen son a las 08:00 am, 12:00pm y 16:00pm.

Durante el transcurso del día si la hora actual coincide con alguno de estos valores y el tiempo marca exactamente minuto 0 y segundo 0, se dispensa el alimento es decir activa la función dispensar(), una vez realizado esto el sistema rearma la condición de ejecución, permitiendo que la dispensación pueda realizarse nuevamente en el siguiente horario establecido.

Figura 41

Horarios de dispensación de comida, (dispensador de alimento)

```
// ************************ HORARIO DE DISPENSACIÓN ******************************
struct tm timeinfo;                 // Estructura para hora local
if (getLocalTime(&timeinfo)) {      // Si hay hora válida
  int hora = timeinfo.tm_hour;      // Hora actual
  int minuto = timeinfo.tm_min;     // Minuto actual
  int segundo = timeinfo.tm_sec;    // Segundo actual

  bool esHoraProgramada =
    ((hora == 8) || (hora == 12) || (hora == 16)); // Horas programadas

  if (esHoraProgramada && minuto == 0 && segundo == 0 && !ejecutado) { // Justo a la hora exacta
    dispensar();             // Ejecuta la dispensación
    ejecutado = true;        // Marca para no repetir en ese segundo
  }

  if ((!esHoraProgramada || minuto != 0 || segundo != 0) && ejecutado) { // Cuando ya pasó el instante exacto
    ejecutado = false;       // Rearma para el siguiente horario
  }
}
```

Fuente: Autoría propia

3.6.3.2 Subsistema dispensador de agua

Como se puede apreciar en la **Figura *42***, podemos observar la sección inicial del código del sistema implementado, comienza con la incorporación de las librerías necesarias para el funcionamiento de microcontrolador ESP32, así mismo en esta parte del programa se establece la base sobre la cual opera todo el sistema, la cual es la habilitación de la comunicación inalámbrica mediante WiFi , así mismo se integra la librería de Firebase, además incluimos la la librería del tiempo el cual nos ayudará a la hora de trabajar con horarios definidos mediante la sincronización con servidores NTP y la librería para comunicaciones seguras mediante HTTPS el cual nos servirá para enviar notificaciones por telegram, como último punto tenemos la definición de los parámetros de la red WiFi como son el nombre de la red WiFi: INNO-FIBER- FLIA PUPIALES y la contraseña: 1704Pupiales#, con esto permitiremos la conexión automática y un funcionamiento continuo, de igual modo siguiendo las líneas de código tenemos la parte de establecimiento de credenciales de Firebase la cual incluye el API Key y la URL de RTDB los cuales nos permitirán el intercambio de datos entre el ESP32 y la plataforma de la nube.

Figura 42

Inclusión de librerías y credenciales WiFi/Firebase, (dispensador de agua)

```
// **********************LIBRERÍAS***********************************
#include <Arduino.h>                      // Funciones base de Arduino
#include <WiFi.h>                         // Conexión WiFi del ESP32
#include <WiFiClientSecure.h>             // Cliente HTTPS para telegram
#include <Firebase_ESP_Client.h>          // Librería para Firebase Realtime Database
#include <time.h>                         // Manejo de hora con NTP

// **********************CREDENCIALES WiFi*********************************
#define WIFI_SSID     "INNO-FIBER- FLIA PUPIALES"             // Nombre de la red WiFi
#define WIFI_PASSWORD "1704Pupiales#"                         // Contraseña del WiFi

// **********************CREDENCIALES FIREBASE*********************************
#define API_KEY       "AIzaSyB3D4sciHu5MYbBuQS5m9STZSm2mHy0Kh4"// API Key del proyecto Firebase
#define DATABASE_URL  "https://criaderocj-87fb9-default-rtdb.firebaseio.com/" // URL de RTDB
```

Fuente: Autoría propia

Siguiendo con las líneas de código tenemos la asignación de pines del ESP32 hacia los diferentes componentes que integrarán el subsistema de dispensación de agua, como podemos observar en la **Figura *43***, donde se definen los pines TRIG GIO14 y ECHO GIO35 correspondientes al sensor ultrasónico HC-SR04 el cual permitirá calcular el nivel de agua almacenado en el tanque. De igual forma podemos observar que el sensor ultrasónico del bebedero utiliza el pin GPIO25 como salida TRIG y el pin GPIO33 como entrada ECHO, Así mismo tenemos el buzzer el cual se encuentra conectado al GPIO13 el cual nos servirá para generar alertas sonoras según el estado del sistema.

Para la bomba 1, encargada de transferir agua desde el tanque hacia el bebedero, estará conectada al GPIO27. De igual modo para la bomba 2 se controla mediante el GPIO32, utilizada para el drenaje y la limpieza del bebedero, cabe mencionar que estos pines estarán conectados al Relé para activar o desactivar las bombas de agua, como punto final tenemos el sensor infrarrojo el cual se conecta al GPIO34 el cual nos permitirá realizar el conteo de huevos.

Figura 43

Asignación de pines, (dispensador de agua)

```
// ********************** PINES DE CONEXIÓN ****************************
// Sensor 1 (TANQUE)
#define TRIG1_PIN 14                    // TRIG del ultrasónico del tanque
#define ECHO1_PIN 35                    // ECHO del ultrasónico del tanque
// Sensor 2 (BEBEDERO)
#define TRIG2_PIN 25                    // TRIG del ultrasónico del bebedero
#define ECHO2_PIN 33                    // ECHO bebedero

#define BUZZER_PIN        13            // Pin del buzzer
#define BOMBA_1_PIN       27            // Bomba 1: tanque  hacia el bebedero
#define BOMBA_2_PIN       32            // Bomba 2: drenaje/limpieza
#define SENSOR_HUEVOS_PIN 34            // Sensor infrarrojo para contar huevos

```

Fuente: Autoría propia

A continuación en la **Figura *44***, podemos observar cómo se definen los límites que permitirán clasificar los niveles de agua, controlar las alertas y la activación de las bombas, siguiendo con el código tenemos la parte del tanque , aquí se definen umbrales de acuerdo a la distancia medida desde el sensor hasta el nivel del agua, si la distancia es menor o igual a 10 cm, el tanque estará lleno y se activará el buzzer , los otros rangos intermedios como medio y bajo, hasta llegar al estado crítico, este último estado se define entre 26 y 30 cm, pero si la distancia es igual o mayor a los 30cm se define como un tanque vacío, si el tanque llega al estado de vacío el sistema de dispensación de agua se bloquea el funcionamiento de las bombas por seguridad, ya que una bomba sin agua suele correr peligro de daños.

Para el bebedero se establecen dos umbrales donde si la distancia medida es mayor o igual a 6 cm, el sistema interpreta que el nivel es bajo y dará paso a la recarga de agua, pero si la distancia es menor o igual a 4cm, el bebedero se considera lleno y la recarga se detiene.

Siguiendo con el código se establecen límites máximos tanto para el proceso de llenado y drenaje, en el cual, para el llenado de la bomba del tanque el tiempo máximo es de 2 cm y la bomba en el bebedero dedicada al drenaje será de 90 segundos, esto se realizó con el fin de evitar que las bombas permanezcan activas si en caso se presentara algún fallo del sistema,

siendo una especie de seguridad con el fin de evitar desbordamientos y daños en las bombas de agua.

Figura 44

Definición de alturas y umbrales del tanque y bebedero, (dispensador de agua)

```
// **********************UMBRALES Y ALTURAS******************
// **********************TANQUE******************************
const int TANQUE_BUZZER_CM        = 10;    // <=10 cm: Estado lleno se activa buzzer
const int TANQUE_REINICIO_CM      = 10;    // Si vuelve a este valor o menos, se desbloquea el sistema
const int TANQUE_LLENO_MAX_CM     = 15;
const int TANQUE_MEDIO_MIN_CM     = 16;
const int TANQUE_MEDIO_MAX_CM     = 20;
const int TANQUE_BAJO_MIN_CM      = 21;
const int TANQUE_BAJO_MAX_CM      = 25;
const int TANQUE_CRITICO_MIN_CM   = 26;
const int TANQUE_CRITICO_MAX_CM   = 30;
const int TANQUE_PARADA_UMBRAL_CM = 30;    // >30 cm: se asume vacío y se bloquean bombas

// ******************Alturas desde el sensor hasta el fondo****************************************
const int ALTURA_TOTAL_TANQUE   = 32;      // Altura real del tanque en cm
const int ALTURA_TOTAL_BEBEDERO = 7;       // Altura real del bebedero en cm

//************************ UMBRALES BEBEDERO **************************
const int FILL_START_CM = 6;               // Si distancia >=6: iniciar recarga
const int FILL_STOP_CM  = 4;               // Si distancia <=4: detener recarga (lleno)
const int BEBEDERO_VACIO_CM = FILL_START_CM;// Punto donde se considera 0% (vacío)

const unsigned long MAX_FILL_MS  = 2UL * 60UL * 60UL * 1000UL; // Tope de tiempo llenando
const unsigned long MAX_DRAIN_MS = 90UL * 1000UL;              // Tope de tiempo drenando
```

Fuente: Autoría propia

Siguiendo con la línea de código tenemos en la **Figura *45***, la parte de conteo de huevos donde se implementa una interrupción (ISR) la cual se ejecuta solo cuando el haz del sensor es interrumpido por el paso de un huevo, cuando se da la interrupción, se registra el momento en que ocurre el pulso y se descarta aquellos que se repiten en menos de 1 milisegundo, esto es porque son considerados como ruido. Además, se ignoran las señales que aparecen justo después de encender las bombas o al momento en que se apagan las bombas, con esto igual se evita que las interferencias eléctricas sean confundidas con el paso de huevos. A continuación, se verifica la estabilidad de la señal leyendo el sensor 3 veces en un tiempo muy corto, en el cual si la señal se mantiene en LOW de forma constate, el pulso se acepta como válido o si no

se lo descarta como ruido, por último, si el pulso es válido se activa una bandera para que el conteo de huevos se realice en el loop principal.

Figura 45

Función de interrupción para el conteo de huevos, (dispensador de agua)

```
// **********************ISR: INTERRUPCIÓN PARA CONTEO DE HUEVOS*************************
void IRAM_ATTR huevoISR() {                                  // ISR: se ejecuta apenas cae el pulso (FALLING)
  unsigned long ahora = millis();                            // Guarda el tiempo exacto del evento
  static unsigned long ultimoISR = 0;                        // Ayuda a filtrar rebotes mínimos
  if (ahora - ultimoISR < 1) return;                         // Anti-rebote rápido (~1ms)
  ultimoISR = ahora;                                         // Actualiza el último pulso

  if (ahora - ultimoCambioBombasMs < IGNORE_AFTER_BOMBA_MS) return; // Ignora ruido tras prender/apagar bombas

  int lows = 0;                                              // Cuenta cuántas lecturas LOW se mantienen
  for (int i = 0; i < 3; i++) {                               // Repite 3 veces para confirmar estabilidad
    if (digitalRead(SENSOR_HUEVOS_PIN) == LOW) lows++;        // Si sigue LOW, suma
    delayMicroseconds(10);                                   // Micro-espera para validar que no fue pico
  }
  if (lows < 2) return;                                      // Si no se mantuvo LOW, se descarta como ruido

  huevoEdgeMs   = ahora;                                     // Guarda el instante del pulso válido
  huevoEdgeFlag = true;                                      // Marca bandera para que loop lo procese
}
```

Fuente: Autoría propia

A continuación tenemos la **Figura *46***, donde podemos observar la función encargada de la conexión a la red WiFi para ello hace uso de las credenciales definidas anteriormente, Además con el fin de evitar que el sistema se bloquee, se establece un tiempo máximo de espera de 15 segundos, el cual como se puede ver se controla mediante la función millis(), aquí durante este tiempo, el programa verifica el estado de la conexión sin detener la ejecución de las demás tareas, seguido a esto si la conexión se realiza correctamente se conectará a la red, y logrará una sincronización de la hora local mediante el protocolo NTP, además, nos mostrará la dirección IP del ESP32, sino el sistema seguirá funcionando en modo offline con el fin de que nunca pare el funcionamiento del sistema en cuando a la dispensación de agua, es por esto que se implementa un mecanismo de reconexión no bloqueante cuyo objetivo será el de intentar restablecer la conexión cada 30 segundos sin afectar el funcionamiento general de nuestro sistema.

Figura 46

Función de conexión y reconexión WiFi, (dispensador de agua)

```
// ************************************FUNCIONES DE WIFI**********************************
void conectarWiFiInicial() {                                  // Conecta con un límite de tiempo
  WiFi.mode(WIFI_STA);                                        // Modo estación (cliente)
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);                       // Inicia conexión con credenciales
  Serial.print("[WiFi] Conectando");                          // Muestra estado

  unsigned long inicio = millis();                            // Marca inicio
  while (WiFi.status() != WL_CONNECTED && millis() - inicio < 15000) { // Espera máximo 15s
    delay(500);                                               // Pausa para no saturar CPU
    Serial.print(".");                                        // Indicador visual
  }
  Serial.println();                                           // Salto de línea

  if (WiFi.status() == WL_CONNECTED) {                        // Si conectó
    Serial.print("[WiFi] Conectado. IP: ");                   // Mensaje ok
    Serial.println(WiFi.localIP());                           // Imprime IP
  } else {
    Serial.println("[WiFi] No se pudo conectar en 15s. Sigo OFFLINE (solo local)."); // Modo offline
  }
}

void reintentarWiFiNoBloqueante(unsigned long now) {          // Reintenta sin detener el sistema
  if (WiFi.status() == WL_CONNECTED) return;                  // Si ya está conectado, no hace nada
  if (now - ultimoIntentoWiFiMs < 30000) return;              // Reintenta cada 30 segundos

  Serial.println("[WiFi] Desconectado, reintentando...");     // Log
  WiFi.disconnect();                                          // Limpia estado previo
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);                       // Intenta de nuevo
  ultimoIntentoWiFiMs = now;                                  // Guarda el momento del intento
}
```

Fuente: Autoría propia

En la **Figura *47***, tenemos la parte encargada de la comunicación entre nuestro ESP32 y la base de datos, para ello lo primero que tenemos es la parte de iniciarFirebase() la cual se encarga de inicializar Firebase para ello debe existir conexión WiFi, seguido tenemos la parte donde se configuran las credenciales como la clave API y la URL de la base de datos en tiempo real, seguido se da paso a la inicialización el cual si se logra con éxito, firebase quedará habilitado de igual modo se activará la reconexión automática a WiFi

Sin embargo en caso de no tener conexión se emplea la función reintentarFirebaseNoBloqueante() lo que nos permitirá reintentar la conexión cada 60 segundos.

Figura 47

Función de conexión de Firebase, (dispensador de agua)

```
//*********************************FUNCIONES: FIREBASE*********************************
void iniciarFirebase() {                                    // Inicializa Firebase si hay WiFi
  if (WiFi.status() != WL_CONNECTED) {                      // Si no hay WiFi, se evita iniciar
    Serial.println("[Firebase] No inicio: no hay WiFi.");   // Log
    firebaseReady = false;                                  // Bandera en falso
    return;                                                 // Sale
  }

  Serial.println("[Firebase] Inicializando...");            // Log
  config.api_key      = API_KEY;                            // Configura API key
  config.database_url = DATABASE_URL;                       // Configura URL de RTDB

  if (Firebase.signUp(&config, &auth, "", "")) {            // Signup anónimo
    signupOK = true;                                        // Bandera ok
    Serial.println("[Firebase] SignUp OK");                 // Log
  } else {
    signupOK = false;                                       // Bandera fail
    firebaseReady = false;                                  // Firebase no listo
    Serial.print("[Firebase] SignUp ERROR: ");              // Log
    Serial.println(config.signer.signupError.message.c_str());// Error real
    return;                                                 // Sale para no continuar
  }

  Firebase.begin(&config, &auth);                           // Inicia Firebase
  Firebase.reconnectWiFi(true);                             // Permite reconexión automática
  firebaseReady       = true;                               // Marca listo
  ultimoOkFirebaseMs  = millis();                           // Guarda instante ok
  Serial.println("[Firebase] Listo (modo mejor esfuerzo).");// Log final
}

void reintentarFirebaseNoBloqueante(unsigned long now) {    // Reintenta Firebase cada cierto tiempo
  if (firebaseReady) return;                                // Si ya está listo, no reintenta
  if (now - ultimoIntentoFbMs < 60000) return;              // Reintenta cada 60 segundos
```

Fuente: Autoría propia

Siguiendo con las líneas de código en la **Figura *48***, llegamos a la parte de las funciones para envío de mensajes por Telegram, como primer punto tenemos la función agregarMensajeTelegram() esta función se encarga de almacenar cada mensaje en la cola de envío, si la cola está llena, descarta el mensaje con lo cual se evitar sobrecargar el sistema, seguido a esto tenemos la función, procesarColaTelegram() el cual se encarga de enviar los mensajes pendientes, verificando que exista conexión WiFi y respetando un intervalo mínimo de 2 segundos entre envíos, siguiendo las líneas de código llegamos a la función enviarTelegram() para lo cual antes de entrar se debe verificar la conexión a internet, si existe entonces se codifica el contenido del mensaje. Como última parte del código tenemos la

función manejarCambioWifi() el cual está al pendiente del estado de la conexión WiFi, en caso de desconexión, la cola de mensajes se limpia automáticamente para evitar el envío de mensajes desactualizados.

Figura 48

Funciones para el envío de notificaciones por telegram, (dispensador de agua)

```
// ********************* FUNCIONES PARA TELEGRAM*************************************

void agregarMensajeTelegram(const String &mensajeCodificado) { // Encola un mensaje codificado
  int siguiente = (finCola + 1) % MAX_COLA;                    // Calcula siguiente posición circular
  if (siguiente == inicioCola) {                               // Si la cola está llena
    Serial.println("[Telegram] Cola llena, mensaje descartado (para mantener orden)."); // Aviso
    return;                                                    // Sale sin guardar
  }
  colaMensajes[finCola].texto = mensajeCodificado;             // Guarda en la cola
  finCola = siguiente;                                         // Avanza fin
}

void procesarColaTelegram() {                                  // Envía mensajes respetando tiempos
  if (inicioCola == finCola) return;                           // Si cola vacía, sale
  if (millis() - ultimoEnvioTelegram < INTERVALO_TELEGRAM_MS) return; // Respeta 2s
  if (WiFi.status() != WL_CONNECTED) return;                   // Sin WiFi, no envía

  WiFiClientSecure client;                                     // Cliente HTTPS
  client.setInsecure();                                        // Evita validación de certificado
  if (client.connect("api.telegram.org", 443)) {               // Conexión al servidor
    String url = "https://api.telegram.org/bot8001400737:AAGn-SIVTGkDmIAVesbpZxg1uQzekGyeZgg/sendMessage?chat_id=5585325975&text=" +
                 colaMensajes[inicioCola].texto;               // URL completa con mensaje

    client.print("GET " + url + " HTTP/1.1\r\nHost: api.telegram.org\r\nConnection: close\r\n\r\n"); // Envía GET
    delay(100);                                                // Espera corta para completar envío
    client.stop();                                             // Cierra conexión

    inicioCola = (inicioCola + 1) % MAX_COLA;                  // Saca el mensaje enviado
    ultimoEnvioTelegram = millis();                            // Actualiza momento del envío
  } else {
    Serial.println("[Telegram] No se pudo conectar a api.telegram.org, se reintentará."); // Queda en cola
  }
}
```

Fuente: Autoría propia

En la **Figura *49***, podemos ver las funciones para el control de las bombas del sistema, como primer punto tenemos la función bomba1On() la cual gestiona la bomba que se encuentra en el tanque de agua misma que es utilizada para la recarga del bebedero como se puede observar esta función actúa únicamente cuando se detecta un cambio real en el estado de la bomba , Además, cuando inicia la recarga del bebedero se envían notificaciones por Telegram al coturnicultor . Siguiendo con las líneas de código tenemos, la función bomba2On() la cual controla la bomba destinada al drenaje para la limpieza, sigue la misma lógica que la bomba 1.

Figura 49

Funciones de control para bombas de recarga y drenaje, (dispensador de agua)

```
340  //****************************** FUNCIONES PARA BOMBAS*************************************
341  void bomba1On(bool on) {                                    // Control de bomba 1 (recarga)
342    if (bomba1Estado != on) {                                 // Solo si cambia realmente
343      ultimoCambioBombasMs = millis();                        // Marca cambio para filtrar ruido
344      bomba1Estado = on;                                      // Guarda el nuevo estado
345
346      if (on) enviarTelegram("🔄💧Iniciando recarga del bebedero"); // Aviso de encendido
347      else   enviarTelegram("✅💧Bebedero recargado satisfactoriamente"); // Aviso de apagado
348    }
349    digitalWrite(BOMBA_1_PIN, on ? LOW : HIGH);               // Relé activo en LOW
350  }
351
352  void bomba2On(bool on) {                                    // Control de bomba 2 (drenaje)
353    if (bomba2Estado != on) {                                 // Solo si cambió
354      ultimoCambioBombasMs = millis();                        // Marca cambio para ventana ciega
355      bomba2Estado = on;                                      // Guarda estado
356    }
357    digitalWrite(BOMBA_2_PIN, on ? LOW : HIGH);               // Relé activo en LOW
358  }
```

Fuente: Autoría propia

Siguiendo con el código en la **Figura *50***, podemos ver el bloque encargado de medir la distancia con el sensor ultrasónico, como primer punto, se realiza la lectura generando el pulso de disparo y midiendo el tiempo de retorno del eco, el cual se convierte a centímetros y se valida descartando valores fuera de un rango físico razonable, seguido a esto, se toman varias muestras y se calcula la mediana, A continuación, se aplica un promedio recortado para lo cual se consideran valores cercanos a la mediana, como último punto el valor resultante se suaviza mediante un promedio móvil exponencial (EMA), como resultado tendremos una distancia válida.

Figura 50

Lectura y estabilización de la lectura para sensores ultrasónicos, (dispensador de agua)

```
//************************FUNCIONES SENSORES ULTRASÓNICOS*****************************
static float quickSelect(float* arr, int n, int k) {           // Selección de k-ésimo (mediana) sin ordenar
  int left = 0, right = n - 1;                                 // Límites
  while (true) {                                               // Repite hasta encontrar k
    if (left == right) return arr[left];                       // Caso base
    float pivot = arr[(left + right) / 2];                     // Pivote
    int i = left, j = right;                                   // Índices
    while (i <= j) {                                           // Partición
      while (arr[i] < pivot) i++;                              // Avanza
      while (arr[j] > pivot) j--;                              // Retrocede
      if (i <= j) {                                            // Intercambio
        float t = arr[i]; arr[i] = arr[j]; arr[j] = t;         // Swap
        i++; j--;                                              // Mueve índices
      }
    }
    if (k <= j)      right = j;                                // Busca a la izquierda
    else if (k >= i) left = i;                                 // Busca a la derecha
    else             return arr[k];                             // Ya quedó ubicado
  }
}

float leerDistanciaCrudaCM(uint8_t trigPin, uint8_t echoPin) { // Lectura directa del ultrasónico
  digitalWrite(trigPin, LOW);                                  // Pulso limpio
  delayMicroseconds(3);                                        // Estabiliza
  digitalWrite(trigPin, HIGH);                                 // Disparo
  delayMicroseconds(10);                                       // Duración del TRIG
  digitalWrite(trigPin, LOW);                                  // Fin

  long duracion = pulseIn(echoPin, HIGH, 25000);               // Tiempo del eco con timeout
  if (duracion <= 0) return 0.0f;                              // Si no midió, devuelve 0

```

Fuente: Autoría propia

En la **Figura *51*** , podemos ver que la función verifica que el sistema no se encuentre bloqueado por falta de agua en el tanque ya que, si está bloqueado el proceso de limpieza no se realiza, seguido se obtiene la fecha y hora actual gracias a la sincronización con el protocolo NTP.

Una vez que se sincroniza se da la orden de que la limpieza se realice cada miércoles a las 05:00., además se envía una notificación al usuario por Telegram donde se informa al coturnicultor que se realizó el inicio del proceso de limpieza, así como su finalización.

Figura 51

Limpieza del bebedero mediante drenaje, (dispensador de agua)

```
448  //*****************************FUNCIONES PARA LIMPIEZA PROGRAMADA (MIÉRCOLES 05:00) *********
449  void gestionarProgramacionDrenaje() {                          // Lanza limpieza semanal del bebedero
450    if (bloqueoTanque) return;                                   // Si tanque vacío, no se limpia
451
452    struct tm tinfo;                                             // Estructura de fecha/hora
453    if (!getLocalTime(&tinfo)) return;                           // Sin NTP, no programa nada
454
455    int dia  = tinfo.tm_mday;                                    // Día del mes
456    int wday = tinfo.tm_wday;                                    // Día de semana (0 domingo)
457
458    if (dia != ultimoDiaRevisado) {                              // Si cambió el día
459      ultimoDiaRevisado = dia;                                   // Actualiza referencia
460      drenajeLanzadoHoy = false;                                 // Permite ejecutar si corresponde
461    }
462
463    bool esDiaLimpieza = (wday == 3);                             // Miércoles
464    if (!esDiaLimpieza) return;                                   // Si no es miércoles, sale
465
466    if (!drenajeLanzadoHoy && tinfo.tm_hour == 5 && tinfo.tm_min == 0) { // 05:00 exacto
467      bomba1On(false);                                           // Asegura llenado apagado
468      bomba2On(false);                                           // Asegura drenaje apagado
469
470      estado = Estado::DRAIN_CYCLE;                               // Entra a estado de drenaje
471      estadoInicioMs = millis();                                  // Marca inicio
472      bomba2On(true);                                            // Enciende bomba 2
473
474      drenajeLanzadoHoy = true;                                  // Evita repetir en el mismo día
475
476      Serial.println("[LIMPIEZA] Inicio limpieza miércoles 05:00 (Bomba 2 ON)"); // Log
477      enviarTelegram("🔄💧👇Inicio de limpieza del bebedero");     // Aviso Telegram
478    }
479  }
```

Fuente: Autoría propia

Siguiendo con el código, en la **Figura *52***, tenemos la inicialización general del sistema, aquí se configura el monitor serial para depuración y se definen los pines de entrada y salida correspondientes a los sensores ultrasónicos que estarán presentes en el tanque y el bebedero, además del buzzer, las bombas y el sensor infrarrojo. Luego de esto el sistema inicia en estado seguro, apagando buzzer y bombas, además reinicia las variables de conteo y producción diaria.

A continuación, el ESP32 intenta conectarse a la red WiFi e inicializa Firebase en caso de que si haya conexión a internet se sincroniza la hora local mediante el protocolo NTP, seguido a esto si el servicio de firebase está disponible, se envía el conteo inicial a la base de datos y a su vez se envía una notificación por Telegram mencionando que el sistema se ha iniciado correctamente.

Figura 52

Inicialización del sistema, (dispensador de agua)

```
516  //******************** INICIALIZACIÓN GENERAL SETUP************************************
517  void setup() {
518    Serial.begin(115200);                                    // Inicia monitor serial
519
520    pinMode(TRIG1_PIN, OUTPUT);                              // TRIG tanque
521    pinMode(ECHO1_PIN, INPUT);                               // ECHO tanque
522    pinMode(TRIG2_PIN, OUTPUT);                              // TRIG bebedero
523    pinMode(ECHO2_PIN, INPUT);                               // ECHO bebedero
524
525    pinMode(BUZZER_PIN, OUTPUT);                             // Buzzer salida
526    pinMode(BOMBA_1_PIN, OUTPUT);                            // Bomba 1 salida
527    pinMode(BOMBA_2_PIN, OUTPUT);                            // Bomba 2 salida
528
529    pinMode(SENSOR_HUEVOS_PIN, INPUT);                       // Sensor huevos entrada
530
531    digitalWrite(BUZZER_PIN, LOW);                           // Buzzer apagado
532    bomba1On(false);                                         // Bomba 1 apagada
533    bomba2On(false);                                         // Bomba 2 apagada
534
535    contadorHuevos        = 0;                               // Reinicia conteo
536    huevosDia             = 0;                               // Reinicia producción diaria
537    diaActualProduccion   = -1;                              // Fuerza reinicio de fecha
538    huevosDirty           = true;                            // Pendiente de sincronizar
539
540    ultimoHuevoValidoMs = 0;                                 // Reinicia referencia
541
542    Serial.println("[INFO] Sistema iniciado. Conteo de huevos = 0."); // Log
543
544    attachInterrupt(digitalPinToInterrupt(SENSOR_HUEVOS_PIN), huevoISR, FALLING); // ISR por flanco de bajada
545
546    conectarWiFiInicial();                                   // Conexión inicial WiFi
547    wifiEstabaConectado = (WiFi.status() == WL_CONNECTED);   // Guarda estado inicial
548
```

Fuente: Autoría propia

Siguiendo la lógica del código, en la **Figura *53***, podemos ver la parte principal del sistema donde tenemos el loop() aquí se ejecuta el funcionamiento continuo del sistema , como primer punto se verifica el estado del WiFi, en caso de tener acceso a WiFi los mensajes de envían de forma ordenada , seguido de esto el sistema procesa el conteo de huevos a partir de los datos obtenidos por la interrupción del sensor infrarrojo, si el pulso es válido, se incrementará el conteo diario además se actualizan los datos en Firebase, de igual modo cuando se llegue 55 huevos se enviará una notificación a telegram como alerta previa , posteriormente cuando se llegue a 60 huevos, se enviara una notificación y se reiniciara el conteo. De igual modo, en este último bloque de programación podemos ver que se obtienen lecturas estables del nivel del tanque y del bebedero gracias a esto se detecta niveles como lleno, crítico, y de

acuerdo a esto se envían las notificaciones, si en caso el nivel detectado en el tanque es vacío se bloquea el funcionamiento. También podemos ver que el bebedero se gestiona mediante una máquina de estados que controla la recarga y el drenaje según el nivel de agua y los tiempos establecidos.

Como punto final el sistema enviará constantemente los niveles de agua, y el conteo de huevos a Firebase para posteriormente ver estos datos en un dashboard.

Figura 53

Lógica principal de operación del sistema, (dispensador de agua)

```
563  // ********************EJECUCIÓN PRINCIPAL************************
564  void loop() {
565    unsigned long now = millis();                                  // Tiempo actual del ciclo
566
567    manejarCambioWifi();                                           // Si cae WiFi, limpia cola
568
569    bool          hayEdge   = false;                               // Bandera local del pulso
570    unsigned long edgeTime  = 0;                                   // Tiempo del pulso
571
572    noInterrupts();                                                // Protege lectura de variables ISR
573    if (huevoEdgeFlag) {                                           // Si ISR marcó un huevo
574      hayEdge       = true;                                        // Copia bandera
575      edgeTime      = huevoEdgeMs;                                 // Copia tiempo
576      huevoEdgeFlag = false;                                       // Limpia bandera global
577    }
578    interrupts();                                                  // Restaura interrupciones
579
580    if (hayEdge) {                                                 // Si hubo huevo válido
581      bool bombasOn = bomba1Estado || bomba2Estado;                // Revisa si bombas están encendidas
582
583      bool intervaloOK = true;                                     // Por defecto, válido
584      if (bombasOn) {                                              // Solo filtra si bombas ON
585        unsigned long delta = edgeTime - ultimoHuevoValidoMs;      // Tiempo entre eventos
586        if (ultimoHuevoValidoMs != 0 && delta < MIN_INTERVAL_CON_BOMBA) { // Si es demasiado rápido
587          intervaloOK = false;                                     // Se descarta
588          Serial.print("[HUEVOS] Pulso ignorado por intervalo con bomba, delta="); // Log
589          Serial.print(delta);                                     // Muestra delta
590          Serial.println(" ms");                                   // Unidades
591        }
592      }
593
594      if (intervaloOK) {                                           // Si se acepta como huevo real
595        ultimoHuevoValidoMs = edgeTime;                            // Guarda el último huevo válido
621        if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED) { // Si Firebase disponible
622          if (Firebase.RTDB.setInt(&fbdo, "/esp32_2/conteo_huevos", contadorHuevos)) { // Sube conteo
623            huevosDirty        = false;                            // Ya sincronizado
624            ultimoOkFirebaseMs = edgeTime;                         // Marca último OK
625          }
```

Fuente: Autoría propia

3.6.4 Desarrollo de la interfaz web

La interfaz web desarrollada corresponde principalmente a un dashboard de monitoreo el cual es una herramienta fundamental para la visualización y el monitoreo por parte del coturnicultor hacia el estado en tiempo real de los contenedores y tanques de agua, así como el control de la cantidad de huevos producidos diariamente, es por tal motivo que se diseñó e implementó un dashboard web accesible desde cualquier navegador con conexión a Internet, con ello garantizamos el acceso desde distintos dispositivos como computadoras, tablets o teléfonos móviles, para el caso del coturnicultor será muy útil ya que cuenta con un teléfono celular personal, este dashboard nos mostrará los datos almacenados en Firebase Realtime Database, ya que su uso facilita la comunicación bidireccional entre el dashboard y los microcontroladores ESP32, asegurando que los cambios registrados en el sistema se reflejen de manera inmediata en el dashboard.

Para su desarrollo se hizo uso de la herramienta Visual Studio Code, como entorno de programación, para ello se crearon dos archivos, el primero un archivo HTML encargado del diseño y la parte visual del dashboard mientras que el segundo archivo creado fue un archivo JavaScript(index.js) en cual se encarga de la lógica de funcionamiento, la parte de conexión con Firebase Realtime Database y por ende la actualización de los datos visualizados, como punto final mencionar que los códigos completos correspondientes al desarrollo del dashboard se presentan en el **Anexo E**.

3.6.4.1 Estructura e integración de la interfaz web

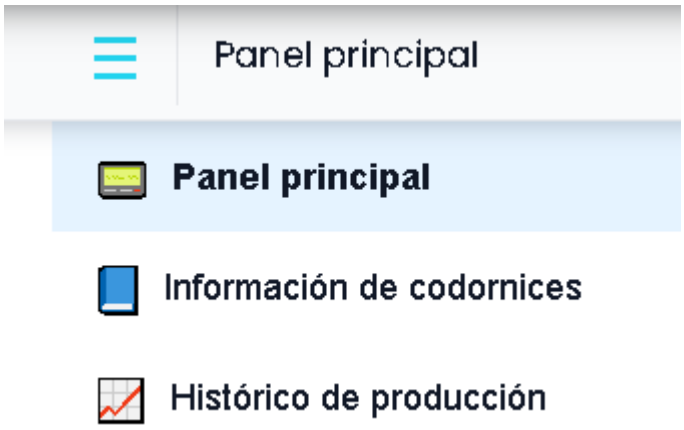
En cuanto a la estructura del dashboard, se lo ha dividido en tres ventanas con el objetivo de organizar la información y facilitar el monitoreo del sistema de producción como se puede observar en la **Figura *54***.

La primera ventana corresponde al panel principal, el cual permite la visualización en tiempo real del nivel del tanque de agua y del contenedor de alimento, así como la cantidad de

huevos producidos durante el día, la segunda ventana presenta información complementaria relacionada con las codornices, con el objetivo de apoyar la toma de decisiones del coturnicultor, por último, la tercera ventana corresponde al histórico de producción, donde se almacena y se visualiza la información generada en el transcurso del tiempo, además de que la información siempre estará disponible.

Figura 54

Ventanas del menú de navegación del dashboard web



Fuente: Autoría propia

- ***Panel Principal***

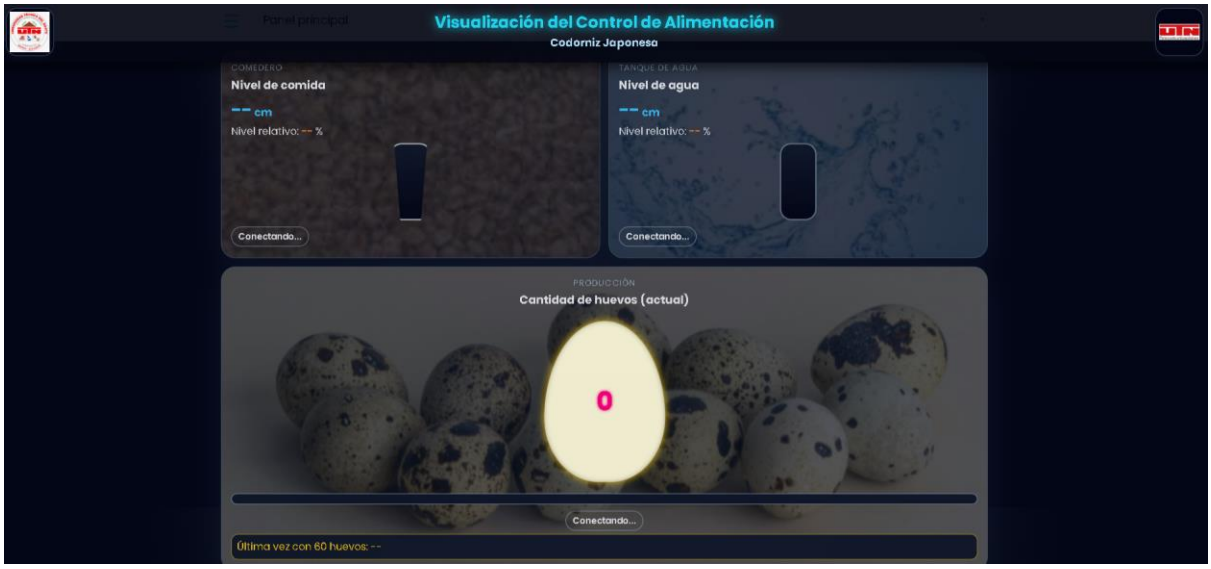
En este apartado como podemos observar en la **Figura *55***, tenemos la presencia del panel principal del dashboard web, este panel se lo realizó con el objetivo de monitorear el sistema de alimentación y agua, además de la producción de huevos del criadero de codornices, el diseño se lo realizó en base a la necesidad de mostrar los datos de una manera clara y visualmente amigable para el coturnicultor, este panel integra indicadores gráficos y bloques internos que permiten representar y visualizar en tiempo real el nivel del contenedor y tanque, uno de los bloques se ubicó en la parte superior izquierda del dashboard y esta denominado como comedero, además, se implementó el valor del nivel medido en centímetros, y el nivel relativo el cual está representado en porcentaje el cual llega de 0% a 100%, además se incluye

un indicador de estado que clasifica el estado de contenedor de cuerdo al nivel (lleno, medio, bajo, crítico y vacío). En la parte superior derecha a su vez se implementó un bloque al que se denominó tanque de agua, bloque cumple los mismos parámetros y diseño que el bloque del comedero.

En la Parte central se ha implementado otro bloque el cual se lo ha denominado producción, aquí se visualizan el número de huevos producidos y registrados hasta el momento el cual seguirá aumentando de acuerdo al número de huevos que se depositen en la bandeja recolectora, además, en la parte inferior del panel principal se ha añadido una barra de progreso asociada lo cual permitirá visualizar el avance del ciclo productivo, por último, se incorporó el registro de la última recolección correspondiente a un ciclo de 60 huevos, con esta implementación el coturnicultor podrá contar con una referencia clara sobre la fecha y hora en la que se realizó la última recolección, con esto tener un mejor control de la producción diaria y a su vez una mejor planificación de las actividades de recolección de los huevos.

Figura 55

Ventana del panel principal para el monitoreo del subsistema de alimentación, agua y producción



Fuente: Autoría propia

- ***Información de codornices***

A continuación tenemos la ventana informativa sobre los datos esenciales de la codorniz japonesa como se puede apreciar en la **Figura *56***, esta ventana se añadió con el objetivo de dar a conocer aspectos relacionados con su producción de huevos, la condiciones ambientales óptimas y los requerimientos en cuanto a la alimentación y agua que se debe suministrar por ave, gracias a esto el coturnicultor podrá contar con una referencia técnica básica lo que facilitará la comprensión de los factores que influyen en el rendimiento productivo en la producción de huevos.

Figura 56

Ventana informativa sobre los datos esenciales de la codorniz japonesa

Fuente: Autoría propia

- ***Histórico de producción***

En este apartado tenemos el histórico de producción el cual es la ventana más importante ya que permite registrar y almacenar los datos generados durante el funcionamiento del sistema, este histórico de producción se divide en varios bloques, como el de producción diaria de huevos, los ciclos completos de 60 huevos, el calendario de días con recolecciones, un resumen organizado por semanas y un bloque dedicado a la gestión e los datos. Gracias al

contenido de esta venta se podrá evaluar el desempeño productivo del criadero tras la implementación del sistema.

Dando paso al bloque de producción diaria y ciclos de 60 huevos, podemos observar en la **Figura *57***, la implementación de un mecanismo de selección temporal el cual permitirá al coturnicultor escoger el año y el mes que quiera visualizar el desempeño productivo.

Figura 57

Selección temporal para visualizar producción diaria de huevos

Fuente: Autoría propia

Siguiendo en el mismo bloque en la ***Figura 58***, tenemos la composición de la gráfica de acuerdo al mes y año seleccionado, para lo cual se ha establecido un eje horizontal el cual corresponde a un día del mes, mientras que el eje vertical indica el número de huevos producidos, el cual se lo estableció hasta 40, este valor se estableció considerando la capacidad productiva del criadero, si bien el criadero cuenta con 30 codornices, no todas las aves presentan un comportamiento productivo constante debido a diversos factores, además esta gráfica nos permitirá identificar variaciones en la producción es decir gracias a estos tendremos

una manera visual de detectar los días con baja y alta producción y analizar el comportamiento productivo a lo largo del mes seleccionado, como parte final de este bloque se añadieron dos componentes de resumen por mes, en el cual el primero muestra el número total de huevos producidos en el mes, mientras que el otro componente muestra el total de ciclos de 60 huevos producidos.

Figura 58

Gráfica del número de huevos producidos por día

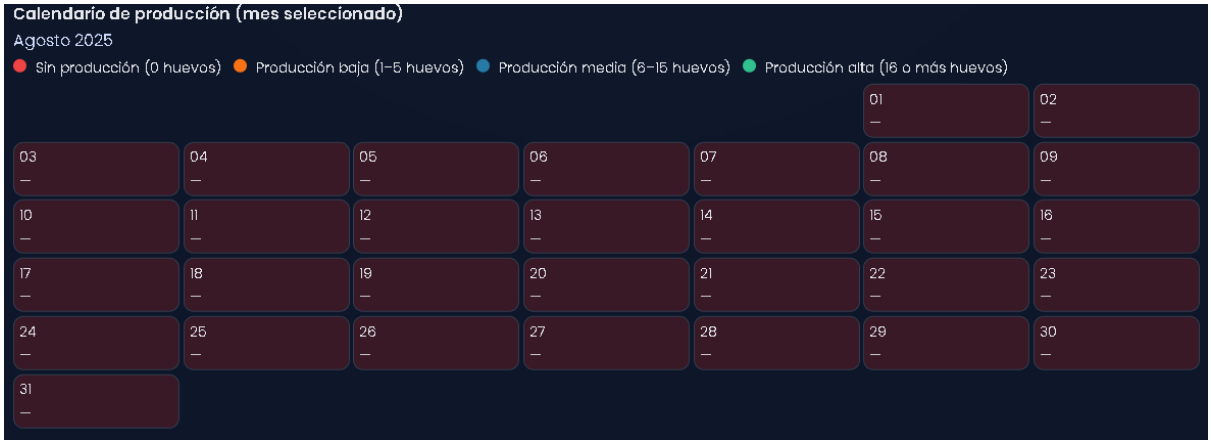
Fuente: Autoría propia

Siguiendo con el diseño del dashboard en la **Figura *59***, tenemos el bloque de calendario de producción el cual fue diseñado con el objetivo de ofrecer una forma de visualización rápida sobre el comportamiento productivo diario, para lo cual se hizo uso de una codificación por colores lo que permite clasificar cada día según el nivel de producción

alcanzado, en sí, este tipo de representación facilitará aún más a la identificación de días con una alta producción o días sin producción, sin la necesidad de interpretar gráficos detallados.

Figura 59

Calendario de registro mensual de la producción de huevos



Fuente: Autoría propia

Dando continuidad en la **Figura *60***, tenemos el bloque de registro por mes, este registro tiene un resumen el cual esta divido por semanas lo que ayudará al coturnicultor a evaluar el comportamiento de la producción en periodos específicos dentro del mes, como el número de huevos producidos durante toda la semana y cuantos ciclos completos de 60 huevos se realizaron en esa semana, con ello se podrá observar el rendimiento productivo del criadero.

En cuanto al diseño, la tabla se encuentra compuesta por cinco columnas principales, cada una con una función específica, donde la primera columna hace referencia a la semana dentro del mes, en la segunda columna tenemos el rango de días que comprende cada semana, la tercera columna presenta el total de huevos producidos durante cada semana, en la cuarta columna tenemos el valor del número de recolecciones de ciclos de 60 huevos, y la última columna muestra las fechas del momento en que se completaron dichos ciclos, con esto el coturnicultor podrá tener un respaldo para el control y seguimiento de la producción.

Figura 60

Tabla de registro mensual de producción de huevos por semanas

Registro por mes (resumen por semanas)

Semana	Rango de días	Huevos por semana	Recolecciones (ciclos de 60)	Fechas de recolección
Semana 1	Días 1–7	0	0	—
Semana 2	Días 8–14	0	0	—
Semana 3	Días 15–21	0	0	—
Semana 4	Días 22–28	0	0	—
Semana 5	Días 29–31	0	0	—
Totales del mes		0	0	

Fuente: Autoría propia

Como último bloque añadido en la parte de histórico de producción tenemos la parte de gestión de datos como se puede observar en la, este gestor de datos se lo realizó con el fin de permitir al coturnicultor administrar y depurar la información almacenada en caso de asi requerirlo, para ello lo primero que se debe realizar es seleccionar la pestaña superior derecha para abrir la ventana del gestor de datos, una vez dentro se podrá eliminar según lo requerido, para ello se lo ha organizado en opciones de eliminación por día, mes y año. Para la selección temporal se han añadido pestañas de desplazamiento, lo que permite al coturnicultor elegir de manera sencilla el periodo que desea gestionar, en el caso de la selección para los años, se definió un rango comprendido entre los años 2000 y 2100, ya que este rango se considera suficiente para cubrir el ciclo de vida útil del sistema de control de alimento, además para ejecutar la eliminación del periodo seleccionado se hizo la incorporación de dos botones de acción donde el primer botón ayuda a la eliminación de la producción diaria, en cambio el segundo botón ayudará a eliminar los ciclos de 60 huevos.

Figura 61

Gestión de datos y eliminación de producción por medio de selección temporal

Fuente: Autoría propia

Siguiendo con el bloque de gestión de los datos en la **Figura *62***, podemos observar la incorporación de dos ventanas flotantes de confirmación, los cuales servirán como un tipo de seguridad , estas ventanas se activan al seleccionar alguno de los botones tanto para eliminar la producción diaria o los ciclos de 60 huevos, solo si el coturnicultor esta de acuerdo en confirmar la opcion de eliminado se ejecutara la acción , caso contrario no ejecutará la acción.

Figura 62

Ventanas flotantes de confirmación de borrado manual de registros de producción diaria y ciclos

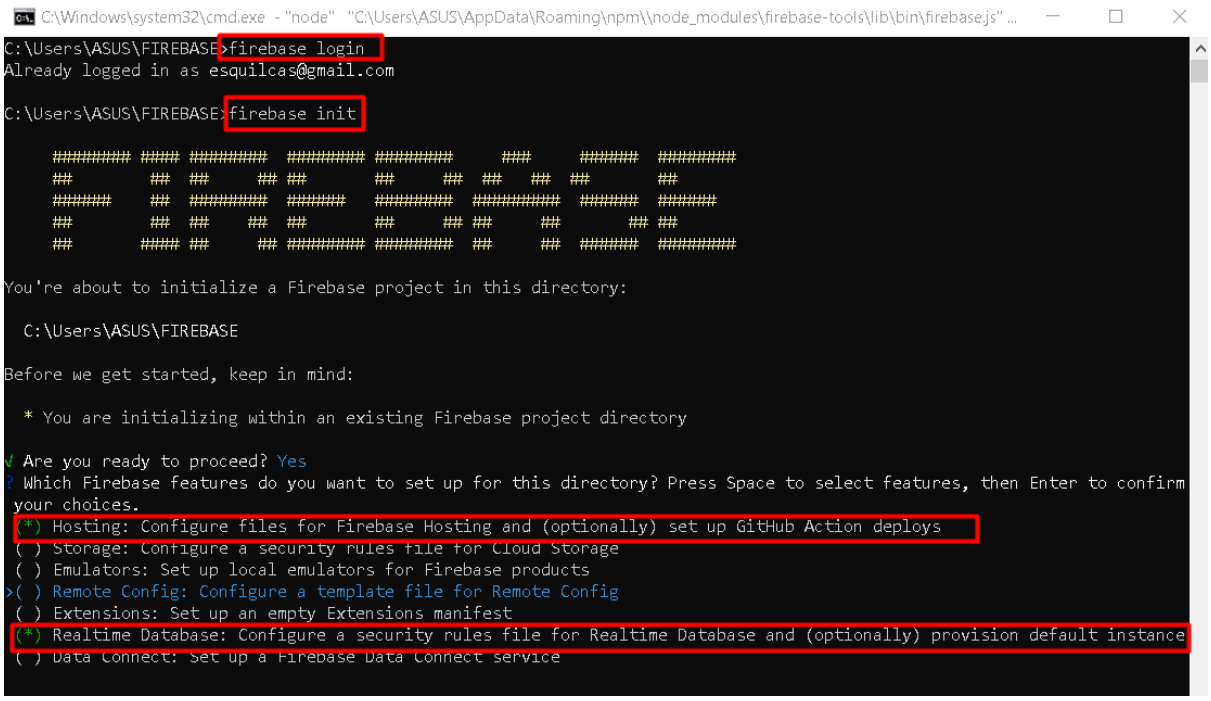
Fuente: Autoría propia

3.6.4.2 Configuración y despliegue del dashboard en Firebase Hosting

Una vez que tengamos instalado Firebase CLI, en la **Figura *63***, podemos observar la autentificación por medio del comando *firebase login*, una vez que se autentifica se da paso a la inicialización del proyecto para ello se ejecuta el siguiente comando *firebase init*, durante este proceso se selecciona servicios como Firebase Hosting y Firebase Realtime Database, así como la asociación del directorio local con el proyecto existente en Firebase.

Figura 63

Proceso de autenticación y selección de servicios de firebase

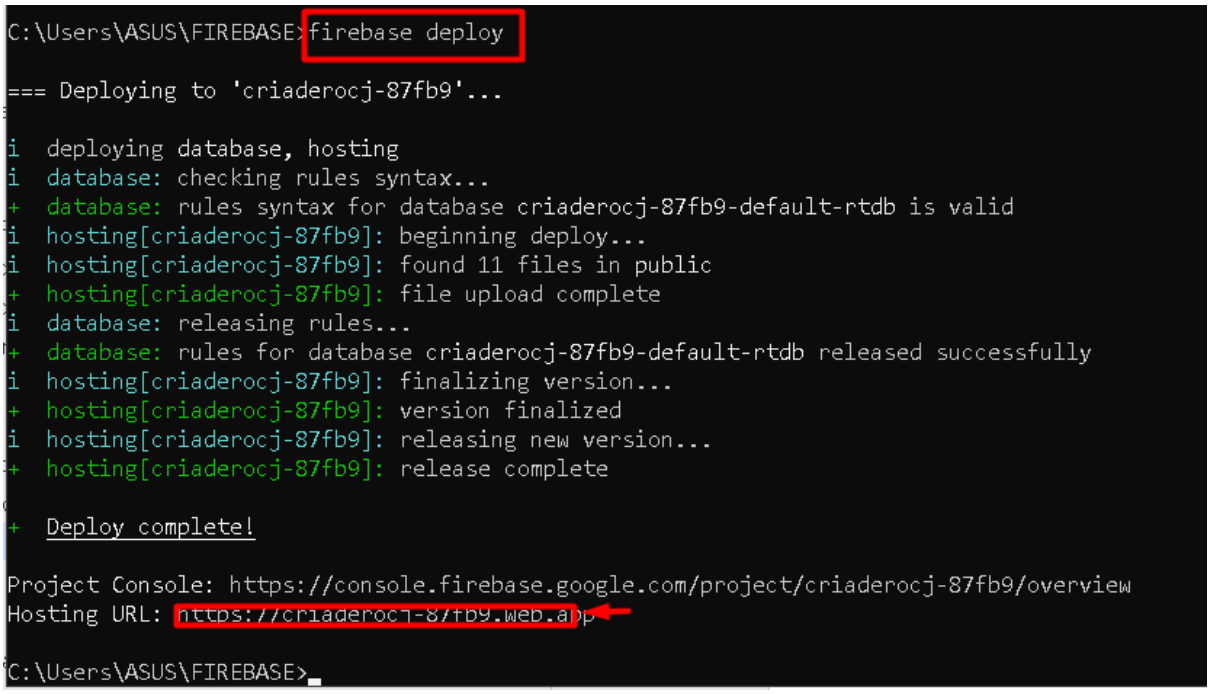


Fuente: Autoría propia

Como paso final en la **Figura *64***, podemos observar la ejecución del comando *firebase deploy*, por medio de esto la aplicación será publicada en firebase hosting, con esto podremos tener el dashboard siempre en línea haciendo uso del URL que nos proporciona luego de la ejecución.

Figura 64

Despliegue del dashboard web mediante firebase hosting



Fuente: Autoría propia

3.7 Implementación del sistema

Siguiendo con la metodología propuesta llegamos a la fase de implementación del sistema, en esta fase se evidenciarán las actividades relacionadas con el diseño y construcción de la jaula, el montaje de los componentes electrónicos, además se establecerá un lugar estratégico para la ubicación de la jaula y como punto final se dará paso a la integración total del sistema, mismo que estará conformada por hardware y software.

3.7.1 Diseño de jaula

En este apartado damos inicio con el diseño de la jaula el cual nos servirá como referencia para su posterior construcción, cabe mencionar que la jaula se dimensionó de acuerdo al número de codornices presentes en la jaula y de acuerdo al lugar de implementación,

Así mismo, el diseño de basó en parámetros y características técnicas tales como la inclinación del piso, área óptima disponible por ave y la disposición interna de la estructura,

conforme a lo establecido en el apartado 2.1.2.7. De igual manera el diseño de la jaula integra componentes como tanque y contenedor destinados a los subsistemas de dispensación de alimento y agua, así como elementos para la recolección de huevos, y la mejora de la higiene de la jaula.

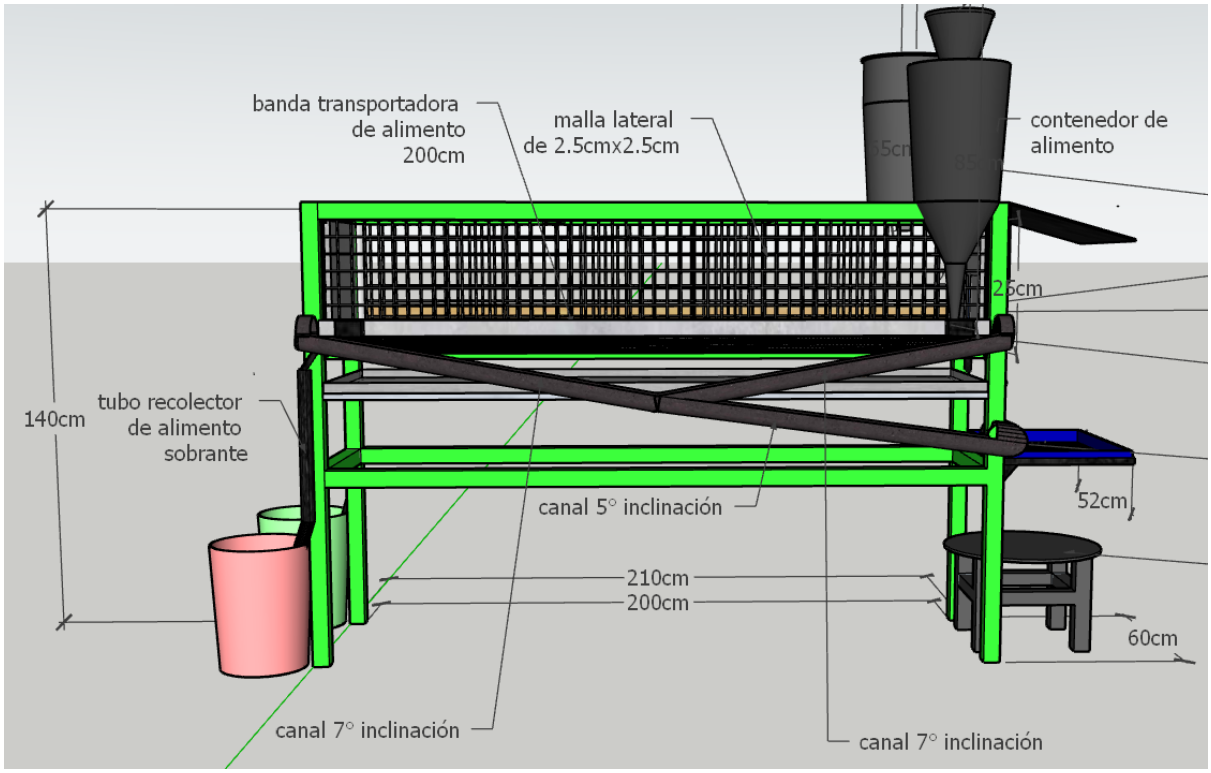
3.7.1.1 Vista frontal

En la **Figura *65***, podemos observar la vista frontal del diseño de la jaula, para lo cual tenemos como puntos importantes la presencia del canal de recolección de huevos mismo que se lo diseñó con una inclinación de 7° y 5° con respecto al eje horizontal, este valor garantiza el desplazamiento continuo del huevo por acción de la gravedad, este ángulo se lo tomo debido a previas pruebas experimentales realizadas, Además tenemos la banda transportadora de alimento la cual fue diseñada con una longitud aproximada de 200 cm, un ancho de 7 cm y una altura lateral de 5 cm, estas dimensiones permiten el transporte adecuado del alimento balanceado y su distribución uniforme a lo largo del comedero, además el sistema incorpora un tubo recolector de alimento sobrante, cuyo objetivo es canalizar el balanceado no consumido hacia un balde de recolección, gracias a esto reducir pérdidas y reutilizar el alimento que se almacena en el balde de recolección, por último tenemos la presencia del diseño de malla lateral el cual se lo coloca en la parte frontal y posterior de la jaula, esta malla tiene dimensiones de 2.5cmx2.5cm, esta malla se ha escogido debido a que permite esta malla actúa como rejilla de acceso hacia el comedero y bebedero permitiendo que la codorniz solo introduzca la cabeza para consumir el alimento y agua, con esto se reducen desperdicios y además de mantener el orden. Si bien en algunos diseños comerciales emplean mallas con aberturas cercanas a 19 mm × 19 mm de acuerdo a (Ferrantinet, 2026), en el presente diseño se optó por una malla lateral de 2,5 cm × 2,5 cm, manteniéndose dentro del rango funcional adecuado para codornices adultas en etapa de puesta, además su elección se fundamenta en base a observación directa de

jaulas utilizadas en comercios locales de la ciudad de Ibarra, los cuales se dedican a la venta de codornices.

Figura 65

Vista frontal del diseño de la jaula



Fuente: Autoría propia

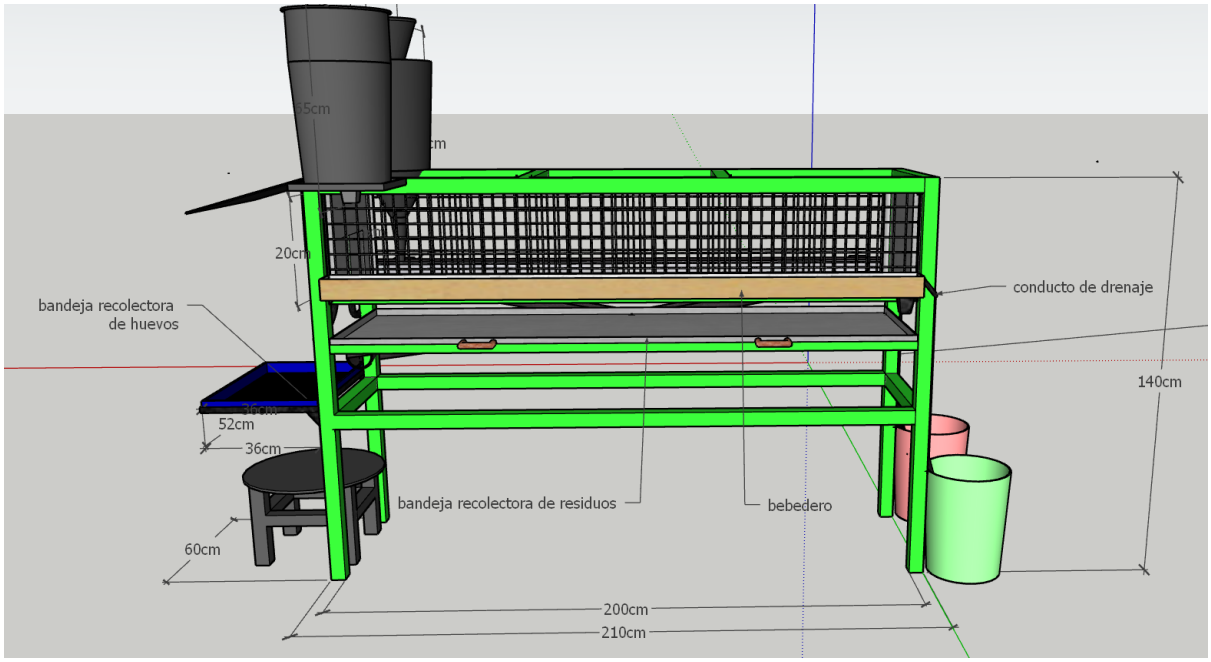
3.7.1.2 Vista posterior

En la **Figura *66***, podemos observar la vista posterior del diseño de la jaula donde se puede apreciar la dimensión total de toda la jaula la cual cuenta con dimensiones de 210cm de largo, 60 cm de ancho y 140 cm de alto, estas dimensiones se establecieron de acuerdo a (Sembralia, 2021) , ya que menciona que se debe proporcionar al menos 145 cm² de espacio en el suelo por ave para obtener una mejor producción. Siguiendo con el diseño tenemos el bebedero el cual cuenta con dimensiones de 200cm de largo, 10 cm de ancho y 5 cm de alto, aunque no existen medidas únicas para bebederos de codorniz existen manuales técnicos los cuales mencionan que durante este período productivo, el sistema debe garantizar un acceso continuo al agua, el

bebedero debe ser de baja altura y facilitar el acceso al agua , además según (Sembralia, 2021), por cada codorniz se debe contar con al menos 0.6 cm de espacio en el bebedero por lo que las dimensiones cumplen con este requerimiento, seguido, podemos apreciar la implementación de un conductor de drenaje el cual será de mucha ayuda para que el agua drenada durante la limpieza del bebedero se almacene en un balde de recolección con el fin de darle un uso extra, además como punto final podemos apreciar la bandeja recolectora de desechos producidos por las codornices, esta bandeja cuenta con dimensiones de 200cm de largo,60cm de ancho y bordes de 2cm, aunque no hay información sobre medidas exactas de bandejas, esta bandeja se dimensionó de esa manera con el fin de cubrir toda el área del suelo con el objetivo de recibir los desechos de toda la jaula sin excepción, además se fundamenta ya que de acuerdo a (Egg Chicken Cage, 2025), menciona que estas bandejas permiten la recolección de los desechos producidos y a su vez mantiene la higiene.

Figura 66

Vista posterior del diseño de la jaula



Fuente: Autoría propia

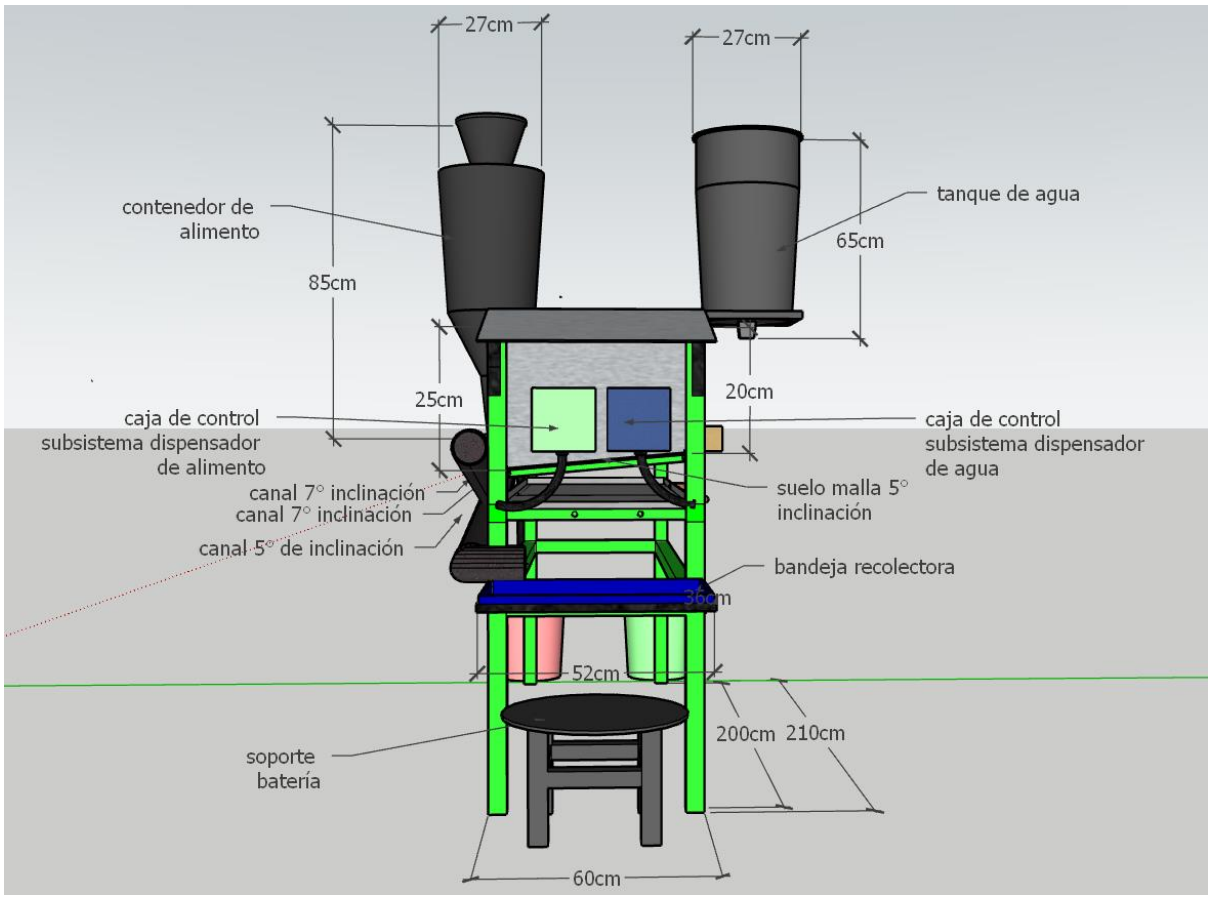
3.7.1.3 Vista lateral

En la **Figura *67***, tenemos la vista lateral del diseño de la jaula, aquí podemos observar el diseño del contendor de alimento el cual cuenta con dimensiones de 85cm de alto, tomando como referencia la boca del dispensador hasta la parte superior de la tapa y un ancho de 27cm, el diseño del contenedor de alimento se caracteriza por su forma conoidal, se optó por esta forma ya que de acuerdo a (Arvind Anticor, 2024), esta forma conoidal permite que el alimento fluya con mayor facilidad y minimiza los bloqueos. Siguiendo con el diseño tenemos el diseño del tanque de agua el cual cuenta con unas dimensiones de 65cm de alto y 27 cm de ancho.

Además, tenemos la presencia de 2 cajas, una de ellas será la encargada del control del subsistema encargada del dispensador de alimento, mientras que la segunda caja será la encargada del dispensador de agua, siguiendo con el diseño tenemos la bandeja recolectora de huevos la cual tiene dimensiones de 52 cm de largo, 36 cm de ancho y 3 cm de alto, finalmente podemos observar un banco el cual nos servirá como soporte para la batería UPS a implementar.

Figura 67

Vista lateral del diseño de la jaula



Fuente: Autoría propia

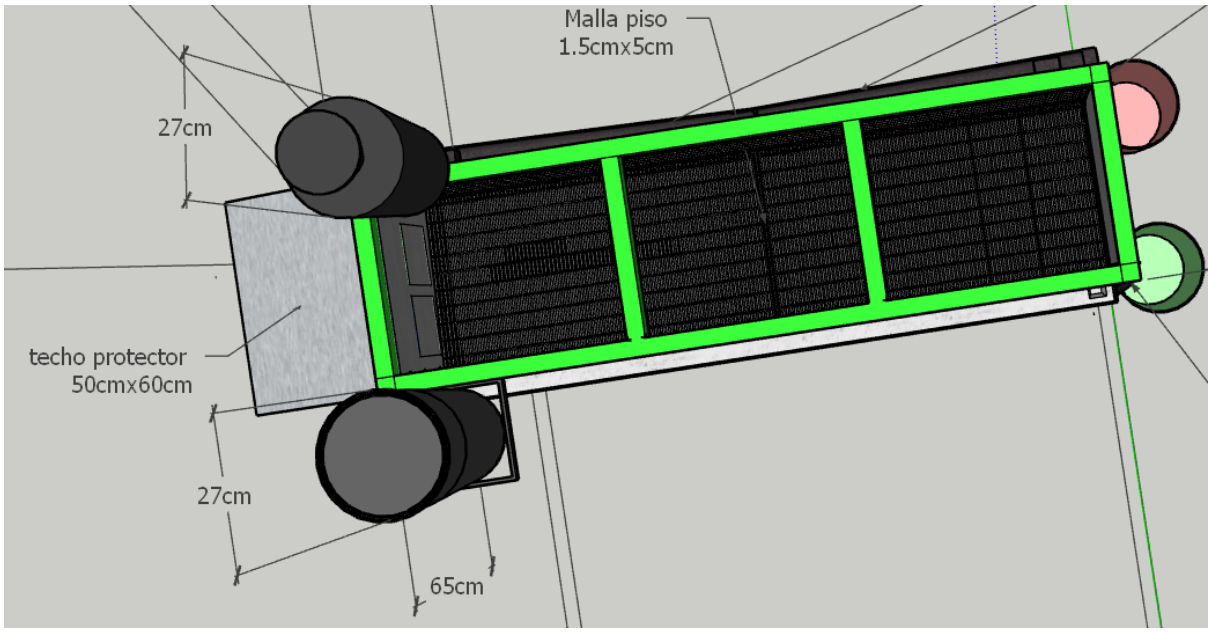
3.7.1.4 Vista superior

En la ***Figura 68***, tenemos la vista superior del diseño de la jaula, en esta vista podemos apreciar el diseño de la malla implementada en el piso de la jaula, para ello se ha hecho uso de una malla tipo ranura de 1,5 cm × 5 cm, esto debido a que el lado más corto sirve como apoyo para las codornices y evita que se lastimen, mientras que la parte larga ayuda para que los desechos de las aves caigan sin quedar estancadas en la malla , a su vez mejora la higiene dentro de la jaula, esta selección se realiza basándose en las especificaciones técnicas de jaulas comerciales para codornices donde según (FDI Poultry Equipment, 2018), donde reporta el uso de mallas de piso de dimensiones 1,27cm x 2,54 cm, con ello se evidencia el criterio de

aperturas alargadas con lado corto para pisos avícolas, como último componente tenemos la implementación de un techo protector el cual cuenta con las dimensiones de 50cm x 60cm, este protector se lo ha diseñado con el fin de proteger la bandeja recolectora de huevos en casos de derrame de agua o polvo.

Figura 68

Vista superior del diseño de la jaula



Fuente: Autoría propia

3.7.2 Construcción de la jaula

A partir del diseño de la jaula definida en la sección 3.7.1, se da paso a la construcción de la jaula de las codornices, para lo cual se consideraron varios aspectos como la correcta elección de materiales adecuados que garanticen resistencia, durabilidad, además la jaula debe cumplir con los parámetros óptimos para el alojamiento y bienestar de acuerdo al número de codornices propuesto.

El proceso de construcción se desarrolló por partes, iniciando con la construcción de la estructura de la jaula y la colocación de las mallas frontal, posterior y de piso, seguido de la construcción del contenedor de alimento, el tanque de agua, la banda

transportadora(comedero), bebedero, bandeja recolectora de desechos, soportes y como punto final, la construcción de la jaula en su totalidad.

3.7.2.1 Estructura de la jaula y colocación de mallas

En este apartado damos continuidad con la construcción de la jaula, iniciando con la construcción de la estructura principal, para ello se hizo uso de tiras de madera de medidas de 5cmx5cm de grosor, de acuerdo al diseño previamente definido se da paso al armado de la estructura haciendo uso de clavos y tornillos, de igual modo para reforzar la rigidez de la estructura se utilizaron pernos en los puntos de unión principales, además como se puede observar en la , se realizó la colocación de la malla tanto en el piso para lo cual se usó la malla con medidas de 1.5cmx5cm adecuada para permitir el paso de desechos y facilitar la limpieza del sistema , en la parte frontal y posterior se colocó la malla con dimensiones de 2.5cmx2.5cm la cual garantiza una adecuada ventilación y facilita el acceso al agua y al alimento a las codornices .

Figura 69

Construcción de la estructura de la jaula y colocación de malla

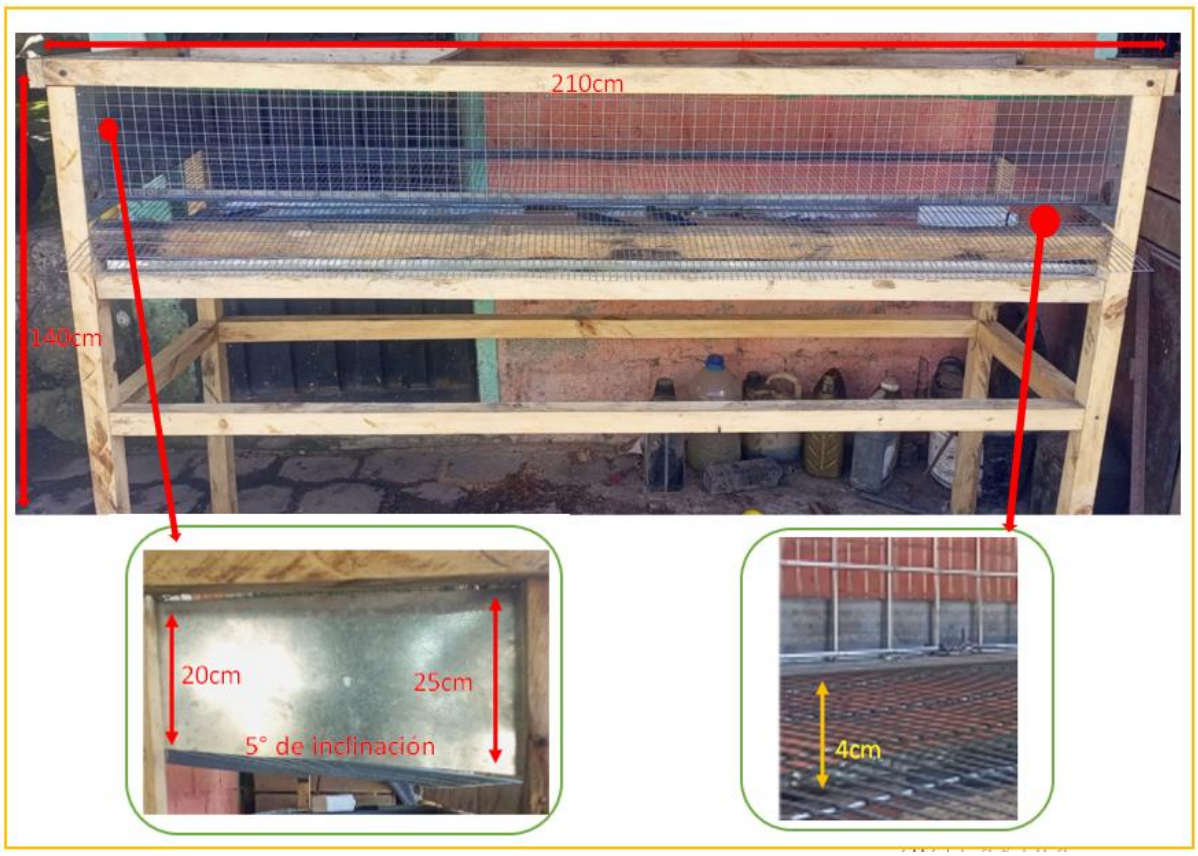
Fuente: Autoría propia

En la **Figura *70***, podemos ver el resultado final de la construcción de la estructura de la jaula y malla, como se puede apreciar tenemos de acuerdo a las dimensiones sugeridas en la parte del diseño el cual contempla las dimensiones para la jaula de 210cm de largo, y 60cm de ancho y 140 cm de alto, de igual modo se puede apreciar que a los lados se integró lámina galvanizada con el fin de proteger a las codornices de corrientes de aire y mejorar la higiene, además servirá como protección para evitar contaminar de plumas o impurezas hacia la parte lateral donde estarán ubicados la bandeja de recolección y demás componentes electrónicos, además de la inclinación de 5° con respecto al eje horizontal con el fin de facilitar el desplazamiento de los huevos hacia la zona de recolección, de igual modo la distancia entre la malla lateral y la malla del piso fue de 4 cm, ya que de acuerdo (Tapia Garófalo , 2010), el huevo de codorniz japonesa presentan un diámetro longitudinal de 3.14 cm y un diámetro

transversal de 2.41 cm, por lo que la distancia asignada es suficiente para garantizar el desplazamiento adecuado de los huevos de codorniz.

Figura 70

Resultado final de la estructura de la jaula y colocación de malla



Fuente: Autoría propia

3.7.2.2 Contenedor de alimento

Siguiendo con la construcción de la jaula tenemos la construcción del contenedor de alimento el cual formará parte de la estructura de la jaula, como se puede observar en la, para su construcción se hace uso de un embudo y un valde, los cuales fueron pegados con pegamento especial para unirlos y se convierta en un solo cuerpo. Además, mencionar que este contenedor facilita la conducción del alimento hacia el mecanismo de dispensación, lo que garantiza una dispensación adecuada hacia el comedero.

Figura 71

Construcción del contenedor de alimento

Fuente: Autoría propia

En la **Figura *72***, podemos observar la construcción final del contenedor de alimento, al cual se le ha añadido un soporte de madera, con el fin de poder implementar ahí el servomotor, el cual será el encargado de dosificar la comida, además como punto final se le añadió un recipiente con tapa en la parte superior del contenedor, el cual servirá mediante su entrada para el abastecimiento de alimento.

Figura 72

Construcción final del contenedor de alimento

Fuente: Autoría propia

3.7.2.3 Tanque de agua

Para la construcción del tanque de agua, se hizo uso de dos baldes reutilizados, siguiendo las dimensiones del diseño, aquí el balde principal será el que se abastezca de agua al sistema, mientras que el segundo balde servirá como recipiente para abastecer de agua al balde principal , además mencionar que para que el agua ingrese al balde principal se lo hace mediante gravedad y por la adición de un tubo como se puede observar en la **Figura *73***, esta implementación facilita el proceso de abastecimiento de agua sin requerir elementos mecánicos adicionales y a su vez promueve el uso eficiente de materiales reutilizados.

Figura 73

Construcción del tanque de agua

Fuente: Autoría propia

En la **Figura *74***, podemos ver la construcción final del tanque de agua, el cual en la parte inferior se le añadió un recipiente extra para colocar la bomba de agua, además mencionar que se le añadió una tapa con el fin de evitar e ingreso de agentes externos al balde principal o en el peor de los casos evitar el taponamiento del tubo que conecta el balde secundario con el balde principal, con ello evitar desbordamientos.

Figura 74

Construcción final del tanque de agua

Fuente: Autoría propia

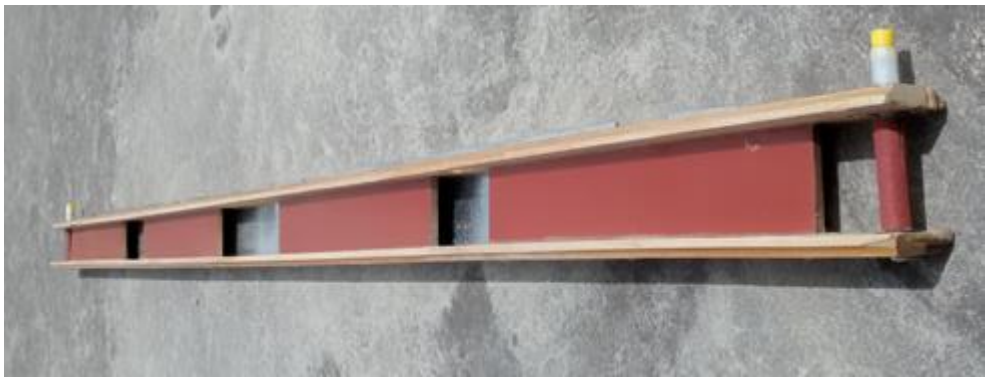
3.7.2.4 Banda transportadora (comedero)

Siguiendo con la construcción e la jaula, damos paso a la construcción de la banda transportadora, esta banda cumplirá una función muy importante dentro del sistema el cual será de proveer de alimento de manera uniforme a lo largo del comedero por lo que basándose en las dimensiones del diseño previamente establecido, se da paso a su construcción, para la construcción de la banda transportadora se usaron materiales reutilizables con el fin de

optimizar recursos y reducir costos sin afectar su funcionamiento, para su soporte lateral se hizo uso de dos tablas las cuales fueron lijadas y trabajadas de manera pareja y en la parte inferior, se añadió lámina galvanizada, además como se puede observar en la , se implementó un sistema de rodamiento para el eje del rodillo , mediante el uso de rulimanes de media pulgada lo cual permitió un giro suave y continuo ya que en el sistema se añadieron un total de 4 rulimanes, un par en cada extremo de la banda transportadora, adicionalmente se añadió material áspero en el rodillo el cual junto con la banda elaborada a partir de cuerina, genera fricción necesaria para un desplazamiento eficiente de la banda transportadora.

Figura 75

Construcción de la banda transportadora(comedero)



Fuente: Autoría propia

En la ***Figura 76***, podemos observar el resultado final de la banda transportadora la cual está lista para su integración en la jaula, además permitirá una adecuada dosificación y contribuirá a mejorar el proceso de alimentación de las codornices.

Figura 76

Construcción final de la banda transportadora(comedero)

Fuente: Autoría propia

3.7.2.5 Bebedero

En este apartado tenemos la construcción del bebedero, para lo cual, mediante el diseño previamente establecido, damos paso a su construcción, como podemos ver en el diseño, las dimensiones son de 200cm de largo, 10 cm de ancho y 5cm de alto, de acuerdo a estas medidas, se ha optado por usar un tubo rectangular de PVC el cual cumple con estas dimensiones, sin embargo el tubo rectangular adquirido es de dos canales para lo cual se lo modifica eliminando la división central como se puede observar en la, esto se lo realiza con el fin de que el bebedero sea solo de un canal, además cabe mencionar que al ser un tubo PVC, es un material adecuado para el contacto con el agua, ya que según (Diers, 2025), menciona que el PVC utilizado es seguro en sistemas de conducción de agua y no es dañino, por lo tanto no presentará daños y no será perjudicial para la salud de las codornices.

Figura 77

Construcción del bebedero

Fuente: Autoría propia

Además, se le añade un recipiente en la parte inferior el cual será usado posteriormente por una de las bombas enfocadas al drenaje del agua, como se puede observar en la ***Figura 78***, además se verificó la correcta nivelación del canal para asegurar que el flujo de agua sea uniforme en toda su extensión.

Figura 78

Construcción final del bebedero

Fuente: Autoría propia

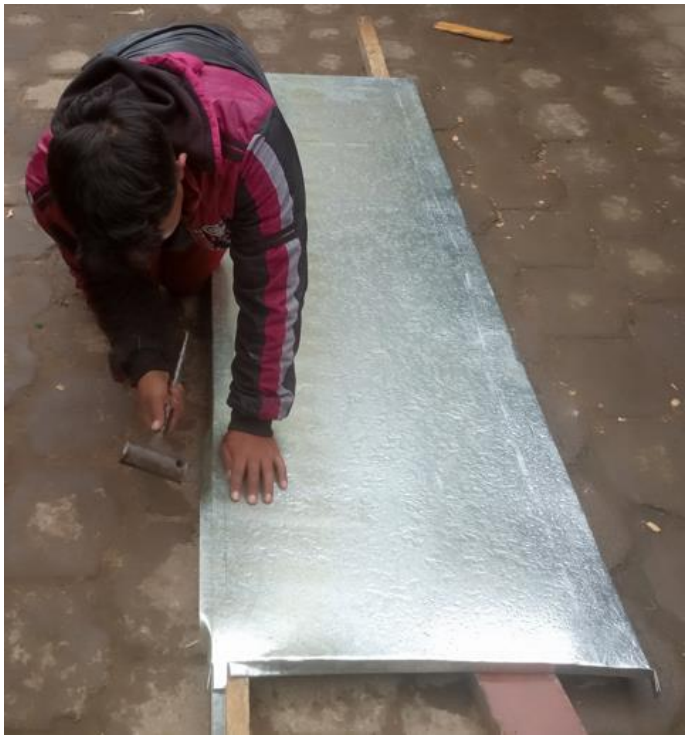
3.7.2.6 Bandeja recolectora de desechos

Siguiendo el diseño de la jaula , damos paso a la construcción de la bandeja recolectora de desechos, como se puede observar en la **Figura *79*** , lo primero que se realiza es cortar la

lámina galvanizada de acuerdo a las dimensiones establecidas, una vez que se corta , se procede a realizar un doblez alrededor de la bandeja con el fin de que la bandeja quede rígida , se hizo uso de esta lámina galvanizada ya que de acuerdo a (Shijiazhuang Kangfa Machinery Equipment Co, 2025), este material es resistente a la corrosión, por lo que es apto para entornos avícolas debido a que en estos entornos existe presencia de humedad y excrementos, además de ser un material que facilita las tareas de higienización.

Figura 79

Construcción de bandeja recolectora de desechos



Fuente: Autoría propia

En la **Figura *80***, podemos observar la construcción final de la bandeja de recolección de desechos, como se puede observar se añadió dos manijas las cuales servirán para una fácil manipulación por parte del coturnicultor a la hora de realizar la limpieza, así mismo esta bandeja se situará debajo del piso de la jaula, para que pueda recolectar los desechos de toda la jaula.

Figura 80

Construcción final de la bandeja de recolección de desechos

Fuente: Autoría propia

3.7.2.7 Canal y bandeja recolectora de huevos

En este apartado damos paso a la construcción del canal y de la bandeja recolectora de huevos, para ello nos basamos en el diseño previamente establecido, como podemos ver en la **Figura *81***, los materiales usados para la construcción de la bandeja recolecto, para ello se ha hecho uso de una bandeja con dimensiones de 52 cm de largo, 36 cm de ancho y 3 cm, además para evitar que el huevo caiga de manera brusca y minimizar el impacto se ha implementado esponja y un material llamado espuma EVA de 5mm, de igual modo siguiendo con la construcción , tenemos la parte del canal de recolección de huevos, aquí lo que se realizó fue adaptar el tuvo PVC de 2 pulgadas con una inclinación de 7° y 5° con respecto al eje horizontal , cabe mencionar que ambas partes con inclinación de 7° conectan en el medio con el otro tubo PVC el cual tendrá una inclinación de 5°, de igual modo para evitar cualquier rotura del huevo y minimizar su impacto se forró el tubo PVC de 2 pulgadas con espuma EVA de 5mm,

Gracias a esto obtenemos una inclinación para un correcto rodamiento por parte de los huevos desde el piso de malla hasta la bandeja recolectora.

Como último punto, el ángulo fue definido en base a pruebas prácticas de rodamiento, buscando un equilibrio entre desplazamiento eficiente y protección del huevo.

Figura 81

Construcción de canal y bandeja recolectora de huevos

Fuente: Autoría propia

Finalmente, en la **Figura *82***, podemos apreciar la construcción del canal y bandeja de recolección de huevos.

Figura 82

Construcción final de canal y bandeja de recolección de huevos

Fuente: Autoría propia

3.7.2.8 Soportes

A continuación tenemos la parte de construcción de los soportes para el contenedor de alimento , tanque de agua, banda transportadora(comedero), bebedero, canal y bandeja de recolección, dichos soportes fueron construidos de acuerdo a la ubicación de cada componente en la jaula además se elaboraron mediante el uso de hierros y ángulos lo que proporciona firmeza a la hora de colocar los elementos construidos, como se puede observar en la **Figura *83***, los soportes fueron fijados soldadura lo que asegura su estabilidad a lo largo del tiempo.

Figura 83

Construcción de soportes

Fuente: Autoría propia

Finalmente, podemos observar en la **Figura *84***, los soportes ya elaborados los cuales serán de mucha utilidad para colocar los elementos en la jaula.

Figura 84

Construcción final de los soportes

Fuente: Autoría propia

3.7.3 Ubicación y estructura final de la jaula

A continuación, en la ***Figura 85***, podemos observar el cuarto asignado por parte del coturnicultor para la ubicación de la jaula, este cuarto cuenta con dimensiones de 500cm de largo, 200 cm de ancho y 200cm de altura, además al momento de entrega se pudo constatar que el cuarto funcionaba como bodega por lo que el techo estaba armado por tablas y no contaba con la suficiente iluminación, además tenía ausencia de tomacorrientes por lo que fue necesario adecuar el cuarto para satisfacer estas necesidades.

Figura 85

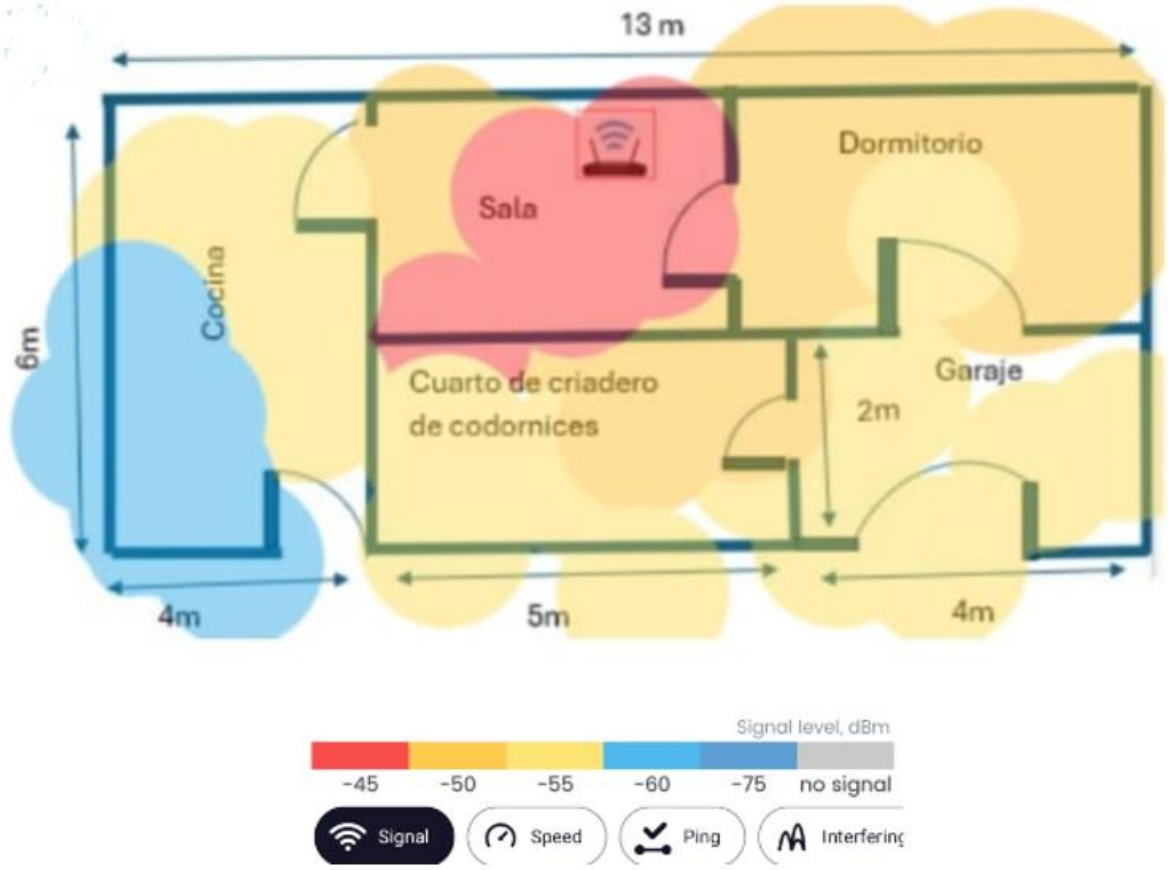
Sitio asignado para la ubicación de la jaula

Fuente: Autoría propia

Otro punto importante a considerar fue la intensidad del nivel de la señal WiFi dentro del lugar donde se ubicará la jaula, para ello se hace un mapeo real tomando como punto de referencia la ubicación actual del router y las medidas exactas del mapa de la casa, este análisis se lo realizó mediante un mapa de calor elaborado gracias a la herramienta WiFi HeadMap como se puede observar en la **Figura *86***, una vez con los resultados obtenidos podemos comprobar que el sitio tiene una señal WiFi entre -50dBm a -55dBm, lo cual representa una señal buena y aceptable, por lo que será satisfactorio para que nuestro sistema funcione correctamente.

Figura 86

Mapa de calor de la intensidad de señal WiFi para el sitio de ubicación de la jaula



Fuente: Autoría propia

Como se puede observar en la **Figura *87***, tenemos la estructura final de la jaula la cual ha sido conformada mediante la integración estratégica de todos los elementos construidos anteriormente, tomando como referencia el diseño previamente definido, además a la jaula se le dio un color verde, este color fue seleccionado debido a que genera un entorno visual más natural y reduce el estrés para las aves de acuerdo a (Derelí Fídan & Kaya, 2023).

Figura 87

Ubicación y estructura final de la jaula

Fuente: Autoría propia

3.7.4 Carcasa y montaje de componentes electrónicos

En esta sección damos paso a la parte de carcasa y montaje de componentes electrónicos los cuales conforman la parte del sistema del criadero de codornices, para ello estos componentes electrónicos se acoplarán y montarán dentro de una carcasa la cual cumplirá la función de proteger contra el polvo y proteger ante alguna manipulación accidental presentada por parte del coturnicultor, otro punto importante a destacar es la consideración de puntos como la facilidad del mantenimiento de los componentes electrónicos así como el orden y buen aspecto visual del criadero. Además, el montaje se lo con el objetivo de proteger y asegurar el buen desempeño de los sensores, actuadores y demás componentes. Para la selección de las

carcasas de toma en cuenta las dimensiones de los componentes seleccionados lo cuales podemos observar en la sección 3.4.1, la cual hace referencia a los componentes físicos del sistema.

3.7.4.1 Caja de control del comedero

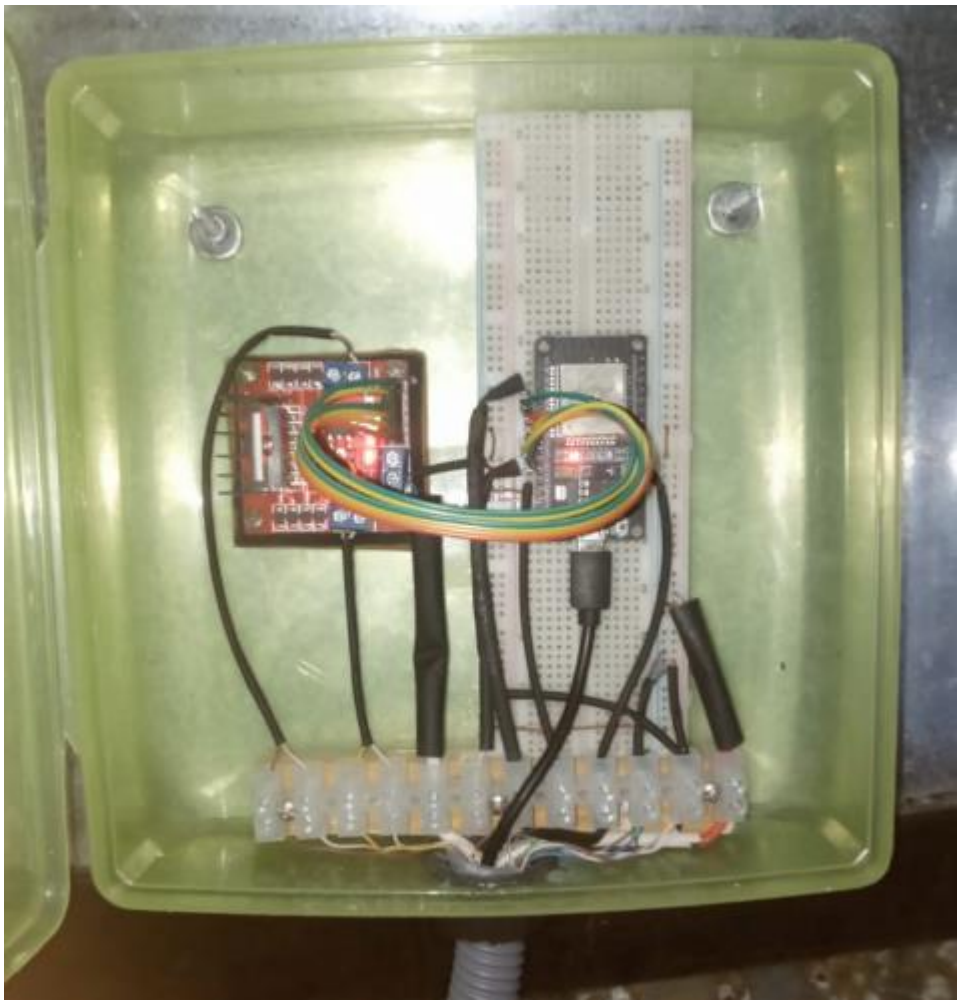
Como se puede observar en la **Figura *88***, para la parte de control de componentes se eligió una carcasa de plástico rígida, la cual fue seleccionada debido a su resistencia y dimensiones adecuadas para alojar todos los elementos de la parte de control del subsistema de alimento de manera ordenada y segura.

Seguido, en el interior de la carcasa se ubicaron componentes como la placa protoboard, sobre la cual se montó el microcontrolador ESP32, a su vez tenemos la distribución con cables tanto para la parte eléctrica como la parte de los sensores, actuadores y demás componentes, además, se añadió el puente H(L298N), con esto, la caja de control permitirá la interacción adecuada para el correcto funcionamiento de este subsistema.

En cuanto a la parte del cableado, fue organizado y asegurado mediante el uso de una bornera plástica ubicada en la parte inferior de la carcasa de plástico, lo que permitió sujetar los cables y permitir una conexión segura ante movimientos bruscos finalmente mencionar que esta distribución y este tipo de montaje se lo realizó de acuerdo al requerimiento de usuario SR16 y SR17 con el fin de garantizar al usuario un fácil mantenimiento o cambio de pieza en caso de presentar algún fallo en el futuro.

Figura 88

Caja de control del subsistema de dispensación de alimento



Fuente: Autoría propia

3.7.4.2 Caja de control de bebedero

A continuación, tenemos la el montaje y carcasa del control del bebedero, como podemos observar en la **Figura *89***, para esta caja se eligió una carcasa de plástico rígida, la cual fue seleccionada debido a su resistencia y dimensiones adecuadas para alojar todos los elementos de la parte de control del subsistema de agua de manera ordenada y segura.

Seguido, en el interior de la carcasa se ubicaron componentes como la placa protoboard, sobre la cual se montó el microcontrolador ESP32, a su vez tenemos la distribución con cables

tanto para la parte eléctrica como la parte de los sensores, actuadores y demás componentes, además, se añadió la parte de los módulos relé los cuales se los sujetó con el uso de tornillos.

En cuanto a la parte del cableado, fue organizado y asegurado mediante el uso de una bornera plástica ubicada en la parte inferior de la carcasa de plástico, lo que permitió sujetar los cables y permitir una conexión segura ante movimientos bruscos finalmente mencionar que esta distribución y este tipo de montaje se lo realizó de acuerdo al requerimiento de usuario SR16 y SR17 con el fin de garantizar al usuario un fácil mantenimiento o cambio de pieza en caso de presentar algún fallo en el futuro.

Figura 89

Caja de control de subsistema de dispensación de agua

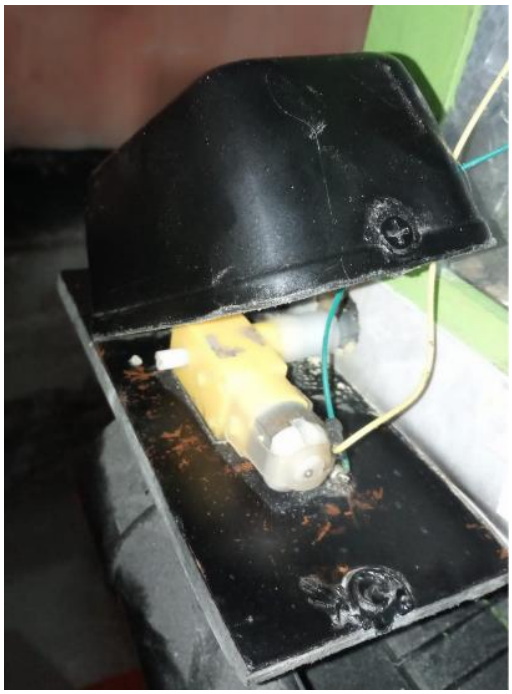
Fuente: Autoría propia

3.7.4.3 Motores

En la ***Figura 90***, podemos evidenciar el montaje del motor DC amarillo el cual se lo colocó en un soporte rígido con el fin de evitar movimientos bruscos al momento de activarse los motores, como podemos observar este componente se encuentra montada dentro de una carcasa plástica , la cual se la adaptó de acuerdo a las necesidades durante el funcionamiento del motor, así mismo la carcasa se la puede abrir de manera fácil en caso de querer realizar algún tipo de mantenimiento o cambio de la pieza afectada, la función principal de esta carcasa es ayudar a proteger este componente en caso de presentar posibles golpes y no afectar el componente electrónico con el fin de evitar posibles fallos a la hora de distribuir la comida. Además, como punto final mencionar que el eje del motor se encuentra acoplado al mecanismo de transmisión de la banda transportadora, lo que permite el accionamiento controlado del comedero conforme a los tiempos y secuencias definidos en la programación del microcontrolador.

Figura 90

Montaje y carcasa de motor DC



Fuente: Autoría propia

3.7.4.4 Servomotor

Siguiendo con el montaje en la ***Figura 91***, tenemos el servomotor el cual es usado como control para la dosificación de alimento hacia la banda transportadora. Este componente se encuentra montado dentro de una carcasa plástica rígida la cual fue seleccionada de acuerdo a que cumple con las dimensiones correctas para alojar el servomotor. De igual modo para asegurarlo correctamente se hizo uso de tuercas y pernos.

Figura 91

Montaje y carcasa de servomotor

Fuente: Autoría propia

3.7.4.5 Sensores ultrasónicos HC-SR04

Como se observa en la ***Figura 92***, se hizo uso de una carcasa dura y rígida de acuerdo a las dimensiones del sensor, después de esto se hizo el montaje con el fin de que el sensor conserve una posición estable, lo cual será optimo para obtener mediciones confiables del nivel de alimento o agua. Mencionar también que esta carcasa permite la libre emisión y recepción de las ondas ultrasónicas sin interferencias.

Figura 92

Montaje y carcasa del sensor ultrasónico HC-SR04

Fuente: Autoría propia

3.7.4.6 Sensor infrarrojo FC-51

Como se observa en la **Figura *93***, el montaje del sensor infrarrojo FC-51 fue dentro de una carcasa rígida y cerrada afines a las dimensiones y características del sensor, con el fin de que el sensor se mantenga fijo durante el funcionamiento del sistema se lo aseguró a la carcasa mediante el uso de tuercas y pernos, lo que permitió su fijación estable a la carcasa.

Figura 93

Montaje y carcasa del sensor infrarrojo FC-51

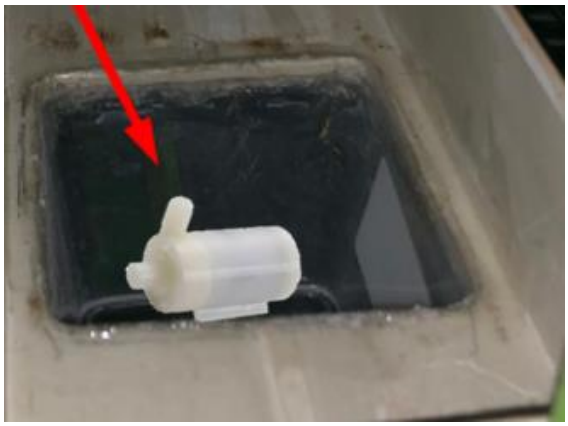
Fuente: Autoría propia

3.7.4.7 Bomba de agua

Como se observa en la **Figura *94***, la mini bomba de agua fue ubicada en el interior del recipiente, permaneciendo completamente sumergida durante su funcionamiento. Para seleccionar la carcasa se hizo uso de un recipiente de acuerdo a las dimensiones de la bomba de agua, el uso de este recipiente fue fundamental para el sistema, ya que ayuda a que el componente quede completamente sumergido en el agua, debido a que, al no tener esta forma de protección el motor puede sobrecalentarse y dañarse.

Figura 94

Montaje y carcasa de mini bomba de agua



Fuente: Autoría propia

3.7.4.8 Buzzer

A continuación, tenemos la ubicación del buzzer, como se puede observar en la **Figura *95***, el buzzer fue colocado en la parte exterior del comedero y tanque de agua, además su carcasa fue diseñada con perforaciones en su parte frontal con el fin de permitir la salida del sonido, así como protección a agentes externos que puedan presentarse.

Figura 95

Montaje y carcasa del buzzer

Fuente: Autoría propia

3.7.5 Integración en el sistema

En esta sección llegamos a la parte de integración del sistema, donde se da paso a la colocación de los diferentes componentes electrónicos en distintos puntos de la estructura de la jaula, para lo cual se lo realiza de manera estratégica de acuerdo a los dos subsistemas existentes , además se toma en cuenta la parte de alimentación eléctrica ya que de aquí se alimentar a todo el sistema, así mismo, a las cajas de control ya que de aquí se distribuirán mediante cable a los distintos componentes electrónicos.

Como último punto mencionar que para la integración se hizo uso de tornillos, tuercas, pegamento, y canaletas los cuales fueron esenciales para mantener en una posición fija y estable de los componentes electrónicos, con esto permitir un buen funcionamiento de todo el sistema propuesto.

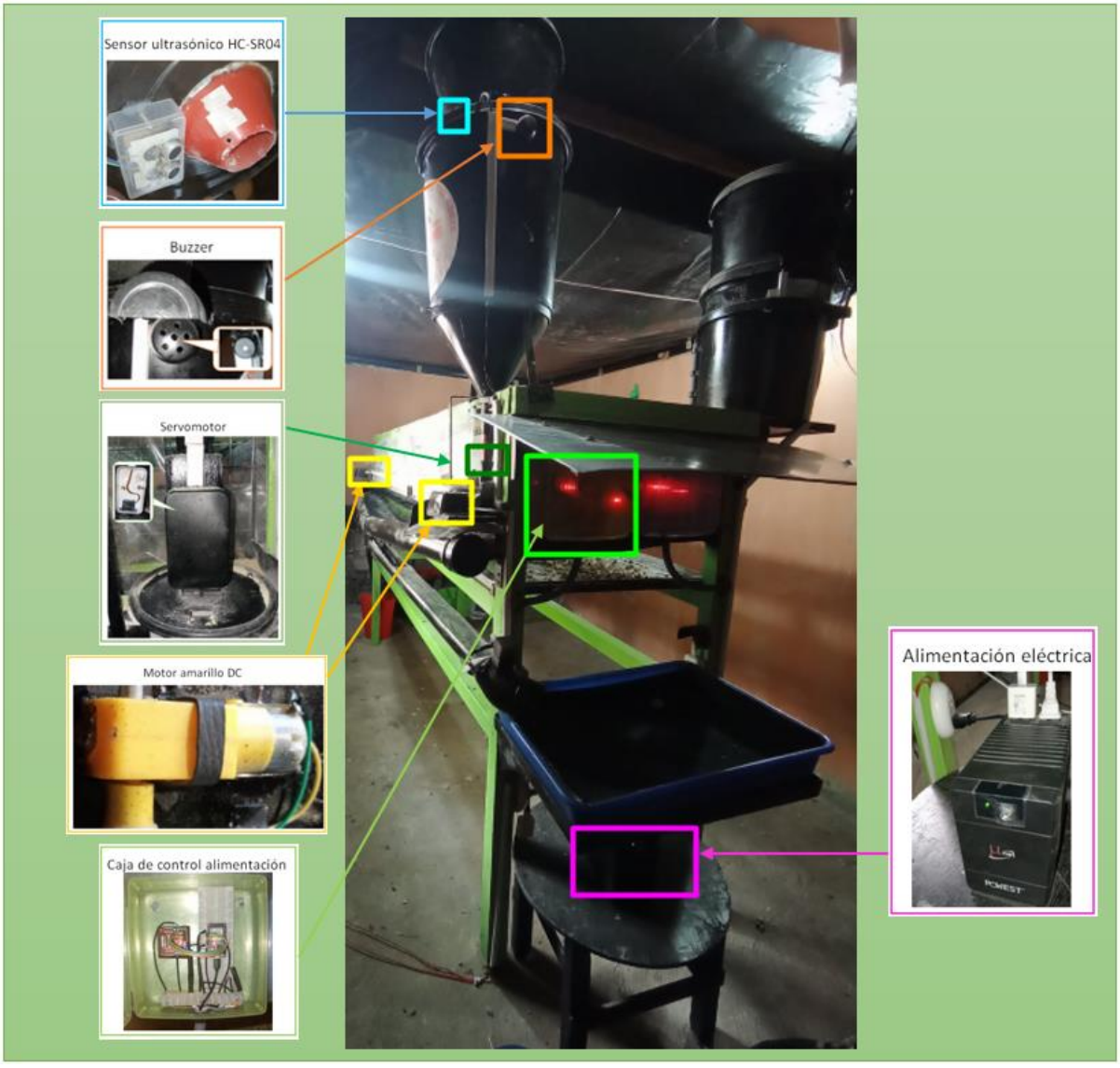
3.7.5.1 Subsistema de dispensación de alimento

A continuación en la **Figura *96***, podemos observar la integración de los elementos que conforman el subsistema de dispensación de alimento, estos elementos se los distribuyó y colocó en lugares estratégicos los cuales permiten el buen funcionamiento del subsistema de dispensación de alimentación, los elementos integrados fueron un sensor ultrasónico HC-SR04 el cual fue ubicado de manera estratégica en el interior del contenedor de alimento , específicamente en la tapa del contenedor, con el fin de que se pueda realizar la medición de la distancia hasta el fondo del contenedor, siguiendo, tenemos el buzzer, el cual se lo colocó en la parte superior del contenedor el cual fue fijado en la parte exterior este componente servirá para alertar de manera sonora al usuario hasta donde deba abastecer el alimento, siguiendo tenemos la parte del servomotor el cual fue ubicado en la boca del embudo, el cual nos permitirá dosificar el alimento de acuerdo a la programación realizada, seguido tenemos la implementación de los motores DC, los cuales cumplen una función importante la cual es mover la banda dispensadora de alimento con el fin de distribuir uniformemente el alimento,

estos motores se los coloca en los extremos de la banda transportadora, además tenemos la caja de control el cual se coloca en un lugar estratégico para evitar su contacto directo, finalmente se ha colocado la parte de alimentación el cual está conformado por el tomacorriente y el UPS el cual de igual modo se lo colocó de forma estratégica el cual se lo colocó ahí con el fin de evitar su contacto directo y por seguridad.

Figura 96

Integración de componentes electrónicos en el subsistema de dispensación de alimento



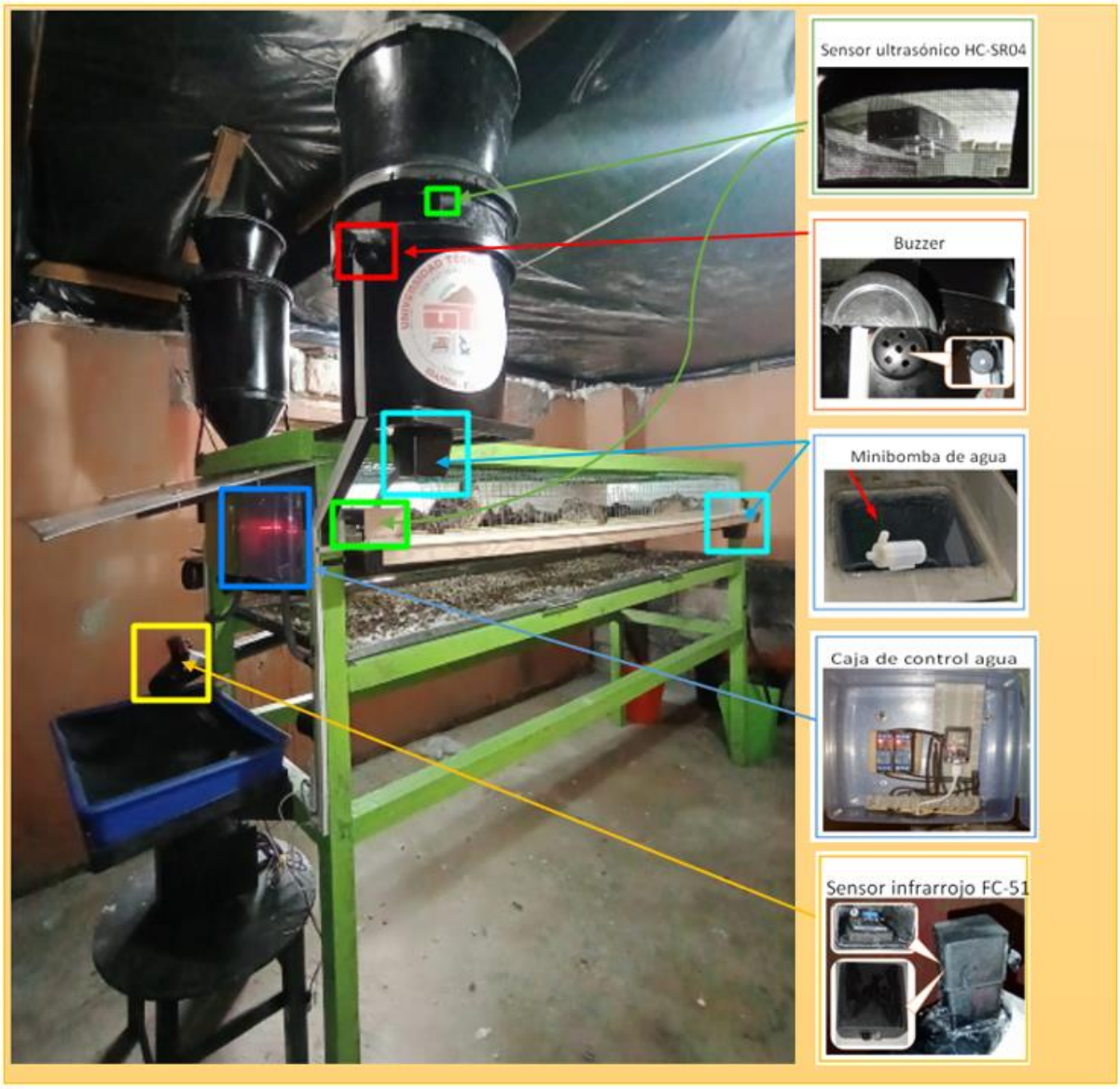
Fuente: Autoría propia

3.7.5.2 Subsistema de dispensación de agua

En la **Figura *97***, podemos observar la integración de los componentes electrónicos correspondientes al subsistema de dispensación de agua, aquí tenemos la integración de los sensores ultrasónicos HC-SR04 donde el primero está ubicado en la tapa del tanque de agua con el fin de medir correctamente la distancia hasta el fondo del tanque, el segundo sensor ultrasónico está ubicado de igual manera en el bebedero debajo del tanque de agua, este sensor se lo coloca de manera estratégica con el fin de medir el agua disponible en el bebedero y según su estado accionar las bombas de agua para abastecer nuevamente de agua limpia al bebedero, igualmente el buzzer se lo coloca en la parte superior exterior del tanque de agua, esto con el fin de alerta de manera sonora al coturnicultor al momento de recargar nuevamente de agua y evitar desbordamientos, seguido tenemos las minibombas de agua de los cuales uno se ponga en la parte inferior del tanque de agua para que abastezca el bebedero, la otra bomba se la coloca en el bebedero con el fin de realizar la limpieza y activar el drenado, además tenemos la parte de la caja de control la cual se la ubico en un lugar estratégico con el fin de evitar el contacto directo con la caja, finalmente tenemos la integración del sensor infrarrojo FC-51 el cual nos servirá para el conteo de huevos, este fue colocado estratégicamente en la salida del canal recolector de huevos el cual finaliza en la bandeja de recolección.

Figura 97

Integración de componentes electrónicos en el subsistema de dispensación de agua



Fuente: Autoría propia

Con la integración de los componentes electrónicos en la estructura física de la jaula, se completa el sistema total, terminando su desarrollo para dar inicio con el capítulo IV el cual corresponde a las pruebas de funcionamiento.

Capítulo IV: Pruebas de Funcionamiento

Con el objetivo de verificar el correcto desempeño del sistema IoT implementado en el criadero de codornices, se da paso a la siguiente fase de la metodología propuesta, este capítulo da inicio con el correcto funcionamiento de los componentes electrónicos lo cual se evidenciará mediante la correcta activación de los motores y servo encargados de la dispensación de alimento y las mini bombas de agua encargados de la dispensación de agua, por otro lado, generación de alertas de acuerdo a las condiciones establecidas, por otro lado, el correcto funcionamiento del dashboard de monitoreo donde muestra los niveles de alimento y agua dentro del contenedor y tanque, también el correcto registro de los huevos detectados, finalizando con el uso de la bandeja de recolección tanto de huevos como de desechos de las codornices así como la evidencia del impacto del sistema en la producción.

4.1 Subsistema de dispensación de alimento

Como se observa en la **Figura *98***, el sistema de dispensación de alimento se encuentra en funcionamiento para lo cual se hace uso del balanceado Exibal de acuerdo a la recomendación establecida en base al análisis obtenido en la sección 3.2.4, el sistema de dispensación permite la caída controlada del alimento hacia el comedero el cual consta de la banda transportadora. Durante esta prueba se verificó que el mecanismo de dispensación libera el alimento de manera uniforme y controlada, con esto se evita acumulaciones de comida y por ende evita desbordamientos de comida, además mencionar que la cantidad dispensada corresponde a 255 gramos, los cuales se sustentan de cuerdo a la sección 2.1.2.2, la cual establece que la cantidad de gramos que se debe suministrar por codorniz es de 25gramos, por lo que con un total de 30 codornices presentes en la jaula del sistema desarrollado, se debe dispensar en el día 750 gramos aproximadamente, por último mencionar que este valor se lo divide en 3 etapas, ya que la dispensación se realiza en un tiempo de ejecución de 30 segundos en los cuales entran en funcionamiento los motores y el servomotor el cual gira 180° abriendo

la compuerta de dispensación, su activación de inicia en un horario establecido lo cual corresponde a las 8:00am, 12:00pm y 16:00pm y deteniéndola una vez cumplido el tiempo definido.

Figura 98

Dispensación de alimento de acuerdo a la hora programada 8:00 am, 12:00 pm, 16:00 pm

Fuente: Autoría propia

4.1.1 Evidencia de dispensación uniforme y continua de alimento

Como se observa en la ***Figura 99***, las codornices acceden directamente al alimento dispensado a lo largo del comedero, evidenciando que la distribución se realiza de manera uniforme y continua, este tipo de dispensación permite que las codornices se alimenten sin presentar aglomeraciones cumpliendo adecuadamente con el objetivo planteado, debido a esta acción podemos comprobar que el subsistema de alimentación funciona correctamente.

Figura 99

Evidencia del acceso uniforme al alimento durante las pruebas de funcionamiento

Fuente: Autoría propia

4.1.2 Prueba de abastecimiento manual del contenedor de alimento

Como se puede observar en la ***Figura 100***, tenemos el proceso de abastecimiento del alimento la cual se realiza de manera manual por parte del coturnicultor, para ello se deposita el alimento por la parte superior del sistema hasta que emita la alerta sonora por un tiempo de 5 segundos, De acuerdo a las pruebas realizadas podemos mencionar que el diseño del

contenedor permite al coturnicultor un abastecimiento de una manera fácil, además almacenarlo de forma adecuada y en perfecto estado para su respectiva dispensación.

De acuerdo a este diseño también podemos confirmar que resulta muy práctico en la operación cotidiana del coturnicultor facilitando la parte de abastecimiento de alimento.

Figura 100

Abastecimiento de comida en el contenedor de alimento

Fuente: Autoría propia

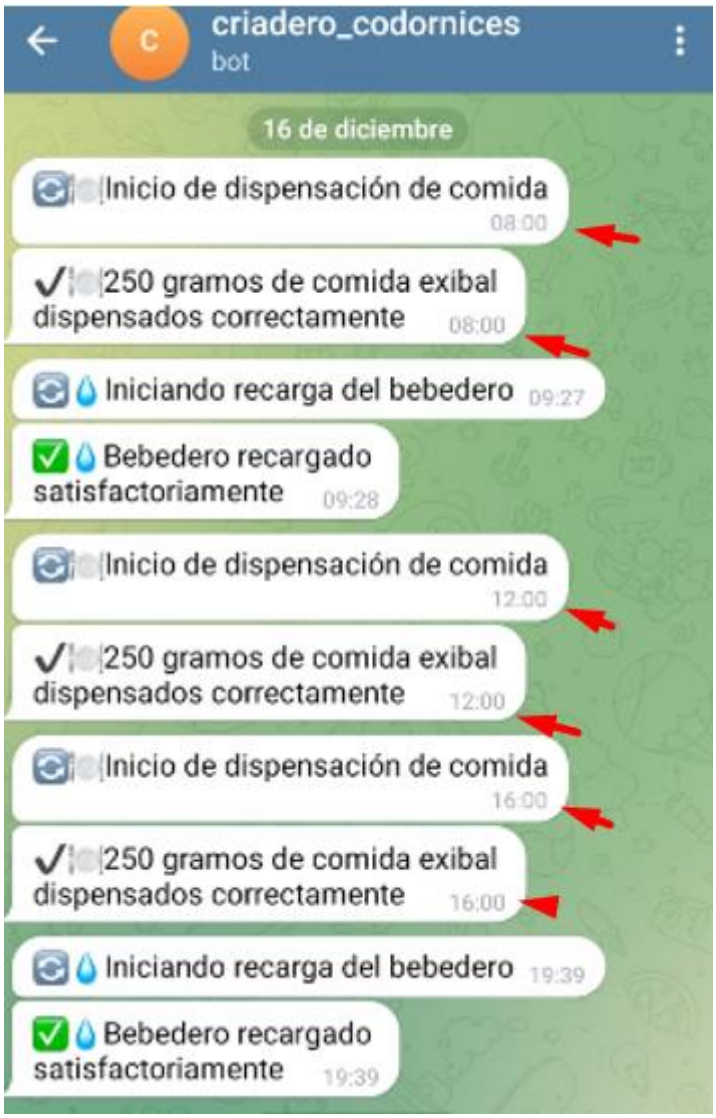
4.1.3 Sistema de notificaciones por Telegram (subsistema de dispensación de alimento)

Una vez que se realiza la dispensación de alimento se envían las notificaciones generadas al inicio y final de la dispensación por telegram al teléfono celular del coturnicultor, lo cual se puede evidenciar en la **Figura *101***.

Este resultado confirma el correcto funcionamiento del subsistema de alimentación y a su vez valida su integración con la estructura de la jaula implementada, evidenciando el funcionamiento confiable del sistema

Figura 101

Evidencia de notificaciones por Telegram sobre dispensación de alimento en la hora programada



Fuente: Autoría propia

El proceso de abastecimiento de alimento es muy importante para garantizar un buen funcionamiento del sistema es por ello que, para garantizar un buen abastecimiento del contenedor, el sistema nos envía una notificación al momento que el alimento se encuentre en estado lleno, crítico y vacío como podemos observar en la ***Figura 102***, con esto el coturnicultor estará al pendiente del estado del contenedor de alimento y poder tomar acciones necesarias.

Figura 102

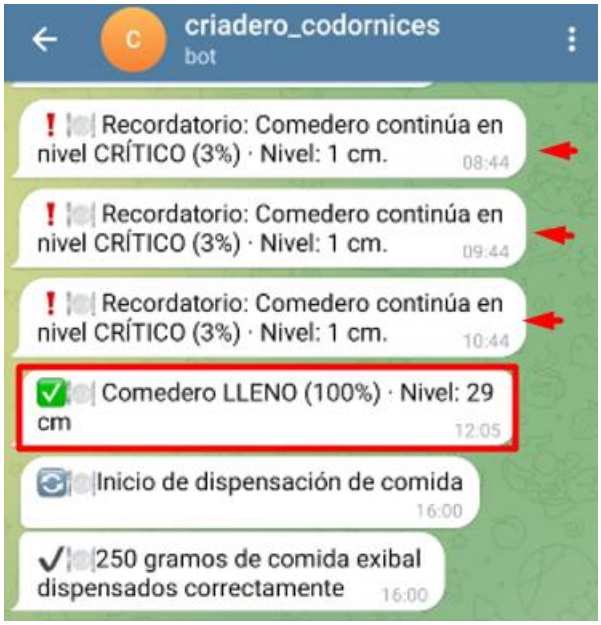
Evidencia de pruebas de notificaciones por Telegram de acuerdo a estados críticos del contenedor de alimento

Fuente: Autoría propia

A continuación también se evidencia el buen funcionamiento de los recordatorios de acuerdo a los niveles crítico y vacío, estas notificaciones se envían en el tiempo de acuerdo a la programación realizada , para lo cual cuando el contenedor presente un estado crítico se envía la notificación a telegram cada hora con el fin de alertar al usuario y tome acciones rápidas y que el contenedor no llegue a un estado de vacío, esto se puede evidenciar en la **Figura *103***, mientras que las notificaciones cuando está el contenedor se encuentra en estado vacío se lo envía cada 3 horas con el fin de no llenar de muchas notificaciones y evitar saturación de alertas como se puede observar en la **Figura *104***, el envío de notificaciones seguirá hasta que el coturnicultor vuelva a abastecer de alimento el sistema.

Figura 103

Evidencia de pruebas de notificaciones por Telegram del recordatorio de mensajes ante estado crítico del contenedor de alimento



Fuente: Autoría propia

Figura 104

Evidencia de pruebas de notificaciones por Telegram del recordatorio de mensajes ante estado vacío del contenedor de alimento

Fuente: Autoría propia

4.1.4 Recolector de alimento sobrante

Como último punto tenemos la parte del funcionamiento del mecanismo para la recolección de alimento sobrante en la banda transportadora, como se puede observar en la **Figura *105***, se puede corroborar que el mecanismo recolecta perfectamente el alimento sobrante, esto se puede evidenciar por el balde rojo el cual contiene restos de alimento. El alimento recolectado puede ser reutilizado con ello se evita desperdicios de comida a comparación del método tradicional.

Figura 105

Evidencia de la recolección de alimento sobrante en la banda transportadora de alimento

Fuente: Autoría propia

4.2 Subsistema de dispensación de agua

A continuación, en la

Figura 106, podemos observar el subsistema de dispensación de agua el cual permite el suministro continuo de agua hacia el canal del bebedero, manteniendo un nivel adecuado para el consumo de las codornices. Durante su funcionamiento se verificó que el agua fluye de manera uniforme a lo largo del canal, facilitando el acceso fácil por parte de las codornices, además no todas las codornices toman agua al mismo tiempo, el bebedero satisface perfectamente con el objetivo de abastecer de agua a todas las codornices alojadas en la jaula.

Además se puede comprobar que el nivel de agua se mantiene dentro de los rangos establecidos con lo cual se evita el desbordamiento y desabastecimiento de agua en el bebedero, de igual modo, podemos mencionar que el diseño del bebedero permite un flujo del agua constante, y evita que el agua quede estancada por varios días, para ello también juega un papel muy importante la programación para la limpieza, donde realiza el proceso de drenar toda el agua almacenada en el bebedero con el fin de expulsar el agua presente en ese momento, con este proceso evitamos la proliferación de microorganismos y mantenemos una higiene adecuada para las codornices.

Figura 106

Dispensación de agua en el bebedero

Fuente: Autoría propia

4.2.1 Prueba de abastecimiento manual del tanque de agua

Como se observa en la **Figura *107***, el proceso de abastecimiento del tanque de agua se realiza de forma manual por parte del coturnicultor, vertiendo el agua en el contenedor por la parte superior del sistema hasta que se active la notificación sonora por 5 segundos, además el diseño del tanque permite una recarga sencilla.

Figura 107

Prueba de abastecimiento manual del tanque de agua

Fuente: Autoría propia

4.2.2 Sistema de notificaciones por Telegram (subsistema de dispensación de agua)

Finalmente, tenemos el envío de notificaciones por Telegram como podemos observar en la **Figura *108***, cuando el nivel del tanque de agua llegue al estado de lleno y crítico, esto con el fin de que el coturnicultor tome acciones frente a los estados notificados, además tenemos la acción de limpieza del bebedero lo cual culmina con la notificación del abastecimiento del bebedero.

Figura 108

Evidencia de pruebas de notificaciones por Telegram sobre el estado de nivel de tanque de agua y limpieza programada

Fuente: Autoría propia

Con esto podemos asegurar un buen funcionamiento del subsistema de dispensación de agua, cumpliendo con lo propuesto.

4.2.3 Recolector de agua drenada durante la limpieza del bebedero

Como último punto tenemos la parte del funcionamiento del mecanismo para la recolección de agua drenada durante los días de limpieza del bebedero, como podemos observar en la **Figura *109***, el agua drenada se almacena en el balde verde con el fin de no desperdiciar este recurso, con ello podemos mencionar que el sistema implementado evita desperdicios de agua ya que el agua puede ser reutilizado ya sea para regar plantas ornamentales, u otro tipo de uso que considere adecuado por parte del coturnicultor.

Figura 109

Evidencia de la recolección de agua drenada durante los días de limpieza del bebedero

Fuente: Autoría propia

4.3 Recolección y conteo automático de huevos

Siguiendo con las pruebas de funcionamiento llegamos al apartado enfocado en la recolección y conteo automático de los huevos, para lo cual se evidenció el correcto rodamiento de los huevos desde el piso de la jaula el cual cuenta con una inclinación de 5° con respecto al eje horizontal hasta llegar al canal recolector, el cual cuenta con una inclinación adecuada de 7° y 5° para que por acción de la gravedad el huevo se desplace hasta la bandeja de recolección.

En la **Figura *110*** podemos observar la acumulación de huevos en la bandeja recolectora, gracias a la inclinación tanto del piso como del canal de recolección, y al adecuado desplazamiento de huevos, lo que prueba que el diseño satisface completamente el objetivo de

recolección de huevos, además como punto final mencionar que los huevos no presentan roturas durante este proceso.

Figura 110

Acumulación de huevos en la bandeja recolectora

Fuente: Autoría propia

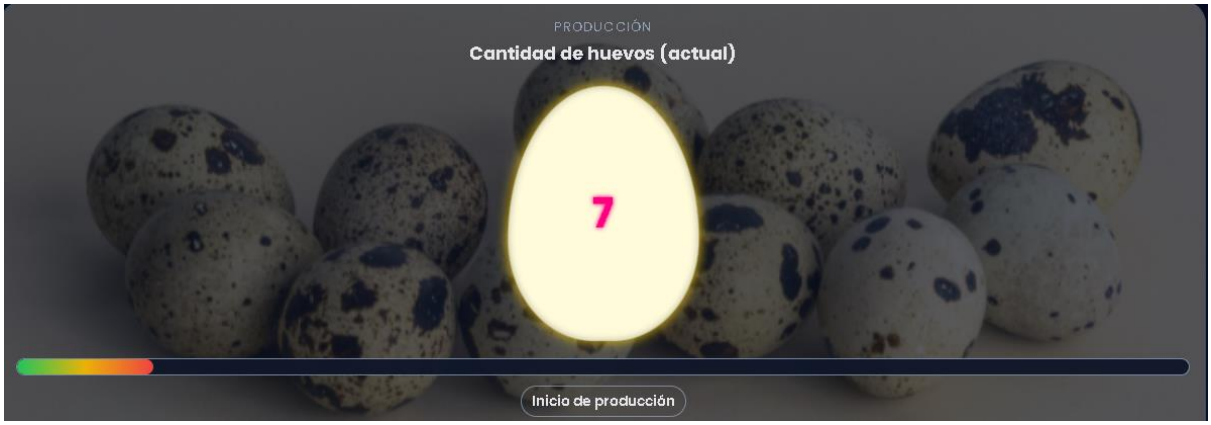
4.3.1 Evidencia del conteo del número de huevos

En este apartado con el objetivo de verificar que el número de huevos recolectados en la bandeja recolectora corresponda al conteo realizado por el sistema, lo hacemos mediante la verificación el dashboard de visualización el cual registra el número de huevos producidos en tiempo real mediante la actualización automática del contador, con lo que podemos confirmar el número de huevos que recorren el canal, con esto podemos hacer una comparación con los huevos presentes en la bandeja , con esta comparación podemos probar que el contador del sistema funciona correctamente, como se observa en la **Figura *111***, tenemos el bloque del dahsboard el cual muestra el número de huevos contados hasta el momento donde tenemos un

valor actual de 7 huevos detectados , mientras que en la **Figura *112*** , tenemos el número de huevos físicos los cuales corresponden a un total de 7 huevos.

Figura 111

Número de huevos detectados



Fuente: Autoría propia

Figura 112

Almacenamiento de huevos en la bandeja recolectora

Fuente: Autoría propia

Una vez que se realizó esta comparación entre el bloque del dashboard enfocado a la parte de visualización del conteo actual de los huevos y el número de huevos presentes en la bandeja recolectora, podemos mencionar que el valor coincide lo que permite validar la

precisión del sistema de conteo implementado, además con esto podemos afirmar que el mecanismo de rodamiento por el canal recolector, junto con el sistema de detección de huevos, funciona de manera adecuada.

4.3.2 Sistema de notificaciones por Telegram (Recolección ciclos de 60 huevos)

Una vez que la cantidad de huevos llegue a 55, se envía una notificación al coturnicultor con el fin de prevenir y alerta sobre que próximamente se completará el ciclo de 60 huevos, a su vez si llega a 60 unidades, el contador se reinicia y envía una notificación por Telegram al coturnicutor alertando para dar paso al proceso de recolección de huevos, como se puede observar en la **Figura *113***, esta notificación únicamente se envía cuando el número de huevos llegue a 60, como evidencia se toma la notificación del día 22 de diciembre del 2025, día en el que se cumplió el valor de 60 huevos.

Figura 113

Evidencia de notificaciones por Telegram (Ciclos de 60 huevos)

Fuente: Autoría propia

4.3.3 Evidencia del proceso de recolección de huevos

Durante las pruebas de funcionamiento, el sistema fue configurado para realizar el conteo automático de huevos hasta alcanzar el valor programado de 60 unidades, luego de llegar a este valor el contador se reinicia. Como se evidencia en la **Figura *114***, una vez completado dicho ciclo, se procede a la recolección manual de los huevos almacenados en la bandeja recolectora y su colocación en recipientes adecuados para su venta.

Figura 114

Recolección de huevos una vez completado el ciclo de 60 huevos

Fuente: Autoría propia

4.4 Bandeja recolectora de desechos

En la **Figura *115***, se evidencia el funcionamiento de la bandeja de recolección de desechos generados por las codornices presentes en la jaula, para ello esta bandeja se la ha ubicado en la parte inferior de la estructura, lo que permite que los desechos caigan directamente por gravedad y sean almacenados en toda la superficie de la bandeja. Para verificar el correcto funcionamiento se pudo evidenciar que la bandeja cumple perfectamente con el objetivo ya que durante el tiempo de pruebas facilitó la recolección de los desechos sin interferir con las áreas de alimentación, agua, bandeja de recolección de huevos y sistema de

alimentación eléctrica. Además, el diseño de la bandeja permitió una extracción sencilla por parte del coturnicultor a la hora de realizar la limpieza.

Figura 115

Prueba funcional de la bandeja recolectora de desechos

Fuente: Autoría propia

Como se observa en la ***Figura 116***, los desechos generados por las codornices cumplen una función extra, ya que sirven como abono orgánico permitiendo al coturnicultor el cual se

dedica la mayor parte del tiempo a la agricultura, aprovechar este recurso para mejorar las condiciones del terreno derivando en un aumento de la productividad de los cultivos agrícolas que posee.

Por lo tanto se puede mencionar que el aprovechamiento de los desechos recolectados representa un beneficio adicional del sistema IoT implementado, ya que aprovecha un subproducto, aportando valor ambiental y productivo al proyecto.

Figura 116

Aplicación de los desechos recolectados como abono orgánico en terreno agrícola

Fuente: Autoría propia

Dashboard

4.5 Visualización de estado de contenedores y conteo de huevos (Dashboard)

Durante las pruebas de funcionamiento, se evaluó la correcta visualización de los datos generados por el sistema en la interfaz del dashboard. Como se puede observar en la **Figura *117***, tenemos el panel principal el cual muestra en tiempo real el nivel de alimento del comedero, el nivel de agua del tanque y la cantidad actual de huevos producidos.

Figura 117

Panel principal del dashboard

Fuente: Autoría propia

Para comprobar su correcto funcionamiento, se tomaron evidencias de acuerdo a las condiciones reales del sistema durante el tiempo de pruebas, como se puede observar en la **Figura *118***, se muestran las variaciones de acuerdo a los niveles (Lleno, Medio, Bajo, Crítico, Vacío) dentro del contendor de alimento y tanque de agua, lo que se refleja de forma inmediata en el dashboard. Asimismo, se comprobó que el conteo de huevos se actualiza correctamente conforme se detecta el paso de cada huevo por el canal recolector.

Figura 118

Evidencias del estado de nivel del contenedor de alimento y tanque de agua

Fuente: Autoría propia

Mediante los resultados obtenidos se puede afirmar que el panel principal del dahsboard cumple adecuadamente su función de monitoreo del contendor de alimento y tanque de agua del sistema implementado, validando el correcto monitoreo en tiempo real.

4.6 Visualización de información de codornices (Dashboard)

En este panel se valida su funcionamiento mediante el correcto despliegue del módulo informativo, cuyo objetivo es proporcionar datos esenciales sobre la codorniz japonesa en cuanto a la producción de huevos, condiciones ambientales, alimentación y agua, como se observa en la **Figura *119***, se comprueba que la navegación hacia este módulo se lo realiza de manera fácil.

Figura 119

Evidencia del módulo informativo sobre codornices japonesas

Fuente: Autoría propia

4.7 Visualización de histórico de producción (Dashboard)

En este apartado tenemos distintos módulos los cuales consisten en la producción diaria de huevos y ciclos de 60 huevos de acuerdo al mes seleccionado, el calendario de producción, registro por mes, y la parte de gestión de datos.

4.7.1 Gráfica de la producción diaria de huevos y ciclos de 60 huevos

Dando inicio con la gráfica de la producción diaria de huevos, tenemos como primer punto la parte de selección del periodo a visualizar, como podemos observar en la **Figura *120***, tenemos la parte para seleccionar el año, y el mes requerido para su visualización, para nuestro caso se hace la selección del mes de diciembre del año 2025, mes en el que se comenzó a realizar las pruebas de funcionamiento.

Figura 120

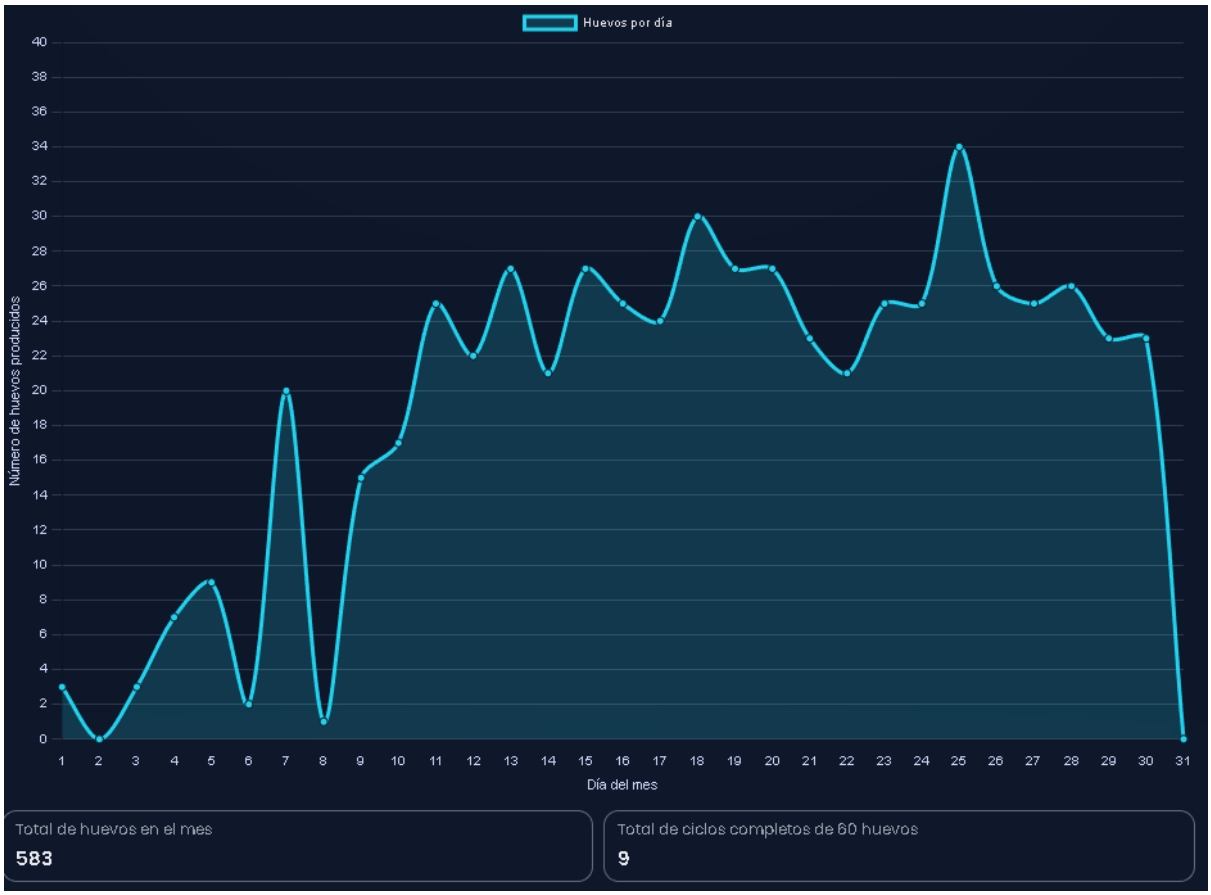
Selector de año y mes para visualizar datos de gráfica de la producción diaria

Fuente: Autoría propia

Una vez que seleccionamos el mes a evaluar nos mostrará la gráfica con los datos obtenidos en el transcurso de los días del mes, la cantidad del número de huevos producidos en el día, con esto podremos identificar variaciones de producción, De acuerdo a esto seleccionamos el mes de diciembre como podemos observar en la **Figura *121***, aquí el día con menor producción fue el 02 de diciembre, esto probablemente al periodo de adaptación de las codornices al nuevo entorno de producción, además se puede evidenciar que el día con mayor producción fue el día 25 de diciembre con un total de 34 huevos, así mismo el día 31 de diciembre no hubo producción, con la gráfica podemos evidenciar el aumento de la producción así como una producción constante, además en la parte inferior de la gráfica tenemos la cantidad de huevos registrados con los que respecta al mes con un total de huevos producidos de 583 huevos y 9 ciclos de 60 huevos, de igual modo con el fin de finalizar las pruebas de funcionamiento nos dirigimos al mes de enero del 2026, como podemos observar en la **Figura *122***, donde tenemos la producción en proceso del mes de enero en el cual los días 9 y 18 de enero presentan una producción menor en comparación del resto de días, además destacar que el día 4 de enero se obtuvo una mayor producción con un total de 34 huevos, en la parte inferior de la gráfica tenemos la cantidad de huevos registrados con los que respecta al mes con un total de huevos producidos de 481 huevos y 6 ciclos de 60 huevos.

Figura 121

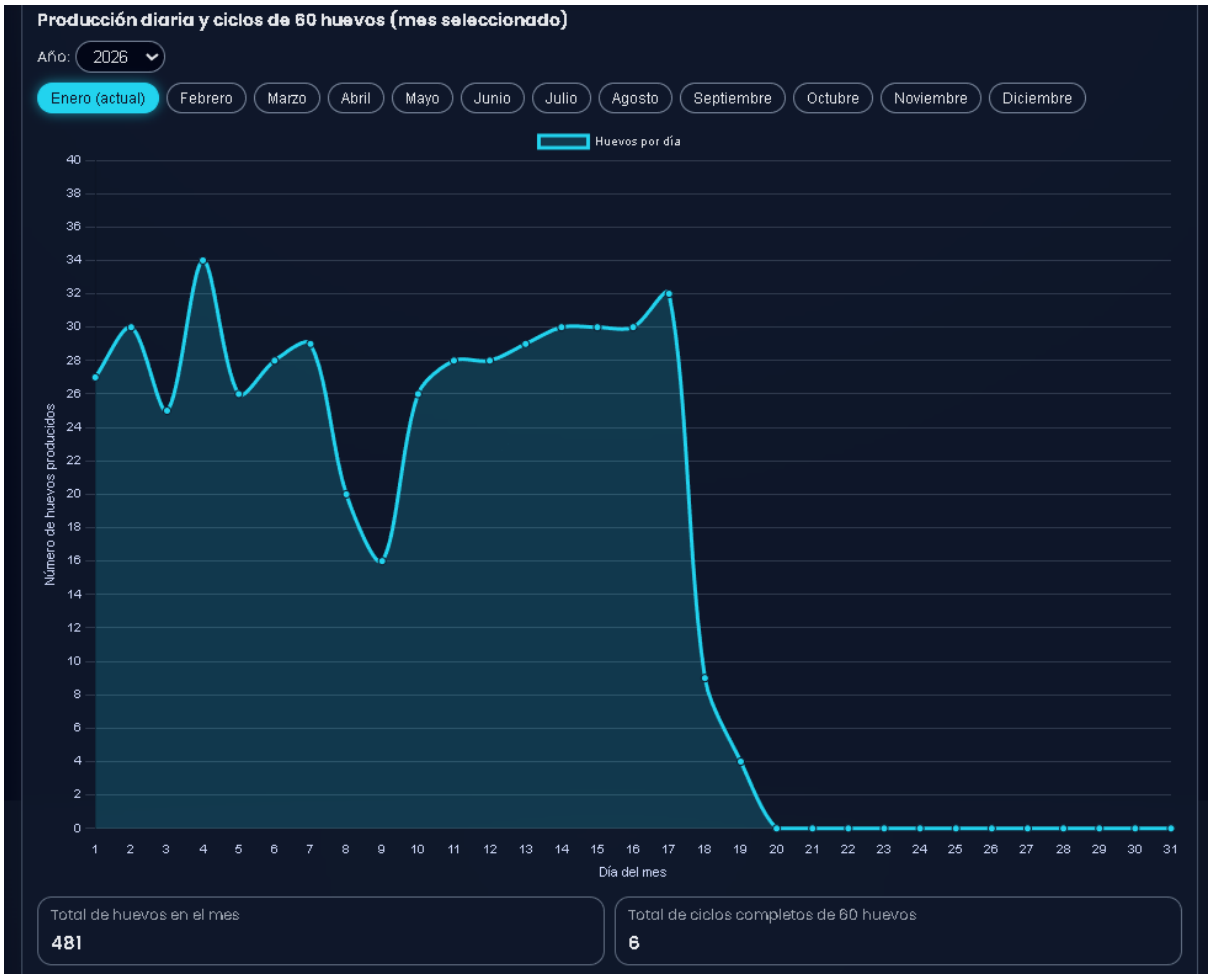
Gráfica de producción diaria de huevos del mes de diciembre de 2025



Fuente: Autoría propia

Figura 122

Gráfica de producción diaria de huevos del mes de enero del 2026



Fuente: Autoría propia

4.7.2 Calendario de producción

Dando continuidad a la visualización de histórico de producción tenemos el módulo de calendario de producción, el cual nos permite visualizar de manera organizada la cantidad de huevos producidos por día, como se puede observar en la **Figura *123*** , este calendario muestra el registro correspondiente de acuerdo al mes seleccionado previamente, en este caso el mes de diciembre del año 2025, el calendario se diferencia al primer gráfico debido a que muestra los niveles de producción de una manera visual, debido a que se codificó por colores como el color rojo representa que no hay producción (0 huevos), el color anaranjado representa una

producción baja equivalente a un rango entre(1-5 huevos), seguido tenemos la producción media la cual se la representa de color azul el cual incluye un rango de (6-15 huevos), finalmente tenemos la parte de producción alta la cual se muestra por medio del color verde mismo que se encuentra en un rango de (16 o más huevos).

Asimismo, se validó que el sistema marque los días en los que se completó el ciclo de producción programado de 60 huevos, facilitando el seguimiento de los eventos relevantes del proceso productivo. Con esta representación el coturnicultor podrá analizar el comportamiento de la producción a lo largo del tiempo y comparar resultados entre diferentes fechas.

Figura 123

Calendario de producción del mes de diciembre del 2025

Fuente: Autoría propia

De igual modo en la **Figura *124***, tenemos el calendario de producción correspondiente al mes de enero de 2026, con el fin de satisfacer el tiempo programado de pruebas de funcionamiento.

Figura 124

Calendario de producción del mes de enero del 2026 (en proceso)

Fuente: Autoría propia

4.7.3 Registro por mes (resumen por semanas)

A continuación en la **Figura *125***, tenemos el registro con lo que respecta a la producción de huevos del mes de diciembre del año 2025, en el cual tenemos el registro por 5 semanas, así como el rango de días, además tenemos el número de huevos producidos por semana como el registro de recolecciones con lo que respecta a ciclos de 60 huevos, por último tenemos las fechas de recolección, de acuerdo a la gráfica de este mes tenemos que la semana con mayor producción es la semana 4 la cual comprende los días 15,16,17,18,19,20 y 21 con un total de 183 huevos, mientras que el total de huevos producidos en el mes es de un total de 583 huevos.

Figura 125

Registro del número de huevos del mes de diciembre del año 2025

Registro por mes (resumen por semanas)

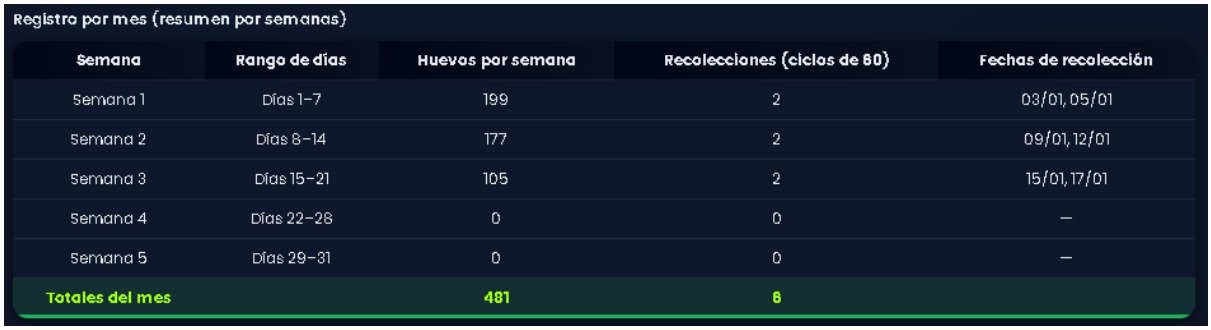
Semana	Rango de días	Huevos por semana	Recolecciones (ciclos de 80)	Fechas de recolección
Semana 1	Días 1–7	44	0	–
Semana 2	Días 8–14	128	2	09/12, 12/12
Semana 3	Días 15–21	183	3	15/12, 17/12, 19/12
Semana 4	Días 22–28	182	3	22/12, 24/12, 26/12
Semana 5	Días 29–31	46	1	29/12
Totales del mes		**583**	**9**	

Fuente: Autoría propia

Siguiendo con el registro en la **Figura *126***, tenemos el registro del mes de enero del 2026, mismo que sigue en proceso de registro, como podemos observar la primera semana fue muy productiva ya que se contabilizaron un total de 199 huevos, mientras que la segunda semana también productiva registro una cantidad de 177 huevos, además se registraron 2 ciclos de 60 huevos con lo que respecta a cada semana.

Figura 126

Registro del número de huevos del mes de enero del año 2026(en proceso)



Registro por mes (resumen por semanas)

Semana	Rango de días	Huevos por semana	Recolecciones (ciclos de 60)	Fechas de recolección
Semana 1	Días 1–7	199	2	03/01, 05/01
Semana 2	Días 8–14	177	2	09/01, 12/01
Semana 3	Días 15–21	105	2	15/01, 17/01
Semana 4	Días 22–28	0	0	—
Semana 5	Días 29–31	0	0	—
Totales del mes		481	6	

Fuente: Autoría propia

4.7.4 Gestión de datos (borrado manual)

Siguiendo con la parte de visualización de histórico de producción tenemos la parte de gestión de datos (borrado manual), este módulo permite al coturnicultor eliminar datos en caso de presentar algún tipo de inconsistencia como lecturas erróneas por daños en los sensores, incorrecta manipulación en el sistema, o el simple hecho de eliminar datos antiguos que ya no tengan mucha relevancia para el análisis de la producción, una vez mencionado esto se realizará la prueba con datos del calendario de producción como se puede observar en la , para esta prueba y ver si el borrado de datos funciona perfectamente se toma el dato correspondiente al día 14 de enero del 2026, en el cual se genera una inconsistencia debido a que se sumó un total de 60 huevos, además de registrar el ciclo de 60 huevos.

Figura 127

Evidencia de funcionamiento del borrado manual de gestión de datos

Fuente: Autoría propia

Siguiendo con la parte de gestión tenemos 3 opciones de eliminado, la primera opción corresponde al día elegido, sin afectar al resto de días, la segunda opción sirve para eliminar los datos del mes elegido, por último una tercera opción la cual sirve para eliminar los registros del año elegido, como podemos observar en la **Figura *128***, elegimos la primera opción la cual nos permitirá eliminar las inconsistencias del día 14 de enero del 2026, para realizar esta acción el sistema dispone de controles de selección de día, mes y año, mediante los cuales el usuario puede definir el período de datos que desea eliminar ya que se ha implementado un mecanismo de desplazamiento (scroll), facilitando la selección del valor requerido, una vez que se selecciona el valor , damos paso a seleccionar la opción de borrar producción diaria ,seguido nos muestra una ventana flotante como método de seguridad el cual nos sirve para confirmar la eliminación del dato de acuerdo al día seleccionado.

Figura 128

Borrado de la producción de huevos de acuerdo al día seleccionado

Fuente: Autoría propia

En la **Figura *129***, podemos ver que una vez que se confirme la eliminación de la producción diaria del día seleccionado, automáticamente se borrará del calendario de producción.

Figura 129

Evidencia del proceso de eliminación de producción diaria de huevos inconsistente

Fuente: Autoría propia

Del mismo modo en la ***Figura 130***, podemos observar la eliminación de la producción del ciclo de 60 huevos registrado en el día seleccionado, así mismo, al presionar en el botón de

borrar ciclos de 60 huevos, nos aparece una ventana flotante para confirmar la eliminación de la producción.

Figura 130

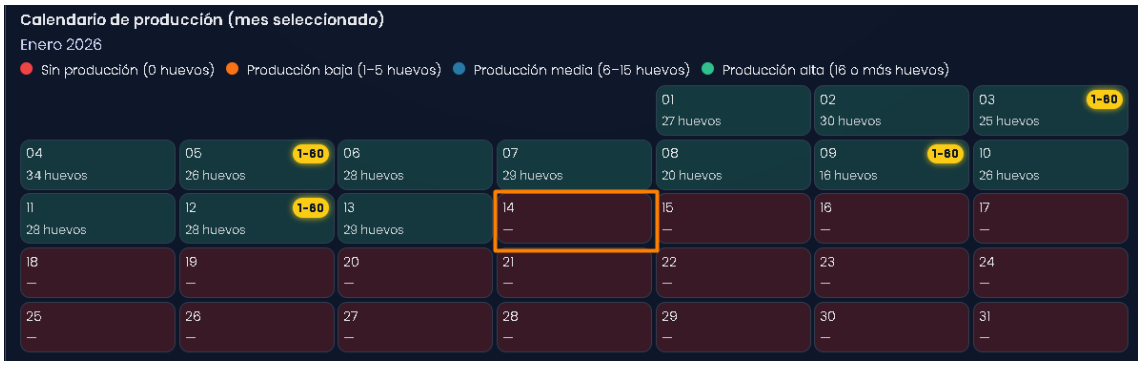
Borrado de la producción de ciclos de 60 huevos de acuerdo al día seleccionado

Fuente: Autoría propia

Una vez que se confirma el borrado de la producción diaria correspondiente al ciclo de 60 huevos, podemos evidenciar en la ***Figura 131***, que el dato ha sido perfectamente eliminado

Figura 131

Evidencia del proceso de eliminación de producción diaria de ciclos de 60 huevos inconsistente



Fuente: Autoría propia

4.8 Comparación de resultados antes y después de la implementación del sistema

En este apartado tenemos la comparación de resultados la cual se realizó con el objetivo de evaluar el impacto del sistema IoT en la producción de huevos del criadero de codornices, para ello, se analizaron los registros obtenidos mediante el sistema implementado y se compararon con los datos obtenidos previamente bajo la experimentación realizada sobre método tradicional el cual se encuentra presente en la sección 3.2.3.

Como primer punto evaluamos la producción generada por el método tradicional para lo cual se estableció una fase de pruebas de 6 semanas iniciando el día 17 de junio del 2025 hasta el 23 de junio del 2025, en este periodo se consideró las 2 primeras semanas como un tiempo de adaptación a su nuevo entorno, de acuerdo a la experimentación realizada, se obtuvo un total de 570 huevos durante el tiempo de experimentación, además se debe mencionar que el tipo de alimento suministrado fue de tres tipos diferentes, pero el que demostró una mayor producción fue el balanceado Exibal el cual se recomendó al coturnicultor y se lo aplicó posteriormente al sistema de control de alimento IoT.

Figura 132

Producción generada mediante el método tradicional

Tipo de Alimento	N° Semanas/ Huevos						Total de huevos /tipo de alimento
	S1	S2	S3	S4	S5	S6	
B	20	26	32	40	44	44	206
E	23	22	32	44	46	45	212
MB	20	22	24	28	29	29	152
Total	63	70	88	112	119	118	570

Mayor producción

Fuente: Autoría propia

Como segundo punto se evalúa la producción obtenida una vez que se ha implementado el sistema, para lo cual se inicia con la fase de pruebas a partir del 1 de diciembre del 2025 y culmina el 11 de enero del 2026, como podemos observar el total de días comprende a 42 días en el cual las dos primeras semanas son destinadas a la adaptación de las codornices a su nuevo entorno, como podemos observar en la **Figura *133***, tenemos el registro de un total de 583 huevos los cuales comprenden al mes de diciembre cuyo mes cuenta con 31 días, siguiendo con el mes de enero tenemos un total de 199 huevos la primera semana(1- 7 días), mientras que los 3 días faltantes los cuales comprenden los días 09,10 y 11 de enero con el fin de completar el periodo de pruebas se obtuvo un total de 70 huevos, con esto podemos mencionar que el total de huevos producidos por un lapso de pruebas de 6 semanas(42 días) es de 852 huevos.

Figura 133

Producción total del mes de diciembre del 2025 y enero del 2026

Registro de producción /diciembre del 2025

Registro por mes (resumen por semanas)

Semana	Rango de días	Huevos por semana	Recolecciones (ciclos de 60)	Fechas de recolección
Semana 1	Días 1–7	44	0	—
Semana 2	Días 8–14	128	2	09/12, 12/12
Semana 3	Días 15–21	183	3	15/12, 17/12, 19/12
Semana 4	Días 22–28	182	3	22/12, 24/12, 26/12
Semana 5	Días 29–31	46	1	29/12
Totales del mes		583	9	

Registro de producción /enero del 2026

Registro por mes (resumen por semanas)

Semana	Rango de días	Huevos por semana	Recolecciones (ciclos de 60)	Fechas de recolección
Semana 1	Días 1–7	199	2	03/01, 05/01
Semana 2	Días 8–14	177	2	09/01, 12/01
Semana 3	Días 15–21	123	2	15/01, 17/01
Semana 4	Días 22–28	0	0	—
Semana 5	Días 29–31	0	0	—
Totales del mes		499	6	

Fuente: Autoría propia

Una vez que tenemos el análisis, podemos mencionar que, en la forma tradicional se registró un total de 570 huevos, mientras que con la implementación del sistema IoT de control de alimento se registró un total de 852, esto con lo que respecta al tiempo de pruebas realizado el cual consistió en un total de 6 semanas(42 días), con esto podemos mencionar y comprobar que, con la implementación del sistema la mejora en la producción es mayor a comparación del método tradicional, Además, mencionar que el sistema IoT implementado garantizó la dispensación de porciones controladas y en horarios programados, esto aseguró que las codornices reciban una alimentación controlada, además de mantener un suministro constante de agua, por otro lado con el sistema implementado se puede afirmar que se redujo la intervención manual derivando en un ahorro de tiempo, así como el ahorro de alimento debido

a su correcto suministro, además de un ahorro económico, derivando en un aumento progresivo en la producción de huevos.

En conclusión, con los resultados obtenidos podemos evidenciar que la implementación del sistema IoT para el control de alimento mejora significativamente el control del proceso productivo, ya que incrementan la cantidad de huevos, además de la calidad ya que son muchos más grandes y resistentes, por otro lado tiene la ventaja de monitoreo y control del alimento a comparación del método tradicional, confirmando que el sistema propuesto no solo optimiza la gestión dentro del criadero, sino que también contribuye de manera directa al aumento de la productividad, con esto se cumple con el objetivo general y alcance propuesto.

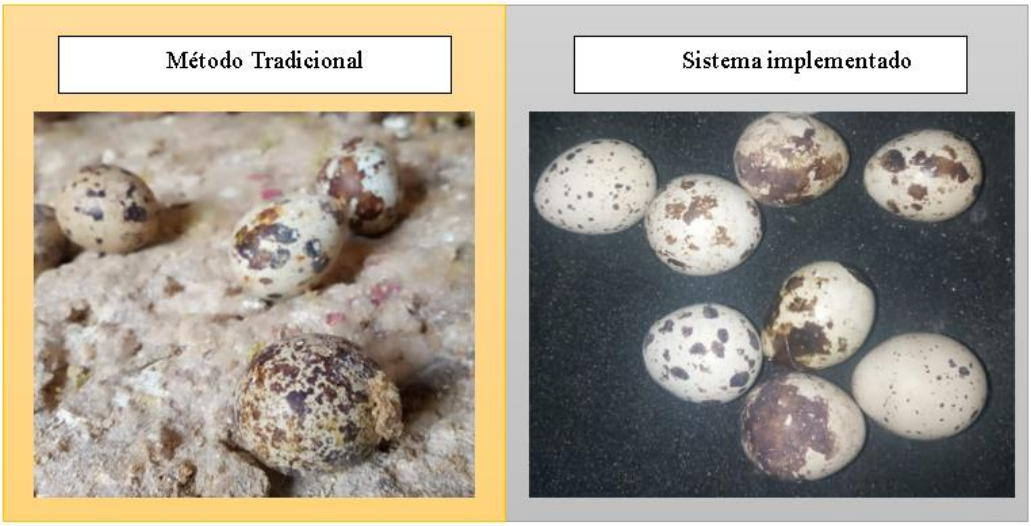
4.8.1 Evidencia del impacto del sistema en la producción

En la

Figura 134, tenemos una comparación entre los huevos producidos por la manera tradicional y los huevos producidos una vez que se ha implementado el sistema de control de alimentación IoT, como se puede observar los huevos producidos por el sistema son mucho más grandes y están completamente limpios a comparación de los huevos producidos por el método tradicional los cuales se puede evidenciar que son más pequeños y presentan suciedad.

Figura 134

Comparación de huevos producidos mediante el método tradicional y mediante la implementación del sistema



Fuente: Autoría propia

Además por palabras del señor Luis Enrique Pupiales Brusil, menciona que gracias a la implementación del sistema puede optimizar su tiempo de mejor manera ya que es el sistema el que se encarga de la dispensacion adecuada de alimento y agua, lo cual permite que las codornices tengan acceso a estos recursos de manera constante , por lo que puede enfocar ese tiempo a otras actividades sin la preocupación de hacerlo de manera manual como lo hacia anteriormente.

Asimismo, menciona que tuvo un impacto econónimo positivo ya que gracias al sistema obtuvo huevos de mayor tamaño, mejor calidad y mucho más hiegiénicos, lo que le facilita a la hora de colocar en los recipientes para sacarlos a la venta como se puede observar en **Figura *135***, otro punto que menciona es que presenció la disminución de huevos rotos lo cuales eran muy comunes antes de implementar el sistema.

Por último, destaca que los desechos generados por las codorncies también le provocó un impacto favorable, debido a que estos desechos los usa como abono para sus cultivos, finalmente mencionó que otro impacto positivo fue la presencia de los baldes de recolección de residuos ya que manifiesta que el balanceado recolectado es colocado nuevamente en el contendor de alimento evitando el desperdicio de alimento, además el desecho del drenaje lo

usa para regar sus plantas ornamentales, mencionando que el sistema cumple con sus espectativas.

Figura 135

Evidencia del impacto del sistema en la producción

Fuente: Autoría propia

4.9 Análisis costo/beneficio

El análisis costo/beneficio permite evaluar la viabilidad económica del sistema propuesto, considerando la relación entre los recursos invertidos y los beneficios obtenidos a partir de su implementación en el criadero de codornices. Para este estudio se tomaron en cuenta los costos asociados a la adquisición de componentes electrónicos, materiales de construcción

4.9.1 Costos de Software

En la **Tabla *34*** se evidenciarán los elementos implementados en el software para el correcto funcionamiento del sistema y su tipo de licencia.

Tabla 34

Costos por software

Software	Tipo de licencia	Costo $
Arduino IDE	Libre/OpenSource	$0.00
Firebase	Gratuito	$0.00
Telegram bot API	Gratuito	$0.00
Visual Studio Code	Gratuito	$0.00
Total		$0.00

Fuente: Autoría propia

4.9.2 Costos de Hardware

En la **Tabla 35** se evidencian los materiales y componentes utilizados en el diseño e implementación de la jaula de codornices especificando los dispositivos, cantidad y costo.

Tabla 35

Costos por hardware

Jaula e insumos			
Materiales / Componentes	Cantidad	Precio Unitario	Costo$
Tiras de madera 5x5cm	10	$ 1.00	$ 10.00
Malla Soldada Galvanizada 76cm	2 m	$4.70/m	$9.40
Malla soldada galvanizada 1x1x1m	2.5 m	$2.87/m	$7.17
Malla Maviju fibra vidrio	1m	$1.29/m	$1.29

Malla plástica 3/4”x3/4”	2.5 m	$2.70/m	$6.74
Angulo metálico negro 20mmx2mm (3/4”x2mm)	2	$4.70	$9.39
Lámina galvanizada	1	$25.00	$25.00
Baldes	5	$3.00	$15.00
Embudo	1	$2.00	$2.00
Bandeja plástica	1	$2.50	$2.50
Recipientes plásticos	10	$0.50	$5.00
Manguera corrugada ½	2 m	$1.00/m	$2.00
Tubo rectangular PVC	1	$8.95	$8.95
Tubo PVC 75 mm	1	$3.13	$3.13
Canaletas plásticas 2mx10mmx22mm	3	$1.25	$3.75
Borneras plásticas	2	$2.00	$4.00
Subsistema de dispensación de alimento			
Componentes electrónicos			
Sensor ultrasónico (HC-SR04)	1	$2.00	$2.00
Microcontrolador (ESP 32)	1	$15.00	$15.00
Protoboard	1	$10.00	$10.00
Servomotor (SG90)	1	$4.00	$4.00
Motor (DC) reductor amarillo	2	$2.50	$5.00
Buzzer activo	1	$1.00	$1.00
Puente H (L298N)	1	$3.00	$3.00
Cables de alimentación	3m	$1.00/m	$3.00
Subsistema de dispensación de agua			
Componentes eléctricos			
Mini bomba de agua	2	$3.50	$7.00
Microcontrolador (ESP32)	1	$15.00	$15.00

Sensor ultrasónico (HC-SR04)	2	$2.00	$4.00
Sensor infrarrojo (FC-51)	1	$2.00	$2.00
Módulo rele 1 canal 5V	2	$2.50	$5.00
Protoboard	1	$10.00	$10.00
Cables de alimentación	3m	$1.00	$3.00
Buzzer activo	1	$1.00	$1.00
Alimentación eléctrica			
UPS	1	$38.00	$38.00
Adaptador 2 USB 5V 3.4 A	1	$9.48	$9.48
Total			$252.8

Fuente: Autoría propia

El análisis de costo-beneficio de la implementación de la jaula de codornices evidencia la relación favorable en la inversión realizada y los beneficios obtenidos, el proyecto tiene una inversión total de $252.8 destinada solo a materiales y componentes electrónicos usados para la construcción del sistema, A esta inversión no se incluye costos de software o licencias, ni costos de la mano de obra, ni la parte de soportes, ni tornillería, ni pintura. En cuanto al beneficio obtenido tras la construcción e implementación del sistema se tiene un ahorro del tiempo ya que con la implementación del sistema se reduce la necesidad de la supervisión manual, aumentando el tiempo personal para dedicarlo a otras actividades, además como beneficio principal es la obtención de huevos de mayor tamaño, mejor calidad y mucho más higiénicos derivando en un impacto económico positivo, de igual modo otro beneficio es el aprovechamiento de los recursos obtenidos por el criadero como los desechos y residuos producidos por las codornices, los cuales sirven como abono orgánico contribuyendo al aumento de la producción agrícola.

Conclusiones y Recomendaciones

Conclusiones

- De acuerdo a los resultados obtenidos podemos evidenciar que la implementación del sistema IoT para el control de alimento mejora significativamente el control del proceso productivo, ya que incrementan la cantidad de huevos, además de la calidad ya que son muchos más grandes y resistentes, por otro lado tiene la ventaja de monitoreo y control del alimento a comparación del método tradicional, confirmando que el sistema propuesto no solo optimiza la gestión dentro del criadero, sino que también contribuye de manera directa al aumento de la productividad, con esto se cumple con el objetivo general y alcance propuesto.
- El correcto funcionamiento del sistema no se lo evidencia únicamente a través del aumento en la cantidad de huevos producidos y su calidad, además se lo evalúa de acuerdo a las mejoras presentadas como en la mejora de la higiene dentro del criadero de las codornices, así como una correcta organización de los procesos para la dispensación de alimento y agua, además del ahorro de tiempo debido a la reducción de la intervención manual para el proceso de alimentación.
- De acuerdo a los resultados obtenidos, se pudo evidenciar que el incremento de la producción de huevos se da por la implementación del sistema IoT de control de alimento, además de un buen diseño de la estructura de la jaula y principalmente a la dispensación controlada de alimento y al suministro continuo de agua en horarios programados.
- El balanceado Exibal demostró ser el balanceado más óptimo ya que, mediante su implementación en el sistema, este aumentó la producción de huevos a comparación del morochillo, además de tener un costo ligeramente mas económico en el mercado actual, convirtiéndola en un balanceado ideal para la etapa de postura de las codornices.

- Gracias a la implementación del dashboard el coturnicultor puede llevar un control preciso de la producción diaria, semanal y mensual, además de estar en constante monitoreo en cuanto al estado de los contenedores, lo que permite que el coturnicultor pueda abastecer de alimento y agua al contenedor y tanque solo cuando los niveles estén en un estado crítico.
- Se concluye que la integración del sistema de notificaciones mediante Telegram es una herramienta demasiado útil debido a que cumple perfectamente con su función la cual es alertar al productor sobre eventos críticos, como bajos niveles de alimento o agua, con ello se garantiza una respuesta oportuna antes estos eventos presentados.
- Las pruebas de funcionamiento confirmaron que el sistema funciona de forma estable, incluso ante la perdida de la conectividad a internet ya que el sistema sigue con sus funciones evitando así dejar a las codornices sin alimento y agua.

Recomendaciones

- Durante al abastecimiento de comida y agua, se recomienda hacer caso a la alerta sonora, ya que, en caso de no hacerlo, esto puede dar como resultado desbordamientos de agua y alimento, lo que puede dar como resultado desperdicio de recursos además de un funcionamiento inadecuado del sistema.
- Se recomienda realizar un mantenimiento preventivo de los componentes electrónicos y estructura de la jaula cada 5 semanas, con el fin de garantizar la precisión de las mediciones y prolongar la vida útil del sistema, como de la limpieza de la carcasa de los componentes, el canal de recolección de huevos, así como el bebedero y parte de la banda transportadora.
- En cuanto al uso del dashboard se recomienda revisar la página 255, 256 y 257, ya que aquí se muestra el uso del dashboard, esto con el fin de asegurar una correcta interpretación de los datos y una adecuada toma de decisiones de acuerdo a la información almacenada.
- Para futuros trabajos, se recomienda tomar como base el diseño de este sistema como la ubicación del contenedor de alimento, el taque de agua, ubicación del bebedero y banda transportadora, con el fin de implementarlo en sistemas mucho más grandes donde el número de aves sea mucho mayor al establecido en el proyecto.

Bibliografía

Echavarria, C. A., & Acero Castillo, R. L. (2023). *Aprovisionamiento de alimento y de agua para pollos de engorde con dispositivos IoT.* Obtenido de https://repository.usta.edu.co/bitstream/handle/11634/51021/2023CarloEchavarriaRafaelAcero.pdf?sequence=1

Lindao Vera, E. E. (2023). *Análisis sobre la rentabilidad de la producción y comercialización de la codorniz en la región Costa del Ecuador.* Obtenido de http://dspace.utb.edu.ec/bitstream/handle/49000/14946/E-UTB-FACIAG-%20AGROP-000078.pdf?sequence=1&isAllowed=y

Valle Muñoz, S. A., Bustamante Castro, M. G., Rodríguez, R. A., Guillet, H., & Vivas, J. (2015). *Manual Crianza y manejo de codornices*. Obtenido de https://repositorio.una.edu.ni/3323/1/tnl01v181.pdf

Zosimo, J. (07 de agosto de 2023). *Cómo almacenar huevos de codorniz.* Obtenido de https://www.thehappychickencoop.com/how-to-store-quail-eggs/

AGElectrónica. (17 de 06 de 2024). *SENSOR DE PROXIMIDADINFRARROJO FC-51*. Obtenido de https://agelectronica.lat/pdfs/textos/O/OKY3127.PDF

Agrotendencia. (10 de abril de 2020). *Codorniz: tipos, beneficios, propiedades y cuidados.* Obtenido de https://agrotendencia.tv/agropedia/avicultura/la-cria-de-codorniz/

Agudelo, F. D., Víctor Libardo, H. N., & Torres Novoa, D. M. (2021). *INGREDIENTES ALTERNATIVOS EN LA ALIMENTACIÓN DE CODORNICES.* Obtenido de https://www.researchgate.net/publication/350743636_Ingredientes_alternativos_en_la_alimentacion_de_codornices

Amazon Web Services. (2023). *¿Qué es IoT (Internet de las cosas)?* Obtenido de https://aws.amazon.com/es/what-is/iot/

Amazon Web Services. (2025). *AWS IoT Core.* Obtenido de https://aws.amazon.com/es/iot-core/

Angelfire. (03 de noviembre de 2001). *Cría de codornices*. Obtenido de https://www.angelfire.com/ia2/ingenieriaagricola/avicultura_codornices.htm#CONDICIONES%20AMBIENTALES:

Arvind Anticor. (2024). *Industrial hoppers: Types, applications, and benefits.* Obtenido de https://arvindanticor.com/industrial-hoppers-types-applications-and-benefits/

AV Electronics. (2025). *Micro DC Motor 3V-6V 8000RPM.* Obtenido de https://avelectronics.cc/producto/micro-dc-motor-3v-6v-8000rpm/

Bioalimentar. (2024). Obtenido de https://www.bioalimentar.com/balanceados

Biomentos Aves. (2025). *Balanceados BiOmentos Aves.* Obtenido de https://es.scribd.com/document/797539236/Balanceados-BiOmentos-Aves

bismark. (27 de septiembre de 2018). *Tipos de sensores para IoT*. Obtenido de https://bismark.net.co/tipos-de-sensores-para-iot/

BM Editores . (09 de junio de 2022). *La automatización, clave para reducir costos de producción.* Obtenido de https://bmeditores.mx/avicultura/la-automatizacion-clave-para-reducir-costos-de-produccion/

BolanosDJ. (2018). *Buzzer Activo – Especificaciones del buzzer.* Obtenido de https://www.bolanosdj.com.ar/MOVIL/ARDUINO2/BuzzerActivo.pdf

Cevallos Suarez, M. P. (octubre de 2015). *PLAN DE DESARROLLO Y ORDENAMIENTO TERRITORIAL PDOT, DE LA PARROQUIA "LA ESPERANZA" 2015 – 2019.* Obtenido de Gobierno Autónomo Descentralizado Parroquial Rural "La Esperanza": https://www.imbabura.gob.ec/phocadownloadpap/K-Planes-programas/PDOT/Parroquial/PDOT%20LA%20ESPERANZA.pdf

Clima.com. (2024). *Clima.com.* Obtenido de https://www.clima.com/ecuador/imbabura/hacienda-san-clemente-2

Components101. (2021). *L298N Dual H-Bridge Motor Driver.* Obtenido de https://components101.com/sites/default/files/component_datasheet/L298N-Motor-Driver-Datasheet.pdf

Components101. (2023). *5V Single-Channel Relay Module.* Obtenido de https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet

Cumpa Gavidia, M. (10 de septiembre de 2009). *CRIANZA Y MANEJO DE CODORNICES.*

Cumpa Gavidia, M. (28 de marzo de 2022). *Nutrición y alimentación de las codornices japonesas (Parte III).* Obtenido de https://actualidadavipecuaria.com/nutricion-y-alimentacion-de-las-codornices-japonesas-parte-iii/

Derelí Fídan, E., & Kaya, M. (11 de septiembre de 2023). *Effect of LED Light Color and Stocking Density on Growth Performance,Carcass, and Meat Quality Characteristics of Japanese Quails.* Obtenido de https://vetdergikafkas.org/uploads/pdf/pdf_KVFD_3028.pdf

Diers, P. (abril de 2025). *¿Es el PVC seguro para el agua potable? La verdad sobre las tuberías de PVC.* Obtenido de https://www.vinylinfo.org/news/is-pvc-safe-for-drinking-water-the-truth-about-pvc-pipes/

Dualtronica. (2025). *Micro Servo Giro continuo FS90R 1.5 kg-cm, 360°.* Obtenido de https://dualtronica.com/motores/523-micro-servo-giro-continuo-fs90r-15-kg-cm-360.html

Dusun. (2023). *El mejor microcontrolador para IoT para mejorar su desarrollo de IoT.* Obtenido de https://www.dusuniot.com/es/blog/best-microcontroller-for-iot-to-elevate-your-iot-development/

Egg Chicken Cage. (2025). *Quail Cage Systems: Maximizing Space, Efficiency, and Profitability in Poultry Farming.* Obtenido de https://www.eggchickencage.com/quail-cage/

El Productor. (21 de junio de 2017). *Cría de codornices, una actividad rentable y económica.* Obtenido de https://elproductor.com/2017/06/cria-de-codornices-una-actividad-rentable-y-economica/

El Sitio Avícola. (18 de diciembre de 2014). *Rendimiento de codornices reproductoras alimentadas con harina de Jatropha*. Obtenido de https://www.elsitioavicola.com/articles/2652/rendimiento-de-codornices-reproductoras-alimentadas-con-harina-de-jatropha/

ElecFreaks. (2025). *Ultrasonic Ranging Module HC - SR04.* Obtenido de https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

ElectroStore. (2025). *MINI BOMBA DE AGUA SUMERGIBLE 5V.* Obtenido de https://www.grupoelectrostore.com/producto/mini-bomba-de-agua-sumergible-5v/

ElectroStore. (2025). *RASPBERRY PI PICO W (WIFI).* Obtenido de https://www.grupoelectrostore.com/producto/raspberry-pi-pico-w-wifi/

ElectroStore. (2025). *RELÉ ESTADO SÓLIDO SSR-25DA 25A INPUT 3-32VDC OUTPUT 24-380VAC.* Obtenido de https://www.grupoelectrostore.com/producto/rele-estado-solido-ssr-25da-25a-input-3-32vdc-output-24-380vac/

ElectroStore. (2025). *SENSOR NIVEL DE AGUA ZPC1 FLOTADOR EN ÁNGULO ALTO VOLTAJE.* Obtenido de https://www.grupoelectrostore.com/producto/sensor-nivel-de-agua-zpc1-flotador-en-angulo-alto-voltaje/

Especificar. (13 de diciembre de 2021). *Todo sobre los actuadores*. Obtenido de https://especificarmag.com.mx/todo-sobre-los-actuadores-html/#:~:text=Tipos%20de%20actuadores&text=Los%20actuadores%20hidr%C3%A

1ulicos%20utilizan%20l%C3%ADquido,para%20producir%20el%20movimiento%20deseado

Exibal. (2024). *Exibal.* Obtenido de https://www.exibal.com/

Exibal. (s.f.). *ALIMENTO CONCENTRADO PARA POLLOS FORMULA ESPECIAL.* Obtenido de https://www.exibal.com/areas-de-negocios/linea-pecuaria/pollos/

FDI Poultry Equipment. (2018). *Jaula para codorniz ponedora/reproductora.* Obtenido de Características y beneficios de la jaula apilada para codornices ponedoras: https://www.fdipoultryequipment.com/wp-content/uploads/2020/12/Quail_Info_Pamphlet_2018_SpanishLatinAmerica.pdf

Ferrantinet. (2026). *Jaula para codornices para puesta.* Obtenido de https://www.ferrantinet.com/en/quails-cages/96-cage-for-laying-quail.html

Firebase. (2025). *Firebase.* Obtenido de https://firebase.google.com/?gclsrc=aw.ds&gad_source=1&gad_campaignid=12302357971&gbraid=0AAAADpUDOhvfWZZSwxZXLS2iHuI74NI0&gclid=Cj0KCQiAgbnKBhDgARIsAGCDdlehRUROdlLOcSy7Sv_HqPKXnS8i_yXM_K0NSAEzefyGtljt7WAdA5AaAsnzEALw_wcB&hl=es-419

Flores Rivera, J. G. (2019). *Evaluación de la calidad del huevo en codornices japonesas (Coturnix coturnix japónica) a diferentes días de conservación en el CIPCA.* Obtenido de https://repositorio.uea.edu.ec/bitstream/123456789/586/1/T.AGROP.B.UEA.1107.pdf

FriendlyWires. (2023). *SG90 Servo Datasheet.* Obtenido de https://www.friendlywire.com/projects/ne555-servo-safe/SG90-datasheet.pdf

Gallinas el extremeño. (2024). *Codorniz japonesa [Fotografía].* Obtenido de https://gallinaselextremeno.jimdofree.com/otros-enlaces/patr%C3%B3n-morfol%C3%B3gico/codorniz-japonesa/

García Vaca , B. I., & Mora Cruz, F. R. (2021). *DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO IOT PARA EL MONITOREO DE PARAMETROS AMBIENTALES APLICADOS A LA AVICULTURA PARA LA CRIANZA DE POLLOS DE GRANJA UTILIZANDO HARDWARE DE BAJO COSTO Y AWS (AMAZON WEB SERVICES).* Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/21122/1/UPS-GT003439.pdf

Garduño Aparicio, A. (2015). *ADQUISICIÓN DE ESPECTROS ÓPTICOS PARA LA ESTIMACIÓN DE TEMPERATURA ELECTRÓNICA.* Obtenido de https://core.ac.uk/download/pdf/76002146.pdf

GEEKFLARE. (14 de mayo de 2024). *Protocolos de comunicación IoT: MQTT vs. CoAP vs. HTTP.* Obtenido de https://geekflare.com/es/mqtt-vs-coap-vs-http/

Generación IoT. (12 de julio de 2022). *IoT en la agricultura inteligente: Modernizar la avicultura con dispositivos conectados.* Obtenido de https://internetdelascosas.xyz/articulo.php?id=597&titulo=IoT-en-la-agricultura-inteligente-modernizar-la-avicultura-con-dispositivos-conectados

Gobierno Autónomo Descentralizado Municipal San Miguel de Ibarra. (25 de marzo de 2025). *PLAN DE DESARROLLO Y ORDENAMIENTOTERRITORIAL CANTÓN 2023-2027.* Obtenido de https://www.ibarra.gob.ec/site/download/plan-de-desarrollo-y-ordenamiento-territorial-pdot-2024-2027/

Gobierno Parroquial Rural “La Esperanza”. (octubre de 2015). *PLAN DE DESARROLLO Y ORDENAMIENTO TERRITORIAL PDOT, DE LA PARROQUIA “LA ESPERANZA” 2015 – 2019.* Obtenido de Gobierno Autónomo Descentralizado Parroquial Rural “La Esperanza”: https://www.imbabura.gob.ec/phocadownloadpap/K-Planes-programas/PDOT/Parroquial/PDOT%20LA%20ESPERANZA.pdf

GoDaddy. (26 de noviembre de 2024). *Telegram: Qué es, cómo funciona y por qué usarlo como app de mensajería.* Obtenido de https://www.godaddy.com/resources/latam/general/que-es-telegram

González Montaño , J. F. (10 de noviembre de 2023). *IMPLEMENTACIÓN DE UN SISTEMA AUTOMÁTICO PARA EL CONTROL Y MONITOREO DE PROCESOS DE ALIMENTACIÓN AVÍCOLA UTILIZANDO UN PLC LOGO 8 EN LA CIUDAD DE ZAMORA EN EL PERIODO ABRIL - SEPTIEMBRE 2023*. Obtenido de http://dspace.tecnologicosudamericano.edu.ec/jspui/bitstream/123456789/793/1/Proyecto%20de%20Titulacion%20Jimmy%20Gonzalez.pdf

Grapsas, T. (16 de septiembre de 2018). *¿Qué es cloud computing o computación en la nube? Conoce sobre el término a continuación*. Obtenido de https://rockcontent.com/es/blog/computacion-en-la-nube/

Imbaquingo Nazate , N. P. (diciembre de 2019). *EVALUACIÓN DE TRES NIVELES DE HARINA DE BLEDO (Amaranthus retroflexus) EN DIETAS PARA CODORNICES (Coturnix coturnix japónica) EN LA ETAPA DE POSTURA EN LA GRANJA EXPERIMENTAL LA PRADERA, CHALTURA* . Obtenido de https://repositorio.utn.edu.ec/bitstream/123456789/9974/2/03%20AGP%20255%20TRABAJO%20DE%20GRADO.pdf

INEC. (abril de 2024). *Encuesta de Superficie y Producción Agropecuaria Continua (ESPAC).* Obtenido de https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_agropecuarias/espac/2023/Boletin_tecnico_ESPAC_2023.pdf

Instituto Nacional de Investigaciones Agropecuarias. (s.f.). *Maiz duro.* Obtenido de Zea maiz: https://tecnologia.iniap.gob.ec/maiz-duro/

Istec. (05 de enero de 2022). *Internet of things, IoT*. Obtenido de https://www.istecdigital.es/internet-of-things-iot/

italcol. (s.f.). *Maíz Amarillo.* Obtenido de https://italcol.com/producto/maiz-amarillo/

JA-bots. (2025). *Sensor de Distancia ToF VL53L0X.* Obtenido de https://ja-bots.com/producto/sensor-de-distancia-tof-vl53l0x/

Jahnke, A. (31 de julio de 2020). *Las 4 etapas de la arquitectura IoT.* Obtenido de https://es.digi.com/blog/post/the-4-stages-of-iot-architecture

Kamami. (2025). *Grove Human Presence Sensor - module with AK9753 motion sensor.* Obtenido de https://kamami.pl/en/retired-products/583166-grove-human-presence-sensor-module-with-ak9753-motion-sensor-101020554.html

Lima Pambaquishpe , L. A. (2023). *"APICULTURA INTELIGENTE MEDIANTE EL USO DE SENSORES QUE PERMITAN EL MONITOREO Y DIAGNÓSTICO DEL ESTADO DE SALUD DE LA COLMENA EVITANDO LA PERDIDA, EN EL SECTOR DE IMBAYA.*

Ludke, M., Pimentel, A., Ludke, J., Silva, J., Rabello, C., & Santos, J. (2018). *Laying performance and egg quality of Japanese quails fed diets containing castor meal and enzyme complex.* Obtenido de SciELO Brasil: https://www.scielo.br/j/rbca/a/N7BDYWTWMr8Zvv4tz5SmsCD/?format=pdf&lang=en

Mapcarta. (2025). Obtenido de https://mapasamerica.dices.net/ecuador/mapa.php?nombre=San-Clemente&id=10923

Martínez Zambrano, C. X. (2013). *ENGORDE DE LA CODORNIZ (Coturnix coturnix japónica) SIN SEXAR CON TRES PROMOTORES DE CRECIMIENTO EN LA ZONA DE MOCACHE.* Obtenido de https://repositorio.uteq.edu.ec/server/api/core/bitstreams/17c2db91-c096-4799-9e93-b478abb8b310/content

MaxBotix. (2025). *MB7076 XL-MaxSonar-WRLA1*. Obtenido de https://maxbotix.com/products/mb7076

Megatronica. (2025). *Módulo Esp8266 Ch340 Nodemcu V3 Lua Wifi con puerto tipo C.* Obtenido de https://megatronica.cc/producto/modulo-esp8266-ch340-nodemcu-v3-lua-wifi-con-puerto-tipo-c/

Megatronica. (2025). *Módulo Puente H L298 L298n 2a Robot Driver Motor.* Obtenido de https://megatronica.cc/producto/modulo-puente-h-l298-l298n-2a-robot-driver-motor/

Megatronica. (2025). *Motor reductor Motor Dc Reductora Amarillo Arduino 3V a 9V BIAXIAL.* Obtenido de https://megatronica.cc/producto/motor-reductor-motor-dc-reductora-amarillo-arduino-3v-a-9v-bi-axial/

Mendieta Suárez, E. F. (2015). *EFECTO DE LA ADICIÓN DE MICROORGANISMOS BENÉFICOS (Rhodopseudomonas spp, Lactobacillus spp, Sacharomyces spp), EN LA PRODUCCIÓN DE HUEVOS DE CODORNIZ (Coturnix coturnix japónica).* Obtenido de https://dspace.unl.edu.ec/jspui/bitstream/123456789/11260/1/Edison%20Mendieta%20AARN.pdf

Microsoft. (2025). *Azure IoT Hub.* Obtenido de https://azure.microsoft.com/es-es/products/iot-hub

mokolora. (14 de septiembre de 2021). *Comparación entre LoRa y otras tecnologías inalámbricas.* Obtenido de https://www.mokolora.com/es/lora-and-wireless-technologies/

Naciones Unidas Ecuador. (2024). *Acerca de nuestro trabajo para los Objetivos de Desarrollo Sostenible en Ecuador.* Obtenido de Los Objetivos de Desarrollo Sostenible en Ecuador: https://ecuador.un.org/es/sdgs

Novatronic. (2020). *Arduino MEGA 2560 R3 Version CH340.* Obtenido de https://novatronicec.com/index.php/product/arduino-mega-chip-ch340/

Novatronic. (2020). *Bomba de agua de diafragma.* Obtenido de https://novatronicec.com/index.php/product/bomba-de-agua-de-diafragma/

Novatronic. (2020). *Buzzer Activo 5V 14x7mm 4KHz 2mA.* Obtenido de https://novatronicec.com/index.php/product/buzzer-activo-5v-14x7mm-4khz-2ma/

Novatronic. (2020). *Buzzer Pasivo 5V 14x7mm 4KHz 2mA.* Obtenido de https://novatronicec.com/index.php/product/buzzer-pasivo-5v-14x7mm-4khz-2ma/

Novatronic. (2020). *Modulo de relé de 1 canal con salida opto acoplada 5V.* Obtenido de https://novatronicec.com/index.php/product/modulo-de-rele-de-1-canal-con-salida-opto-acoplada-5v/

Novatronic. (2020). *Motor alto torque JGY370 DC12V 10RPM.* Obtenido de https://novatronicec.com/index.php/product/motor-alto-torque-jgy370-dc12v-10rpm/

Novatronic. (2020). *Sensor de movimiento PIR.* Obtenido de https://novatronicec.com/index.php/product/sensor-de-movimiento-pir/

Novatronic. (2020). *Servo SG90 9g 180 grados.* Obtenido de https://novatronicec.com/index.php/product/servo-sg90-9g-180-grados-2/

Novatronic. (2020). *TB6612FNG Driver para motor DC.* Obtenido de https://novatronicec.com/index.php/product/tb6612fng-driver-para-motor-dc/

Oficina Estatal de Información para el Desarrollo Rural Sustentable . (junio de 2009). *Encuesta y consulta bibliográfica sobre codorniz*. Obtenido de https://www.nacionmulticultural.unam.mx/empresasindigenas/docs/1925.pdf

Otálora, R. (2017). *Sistema de producción de codornices.* Obtenido de https://avinews.com/sistemas-produccion-codornices/

Pérez Clemente, J. E. (2023). *EVALUACIÓN DEL SISTEMA DE MONITOREO INTELIGENTE CON IoT EN GRANJA AVÍCOLA.* Obtenido de https://repositorio.lamolina.edu.pe/bitstream/handle/20.500.12996/5843/perez-clemente-jorge-enrique.pdf?sequence=1&isAllowed=y

Quiroz Martínez, S. J. (2017). *ESTUDIO DE FACTIBILIDAD PARA LA CREACIÓN DE UNA MICROEMPRESA PRODUCTORA Y COMERCIALIZADORA DE PRODUCTOS DERIVADOS DE LA CODORNIZ EN LA CIUDAD DE IBARRA, PROVINCIA DE IMBABURA.* Obtenido de https://repositorio.utn.edu.ec/bitstream/123456789/8289/1/02%20IEF%20193%20TRABAJO%20DE%20GRADO.pdf

RD Station Marketing. (18 de noviembre de 2022). *¿Para qué sirve WhatsApp?: guía definitiva y completa.* Obtenido de https://www.rdstation.com/blog/es/para-que-sirve-whatsapp/

Revelo, C., & Terán , N. (Enero de 2012). *ESTUDIO DE FACTIBILIDAD PARA LA CREACIÓN DE UNA MICROEMPRESA PRODUCTORA Y COMERCIALIZADORA DE HUEVOS DE CODORNIZ, ENFOCADA A LA REINSERCIÓN LABORAL DEL ADULTO MAYOR EN LA CIUDAD DE IBARRA.* Obtenido de https://repositorio.utn.edu.ec/bitstream/123456789/1683/1/02%20IEF%20032%20TESIS.pdf

Rodas Z., D. (Junio de 2004). *Proyecto de Factibilidad de Cría, Producción y Comercialización de huevos de codorniz(Coturnix coturnix japónica), en la provincia de Pichincha*. Obtenido de https://core.ac.uk/download/pdf/147371217.pdf

Sagñay Sagñay , J. K. (2021). *POTENCIAL PRODUCTIVO DE LA CODORNIZ JAPONESA (Coturnix coturnix japónica) EN EL ECUADOR.* Obtenido de http://dspace.espoch.edu.ec/bitstream/123456789/15616/1/17T01643.pdf

Salazar, J. (2021). *Redes inalámbricas*. Obtenido de https://upcommons.upc.edu/bitstream/handle/2117/100918/LM01_R_ES.pdf

Sánchez Arias, S. (24 de mayo de 2023). *Huevos de codorniz: valor nutricional y beneficios.* Obtenido de https://mejorconsalud.as.com/huevos-codorniz/

Sembralia. (18 de septiembre de 2021). *Jaulas para codornices ponedoras y de engorde. Tipos, recomendaciones y medidas de batería de 1, 2 y 3 pisos*. Obtenido de https://sembralia.com/blogs/blog/jaulas-para-codornices

SHENZHEN Zhongke Century Technology Co. (2025). *5V DC sin escobillas de pequeña mascota Mini Dispensador de agua sumergibles de tanque de peces de acuario de agua de fuente de la bomba de circulación Immersible.* Obtenido de https://es.made-in-china.com/co_szzksj/product_DC-5V-Small-Brushless-Pet-Mini-Water-Dispenser-Submersible-Aquarium-Fish-Tank-Fountain-Water-Circulation-Immersible-Pump_regengnig.html

Shijiazhuang Kangfa Machinery Equipment Co. (agosto de 2025). *Por qué elegir jaulas para pollos de acero galvanizado para su gallinero.* Obtenido de https://www.kffarming.com/news/galvanized-steel-chicken-cage-for-layers.html

SIISA GLOBAL. (21 de julio de 2021). *Microcontroladores. ¿Qué son? y su importancia en la industria.* Obtenido de https://es.linkedin.com/pulse/microcontroladores-qu%C3%A9-son-y-su-importancia-en-la-industria-

SparkFun Electronics. (2015). *HC-SR04 Ultrasonic Range Sensor Datasheet.* Obtenido de https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Tapia Garófalo , X. A. (2010). *EVALUACIÓN ECONÓMICA DE DIFERENTES NIVELES DE PROTEÍNA BRUTA UTILIZADOS EN LA ALIMENTACIÓN PARA CODORNICES EN PRODUCCIÓN.* Obtenido de

https://dspace.unl.edu.ec/jspui/bitstream/123456789/5533/1/TAPIA%20GAR%C3%93FALO%20XIMENA.pdf

Teca Méndez, E. E. (2017). *MANIFESTACIONES CULTURALES INMATERIALES COMO EJE DEL TURISMO SUSTENTABLE EN LA COMUNIDAD DE SAN CLEMENTE,PARROQUIA LA ESPERANZA, PROVINCIA DE IMBABURA".* Obtenido de https://repositorio.utn.edu.ec/bitstream/123456789/7245/1/PG%20525%20TESIS.pdf

Tecmikro. (2025). *Servomotor MG995 Tower Pro.* Obtenido de https://tecmikro.com/motores/408-servomotor-mg995-tower-pro.html

Terán Caicedo, N. A. (2020). *Estudio de factibilidad para la creación de una microempresa incubadora de producción y comercialización de huevos de codornices en la ciudad de Ibarra provincia de Imbabura.* Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/18925/1/UPS-MSQ031.pdf

uElectronics. (febrero de 2018). *Hoja de datos – Bomba de agua sumergible para Arduino.* Obtenido de https://uelectronics.com/wp-content/uploads/2018/02/Hoja-de-datos-Bomba-de-Agua-sumergible-Arduino-1.pdf

uElectronics. (agosto de 2023). *ESP32 DevKit V1 USB-C y Micro-USB – Pinout.* Obtenido de https://uelectronics.com/wp-content/uploads/2023/08/ESP32-DEVKIT-V1-usbc-y-microusb-Pinout.pdf

Unir. (17 de mayo de 2023). *La arquitectura IoT y sus usos en distintos sectores*. Obtenido de https://www.unir.net/ingenieria/revista/arquitectura-iot/#:~:text=La%20arquitectura%20IoT%20es%20un,donde%20ser%C3%A1n%20procesados%20y%20almacenados.

Unit Electronics. (2025). *ESP32 DEVKIT V1 30 Pines USB-C/MicroUSB.* Obtenido de https://uelectronics.com/producto/esp32-devkit-v1-30-pines-usb-c-microusb/

UnitElectronics. (2025). *Seguidor de Linea TCRT5000 Optico Infrarrojo.* Obtenido de https://uelectronics.com/producto/sensor-seguidor-de-linea-tcrt5000-optico-infrarrojo/

Valle Muñoz, S. A., Bustamante Castro, M. G., Rodríguez, R. A., Vivas, J. A., & Guillet, H. (2015). *Manual crianza y manejo de codornices.* Obtenido de https://repositorio.una.edu.ni/3323/1/tnl01v181.pdf

VILLACIS VIVAR , L. P., & VIZHCO MINCHALA , C. I. (2016). *EVALUACIÓN DE DOS TIPOS DE FITASA SOBRE LA PRODUCTIVIDAD Y CALIDAD DEL HUEVO EN CODORNICES*. Obtenido de https://dspace.ucuenca.edu.ec/bitstream/123456789/23619/1/Tesis-Fitasa-Codorniz.pdf

Zendesk. (26 de junio de 2024). *TOP 5 aplicaciones de mensajería instantánea para empresas*. Obtenido de https://www.zendesk.com.mx/blog/aplicaciones-mensajeria/

Anexos

Anexo A. Entrevista realizada al coturnicultor

La entrevista fue realizada hacia el coturnicultor con el fin de obtener datos necesarios para la construcción de sistema de control de alimentación, como se puede evidenciar en la **Figura *136*** . A continuación, se agrega la entrevista con las preguntas abordadas además de las respuestas obtenidas.

Entrevista dirigida al propietario de las codornices

Fecha: 18 / 06 / 2025

Entrevistador/a: Edison Steev Quilca Serrano

Entrevistado/a: Luis Enrique Pupiales Brucil

Objetivo: Recopilar información relevante sobre el estado actual del criadero, identificando las necesidades del usuario y los requerimientos operativos necesarios para el diseño e implementación del sistema IoT para el control de alimentación de coturnix japonica (codorniz japonesa).

Durante la entrevista se abordaron aspectos relacionados con el manejo actual del criadero, así como las proyecciones y expectativas a futuro respecto a su mejora y optimización. La información obtenida permitió identificar las necesidades del usuario y establecer lineamientos generales que sirvieron como base para la definición de los requerimientos del sistema y su posterior diseño e implementación.

Preguntas de la Entrevista

¿Podría describir cómo funciona actualmente el criadero de codornices en cuanto a la alimentación y el suministro de agua?

Respuesta:

El entrevistado menciona que actualmente, el alimento se coloca directamente en botellas plástica de la forma tradicional, en horarios irregulares debido a su tiempo laboral, mientras que el agua se repone en horarios irregulares.

2. ¿Qué dificultades presenta el manejo manual del alimento y del agua en el criadero?

Respuesta:

El entrevistado supo mencionar que el manejo manual requiere presencia constante en el criadero, lo que toma tiempo. Además, en ocasiones no se detecta a tiempo cuando el alimento o el agua se están agotando, lo que puede afectar el estado nutricional de las codornices.

3. ¿Con qué frecuencia revisa el nivel de alimento y agua?

Respuesta:

Según la respuesta del entrevistado menciona que revisa varias 1 ves al día especialmente en horas de la mañana, aunque no existe un control exacto ni registros del consumo de alimento y agua.

4. ¿Ha tenido problemas por falta o exceso de alimento o agua?

Respuesta:

el entrevistado refiere que sí ha tenido problemas, debido que en ocasiones el alimento se ha terminado antes de lo previsto, por falta de una dosis adecuada. En el caso del agua, también genera problemas debido a que se ensucia y no tiene un día exacto de cambio de agua.

5. ¿Considera importante conocer el nivel de alimento en el comedero?

Respuesta:

El entrevistado menciona que sí, es importante ya que con esto me permitiría tener el control del tiempo y solo se acercaría a abastecer cuando el alimento se encuentre en estado crítico.

6. ¿Considera necesario monitorear el nivel de agua del tanque?

Respuesta:

Menciona que sí, ya que el agua es fundamental para tener una cantidad específica para dispensar y también manteniendo el agua limpia.

7. ¿En qué horarios se alimenta a las codornices?

Respuesta:

Como menciona el entrevistado se realiza en horarios como en la mañana, aunque algunas veces pueden variar según la disponibilidad de tiempo.

8. ¿Le resultaría útil una dispensación automática y programada?

Respuesta:

Menciona que sí, ya que le permitiría mantener horarios específicos y reducir la necesidad de la dispensación manual.

9. ¿Con qué frecuencia considera adecuado dispensar el alimento?

Respuesta:

Menciona que sería adecuado hacerlo de forma programada, asegurando una distribución uniforme del alimento durante el día.

10. ¿Qué información le gustaría visualizar del sistema?

Respuesta:

El entrevistado menciona que le gustaría visualizar los niveles de alimento y agua, como también la cantidad de huevos recolectados.

11. ¿Es importante que la información sea clara y fácil de interpretar?

Respuesta:

menciona que sí, debido a que le facilitara la toma de decisiones y el manejo del criadero.

12. ¿Le gustaría recibir notificaciones cuando los niveles sean bajos?

Respuesta:

El entrevistado refiere que sí, sería muy útil recibir alertas para conocer y actuar de manera inmediata.

13. ¿Por qué medio prefiere recibir las notificaciones?

Respuesta:

Menciona que sería mejor mediante mensajes en el teléfono celular, ya que paso más tiempo con este dispositivo y tengo mayor accesibilidad al servicio de internet.

14. ¿Invertiría en un sistema automatizado si mejora la eficiencia?

Respuesta:

menciona que sí, porque me ayudaría a que el sistema reduzca el tiempo, el trabajo manual y mejore el control del criadero.

15. ¿Considera que la automatización mejoraría la eficiencia del criadero?

Respuesta:

El entrevistado menciona que sí, porque las codornices pueden tener una mejor producción.

16. ¿Estaría dispuesto a utilizar el sistema?

Respuesta:

Sí, siempre que sea fácil de usar y confiable.

17. ¿Qué recomendaciones daría para el sistema?

Respuesta:

Que sea de fácil uso, y que las notificaciones sean entendibles.

Figura 136

Evidencia de la entrevista realizada al coturnicultor

Fuente: Autoría propia

Anexo B. Datasheets de los componentes electrónicos

- ESP32 DevKit V1

https://uelectronics.com/wp-content/uploads/2023/08/ESP32-DEVKIT-V1-usbc-y-microusb-Pinout.pdf

https://documentation.espressif.com/esp32_datasheet_en.pdf

- Sensor de distancia HC-SR04

https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

- Servomotor SG90 180°

https://www.friendlywire.com/projects/ne555-servo-safe/SG90-datasheet.pdf

- Buzzer activo

https://www.bolanosdj.com.ar/MOVIL/ARDUINO2/BuzzerActivo.pdf

- Driver controlador de motor L298N

https://components101.com/sites/default/files/component_datasheet/L298N-Motor-Driver-Datasheet.pdf

- Sensor infrarrojo fc51

https://agelectronica.lat/pdfs/textos/O/OKY3127.PDF

- Módulo relé 5v

https://datasheet.lcsc.com/lcsc/1811021636_SONGLE-RELAY-SRD-05VDC-SL-C_C35449.pdf

- Minibomba de agua sumergible

https://uelectronics.com/wp-content/uploads/2018/02/Hoja-de-datos-Bomba-de-Agua-sumergible-Arduino-1.pdf

Anexo C. Código subsistema de dispensador de alimento

```
//TEMA: Código para el dispensador de comida
//AUTOR: Edison Quilca
//*************************Librerías************************
#include <Arduino.h>             // Funciones base
#include <WiFi.h>                 // Conexión WiFi
#include <WiFiClientSecure.h>   // Cliente HTTPS (TLS) para Telegram
#include <Firebase_ESP_Client.h>  // Firebase RTDB
#include <ESP32Servo.h>            // Control de servo
#include <time.h>                 // Hora NTP
#include "esp_task_wdt.h"         // Watchdog

//******************Credenciales red WiFi******************

#define WIFI_SSID     "INNO-FIBER- FLIA PUPIALES" // Nombre de la red WiFi
#define WIFI_PASSWORD "1704Pupiales#"             // Contraseña de la red WiFi

//******************Asignación de pines ******************
#define TRIG_PIN   12   // TRIG del sensor ultrasónico
#define ECHO_PIN   34   // ECHO del ultrasónico
#define SERVO_PIN  13   // Pin del servo
#define IN1        26   // Control motor
#define IN2        25   // Control motor
#define ENA        33   // PWM del motor A
#define IN3        32   // Control motor B
#define IN4        27   // Control motor B
#define ENB        14   // PWM del motor B
#define BUZZER_PIN 4    // Pin del buzzer

//****************** Credenciales Firebase para conectar a la base de datos******************
#define API_KEY      "AIzaSyB3D4scS5m9STZSm2mHy0Kh" // API Key de Firebase
#define DATABASE_URL "https://criaderocj-default-rtdb.firebaseio.com/" // URL RTDB

FirebaseData fbdo;        // Objeto para operaciones en RTDB
FirebaseAuth auth;        // Objeto de autenticación
FirebaseConfig config;    // Configuración de Firebase
bool signupOK      = false; // Marca si el signup fue correcto
bool firebaseReady = false; // Marca si Firebase quedó inicializado

//****************** Parámetros de autenticaicón con telegram******************

const char* TELEGRAM_BOT_TOKEN = "8001400737:SIVTGkDmIAVesbpZxg1uQzekGyeZgg"; // Token del bot
const char* TELEGRAM_CHAT_ID   = "55853275";                                  // ID del chat
destino
WiFiClientSecure telegramClient; // Cliente seguro para conectar a api.telegram.org

Servo dispensador;        // Objeto servo del dispensador
bool ejecutado = false;   // Flag para no dispensar varias veces en el mismo segundo
```

```
// Modelo físico del comedero
const float ALTURA_TOTAL_COMEDERO_CM = 40.0f; // Altura total interna (referencia)
const float ALTURA_LLENO_CM          = 10.0f; // Distancia aprox. cuando está lleno (referencia)

// ================= CONTROL DE BUZZER =================
bool buzzerActivo       = false;          // Indica si está sonando ahora
bool buzzerArmado       = true;           // Permite sonar 1 vez cuando llegue a “lleno”
unsigned long buzzerInicioMs = 0;         // Guarda cuándo se encendió
const unsigned long BUZZER_DURACION_MS = 5000; // Tiempo de sonido (5s)

bool enEstadoNoLleno = false;             // Detecta si salió de “lleno”
unsigned long inicioNoLlenoMs = 0;        // Momento en que entró a “no lleno”
const unsigned long TIEMPO_MIN_NO_LLENO_MS = 30000; // Para rearmar buzzer (30s)

// ********************* FILTRO DE LECTURA*********************
float emaComedero = NAN;                  // EMA (promedio móvil exponencial)

unsigned long ultimoTiempoLectura = 0;    // Última vez que se leyó el sensor
const unsigned long intervaloLectura = 400; // Cada 400 ms (sin saturar)

int ultimaLecturaEstable = -1;            // Lectura final “estable” con histéresis
const int HISTERESIS_CM = 2;              // Cambios <=2 cm se ignoran


// ********************* CONTROL NOTIFICACIÓN "LLENO" *********************
bool comederoLlenoNotificado  = false; // Para no repetir aviso de lleno
bool primeraLecturaComedero   = true;  // Marca primera lectura
int  ultimaDistanciaComedero  = 0;     // Guarda lectura anterior

const int UMBRAL_LLENO_CM      = 12;   // Umbral de “lleno” por distancia
const int REARME_LLENO_DELTA   = 2;    // Histéresis: se rearma al alejarse +2cm

//**********************ENVÍO DIRECTO A TELEGRAM**********************
static String urlEncode(const String &s) {       // Función para codificar texto en URL
  const char *hex = "0123456789ABCDEF";          // Tabla para convertir a hexadecimal
  String out;                                    // String donde se arma el resultado
  out.reserve(s.length() * 3);                   // Reserva memoria

  for (size_t i = 0; i < s.length(); i++) {      // Recorre cada carácter del mensaje
    uint8_t c = (uint8_t)s[i];                   // Lee el byte actual

    // Si es un carácter “seguro” para URL, se deja tal cual
    if ((c >= 'a' && c <= 'z') ||
        c == '-' || c == '_' || c == '.' || c == '~') {
      out += (char)c;                            // Se agrega sin cambios
    } else if (c == ' ') {
      out += "%20";                              // Espacio se reemplaza por %20
    } else {
      out += '%';                                // Otros caracteres se codifican
      out += hex[(c >> 4) & 0x0F];               // Parte alta del byte
      out += hex[c & 0x0F];                      // Parte baja del byte
    }
  }
  return out;                                    // Devuelve el texto codificado
}

static bool telegramSendText(const String &mensaje) {    // Envío HTTP a Telegram
  if (WiFi.status() != WL_CONNECTED) {                    // Si no hay WiFi, no intenta
    Serial.println("[Telegram] No hay WiFi, mensaje NO enviado: " + mensaje);
    return false;                                         // Indica fallo
  }

  telegramClient.setInsecure();                           // No valida certificado
  telegramClient.setTimeout(5);                           // Timeout para no quedarse colgado

  if (!telegramClient.connect("api.telegram.org", 443)) { // Conecta al servidor Telegram
    Serial.println("[Telegram] Falló conexión con api.telegram.org");
    return false;                                          // Indica fallo de conexión
  }

  String safeText = urlEncode(mensaje);                    // Codifica el mensaje para URL

  String url = "/bot";                                     // Prefijo /bot
```

```
  url += TELEGRAM_BOT_TOKEN;                                  // Agrega token
  url += "/sendMessage?chat_id=";                             // Endpoint y parámetro chat
  url += "&text=";                                            // Parámetro text
  url += safeText;                                            // Agrega el mensaje codificado

  telegramClient.print(String("GET ") + url + " HTTP/1.1\r\n"); // Línea GET
  telegramClient.print("Host: api.telegram.org\r\n");          // Header Host
  telegramClient.print("Connection: close\r\n\r\n");          // Cierra al terminar

  bool ok = false;                                           // Bandera de éxito
  unsigned long start = millis();                             // Marca inicio de espera
  String firstLine = "";                                     // Guardará primera línea HTTP

  while (millis() - start < 2000) {                           // Espera hasta 2s respuesta
    if (telegramClient.available()) {                         // Si hay datos disponibles
      firstLine = telegramClient.readStringUntil('\n');       // Lee primera línea
      firstLine.trim();                                       // Quita espacios/saltos
      if (firstLine.startsWith("HTTP/1.1 200")) ok = true;    // 200 = OK
      break;                                                  // Sale del while
    }
    delay(10);                                                // Pequeña pausa
  }

  start = millis();                                           // Reinicia timer
  while (telegramClient.connected() && millis() - start < 1200) { // Consume el resto
  }

  telegramClient.stop();                                      // Cierra conexión

  if (ok) {
    Serial.println("[Telegram] Mensaje enviado: " + mensaje); // Log si se envió
  } else {
    Serial.println("[Telegram] Envío FALLÓ (no 200). Primera línea: " + firstLine);
  }
  return ok;                                                  // Devuelve true/false
}
}

bool enviarMensajeTelegram(const char* mensaje) {          // Wrapper (const char*)
  return telegramSendText(String(mensaje));
}



// ================= LECTURA ULTRASONIDO ==================
static float quickSelect(float* arr, int n, int k) {   // Selecciona el k-ésimo menor (mediana)
  int left = 0, right = n - 1;                          // Límites del arreglo
  while (true) {                                        // Bucle hasta encontrar resultado
    if (left == right) return arr[left];                // Caso base: solo un elemento
    float pivot = arr[(left + right) / 2];              // Pivote aproximado
    int i = left, j = right;                            // Índices de partición
    while (i <= j) {                                    // Particiona en torno al pivote
      while (arr[i] < pivot) i++;                       // Mueve i a la derecha
      while (arr[j] > pivot) j--;                       // Mueve j a la izquierda
      if (i <= j) {                                     // Intercambia si corresponde
        float t = arr[i]; arr[i] = arr[j]; arr[j] = t; // Swap
        i++; j--;                                       // Avanza índices
      }
    }
    if (k <= j)      right = j;                         // Si k quedó a la izquierda
    else             return arr[k];                     // Si k quedó en el “medio”
  }
}

float leerDistanciaCrudaCM(uint8_t trigPin, uint8_t echoPin) { // Lectura directa del HC-SR04
  digitalWrite(trigPin, LOW);            // Asegura TRIG en bajo
  delayMicroseconds(3);                 // Pequeña espera
  digitalWrite(trigPin, HIGH);          // Pulso HIGH
  delayMicroseconds(10);                // 10us de disparo
  digitalWrite(trigPin, LOW);           // Termina pulso

  long duracion = pulseIn(echoPin, HIGH, 25000); // Tiempo del eco (máx 25ms)
  if (duracion <= 0) return 0.0f;               // Si no hubo eco, inválido
```

```
  float d = duracion * 0.0343f / 2.0f;           // Convierte a cm (velocidad sonido)
  if (d < 1.0f || d > 400.0f) return 0.0f;       // Filtra rangos raros
  return d;                                      // Retorna distancia válida
}

int medirEstableInt(uint8_t trigPin, uint8_t echoPin, float &emaRef) { // Lectura filtrada
  const int N = 10;                 // Número de muestras
  float vals[N];                    // Arreglo de lecturas
  int cnt = 0;                      // Contador de lecturas válidas

  for (int i = 0; i < N; i++) {                    // Toma N lecturas
    float v = leerDistanciaCrudaCM(trigPin, echoPin); // Lee distancia cruda
    if (v > 0) vals[cnt++] = v;                    // Guarda solo si es válida
    delay(2);                                      // Pequeña pausa
  }

  if (cnt == 0) {                                  // Si no hubo lecturas válidas
    if (isnan(emaRef)) return 0;                   // Si ni EMA existe, devuelve 0
    return (int)roundf(emaRef);                    // Si existe, usa EMA anterior
  }

  float tmp[10];                                   // Arreglo temporal para mediana
  for (int i = 0; i < cnt; i++) tmp[i] = vals[i];  // Copia datos válidos
  float med = quickSelect(tmp, cnt, cnt / 2);      // Calcula mediana

  float suma = 0;                                  // Suma para promedio recortado
  int ok = 0;                                      // Contador dentro del rango
  float lo = med * 0.90f, hi = med * 1.10f;        // Banda ±10%

  for (int i = 0; i < cnt; i++) {                  // Recorre lecturas válidas
    if (vals[i] >= lo && vals[i] <= hi) {          // Solo si está cerca de la mediana
      suma += vals[i];                             // Suma
      ok++;                                        // Cuenta
    }
  }
  float trimmed = (ok > 0) ? (suma / ok) : med;    // Promedio recortado o mediana

  const float alpha = 0.70f;                       // Peso del EMA
  if (isnan(emaRef)) emaRef = trimmed;             // Inicializa EMA
  else emaRef = alpha * trimmed + (1.0f - alpha) * emaRef; // Actualiza EMA

  return (int)roundf(emaRef);                      // Retorna distancia filtrada (entero)
}

// ********************** DISPENSADOR ************************************
void dispensar() {
  enviarMensajeTelegram("🔄🍽️Inicio de dispensación de comida"); // Envía notificacion de inicio de
dispensación a Telegram
  dispensador.write(180);             // Mueve servo a 180° (abre)
  delay(300);                         // Espera para que el servo llegu
  digitalWrite(IN1, HIGH);            // Activa sentido motor A
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);            // Activa sentido motor B
  digitalWrite(IN4, LOW);
  delay(100);                         // Pequeña espera antes del PWM

  ledcWrite(4, 255);                  // PWM máximo en canal 4 (ENA)
  }
  ledcWrite(4, 0);                    // Apaga PWM motor A
  digitalWrite(IN1, LOW);             // Apaga señales motor A
  digitalWrite(IN2, LOW);
  ledcWrite(5, 0);                    // Apaga PWM motor B
  digitalWrite(IN3, LOW);             // Apaga señales motor B
  digitalWrite(IN4, LOW);
  dispensador.write(0);               // Regresa servo a 0° (cierra)
  delay(300);                         // Espera final
  enviarMensajeTelegram("✔️🍽️250 gramos de comida exibal dispensados correctamente"); // Envía
notificacion de fin dedispensación a Telegram
  Serial.println("[DISPENSADOR] FIN"); // Log al monitor serial
}

// ********************** WIFI**********************
void conectarWiFiRobusto() {
  WiFi.mode(WIFI_STA);                        // Modo estación (cliente)
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);        // Inicia conexión WiFi
  unsigned long inicio = millis();             // Marca inicio
  Serial.print("[WiFi] Conectando");           // Log

  while (WiFi.status() != WL_CONNECTED && millis() - inicio < 15000) { // Espera máx 15s
    delay(300);                                // Espera
    Serial.print(".");                         // Progreso
  }
  Serial.println();                            // Salto de línea

  if (WiFi.status() != WL_CONNECTED) {         // Si no conectó
    Serial.println("[WiFi] No se pudo conectar en 15s. Sigo OFFLINE (solo local).");
  } else {                                     // Si conectó
    Serial.print("[WiFi] Conectado. IP: ");
    Serial.println(WiFi.localIP());            // Muestra IP
  }
}

// Reintento no bloqueante (cada 30s)
void reintentarWiFiNoBloqueante(unsigned long ahora) {
  if (WiFi.status() == WL_CONNECTED) return;         // Si ya hay WiFi, no hace nada
  if (ahora - ultimoIntentoWiFiMs < 30000) return;   // Si no han pasado 30s, no reintenta

  Serial.println("[WiFi] Desconectado, reintentando..."); // Log
  WiFi.disconnect();                                  // Limpia estado
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);               // Reintenta conexión
  ultimoIntentoWiFiMs = ahora;                        // Guarda momento del intento
}

// ****************************FIREBASE**********************************
void iniciarFirebase() {
  if (WiFi.status() != WL_CONNECTED) {       // Si no hay WiFi, no intenta Firebase
    Serial.println("[Firebase] No inicio: no hay WiFi.");
    firebaseReady = false;                   // Marca no listo
    return;                                  // Sale
  }
  Serial.println("[Firebase] Inicializando..."); // Log
  config.api_key      = API_KEY;             // Asigna API key
  config.database_url = DATABASE_URL;        // Asigna URL del RTDB
  if (Firebase.signUp(&config, &auth, "", "")) { // Signup anónimo
    signupOK = true;                         // Marca OK
    Serial.println("[Firebase] SignUp OK");
  } else {                                   // Si falló signup
    signupOK      = false;                   // Marca error
    firebaseReady = false;                   // Firebase no listo
    Serial.print("[Firebase] SignUp ERROR: ");
    Serial.println(config.signer.signupError.message.c_str()); // Mensaje de error
    return;                                  // Sale
  }

  firebaseReady = true;                 // Marca listo
  Serial.println("[Firebase] Listo (modo mejor esfuerzo).");
}
void reintentarFirebaseNoBloqueante(unsigned long ahora) {
  if (firebaseReady) return;                 // Si ya está listo, no reintenta
  if (ahora - ultimoIntentoFbMs < 60000) return; // Reintenta cada 60s
  Serial.println("[Firebase] Reintentando iniciar..."); // Log
  iniciarFirebase();                         // Intenta iniciar otra vez
  ultimoIntentoFbMs = ahora;                 // Guarda momento del intento
}
static String estadoNotificado = "";   // Guarda el último estado que se notificó
unsigned long lastVacioMsgTime   = 0;  // Última vez que mandó mensaje "vacío"
unsigned long lastCriticoMsgTime = 0;  // Última vez que mandó mensaje "crítico"
const unsigned long INTERVALO_VACIO_MS   = 3UL * 60UL * 60UL * 1000UL; // Repetir "vacío" cada 3h
const unsigned long INTERVALO_CRITICO_MS = 1UL * 60UL * 60UL * 1000UL; // Repetir "crítico" cada 1h
// ***************SETUP ************************************
void setup() {
  Serial.begin(115200);                // Inicia comunicación serial

  ledcSetup(4, 1000, 8);                // Canal PWM 4, 1kHz, 8 bits
  ledcAttachPin(ENA, 4);               // ENA queda asociado al canal 4
  ledcWrite(4, 0);                     // Arranca con PWM = 0

  ledcSetup(5, 1000, 8);               // Canal PWM 5, 1kHz, 8 bits
```

```
  ledcAttachPin(ENB, 5);               // ENB asociado al canal 5
  ledcWrite(5, 0);                     // Arranca con PWM = 0

  pinMode(TRIG_PIN, OUTPUT);           // TRIG como salida
  pinMode(ECHO_PIN, INPUT);              // ECHO como entrada
  pinMode(IN1, OUTPUT);                // Pines del driver como salida
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(BUZZER_PIN, OUTPUT);         // Buzzer como salida
  digitalWrite(BUZZER_PIN, LOW);       // Buzzer apagado al inicio

  dispensador.attach(SERVO_PIN);        // Asocia servo al pin indicado
  dispensador.write(0);                // Posición inicial del servo

  conectarWiFiRobusto();               // Intenta conectar WiFi
  iniciarFirebase();                   // Intenta iniciar Firebase

  configTime(-5 * 3600, 0, "pool.ntp.org", "time.nist.gov"); // Config hora NTP (UTC-5)

  esp_task_wdt_init(10, true);         // Config watchdog (10s, reset si se cuelga)
  esp_task_wdt_add(NULL);              // Agrega tarea actual al watchdog

  enviarMensajeTelegram("⚙️🍽️ Sistema de comedero iniciado"); // Aviso de inicio por Telegram
}

// ****************************LOOP ****************************
void loop() {
  esp_task_wdt_reset();                // Reinicia contador del watchdog

  unsigned long ahora = millis();      // Tiempo actual desde que inició el programa

  //  **************************** LECTURA ULTRASONIDO CADA 400 ms + HISTERESIS
****************************
  static int distFiltrada = 0;         // Mantiene el valor filtrado entre iteraciones

  if (ahora - ultimoTiempoLectura >= intervaloLectura) {   // Si ya pasó el intervalo
    distFiltrada = medirEstableInt(TRIG_PIN, ECHO_PIN, emaComedero); // Lee filtrado
    ultimoTiempoLectura = ahora;                            // Actualiza tiempo de lectura
  }

  if (ultimaLecturaEstable < 0) {     // Primera vez que entra
    ultimaLecturaEstable = distFiltrada; // Toma lectura inicial
  } else {
    if (abs(distFiltrada - ultimaLecturaEstable) > HISTERESIS_CM) { // Si el cambio es real
      ultimaLecturaEstable = distFiltrada; // Acepta el nuevo valor
    }
  }

  int distEntera = ultimaLecturaEstable; // Distancia final usada en decisiones
  float distancia = (float)distEntera;   // Versión float para cálculos


  //  ****************************CÁLCULO DE ALTURA Y PORCENTAJE CONTENEDOR DE
COMIDA****************************
  float alturaComida = ALTURA_TOTAL_COMEDERO_CM - distancia; // Convierte distancia a “altura”
  if (alturaComida < 0) alturaComida = 0;                    // Evita negativos
 float alturaMax = ALTURA_TOTAL_COMEDERO_CM - ALTURA_LLENO_CM; // Altura útil máxima
  if (alturaComida > alturaMax) alturaComida = alturaMax;       // Limita al máximo
  int alturaEntera = (int)roundf(alturaComida);  // Redondea altura a entero
  float porcentajeF = (alturaComida / alturaMax) * 100.0f; // Calcula porcentaje
  int porcentaje = (int)roundf(porcentajeF);               // Porcentaje entero
  String nuevoEstado; // Variable donde se guarda el estado textual
  if (distEntera <= 12) {      // Si la distancia es pequeña, está lleno
    nuevoEstado = "lleno";    // Estado lleno
    porcentaje = 100;        // Fuerza 100%
  }
  else if (distEntera <= 22) nuevoEstado = "medio";   // Rango medio
  else if (distEntera <= 31) nuevoEstado = "bajo";    // Rango bajo
  else if (distEntera <= 39) nuevoEstado = "critico"; // Rango crítico
  else {                                              // Si supera 39 cm
    nuevoEstado = "vacio";      // Estado vacío
    porcentaje = 0;           // 0%
```

```
    alturaEntera = 0;           // Nivel 0 cm
  }
  Serial.printf("[COMEDERO] dist=%d cm | altura=%d cm | %d %% | estado=%s\n",
                distEntera, alturaEntera, porcentaje, nuevoEstado.c_str()); // Debug en Serial

  //**************************** ENVÍO A FIREBASE ****************************
  if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED) { // Verifica condiciones
    Firebase.RTDB.setInt(&fbdo, "/esp32_1/nivel_comida_cm", distEntera);      // Guarda distancia
    Firebase.RTDB.setInt(&fbdo, "/esp32_1/nivel_comida_real_cm", alturaEntera); // Guarda nivel real
  }

  // ********************************NOTIFICACIÓN ESTDADO LLENO*********************************
  if (primeraLecturaComedero) {           // Solo la primera vez
    ultimaDistanciaComedero = distEntera; // Inicializa distancia anterior
    primeraLecturaComedero  = false;      // Marca que ya pasó la primera lectura
  }

  if (distEntera > UMBRAL_LLENO_CM + REARME_LLENO_DELTA) { // Si se alejó del lleno
    comederoLlenoNotificado = false;                        // Rearma notificación
  }

  if (!comederoLlenoNotificado &&                          // Si aún no notificó
      ultimaDistanciaComedero > UMBRAL_LLENO_CM &&         // Antes estaba “no lleno”
      distEntera <= UMBRAL_LLENO_CM) {                     // Ahora entró a “lleno”

    String msg = "✅🍽️ Comedero LLENO (100%) · Nivel: " + String(alturaEntera) + " cm"; // Mensaje
    if (enviarMensajeTelegram(msg.c_str())) {              // Si se envió bien
      comederoLlenoNotificado = true;                      // Marca como notificado
    } else {
      Serial.println("[Telegram] LLENO no enviado, no marco notificado"); // Log de fallo
    }
  }

  ultimaDistanciaComedero = distEntera; // Actualiza distancia anterior


  // ************************NOTIFICACIONES A TELEGRAM CUANDO ESTADO SEA VACÍO Y
CRÍTICO*****************************
  if (nuevoEstado == "vacio") { // Si está vacío
    if (lastVacioMsgTime == 0 || (ahora - lastVacioMsgTime) >= INTERVALO_VACIO_MS) { // Control de
tiempo
      String msg; // Mensaje a enviar
      if (estadoNotificado != "vacio") { // Si recién cambió a vacío
        msg = "⛔🍽️ Comedero VACÍO · Nivel: 0 cm · Sin alimento disponible. Recargar con urgencia.";
      } else { // Si ya estaba vacío (recordatorio)
        msg = "⛔🍽️ Recordatorio: Comedero continúa VACÍO · Sin alimento disponible.";
      }
      if (enviarMensajeTelegram(msg.c_str())) { // Si se envió bien
        lastVacioMsgTime = ahora;               // Actualiza timer
      } else {
        Serial.println("[Telegram] VACIO no enviado, no actualizo timer"); // Log de error
      }
    }
  }
  if (nuevoEstado == "critico") { // Si está crítico
    if (lastCriticoMsgTime == 0 || (ahora - lastCriticoMsgTime) >= INTERVALO_CRITICO_MS) { // Control
de tiempo
      String msg; // Mensaje a enviar
      if (estadoNotificado != "critico") { // Cambio reciente a crítico
        msg = "❗🍽️ Comedero en nivel CRÍTICO (" + String(porcentaje) +
              "%) · Nivel: " + String(alturaEntera) + " cm · Recargar lo antes posible.";
      } else { // Recordatorio si sigue crítico
        msg = "❗🍽️ Recordatorio: Comedero continúa en nivel CRÍTICO (" +
              String(porcentaje) + "%) · Nivel: " + String(alturaEntera) + " cm.";
      }
      if (enviarMensajeTelegram(msg.c_str())) { // Si se envió
        lastCriticoMsgTime = ahora;             // Actualiza timer
      } else {
        Serial.println("[Telegram] CRITICO no enviado, no actualizo timer");
      }
    }
  }
  if (nuevoEstado == "medio" || nuevoEstado == "lleno") { // Estados normales
```

```
    lastVacioMsgTime = 0;                                  // Reinicia timer de vacío
    lastCriticoMsgTime = 0;                                // Reinicia timer de crítico
  }
  estadoNotificado = nuevoEstado; // Guarda el estado actual como último notificado

  // ************************ LÓGICA BUZZER *********************************
  if (nuevoEstado == "lleno" && buzzerArmado && !buzzerActivo) { // Si está lleno y aún no sonó
    digitalWrite(BUZZER_PIN, HIGH);   // Enciende buzzer
    buzzerActivo   = true;            // Marca activo
    buzzerInicioMs = millis();        // Guarda inicio
    buzzerArmado   = false;           // Desarma para que no repita
    Serial.println("[BUZZER] Comedero LLENO -> buzzer ON 5s");
  }
  if (buzzerActivo && (millis() - buzzerInicioMs >= BUZZER_DURACION_MS)) { // Si ya cumplió 5s
    digitalWrite(BUZZER_PIN, LOW); // Apaga buzzer
    buzzerActivo = false;          // Marca inactivo
    Serial.println("[BUZZER] Tiempo cumplido -> buzzer OFF");
  }

  if (nuevoEstado != "lleno") { // Si NO está lleno
    if (!enEstadoNoLleno) {     // Si recién entra a “no lleno”
      enEstadoNoLleno = true;   // Marca estado “no lleno”
      inicioNoLlenoMs = millis(); // Guarda inicio del conteo
    } else {                    // Si ya estaba en “no lleno”
      if (!buzzerArmado && (millis() - inicioNoLlenoMs >= TIEMPO_MIN_NO_LLENO_MS)) { // Si pasó 30s
        buzzerArmado = true;    // Rearma el buzzer
        Serial.println("[BUZZER] Rearmado tras tiempo en nivel no lleno");
      }
    }
  } else { // Si volvió a lleno
    enEstadoNoLleno = false; // Reinicia seguimiento del estado no lleno
  }

  // ************************ HORARIO DE DISPENSACIÓN ****************************
  struct tm timeinfo;                 // Estructura para hora local
  if (getLocalTime(&timeinfo)) {      // Si hay hora válida
    int hora = timeinfo.tm_hour;      // Hora actual
    int minuto = tieinfo.tm_min;     // Minuto actual
    int segundo = tmeinfo.tm_sec;    // Segundo actual

    bool esHoraProgramada =
      ((hora == 8) || (hora == 12) || (hora == 16)); // Horas programadas

    if (esHoraProgramada && minuto == 0 && segundo == 0 && !ejecutado) { // Justo a la hora exacta
      dispensar();              // Ejecuta la dispensación
      ejecutado = true;         // Marca para no repetir en ese segundo
    }

    if ((!esHoraProgramada || minuto != 0 || segundo != 0) && ejecutado) { // Cuando ya pasó el
instante exacto
      ejecutado = false;        // Rearma para el siguiente horario
    }
  }

  // ****************************** REINTENTOS WIFI / FIREBASE ***************************
  reintentarWiFiNoBloqueante(ahora); // Reintento WiFi cada 30s si hace falta
  if (!firebaseReady && WiFi.status() == WL_CONNECTED) { // Si Firebase no está listo pero hay WiFi
    reintentarFirebaseNoBloqueante(ahora);               // Reintento Firebase cada 60s
  }
}
```

Anexo D. Código subsistema de bebedero

```
//TEMA:Código para el dispensador de agua
//AUTOR: Edison Quilca
// **********************LIBRERÍAS***********************************
#include <Arduino.h>                      // Funciones base de Arduino
#include <WiFi.h>                         // Conexión WiFi del ESP32
#include <WiFiClientSecure.h>             // Cliente HTTPS para telegram
#include <Firebase_ESP_Client.h>          // Librería para Firebase Realtime Database
#include <time.h>                         // Manejo de hora con NTP

// **********************CREDENCIALES WiFi***********************************
#define WIFI_SSID     "INNO-FIBER- FLIA PUPIALES"             // Nombre de la red WiFi
```

```
#define WIFI_PASSWORD "1704Pupiales#"                          // Contraseña del WiFi

// **********************CREDENCIALES FIREBASE**********************************
#define API_KEY       "AIzaSyBD4sciHu5M9STZSm2mHy0Kh4"// API Key del proyecto Firebase
#define DATABASE_URL  "https://criaderoj-default-rtdb.firebaseio.com/" // URL de RTDB


// ********************** PINES DE CONEXIÓN ***************************
// Sensor 1 (TANQUE)
#define TRIG1_PIN 14                      // TRIG del ultrasónico del tanque
#define ECHO1_PIN 35                      // ECHO del ultrasónico del tanque
// Sensor 2 (BEBEDERO)
#define TRIG2_PIN 25                      // TRIG del ultrasónico del bebedero
#define ECHO2_PIN 33                      // ECHO bebedero

#define BUZZER_PIN        13              // Pin del buzzer
#define BOMBA_1_PIN       27              // Bomba 1: tanque  hacia el bebedero
#define BOMBA_2_PIN       32              // Bomba 2: drenaje/limpieza
#define SENSOR_HUEVOS_PIN 34              // Sensor infrarrojo para contar huevos

// **********************OBJETOS Y BANDERAS DE FIREBASE************************
FirebaseData fbdo;                         // Objeto para enviar/recibir datos en Firebase
FirebaseAuth auth;                         // Autenticación (signup anónimo)
FirebaseConfig config;                     // Configuración general de Firebase
bool signupOK      = false;                // Indica si el signup fue correcto
bool firebaseReady = false;                // Indica si Firebase está listo para operar


// **********************VARIABLES GLOBALES PRINCIPALES*****************************
int   contadorHuevos = 0;                  // Conteo actual de huevos (se reinicia al llegar a 60)
float emaTanque      = NAN;                // EMA para suavizar lectura del tanque
float emaBebedero    = NAN;                // EMA para suavizar lectura del bebedero

// **********************PRODUCCIÓN DIARIA*************************************
int huevosDia = 0;                         // Huevos acumulados en el día actual
int diaActualProduccion = -1;              // Sirve para detectar el cambio de día

// ****************alerta 55 huevos para telegram) -----
bool aviso55Enviado = false;      // evita repetir el mensaje cuando llega a 55

// **********************UMBRALES Y ALTURAS******************
// **********************TANQUE*****************************
const int TANQUE_BUZZER_CM        = 10;    // <=10 cm: Estado lleno se activa buzzer
const int TANQUE_REINICIO_CM      = 10;    // Si vuelve a este valor o menos, se desbloquea el sistema
const int TANQUE_LLENO_MAX_CM     = 15;
const int TANQUE_MEDIO_MIN_CM     = 16;
const int TANQUE_MEDIO_MAX_CM     = 20;
const int TANQUE_BAJO_MIN_CM      = 21;
const int TANQUE_BAJO_MAX_CM      = 25;
const int TANQUE_CRITICO_MIN_CM   = 26;
const int TANQUE_CRITICO_MAX_CM   = 30;
const int TANQUE_PARADA_UMBRAL_CM = 30;    // >30 cm: se asume vacío y se bloquean bombas

// ******************Alturas desde el sensor hasta el fondo*****************************************
const int ALTURA_TOTAL_TANQUE   = 32;      // Altura real del tanque en cm
const int ALTURA_TOTAL_BEBEDERO = 7;       // Altura real del bebedero en cm

//************************ UMBRALES BEBEDERO **************************
const int FILL_START_CM = 6;               // Si distancia >=6: iniciar recarga
const int FILL_STOP_CM  = 4;               // Si distancia <=4: detener recarga (lleno)
const int BEBEDERO_VACIO_CM = FILL_START_CM;// Punto donde se considera 0% (vacío)

const unsigned long MAX_FILL_MS  = 2UL * 60UL * 60UL * 1000UL; // Tope de tiempo llenando
const unsigned long MAX_DRAIN_MS = 90UL * 1000UL;              // Tope de tiempo drenando


// // **********************MÁQUINA DE ESTADOS (BEBEDERO)************************
enum class Estado { IDLE, FILL_FROM_TANK, DRAIN_CYCLE, REFILL_AFTER_DRAIN }; // Estados del proceso
Estado estado = Estado::IDLE;               // Estado inicial: reposo
unsigned long estadoInicioMs = 0;           // Marca de tiempo del inicio del estado


// // **********************BUZZER Y AVISOS (TANQUE)************************************
```

```
bool bloqueoTanque           = false;       // Si el tanque está vacío: se bloquea el sistema
bool buzzerEnCurso           = false;       // Indica si el buzzer está activo ahora
unsigned long buzzerInicioMs = 0;           // Momento en que empezó a sonar el buzzer
bool primeraLecturaTanque    = true;        // Evita comparar sin dato previo
int  ultimoNivelTanque       = 0;           // Última distancia del tanque registrada

bool avisoBajoEnviado    = false;           // Bandera de aviso bajo (en este código casi no se usa)
bool avisoCriticoEnviado = false;           // Bandera para no repetir alerta crítica inmediatamente

unsigned long ultimoCriticoMs = 0;          // Guarda cuándo se envió el último aviso crítico
const unsigned long RECORDATORIO_CRITICO_MS = 3UL * 60UL * 60UL * 1000UL; // Recordatorio cada 3h

// **********************LIMPIEZA PROGRAMADA*********************
int  ultimoDiaRevisado = -1;                // Permite reiniciar banderas cuando cambia el día
bool drenajeLanzadoHoy = false;             // Evita repetir limpieza en el mismo día

unsigned long ultimoIntentoWiFiMs = 0;      // Control de reintento de WiFi
unsigned long ultimoIntentoFbMs   = 0;      // Control de reintento de Firebase
unsigned long ultimoOkFirebaseMs  = 0;      // Marca el último instante con Firebase “ok”
unsigned long ultimoEnvioNivelMs  = 0;      // Control para subir niveles cada cierto tiempo

volatile bool          huevoEdgeFlag = false; // Bandera ISR: hubo un pulso válido
volatile unsigned long huevoEdgeMs   = 0;     // Momento exacto del pulso detectado

const unsigned long MIN_INTERVAL_CON_BOMBA = 120; // Mínimo entre pulsos cuando bombas ON
unsigned long ultimoHuevoValidoMs = 0;            // Último pulso aceptado como huevo real

bool bomba1Estado = false;                   // Estado actual de bomba 1
bool bomba2Estado = false;                  // Estado actual de bomba 2
unsigned long ultimoCambioBombasMs = 0;       // Marca cambios para filtrar ruido

bool huevosDirty           = false;          // Indica que falta sincronizar conteo
bool ultimo60Dirty         = false;           // Indica que falta sincronizar evento “60 huevos”
char ultimo60Timestamp[32] = "";              // Guarda timestamp del evento “60 huevos”

 // **********************TELEGRAM: COLA ORDENADA DE MENSAJES**********************************

#define MAX_COLA 10                          // Tamaño máximo de la cola
struct MensajeTelegram {                    // Estructura del mensaje en cola
  String texto;                             // Texto codificado para URL
};
MensajeTelegram colaMensajes[MAX_COLA];     // Cola circular
int inicioCola = 0, finCola = 0;            // Punteros de la cola
unsigned long ultimoEnvioTelegram = 0;       // Control del último envío
const unsigned long INTERVALO_TELEGRAM_MS = 2000; // 2 segundos entre mensajes
bool wifiEstabaConectado = false;            // Detecta caídas de WiFi

// **********************DEPURACIÓN (DESACTIVADA)***************************

static unsigned long ultimoChequeoDepuracionMs = 0;

 // ********************** FUNCIONES DE PORCENTAJE Y MENSAJES DEL TANQUE*****************************

int porcentajeTanque(int dCm) {                          // Convierte distancia del tanque a
porcentaje
  if (dCm <= 0) return -1;                               // Lectura inválida
  if (dCm <= TANQUE_BUZZER_CM)        return 100;        // Muy cerca del sensor =lleno
  if (dCm >= TANQUE_PARADA_UMBRAL_CM) return 0;          // Muy lejos=vacío

  float pct = 100.0f * (float)(TANQUE_PARADA_UMBRAL_CM - dCm) /  // Interpolación lineal
              (float)(TANQUE_PARADA_UMBRAL_CM - TANQUE_BUZZER_CM); // Rango 10..30 cm
  if (pct < 0)   pct = 0;                                // Limita inferior
  if (pct > 100) pct = 100;                              // Limita superior
  return (int)roundf(pct);                               // Redondea a entero
}

int porcentajeBebedero(int dCm) {                        // Convierte distancia del bebedero a
porcentaje
  if (dCm <= 0) return -1;                               // Lectura inválida
  if (dCm <= FILL_STOP_CM)       return 100;             // Lleno
  if (dCm >= BEBEDERO_VACIO_CM)  return 0;               // Vacío

  float pct = 100.0f * (float)(BEBEDERO_VACIO_CM - dCm) / // Interpolación 6..4 cm
              (float)(BEBEDERO_VACIO_CM - FILL_STOP_CM);  // Normalización del rango
```

```
  if (pct < 0)   pct = 0;                                   // Limita inferior
  if (pct > 100) pct = 100;                                 // Limita superior
  return (int)roundf(pct);                                  // Redondeo final
}

String mensajeTanque(int dCm) {                             // Construye mensaje corto según estado del
tanque
  int pct = porcentajeTanque(dCm);                          // Obtiene el porcentaje calculado
  if (pct < 0) pct = 0;                                     // Si hubo error, se asume 0
  int nivelReal = ALTURA_TOTAL_TANQUE - dCm;                // Convierte distancia a "altura de agua"
  if (nivelReal < 0) nivelReal = 0;                         // Evita negativos por lecturas raras

  if (pct >= 95) {                                          // Si está prácticamente lleno
    return "✅💧Tanque LLENO (100%) - " + String(nivelReal) + " cm agua"; // Mensaje de confirmación
  }

  if (pct <= 10) {                                          // Si está casi vacío
    return "⚠️💧Tanque VACÍO, el sistema se bloqueará hasta que recargue el tanque (" +
           String(pct) + "%) - " + String(nivelReal) + " cm agua";       // Mensaje de advertencia
  }

  return "";                                                // En estados intermedios no se envía aviso
}

// ********************** PROTOTIPOS (DECLARACIONES)*********************************
String urlencode(String str);                               // Convierte texto a formato URL
void   conectarWiFiInicial();                               // Conexión inicial WiFi (con timeout)
void   reintentarWiFiNoBloqueante(unsigned long now);       // Reintento WiFi sin detener el sistema
void   iniciarFirebase();                                   // Inicio de Firebase
void   reintentarFirebaseNoBlqueante(unsigned long now); // Reintento Firebase sin detener
void   enviarTelegram(const String &mensaje);               // Encola mensajes de Telegram
int    medirEstableInt(uint8_t trigPin, uint8_t echoPin, float &emaRef); // Lectura ultrasónica
estable
void   gestionarProgramacionDrenaje();                      // Limpieza programada
void   bomba1On(bool on);                                   // Control bomba 1
void   bomba2On(bool on);                                   // Control bomba 2
void   intentarSyncPendiente(unsigned long now);            // Sube pendientes a Firebase

// **********************ISR: INTERRUPCIÓN PARA CONTEO DE HUEVOS*************************
void IRAM_ATTR huevoISR() {                                 // ISR: se ejecuta apenas cae el pulso
(FALLING)
  unsigned long ahora = millis();                           // Guarda el tiempo exacto del evento
  static unsigned long ultimoISR = 0;                       // Ayuda a filtrar rebotes mínimos
  if (ahora - ultimoISR < 1) return;                        // Anti-rebote rápido (~1ms)
  ultimoISR = ahora;                                        // Actualiza el último pulso

  if (ahora - ultimoCambioBombasMs < IGNORE_AFTER_BOMBA_MS) return; // Ignora ruido tras
prender/apagar bombas

  int lows = 0;                                             // Cuenta cuántas lecturas LOW se mantienen
  for (int i = 0; i < 3; i++) {                              // Repite 3 veces para confirmar
estabilidad
    if (digitalRead(SENSOR_HUEVOS_PIN) == LOW) lows++;      // Si sigue LOW, suma
    delayMicroseconds(10);                                  // Micro-espera para validar que no fue
pico
  }
  if (lows < 2) return;                                     // Si no se mantuvo LOW, se descarta como
ruido

  huevoEdgeMs   = ahora;                                    // Guarda el instante del pulso válido
  huevoEdgeFlag = true;                                     // Marca bandera para que loop lo procese
}
// ************************************FUNCIONES DE WIFI**********************************
void conectarWiFiInicial() {                                // Conecta con un límite de tiempo
  WiFi.mode(WIFI_STA);                                      // Modo estación (cliente)
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);                      // Inicia conexión con credenciales
  Serial.print("[WiFi] Conectando");                        // Muestra estado

  unsigned long inicio = millis();                          // Marca inicio
  while (WiFi.status() != WL_CONNECTED && millis() - inicio < 15000) { // Espera máximo 15s
    delay(500);                                             // Pausa para no saturar CPU
    Serial.print(".");                                      // Indicador visual
  }
  Serial.println();                                         // Salto de línea
```

```
  if (WiFi.status() == WL_CONNECTED) {                    // Si conectó
    Serial.print("[WiFi] Conectado. IP: ");                // Mensaje ok
    Serial.println(WiFi.localIP());                       // Imprime IP
  } else {
    Serial.println("[WiFi] No se pudo conectar en 15s. Sigo OFFLINE (solo local)."); // Modo offline
  }
}

void reintentarWiFiNoBloqueante(unsigned long now) {      // Reintenta sin detener el sistema
  if (WiFi.status() == WL_CONNECTED) return;              // Si ya está conectado, no hace nada
  if (now - ultimoIntentoWiFiMs < 30000) return;          // Reintenta cada 30 segundos

  Serial.println("[WiFi] Desconectado, reintentando...");  // Log
  WiFi.disconnect();                                      // Limpia estado previo
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);                   // Intenta de nuevo
  ultimoIntentoWiFiMs = now;                              // Guarda el momento del intento
}

//*********************************FUNCIONES: FIREBASE*********************************
void iniciarFirebase() {                                  // Inicializa Firebase si hay WiFi
  if (WiFi.status() != WL_CONNECTED) {                    // Si no hay WiFi, se evita iniciar
    Serial.println("[Firebase] No inicio: no hay WiFi.");  // Log
    firebaseReady = false;                                // Bandera en falso
    return;                                               // Sale
  }

  Serial.println("[Firebase] Inicializando...");          // Log
  config.api_key      = API_KEY;                          // Configura API key
  config.database_url = DATABASE_URL;                     // Configura URL de RTDB

  if (Firebase.signUp(&config, &auth, "", "")) {          // Signup anónimo
    signupOK = true;                                      // Bandera ok
    Serial.println("[Firebase] SignUp OK");               // Log
  } else {
    Serial.println(config.signer.signupError.message.c_str());// Error real
    return;                                               // Sale para no continuar
  }

  Firebase.begin(&config, &auth);                         // Inicia Firebase
  Firebase.reconnectWiFi(true);                           // Permite reconexión automática
  firebaseReady       = true;                             // Marca listo
  ultimoOkFirebaseMs  = millis();                         // Guarda instante ok
  Serial.println("[Firebase] Listo (modo mejor esfuerzo).");// Log final
}

void reintentarFirebaseNoBloqueante(unsigned long now) {  // Reintenta Firebase cada cierto tiempo
  if (firebaseReady) return;                              // Si ya está listo, no reintenta
  if (now - ultimoIntentoFbMs < 60000) return;            // Reintenta cada 60 segundos

  Serial.println("[Firebase] Reintentando iniciar...");   // Log
  iniciarFirebase();                                      // Intenta inicializar de nuevo
  ultimoIntentoFbMs = now;                                // Guarda el intento
}


// ********************** FUNCIONES PARA TELEGRAM*************************************

void agregarMensajeTelegram(const String &mensajeCodificado) { // Encola un mensaje codificado
  int siguiente = (finCola + 1) % MAX_COLA;                    // Calcula siguiente posición circular
  if (siguiente == inicioCola) {                               // Si la cola está llena
    Serial.println("[Telegram] Cola llena, mensaje descartado (para mantener orden)."); // Aviso
    return;                                                    // Sale sin guardar
  }
  colaMensajes[finCola].texto = mensajeCodificado;             // Guarda en la cola
  finCola = siguiente;                                         // Avanza fin
}

void procesarColaTelegram() {                                  // Envía mensajes respetando tiempos
  if (inicioCola == finCola) return;                           // Si cola vacía, sale
  if (millis() - ultimoEnvioTelegram < INTERVALO_TELEGRAM_MS) return; // Respeta 2s
  if (WiFi.status() != WL_CONNECTED) return;                   // Sin WiFi, no envía

  WiFiClientSecure client;                                     // Cliente HTTPS
```

```
  client.setInsecure();                                          // Evita validación de certificado
  if (client.connect("api.telegram.org", 443)) {                 // Conexión al servidor
    String url = "https://api.telegram.org/00737:AAGn-
SIVTGkDmIAVesbpZxg1ug/sendMessage?chat_id=5585325975&text=" +
                 colaMensajes[inicioCola].texto;                 // URL completa con mensaje

    client.print("GET " + url + " HTTP/1.1\r\nHost: api.telegram.org\r\nConnection: close\r\n\r\n");
// Envía GET
    delay(100);                                                  // Espera corta para completar envío
    client.stop();                                               // Cierra conexión

    inicioCola = (inicioCola + 1) % MAX_COLA;                    // Saca el mensaje enviado
    ultimoEnvioTelegram = millis();                              // Actualiza momento del envío
  } else {
    Serial.println("[Telegram] No se pudo conectar a api.telegram.org, se reintentará."); // Queda en
cola
  }
}

void enviarTelegram(const String &mensaje) {                    // Encola un mensaje (si hay internet)
  if (WiFi.status() != WL_CONNECTED) {                           // Si no hay WiFi
    Serial.println("[Telegram] Mensaje descartado (sin WiFi): " + mensaje); // Se descarta
    return;                                                      // Sale
  }
  if (mensaje.length() == 0) return;                             // Si está vacío, no envía

  String cod = urlencode(mensaje);                               // Codifica el mensaje para URL
  agregarMensajeTelegram(cod);                                   // Lo agrega a la cola ordenada
}

void manejarCambioWifi() {                                       // Limpia cola si se cae el WiFi
  bool conectadoAhora = (WiFi.status() == WL_CONNECTED);         // Estado actual
  if (!conectadoAhora && wifiEstabaConectado) {                  // Si antes había WiFi y ahora no
    inicioCola = finCola;                                        // Vacía cola para no enviar mensajes
viejos
    Serial.println("[WiFi] Desconectado, cola de Telegram vaciada."); // Log
  }
  wifiEstabaConectado = conectadoAhora;                          // Actualiza estado anterior
}

//****************************** FUNCIONES PARA BOMBAS*************************************
void bomba1On(bool on) {                                         // Control de bomba 1 (recarga)
  if (bomba1Estado != on) {                                      // Solo si cambia realmente
    ultimoCambioBombasMs = millis();                             // Marca cambio para filtrar ruido
    bomba1Estado = on;                                           // Guarda el nuevo estado

    if (on) enviarTelegram("🔄💧Iniciando recarga del bebedero"); // Aviso de encendido
    else   enviarTelegram("✅💧Bebedero recargado satisfactoriamente"); // Aviso de apagado
  }
  digitalWrite(BOMBA_1_PIN, on ? LOW : HIGH);                    // Relé activo en LOW
}

void bomba2On(bool on) {                                         // Control de bomba 2 (drenaje)
  if (bomba2Estado != on) {                                      // Solo si cambió
    ultimoCambioBombasMs = millis();                             // Marca cambio para ventana ciega
    bomba2Estado = on;                                           // Guarda estado
  }
  digitalWrite(BOMBA_2_PIN, on ? LOW : HIGH);                    // Relé activo en LOW
}

//************************FUNCIONES SENSORES ULTRASÓNICOS*****************************
static float quickSelect(float* arr, int n, int k) {             // Selección de k-ésimo (mediana) sin
ordenar
  int left = 0, right = n - 1;                                   // Límites
  while (true) {                                                 // Repite hasta encontrar k
    if (left == right) return arr[left];                         // Caso base
    float pivot = arr[(left + right) / 2];                       // Pivote
    int i = left, j = right;                                     // Índices
    while (i <= j) {                                             // Partición
      if (i <= j) {                                              // Intercambio
        float t = arr[i]; arr[i] = arr[j]; arr[j] = t;           // Swap
        i++; j--;                                                // Mueve índices
      }
    }
```

```
    if (k <= j)      right = j;                                  // Busca a la izquierda
    else if (k >= i) left = i;                                   // Busca a la derecha
    else             return arr[k];                               // Ya quedó ubicado
  }
}

float leerDistanciaCrudaCM(uint8_t trigPin, uint8_t echoPin) { // Lectura directa del ultrasónico
  digitalWrite(trigPin, LOW);                                   // Pulso limpio
  delayMicroseconds(3);                                         // Estabiliza
  digitalWrite(trigPin, HIGH);                                  // Disparo
  delayMicroseconds(10);                                        // Duración del TRIG
  digitalWrite(trigPin, LOW);                                   // Fin

  long duracion = pulseIn(echoPin, HIGH, 25000);                // Tiempo del eco con timeout
  if (duracion <= 0) return 0.0f;                               // Si no midió, devuelve 0

  float d = duracion * 0.0343f / 2.0f;                          // Conversión a cm
  if (d < 1.0f || d > 400.0f) return 0.0f;                      // Filtra valores imposibles
  return d;                                                     // Devuelve distancia válida
}

int medirEstableInt(uint8_t trigPin, uint8_t echoPin, float &emaRef) { // Medición estable
  const int N = 10;                                             // Número de muestras rápidas
  float vals[N]; int cnt = 0;                                   // Arreglo de muestras válidas

  for (int i = 0; i < N; i++) {                                 // Toma N lecturas
    float v = leerDistanciaCrudaCM(trigPin, echoPin);           // Lectura cruda
    if (v > 0) vals[cnt++] = v;                                 // Guarda si fue válida
    delay(2);                                                   // Pequeña pausa
  }

  if (cnt == 0) {                                               // Si no hubo lecturas válidas
    if (isnan(emaRef)) return 0;                                // Sin referencia, devuelve 0
    return (int)roundf(emaRef);                                 // Con referencia, reutiliza EMA
anterior
  }

  float tmp[10];                                                // Copia para calcular mediana
  for (int i = 0; i < cnt; i++) tmp[i] = vals[i];               // Copia valores
  float med = quickSelect(tmp, cnt, cnt/2);                     // Mediana para robustez

  float suma = 0; int ok = 0;                                   // Para promedio recortado
  float lo = med * 0.90f, hi = med * 1.10f;                     // Acepta ±10% alrededor de mediana
  for (int i = 0; i < cnt; i++) {                               // Recorre lecturas válidas
    if (vals[i] >= lo && vals[i] <= hi) {                       // Si cae en rango confiable
      suma += vals[i]; ok++;                                    // Acumula
    }
  }
  float trimmed = (ok > 0) ? (suma / ok) : med;                 // Promedio recortado o mediana

  if (!isnan(emaRef)) {                                         // Si ya había EMA previa
    float maxStep = 3.0f;                                       // Limita cambios por ciclo
    if (trimmed > emaRef + maxStep) trimmed = emaRef + maxStep;// Recorta subidas bruscas
    else if (trimmed < emaRef - maxStep) trimmed = emaRef - maxStep; // Recorta bajadas bruscas
  }

  const float alpha = 0.20f;                                    // Peso del valor nuevo (20%)
  if (isnan(emaRef)) emaRef = trimmed;                          // Si es primera vez, inicia EMA
  else emaRef = alpha + (1.0f - alpha) * emaRef;     // EMA para suavizar lectura

  return (int)roundf(emaRef);                                   // Retorna lectura estable entera
}

//********************** FUNCIONES PARA BUZZER************************
void iniciarBuzzer5s(unsigned long now) {                       // Activa buzzer por 5s
  buzzerEnCurso  = true;                                        // Marca que está activo
  buzzerInicioMs = now;                                         // Guarda inicio
  digitalWrite(BUZZER_PIN, HIGH);                               // Enciende buzzer
}

void actualizarBuzzer(unsigned long now) {                      // Apaga buzzer cuando se cumple el
tiempo
  if (buzzerEnCurso && (now - buzzerInicioMs >= 5000UL)) {      // Si ya pasaron 5s
    buzzerEnCurso = false;                                      // Desactiva bandera
```

```
    digitalWrite(BUZZER_PIN, LOW);                                  // Apaga buzzer
  }
}
//*****************************FUNCIONES PARA LIMPIEZA PROGRAMADA (MIÉRCOLES 05:00) *********
void gestionarProgramacionDrenaje() {                               // Lanza limpieza semanal del bebedero
  if (bloqueoTanque) return;                                        // Si tanque vacío, no se limpia

  struct tm tinfo;                                                  // Estructura de fecha/hora
  if (!getLocalTime(&tinfo)) return;                                // Sin NTP, no programa nada

  int dia  = tinfo.tm_mday;                                         // Día del mes
  int wday = tinfo.tm_wday;                                         // Día de semana (0 domingo)

  if (dia != ultimoDiaRevisado) {                                   // Si cambió el día
    ultimoDiaRevisado = dia;                                        // Actualiza referencia
    drenajeLanzadoHoy = false;                                      // Permite ejecutar si corresponde
  }

  bool esDiaLimpieza = (wday == 3);                                  // Miércoles
  if (!esDiaLimpieza) return;                                        // Si no es miércoles, sale


    estado = Estado::DRAIN_CYCLE;                                    // Entra a estado de drenaje
    estadoInicioMs = millis();                                       // Marca inicio
    bomba2On(true);                                                 // Enciende bomba 2

    drenajeLanzadoHoy = true;                                       // Evita repetir en el mismo día

    Serial.println("[LIMPIEZA] Inicio limpieza miércoles 05:00 (Bomba 2 ON)"); // Log
    enviarTelegram("🔄💧🦠Inicio de limpieza del bebedero");     // Aviso Telegram
  }
}

// *********************************SINCRONIZACIÓN PENDIENTE*****************************
void intentarSyncPendiente(unsigned long now) {                     // Sube datos pendientes cuando vuelve
internet
  if (!firebaseReady || !Firebase.ready() || WiFi.status() != WL_CONNECTED) return; // Solo si está
disponible

  if (huevosDirty) {                                                // Si conteo quedó pendiente
    if (Firebase.RTDB.setInt(&fbdo, "/esp32_2/conteo_huevos", contadorHuevos)) { // Intenta subirlo
      huevosDirty        = false;                                   // Ya quedó sincronizado
      ultimoOkFirebaseMs = now;                                     // Actualiza último OK
      Serial.println("[SYNC] conteo_huevos sincronizado tras reconexión."); // Log
    } else {
      Serial.print("[SYNC] Error subiendo conteo_huevos: ");        // Log
      Serial.println(fbdo.errorReason());                           // Motivo del error
    }
  }

  if (ultimo60Dirty && strlen(ultimo60Timestamp) > 0) {             // Si evento “60 huevos” quedó
pendiente
    FirebaseJson json;                                               // JSON del evento
    json.set("timestamp", ultimo60Timestamp);                        // Fecha/hora
    json.set("cantidad", 60);                                        // Cantidad fija

    if (Firebase.RTDB.setJSON(&fbdo, "/esp32_2/ultima_vez_60_huevos", &json)) { // Sube JSON
      ultimo60Dirty      = false;                                    // Ya no queda pendiente
      ultimoOkFirebaseMs = now;                                      // Actualiza último OK
      Serial.println("[SYNC] ultima_vez_60_huevos sincronizado tras reconexión."); // Log
    } else {
      Serial.print("[SYNC] Error subiendo ultima_vez_60_huevos: "); // Log
      Serial.println(fbdo.errorReason());                            // Motivo
    }
  }
}

void verificarDepuracionTrimestralFija(unsigned long now) {      // Función anulada
  return;                                                           // No ejecuta nada
}

//******************** INICIALIZACIÓN GENERAL SETUP*************************************
void setup() {
  Serial.begin(115200);                                             // Inicia monitor serial
```

```
  pinMode(TRIG1_PIN, OUTPUT);                                   // TRIG tanque
  pinMode(ECHO1_PIN, INPUT);                                    // ECHO tanque
  pinMode(TRIG2_PIN, OUTPUT);                                   // TRIG bebedero
  pinMode(ECHO2_PIN, INPUT);                                    // ECHO bebedero

  pinMode(BUZZER_PIN, OUTPUT);                                  // Buzzer salida
  pinMode(BOMBA_1_PIN, OUTPUT);                                 // Bomba 1 salida
  pinMode(BOMBA_2_PIN, OUTPUT);                                 // Bomba 2 salida

  pinMode(SENSOR_HUEVOS_PIN, INPUT);                            // Sensor huevos entrada

  digitalWrite(BUZZER_PIN, LOW);                                // Buzzer apagado
  bomba1On(false);                                              // Bomba 1 apagada
  bomba2On(false);                                              // Bomba 2 apagada

  contadorHuevos        = 0;                                    // Reinicia conteo
  huevosDia             = 0;                                    // Reinicia producción diaria
  diaActualProduccion   = -1;                                   // Fuerza reinicio de fecha
  huevosDirty           = true;                                 // Pendiente de sincronizar
  aviso55Enviado = false;

 ultimoHuevoValidoMs = 0;                                      // Reinicia referencia

  Serial.println("[INFO] Sistema iniciado. Conteo de huevos = 0."); // Log

  attachInterrupt(digitalPinToInterrupt(SENSOR_HUEVOS_PIN), huevoISR, FALLING); // ISR por flanco de
bajada

  conectarWiFiInicial();                                        // Conexión inicial WiFi
  wifiEstabaConectado = (WiFi.status() == WL_CONNECTED);        // Guarda estado inicial

  iniciarFirebase();                                            // Inicializa Firebase

  configTime(-5 * 3600, 0, "pool.ntp.org", "time.nist.gov");    // NTP Ecuador (UTC-5)

  if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED) { // Si hay nube disponible
    if (Firebase.(&fbdo, "/esp32_2/conteo_huevos", contadorHuevos)) { // Sube conteo inicial
      huevosDirty        = false;                               // Ya está sincronizado
      ultimoOkFirebaseMs = millis();                            // Guarda instante ok
    }
  }

  enviarTelegram("⚙💧Sistema de bebedero iniciado");           // Mensaje de arranque por Telegram
}

// ********************EJECUCIÓN PRINCIPAL LOOP*************************
void loop() {
  unsigned long now = millis();                                 // Tiempo actual del ciclo

  manejarCambioWifi();                                          // Si cae WiFi, limpia cola

  bool          hayEdge   = false;                              // Bandera local del pulso
  unsigned long edgeTime  = 0;                                  // Tiempo del pulso

  noInterrupts();                                               // Protege lectura de variables ISR
  if (huevoEdgeFlag) {                                          // Si ISR marcó un huevo
    hayEdge       = true;                                       // Copia bandera
    edgeTime      = huevoEdgeMs;                                // Copia tiempo
    huevoEdgeFlag = false;                                      // Limpia bandera global
  }
  interrupts();                                                 // Restaura interrupciones

  if (hayEdge) {                                                // Si hubo huevo válido
    bool bombasOn = bomba1Estado || bomba2Estado;               // Revisa si bombas están encendidas

    bool intervaloOK = true;                                    // Por defecto, válido
    if (bombasOn) {                                             // Solo filtra si bomba

        intervaloOK = false;                                    // Se descarta
        Serial.print("[HUEVOS] Pulso ignorado por intervalo con bomba, delta="); // Log
        Serial.print(delta);                                    // Muestra delta
        Serial.println(" ms");                                  // Unidades
```

```
    }
  }

  if (intervaloOK) {                                    // Si se acepta como huevo real
    ultimoHuevoValidoMs = edgeTime;                     // Guarda el último huevo válido

    struct tm timeinfo;                                 // Variable para fecha/hora
    bool haveTime = getLocalTime(&timeinfo);            // Intenta obtener hora NTP
    if (haveTime) {                                     // Si hay hora
      int hoy = timeinfo.tm_mday;                       // Día del mes
      if (diaActualProduccion == -1) {                  // Primera vez
        diaActualProduccion = hoy;                      // Inicializa día
        huevosDia = 0;                                  // Reinicia conteo diario
      } else if (hoy != diaActualProduccion) {          // Si cambió el día
        diaActualProduccion = hoy;                      // Actualiza día
        huevosDia = 0;                                  // Reinicia conteo diario
      }
    }

    contadorHuevos++;                                   // Suma al conteo total
    huevosDia++;                                        // Suma al conteo diario
    huevosDirty = true;                                 // Marca pendiente por si no hay red


    if (!aviso55Enviado && contadorHuevos == 55) {            //Envía alerta a telegram, cuando
cumpla 55 huevos
      aviso55Enviado = true;
      enviarTelegram("🥚⚠️ Alerta: ya hay 55 huevos acumulados. Prepararse para recolección.");
    }
    Serial.print("🥚 Huevo detectado ULTRA RÁPIDO (bombas=");  // Log
    Serial.print(bombasOn ? "ON" : "OFF");                     // Estado de bombas
    Serial.print("). Conteo actual: ");                        // Texto
    Serial.println(contadorHuevos);                           // Valor actual

    if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED) { // Si Firebase
disponible
      if (Firebase.RTDB.setInt(&fbdo, "/esp32_2/conteo_huevos", contadorHuevos)) { // Sube conteo
        huevosDirty        = false;                       // Ya sincronizado
        ultimoOkFirebaseMs = edgeTime;                    // Marca último OK
      }

      if (haveTime) {                                     // Si se tiene hora, se registra
producción diaria
        char fecha[11];                                   // "YYYY-MM-DD"
        strftime(fecha, sizeof(fecha), "%Y-%m-%d", &timeinfo); // Formatea fecha

        FirebaseJson jsonDia;                             // JSON diario
        jsonDia.set("timestamp", fecha);                  // Fecha
        jsonDia.set("total_dia", huevosDia);              // Total diario

        String pathDia = String("/esp32/produccion_diaria/") + fecha; // Ruta por fecha
        if (Firebase.RTDB.setJSON(&fbdo, pathDia.c_str(), &jsonDia)) { // Envía JSON
          Serial.print("[PROD DIA] ");                    // Log
          Serial.print(fecha);                            // Fecha
          Serial.print(" = ");                            // Separador
          Serial.println(huevosDia);                      // Total
        } else {
          Serial.print("[PROD DIA] Error: ");             // Log error
          Serial.println(fbdo.errorReason());             // Motivo
        }
      }
    }

    if (contadorHuevos == 60) {                           // Al llegar a 60 huevos
      struct tm timeinfo60;                               // Para registrar hora del evento
      if (!getLocalTime(&timeinfo60)) {                   // Si no hay hora
        strcpy(ultimo60Timestamp, "sin_fecha");           // Texto alternativo
      } else {
        strftime(ultimo60Timestamp, sizeof(ultimo60Timestamp), "%Y-%m-%d %H:%M:%S", &timeinfo60); //
Fecha completa
      }
      ultimo60Dirty = true;                               // Queda pendiente si falla la nube

      enviarTelegram("🥚🔔 Hora de recolectar, se registraron 60 huevos"); // Aviso al usuario
```

```
        if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED) { // Si Firebase
disponible
          FirebaseJson json;                                    // JSON del evento
          json.set("timestamp", ultimo60Timestamp);             // Fecha/hora
          json.set("cantidad", 60);                             // Cantidad

          if (Firebase.RTDB.setJSON(&fbdo, "/esp32_2/ultima_vez_60_huevos", &json)) { // Guarda último
evento
            ultimo60Dirty      = false;                         // Ya sincronizado
            ultimoOkFirebaseMs = edgeTime;                      // Marca último OK
          } else {
            Serial.print("[60] Error ultima_vez_60_huevos: ");  // Log
            Serial.println(fbdo.errorReason());                 // Motivo
          }

          if (!Firebase.RTDB.pushJSON(&fbdo, "/esp32_2/historial_60", &json)) { // Historial
            Serial.print("[60] Error push historial_60: ");     // Log
            Serial.println(fbdo.errorReason());                 // Motivo
          }

          if (Firebase.RTDB.setInt(&fbdo, "/esp32_2/conteo_huevos", 0)) { // Reinicia en nube
            huevosDirty        = false;                         // Sin pendiente
            ultimoOkFirebaseMs = edgeTime;                      // Marca OK
          } else {
            Serial.print("[60] Error reset conteo_huevos: ");   // Log
            Serial.println(fbdo.errorReason());                 // Motivo
          }
        }

        contadorHuevos = 0;                                     // Reinicia conteo local
        huevosDirty    = true;                                  // Marca para sincronizar el nuevo
valor
      }
    }
  }

  // ===== Lecturas estables de sensores =====
  int dTanque   = medirEstableInt(TRIG1_PIN, ECHO1_PIN, emaTanque);   // Distancia estable tanque
  int dBebedero = medirEstableInt(TRIG2_PIN, ECHO2_PIN, emaBebedero); // Distancia estable bebedero

  Serial.print("[SENSORES] Tanque: ");                          // Log
  Serial.print(dTanque);                                        // Valor tanque
  Serial.print(" cm | Bebedero: ");                             // Separador
  Serial.print(dBebedero);                                      // Valor bebedero
  Serial.print(" cm | Huevos: ");                               // Separador
  Serial.println(contadorHuevos);                               // Conteo

  // ===== Lógica de tanque =====
  if (dTanque > 0) {                                            // Solo si lectura válida
    if (dTanque >= TANQUE_MEDIO_MIN_CM) {                       // Si se alejó de lleno hacia medio
      tanqueLlenoNotificado = false;                            // Rearma aviso de “lleno”
    }

    if (dTanque >= TANQUE_CRITICO_MIN_CM && dTanque <= TANQUE_CRITICO_MAX_CM) { // Rango crítico
      if (!avisoCriticoEnviado) {                               // Primera vez en crítico
        enviarTelegram("⚠️Nivel crítico en el tanque");         // Alerta inicial
        avisoCriticoEnviado = true;                             // Marca avisado
      } else if (now - ultimoCriticoMs >= RECORDATORIO_CRITICO_MS) { // Si ya pasaron 3 horas
        enviarTelegram("⚠️Recordatorio: el tanque sigue en nivel CRÍTICO, recargar urgentemente"); //
Recordatorio
        ultimoCriticoMs = now;                                  // Reinicia contador del recordatorio
      }
    } else {                                                    // Si ya no está crítico
      avisoCriticoEnviado = false;                              // Permite avisar de nuevo si vuelve
      ultimoCriticoMs = 0;                                      // Reinicia temporizador
    }

    if (!bloqueoTanque && dTanque > TANQUE_PARADA_UMBRAL_CM) {  // Si se considera vacío y no estaba
bloqueado
      bloqueoTanque = true;                                     // Bloquea el sistema
      bomba1On(false);                                          // Apaga bombas
      bomba2On(false);                                          // Apaga bombas
      estado = Estado::IDLE;                                    // Fuerza reposo
```

```
      String msg = mensajeTanque(dTanque);                    // Mensaje (vacío/lleno)
      enviarTelegram(msg);                                    // Envía alerta
    }

    if (bloqueoTanque && dTanque <= TANQUE_REINICIO_CM) {     // Si recargó y vuelve al rango
“lleno”
      bloqueoTanque = false;                                  // Quita bloqueo
      String msg = mensajeTanque(dTanque);                    // Mensaje de recuperación
      enviarTelegram(msg);                                    // Notifica
    }

    if (primeraLecturaTanque) {                               // Primera lectura tras encender
      ultimoNivelTanque    = dTanque;                         // Guarda referencia
      primeraLecturaTanque = false;                           // Ya hay dato previo

      if (!tanqueLlenoNotificado && dTanque > 0 && dTanque <= TANQUE_BUZZER_CM) { // Si ya arrancó
lleno
        iniciarBuzzer5s(now);                                 // Buzzer 5s
        String msg = mensajeTanque(dTanque);                  // Mensaje de lleno
        enviarTelegram(msg);                                  // Envía aviso
        tanqueLlenoNotificado = true;                         // Evita repetir


  actualizarBuzzer(now);                                      // Apaga buzzer al cumplir 5s
  gestionarProgramacionDrenaje();                             // Revisa limpieza del miércoles

  // ===== Máquina de estados del bebedero =====
  if (!bloqueoTanque) {                                       // Solo si el tanque permite operar
    if (estado == Estado::IDLE && dBebedero >= FILL_START_CM && dBebedero > 0) { // Si está bajo
      estado = Estado::FILL_FROM_TANK;                        // Cambia a recarga
      estadoInicioMs = now;                                   // Marca inicio
      bomba1On(true);                                         // Enciende bomba 1
    }

    switch (estado) {                                         // Control por estado
      case Estado::FILL_FROM_TANK: {
        bool reached = (dBebedero > 0 && dBebedero <= FILL_STOP_CM); // Se llenó
        bool timeout = (now - estadoInicioMs >= MAX_FILL_MS);       // O se pasó del tiempo
        if (reached || timeout) {                             // Cualquiera detiene
          bomba1On(false);                                    // Apaga bomba 1
          estado = Estado::IDLE;                              // Vuelve a reposo
        }
      } break;

      case Estado::DRAIN_CYCLE: {
        bool reached = (dBebedero > 0 && dBebedero >= 6);         // Drenó suficiente
        bool timeout = (now - estadoInicioMs >= MAX_DRAIN_MS);    // O se pasó del tiempo
        if (reached || timeout) {
          bomba2On(false);                                    // Apaga bomba 2
          estado = Estado::REFILL_AFTER_DRAIN;                // Pasa a recargar
          estadoInicioMs = now;                               // Marca inicio
          bomba1On(true);                                     // Enciende bomba 1
        }
      } break;

      case Estado::REFILL_AFTER_DRAIN: {
        bool reached = (dBebedero > 0 && dBebedero <= FILL_STOP_CM); // Se llenó
        bool timeout = (now - estadoInicioMs >= MAX_FILL_MS);       // O se pasó del tiempo
        if (reached || timeout) {
          bomba1On(false);                                    // Apaga bomba 1
          estado = Estado::IDLE;                              // Vuelve a reposo
        }
      } break;

      case Estado::IDLE:
      default:
        break;                                                // No hace nada si está en reposo
    }
  } else {
    bomba1On(false);                                          // Si tanque bloqueado, apaga todo
    bomba2On(false);                                          // Si tanque bloqueado, apaga todo
    estado = Estado::IDLE;                                    // Fuerza reposo
  }
```

```
  // ===== Envío de niveles a Firebase cada 2 segundos =====
  if (firebaseReady && Firebase.ready() && WiFi.status() == WL_CONNECTED && now - ultimoEnvioNivelMs >
2000) {
    ultimoEnvioNivelMs = now;                                    // Actualiza marca de tiempo

    if (dTanque > 0) {                                           // Solo si lectura válida
      int nivelRealTanque = ALTURA_TOTAL_TANQUE - dTanque;        // Nivel real en cm
      if (nivelRealTanque < 0) nivelRealTanque = 0;               // Evita negativos
      Firebase.RTDB.setInt(&fbdo, "/esp32_2/nivel_agua_cm", nivelRealTanque); // Sube a Firebase
    }

    if (dBebedero > 0) {                                         // Solo si lectura válida
      int nivelRealBeb = ALTURA_TOTAL_BEBEDERO - dBebedero;       // Nivel real en cm
      if (nivelRealBeb < 0) nivelRealBeb = 0;                     // Evita negativos
      Firebase.RTDB.setInt(&fbdo, "/esp32_2/nivel_bebedero_cm", nivelRealBeb); // Sube a Firebase
    }

    ultimoOkFirebaseMs = now;                                    // Marca actividad reciente con
Firebase
  }

  reintentarWiFiNoBloqueante(now);                               // Reintenta WiFi si está caído
  if (!firebaseReady && WiFi.status() == WL_CONNECTED) {          // Si hay WiFi pero Firebase no
    reintentarFirebaseNoBloqueante(now);                         // Reintenta Firebase sin bloquear
  }

  intentarSyncPendiente(now);                                    // Intenta subir datos pendientes
  procesarColaTelegram();                                        // Envía Telegram con orden y pausa

  delay(5);                                                      // Pausa pequeña para estabilidad
}

//************************FUNCIÓN URLENCODE (PARA TELEGRAM)*****************************
String urlencode(String str) {                                   // Codifica el mensaje para enviarlo
en URL
  String encoded = "";                                           // Resultado final
  char c;                                                        // Carácter actual
  const char* hex = "0123456789ABCDEF";                          // Tabla hexadecimal

  for (int i = 0; i < str.length(); i++) {                       // Recorre el mensaje
    c = str.charAt(i);                                           // Toma el carácter actual
    if (isalnum(c)) {                                            // Si es letra o número
      encoded += c;                                              // Se agrega sin cambios
    } else {                                                     // Si es espacio, símbolo, emoji, etc.
      encoded += '%';                                            // Prefijo de codificación
      encoded += hex[(c >> 4) & 0xF];                             // Parte alta en hex
      encoded += hex[c & 0xF];                                    // Parte baja en hex
    }
  }
  return encoded;                                                // Devuelve texto ya codificado
}
```

Anexo E. Código del dashboard web (HTML y JavaScript)

Index HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Visualización del Control de Alimentación - Codorniz Japonesa</title>

  <!-- Fuente -->
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link
    href="https://fonts.googlea=Poppins:wght@300;400;600;700&display=swap"
    rel="stylesheet"
  />

  <!-- Firebase -->
```

```
    <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase.js"></script>
    <!-- Chart.js -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <style>
      * {
        box-sizing: border-box;
        margin: 0;
        padding: 0;
      }

      html,
      body {
        height: 100%;
      }

      body {
        font-family: "Poppins", sans-serif;
        min-height: 100vh;
        background: radial-gradient(circle at top, #1f2937 0, #020617 55%, #020617 100%);
        color: #e5e7eb;
        display: flex;
        flex-direction: column;
        overflow-x: hidden;
      }

      /* ---------- TOPBAR ---------- */
      .topnav {
        width: 100%;
        padding: 8px 14px;
        background: rgba(3, 7, 18, 0.92);
        display: flex;
        align-items: center;
        justify-content: space-between;
        box-shadow: 0 4px 18px rgba(0, 0, 0, 0.8);
        position: sticky;
        top: 0;
        z-index: 30;
      }

      .topnav-center {
        display: flex;
        flex-direction: column;
        align-items: center;
        flex-grow: 1;
      }

      .topnav-center h1 {
        font-size: 1.25rem;
        color: #22d3ee;
        text-shadow: 0 0 12px #22d3ee;
        text-align: center;
        line-height: 1.2;
      }

      .topnav-center h2 {
        font-size: 0.85rem;
        color: #bae6fd;
        opacity: 0.95;
        text-align: center;
        margin-top: 2px;
      }

      .topnav-left,
      .topnav-right {
        display: flex;
        align-items: center;
        justify-content: center;
        gap: 0;
```

```
      }

      .logo-box {
        width: 60px;
        height: 60px;
        border-radius: 14px;
        background: radial-gradient(circle at top, #020617 0, #020617 60%);
        align-items: center;
        justify-content: center;
        box-shadow: 0 0 10px rgba(15, 23, 42, 0.9);
        border: 1px solid rgba(148, 163, 184, 0.4);
      }

      .logo {
        max-width: 100%;
        max-height: 100%;
        display: block;
      }

      @media (max-width: 600px) {
        .topnav {
          padding: 6px 10px;
        }

        .topnav-center h1 {
          font-size: 1.05rem;
        }

        .topnav-center h2 {
          font-size: 0.8rem;
        }

        .logo-box {
          width: 46px;
          height: 46px;
          padding: 3px;
        }
      }

      /* ---------- MENÚ ---------- */
      .menu-bar {
        width: 100%;
        background: #f9fafb;
        display: flex;
        align-items: center;
        border-bottom: 1px solid #e5e7eb;
        box-shadow: 0 2px 10px rgba(15, 23, 42, 0.2);
        z-index: 25;
      }

      .menu-inner {
        width: 100%;
        max-width: 980px;
        margin: 0 auto;
        display: flex;
        align-items: center;
      }

      .menu-toggle {
        border: none;
        background: transparent;
        padding: 10px 14px;
        cursor: pointer;
        display: flex;
        align-items: center;
        justify-content: center;
      }

      .menu-icon {
```

```
        font-size: 1.5rem;
        color: #22d3ee;
      }

      .menu-title {
        border-left: 1px solid #e5e7eb;
        padding: 10px 14px;
        flex: 1;
        display: flex;
        align-items: center;
        justify-content: space-between;
        font-size: 0.9rem;
        color: #111827;
        cursor: pointer;
      }

      .menu-title span.arrow {
        font-size: 0.8rem;
        margin-left: 6px;
        color: #6b7280;
      }

      .menu-dropdown {
        display: none;
        background: #ffffff;
        border-bottom: 1px solid #e5e7eb;
        box-shadow: 0 8px 20px rgba(15, 23, 42, 0.25);
        z-index: 24;
      }

      .menu-dropdown-open {
        display: block;
      }

      .menu-dropdown-inner {
        width: 100%;
        max-width: 980px;
        margin: 0 auto;
        display: flex;
        flex-direction: column;
      }

      .menu-item {
        border: none;
        background: transparent;
        text-align: left;
        font-size: 0.9rem;
        padding: 10px 16px;
        cursor: pointer;
        color: #111827;
        display: flex;
        align-items: center;
        gap: 8px;
      }

      .menu-item:hover {
        background: #f3f4f6;
      }

      .menu-item-active {
        background: #e5f3ff;
        font-weight: 600;
        color: #111827;
      }

      .menu-item-icon {
        font-size: 1.1rem;
      }
```

```
      @media (max-width: 600px) {
        .menu-title {
          font-size: 0.85rem;
          padding: 8px 10px;
        }

      }

      /* ---------- ÁREA DE PÁGINAS ---------- */
      .pages {
        flex: 1;
        width: 100%;
        max-width: 980px;
        margin: 0 auto;
        padding: 12px 12px 20px;
        display: flex;
        flex-direction: column;
        gap: 12px;
      }

      .page {
        display: none;
        animation: fadeIn 0.25s ease-out;
        position: relative;
      }

      .page-active {
        display: block;
      }

      @keyframes fadeIn {
        from {
          opacity: 0;
          transform: translateY(4px);
        }
        to {
          opacity: 1;
          transform: translateY(0);
        }
      }

      /* ---------- TARJETAS GENERALES ---------- */
      .card {
        background: rgba(15, 23, 42, 0.95);
        border-radius: 18px;
        padding: 12px 12px 14px;
        border: 1px solid rgba(148, 163, 184, 0.4);
        box-shadow: 0 10px 32px rgba(15, 23, 42, 0.9);
        backdrop-filter: blur(10px);
        -webkit-backdrop-filter: blur(10px);
        transition: 0.18s ease;
      }


      .status {
        display: inline-block;
        margin-top: 4px;
        padding: 3px 8px;
        border-radius: 999px;
        font-size: 0.7rem;
        font-weight: 600;
        border: 1px solid rgba(148, 163, 184, 0.75);
      }

      /* ---------- PÁGINA 1: DASHBOARD ---------- */
      .levels-row {
        display: flex;
        gap: 10px;
        margin-bottom: 10px;
```

```
    flex-wrap: wrap;
  }

  .card-comida,
  .card-agua {
    flex: 1;
    min-width: 0;
    position: relative;
    overflow: hidden;
  }

  .card-comida::before,
  .card-agua::before {
    content: "";
    position: absolute;
    inset: 0;
    opacity: 0.18;
    pointer-events: none;
    background-size: cover;
    background-position: center;
  }

  .card-comida::before {
    background-image: url("./balanceado.png");
  }

  .card-agua::before {
    background-image: url("./agua.png");
  }

  .card-content {
    position: relative;
    z-index: 1;
  }

  .reading {
    font-size: 1.4rem;
    font-weight: 700;
    color: #38bdf8;
  }

  .reading small {
    font-size: 0.85rem;
  }

  .reading-sub {
    font-size: 0.78rem;
    color: #e5e7eb;
  }

  .reading-sub span {
    font-weight: 600;
    color: #f97316;
  }

  .tank-wrapper {
    margin: 6px auto 4px;
    display: flex;
    justify-content: center;
  }

  .tank {
    position: relative;
    width: 46px;
    height: 96px;
    border-radius: 14px;
    border: 2px solid rgba(148, 163, 184, 0.7);
    background: linear-gradient(to bottom, rgba(15, 23, 42, 0.95), #020617);
    overflow: hidden;
```

```
      }

      .tank-fill {
        position: absolute;
        left: 0;
        bottom: 0;
        width: 100%;
        height: 0%;
        transition: height 0.35s ease;
      }

      #fill-agua {
        background: linear-gradient(to top, #0ea5e9, #38bdf8, #a5f3fc);
      }

      .tank-comida {
        width: 54px;
        height: 96px;
        border: none;
        background: transparent;
        box-shadow: none;
      }


      .tank-comida .tank-fill {
        clip-path: polygon(12% 0, 88% 0, 74% 100%, 26% 100%);
        background: linear-gradient(to top, #b91c1c, #f97316, #fde68a);
        box-shadow: 0 0 14px rgba(248, 250, 252, 0.4);
      }

      /* ---------- HUEVOS ---------- */
      .card-huevos {
        position: relative;
        overflow: hidden;
        justify-content: center;
        min-height: 180px;
      }

      .card-huevos::before {
        content: "";
        position: absolute;
        inset: 0;
        opacity: 0.35;
        background: url("./huevos.png") center/cover no-repeat;
        pointer-events: none;
      }

      .card-huevos .card-content {
        position: relative;
        z-index: 1;
        display: flex;
        flex-direction: column;
        align-items: center;
      }

      .egg-container {
        width: clamp(80px, 22vw, 150px);
        aspect-ratio: 3 / 4;
        background: #fffbdb;
        border-radius: 50% 50% 45% 45% / 60% 60% 40% 40%;
        margin: 12px auto 8px;
        box-shadow: 0 0 10px #facc15, inset 0 0 6px rgba(0, 0, 0, 0.3);
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: clamp(1.4rem, 4.5vw, 2rem);
        font-weight: bold;
        color: #ff0080;
        text-shadow: 0 0 4px #ff0080;
```

```
    animation: eggPulse 2s ease-in-out infinite alternate;
  }

  @keyframes eggPulse {
    from {
      transform: scale(1);
    }
    to {
      transform: scale(1.04);
    }
  }

  .barra-huevos {
    width: 100%;
    height: 14px;
    background-color: rgba(15, 23, 42, 0.96);
    border-radius: 999px;
    margin: 6px 0 4px;
    overflow: hidden;
    border: 1px solid rgba(148, 163, 184, 0.8);
  }

  .barra-huevos-inner {
    height: 100%;
    border-radius: 999px;
    background: linear-gradient(to right, #22c55e, #eab308, #ef4444);
    width: 0%;
    transition: width 0.3s ease;
  }

  .hora-60 {
    font-size: 0.78rem;
    margin-top: 4px;
    background: rgba(15, 23, 42, 0.96);
    border: 1px solid rgba(250, 204, 21, 0.6);
    width: 100%;
    max-width: 100%;
  }

  @media (max-width: 640px) {
    .levels-row {
      flex-direction: row;
    }

    .card-comida,
    .card-agua {
      flex: 1 1 calc(50% - 5px);
    }

    .pages {
      padding: 10px 8px 16px;
    }

    .card-huevos {
      min-height: 160px;
    }
  }

  /* ---------- PÁGINA 2: INFO CODORNICES ---------- */
  #page-info {
    position: relative;
    color: #3b2f2f;
  }

  #page-info > .card-info-quail {
    position: relative;
    z-index: 1;
    background: rgba(255, 250, 240, 0.9);
    border: 1px solid rgba(196, 164, 132, 0.9);
```

```
    color: #4a2c2a;
    box-shadow: 0 18px 40px rgba(15, 23, 42, 0.85);
    backdrop-filter: blur(3px);
    -webkit-backdrop-filter: blur(3px);
    overflow: hidden;
  }

  #page-info > .card-info-quail::before {
    content: "";
    position: absolute;
    inset: 0;
    background: url("codorniz.png") center/contain no-repeat;
    opacity: 0.9;
    filter: brightness(1.2) contrast(1.12);
    pointer-events: none;
    z-index: 0;
  }

  #page-info > .card-info-quail > * {
    position: relative;
    z-index: 1;
  }

  #page-info .card-subtitle {
    color: #8b5a2b;
  }

  #page-info h2 {
    color: #5a3e2b;
  }

  .info-header-box {
    background: rgba(255, 250, 240, 0.8);
    border-radius: 14px;
    padding: 8px 10px;
    border: 1px solid rgba(214, 182, 138, 0.9);
    margin-top: 6px;
    margin-bottom: 10px;
    box-shadow: 0 10px 28px rgba(0, 0, 0, 0.22);
    font-size: 0.8rem;
    color: #3b2f2f;
    text-align: justify;
  }

  .info-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
    gap: 10px;
    margin-top: 4px;
  }

  #page-info .info-grid .card {
    background: rgba(255, 250, 240, 0.42);
    border: 1px solid rgba(214, 182, 138, 0.95);
    box-shadow: 0 12px 32px rgba(0, 0, 0, 0.28);
    backdrop-filter: blur(2px);
    -webkit-backdrop-filter: blur(2px);
  }

  .info-item-title {
    font-weight: 600;
    margin-bottom: 4px;
    font-size: 0.95rem;
  }

  .info-item-text {
    font-size: 0.82rem;
    color: #1f2933;
    text-align: justify;
```

```
      }

      .info-icon {
        font-size: 1.4rem;
        margin-right: 6px;
      }

      /* ---------- PÁGINA 3: HISTÓRICO ---------- */
      .card-history h2 {
        margin-bottom: 4px;
      }

      .history-intro-card {
        margin-top: 8px;
        margin-bottom: 8px;
        background: rgba(15, 23, 42, 0.96);
        border-radius: 14px;
        border: 1px solid rgba(148, 163, 184, 0.7);
        padding: 8px 10px;
        font-size: 0.78rem;
        color: #e5e7eb;
        text-align: justify;
      }

      .historial-title {
        margin-top: 8px;
        font-size: 0.78rem;
        font-weight: 600;
        color: #e5e7eb;
      }

      .month-year-row {
        margin: 6px 0 8px;
        display: flex;
        flex-wrap: wrap;
        gap: 8px;
        align-items: center;
        justify-content: space-between;
      }

      .month-tabs {
        display: flex;
        flex-wrap: wrap;
        gap: 6px;
      }

      .month-tab {
        padding: 4px 10px;
        font-size: 0.78rem;
        border-radius: 999px;
        border: 1px solid rgba(148, 163, 184, 0.8);
        background: rgba(15, 23, 42, 0.9);
        color: #e5e7eb;
        cursor: pointer;
        transition: 0.15s ease;
      }

      .month-tab:hover {
        background: rgba(30, 64, 175, 0.9);
        border-color: rgba(129, 140, 248, 0.9);
      }

      .month-tab-active {
        background: #22d3ee;
        color: #0f172a;
        border-color: #22d3ee;
        box-shadow: 0 0 10px rgba(34, 211, 238, 0.6);
      }

```

```
      .year-select-wrapper {
        display: flex;
        align-items: center;
        gap: 4px;
        font-size: 0.75rem;
        color: #d1d5db;
      }

      .calendar-wrapper {
        margin-top: 10px;
      }

      .calendar-month-label {
        margin-top: 2px;
        font-size: 0.78rem;
        color: #cbd5f5;
      }

      .calendar-legend {
        font-size: 0.7rem;
        margin: 4px 0;
        display: flex;
        gap: 10px;
        flex-wrap: wrap;
      }

      .legend-dot {
        width: 10px;
        height: 10px;
        border-radius: 999px;
        display: inline-block;
        margin-right: 4px;
      }

      .legend-none {
        background: #ef4444;
      }

      .legend-low {
        background: #f97316;
      }

      .legend-mid {
        background: rgba(56, 189, 248, 0.6);
      }

      .legend-high {
        background: rgba(52, 211, 153, 0.9);
      }

      .calendar-grid {
        display: grid;
        grid-template-columns: repeat(7, 1fr);
        gap: 3px;
        margin-top: 4px;
      }


      }

      .calendar-day--empty {
        background: transparent;
        border: none;
      }

      .calendar-cycle-badge {
        padding: 1px 4px;
        border-radius: 999px;
        font-size: 0.6rem;
```

```
          background: #facc15;
          color: #111827;
          box-shadow: 0 0 6px rgba(250, 204, 21, 0.9);
          font-weight: 700;
          white-space: nowrap;
        }

        @media (max-width: 600px) {
          .calendar-day {
            min-height: 38px;
          }
          .calendar-grid {
            gap: 2px;
          }
        }

        .summary-row {
          display: flex;
          flex-wrap: wrap;
          gap: 8px;
          margin: 8px 0 4px;
        }

        .summary-card {
          flex: 1 1 150px;
          background: rgba(15, 23, 42, 0.96);
          border-radius: 12px;

          flex-direction: column;
          justify-content: center;
          min-height: 48px;
        }

        .summary-label {
          font-size: 0.72rem;
          color: #9ca3af;
          margin-bottom: 2px;
        }

        .summary-value {
          font-size: 0.95rem;
          font-weight: 600;
          color: #e5e7eb;
        }

        .manage-wrapper {
          margin-top: 10px;
          padding: 8px 10px;
          border-radius: 12px;
          background: rgba(15, 23, 42, 0.96);
          border: 1px dashed rgba(148, 163, 184, 0.7);
        }

        .manage-row {
          display: flex;
          flex-wrap: wrap;
          gap: 8px;
          align-items: center;
          margin-bottom: 6px;
        }

        .manage-label {
          font-size: 0.75rem;
          color: #d1d5db;
        }

        .manage-select {
          background: #020617;
          color: #e5e7eb;
```

```
          border-radius: 999px;
          border: 1px solid rgba(148, 163, 184, 0.8);
          padding: 3px 10px;
          font-size: 0.8rem;
          outline: none;
        }

        .manage-select:focus {
          border-color: #22d3ee;
          box-shadow: 0 0 0 1px rgba(34, 211, 238, 0.6);
        }

        .manage-section-title {
          font-size: 0.74rem;
          font-weight: 600;
          color: #e5e7eb;
          margin-bottom: 2px;
          display: flex;
          align-items: center;
          gap: 6px;
        }

        .manage-section-title span.badge {
          font-size: 0.65rem;
          padding: 2px 6px;
          border-radius: 999px;
          border: 1px solid rgba(148, 163, 184, 0.8);
          background: rgba(15, 23, 42, 0.9);
          color: #e5e7eb;
        }

        .manage-section-desc {
          font-size: 0.7rem;
          color: #9ca3af;
          margin-bottom: 6px;
        }

        .manage-block {
          padding: 6px 0 8px;
          border-top: 1px dashed rgba(55, 65, 81, 0.7);
          margin-top: 6px;
        }

        .manage-block:first-child {
          border-top: none;
          margin-top: 0;
        }

        .manage-buttons {
          display: flex;
          flex-wrap: wrap;
          gap: 6px;
          margin-top: 2px;
          margin-bottom: 2px;
        }

        /* BOTONES CELESTES: producción diaria (día/mes/año) */
        .manage-btn
          cursor: pointer;
          font-weight: 500;
          transition: 0.15s ease;
        }

        .manage-btn:hover {
          background: #22d3ee22;
        }

        /* BOTONES NARANJAS: ciclos de 60 huevos (día/mes/año) */
        .manage-btn-warning {
```

```
          border-color: #f97316;
          color: #ffffff;
        }

        .manage-btn-warning:hover {
          background: #f9731622;
        }

        /* (Opcional) estilo rojo si algún día usas manage-btn-danger */
        .manage-btn-danger {
          border-color: #ef4444;
          color: #ef4444;
        }

        .manage-btn-danger:hover {
          background: #ef444422;
        }

        .manage-status {
          font-size: 0.72rem;
          color: #9ca3af;
          margin-top: 6px;
        }

        /* ---------- TABLA REGISTRO POR MES (RESPONSIVE) ---------- */
        .historial-table-wrapper-mobile {
          width: 100%;
          overflow-x: auto;
          -webkit-overflow-scrolling: touch;
          border-radius: 12px;
          margin-top: 8px;
          box-shadow: 0 4px 15px rgba(0, 0, 0, 0.6);
          position: relative;
          scrollbar-width: thin;
          scrollbar-color: #22c55e #020617;
        }

        .historial-table-wrapper-mobile::after {
          content: "";
          position: absolute;
          left: 0;
          right: 0;
          bottom: 0;
          height: 3px;
          border-radius: 999px;
          background: #22c55e;
          opacity: 0.85;
          pointer-events: none;
        }

        .historial-table-wrapper-mobile::-webkit-scrollbar {
          height: 6px;
        }

        .historial-table-wrapper-mobile::-webkit-scrollbar-track {
          background: #020617;
          border-radius: 999px;
        }

        .historial-table-wrapper-mobile::-webkit-scrollbar-thumb {
          background: #22c55e;
          border-radius: 999px;
        }

        .historial-table {
          width: 100%;
          min-width: 650px;
          border-collapse: collapse;
          font-size: 0.78rem;
```

```
    background: rgba(15, 23, 42, 0.98);
  }

  .historial-table thead th {
    background: linear-gradient(to right, #020617, #0b1120);
    font-size: 0.75rem;
    white-space: nowrap;
  }

  .historial-table th,
  .historial-table td {
    padding: 6px 8px;
    text-align: center;
    border-bottom: 1px solid rgba(255, 255, 255, 0.08);
    white-space: nowrap;
  }

  .historial-summary-row td {
    font-weight: 700;
    color: #a3ff12;
    background: rgba(34, 197, 94, 0.14);
  }

  .historial-empty-row td {
    text-align: center;
    color: #9ca3af;
  }

  @media (max-width: 600px) {
    .historial-table {
      font-size: 0.68rem;
    }
    .historial-table th,
    .historial-table td {
      padding: 4px 6px;
    }
  }

  /* ---------- GESTIÓN DE DATOS: DESPLEGABLE ---------- */
  .historial-title-row {
    margin-top: 14px;
    display: flex;
    align-items: center;
    justify-content: space-between;
    gap: 8px;
  }

  .gestion-toggle-btn {
    border: 1px solid rgba(248, 250, 252, 0.5);
    background: rgba(15, 23, 42, 0.95);
    color: #22d3ee;
    border-radius: 999px;
    padding: 4px 10px;
    font-size: 0.85rem;
    cursor: pointer;
    display: flex;
    align-items: center;
    gap: 6px;
    box-shadow: 0 6px 16px rgba(0, 0, 0, 0.7);
    transition: 0.15s ease;
  }

  .gestion-toggle-btn:hover {
    background: rgba(34, 211, 238, 0.12);
  }

  .gestion-toggle-btn.gestion-open {
    border-color: #f97316;
    color: #f97316;
```

```
      }

      .manage-wrapper-collapsed {
        display: none;
      }

      /* ---------- MODAL DE CONFIRMACIÓN ---------- */
      .confirm-backdrop {
        position: fixed;
        inset: 0;
        background: rgba(15, 23, 42, 0.86);
        display: flex;
        align-items: center;
        justify-content: center;
        z-index: 80;
      }

      .confirm-hidden {
        display: none;
      }

      .confirm-card {
        background: rgba(15, 23, 42, 0.98);
        border-radius: 16px;
        padding: 14px 16px 12px;
        max-width: 340px;
        width: 90%;
        border: 1px solid rgba(148, 163, 184, 0.9);
        box-shadow: 0 20px 50px rgba(0, 0, 0, 0.95);
        font-size: 0.85rem;
      }

      .confirm-title {
        font-size: 0.9rem;
        margin-bottom: 6px;
        color: #f97316;
      }

      .confirm-message {
        font-size: 0.8rem;
        color: #e5e7eb;
        margin-bottom: 10px;
      }

      .confirm-buttons {
        display: flex;
        justify-content: flex-end;
        gap: 8px;
        margin-top: 4px;
      }

      .confirm-btn {
        font-size: 0.8rem;
        padding: 4px 10px;
        border-radius: 999px;
        border: 1px solid rgba(148, 163, 184, 0.9);
        background: #020617;
        color: #e5e7eb;
        cursor: pointer;
        transition: 0.15s ease;
      }

      .confirm-btn-cancel:hover {
        background: rgba(148, 163, 184, 0.18);
      }

      .confirm-btn-ok {
        border-color: #ef4444;
      }
```

```
      .confirm-btn-ok:hover {
        background: rgba(239, 68, 68, 0.18);
      }
    </style>
  </head>
  <body>
    <!-- TOPBAR -->
    <div class="topnav">
      <div class="topnav-left">
        <div class="logo-box">
          <img src="utn.png" alt="UTN" class="logo" />
        </div>
      </div>
      <div class="topnav-center">
        <h1>Visualización del Control de Alimentación</h1>
        <h2>Codorniz Japonesa</h2>
      </div>
      <div class="topnav-right">
        <div class="logo-box">
          <img src="citel.png" alt="Telecomunicaciones" class="logo" />
        </div>
      </div>
    </div>

    <!-- MENÚ -->
    <div class="menu-bar">
      <div class="menu-inner">
        <button class="menu-toggle" id="menu-toggle">
          <span class="menu-icon">☰</span>
        </button>
        <div class="menu-title" id="menu-title">
          <span id="menu-title-label">Panel principal</span>
          <span class="arrow">▾</span>
        </div>
      </div>
    </div>

    <nav class="menu-dropdown" id="menu-dropdown">
      <div class="menu-dropdown-inner">
        <button class="menu-item menu-item-active" data-page="dashboard">
          <span class="menu-item-icon">⌨</span>
          <span>Panel principal</span>
        </button>
        <button class="menu-item" data-page="info">
          <span class="menu-item-icon">▨</span>
          <span>Información de codornices</span>
        </button>
        <button class="menu-item" data-page="history">
          <span class="menu-item-icon">📈</span>
          <span>Histórico de producción</span>
        </button>
      </div>
    </nav>

    <!-- CONTENIDO -->
    <main class="pages">
      <!-- PÁGINA 1 -->
      <section id="page-dashboard" class="page page-active">
        <div class="levels-row">
          <!-- COMIDA -->
          <div class="card card-comida">
            <div class="card-content">
              <div class="card-subtitle">Comedero</div>
              <h2>Nivel de comida</h2>
              <div class="reading">
                <span id="distance-comida">--</span><small> cm</small>
              </div>
              <div class="reading-sub">
```

```
                Nivel relativo: <span id="porcentaje-comida">--</span> %
              </div>
              <div class="tank-wrapper">
                <div class="tank tank-comida">
                  <div class="tank-fill" id="fill-comida"></div>
                </div>
              </div>
              <div class="status" id="status-comida">Cargando...</div>
            </div>
          </div>

          <!-- AGUA -->
          <div class="card card-agua">
            <div class="card-content">
              <div class="card-subtitle">Tanque de agua</div>
              <h2>Nivel de agua</h2>
              <div class="reading">
                <span id="distance-agua">--</span><small> cm</small>
              </div>
              <div class="reading-sub">
                Nivel relativo: <span id="porcentaje-agua">--</span> %
              </div>
              <div class="tank-wrapper">
                <div class="tank tank-agua">
                  <div class="tank-fill" id="fill-agua"></div>
                </div>
              </div>
              <div class="status" id="status-agua">Cargando...</div>
            </div>
          </div>
        </div>

        <!-- HUEVOS -->
        <div class="card card-huevos">
          <div class="card-content">
            <div class="card-subtitle">Producción</div>
            <h2>Cantidad de huevos (actual)</h2>
            <div class="egg-container" id="egg-count">0</div>
            <div class="barra-huevos">
              <div class="barra-huevos-inner" id="barra-huevos"></div>
            </div>
            <div class="status" id="status-huevos">Cargando...</div>
            <div class="hora-60" id="hora-60">Última vez con 60 huevos: --</div>
          </div>
        </div>
      </section>

      <!-- PÁGINA 2 -->
      <section id="page-info" class="page">
        <div class="card card-info-quail">
          <div class="card-subtitle">Información</div>
          <h2>Datos esenciales de la codorniz japonesa</h2>
          <div class="info-header-box">
            La codorniz japonesa es un ave pequeña y muy productiva, utilizada en sistemas
intensivos de
            cría por su alta postura de huevos, rápido crecimiento y buena adaptación. Requiere
un
            ambiente estable, agua limpia y una alimentación balanceada para mantener una
producción
            constante.
          </div>
          <div class="info-grid">
            <div class="card">
              <div class="info-item-title">
                <span class="info-icon">🥚</span>Producción de huevos
              </div>
              <p class="info-item-text">
                La codorniz japonesa inicia su producción de huevos aproximadamente a las seis
semanas
```

```
                de vida y mantiene un periodo productivo continuo de 22 a 24 semanas, pudiendo
                extenderse hasta las 30 semanas o más. Durante este ciclo, cada ave puede
    producir entre
                200 y 300 huevos, con un intervalo regular de alrededor de 22 horas entre cada
    postura,
                lo que evidencia su alta eficiencia reproductiva.
              </p>
            </div>
            <div class="card">
              <div class="info-item-title">
                <span class="info-icon">🌡</span>Condiciones ambientales
              </div>
              <p class="info-item-text">
                Para mantener un buen rendimiento productivo, la temperatura ideal está entre 18
    °C y
                30 °C. Los cambios bruscos de clima pueden generar estrés y disminuir la
    producción.
              </p>
            </div>
            <div class="card">
              <div class="info-item-title">
                <span class="info-icon">🍽</span>Alimentación y agua
              </div>
              <p class="info-item-text">
                Cada ave consume aproximadamente 20–25 g de alimento al día. El uso de un
    balanceado
                comercial como Exibal es recomendable porque está formulado con niveles
    adecuados de
                proteína, vitaminas y minerales que favorecen el desarrollo, fortalecen el
    sistema
                inmunológico y mejoran la producción de huevos. Las codornices también requieren
    acceso
                permanente a agua limpia.
              </p>
            </div>
          </div>
        </div>
      </section>

      <!-- PÁGINA 3 -->
    <section id="page-history" class="page">
      <div class="card card-history">
        <div class="card-subtitle">Histórico</div>
        <div class="history-intro-card">
          En esta sección podrás visualizar, la producción diaria de huevos, los ciclos completos
    de 60
        huevos, el calendario de días con recolecciones y un resumen organizado por semanas.
          <i>Usa el selector de año y las pestañas de meses para explorar el historial de tu
    criadero.</i>
        </div>

        <h2>Producción diaria y ciclos de 60 huevos (mes seleccionado)</h2>
        <div class="month-year-row">
          <div class="year-select-wrapper">
            <span>Año:</span>
            <select id="history-year-select" class="manage-select"></select>
          </div>
          <div class="month-tabs" id="month-tabs"></div>
        </div>

        <canvas id="grafico-mensual" height="200"></canvas>

        <div class="summary-row">
          <div class="summary-card">
            <div class="summary-label">Total de huevos en el mes</div>
            <div class="summary-value" id="resumen-huevos-mes">0</div>
          </div>
          <div class="summary-card">
            <div class="summary-label">Total de ciclos completos de 60 huevos</div>
            <div class="summary-value" id="resumen-ciclos-mes">0</div>
```

```
        </div>
      </div>

      <div class="calendar-wrapper">
        <div class="historial-title" style="margin-top:10px;">
          Calendario de producción (mes seleccionado)
        </div>
        <div id="calendar-month-label" class="calendar-month-label">Mes seleccionado</div>
        <div class="calendar-legend">
          <span><span class="legend-dot legend-none"></span> Sin producción (0 huevos)</span>
          <span><span class="legend-dot legend-low"></span> Producción baja (1–5 huevos)</span>
          <span><span class="legend-dot legend-mid"></span> Producción media (6–15
huevos)</span>
          <span><span class="legend-dot legend-high"></span> Producción alta (16 o más
huevos)</span>
        </div>
        <div class="calendar-grid" id="calendar-grid"></div>
      </div>

      <div class="historial-title" style="margin-top:10px;">
        Registro por mes (resumen por semanas)
      </div>
      <div class="historial-table-wrapper-mobile">
        <table class="historial-table">
          <thead>
            <tr>
              <th>Semana</th>
              <th>Rango de días</th>
              <th>Huevos por semana</th>
              <th>Recolecciones (ciclos de 60)</th>
              <th>Fechas de recolección</th>
            </tr>
          </thead>
          <tbody id="historial-tbody">
            <tr class="historial-empty-row">
              <td colspan="5">Sin registros aún</td>
            </tr>
          </tbody>
        </table>
      </div>

      <!-- GESTIÓN DE DATOS DESPLEGABLE -->
      <div class="historial-title-row">
        <span class="historial-title">Gestión de datos (borrado manual)</span>
        <button id="toggle-gestion" class="gestion-toggle-btn" title="Mostrar / ocultar opciones
de borrado">
          🧹
        </button>
      </div>

      <div class="manage-wrapper manage-wrapper-collapsed" id="gestion-wrapper">

        <!-- Mensaje de estado al inicio -->
        <div id="gestion-status" class="manage-status">
          Seleccione el rango (día, mes o año) y luego la acción a realizar.
        </div>

        <!-- Bloque 1: Borrado puntual por día -->
        <div class="manage-block">
          <div class="manage-section-title">
            <span class="badge">Seleccione día a borrar</span>
          </div>
          <p class="manage-section-desc">
            Elimina únicamente los datos correspondientes al día elegido, sin afectar otros
días.
          </p>
          <div class="manage-row">
            <span class="manage-label">Día:</span>
            <select id="gestion-dia" class="manage-select"></select>
```

```
            <span class="manage-label">Mes:</span>
            <select id="gestion-mes" class="manage-select"></select>

            <span class="manage-label">Año:</span>
            <select id="gestion-anio" class="manage-select"></select>
          </div>
          <div class="manage-buttons">
            <button id="btn-borrar-produccion-dia" class="manage-btn">
              Borrar producción diaria (día)
            </button>
            <button id="btn-borrar-ciclos-dia" class="manage-btn manage-btn-warning">
              Borrar ciclos de 60 huevos (día)
            </button>
          </div>
        </div>

        <!-- Bloque 2: Borrado mensual -->
        <div class="manage-block">
          <div class="manage-section-title">
            <span class="badge">Seleccione mes a borrar</span>
          </div>
          <p class="manage-section-desc">
            Elimina todos los registros del mes elegido, manteniendo intactos los otros meses.

        <!-- Bloque 3: Borrado anual -->
        <div class="manage-block">
          <div class="manage-section-title">
            <span class="badge">Seleccione año a borrar</span>
          </div>
          <p class="manage-section-desc">
            Elimina todos los registros del año elegido. <b>Úsalo solo cuando estés seguro.</b>
          </p>
          <div class="manage-row">
            <span class="manage-label">Año a borrar:</span>
            <select id="gestion-anio-anual" class="manage-select"></select>
          </div>
          <div class="manage-buttons">
            <button id="btn-borrar-produccion-anio" class="manage-btn">
              Borrar producción diaria (año)
            </button>
            <button id="btn-borrar-ciclos-anio" class="manage-btn manage-btn-warning">
              Borrar ciclos de 60 huevos (año)
            </button>
          </div>
        </div>

      </div>
    </div>
  </section>
  </main>

    <!-- MODAL DE CONFIRMACIÓN -->
    <div id="confirm-modal" class="confirm-backdrop confirm-hidden">
      <div class="confirm-card">
        <h3 id="confirm-title" class="confirm-title">Confirmar borrado</h3>
        <p id="confirm-message" class="confirm-message">
          ¿Estás seguro de que deseas realizar esta acción? Esta operación no se puede deshacer.
        </p>
        <div class="confirm-buttons">
          <button id="confirm-cancel" class="confirm-btn confirm-btn-cancel">
            Cancelar
          </button>
          </button>
        </div>
      </div>
    </div>

    <script src="scripts/index.js"></script>
```

```
</body>
</html>

```

Index JavaScript

```
/*****************  CONFIG FIREBASE  ******************/
const firebaseConfig = {
  apiKey: "AyB3D4sciHuMYbBuQS5m9STZSm2mHh4",
  authDomain: "criaderocj-8b9.firebaseapp.com",
  databaseURL: "https://criaderocj-19-default-rtdb.firebaseio.com",
  projectId: "criaderocj-87fb9",
  storageBucket: "criaderocj-87fb9.appspot.com",
  messagingSenderId: "584571917796",
  appId: "1:584579c37fcbe3f1",
};

firebase.initializeApp(firebaseConfig);
const db = firebase.database();

/*****************  REFS RTC DB  ******************/
const refComida = db.ref("esp32_1/nivel_comida_real_cm");
const refAgua = db.ref("esp32_2/nivel_agua_cm");
const refHistorial60 = db.ref("esp32_2/historial_60");
const refProduccionDiaria = db.ref("esp32_2/produccion_diaria");

/*****************  ELEMENTOS DOM  ******************/

// Dashboard
const distComidaElem = document.getElementById("distance-comida");
const pctComidaElem = document.getElementById("porcentaje-comida");
const fillComidaElem = document.getElementById("fill-comida");
const statusComidaElem = document.getElementById("status-comida");

const distAguaElem = document.getElementById("distance-agua");
const pctAguaElem = document.getElementById("porcentaje-agua");
const fillAguaElem = document.getElementById("fill-agua");
const statusAguaElem = document.getElementById("status-agua");

const eggCountElem = document.getElementById("egg-count");
const barraHuevosElem = document.getElementById("barra-huevos");

// Histórico (página 3)
const calendarGrid = document.getElementById("calendar-grid");
const calendarMonthEl = document.getElementById("calendar-month-label");
const graficoCanvas = document.getElementById("grafico-mensual");
const monthTabsContainer = document.getElementById("month-tabs");
const historyYearSelect = document.getElementById("history-year-select");
const resumenHuevosMesEl = document.getElementById("resumen-huevos-mes");
const resumenCiclosMesEl = document.getElementById("resumen-ciclos-mes");
const historialTbody = document.getElementById("historial-tbody");

// Gestión de datos – día
const gestionAnioSelect = document.getElementById("gestion-anio");
const gestionMesSelect = document.getElementById("gestion-mes");
const gestionDiaSelect = document.getElementById("gestion-dia");

// Gestión de datos – mes
const gestionMesMesSelect = document.getElementById("gestion-mes-mes");
const gestionAnioMesSelect = document.getElementById("gestion-anio-mes");

// Gestión de datos – año
const gestionAnioAnualSelect = document.getElementById("gestion-anio-anual");

// Botones de borrado
const btnBorrarProdDia = document.getElementById("btn-borrar-produccion-dia");
```

```
  const btnBorrarCiclosDia = document.getElementById("btn-borrar-ciclos-dia");
  const btnBorrarProdMes = document.getElementById("btn-borrar-produccion-mes");
  const btnBorrarCiclosMes = document.getElementById("btn-borrar-ciclos-mes");
  const btnBorrarProdAnio = document.getElementById("btn-borrar-produccion-anio");
  const btnBorrarCiclosAnio = document.getElementById("btn-borrar-ciclos-anio");
  const gestionStatusEl = document.getElementById("gestion-status");

  // Gestión de datos - wrapper y toggle
  const gestionWrapper = document.getElementById("gestion-wrapper");
  const toggleGestionBtn = document.getElementById("toggle-gestion");

  // Modal de confirmación
  const confirmModal = document.getElementById("confirm-modal");
  const confirmTitleEl = document.getElementById("confirm-title");
  const confirmMessageEl = document.getElementById("confirm-message");
  const confirmCancelBtn = document.getElementById("confirm-cancel");
  const confirmOkBtn = document.getElementById("confirm-ok");

  let graficoMensual = null;

  // Páginas
  const pages = {
    dashboard: document.getElementById("page-dashboard"),
    info: document.getElementById("page-info"),
    history: document.getElementById("page-history"),
  };

  // Menú
  const menuToggle = document.getElementById("menu-toggle");
  const menuTitle = document.getElementById("menu-title");
  const menuTitleLabel = document.getElementById("menu-title-label");
  const menuDropdown = document.getElementById("menu-dropdown");
  const menuItems = document.querySelectorAll(".menu-item");

  /*****************  CONSTANTES  *****************/
  const MAX_COMIDA_CM = 30;
  const MAX_AGUA_CM = 32;
  const ALTURA_TOTAL_TANQUE = 32;
  const TANQUE_BUZZER_CM = 10;
  const TANQUE_PARADA_UMBRAL_CM = 30;

  // Modelo físico del comedero (mismos valores que el ESP32)
  const ALTURA_TOTAL_COMEDERO_CM = 40;
  const ALTURA_LLENO_COMEDERO_CM = 10;
  const ALTURA_MAX_COMIDA =
    ALTURA_TOTAL_COMEDERO_CM - ALTURA_LLENO_COMEDERO_CM;

  // Umbrales de tanque de agua (copiados del ESP32)
  const TANQUE_LLENO_MAX_CM = 15;
  const TANQUE_MEDIO_MIN_CM = 16;
  const TANQUE_MEDIO_MAX_CM = 20;
  const TANQUE_BAJO_MIN_CM = 21;
  const TANQUE_BAJO_MAX_CM = 25;
  const TANQUE_CRITICO_MIN_CM = 26;
  const TANQUE_CRITICO_MAX_CM = 30;

  const NOMBRES_MESES = [
    "enero",
    "febrero",
    "marzo",
    "abril",
    "mayo",
    "junio",
    "julio",
    "agosto",
    "septiembre",
    "octubre",
    "noviembre",
    "diciembre",
```

```
];

/*****************  ESTADO RESUMEN  *****************/
let cacheProduccionDiaria = null;
let cacheHistorial60 = null;
let selectedYear = null;
let selectedMonth = null;
let availableYears = [];

/*****************  UTILIDADES  *****************/
function nivelAPorcentaje(valorCm, maxCm) {
  if (typeof valorCm !== "number" || maxCm <= 0) return 0;
  const pct = (valorCm / maxCm) * 100;
  return Math.max(0, Math.min(100, pct));
}

function parseTs(ts) {
  if (!ts || typeof ts !== "string") return null;
  const d = new Date(ts.replace(" ", "T"));
  return isNaN(d.getTime()) ? null : d;
}

// === TANQUE: ALTURA → DISTANCIA → PORCENTAJE ===
function porcentajeTanqueDesdeNivel(nivelRealCm) {
  if (typeof nivelRealCm !== "number") return 0;
  let dCm = ALTURA_TOTAL_TANQUE - nivelRealCm;
  if (dCm < 0) dCm = 0;

  if (dCm <= 0) return 0; // sensor pegado a la tapa
  if (dCm <= TANQUE_BUZZER_CM) return 100;
  if (dCm >= TANQUE_PARADA_UMBRAL_CM) return 0;

  let pct =
    (100 * (TANQUE_PARADA_UMBRAL_CM - dCm)) /
    (TANQUE_PARADA_UMBAL_CM - TANQUE_BUZZER_CM);

  pct = Math.max(0, Math.min(100, pct));
  return Math.round(pct);
}
function mensajeTanqueUI(nivelRealCm) {
  let dCm = ALTURA_TOTAL_TANQUE - nivelRealCm;
  if (dCm < 0) dm = 0;

  const pct = porcentajeTanqueDesdeNivel(nivelRealCm);
  let mensajeBase = "";


    mensajeBase = "✅ Nivel: LLENO";
  }
  // Medio
  else if (dCm >= TANQUE_MEDIO_MIN_CM && dCm <= TANQUE_MEDIO_MAX_CM) {
    mensajeBase = "🟡 Nivel: MEDIO";
  }
  // Bajo
  else if (dCm >= TANQUE_BAJO_MIN_CM && dCm <= TANQUE_BAJO_MAX_CM) {
    mensajeBase = "⚠️ Nivel: BAJO";
  }
  // Crítico
  else if (dCm >= TANQUE_CRITICO_MIN_CM && dCm <= TANQUE_CRITICO_MAX_CM) {
    mensajeBase = "❗ Nivel: CRÍTICO";
  }
  // Vacío / por debajo del punto de parada
  else if (dCm > TANQUE_PARADA_UMBRAL_CM) {
    mensajeBase = "⛔ Nivel: VACÍO";
  }
  // Zona intermedia (entre lleno y medio, o entre bandas)
  else {
    mensajeBase = "✅ Nivel: LLENO";
  }
```

```
    return `${mensajeBase}`;
  }

  /*****************  NAVEGACIÓN  *****************/
  function setActivePage(pageKey, label) {
    Object.keys(pages).forEach((k) => {
      pages[k].classList.remove("page-active");
    });

    if (pages[pageKey]) {
      pages[pageKey].classList.add("page-active");
    }

    menuItems.forEach((item) => {
      item.classList.remove("menu-item-active");
      if (item.dataset.page === pageKey) {
        item.classList.add("menu-item-active");
      }
    });

    if (label) {
      menuTitleLabeltextContent = label;
    }
  }

  function toggleMenu() {
    if (!menuDropdown) return;
    menuDropdown.style.display =
      menuDropdown.style.display === "block" ? "none" : "block";
  }

  if (menuToggle) {
    menuToggle.addEventListener("click", (e) => {
      e.stopPropagation();
      toggleMenu();
    });
  }

  if (menuTitle) {
    menuTitle.addEventListener("click", (e) => {
      e.stopPropagation();
      toggleMenu();
    });
  }

  menuItems.forEach((item) => {
    item.addEventListener("click", () => {
      const pageKey = item.dataset.page;
      const label = item.textContent.trim();
      setActivePage(pageKey, label);
      if (menuDropdown) menuDropdown.style.display = "none";
    });
  });

  document.addEventListener("click", (e) => {
    if (
      menuDropdown &&
      !menuDropdown.contains(e.target) &&
      !menuToggle.contains(e.target) &&
      !menuTitle.contains(e.target)
    ) {
      menuDropdown.style.display = "none";
    }
  });

  setActivePage("dashboard", "Panel principal");
  if (menuDropdown) {
    menuDropdown.style.display = "none";
```

```
  }

  /*****************  LECTURAS EN TIEMPO REAL  *****************/

  // COMIDA
  statusComidaElem.textContent = "Conectando...";
  refComida.on("value", (snapshot) => {
    if (!snapshot.exists()) {
      statusComidaElem.textContent = "Dato no disponible";
      distComidaElem.textContent = "--";
      pctComidaElem.textContent = "--";
      fillComidaElem.style.height = "0%";
      return;
    }

    // Mostramos la altura tal cual en cm
    distComidaElem.textContent = alturaCmRaw.toFixed(0);

    // Limitamos la altura a los mismos límites del firmware
    let alturaCm = alturaCmRaw;
    if (alturaCm < 0) alturaCm = 0;
    if (alturaCm > ALTURA_MAX_COMIDA) alturaCm = ALTURA_MAX_COMIDA;

    // Calculamos la distancia equivalente al sensor
    const distCm = ALTURA_TOTAL_COMEDERO_CM - alturaCm;

    // Porcentaje igual que en el ESP32
    let pct;
    if (ALTURA_MAX_COMIDA > 0) {
      pct = (alturaCm / ALTURA_MAX_COMIDA) * 100;
    } else {
      pct = 0;
    }

    // Ajustes extremos como en el ESP32:
    // - si está en rango "lleno", forzamos 100%
    // - si está en rango "vacío", forzamos 0%
    if (distCm <= 12) {
      pct = 100;
    } else if (distCm > 39) {
      pct = 0;
    }

    pct = Math.max(0, Math.min(100, pct));

    // Actualizamos UI
    pctComidaElem.textContent = pct.toFixed(0);
    fillComidaElem.style.height = pct.toFixed(0) + "%";


    let mensaje;
    if (distCm <= 12) {
      mensaje = "✅ Nivel: LLENO";
    } else if (distCm <= 22) {
      mensaje = "🟡 Nivel: MEDIO";
    } else if (distCm <= 31) {
      mensaje = "⚠️ Nivel: BAJO";
    } else if (distCm <= 39) {
      mensaje = "❗ Nivel: CRÍTICO";
    } else {
      mensaje = "⛔ Nivel: VACÍO";
    }

    statusComidaElem.textContent = mensaje;
  });

  statusAguaElem.textContent = "Conectando...";
  refAgua.on("value", (snapshot) => {
    if (!snapshot.exists()) {
```

```
      statusAguaElem.textContent = "Dato no disponible";
      distAguaElem.textCntent = "--";
      pctAguaElem.textContent = "--";
      fillAguaElem.style.height = "0%";
      return;
    }

    // nivel real en cm
    const nivelRealCm = Number(snapshot.val()) || 0;

    // Mostramos la altura real tal cual
    distAguaElem.textContent = nivelRealCm.toFixed(0);

    // Porcentaje según la misma función que el firmware
    const pctTanque = porcentajeTanqueDesdeNivel(nivelRealCm);
    pctAguaElem.textContent = pctTanque.toFixed(0);
    fillAguaElem.style.height = pctTanque + "%";

    // bandas (lleno/medio/bajo/critico/vacío)
    statusAguaElem.textContent = mensajeTanqueUI(nivelRealCm);
  });

  // HUEVOS
  statusHuevosElem.textContent = "Conectando...";
  refHuevos.on("value", (snapshot) => {
    if (!snapshot.exists()) {
      eggCountElem.textContent = "0";
      barraHuevosElem.style.width = "0%";
      statusHuevosElem.textContent = "Sin datos";
      return;
    }

    const count = Number(snapshot.val()) || 0;
    eggCountElem.textContent = count;

    const pct = Math.max(0, Math.min(100, (count / 60) * 100));
    barraHuevosElem.style.width = pct + "%";

    if (pct >= 100) statusHuevosElem.textContent = "¡Recolección completa!";
    else if (pct >= 50) statusHuevosElem.textContent = "Producción en curso";
    else if (pct > 0) statusHuevosElem.textContent = "Inicio de producción";
    else statusHuevosElem.textContent = "Esperando huevos";
  });

  // ÚLTIMA VEZ 60 HUEVOS
  refUltima60.on("value", (snapshot) => {
    if (!snapshot.exists()) {
      hora60Elem.textContent =
        "Última recolección de ciclo (60 huevos): --";
      return;
    }

    const data = snapshot.val() || {};
    const ts = typeof data === "string" ? data : data.timestamp || "--";
    hora60Elem.textContent =
      "Última recolección de ciclo (60 huevos): " + ts;
  });

  /*****************  RESUMEN MENSUAL  *****************/
  function construirResumenMes(diariaData, historialData, year, month) {
    const lastDay = new Date(year, month + 1, 0).getDate();
    const dias = [];

    for (let d = 1; d <= lastDay; d++) {
      const yyyy = year;
      const mm = String(month + 1).padStart(2, "0");
      const dd = String(d).padStart(2, "0");
      const key = yyyy + "-" + mm + "-" + dd;
```

```
      const infoRaw = diariaData && diariaData[key] ? diariaData[key] : null;
      let huevosDia = 0;
      if (infoRaw) {
        huevosDia =
          Number(
            infoRaw.hueos_dia ?? infoRaw.total_dia ?? infoRaw.huevos ?? 0
          ) || 0;
      }

      dias.push({
        day: d,
        date: new Date(year, month, d),
        key,
        huevos: huevosDia,
        ciclosDia: 0,
        fechasCiclos: [],
      });
    }

    const mapaDias = new Map();
    dias.forEach((dia, idx) => mapaDias.set(dia.key, idx));

    const listaHist = historialData ? Object.values(historialData) : [];
    listaHist.forEach((r) => {
      const d = parseTs(r.timestamp || r.ts || "");
      if (!d) return;

      if (d.getFullYear() !== year || d.getMonth() !== month) return;

      const yyyy = d.getFullYear();
      const mm = String(d.getMonth() + 1).padStart(2, "0");
      const dd = String(d.getDate()).padStart(2, "0");
      const key = yyyy + "-" + mm + "-" + dd;
      const idx = mapaDias.get(key);
      if (idx === undefined) return;

      dias[idx].ciclosDia += 1;
      dias[idx].fechasCiclos.push(dd + "/" + mm);
    });

    return { year, month, dias, lastDay };
  }

  function renderCalendarioMes({ year, month, dias }) {
    if (!calendarGrid) return;
    calendarGrid.innerHTML = "";

    const nombreMes = NOMBRES_MESES[month] || "";
    const labelMes =
      nombreMes.charAt(0).toUpperCase() + nombreMes.slice(1) + " " + year;
    if (calendarMonthEl) calendarMonthEl.textContent = labelMes;

    const firstDay = new Date(year, month, 1);
    const startWeekDay = firstDay.getDay();

    // huecos iniciales
    for (let i = 0; i < startWeekDay; i++) {
      const cell = document.createElement("div");
      cell.classList.add("calendar-day", "calendar-day--empty");
      calendarGrid.appendChild(cell);
    }

    dias.forEach((dia) => {
      const div = document.createElement("div");
      div.classList.add("calendar-day");

      if (dia.huevos === 0) div.classList.add("calendar-day--none");
      else if (dia.huevos <= 5) div.classList.add("calendar-day--low");
      else if (dia.huevos <= 15) div.classList.add("calendar-day--mid");
```

```
      else div.classList.add("calendar-day--high");

      const header = document.createElement("div");
      header.classList.add("calendar-day-header");

      const labelEl = document.createElement("span");
      labelEl.textContent = dia.day.toString().padStart(2, "0");
      header.appendChild(labelEl);

      if (dia.ciclosDia > 0) {
        const badge = document.createElement("span");
        badge.classList.add("calendar-cycle-badge");
        badge.textContent = dia.ciclosDia + "-60";
        header.appendChild(badge);
      }

      const eggsEl = document.createElement("div");
      eggsEl.classList.add("calendar-day-eggs");
      eggsEl.textContent = dia.huevos > 0 ? dia.huevos + " huevos" : "—";

      div.appendChild(header);
      div.appendChild(eggsEl);
      calendarGrid.appendChild(div);
    });

    const totalUsed = startWeekDay + dias.length;
    const resto = totalUsed % 7;
    if (resto !== 0) {
      const faltan = 7 - resto;
      for (let i = 0; i < faltan; i++) {
        const cell = document.createElement("div");
        cell.classList.add("calendar-day", "calendar-day--empty");
        calendarGrid.appendChild(cell);
      }
    }
  }

  function renderGraficoMes({ dias }) {
    if (!graficoCanvas) return;

    const labels = dias.map((d) => d.day.toString());
    const valoresDiarios = dias.map((d) => d.huevos);

    if (graficoMensual) graficoMensual.destroy();
    const ctx = graficoCanvas.getContext("2d");

    graficoMensual = new Chart(ctx, {
      type: "line",
      data: {
        labels,
        datasets: [
          {
            label: "Huevos por día",
            data: valoresDiarios,
            tension: 0.3,
            fill: true,
            borderColor: "#22d3ee",
            backgroundColor: "rgba(34,211,238,0.18)",
            pointRadius: 3,
            pointBackgroundColor: "#22d3ee",
            pointBorderColor: "#0f172a",
            pointBorderWidth: 1,
          },
        ],
      },
      options: {
        plugins: {
          legend: {
            labels: {
```

```
              color: "#e5e7eb",
              font: { size: 10 },
            },
          },
          tooltip: {
            callbacks: {
              label: (ctx) => `${ctx.parsed.y} huevos`,
            },
          },
        },
        scales: {
          x: {
            ticks: {
              color: "#cbd5f5",
              font: { size: 10 },
            },
            grid: { display: false },
            title: {
              display: true,
              text: "Día del mes",
              color: "#cbd5f5",
              font: { size: 11 },
            },
          },
          y: {
            ticks: {
              color: "#cbd5f5",
              font: { size: 10 },
              stepSize: 2,
            },
            grid: { color: "rgba(148,163,184,0.25)" },
            beginAtZero: true,
            max: 40,
            title: {
              display: true,
              text: "Número de huevos producidos",
              color: "#cbd5f5",
              font: { size: 11 },
            },
          },
        },
      },
    });
  }

  /*****************  TABLA RESUMEN SEMANAS  ******************/
  function renderTablaSemanas({ year, month, dias, lastDay }) {
    if (!historialTbody) return;
    historialTbody.innerHTML = "";

    if (!dias || dias.length === 0) {
      td.colSpan = 5;
      td.textContent = "Sin registros aún";
      tr.appendChild(td);
      historialTbody.appendChild(tr);
      return;
    }

    const semanas = [];
    const getWeekIndex = (day) => Math.floor((day - 1) / 7);
    const maxWeekIndex = getWeekIndex(lastDay);

    for (let w = 0; w <= maxWeekIndex; w++) {
      const startDay = w * 7 + 1;
      const endDay = Math.min((w + 1) * 7, lastDay);
      semanas[w] = {
        label: "Semana " + (w + 1),
        startDay,
        endDay,
```

```
        huevos: 0,
        ciclos: 0,
        fechas: [],
      };
    }

    dias.forEach((dia) => {
      const wi = getWeekIndex(dia.day);
      if (!semanas[wi]) return;
      semanas[wi].huevos += dia.huevos;
      semanas[wi].ciclos += dia.ciclosDia;
      if (dia.fechasCiclos.length > 0) {
        semanas[wi].fechas.push(...dia.fechasCiclos);
      }
    });

    let totalMesHuevos = 0;
    let totalMesCiclos = 0;

    semanas.forEach((sem) => {
      totalMesHuevos += sem.huevos;
      totalMesCiclos += sem.ciclos;

      const tr = document.createElement("tr");

      const tdSemana = document.createElement("td");
      tdSemana.textContent = sem.label;

      const tdRango = document.createElement("td");
      tdRango.textContent = "Días " + sem.startDay + "–" + sem.endDay;

      const tdHuevos = document.createElement("td");
      tdHuevos.textContent = sem.huevos;

      const tdCiclos = document.createElement("td");
      tdCiclos.textContent = sem.ciclos;

      const tdFechas = document.createElement("td");
      tdFechas.textContent =
        sem.fechas.length === 0 ? "—" : [...new Set(sem.fechas)].join(", ");

      tr.appendChild(tdSemana);
      tr.appendChild(tdRango);
      tr.appendChild(tdHuevos);
      tr.appendChild(tdCiclos);
      tr.appendChild(tdFechas);
      historialTbody.appendChild(tr);
    });

    const trTotal = document.createElement("tr");
    trTotal.classList.add("historial-summary-row");

    const tdLabel = document.createElement("td");
    tdLabel.textContent = "Totales del mes";

    const tdVacio = document.createElement("td");
    tdVacio.textContent = "";

    const tdHuevosTot = document.createElement("td");
    tdHuevosTot.textContent = totalMesHuevos;

    const tdCiclosTot = document.createElement("td");
    tdCiclosTot.textContent = totalMesCiclos;

    const tdFechasTot = document.createElement("td");
    tdFechasTot.textContent = "";

    trTotal.appendChild(tdLabel);
    trTotal.appendChild(tdVacio);
```

```
    trTotal.appendChild(tdFechasTot);
    historialTbody.appendChild(trTotal);
  }

  /*****************  ACTUALIZAR RESUMEN  ******************/
  function actualizarResumenMensual() {
    if (selectedYear == null || selectedMonth == null) return;

    const diaria = cacheProduccionDiaria || {};
    const historial = cacheHistorial60 || {};

    const resumen = construirResumenMes(
      diaria,
      historial,
      selectedYear,
      selectedMonth
    );

    const totalHuevosMes = resumen.dias.reduce((acc, d) => acc + d.huevos, 0);
    const totalCiclosMes = resumen.dias.reduce(
      (acc, d) => acc + d.ciclosDia,
      0
    );

    if (resumenHuevosMesEl)
      resumenHuevosMesEl.textContent = totalHuevosMes;
    if (resumenCiclosMesEl)
      resumenCiclosMesEl.textContent = totalCiclosMes;

    renderCalendarioMes(resumen);
    renderGraficoMes(resumen);
    renderTablaSemanas(resumen);
  }

  /*****************  AÑOS 2000-2100 (FIJOS)  ******************/
  function updateYearSelects() {
    const startYear = 2000;
    const endYear = 2100;
    const now = new Date();
    const currentYear = now.getFullYear();
    const years = [];
    for (let y = startYear; y <= endYear; y++) years.push(y);
    availableYears = years;

    // Historial (gráfico)
    if (historyYearSelect) {
      const prev = selectedYear;
      historyYearSelect.innerHTML = "";
      years.forEach((y) => {
        const opt = document.createElement("option");
        opt.value = String(y);
        opt.textContent = String(y);
        historyYearSelect.appendChild(opt);
      });

      let newSelected = prev;
      if (!newSelected || newSelected < startYear || newSelected > endYear) {
        newSelected = Math.min(Math.max(currentYear, startYear), endYear);
      }
      selectedYear = newSelected;
      historyYearSelect.value = String(newSelected);
    }

    // Año para día
    if (gestionAnioSelect) {
      const nowY = currentYear;
      gestionAnioSelect.innerHTML = "";
      years.forEach((y) => {
        const opt = document.createElement("option");
```

```
        opt.value = String(y);
        opt.textContent = String(y);
        if (y === nowY) opt.selected = true;
        gestionAnioSelect.appendChild(opt);
      });
    }

    // Año para borrado mensual
    if (gestionAnioMesSelect) {
      const nowY = currentYear;
      gestionAnioMesSelect.innerHTML = "";
      years.forEach((y) => {
        const opt = document.createElement("option");
        opt.value = String(y);
        opt.textContent = String(y);
        if (y === nowY) opt.selected = true;
        gestionAnioMesSelect.appendChild(opt);
      });
    }

    // Año para borrado anual
    if (gestionAnioAnualSelect) {
      const nowY = currentYear;
      gestionAnioAnualSelect.innerHTML = "";
      years.forEach((y) => {
        const opt = document.createElement("option");
        opt.value = String(y);
        opt.textContent = String(y);
        if (y === nowY) opt.selected = true;
        gestionAnioAnualSelect.appendChild(opt);
      });
    }
  }

  function recomputeAvailableYears() {
  }

  /*****************  PESTAÑAS DE MESES  *****************/
  function initMonthTabs() {
    if (!monthTabsContainer) return;
    monthTabsContainer.innerHTML = "";

    const now = new Date();
    const currentYear = now.getFullYear();
    const currentMonth = now.getMonth();

    if (selectedYear == null) selectedYear = currentYear;
    if (selectedMonth == null) selectedMonth = currentMonth;

    for (let m = 0; m < 12; m++) {
      const btn = document.createElement("button");
      btn.classList.add("month-tab");
      btn.dataset.month = String(m);
      let label = NOMBRES_MESES[m] || "";
      label = label.charAt(0).toUpperCase() + label.slice(1);

      if (m === currentMonth && selectedYear === currentYear) {
        label += " (actual)";
      }

      btn.textContent = label;
      btn.addEventListener("click", () => {
        selectedMonth = m;
        document
          .querySelectorAll(".month-tab")
          .forEach((b) => b.classList.remove("month-tab-active"));
        btn.classList.add("month-tab-active");
        actualizarResumenMensual();
      });
```

```
      monthTabsContainer.appendChild(btn);
    }

    const activeBtn = Array.from(
      document.querySelectorAll(".month-tab")
    ).find((b) => Number(b.dataset.month) === selectedMonth);
    if (activeBtn) activeBtn.classList.add("month-tab-active");

    actualizarResumenMensual();
  }

  /*****************  SELECTOR AÑO HISTÓRICO  ******************/
  function initYearSelect() {
    if (!historyYearSelect) return;
    historyYearSelect.addEventListener("change", () => {
      selectedYear = Number(historyYearSelect.value);
      actualizarResumenMensual();
    });
  }

  /*****************  GESTIÓN DE FECHA (BORRADO)  ******************/

  function updateGestionDias() {
    if (!gestionDiaSelect || !gestionMesSelect) return;
    const now = new Date();
    const yearSel =
      gestionAnioSelect && gestionAnioSelect.value
        ? Number(gestionAnioSelect.value)
        : now.getFullYear();
    const monthIndex =
      gestionMesSelect && gestionMesSelect.value !== ""
        ? Number(gestionMesSelect.value)
        : now.getMonth();

    const daysInMonth = new Date(yearSel, monthIndex + 1, 0).getDate();
    const currentSelected = Number(gestionDiaSelect.value) || 1;

    gestionDiaSelect.innerHTML = "";
    for (let d = 1; d <= daysInMonth; d++) {
      const opt = document.createElement("option");
      opt.value = String(d);
      opt.textContent = d.toString().padStart(2, "0");
      if (d === Math.min(currentSelected, daysInMonth)) opt.selected = true;
      gestionDiaSelect.appendChild(opt);
    }
  }

  function initGestionFecha() {
    const now = new Date();
    const currentMonth = now.getMonth();

    // Mes para día
    if (gestionMesSelect) {
      gestionMesSelect.innerHTML = "";
      NOMBRES_MESES.forEach((nombre, idx) => {
        const opt = document.createElement("option");
        opt.value = String(idx);
        opt.textContent =
          nombre.charAt(0).toUpperCase() + nombre.slice(1);
        if (idx === currentMonth) opt.selected = true;
        gestionMesSelect.appendChild(opt);
      });
      gestionMesSelect.addEventLiner("change", updateGestionDias);
    }

    // Mes para borrado mensual
    if (gestionMesMesSelect) {
      gestionMesMesSelect.innerHTML = "";
```

```
      NOMBRES_MESES.forEach((nombre, idx) => {
        const opt = document.createElement("option");
        opt.value = String(idx);
        opt.textContent =
          nombre.charAt(0).toUpperCase() + nombre.slice(1);
        if (idx === currentMonth) opt.selected = true;
        gestionMesMesSelect.appendChild(opt);
      });
    }

    if (gestionAnioSelect) {
      gestionAnioSelect.addEventListener("change", updateGestionDias);
    }

    updateGestionDias();
  }

  /************ BORRADO POR DÍA ************/
  function borrarProduccionDiaSeleccionada() {
    if (!gestionMesSelect || !gestionDiaSelect) return;
    const fallbackYear = new Date().getFullYear();
    const yearSel =
      gestionAnioSelect && gestionAnioSelect.value
        ? Number(gestionAnioSelect.value)
        : fallbackYear;
    const year = yearSel;
    const monthIndex = Number(gestionMesSelect.value);
    const day = Number(gestionDiaSelect.value);

    const mm = String(monthIndex + 1).padStart(2, "0");
    const dd = String(day).padStart(2, "0");
    const key = year + "-" + mm + "-" + dd;

    refProduccionDiaria
      .child(key)
      .remove()
      .then(() => {
        if (gestionStatusEl) {
          gestionStatusEl.textContent =
            "Producción diaria del " +
            dd +
            "/" +
            mm +
            "/" +
            year +
            " borrada (si existía).";
        }
      })
      .catch((err) => {
        if (gestionStatusEl) {
          gestionStatusEl.textContent =
            "Error al borrar producción diaria: " + err.message;
        }
      });
  }

  function borrarCiclosDiaSeleccionada() {
    if (!gestionMesSelect || !gestionDiaSelect) return;
    const fallbackYear = new Date().getFullYear();
    const yearSel =
      gestionAnioSelect && gestionAnioSelect.value
        ? Number(gestionAnioSelect.value)
        : fallbackYear;
    const year = yearSel;
    const monthIndex = Number(gestionMesSelect.value);
    const day = Number(gestionDiaSelect.value);

    refHistorial60.once("value", (snapshot) => {
      if (!snapshot.exists()) {
```

```
        if (gestionStatusEl) {
          gestionStatusEl.textContent = "No hay registros en historial_60.";
        }
        return;
      }

      const promises = [];
      let cantidad = 0;

      snapshot.forEach((child) => {
        const data = child.val() || {};
        const ts = data.timestamp || data.ts || "";
        const d = parseTs(ts);
        if (!d) return;

        if (
          d.getFullYear() === year &&
          d.getMonth() === monthIndex &&
          d.getDate() === day
        ) {
          cantidad++;
          promises.push(child.ref.remove());
        }
      });

      if (cantidad === 0) {
        if (gestionStatusEl) {
          gestionStatusEl.textContent =
            "No se encontraron ciclos de 60 huevos en ese día.";
        }
        return;
      }

      Promise.all(promises)
        .then(() => {
          if (gestionStatusEl) {
            const mm = String(monthIndex + 1).padStart(2, "0");
            const dd = String(day).pad
              "Se borraron " +
              cantidad +
              " registro(s) de ciclos de 60 huevos del " +
              dd +
              "/" +
              mm +
              "/" +
              year +
              ".";
          }
        })
        .catch((err) => {
          if (gestionStatusEl) {
            gestionStatusEl.textContent =
              "Error al borrar ciclos de 60 huevos: " + err.message;
          }
        });
    });
  }

  /************ BORRADO POR MES ************/
  function borrarProduccionMesSeleccionado() {
    if (!gestionMesMesSelect) return;
    const fallbackYear = new Date().getFullYear();
    const yearSel =
      gestionAnioMesSelect && gestionAnioMesSelect.value
        ? Number(gestionAnioMesSelect.value)
        : fallbackYear;
    const year = yearSel;
    const monthIndex = Number(gestionMesMesSelect.value);
    const mm = String(monthIndex + 1).padStart(2, "0");
```

```
    const startKey = year + "-" + mm + "-01";
    const endKey = year + "-" + mm + "-31";

    refProduccionDiaria
      .orderByKey()
      .startAt(startKey)
      .endAt(endKey)
      .once("value", (snapshot) => {
        if (!snapshot.exists()) {
          if (gestionStatusEl) {
            gestionStatusEl.textContent =
              "No se encontraron producciones diarias en " +
              mm +
              "/" +
              year +
              ".";
          }
          return;
        }

        const promises = [];
        let count = 0;

        snapshot.forEach((child) => {
          count++;
          promises.push(child.ref.remove());
        });

        Promise.all(promises)
          .then(() => {
            if (gestionStatusEl) {
              gestionStatusEl.textContent =
                "Se borraron " +
                count +
                " registro(s) de producción diaria del mes " +
                mm +
                "/" +
                year +
                ".";
            }
          })
          .catch((err) => {
            if (gestionStatusEl) {
              gestionStatusEl.textContent =
                "Error al borrar producción diaria del mes: " +
                err.message;
            }
          });
      });
  }

  function borrarCiclosMesSeleccionado() {
    if (!gestionMesMesSelect) return;
    const fallbackYear = new Date().getFullYear();
    const yearSel =
      gestionAnioMesSelect && gestionAnioMesSelect.value
        ? Number(gestionAnioMesSelect.value)
        : fallbackYear;
    const year = yearSel;
    const monthIndex = Number(gestionMesMesSelect.value);

    refHistorial60.once("value", (snapshot) => {
      if (!snapshot.exists()) {
        if (gestionStatusEl {
          gestionStatusEl.textContent = "No hay registros en historial_60.";
        }
        return;
      }
```

```
      const promises = [];
      let count = 0;

      snapshot.forEach((child) => {
        const data = child.val() || {};
        const ts = data.timestamp || data.ts || "";
        const d = parseTs(ts);
        if (!d) return;

        if (d.getFullYear() === year && d.getMonth() === monthIndex) {
          count++;
          promises.push(child.ref.remove());
        }
      });

      if (count === 0) {
        if (gestionStatusEl) {
          const mm = String(monthIndex + 1).padStart(2, "0");
          gestionStatusEl.textContent =
            "No se encontraron ciclos de 60 huevos en el mes " +
            mm +
            "/" +
            year +
            ".";
        }
        return;
      }

      Promise.all(promises)
        .then(() => {
          if (gestionStatusEl) {
            const mm = String(monthIndex + 1).padStart(2, "0");
            gestionStatusEl.textContent =
              "Se borraron " +
              count +
              " registro(s) de ciclos de 60 huevos del mes " +
              mm +
              "/" +
              year +
              ".";
          }
        })
        .catch((err) => {
          if (gestionStatusEl) {
            gestionStatusEl.textContent =
              "Error al borrar ciclos de 60 huevos del mes: " +
              err.message;
          }
        });
    });
  }

  /************ BORRADO POR AÑO ************/
  function borrarProduccionAnioSeleccionado() {
    const fallbackYear = new Date().getFullYear();
    const yearSel =
      gestionAnioAnualSelect && gestionAnioAnualSelect.value
        ? Number(gestionAnioAnualSelect.value)
        : fallbackYear;
    const year = yearSel;

    const startKey = year + "-01-01";
    const endKey = year + "-12-31";

    refProduccionDiaria
      .orderByKey()
      .startAt(startKey)
      .endAt(endKe+y)
```

```
      .once("value", (snapshot) => {
        if (!snapshot.exists()) {
          if (gestionStatusEl) {
            gestionStatusEl.textContent =
              "No se encontraron producciones diarias para el año " +
              year +
              ".";
          }
          return;
        }

        const promises = [];
        let count = 0;

        snapshot.forEach((child) => {
          count++;
          promises.push(child.ref.remove());
        });

        Promise.all(promises)
          .then(() => {
            if (gestionStatusEl) {
              gestionStatusEl.textContent =
                "Se borraron " +
                count +
                " registro(s) de producción diaria del año " +
                year +
                ".";
            }
          })
          .catch((err) => {
            if (gestionStatusEl) {
              gestionStatusEl.textContent =
                "Error al borrar producción diaria del año: " +
                err.message;
            }
          });
      });
  }

  function borrarCiclosAnioSeleccionado() {
    const fallbackYear = new Date().getFullYear();
    const yearSel =
      gestionAnioAnualSelect && gestionAnioAnualSelect.value
        ? Number(gestionAnioAnualSelect.value)
        : fallbackYear;
    const year = yearSel;

    refHistorial60.once("value", (snapshot) => {
      if (!snapshot.exists()) {
        if (gestionStatusEl) {
          gestionStatusEl.textContent = "No hay registros en historial_60.";
        }
        return;
      }

      const promises = [];
      let count = 0;

      snapshot.forEach((child) => {
        const data = child.val() || {};
        const ts = data.timestamp || data.ts || "";
        const d = parseTs(ts);
        if (!d) return;

        if (d.getFullYear() === year) {
          count++;
          promises.push(child.ref.remove());
        }
```

```
      });

      if (count === 0) {
        if (gestionStatusEl) {
          gestionStatusEl.textContent =
            "No se encontraron ciclos de 60 huevos para el año " +
            year +
            ".";
        }
        return;
      }

      Promise.all(promises)
        .then(() => {
          if (gestionStatusEl) {
            gestionStatusEl.textContent =
              "Se borraron " +
              count +
              " registro(s) de ciclos de 60 huevos del año " +
              year +
              ".";
          }
        })
        .catch((err) => {
          if (gestionStatusEl) {
            gestionStatusEl.textContent =
              "Error al borrar ciclos de 60 huevos del año: " +
              err.message;
          }
        });
    });
  }

  /*****************  ESCUCHAS FIREBASE  ******************/
  refProduccionDiaria.on("value", (snapshot) => {
    cacheProduccionDiaria = snapshot.val() || {};
    actualizarResumenMensual();
  });

  refHistorial60.on("value", (snapshot) => {
    cacheHistorial60 = snapshot.val() || {};
    actualizarResumenMensual();
  });

  /*****************  DESPLEGAR / OCULTAR GESTIÓN DE DATOS  ******************/
  if (toggleGestionBtn && gestionWrapper) {

    toggleGestionBtn.addEventListener("click", () => {
      gestionWrapper.classList.toggle("manage-wrapper-collapsed");
      toggleGestionBtn.classList.toggle("gestion-open");
    });
  }

  /*****************  MODAL DE CONFIRMACIÓN  ******************/
  let pendingConfirmAction = null;

  function openConfirmModal(title, message, onConfirm) {
    if (!confirmModal) {
      if (typeof onConfirm === "function") onConfirm();
      return;
    }
    confirmTitleEl.textContent = title || "Confirmar acción";
    confirmMessageEl.textContent =
      message ||
      "¿Estás seguro de que deseas realizar esta acción? Esta operación no se puede deshacer.";
    pendingConfirmAction = onConfirm;
    confirmModal.classList.remove("confirm-hidden");
  }

```

```
function closeConfirmModal() {
  if (!confirmModal) return;
  confirmModal.classList.add("confirm-hidden");
  pendingConfirmAction = null;
}

if (confirmCancelBtn) {
  confirmCancelBtn.addEventListener("click", () => {
    closeConfirmModal();
  });
}

if (confirmOkBtn) {
  confirmOkBtn.addEventListener("click", () => {
    const fn = pendingConfirmAction;
    closeConfirmModal();
    if (typeof fn === "function") {
      fn();
    }
  });
}

if (confirmModal) {
  confirmModal.addEventListener("click", (e) => {
    if (e.target === confirmModal) {
      closeConfirmModal();
    }
  });
}

function setupConfirm(button, title, message, actionFn) {
  if (!button) return;
  button.addEventListener("click", () => {
    openConfirmModal(title, message, actionFn);
  });
}

/*****************  INICIALIZACIÓN  *****************/
updateYearSelects();
initYearSelect();
initMonthTabs();
initGestionFecha();

// Botones con confirmación
setupConfirm(
  btnBorrarProdDia,
  "Confirmar borrado de producción diaria",
  "¿Estás seguro de que deseas borrar la producción diaria del día seleccionado? Esta acción no se puede deshacer.",
  borrarProduccionDiaSeleccionada
);

setupConfrm(
  btnBorrarCiclosDia,
  "Confirmar borrado de ciclos (día)",
setupConfirm(
  btnBorrarCiclosAnio,
  "Confirmar borrado anual de ciclos",
  "¿Estás seguro de que deseas borrar todos los ciclos de 60 huevos del año seleccionado? Esta acción no se puede deshacer.",
  borrarCiclosAnioSeleccionado
);
```