UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

"SUPERVISÓN REMOTA DE CULTIVOS HIDROPÓNICOS EMPLEANDO INTERNET DE LAS COSAS'

AUTOR: BRYAN STEVEN CAÑARTE DELGADO

DIRECTOR: CARLOS XAVIER ROSERO

IBARRA-ECUADOR
MAYO 2021

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:	131460829-8		
APELLIDOS Y NOMBRES:	Cañarte Delgado Bryan Steven		
DIRECCIÓN:	El Olivo, Ibarra		
EMAIL:	bscanarted@utn.edu.ec		
TELÉFONO FIJO:	05-260-3798	**TELÉFONO MÓVIL:**	0988993941

DATOS DE LA OBRA	
TÍTULO:	Supervisión de cultivos hidropónicos empleando internet de las cosas
AUTOR (ES):	Cañarte Delgado Bryan Steven
FECHA: DD/MM/AAAA	06/07/2021
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	☑ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
ASESOR /DIRECTOR:	**PHD. Carlos Xavier Rosero Chandi**

2. CONSTANCIAS

El autor (es) manifiesta (n) 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 (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 6 días del mes de julio de 2021

EL AUTOR:

(Firma).....
Nombre: Bryan Steven Cañarte Delgado

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado “SUPERVISIÓN REMOTA DE CULTIVOS HIDROPÓNICOS EMPLEANDO INTERNET DE LAS COSAS”, presentado por el egresado Bryan Steven Cañarte Delgado, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, 06 de julio de 2021

Firmado electrónicamente por:
CARLOS XAVIER ROSERO CHANDI

Carlos Xavier Rosero
DIRECTOR DE TESIS

Agradecimiento

Agradezco a mis padres por acompañarme durante toda mi carrera y en mi día a día. A la universidad Técnica del Norte por abrirme las puertas al conocimiento y contar con docentes que con su dedicación y conocimiento fueron parte de mi formación profesional.

Agradezco a todos los que fueron mis compañeros de clase y a los amigos que formé durante mi transcurso en la Universidad por todas las experiencias vividas e influencias positivas que me aportaron tanto en lo personal como en lo académico.

Agradecimientos especiales a mi hermano Andy por ayudarme en el cuidado del cultivo hidropónico y a mi padre por su construcción.

Y por último a todas las personas quienes de alguna manera pusieron su parte con la culminación de este proyecto.

Bryan Steven Cañarte Delgado

Dedicatoria

Dedico con todo mi corazón mi tesis a toda mi familia. Principalmente a mis padres que han sido un pilar fundamental en mi vida, por estar conmigo y brindarme su apoyo, sus consejos y su amor. A mi tía, mis tíos y mis abuelitos que desde pequeño me cuidaron, me guiaron y acompañaron. A mis hermanos y a todos mis seres queridos que siempre han procurado por mi bienestar.

Bryan Steven Cañarte Delgado

Resumen

Este trabajo tiene el propósito de presentar el desarrollo de un sistema de supervisión hidropónica capaz de monitorear la temperatura ambiente, temperatura de solución nutritiva, el nivel de solución, el pH y la conductividad eléctrica (CE), además de realizar funciones de riego, regulación del pH, configuración de ciclos de riego y calibración de sondas. El sistema permite su control y configuración a través de una aplicación instalada en un dispositivo Smartphone compatible con la plataforma Android y utiliza una base de datos online creada con el servicio Real Time DataBase de Firebase como medio de intercambio y almacenamiento de información. El diseño del propuesto implementó funciones especiales que cambian el comportamiento del sistema de acuerdo a la información recolectada por los sensores, funciones de seguridad, generación de registro histórico y funciones de registro y vinculación de dispositivos que permiten integrar dentro de la base de datos más de un sistema de supervisión. Se realiza un análisis al funcionamiento del sistema en base a objetivos establecidos y se presenta los resultados obtenidos.

Índice general

Página

Índice de figuras

Índice de cuadros

Introducción

Problema

En el Ecuador, la agricultura es uno de los ejes principales de la economía [1], a finales del 2018 representó un 8 % del Producto Interno Bruto (PIB) [2] y durante los últimos años se ha trabajado por incrementar la productividad y así cumplir con la demanda interna y externa. Solo en Imbabura se cuenta con 283,659 hectáreas destinadas al cultivo, según el informe entregado por el Instituto Nacional de Estadística y Censo (INEC) en el 2018 [3].

Entre los proyectos que se han llevado a cabo para cumplir con la demanda se encuentra el uso de cultivos hidropónicos. Su aceptación en el mercado ha sido tal que en las últimas décadas la venta de productos provenientes de cultivos hidropónicos ha ido incrementando significativamente [4], tomando impulso en provincias de todo el país, siendo un ejemplo de esto, Cuenca [5].

Sin embargo, existen limitaciones que dificultan a la población a incursionar en el cultivo de roductos hidropónicos, muchas de las cuales son referentes a los cuidados que se deben considerar para mantener el equilibrio de los parámetros físicos y biológicos de la planta. Por lo cual, el control sobre estos parámetros debe ser bastante preciso y debe llevarse a cabo por personal calificado [6]. Además, debido al ritmo de vida rápido del sector urbano, la mayoría de las personas no tienen suficiente tiempo para dedicarse a los cuidados necesarios [4].

En vista de esta dificultad, se vuelve necesario el acceso rápido a información precisa que permita conocer la evolución del cultivo en cualquier momento, es decir, se vuelve necesario un sistema que permita monitorear aquellos parámetros fundamentales y que, además, permita acceder a estos datos desde cualquier lugar, permitiendo tomar decisiones rápidas y correcciones inmediatas a cualquier anomalía que se presente [6]. La naturaleza de este sistema lo vuelve un candidato perfecto para la aplicación del concepto de internet de las cosas [7].

La importancia de implementar un sistema capaz de llevar a cabo la supervisión remota aumenta si se considera los límites que tienen los sistemas actuales en cuanto a cantidad y calidad de los productos hidropónicos producidos en la zona urbana, ya que muchos de estos sistemas presentan problemas que afectan al crecimiento de las plantas [4]. Como resultado de no mejo-

rar estas tecnologías, el desarrollo de cultivos hidropónicos se verá estancado.

Objetivos

Objetivo General

Desarrollar un sistema para la supervisión remota de un cultivo hidropónico empleando internet de las cosas

Objetivos Específicos

- Identificar las variables físicas que determinen el óptimo crecimiento de un cultivo hidropónico.
- Desarrollar el software que se implementará sobre una plataforma de hardware libre para el intercambio de información con la nube.
- Validar el funcionamiento del dispositivo en un ambiente controlado.

Justificación

La motivación principal que impulsa el desarrollo de este trabajo es poder aplicar el conocimiento aprendido durante toda la carrera para resolver un problema real, aplicando ingeniería. En este caso, esta propuesta se enfoca a potenciar las cualidades que presentan los cultivos hidropónicos al aplicar internet de las cosas. Los cultivos hidropónicos requieren constante atención para garantizar la salud de la planta. Es por eso que el manejo de estos sistemas se debe realizar con un control preciso y con un personal que debe estar muy atento a cualquier anomalía [4]. Aprovechando las ventajas del internet de las cosas, y al uso de sensores conectados a la red se puede acceder rápida y remotamente a información que permita conocer el desarrollo del cultivo sin necesidad de ir al sitio donde este se encuentra. Esto permite al encargado tomar acciones más rápidas y precisas, y gracias a que este sistema se puede aplicar simultáneamente a una cantidad indefinida de cultivos, el número de plantas que se pueden cuidar aumenta. Por último, este proyecto puede ser empleado como etapa de adquisición de datos para el desarrollo de sistemas de control con propósito de optimización [8].

Alcance

El presente trabajo tiene por objetivo el desarrollar un sistema de supervisión remoto capaz de ser implementado en cultivos hidropónicos. El sistema será capaz de tomar datos del cultivo como son temperatura, PH, nivel de agua por medio de sensores, analizarlos y posteriormente enviarlos a una nube en donde se almacenarán hasta que sean solicitados. Los datos en la nube poseerán información de la evolución histórica del sistema y podrán ser empleados para su optimización. Se contará con la posibilidad de actuar de forma remota sobre salidas físicas ubicadas en el sistema permitiendo al usuario tomar acciones sobre su cultivo hidropónico en base a su criterio y a la información almacenada en la nube. Para este fin se desarrollará una aplicación móvil la cual servirá como interfaz para que el usuario realice las acciones antes descritas.

Capítulo 1

Revisión Literaria

1.1. Hidroponía

La hidroponía es una ténica que permite cultivar y producir plantas sin emplear el suelo o la tierra. [13]. Hace uso de estructuras las cuales pueden ser simples o complejas para producir plantas en sitios o áreas como son azoteas, suelos infértiles, invernaderos entre otros. Se basa en el concepto de un sustrato hídrico como medio para sustituir al suelo sin perder de vista las necesidades de la planta, tales como la temperatura, humedad, agua y nutrientes [21].

1.2. Técnica de cultivo hidropónico

Las técnicas empleadas en los cultivos hidropónicos son varias, desarrolladas para adaptarse a distintos casos de hidroponía, y difieren generalmente en la manera en la que las plantas entran en contacto con la solución nutritiva.

- **NFT (técnica de película nutritiva)**, que consiste en hacer circular el agua por tubos de PVC con bombas de agua y con el uso de sistemas automatizados.
- **Raíz flotante**, utilizan contenedores de diversos materiales y las plantas quedan por encima del agua flotando, mientras que las raíces están sumergidas en el agua. En este caso la oxigenación del agua se hace de forma manual.
- **Sustrato sólido** o, la raíz de la planta se sostiene con el sustrato, que aporta humedad y drena el agua que aporta la solución nutritiva.
- **Aeroponía**, técnica en la que las raíces de la planta quedan suspendidas en el aire, y son pulverizadas constantemente con el agua que contiene los nutrientes, de manera que tiene tanto los nutrientes que necesita como oxígeno.

1.3. Técnica de cultivo NFT

En la técnica NFT las plantas se disponen en orificios distribuidos a lo largo de una tubería perforada a través de la cual se hace circular internamente una solución nutritiva de modo que una pequeña lámina de agua cubra las raíces, permitiendo así la absorción de nutrientes disueltos. La solución circula en un sistema cerrado desde un depósito de almacenamiento e impulsada por una bomba. Este método permite el ingreso de oxigeno por medio de la mezcla del agua en movimiento con el aire circundante y a través de partes de las raíces no cubiertas por la lamina de agua [21].

Figura 1.1: Cultivo NFT.

1.3.1. Ventajas de la técnica de cultivo NFT

El sistema NFT es una de las técnicas más extendidas para la producción comercial de hortalizas a nivel mundial, contando con una importante carga de automatización y diversos diseños disponibles. Entre las ventajas que esta técnica ofrece están:

- Mayor control sobre la solución y los nutrientes
- Mejor eficiencia en el uso de los productos de control de la solución
- Las cosechas se consiguen en un ciclo menor al habitual
- Simplicidad en la implementación de sistemas de operación automatizados
- Un mejor aprovechamiento de espacio

1.4. Diferencias entre cultivo en suelo y el cultivo hidropónico

El suelo cumple un rol importante en el desarrollo de una planta. Posee una capacidad amortiguadora que le brinda estabilidad a las raíces y mantiene todos los parámetros físicos y biológicos en equilibrio. Esto sucede también en los cultivos hidropónicos pero de manera limitada. En el caso del suelo, una combinación inadecuada en los nutrientes o un pH erróneo es corregida gracias a la prescencia de microorganismos y a la propia composición química del suelo. Sin embargo en hidroponía los cambios suceden rápidamente y sin un control adecuado, se puede llegar a matar a las plantas [6].

1.4.1. Factores a tomar en cuenta en el cultivo hidropónico NFT

En hidroponía, la solución nutritiva toma el rol que cumple el suelo y se convierte en el medio por el cual la planta obtiene agua, oxígeno y nutrientes. Esto lleva a que se tengan que controlar determinados parámetros para garantizar el crecimiento adecuado de las plantas. Además, a esto se le debe agregar los parámetros propios de la técnica NFT. Todos los factores a tomar en cuenta se pueden ver en la Fig.1.2.

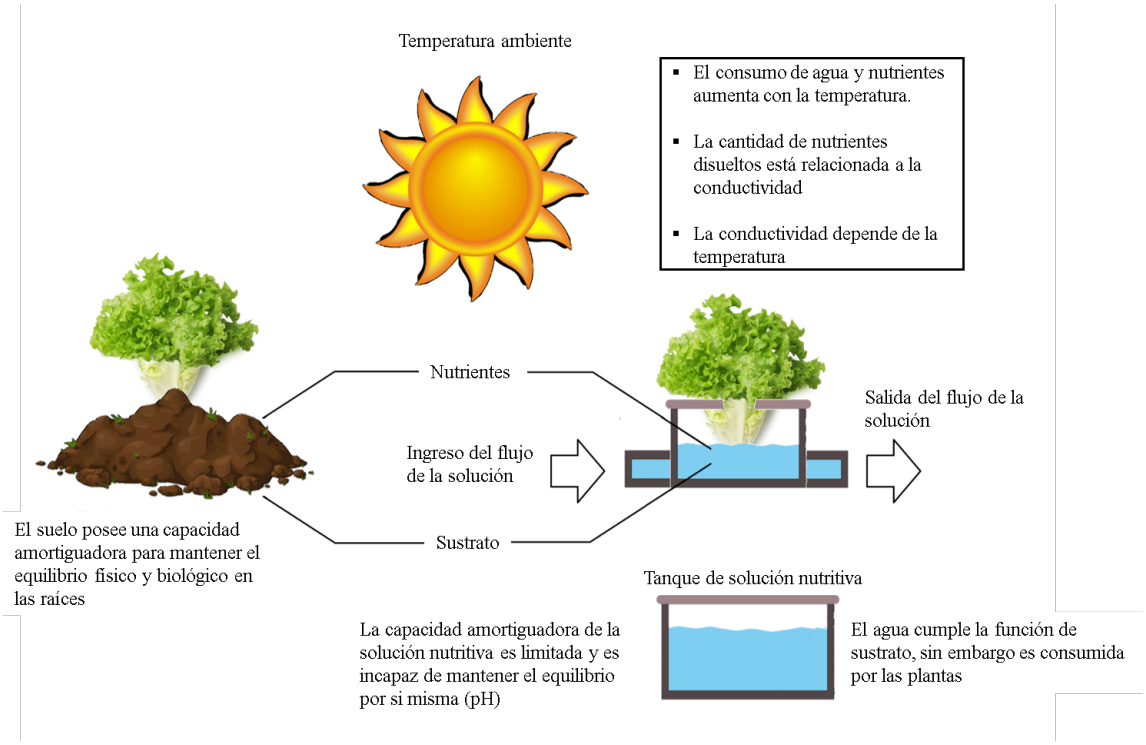


Figura 1.2: Comparativa entre el cultivo en suelo y el cultivo hidropónico.

1.5. Características del cultivo hidropónico de lechuga

El cultivo hidropónico de lechuga es uno de los más populares. Su creciente demanda y sus altos rendimientos lo hacen ideal para crecer en invernaderos y entornos controlados con gran éxito. La lechuga es un cultivo versátil y casi todas sus variedades son adaptables a las distintas técnicas hidropónicas, creciendo muy rápidamente y llegando a ser comestibles en 3 semanas. [20].
Por otro lado, para garantizar el óptimo crecimiento de las lechugas, se requiere mantener un cuidado especial sobre los parámetros que intervienen en su desarrollo. Las condiciones y características que requiere el cultivo hidropónico de lechuga se pueden observar en la tabla 1.1.

Cuadro 1.1: Características del cultivo hidropónico de lechuga

Característica	**Descripción**
Temperatura ambiente	Las temperaturas diurnas deben oscilar entre 20-23 ° C , sin exceder nunca los 25 ° C.
Temperatura de la solución	La temperatura de la solución ideal para las plantas cultivadas en hidroponía está en el rango de 18ºC a 30º C en verano y de 10ºC a 16ºC en invierno.
Conductividad eléctrica	El mejor valor de CE para el cultivo de lechuga es de aproximadamente 1,2 mS / cm.
pH	La lechuga hidropónica crece mejor en un medio que tiene un valor de pH de 5.5-6.0.
Iluminación	La lechuga no es exigente cuando se trata de iluminación. De diez a catorce horas de luz moderada o incluso con poca luz es suficiente

1.6. Internet de las cosas en la agricultura

La agricultura continúa siendo una de las actividades económicas más importantes en un gran número de países y ha logrado expandirse significativamente gracias a los avances tecnológicos de las últimas décadas. Una de estas nuevas tecnologías es el Internet de las Cosas (IoT), el cual ha conseguido introducirse en todo tipo de sectores, siendo el agrícola uno en los que poco a poco se han ido obteniendo resultados exitosos [8]. Como resultado, las soluciones basadas en IoT han creado un amplio panorama de desarrollo de dispositivos y equipos que satisfagan las necesidades de monitoreo y control de las diversas etapas dentro de la campaña de cultivo, las cuales han sido implementadas y adoptadas con gran ímpetu en múltiples países desarrollados [14].

1.7. Arquitecturas de control en cultivos hidropónicos

Existen múltiples trabajos que incorporan diferentes soluciones al problema de la automatización en cultivos hidropónicos. El sistema más simple consiste en el uso de una bomba temporizada que realiza el riego a intervalos pre-programados [15].
Por otro lado, existen sistemas automatizados los cuales integran cualidades de monitoreo y control que mejoran la precisión en el proceso de riego y ajuste de parámetros de la solución como son el pH y la conductividad, permitiendo incrementar la calidad y los resultados de los cultivos [13].

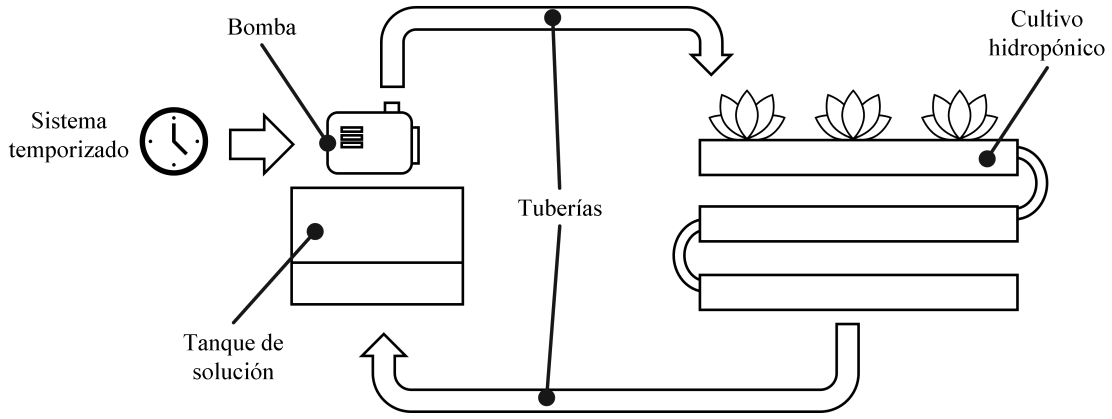


Figura 1.3: Sistema con temporizador para el control de riego

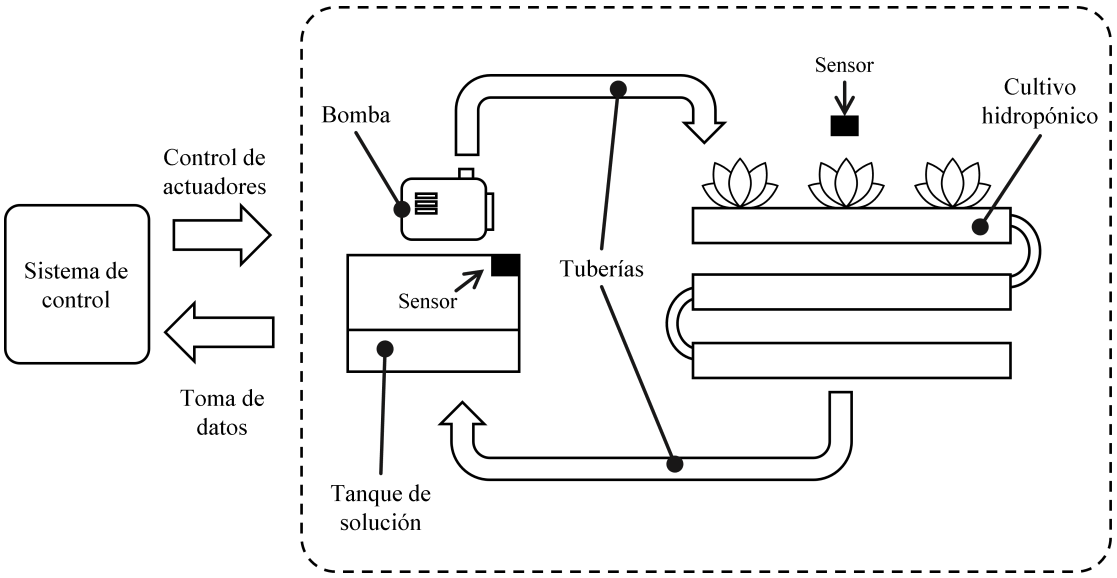


Figura 1.4: Sistema centralizado con monitoreo y control del cultivo

Capítulo 2

Metodología

En este capítulo se presenta la metodología llevada a cabo para el desarrollo de este proyecto. Se detallan las principales consideraciones para el sistema, la lógica de funcionamiento y las características que posee el prototipo construido.

2.1. Desarrollo del proyecto

2.1.1. Principales consideraciones para el sistema

Para el cumplimiento de los objetivos detallados en el capítulo 1, se considero las siguientes premisas que debe cumplir el proyecto:

- Ser un sistema con la capacidad de tomar medidas y acciones sobre el cultivo hidropónico.
- Ser un sistema automático que requiera en lo menor posible intervención del usuario, delegando al operador a la función de supervisor.
- Ser un sistema adaptativo al cultivo, con la capacidad de modificar su comportamiento según la situación.
- Ser un sistema independiente a la conexión a la red, permitiendo que siga en funcionamiento en caso de fallo en el internet.
- Poseer un mecanismo de intercambio y almacenamiento de información por medio de una nube.
- Poseer funcionalidades de control manual que permitan al usuario tomar decisiones sobre el sistema.
- Poseer funciones de seguridad que garanticen la integridad del sistema.
- Poseer la capacidad de crear un registro histórico que pueda ser empleado para conocer el desarrollo de las variables del cultivo hidropónico.

- Poseer indicación visual local con la finalidad de señalar el estado en el que se encuentra el sistema.
- Ser un sistema de supervisión remota, al cual se podrá acceder por medio de una aplicación móvil.

2.2. Modelo conceptual del sistema

Para cumplir con las consideraciones señaladas en la sección anterior, fue necesario desarrollar un modelo para el sistema capaz de cumplir con los requerimientos del proyecto. Como requisitos básicos, este modelo debe permitir la adquisición de datos, control de actuadores, gestión del intercambio de información entre dispositivos remotos y el sistema, además del almacenamiento de datos. La Fig 2.1 indica los cuatro bloques que conforman el modelo empleado para este proyecto.

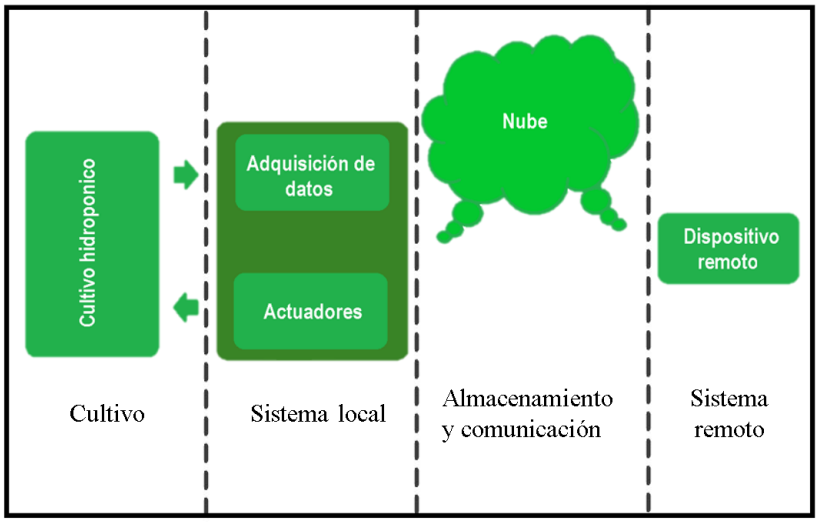


Figura 2.1: Modelo conceptual del sistema.

En este modelo se separa el proyecto en cuatro bloques claramente diferenciados y orientados a funciones especificas. El primer bloque, también llamado “Cultivo”, corresponde a la estructura que sirve de soporte a las plantas, los actuadores, sensores del cultivo hidropónico a supervisar. El siguiente bloque denominado “sistema local”, esta conformado por el hardware del sistema, incluyendo sensores y actuadores. El bloque “Almacenamiento y comunicación”tiene una función fundamental en el proyecto ya que a este se atribuye las funciones de manejo remoto y configuración del sistema local. Además, sirve de intermediario para la comunicación entre este y el dispositivo de supervisión remoto. Por último, el “Sistema remoto”, no es más que el punto de acceso otorgado al usuario por medio del cual este será capaz de visualizar la información del cultivo, configurar el sistema y tomar acciones de ser necesario.

2.3. Estructura física

La estructura física es la base sobre la cual se monta el cultivo hidropónico. En ella, se colocan las plantas, el sistema de riego, y el sistema local, dando un soporte firme y adecuado a todo el sistema. Se contemplo para su diseño la simplicidad para la construcción y desarme, una ubicación adecuada para los actuadores y sensores y la ubicación del propio sistema local. La Fig. 2.2 muestra el diseño preliminar realizado con el software de modelado 3D.

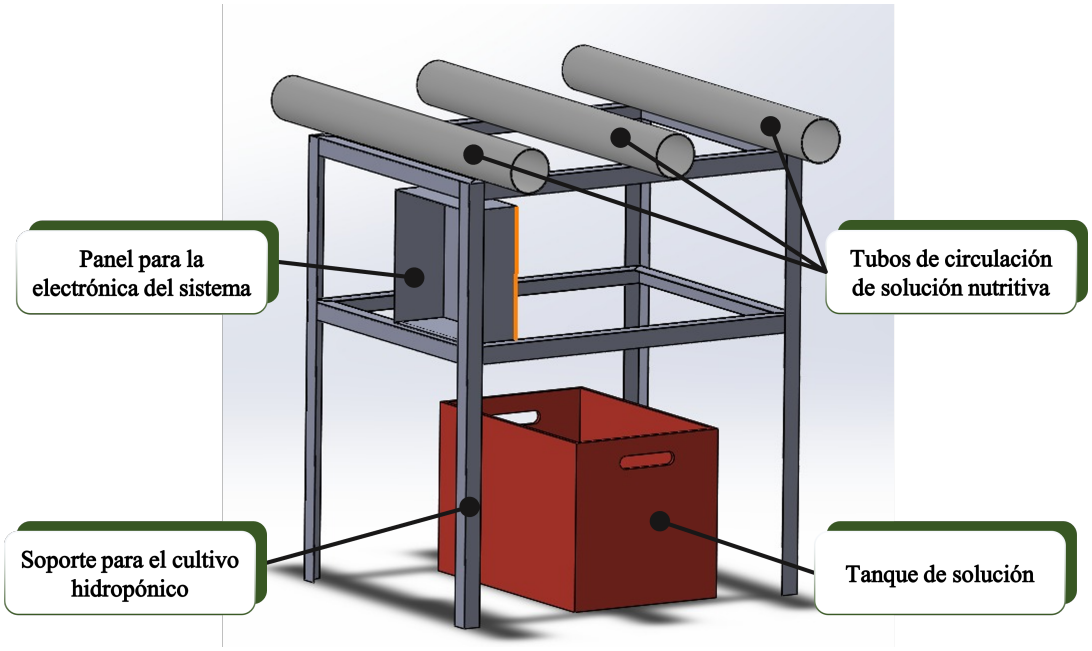


Figura 2.2: Diseño preliminar de la estructura del cultivo hidropónico.

2.3.1. Selección de los párametros de supervisión

En el cultivo convencional el suelo cumple un papel fundamental ya que es el encargado de mantener el equilibrio físico y biológico en las raíces, además de ser el medio por el cual se absorben nutrientes. En hidropónia, es la solución nutritiva la encargada de esta función. Sin embargo, es muy limitada en comparación al suelo, existiendo además nuevos puntos a tomar en cuenta. Para que la solución nutritiva sea capaz de permitir el desarrollo de la planta, está debe ser acondicionada a las necesidades especificas del cultivo. La Fig.2.3 muestra que factores en la solución están relacionados con parámetros físicos que se pueden medir, e incluye factores específicos para el cultivo en hidróponia.

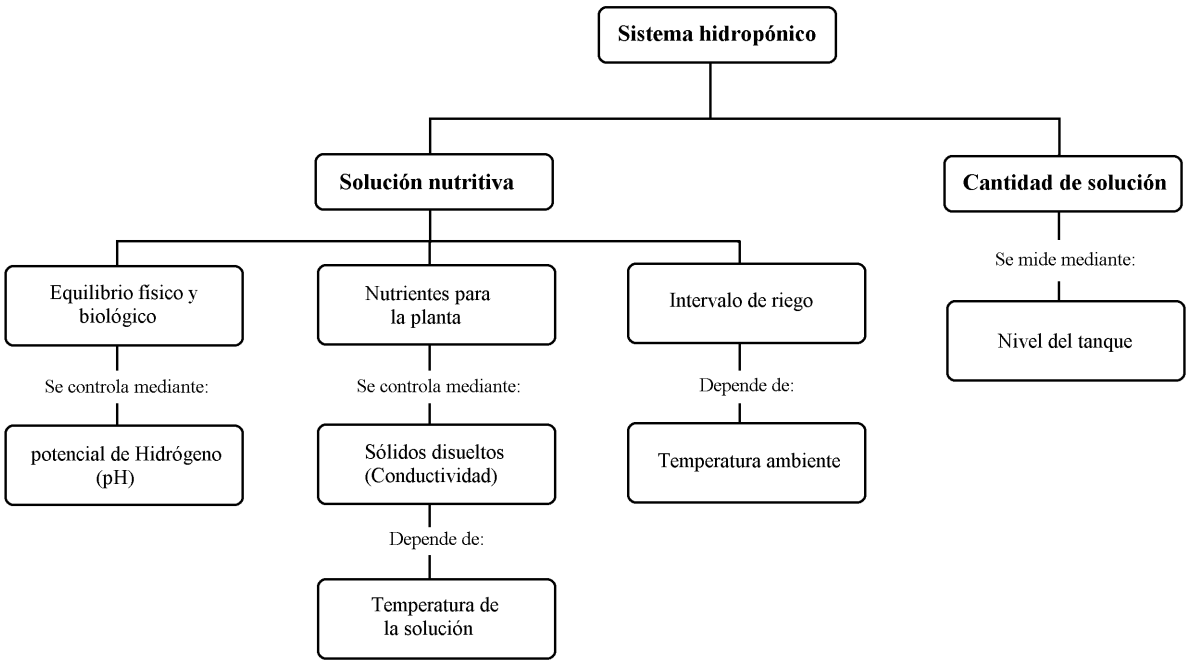


Figura 2.3: Mapa conceptual de los factores del cultivo hidropónico y parámetros relacionados.

2.3.2. Variables de medición

La selección de variables de medición se realizo en base a la Fig.2.3. La tabla 2.1 lista las variables seleccionadas y la función asignada para cada una de ellas.

Cuadro 2.1: Variables a medir y su función en el sistema

Variable	Función
Temperatura ambiente	Empleada para el control automático de los intervalos de riego.
Temperatura de la solución	Empleada para la compensación de temperatura de la medición de conductividad.
Nivel del tanque	Empleado para conocer de forma cuantitativa la cantidad de solución almacenada en el sistema
Potencial de hidrógeno	Cualidad de la solución que debe mantenerse alrededor de un valor determinado para el buen crecimiento de la planta.
Conductividad	Cualidad de la solución que señala de forma cualitativa la cantidad de nutrientes que posee la solución. Debe mantenerse alrededor de un valor determinado

2.3.3. Variables a controlar

Para este sistema los únicas variables a controlar corresponden a los actuadores. Estos son la bomba principal que se encarga del suministro de solución a las plantas y la bomba de dosificación de solución reguladora de pH.

2.4. Hardware del sistema local

El hardware empleado para este proyecto es el mostrado en la Fig.2.4. El corazón de este diseño se basa en una Raspberry PI 3 el cual cumple la función de computadora principal encargada de coordinar las acciones internas de todo el sistema. A esta computadora se encuentra conectada a una tarjeta de adquisición de datos y control de actuadores con la cual se comunica por medio de comandos, y por último, un módulo de indicadores LEDs con entradas auxiliares para la comunicación con el usuario. La Fig.2.4 también muestra los diferentes bloques que componen el hardware.

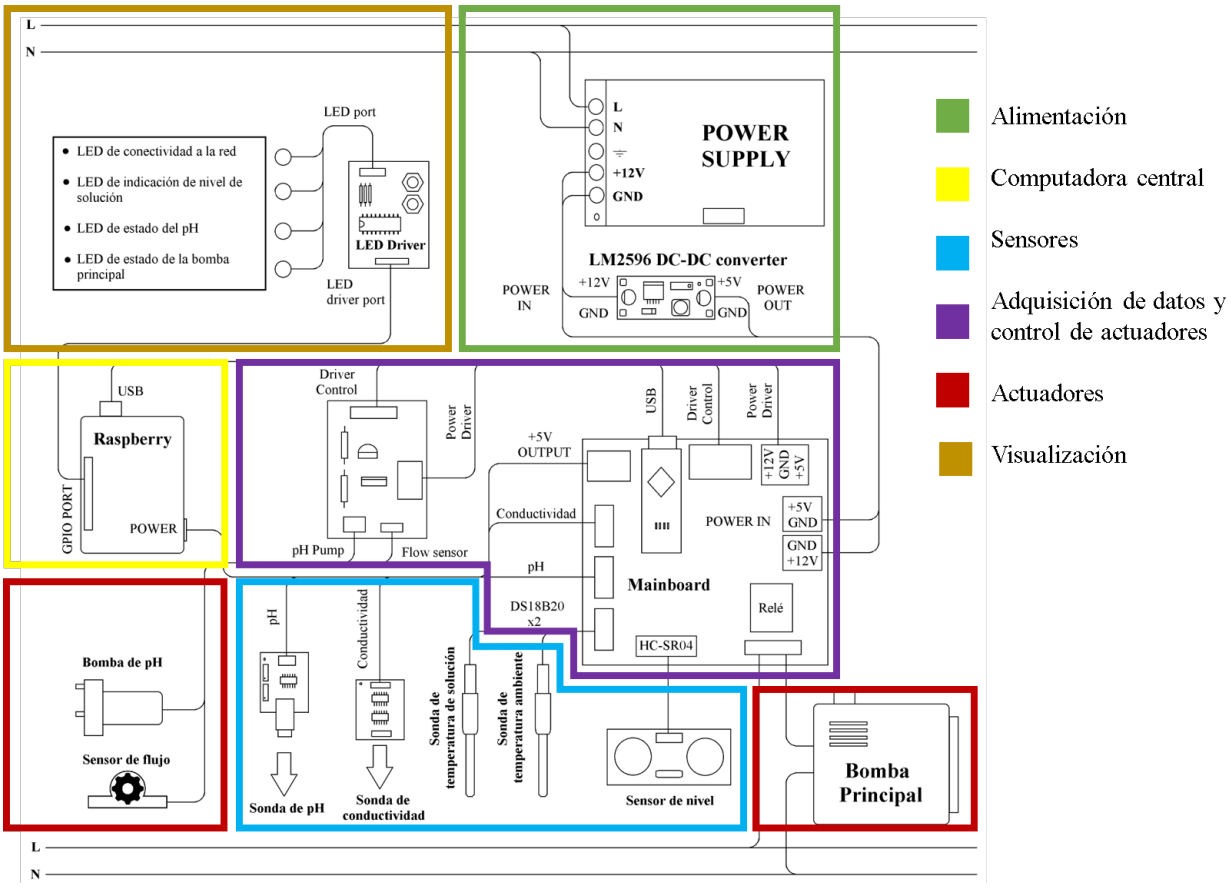


Figura 2.4: Bloques del Hardware del sistema local.

2.5. Adquisición de datos del sistema

La adquisición de datos es la obtención de información del cultivo hidropónico a través de los sensores del sistema. Es realizada por la tarjeta de adquisición de datos y control de salidas, también llamada tarjeta principal, donde la información es almacenada y actualizada hasta su uso por parte de la computadora.

2.5.1. Medición de temperatura

La medición de temperatura es realizada por medio de sensores digitales de la serie DS18B20, los cuales se comunican con la tarjeta de adquisición de datos por medio del bus bidireccional OneWire. Este sensor presenta dos características importantes para el proyecto. Estas son la capacidad de conectar varios de ellos en un mismo bus, como se ve en la Fig 2.5, y la protección de la humedad por una cápsula metálica de acero inoxidable. Esto último les permite a estos sen-

sores ser capaces de operar en las condiciones ambientales presentes en el cultivo hidropónico. La capacidad de integrar varios sensores con un único puerto de comunicación además permitió la simplificación del hardware.

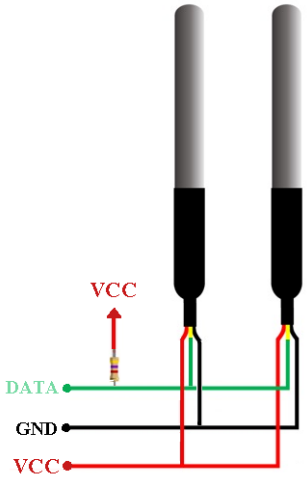


Figura 2.5: Conexión de múltiples sensores de temperatura DS18B20.

2.5.2. Medición de nivel

La medición de nivel se realizada por medio del sensor ultrásonico HC-SR04 ubicado en la tapa del tanque. Su funcionamiento consiste en descontar a la altura total del tanque la medición de la distancia entre el sensor y el nivel de solución. A este cálculo se integra un valor de offset debido a que el sensor se encuentra a una altura mayor que el máximo nivel de solución (Fig. 2.6).

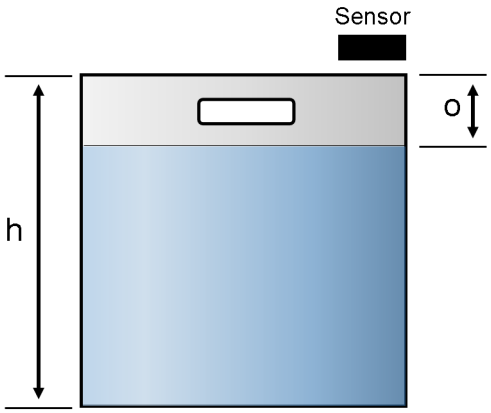


Figura 2.6: Representación simplificada del tanque y los parámetros empleados para la medición de nivel.

La fórmula de medición de nivel es la mostrada en la ecuación 2.1 la cual tiene como respuesta un valor de nivel en cm. Debido a que este resultado es relativo a la altura total del tanque, puede resultar poco intuitivo para el usuario. De la ecuación 2.1 se deriva la ecuación 2.2, la cual presenta el valor del nivel de forma porcentual, facilitando su interpretación.

$$Nivel(cm) = Altura(cm) - Medición(cm) \tag{2.1}$$

$$Nivel(\%) = \frac{Nivel(cm)}{Altura(cm) - Offset(cm)} \times 100 \tag{2.2}$$

2.5.3. Medición de pH

La medición de pH se realiza por medio de la sonda de pH en conjunto con un módulo acondicionador de señal el cual genera en su salida un voltaje proporcional al pH medido. La salida del acondicionador es leída a través del puerto de lectura analógico propio del microcontrolador Atmega328p el cual tiene una resolución de 10 bits. Las ecuaciones empleadas para la medición de pH son la ecuación 2.3 encargada de convertir la lectura analógica de a un valor de voltaje y la ecuación 2.4 que convierte el voltaje a pH. Esta última sera denominada también como: Ecuación de calibración de la sonda de pH.

$$Voltaje(V) = LecturaAnálogica(0-1023) \times \frac{5V}{1023} \tag{2.3}$$

La ecuación 2.4 corresponde a una ecuación lineal donde se puede interpretar a la pendiente (m) como una ganancia y al valor independiente (b) como un offset.

$$pH = m \times Voltaje(V) - b \tag{2.4}$$

2.5.4. Medición de conductividad

Las técnicas de medición de conductividad de líquidos tienen una característica que impide la medición simultánea con otros sensores, la cual consiste en hacer circular una corriente continua o alterna para medir la resistividad de la solución. Esta corriente puede, al estar en contacto con la solución, afectar la lectura de otras sondas por lo cual representa una limitación importante al proyecto. Es debido a este inconveniente que la metodología aplicada para la medición de conductividad consiste en la extracción de una muestra de la solución aislada del resto de sondas. La medición se realiza por medio de la sonda y acondicionador respectivo, el cual genera un voltaje proporcional a la conductividad, y es leído por el puerto analógico del microcontrolador Atmega328p por medio de la ecuación 2.3. La ecuación 2.5 es la empleada para convertir el voltaje leído a un valor válido de conductividad.

$$k(conductividad) = \frac{Q}{R10 \cdot |Vin|} \times Vout \tag{2.5}$$

Realizando un cambio de variables a la ecuación 2.5, se logra llegar a la ecuación 2.7, mucho más reducida la cual es empleada por el proyecto:

$$C = \frac{Q}{R10 \cdot |Vin|} \tag{2.6}$$

Donde C es la constante de celda de la sonda de conductividad.

$$k(conductividad) = C \times Vout \tag{2.7}$$

2.5.5. Medición de flujo para la dosificación de solución reguladora de pH

Para suministrar la cantidad correcta de solución de regulación de pH se hace uso de un caudalímetro diagrama interno se puede observar en la Fig.2.7. Este sensor envía una señal de onda cuadrada cada vez que el rotor interno completa una revolución. La onda cuadrada es leída por medio de una interrupción en la tarjeta principal y es empleada como retroalimentación para conocer el volumen de solución desplazado.

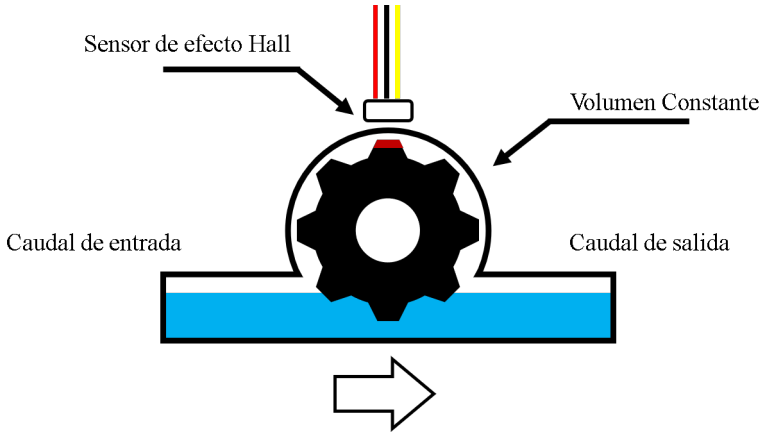


Figura 2.7: Diagrama interno del sensor de caudal.

2.6. Tarjeta principal del sistema

La tarjeta principal o tarjeta de adquisición de datos y control de actuadores (Fig 2.8) es una interfaz entre los sensores, actuadores y la computadora principal. Su función como su nombre lo indica es la de realizar las mediciones de todos los sensores que conforman el sistema y activar o desactivar las salidas. Consta de entradas para los sensores de temperatura, el sensor ultrásonico, la sonda de pH y la sonda de conductividad e integra el hardware necesario para activar una bomba de 110V y controlar una bomba pequeña para la dosificación de solución reguladora de pH. Esta tarjeta se comunica con la computadora principal vía USB por medio de comandos.

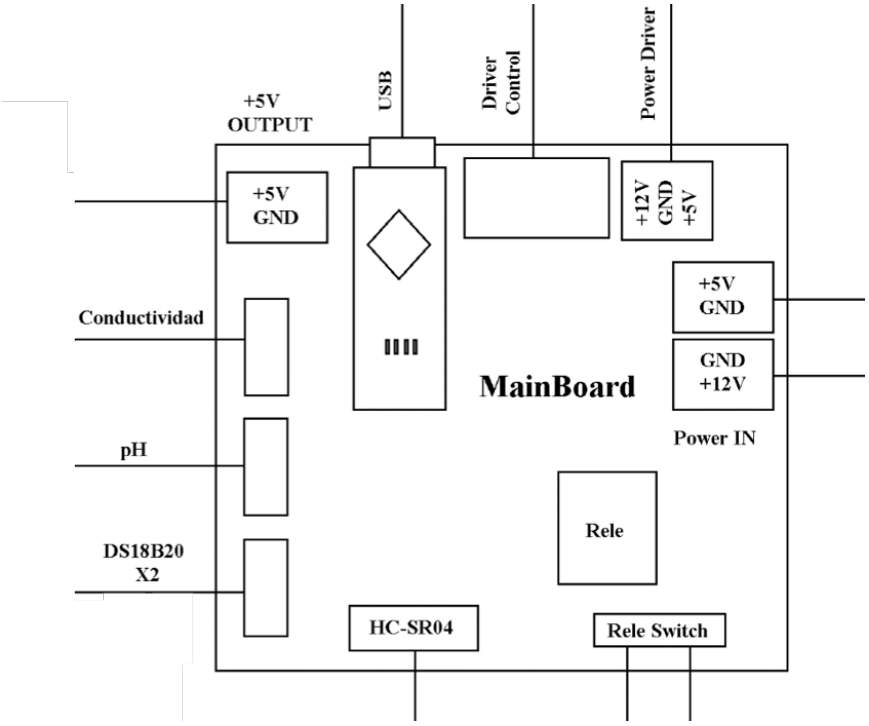


Figura 2.8: Diagrama de la tarjeta de adquisición de datos y control de actuadores.

La tarjeta principal también tiene una función adicional la cual consiste en distribuir los voltajes de alimentación entre los diferentes módulos que componen el sistema.

2.6.1. Lógica de la tarjeta principal

El sofware programado en la tarjeta principal se basa en una máquina de estados finitos con transiciones basadas en tiempos. En la Fig.2.9 se puede observar los estados o funciones que realiza el programa y el tiempo tras el cual se ejecuta cada función. La descripción de cada estado es la siguiente:

- **Configuración inicial de la tarjeta:** Este estado corresponde a la configuración de los puertos, estado inicial de las salidas e inicialización de variables .
- **Actualización al desborde:** Es una función especial dedicada a restablecer las variables de tiempo que se utilizan internamente para calcular los tiempos de transición. Surge como resultado de contrarrestar la limitación que posee la función millis() en cuanto el tiempo máximo que puede contabilizar.
- **Serial Request:** Es la función encargada de la comunicación con la computadora principal. Gestiona la transferencia de información y realiza el control de los actuadores en función a los comandos que se procesen en este estado.
- **Lectura de los sensores:** Función encargada de leer a intervalos regulares los sensores de temperatura por el bus OneWire y el sensor de nivel. Para el caso de la conductividad y el pH, realiza una suma acumulativa de las lecturas que posteriormente sera utilizada para calcular su promedio.
- **Cálculo de la conductividad y el pH:** Función encargada de calcular el valor real de la conductividad y el pH al calcular el promedio de la suma acumulada por las lecturas realizadas a los sensores respectivos.

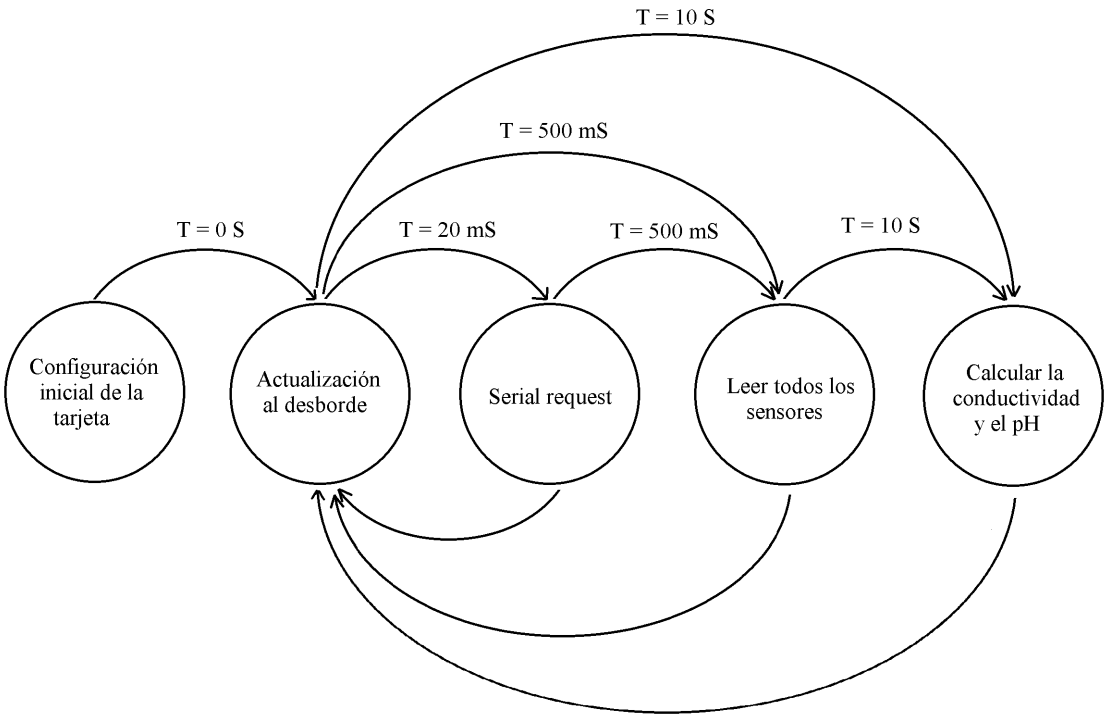


Figura 2.9: Representación de la máquina de estados que maneja la tarjeta principal.

2.6.2. Comunicación entre la computadora y la tarjeta principal

La función “Serial request” es la encargada de gestionar la transferencia de información y la comunicación entre la tarjeta y la computadora principal. Su metodología se basa en el uso de comandos enviados desde la computadora para la realización de tareas. La función es llamada cada 20 mS y se ejecuta siempre que se haya recibido un dato por el puerto serial. Cuando se ejecuta un comando se envía un dato de confirmación que indica a la computadora que la tarea se ha realizado con éxito o ha ocurrido un error. La Fig.2.10 muestra el diagrama de flujos correspondiente a la función “Serial request”. La lista de comandos y su respectiva función es la siguiente:

- **Comando “R”:** Leer sensores de temperatura de solución, temperatura ambiente y nivel de solución. Envía los datos de estos sensores por el puerto serial y envía el dato de confirmación “r”.
- **Comando “H”:** Leer sensores de conductividad y de pH. Envía los datos de estos sensores por el puerto serial y envía el dato de confirmación “h”.
- **Comando “P”:** Comando de control de la bomba principal. Envía el dato de confirmación “p“ y espera por un parámetro de control que puede ser “O” para el encendido de la bomba o “F” para el apagado. Posteriormente envía el dato de confirmación “o” o “f” respectivamente.
- **Comando “-”:** Comando de dosificación de solución reguladora de pH. Espera a por el parámetro de dosificación el cual es recibido en 4 caracteres y envía el dato de confirmación“c”.

Cualquier otro dato que no sea un comando conocido produce el envío del dato de error “Z”.

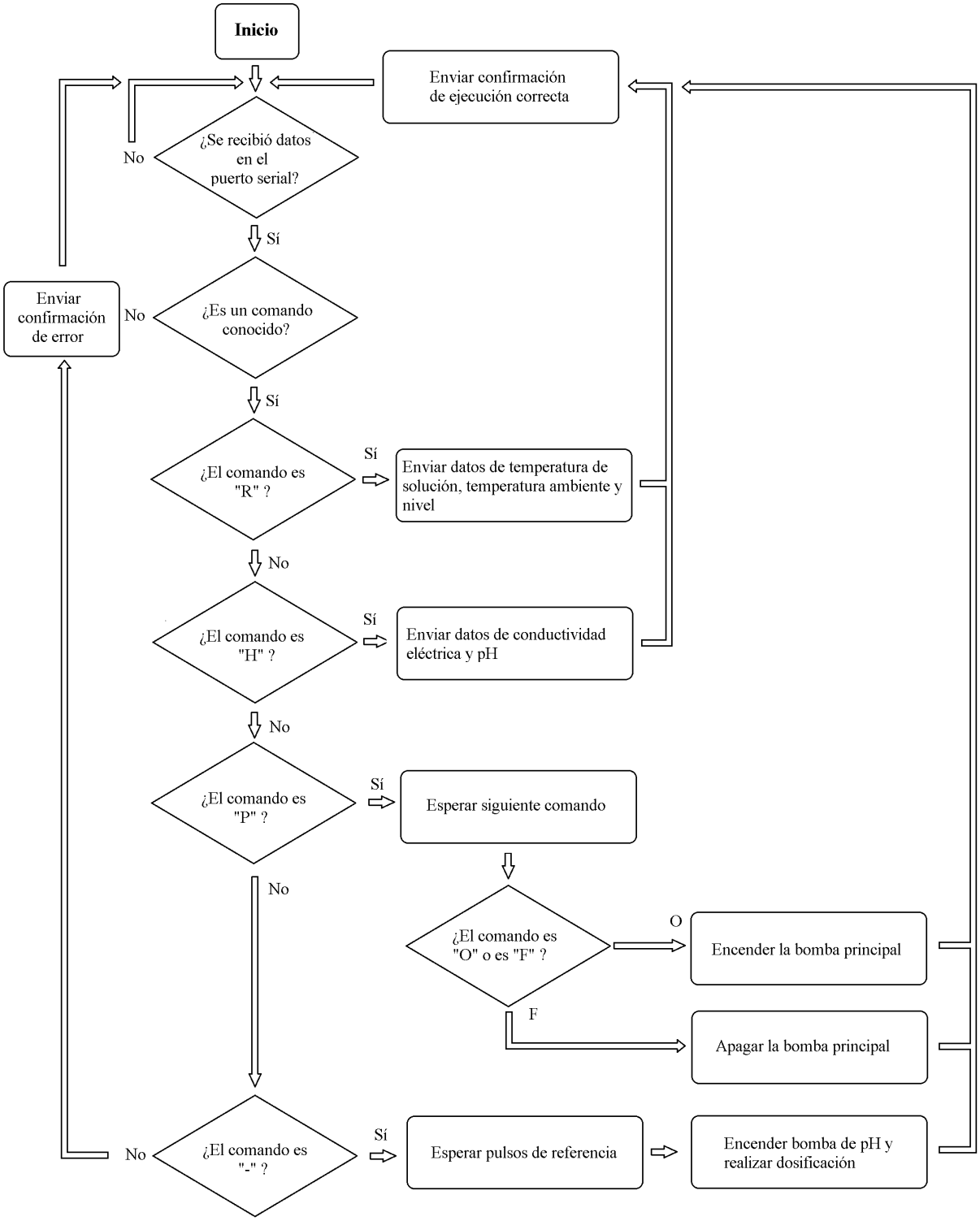


Figura 2.10: Digrama de flujo del procesamiento de los comandos de la tarjeta principal.

2.6.3. Adquisición y pre-procesamiento de datos

La adquisición de datos corresponde a la etapa de lectura de sensores, la cual es realizada cada 500 mS. En este estado se actualizan los valores de temperatura y nivel, mientras que para el caso del pH y la conductividad, debido a requieren un mayor tiempo para su estabilización en la señal, se toman muestras que posteriormente se utilizaran para el cálculo de un promedio mucho más preciso por medio de la ecuación 2.8.

$$Promedio = \frac{Dato_1 + Dato_1 + ... + Dato_n}{N} \tag{2.8}$$

Donde cada dato corresponde a una muestra tomada de las sondas. El número de muestras utilizadas se puede calcular por medio de la ecuación 2.9, con lo que al reemplazar datos, se obtiene un total del 20 muestras. Es decir, el valor calculado de pH y conductividad proviene del promedio de 20 muestras durante un intervalo de 10 S.

$$N = \frac{Intervalo\,de\,actualización\,de\,pH\,y\,conductividad}{Intervalo\,de\,lectura\,de\,los\,sensores} = \frac{10S}{500mS} = 20 \tag{2.9}$$

2.7. Control de actuadores

Los actuadores disponibles en este proyecto son la bomba principal y la bomba dosificadora de solución reguladora de pH. Estos actuadores son controlados por medio de comandos enviados por la computadora, procesados por la función “Serial request”.

2.7.1. Control de la bomba principal

La bomba principal es controlada únicamente por comandos de encendido y apagado provenientes de la computadora principal, la cual realiza una u otra acción según se requiera.

2.7.2. Control de la bomba de pH

El control de la bomba dosificadora de pH se realiza en lazo cerrado con un caudalímetro modelo YF-S402 el cual esta constituido internamente por un rotor móvil, un imán y un sensor de efecto Hall. Para el diseño del controlador se asumió una relación de equivalencia entre un giro completo del rotor y el volumen de solución desplazado. La ecuación de correlación lineal que relaciona ambas variables se calculó posteriormente por medio de una prueba experimental y se integró en el programa de la computadora principal. La lógica de control de la bomba de pH esta dividida en dos partes. En primer lugar, el volumen de solución reguladora a suministrar y su conversión a pulsos se realiza en la computadora principal. La segunda parte ocurre en la tarjeta principal donde la bomba de pH es encendida mientras se realiza el conteo de pulsos. Cuando estos pulsos alcanzan el valor calculado, se apaga la bomba.

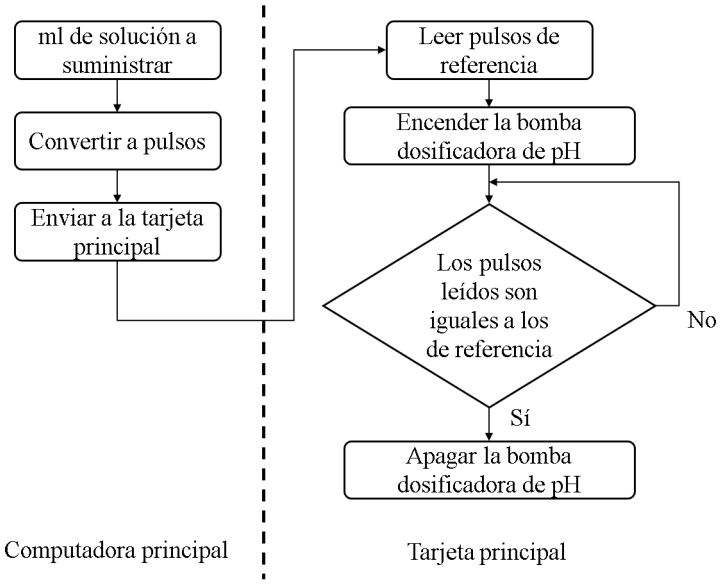


Figura 2.11: Control de la bomba dosificadora de pH.

2.8. Controlador de indicadores LEDs

El controlador de indicadores LEDs es una tarjeta desarrollada con el propósito de manejar los elementos visuales del sistema de forma independiente a la computadora principal. Maneja un array de cuatro diodos LED RGB enumerados desde el LED0 al LED3 siendo capaz de configurar hasta 8 colores distintos y ajustar parpadeos individuales a distintos tiempos. La tarjeta muestra el estado de la conexión a internet del sistema, el nivel de solución en el tanque, el estado de la bomba principal y el estado del pH de la solución. Es comandada por un microcontrolador PIC16f628A y se comunica con la computadora por medio de su USART, empleando comunicación serial sincrónica y comandos.

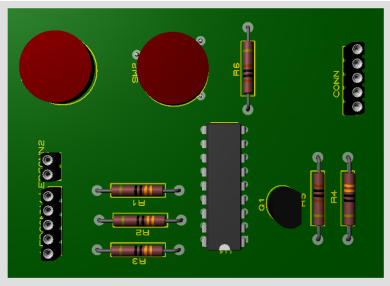


Figura 2.12: Diseño del panel del controlador de LEDs.

2.8.1. Protocolo de comunicación del controlador de indicadores LEDs

El protocolo de comunicación que utiliza el controlador de indicadores LEDs es el SPI. Este opera empleando el modo que se puede ver en la Fig.2.13 donde la línea de selección de dispositivo se activa a nivel alto mientras que la línea de datos es muestreada cada flanco de bajada del reloj. El envío de comandos se realiza en paquetes de 8 bits a 9600 baudios con un refresco cada 10 mS.

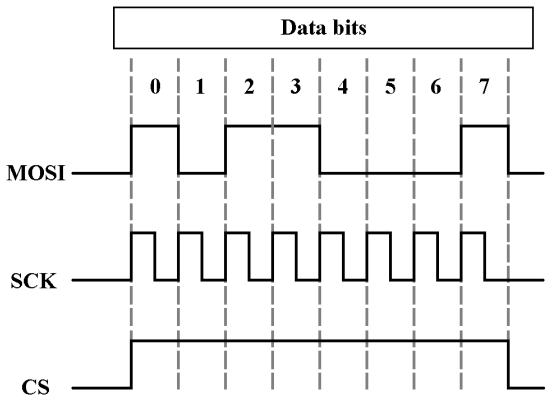


Figura 2.13: Protocolo Serial SPI.

2.8.2. Lista de comandos del controlador de indicadores LEDs

La tarjeta de indicadores LEDs opera con 4 comandos los cuales son los presentados en la tabla.2.2.

Cuadro 2.2: Comandos soportados por el controlador LED

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Comando
0	0	0	X	X	X	X	X	Clear
0	0	1	LS.1	LS.0	B	G	R	Set Color
0	1	0	X	LM1.1	LM1.0	LM0.1	LM0.0	Mode 0-1
0	1	1	X	LM3.1	LM3.0	LM2.1	LM2.0	Mode 2-3

2.8.3. Descripción de comandos

Clear. - Comando que configura todos los leds como apagados y desactiva cualquier efecto configurado.
Set Color. - Comando que configura un color a un LED por medio del argumento LS.[0-1] (Led select) de acuerdo a la tabla 2.3. El color se selecciona por medio de la tabla 2.4.
Mode 0-1. - Comando que configura un modo de parpadeo a los LEDs 0 y 1. El argumento LM[0-1].[0-1] (Led mode), permite seleccionar el tiempo de parpadeo de acuerdo a la tabla 2.5.
Mode 2-3. - Comando que configura un modo de parpadeo a los LEDs 2 y 3. El argumento LM[2-3].[0-1] (Led mode), permite seleccionar el tiempo de parpadeo de acuerdo a la tabla 2.5.

Cuadro 2.3: Selección del LED para asignación de color

LS.1	LS.0	LED Seleccionado
0	0	LED 0
0	1	LED 1
1	0	LED 2
1	1	LED 3

Cuadro 2.4: Asignación de colores

B	G	R	Color
0	0	0	Blanco
0	0	1	Turquesa
0	1	0	Magenta
0	1	1	Azul
1	0	0	Amarillo
1	0	1	Verde
1	1	0	Rojo
1	1	1	Apagado

Cuadro 2.5: Modos de parpadeo

LED 3		**LED 2**		**LED 1**		**LED 0**		
LM3.1	**LM3.0**	**LM2.1**	**LM2.0**	**LM1.1**	**LM1.0**	**LM0.1**	**LM0.0**	**Efecto**
0	0	0	0	0	0	0	0	Sin efecto
0	1	0	1	0	1	0	1	Parpadeo cada 250 mS
1	0	1	0	1	0	1	0	Parpadeo cada 500 mS
1	1	1	1	1	1	1	1	Parpadeo cada 1000 mS

2.9. Computadora principal

La computadora principal es el centro del sistema local. Es la encargada de realizar el procesamiento de los datos tomados por la tarjeta principal, realizar las tareas de riego automático, riego manual, funciones especiales, calibración de sondas, intercambio de información con la nube y control de actuadores. Su software se basa en una máquina de estados finitos corriendo sobre una Raspberry PI 3, con transiciones basadas en condiciones e intervalos de tiempo. Presenta una importante característica la cual esta relacionada con la comunicación con la nube, la cual se realiza de forma paralela a la ejecución del resto del programa, evitando que los tiempos de espera correspondientes a la transferencia de datos con el servidor produzcan retrasos en el resto de tareas. La computadora principal se comunica con el resto de periféricos por medio de comunicación serial, empleando el puerto USB para gobernar a la tarjeta principal y los puertos GPIO para conectarse al controlador de indicadores LED. El ajuste de las distintas funciones que realiza la computadora puede ser realizada desde la nube a través de los bloques configuración y calibración. Se alimenta a 5V por medio de una salida de voltaje ubicada en la tarjeta principal.

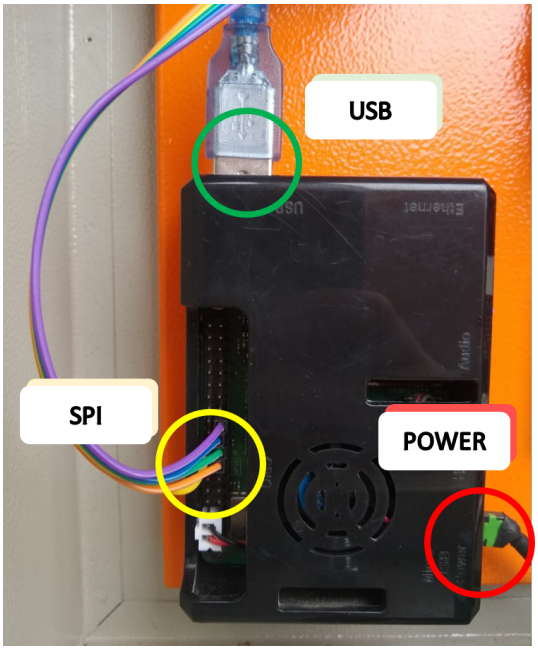


Figura 2.14: Computadora principal y sus conexiones.

2.10. Lógica de la computadora principal

La computadora principal esta gobernada por una máquina de estados finitos para la ejecución de todas sus funciones a excepción de la comunicación con la nube, la cual es realizada por una tarea en paralelo con el objetivo de evitar retrasos en las tareas por la transferencia de información. La Fig.2.15 muestra un esquema de la inicialización de la computadora principal.

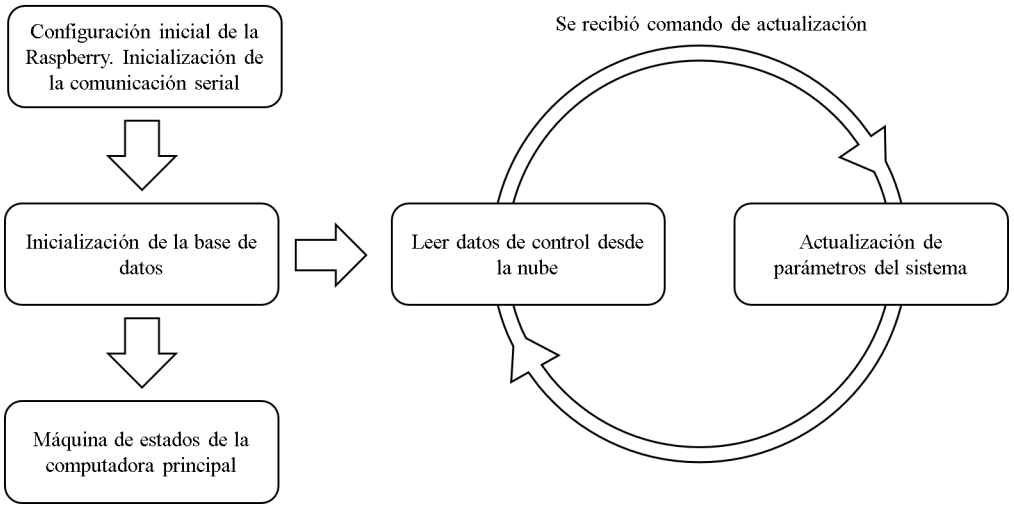


Figura 2.15: Esquema de inicialización de la computadora principal.

La inicialización de la computadora principal se compone de un conjunto de actividades importantes que permiten el funcionamiento normal del sistema. La lista de actividades que realiza la inicialización es la siguiente:

- **Inicialización de comunicación serial:** Inicia la comunicación USB con la tarjeta principal y el controlador de indicadores LEDs.
- **Inicialización de variables:** Establece el valor inicial de las variables del sistema.
- **Configuración del sistema:** Carga los valores de configuración propios del sistema desde el archivo de configuración
- **Inicialización de la base de datos:** Inicializa la comunicación con la base de datos y arranca el subproceso de comunicación con la nube.
- **Registro del dispositivo:** Actividad cuya función es cargar a la nube todos los bloques de operación necesarios para la operación del sistema. El registro del dispositivo emplea la MAC del adaptador de red como código de identificación único y realiza el proceso una sola vez en la primera ejecución del programa. En las próximas ejecuciones, la función actualiza un parámetro en la nube que permite al dispositivo de supervisión remoto tener acceso a los datos del sistema local.

La tarea de comunicación con la nube es un subproceso que se encarga de leer el bloque de control ubicado en la base de datos online y actualizar la configuración del sistema. Cumple un rol importante en la activación de los estados con transiciones por condiciones y en control de actuadores en general. Su intervalo de ejecución es cada 5 S.

2.10.1. Máquina de estados finitos y funciones del sistema

La máquina de estados finitos es el núcleo del software de la computadora principal, esta compuesto por 13 estados independientes los cuales pueden ser clasificados en estados accionados por transiciones de tiempo, es decir, aquellos que se ejecutan transcurrido un intervalo determinado y estados accionados por transiciones a condiciones, los cuales son todos los estados que deben cumplir una condición para su ejecución. El estado 0 por el cual inicia el programa realiza dos actividades que son: “Actualización del tiempo del sistema” una función encargada de mantener la ejecución de todo el programa dentro de los tiempos determinados; y “Actualización de variables del sistema”, encargada de actualizar todas las variables empleadas para las transiciones por condiciones.

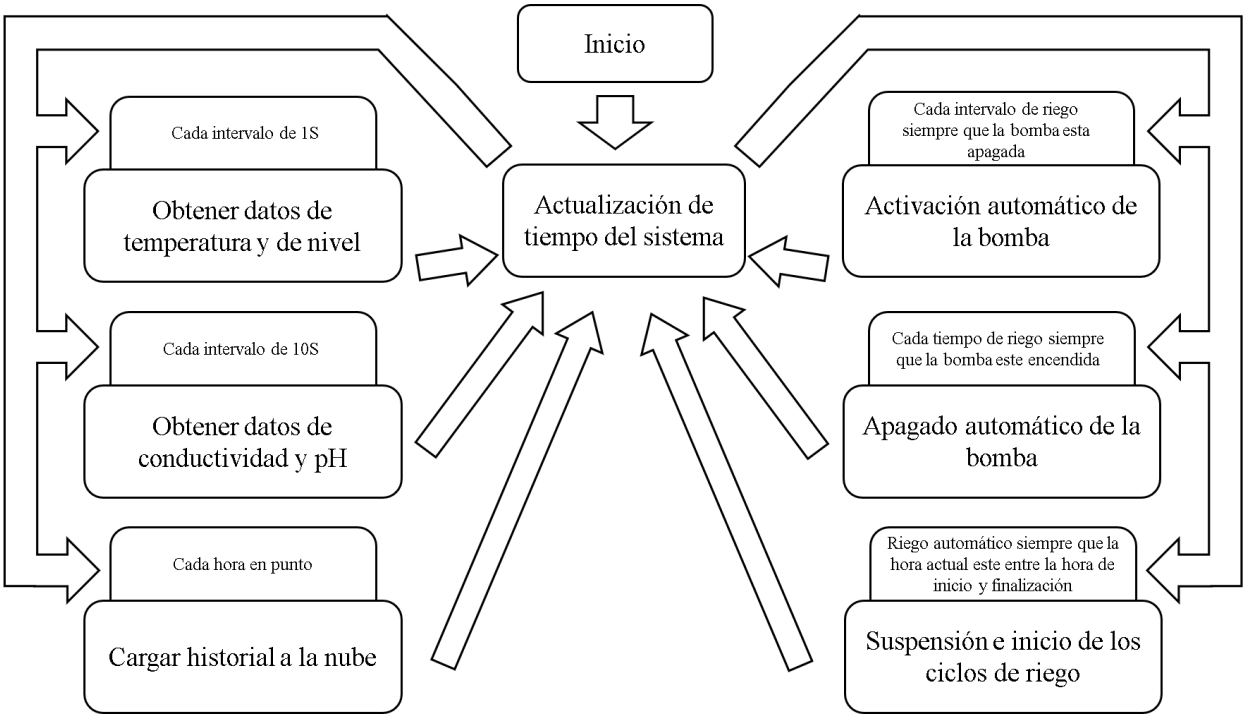


Figura 2.16: Máquina de estados de la computadora principal, sección controlada por tiempos.

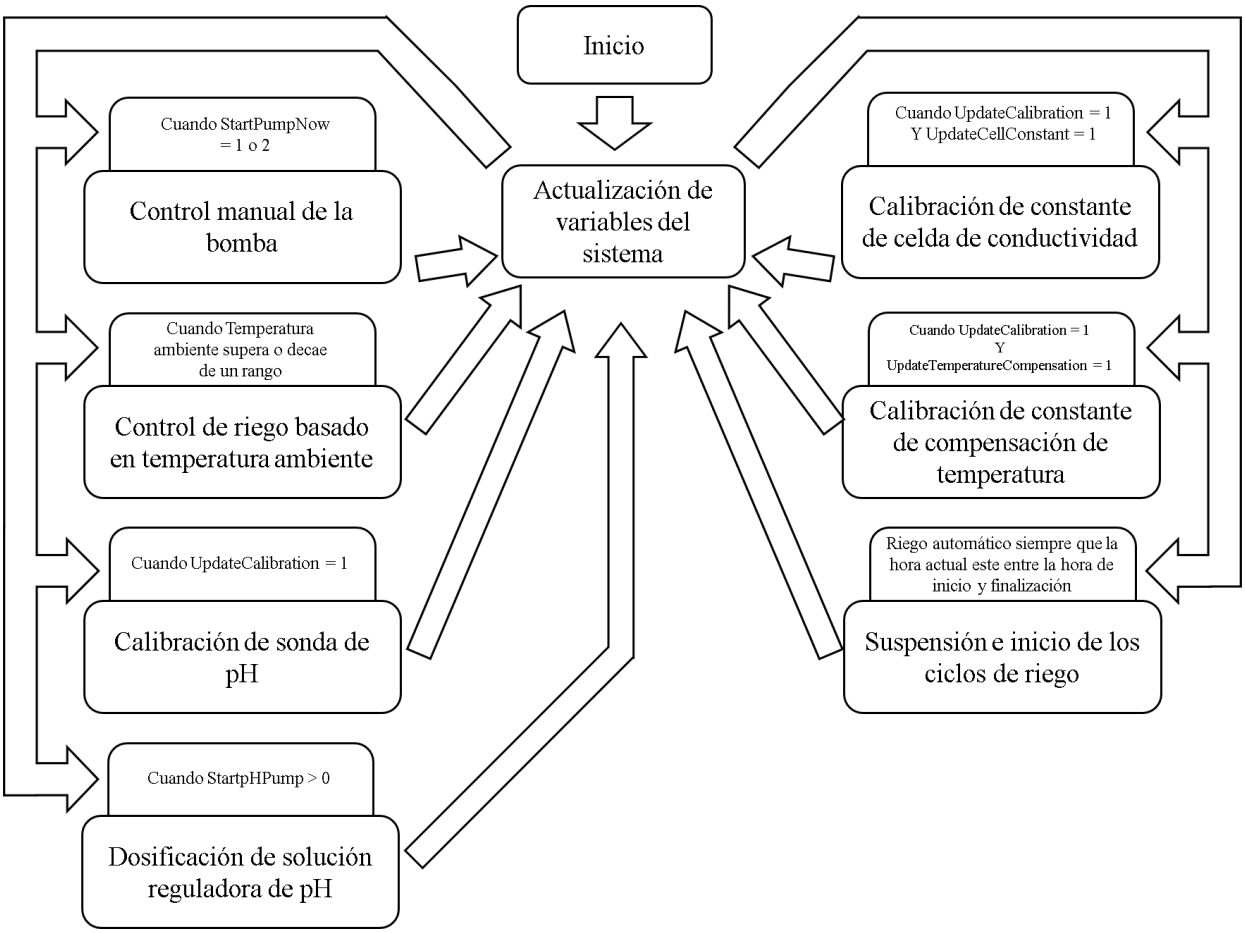


Figura 2.17: Máquina de estados de la computadora principal, sección controlada por condiciones.

2.10.2. Descripción y clasificación de funciones

Las estados o funciones del sistema local pueden ser agrupadas en cuatro categorías de acuerdo a las actividades que realizan. Estos grupos son:

- **Funciones generales**
- **Funciones especiales**
- **Funciones de calibración**
- **Funciones de seguridad**

2.10.3. Funciones generales

Son todas aquellas funciones orientadas a tareas de riego del cultivo, obtención y manejo de datos y control de actuadores. Entran en este grupo las funciones de lectura de sensores, control de la bomba principal y bomba de pH, el riego automático, riego manual y la generación de historial en la nube.

2.10.4. Funciones especiales

Son todas aquellas funciones que modifican el comportamiento del sistema cuando ocurre una determinada condición. Las funciones de control de riego por temperatura y la suspensión e inicio de ciclos de riego son parte de este grupo.

2.10.5. Funciones de calibración

Son funciones encargadas de calibrar los elementos de medición de acuerdo a los parámetros establecidos por el usuario. Estas funciones solo pueden ser activadas a través de la nube por medio de los bloques de control necesarios. Son parte de este grupo las funciones de: calibración de sonda de pH, calibración de celda de conductividad y la calibración de constante de compensación de temperatura.

2.10.6. Funciones de seguridad

Son funciones que permiten que el sistema trabaje de manera correcta e informan al usuario de lo que esta sucediendo en el cultivo. La descripción de las funciones de seguridad se puede ver en la tabla 2.6

Cuadro 2.6: Tabla de funciones de seguridad

Función de seguridad	**Descripción**
Desactivación de la bomba por nivel crítico de solución	Función encargada de desactivar el riego automático cuando el nivel de la solución es demasiado bajo.
Alarma de pH ácido	Función que indica de forma visual que el pH de la solución esta bajo el umbral ácido (menor a 5 pH).
Reconexión a red	Función que se encarga de reconectar el sistema a la red en caso de un fallo con internet

2.10.7. Otras funciones

Dentro de las funciones de la máquina finita de estados, existen otras funciones las cuales se ejecutan a medida que son requeridas. Estas funciones son: función de visualización de los controladores LEDs, algoritmos de compensación de temperatura para conductividad y los algoritmos de comunicación serial.

2.10.8. Función de conductividad de referencia para cálculo de constante de celda de conductividad

Es una función que permite obtener el valor de la conductividad de la solución de referencia 1413 uS a cualquier temperatura dada. Esta conformada por una función de tercer grado la cual describe el comportamiento de la conductividad de dicha solución. La curva de conductividad de la solución de referencia esta dada por los datos encontrados en la tabla.2.7.

Cuadro 2.7: Tabla de conductividad de la solución de referencia 1413 uS - 25 °C

Temperatura	**Conductividad uS**
5	896
10	1020
15	1147
16	1173
17	1199
18	1225
19	1251
20	1278
21	1305
22	1332
23	1359
24	1386
25	1413
26	1440
27	1467
28	1494
29	1521
30	1548
31	1575

La Fig.2.18 forma de esta curva puede ser aproximada a una ecuación lineal, sin embargo la facilidad y precisión que una curva de grado 3 presentaba fue motivo por el cual se decidió implementarla en el código. La metodología de uso para esta función consiste en introducir en la ecuación 2.10 el valor de la temperatura de solución medida por el sensor cuando ocurre un evento de calibración, obteniendo como resultado el valor de conductividad teórico que debe tener la solución de referencia. Este dato se divide para el valor de voltaje medido por la sonda

de conductividad. El resultado de esta operación, como se observa en la ecuación 2.11 es el valor de la constante de celda de conductividad.

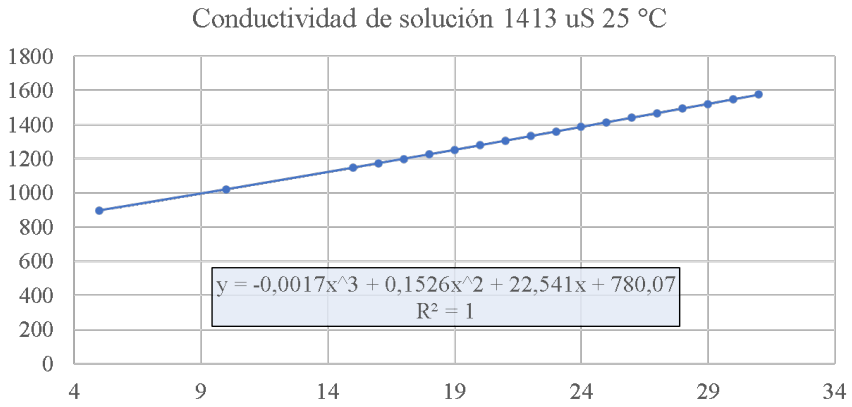


Figura 2.18: Curva de conductividad respecto a la temperatura.

$$k_{25C} = -0,0017T^3 + 0,1526T^2 + 22,541T + 780,07 \tag{2.10}$$

La ecuación 2.11 proviene de la ecuación 2.7, empleada para calcular la conductividad.

$$C = \frac{k_{25C}}{Vout} \tag{2.11}$$

2.10.9. Algoritmo de compensación de temperatura para conductividad

La conductividad de una solución esta fuertemente relacionada con la temperatura a la cual se esta realizando la medición. Un incremento en la temperatura tiene un efecto proporcional en la conductividad, lo cual puede resultar en un valor medido mayor al esperado [47]. Con el fin de contrarrestar este efecto en la medición de conductividad, se introdujo en el algoritmo de medición dos funciones. La función de compensación de temperatura y la función de cálculo de la constante de compensación. La primera función se encarga de llevar la medición de la conductividad a la esperada por el usuario. La segunda función es la encargada de calcular el coeficiente de compensación de temperatura, el cual es un valor con el cual se realizan los cálculos. La metodología utilizada consiste en emplear una solución ya pre-ajustada al valor de la conductividad de referencia y realizar el ajuste del coeficiente de compensación a una temperatura ligeramente diferente a la de referencia, esto debido a que es necesaria una diferencia de temperatura para el cálculo. El algoritmo de compensación de temperatura se basa en la ecuación 2.12.

$$K = \frac{K_\theta}{1 + \frac{\alpha_{T_\theta, T_{Ref}}}{100} \times (T_\theta - T_{Ref})} \tag{2.12}$$

Mientras que el calculo del coeficiente de variación de temperatura se realiza por medio de la ecuación 2.13.

$$\alpha_{T_\theta, T_{Ref}} = \frac{(K_\theta - K_{Ref}) \times 100}{K_{Ref} \times (T_\theta - T_{Ref})} \tag{2.13}$$

Las ecuaciones 2.12 y 2.13 se pueden encontrar en [43].

2.10.10. Calibración local

La calibración local consiste en una función que se encarga de cargar en el sistema un archivo de configuración en el cual se encuentran almacenados una serie de parámetros que regulan el comportamiento del sistema local. Este archivo no puede ser accedido desde la base de datos online y por consiguiente desde el dispositivo de supervisión, por lo que solo puede ser editado directamente desde el gestor de archivos de la computadora principal. El archivo de configuración y la descripción de sus parámetros son mostrados en la Fig.2.19. La función de calibración local es ejecutada únicamente al inicializar el programa, sin embargo existen otras dos funciones del sistema que se encargan de sobrescribir el archivo cuando es necesario. Estas funciones son la función de calibración de celda de conductividad y la función de calibración de coeficiente de variación de temperatura.

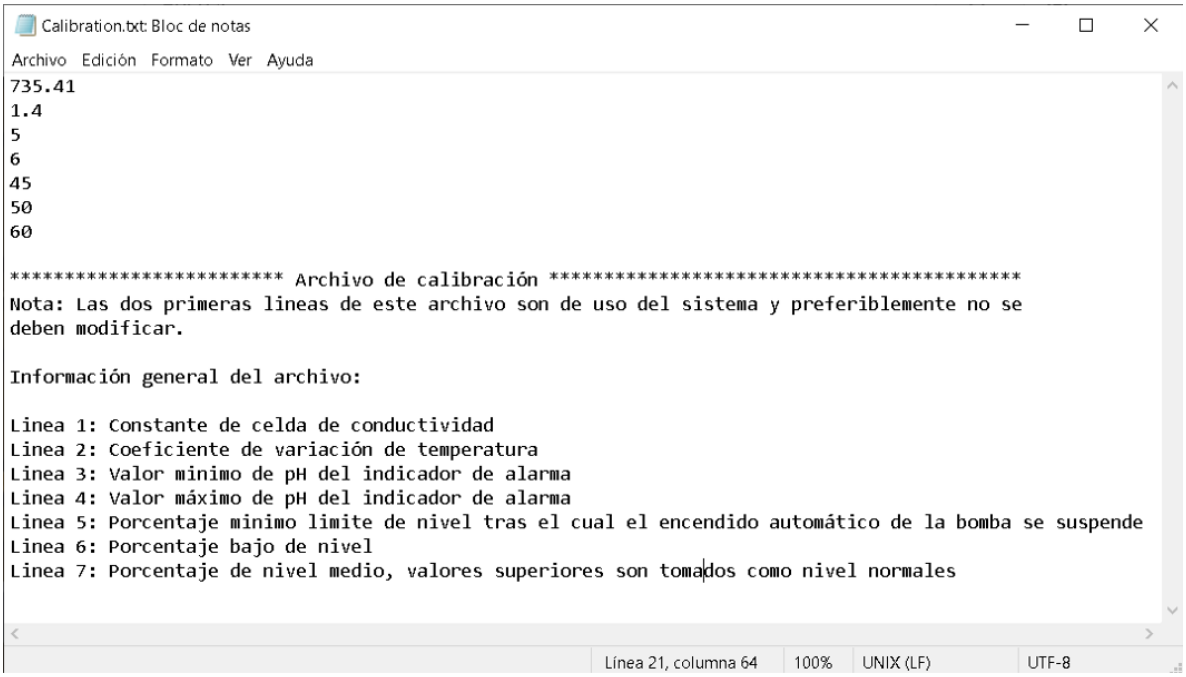


Figura 2.19: Archivo de calibración del sistema local. Fuente propia

2.10.11. Visualización

La función de visualización se encarga de informar por medio de los indicadores LED, el estado de la conexión a la red, nivel de solución, el pH de la solución y el estado de la bomba principal. La función es llamada por los eventos que se pueden ver en la Fig.2.20.

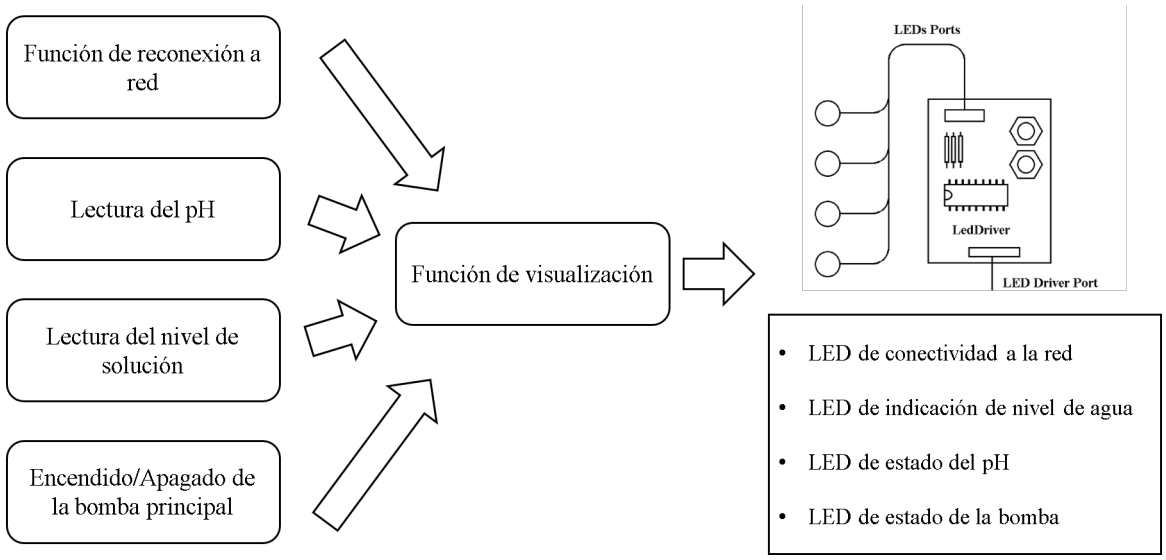


Figura 2.20: Eventos que ejecutan la función de visualización. Fuente propia

Otra característica de esta función es que solo se comunica con el controlador de indicadores LED únicamente cuando se requiere realizar un cambio en el color o en el modo de parpadeo en el LED seleccionado. Ajustar un color y modo ya configurado no iniciara la comunicación serial.

2.10.12. Formato de visualización:

Los indicadores LEDs emplean los colores y el parpadeo para indicar lo que esta ocurriendo en el sistema. El formato de visualización que emplea cada indicador se puede ver a continuación:

- **Indicador de nivel**

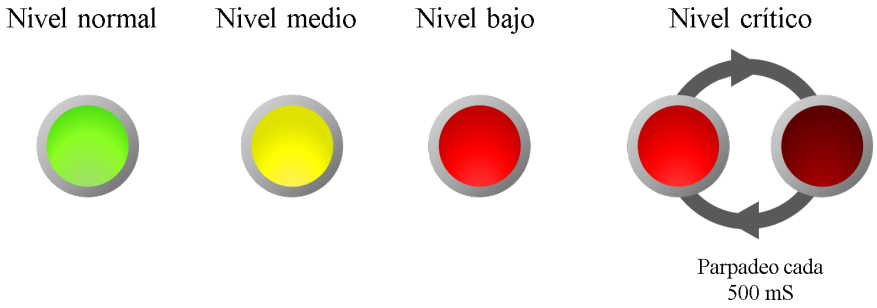


Figura 2.21: Formato de visualización para el indicador de nivel.

- **Indicador de pH**

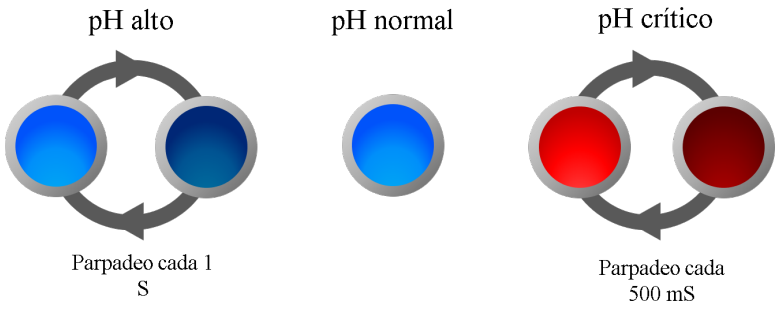


Figura 2.22: Formato de visualización para el indicador de pH.

- **Indicador de red**

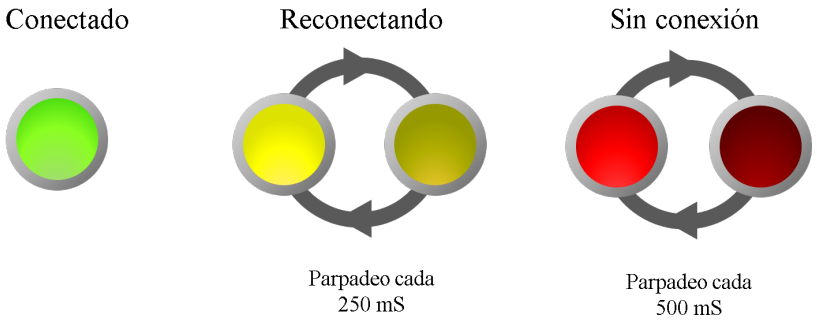


Figura 2.23: Formato de visualización para el indicador de red.

- **Indicador de la bomba principal**

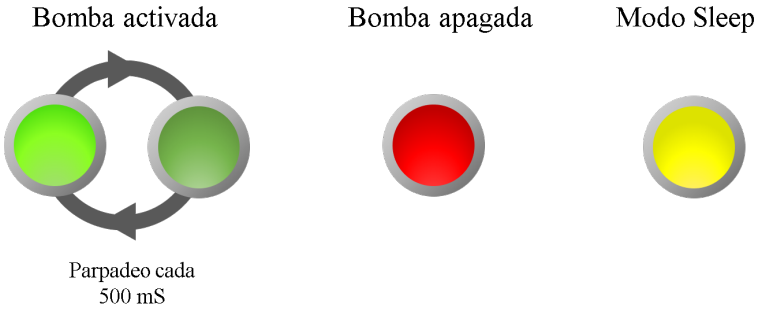


Figura 2.24: Formato de visualización para el indicador de la bomba principal.

2.11. Almacenamiento y comunicación

El bloque de almacenamiento y comunicación perteneciente al modelo del sistema, también denominado nube, es el encargado de coordinar el funcionamiento del sistema local a través de la información almacenada en la base de datos online y los parámetros enviados por el dispositivo de supervisión. Esta construida sobre el servicio Real Time Database brindado por Firebase para el almacenamiento y sincronización de datos en tiempo real. La Fig.2.25 muestra la arquitectura de la nube donde se observa el flujo de información que existe entre los dispositivos, siendo el dispositivo de lado derecho el de supervisión y el del lado izquierdo el sistema local.

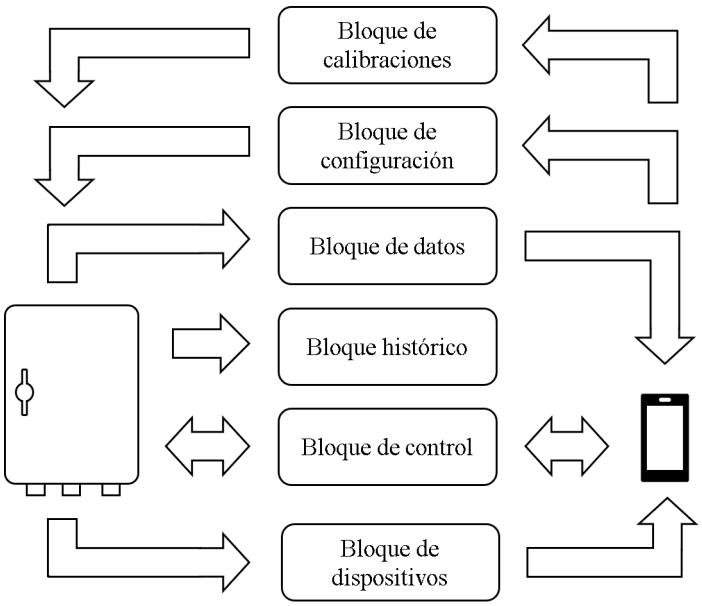


Figura 2.25: Arquitectura de la nube. Fuente propia

La descripción de cada bloque es la siguiente:

- **Bloque de control. -** Gestiona el intercambio de información entre el sistema local, la nube y el dispositivo de supervisión remota.
- **Bloque de datos. -** Almacena la información de los sensores, estado de los actuadores e información general del sistema.
- **Bloque de configuración. -** Almacena la configuración de los ciclos de riego.
- **Bloque de calibraciones. -** Almacena la información de calibración de sondas.
- **Bloque histórico. -** Es un registro histórico que almacena los datos del sistema local.

En la Fig.2.25 se observa la arquitectura de la base de datos y la forma en la que está organizada. Se puede observar que existe una comunicación bidireccional entre el sistema local y el dispositivo de supervisión por medio del bloque de control. Esto se debe a que este bloque es el encargado de gestionar la actualización de parámetros y el flujo de información en todo el sistema de supervisión. El resto de bloques almacenan los datos del sistema y su configuración.

2.11.1. Metodología de registro de dispositivos

El registro de dispositivos es una característica que permite crear en la nube los bloques necesarios para el control y configuración de nuevos sistemas locales. Para ello, existe un bloque adicional llamado bloque de dispositivos, el cual es visible para cualquier sistema local y tiene el propósito de almacenar la dirección MAC de todos los sistemas locales que se registren en la nube. Cuando un sistema local inicia, busca en este bloque su dirección MAC. Si no existe, el sistema local ejecuta una función que crea la arquitectura de la FIG.2.25 empleando como prefijo su propia dirección MAC. Posteriormente, registra su MAC en el bloque de dispositivos como un parámetro que adicionalmente es capaz de almacenar un valor. Esta cualidad es aprovechada para indicar públicamente a los dispositivos de supervisión remota cuando un sistema ya esta vinculado. Esto se realiza por medio de un valor booleano donde falso indica que esta vinculado y verdadero señala que esta disponible.

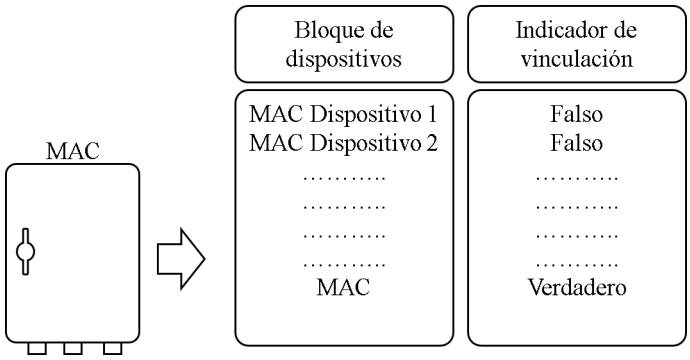


Figura 2.26: Registro del sistema local en el bloque de dispositivos y parámetro de vinculación.

Para el inicio de un sistema local cuya dirección MAC está registrada, ocurre un proceso diferente. En este caso, el sistema local coloca el valor verdadero en el parámetro de vinculación, permitiendo al dispositivo estar listo para vincularse a un dispositivo de supervisión remoto.

2.12. Acceso de dispositivos

Los bloques de la arquitectura de la base de datos pueden ser generados por cada sistema local que se registre en la nube, cambiando únicamente el prefijo en el nombre de los bloques. Esto obliga al sistema local a incluir su dirección MAC en las búsquedas por información en la nube, de tal forma que solo se pueda acceder a los datos que le correspondan. La Fig.2.27 muestra un ejemplo de las etiqueta asignadas para los bloques de la base de datos correspondiente al sistema local del proyecto. Para este caso, la MAC empleada es la: B8:27:EB:16:47:B4.

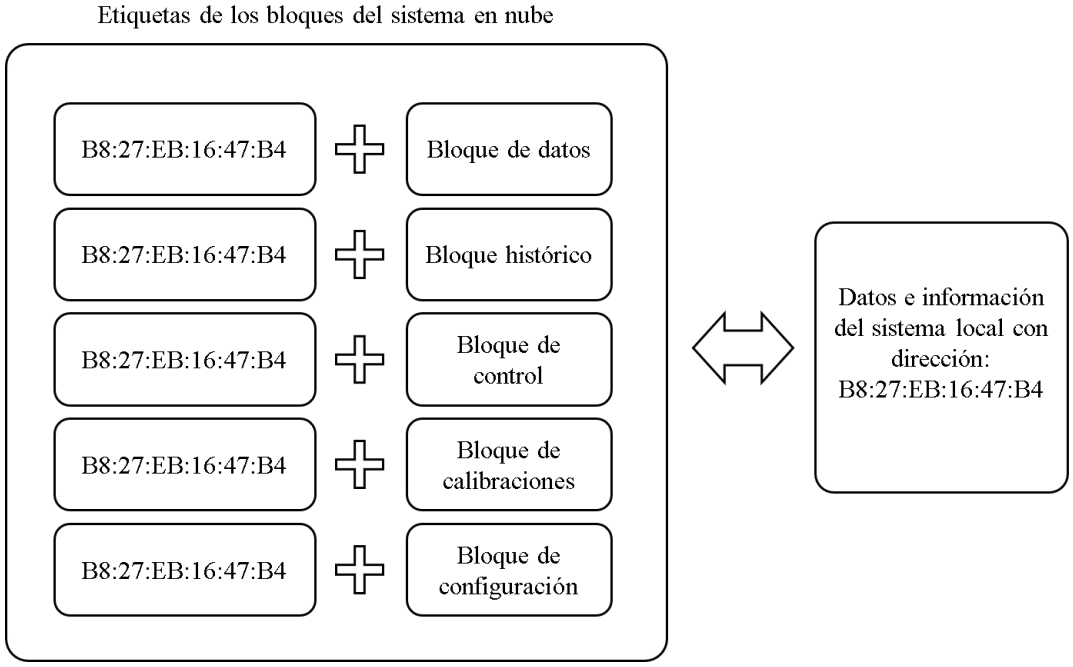


Figura 2.27: Acceso a los bloques del sistema en nube por medio de la MAC.

2.13. Dispositivo de supervisión remoto

El dispositivo de supervisión remoto constituye la última etapa del sistema. Cumple la función de intermediario entre el usuario y el sistema local permitiendo el acceso a los datos del cultivo y la funciones de control, configuración de riego y calibración de sondas. Para el diseño de esta etapa se tomó como hardware a los smartphones de plataforma Android por motivos de portabilidad y facilidad de uso. El software del dispositivo de supervisión es una aplicación desarrollada con el propósito de ser una interfaz de usuario para el acceso a la información almacenada en la base de datos de forma simple y organizada. La aplicación fue desarrollada en la plataforma online AppInventor de google, el cual trabaja de forma gráfica y por medio de diagrama de bloques. La interfaz de AppInventor se puede ver en la Fig.2.28 y la Fig.2.29.

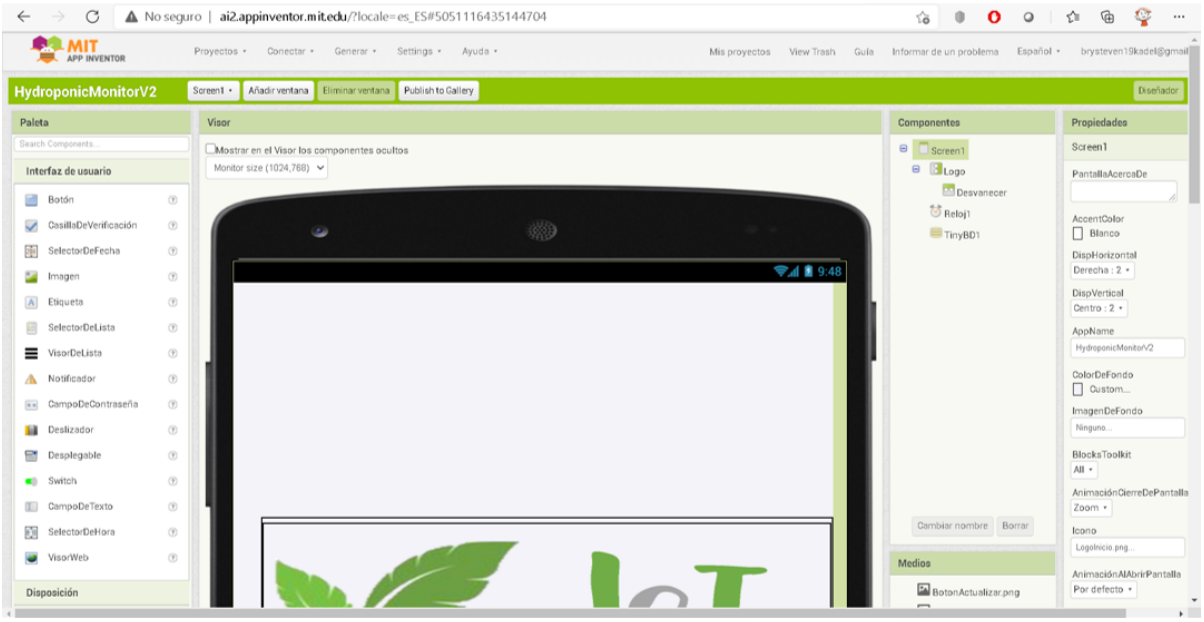


Figura 2.28: Interfaz de App Inventor

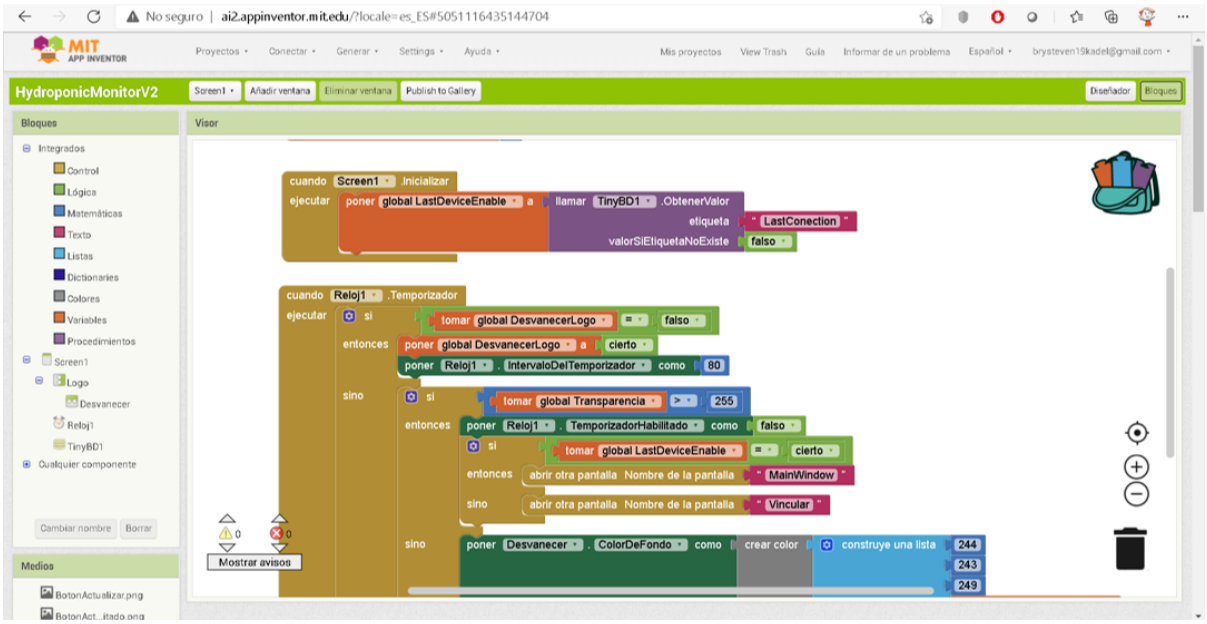


Figura 2.29: Programación en bloques de App Inventor.

2.13.1. Generalidades de la aplicación

La Fig.2.30 muestra la metodología del intercambio de información entre la base de datos y la aplicación de supervisión. La aplicación emplea como código de vinculación a la MAC del sistema local, esto debido a que todos los bloques funcionales del sistema ubicado en la nube emplean como prefijo esta dirección. Como se observa en la figura, el proceso de vinculación es un bucle donde se compara el código enviado con la lista de dispositivos registrados. Si no hay coincidencias, el dispositivo permanecerá en ese bucle, imposibilitando el acceso a la base de datos. Por otro lado, si el código ingresado es válido, se comprueba si otro dispositivo ya posee acceso a la dirección especificada. En caso de no existir acceso previo por otro dispositivo, es decir, dispositivos aún vinculados al sistema local en particular, la aplicación puede proceder al intercambio de información con la nube. De no ser así, se indica al usuario la presencia de un dispositivo vinculado y se mantiene en estado de espera por código de vinculación.

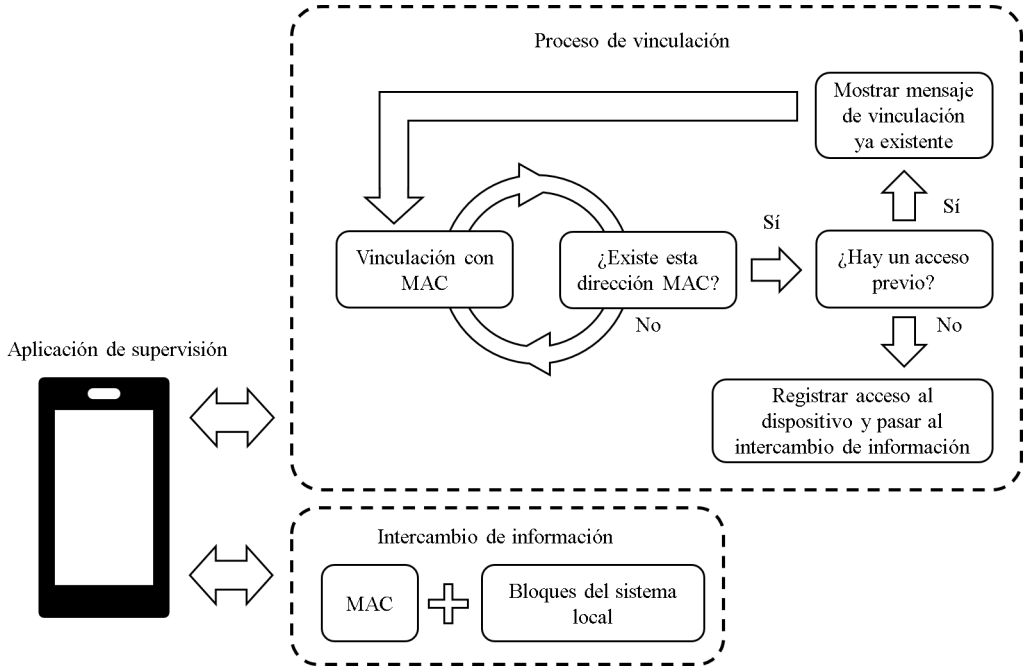


Figura 2.30: Metodología de la comunicación de la aplicación de supervisión con la nube.

Capítulo 3

Inplementación y resultados

3.1. Implementación del hardware

El hardware principal del sistema de supervisión se encuentra en el bloque local, el cual se instaló en la estructura física del cultivo hidropónico. Este fue implementado en un panel eléctrico de doble fondo el cual mide 30x30x20 cm. Para la sujeción de los módulos se hizo uso de tornillos de 3 mm con separadores de 5 mm para evitar el contacto directo con el panel. La distribución de los componentes se realizó considerando que los conectores de lo sensores, las salidas para la bomba y la entrada de alimentación se ubicaran en el lado inferior del panel, mientras que los elementos de visualización fueron ubicados en el lado izquierdo. La Fig.3.1 ilustra la implementación del hardware.

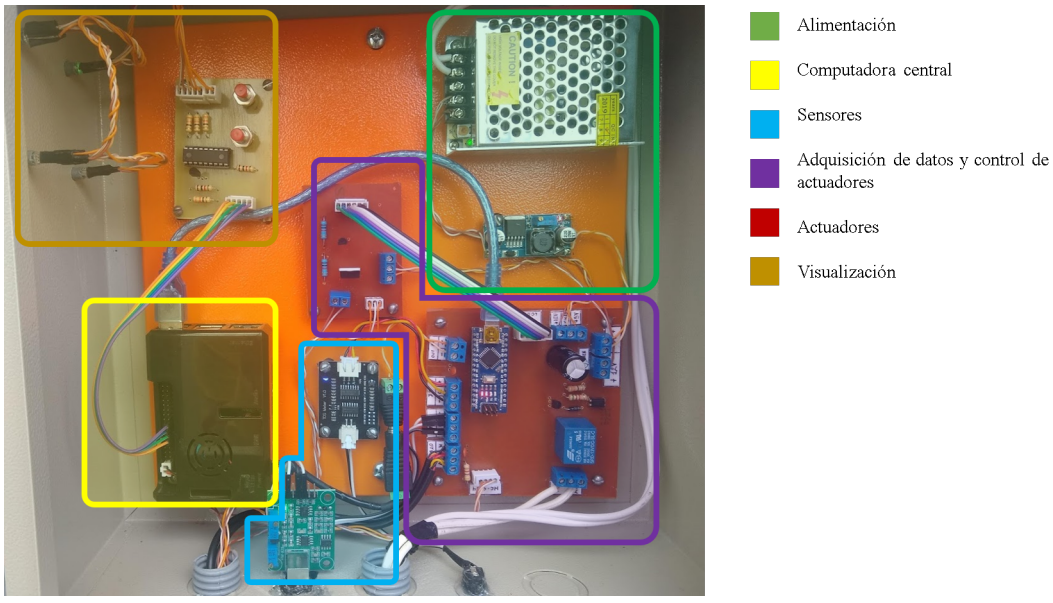


Figura 3.1: Sistema local de supervisión construido.

3.1.1. Conexión de entradas y salidas en la tarjeta principal

La Fig.3.2 muestra la conexión de las entradas y salidas en la tarjeta principal. Para ello se hace uso de conectores y borneras los cuales facilitan la instalación de los sensores de pH, de conductividad, nivel y de temperatura, así como la bomba principal y de pH. La tarjeta principal también realiza la distribución de voltajes de alimentación en todo el sistema a través de los conectores respectivos.

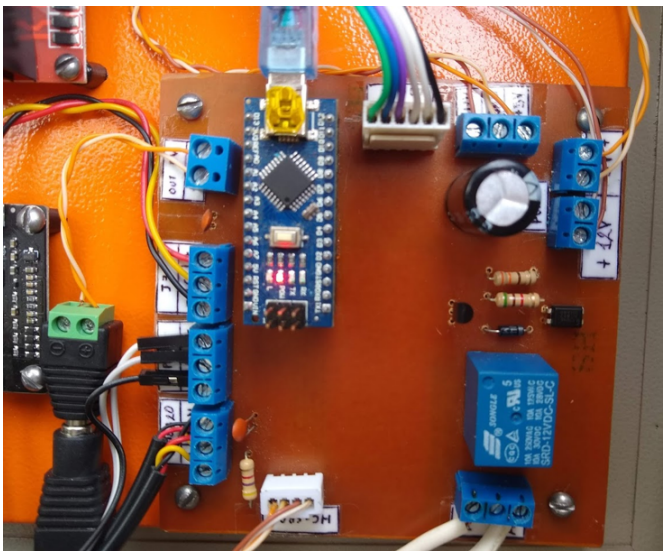


Figura 3.2: Conexiones de la tarjeta principal.

3.1.2. Visualización del funcionamiento del sensor de flujo

La tarjeta principal permite la visualización de los pulsos del sensor de flujo a través del LED integrado en la placa del microcontrolador.

Figura 3.3: Indicador de caudal.

3.1.3. Ubicación de sensores

Los sensores del sistema están distribuidos entre el tanque de solución y la estructura del cultivo. Dentro del tanque la sonda de pH se encuentra parcialmente sumergida en la solución. La sonda de conductividad, por otro lado, solo se conecta en el momento de medición para lo cual se extrae una muestra de solución para la toma de datos. En la tapa del tanque se encuentra instalado el sensor de nivel, mientras que la sonda de temperatura de solución esta totalmente

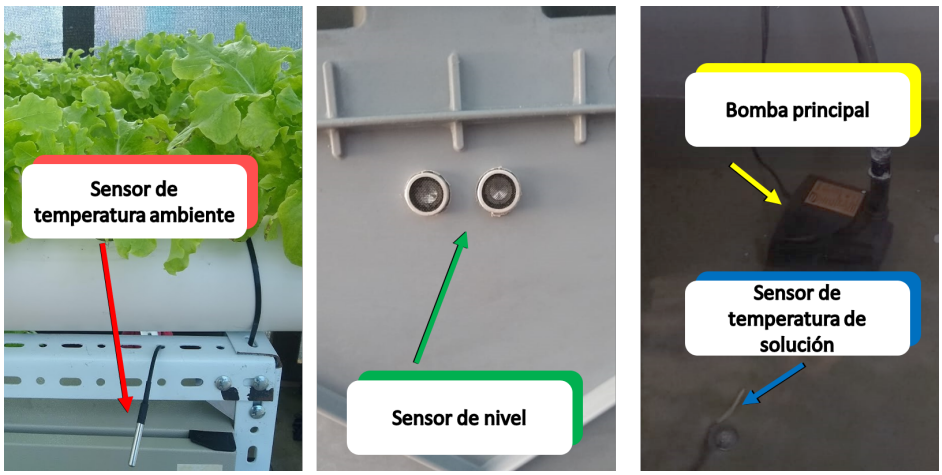


Figura 3.4: Sensores de temperatura ambiente, de solución y sensor de nivel.

La sonda de pH cuenta con un flotador con el fin de impedir que todo el dispositivo de medición se sumerja en la solución. Esto permite mantener la sonda dentro de la solución a medida que el nivel va disminuyendo debido al consumo

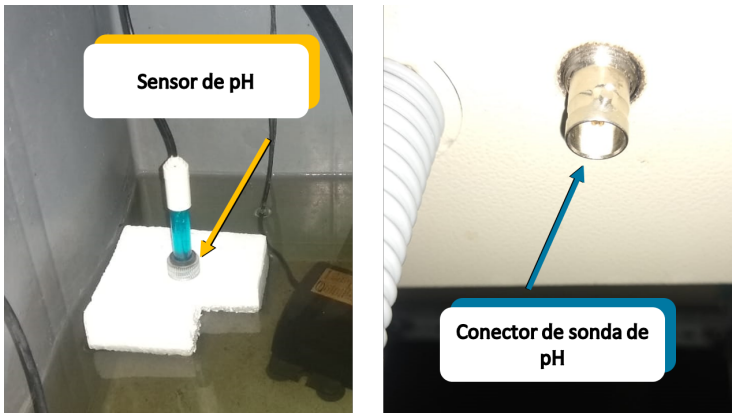


Figura 3.5: Conector y sonda para la medición de pH.

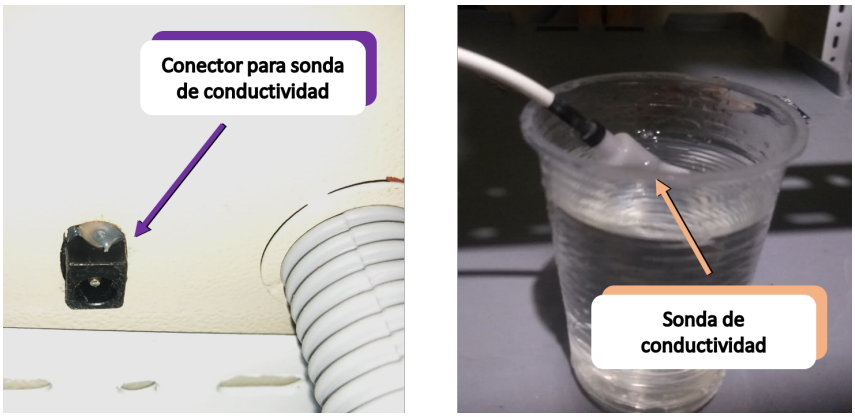


Figura 3.6: Conector y sonda para la medición de conductividad.

3.2. Consideraciones para el uso del sistema a largo plazo

3.2.1. AutoStart para inicio del programa principal

Esta configuración tiene por objetivo iniciar el programa de la computadora principal cuando el sistema se conecta a la red eléctrica, entrando en funcionamiento sin necesidad que el usuario tenga que realizar el proceso de inicio manualmente conectándose a la Raspberry por medios como SSH, VNC, monitor, teclado y mouse. Esta configuración hace uso de un archivo denominado “autostart” el cual contiene una serie de instrucciones para inicializar el programa, además de indicar el intérprete para la ejecución. El archivo se encuentra en la ruta “/home/pi/.config/lxsession/LXDE-pi”.

Figura 3.7: Dirección y archivo autostart.

La ejecución del programa por medio de AutoStart se puede observar en la Fig.3.8.

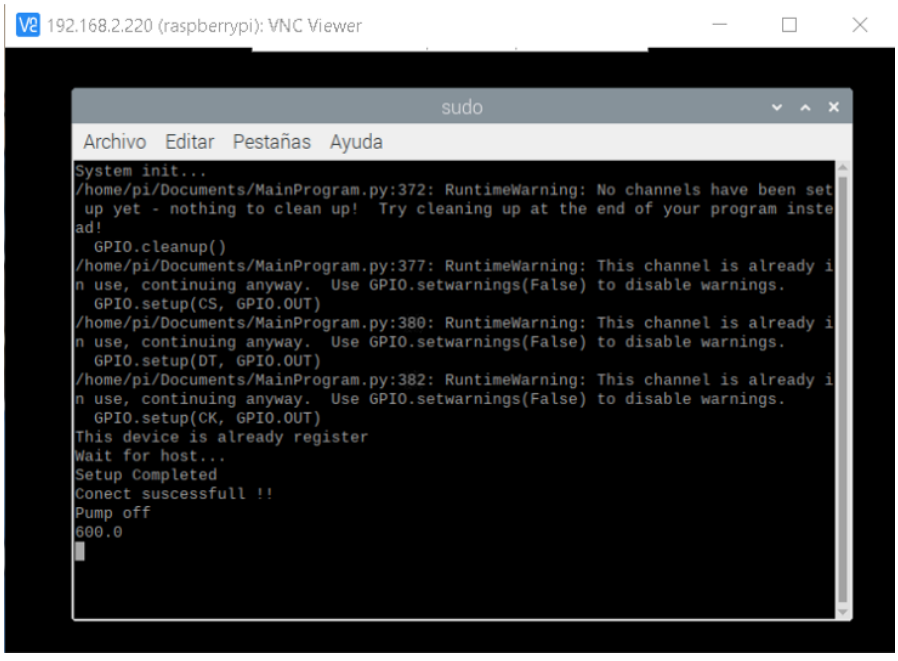


Figura 3.8: Ejecución del programa por medio de AutoStart.

Para garantizar el correcto funcionamiento de AutoStart, se debe insertar la línea del intérprete en el programa principal.

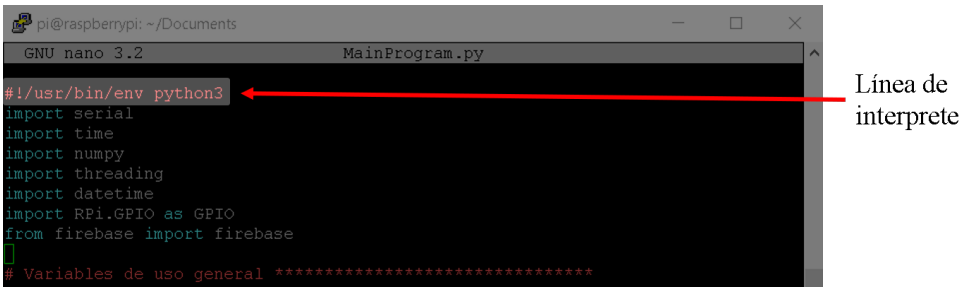


Figura 3.9: Línea de definición del interprete de ejecución.

3.2.2. IP WLAN estática

La IP WLAN estática es una configuración que establece una dirección local WIFI única e invariable. La ventaja que presenta es que en todo momento se conoce la dirección asignada a la Raspberry. La IP utilizada para la Raspberry corresponde a la dirección 192.168.2.220, mientras que la puerta de acceso al router es la dirección 192.168.2.1.

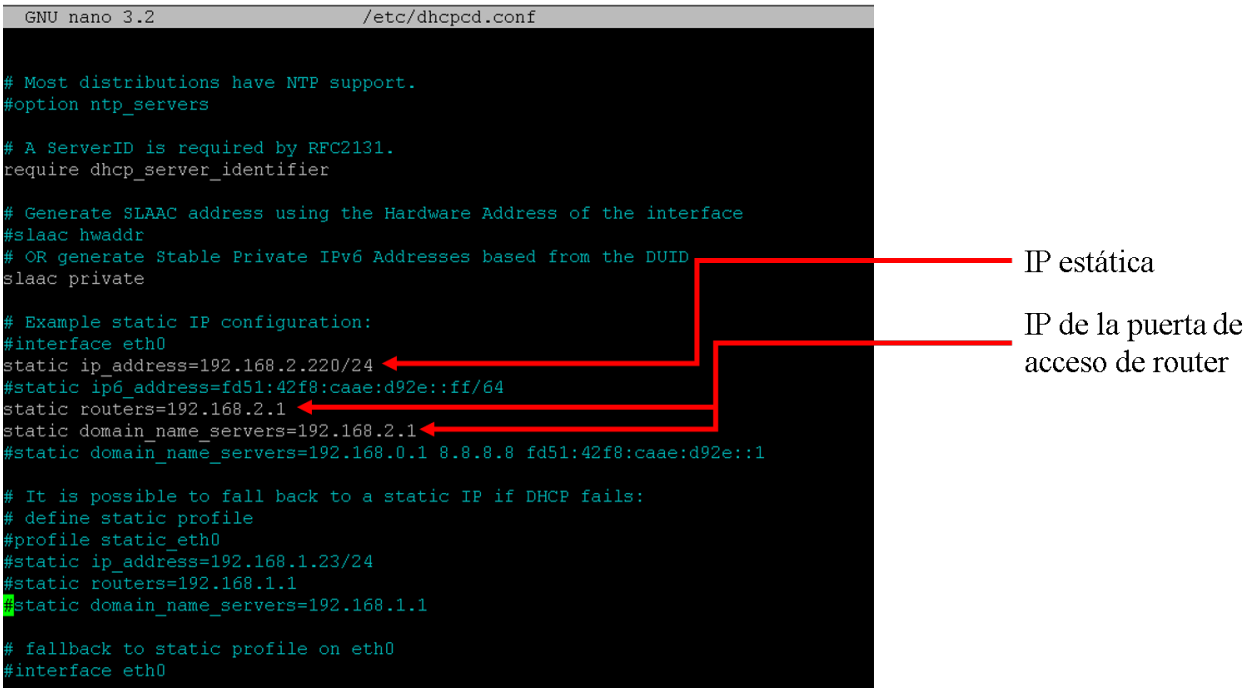


Figura 3.10: Configuración de la dirección IP estática.

3.2.3. Reconexión automática a la red WI-FI

A largo plazo, no se puede garantizar la continuidad de la red WIFI ya que puede darse el caso de que la red caiga por algún fallo. En una computadora o en un Smartphone, el sistema operativo es capaz de reconectarse a la red en cuanto detecte su presencia, sin embargo en la Raspberry no existe esta función, por lo que fue necesario implementarla.
Para la función de reconexión automática existe un archivo ejecutable creado en bash el cual contiene instrucciones que realizan la conexión a la red WIFI cuando el sistema ha detectado que no posee una dirección IP en la interfaz WLAN. La ejecución de este código se realiza una vez cada minuto a través del proceso Cron y el fichero crontab [46].

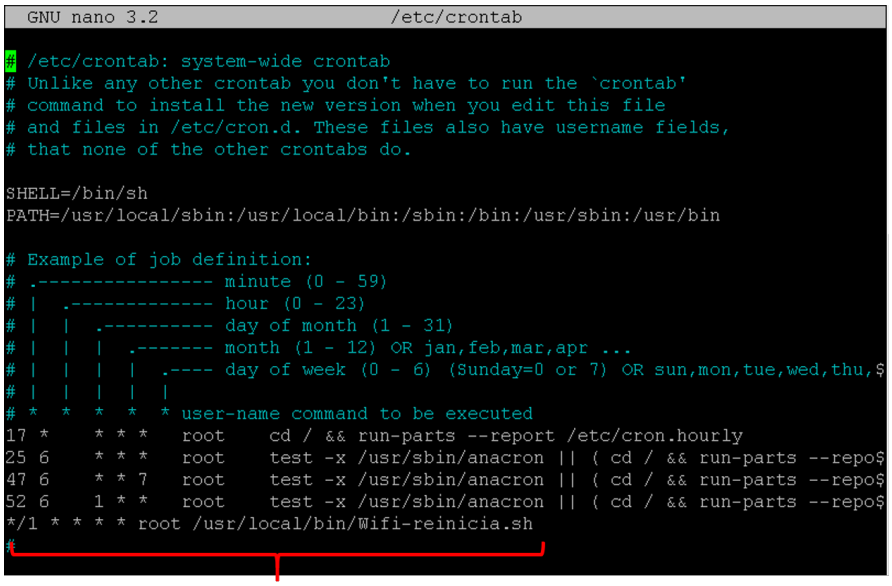


Figura 3.11: Archivo crontab y tarea de reconexión a la red WIFI.

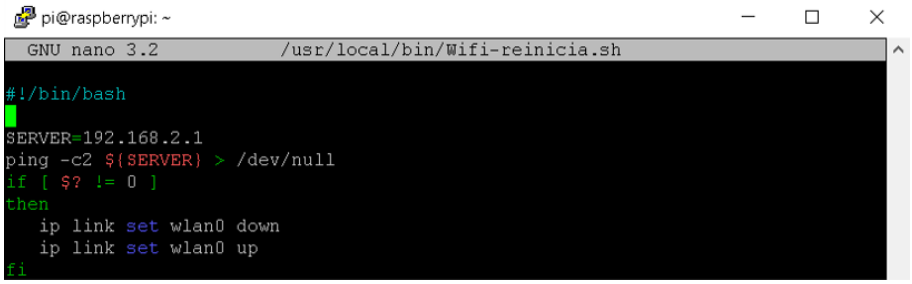


Figura 3.12: Programa bash de reconexión WIFI.

3.3. Implementación de la base de datos online

3.3.1. Bloque de control

Es el bloque encargado de coordinar el flujo de información de todos los bloques pertenecientes a un sistema local particular. Se desarrollo con el propósito de disminuir el volumen de información transferido entre el sistema local, la nube y el dispositivo de supervisión, permitiendo acceder a la información cuando esta es requerida. Consta de 4 parámetros encargados del control de actuadores y de la actualización de la información en el sistema local. En la Fig.3.13 se puede observar los parámetros del bloque de control.

- **StartPumpNow:** Función encargada de encender y apagar la bomba principal. Cuando el parámetro vale 1 la bomba se enciende, cuando vale 2 se apaga. Por defecto se encuentra en 0.
- **StartpHPump:** Función que realiza la dosificación de solución reguladora de pH en ml al cargar un valor al parámetro. Por defecto se encuentra en 0.
- **UpdateCalibration:** Parámetro que carga en el sistema local los datos almacenados en el bloque de calibración. Cuando el parámetro vale 1, se realiza la actualización. Por defecto se encuentra en 0.
- **UpdateSetup:** Parámetro que carga en el sistema local los datos actuales almacenados en el bloque de configuración. Cuando el parámetro vale 1, se realiza la actualización. Por defecto se encuentra en 0.

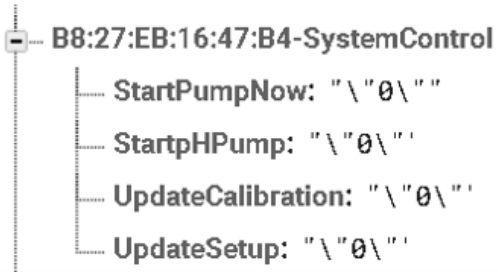


Figura 3.13: Árbol de directorio del sistema de control.

3.3.2. Bloques de datos

En este bloque se almacena la información general del sistema de riego, la información de los sensores y el estado de los actuadores. Los valores de este bloque se actualiza con una periodicidad de 1S. En la Fig.3.14 se puede observar los parámetros del bloque de datos.

- **ActualIntervalTime:** Variable que almacena el intervalo actual de riego del sistema.
- **Conductivy:** Variable que almacena el valor de conductividad medido en la solución.
- **EnvironmentTemp:** Variable que almacena el valor actual de la temperatura ambiente.
- **LevelTank:** Variable que almacena el nivel actual de solución en cm.
- **PorcentLevel:** Variable que almacena el nivel actual de solución en %.
- **PumpState:** Variable que almacena el estado actual de la bomba principal. Cuando el parámetro señala "ON", indica que la bomba esta encendida. Cuando señala "OFF", la bomba esta apagada.
- **Sleep:** Variable que almacena el estado general del riego. Cuando el parámetro señala "YES", indica que se ha finalizado el ciclo de riego diario. Si esta en "NO", se están realizando ciclos de riego.
- **TempSolution:** Variable que almacena la temperatura actual de la solución en °C.
- **pHSolution:** Variable que almacena el valor de pH actual de la solución.

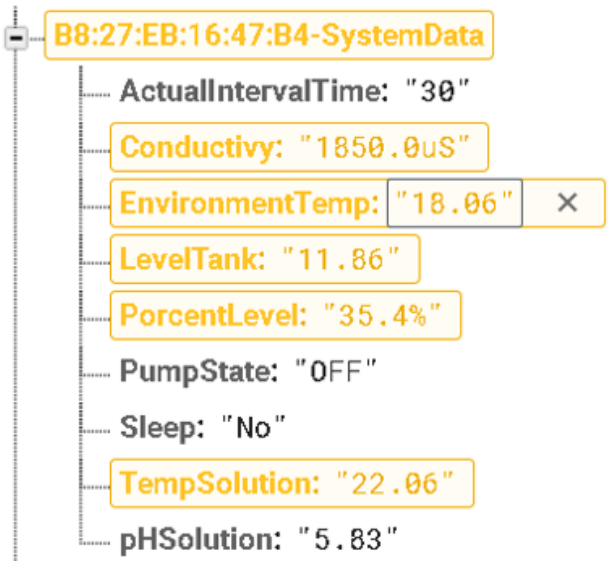


Figura 3.14: Árbol de directorio de los datos del sistema.

3.3.3. Bloques de configuración

Es el bloque que almacena la información de configuración del sistema local. En la Fig.3.15 se puede observar los parámetros que conforman este bloque.

- **HighTempIntervalTime:** Variable de configuración que indica un nuevo intervalo de riego cuando la temperatura supera el valor indicado en WTCTempMax.
- **IntervalTime:** Variable de configuración que almacena el intervalo de riego para el sistema.
- **SleepTime:** Variable de configuración que indica al sistema la hora de finalización de los ciclos de riego.
- **StartTime:** Variable de configuración que indica al sistema la hora de inicio de los ciclos de riego.
- **TankHeight:** Variable de configuración que indica al sistema la altura total del tanque que almacena la solución.
- **WTCTempMax:** Variable de configuración que almacena el valor de temperatura superior límite, tras el cual, se cambia el intervalo de riego al señalado por HighTempIntervalTime.
- **WTCTempMin:** Variable de configuración que almacena el valor de temperatura inferior límite, bajo el cual, se restablece el intervalo de riego al señalado por IntervalTime.
- **WateringTime:** Variable de configuración que almacena la duración del riego del sistema.

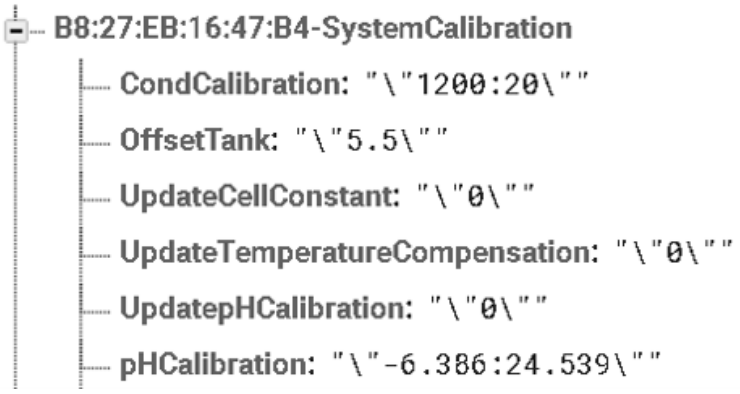


Figura 3.15: Árbol de directorio de la configuración del sistema.

3.3.4. Bloques de calibraciones

Es el bloque que almacena la información de calibración para los sensores y el tanque del sistema local. Esta formado por los parámetros que se pueden observar en la Fig.3.16.

- **CondCalibration:** Variable de calibración que almacena el valor de conductividad de referencia a la temperatura especificada.
- **OffsetTank:** Variable de calibración que indica el valor de offset existente entre el límite máximo del nivel del tanque y la altura real a la que se encuentra el sensor de nivel.
- **UpdateCellConstant:** Variable de calibración que señala al sistema que es necesario actualizar la constante de celda de conductividad. La función que realiza tal tarea solo es llevada a cabo únicamente al habilitarse la actualización del bloque de calibración por medio de "UpdateCalibration", después de habilitar este parámetro. Cuando la propiedad vale 1, se habilita la actualización de celda de conductividad. Por defecto se encuentra en 0.
- **UpdateTemperatureCompensation:** Variable de calibración que señala al sistema que es necesario actualizar la constante de compensación de temperatura. La función que realiza tal tarea solo es llevada a cabo únicamente al habilitarse la actualización del bloque de calibración por medio de "UpdateCalibration", después de habilitar este parámetro. Cuando la propiedad vale 1, se habilita la actualización de constante de compensación de temperatura. Por defecto se encuentra en 0.
- **pHCalibration:** Variable de calibración que almacena los coeficientes de linealización de la sonda de pH.Variable de calibración que señala al sistema que es necesario actualizar la constante de celda de conductividad. La función que realiza tal tarea solo es llevada a cabo únicamente al habilitarse la actualización del bloque de calibración por medio de "UpdateCalibration", después de habilitar este parámetro. Cuando la propiedad vale 1, se habilita la actualización de celda de conductividad. Por defecto se encuentra en 0.

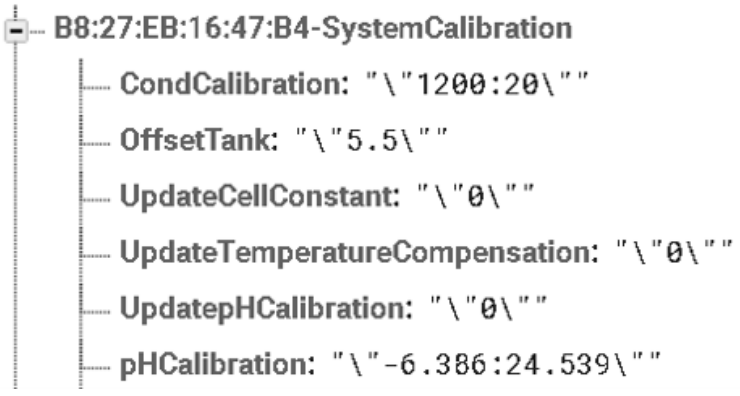


Figura 3.16: Árbol de directorio de las calibraciones del sistema.

3.3.5. Bloques histórico

Es un bloque que almacena información del sistema y de los sensores cada hora en punto con el propósito de crear un registro histórico que permita realizar un seguimiento a la evolución del cultivo durante todo su ciclo. En la Fig.3.17 se puede observar la estructura establecida en este bloque, la cual consiste en una clasificación general por año, seguida del día y mes en una sola etiqueta y por último la información por hora. Estos datos están almacenados en formato JSON y pueden ser exportados a un archivo de extensión “csv”. Los datos almacenados son los siguientes:

- **Conductivy**
- **EnvironmentTemp**
- **LevelTank**
- **PorcentLevel**
- **TempSolution**
- **pHSolution**

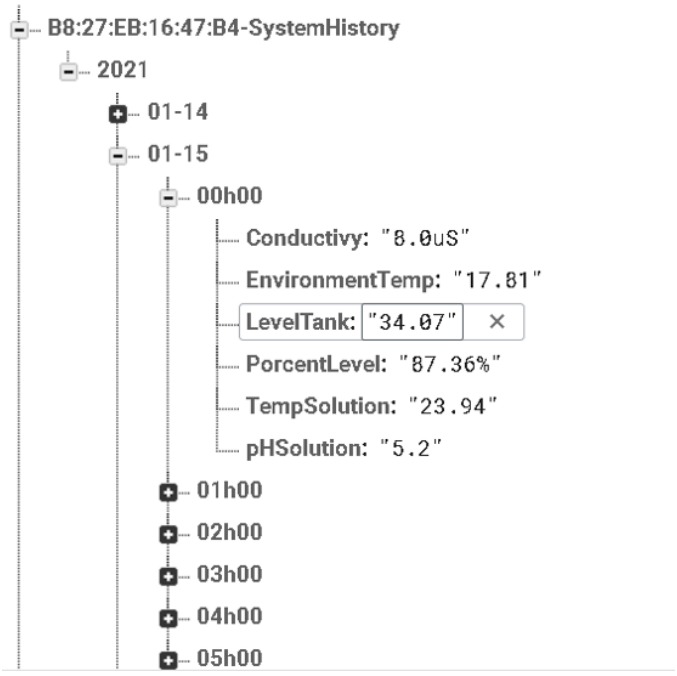


Figura 3.17: Árbol de directorio del registro histórico.

3.3.6. Evolución de la base de datos online

Durante todo el desarrollo del proyecto, la base de datos online ha sufrido una serie de modificaciones con el propósito de mejorar su desempeño para el sistema de supervisión. La primera versión de la base de datos estaba conformada por dos "Keys". En la primera de ellas denominada "System", se encontraban agrupados los datos de los sensores, los datos de configuración y los parámetros para el control de actuadores, mientras que en la segunda, "Systemhistory", se empleaba para almacenar datos históricos. Este primer formato no era eficiente puesto que el sistema local descargaba toda la información almacenada en la nube cada vez que intentaba leer algún parámetro de la base de datos, dando como resultado un pico en descargas de información debido a la transferencia de información diaria. Como se puede ver en la Fig.3.18 durante los días 21, 22, 23 y 23 correspondientes al período de pruebas de la base de datos se tenía un uso acumulado de 613.6 Mb en descargas de información. Para el día 25, fecha para la cual, el sistema en nube ya había sido puesto en marcha, se registro un gran pico de descargas, el cual rondaba alrededor de los 700 Mb. Por consiguiente, fue introducida una modificación en la base de datos, el cual consistía en reorganizar toda su estructura a bloques para el almacenamiento y gestión de información, destinando un único bloque de tamaño reducido al control de las descargas de información.

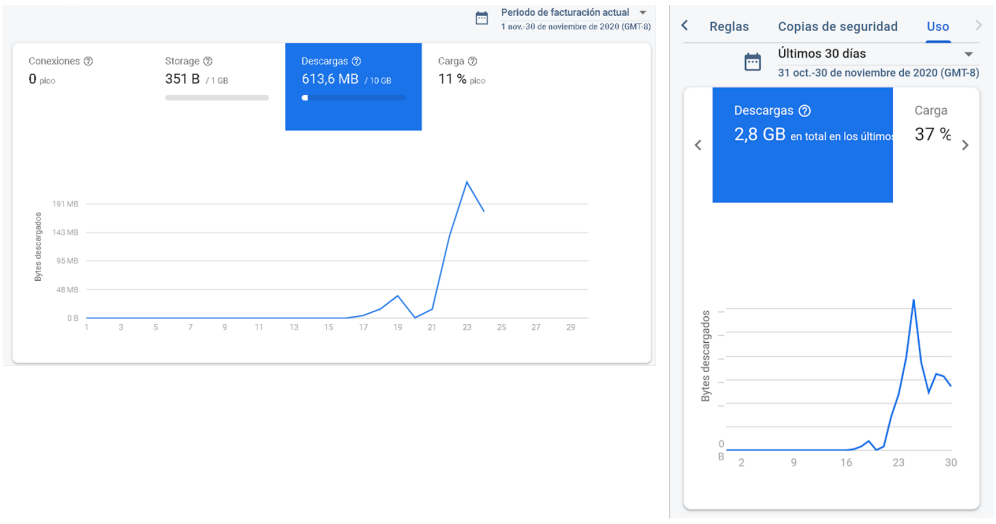


Figura 3.18: Descarga de datos durante el período de prueba y puesta en marcha de la base de datos online.

Tras la implementación de los nuevos bloques para datos, configuración y calibración las descargas diarias se reducieron hasta alcanzar un valor que ronda cerca de los 300 Mb sin llegar a superarlos. Mensualmente, las descargas acumulan un valor de descargas alrededor de 8 a 9 Gb de datos, lo cual esta cerca del límite de 10 Gb impuesto por el servicio gratuito de Firebase. La Fig.3.19 muestra el uso de descargas para el período 1 de marzo a 31 de marzo del 2021.

Figura 3.19: Descarga de datos mensuales.

La última modificación en la estructura de la base de datos llegó con la introducción del sistema de registro automático de dispositivos. Implementando un bloque de dispositivos y la autenticación con la MAC. La versión de esta base de datos es mostrada en la Fig.3.20.

Figura 3.20: Evolución de la base de datos en el transcurso del proyecto. Izquierda: Primera versión de la base de datos. Centro: Segunda versión de la base de datos. Derecha: Versión final de la base de datos.

3.4. Implementación de la aplicación de supervisión para la plataforma Android

3.4.1. Navegación principal y funciones generales

La navegación a través de la aplicación móvil se realiza a través de una barra de menú el cual posee 5 botones que dan acceso de forma organizada a todas las características de control, supervisión y configuración que puede realizar la aplicación. Las funcionalidades de cada botón se pueden observar en la Fig.3.21. La barra de menú es accesible desde cualquier pantalla de la aplicación a excepción de la pantalla de vinculación.

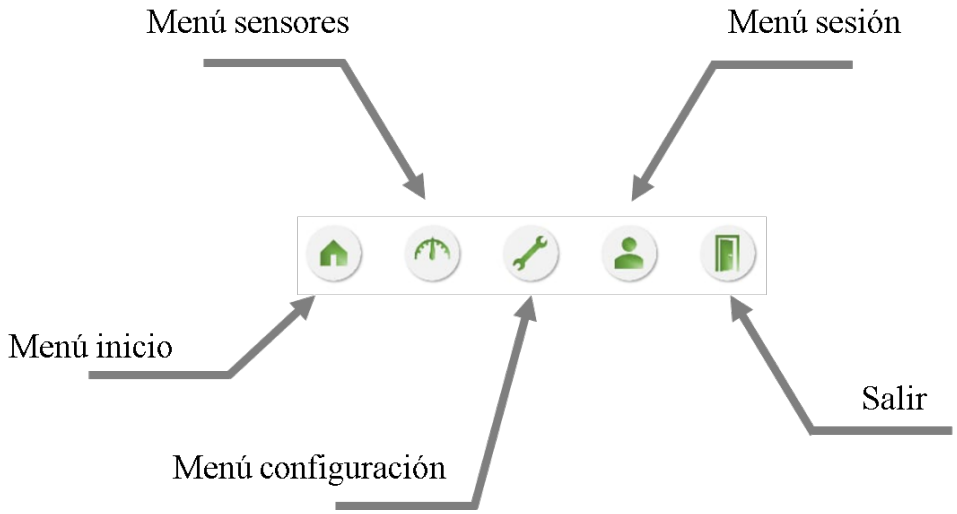


Figura 3.21: Funciones de la aplicación.

3.4.2. Pantalla de vinculación

Corresponde a la primera ventana que se muestra al iniciar por primera vez la aplicación. Tiene por objetivo permitir al usuario acceder aun sistema local por medio del código de vinculación, el cual no es más que la MAC del adaptador WIFI de la Raspberry Pi. La pantalla de vinculación se puede observar en la Fig.3.22. La extensión del código de vinculación es de 12 caracteres, agrupados en pares separados por medio del símbolo ":". Esta ventana se mantiene hasta que el usuario presione el botón de vincular, el cual inicia el procedimiento de conexión al sistema local. En caso de ocurrir algún inconveniente, este se indica por medio de un mensaje al usuario.

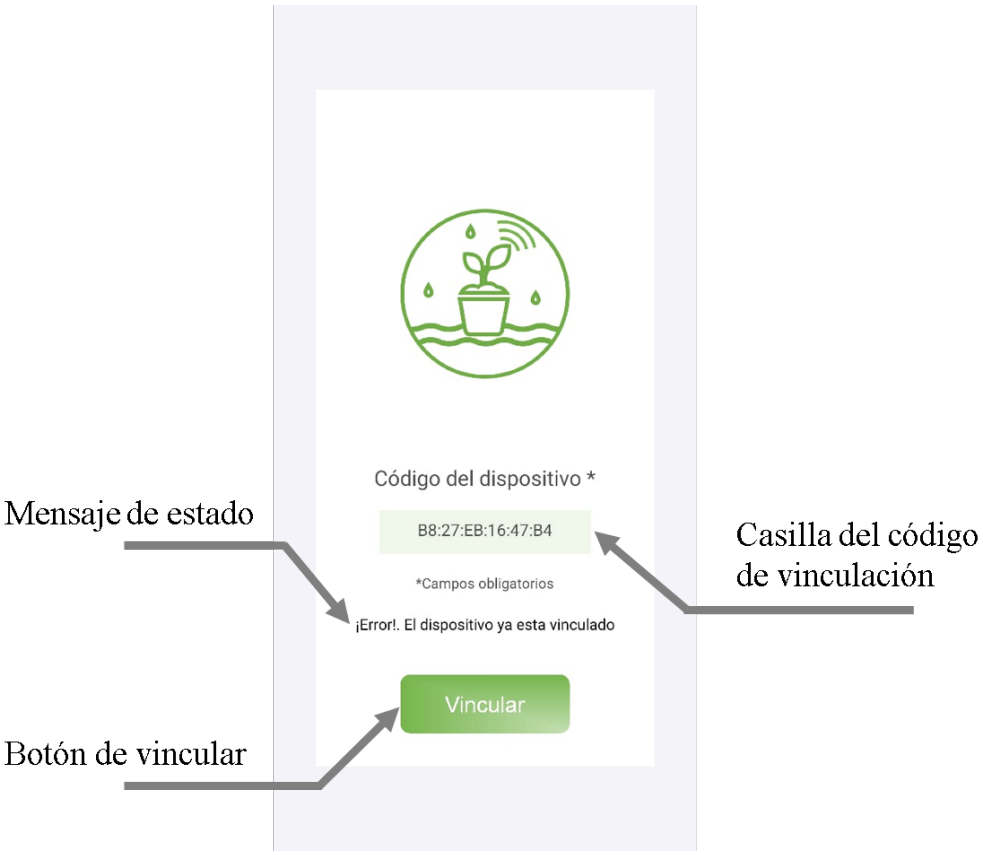


Figura 3.22: Pantalla de vinculación y elementos de su interfaz gráfica.

Una vez vinculado el dispositivo, no vuelve a mostrarse esta pantalla hasta que el usuario se desvincule de manera manual. Dependiendo del caso, pueden suceder 3 situaciones:

- El dispositivo se vincula correctamente
- El dispositivo al que se desea vincular no existe
- El dispositivo al que se desea vincular ya esta vinculado

Figura 3.23: Casos de vinculación de la aplicación. Izquierda: El dispositivo al que se intenta acceder ya esta vinculado. Derecha: El dispositivo al que se intenta acceder no existe.

3.4.3. Pantalla de inicio

La pantalla de inicio es una ventana principal que se muestra al usuario posteriormente al proceso de vinculación y al iniciar la aplicación. Presenta información general del sistema de forma resumida y permite realizar acciones sobre los actuadores. La estructura de esta ventana es la que se puede observar en la Fig.3.24.

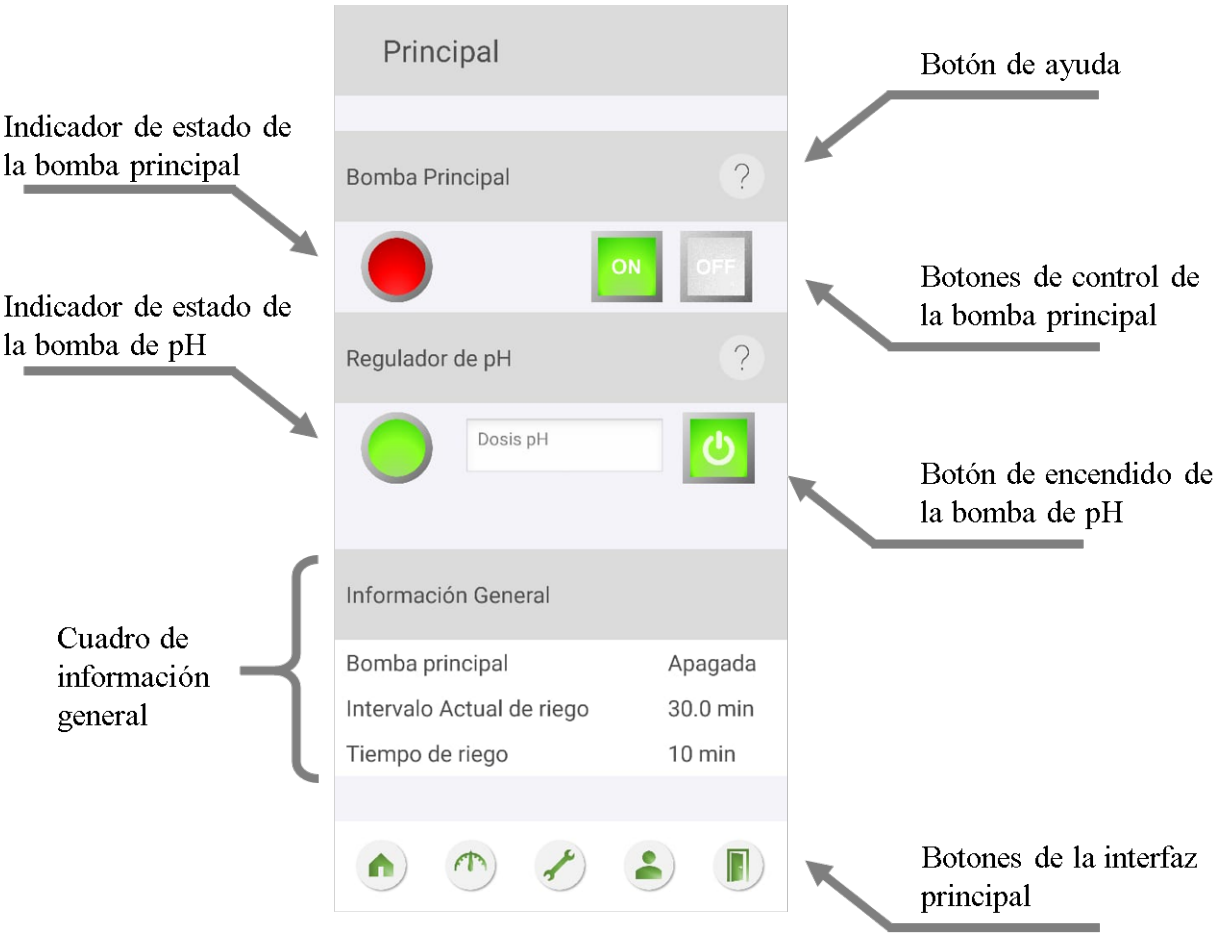


Figura 3.24: Pantalla de inicio y elementos de su interfaz gráfica.

3.4.4. Pantalla de sensores

La pantalla de sensores presenta al usuario la información de todos los sensores que posee el sistema local. El acceso a estos datos se realiza a través de los botones ubicados en la barra de sensores que se puede observar en la Fig.3.25, los cuales activan una subpantalla con la información de sensor correspondiente.

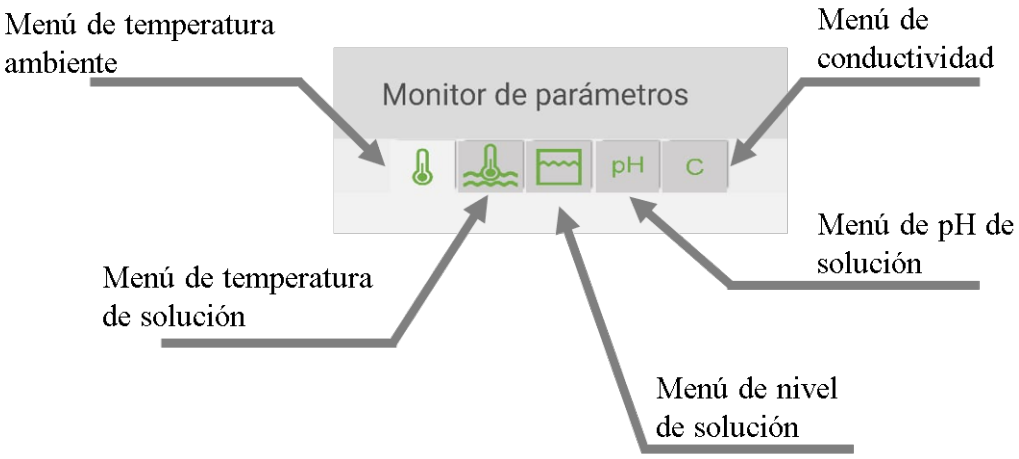


Figura 3.25: Barra de sensores.

Las Fig.3.26, Fig3.27, Fig3.28 y Fig3.29 muestran el diseño de la interfaz gráfica de cada subpantalla construida para los sensores. Los datos mostrados tienen un intervalo de actualización de 5 segundos con el objetivo de disminuir la descarga de información. Las subpantallas que se posee la pantalla de sensores son las siguientes:

- Pantalla del sensor de temperatura del ambiente
- Pantalla del sensor de temperatura de la solución
- Pantalla del sensor de nivel
- Pantalla del sensor de pH
- Pantalla del sensor de conductividad

En la Fig.3.26 se observa las subpantalla para visualizar la temperatura ambiente y la temperatura de solución, las cuales poseen una interfaz similar. La temperatura ambiente y de solución se muestra en grados centígrados.

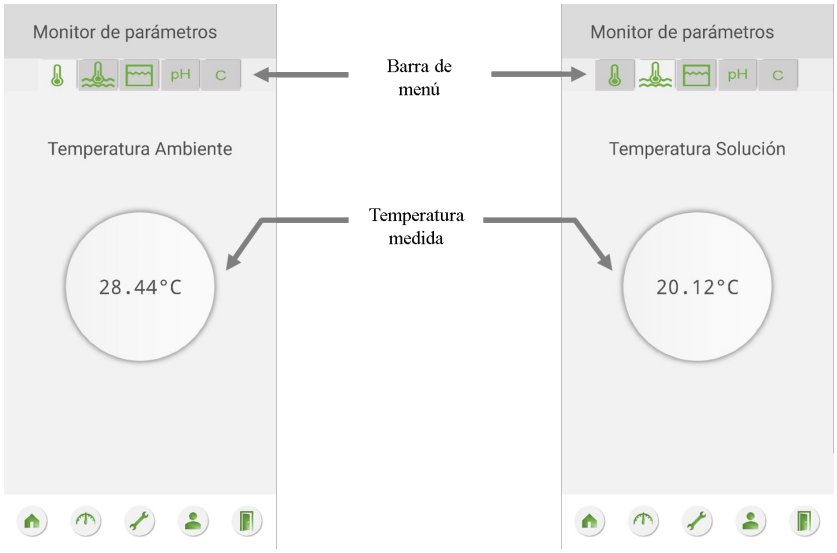


Figura 3.26: Pantallas de visualización para la temperatura ambiente y de solución.

La interfaz para la visualización del nivel es mostrada en la Fig.3.27. En ella se muestra al usuario a información del nivel en forma gráfica y textual, utilizando valores de porcentaje. La ventana también indica el estado del tanque a través de un indicador luminoso con el formato del LED correspondiente al nivel.

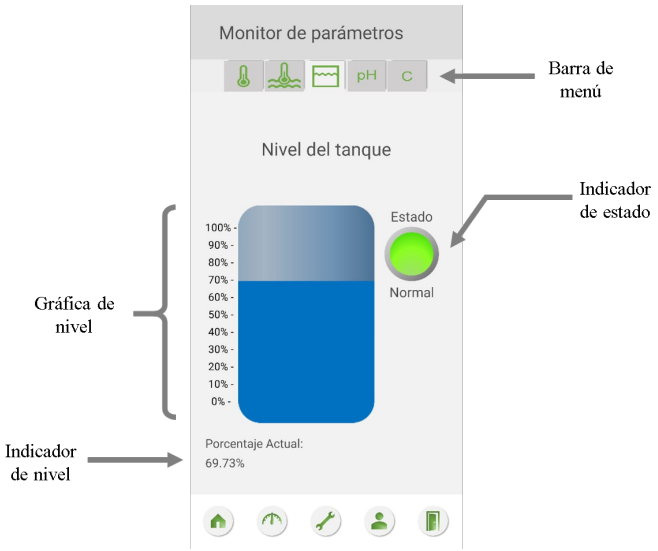


Figura 3.27: Pantalla de visualización del nivel de solución.

Las interfaces de las subpantallas que muestran los valores de conductividad y pH se pueden observar en las Fig. 3.28 y Fig.3.29. Estas ventanas además indican de forma adicional la temperatura de la solución a la cual se está realizando la medición.

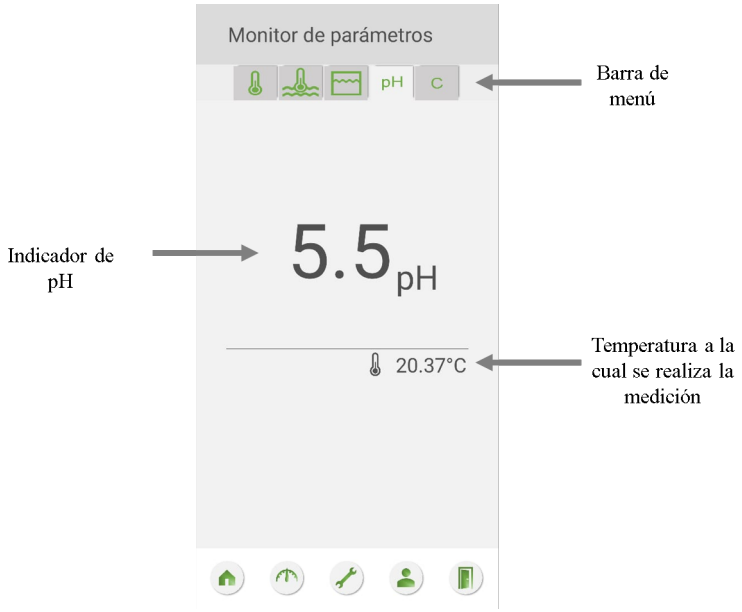


Figura 3.28: Pantalla de visualización del pH de solución.

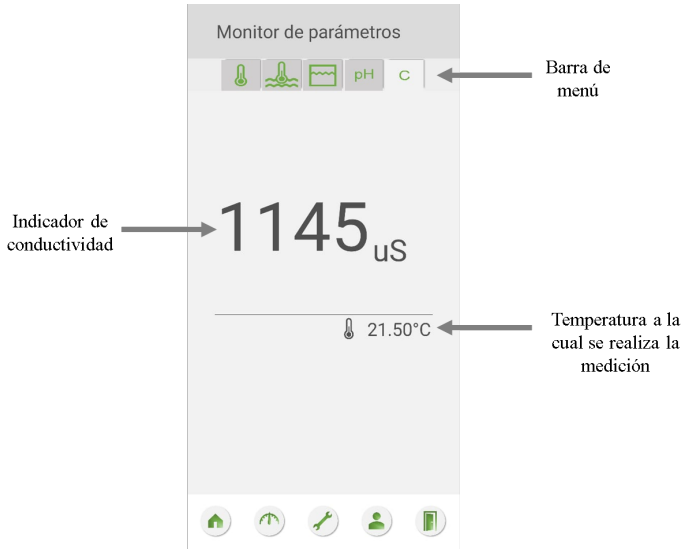


Figura 3.29: Pantalla de visualización de conductividad de solución.

3.4.5. Pantalla de configuraciones

Es la pantalla encargada de ayudar al usuario a realizar todas las configuraciones del sistema. La interfaz integra cuatro botones desde los cuales se pueden acceder a los ajustes de las funciones de ciclos de riego, funciones especiales y calibraciones de sondas.

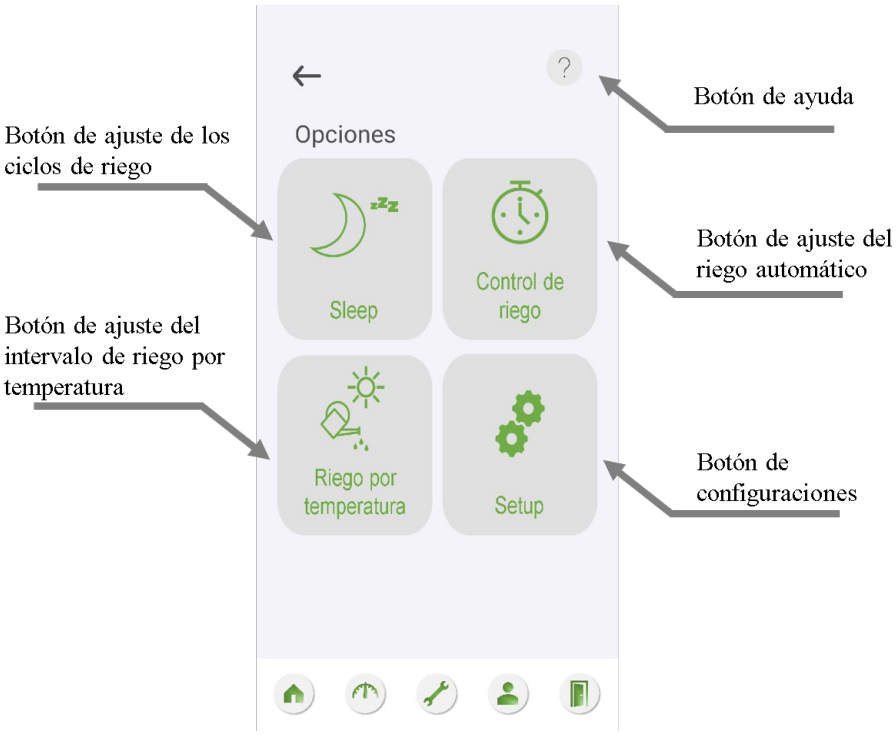


Figura 3.30: Pantalla de configuración.

- **Sleep. -**Es la función que ajusta la hora de inicio y la hora de finalización de los ciclos de riego. El intervalo después de la hora de finalización corresponde al período donde no se activara la bomba principal de forma automática. Esta función no afecta el control manual, por lo cual es posible activar la bomba desde la pantalla de inicio.
- **Control de riego. -**Función que se encarga de ajustar el tiempo e intervalo de riego para el cultivo hidropónico. El ajuste de los tiempos se realiza con intervalos de 5 min y se actualiza en la base de datos y el sistema local por medio del botón “Actualizar”.
- **Riego por temperatura. -**Es una función que reduce el intervalo de riego del cultivo hidropónico cuando la temperatura ambiente supera un valor establecido por “T°C Máx”. El nuevo intervalo de riego puede ser configurado por el valor del parámetro “Riego” y es mantenido hasta que la temperatura ambiente disminuya por debajo de valor de “T°C Min”. Para cargar los ajustes realizados por el usuario, se debe dar clic sobre el botón “Actualizar”.

- **Setup. -**Este botón abre las opciones que permiten al usuario realizar las calibraciones del sistema.

Las subpantallas que se inician al dar clic a cada botón se pueden observar en las siguientes figuras.

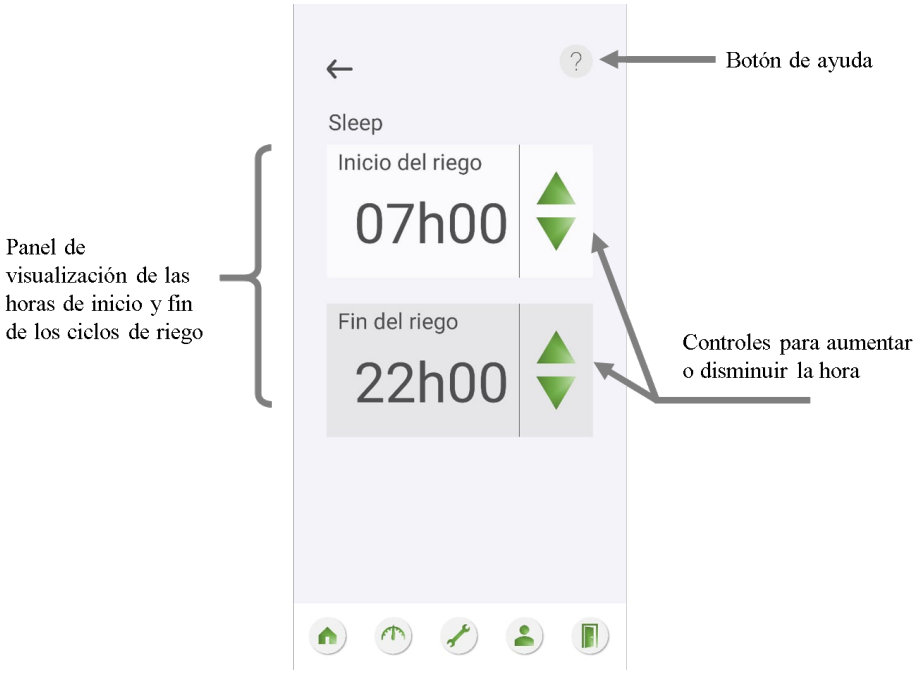


Figura 3.31: Pantalla de Sleep (Configuración de ciclos de riego)

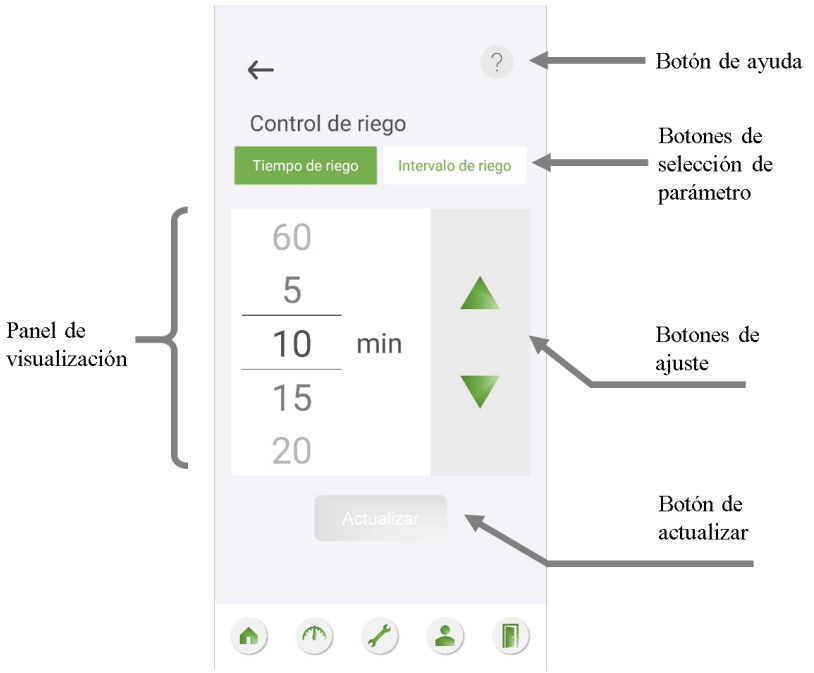


Figura 3.32: Pantalla de ajuste del riego automático.

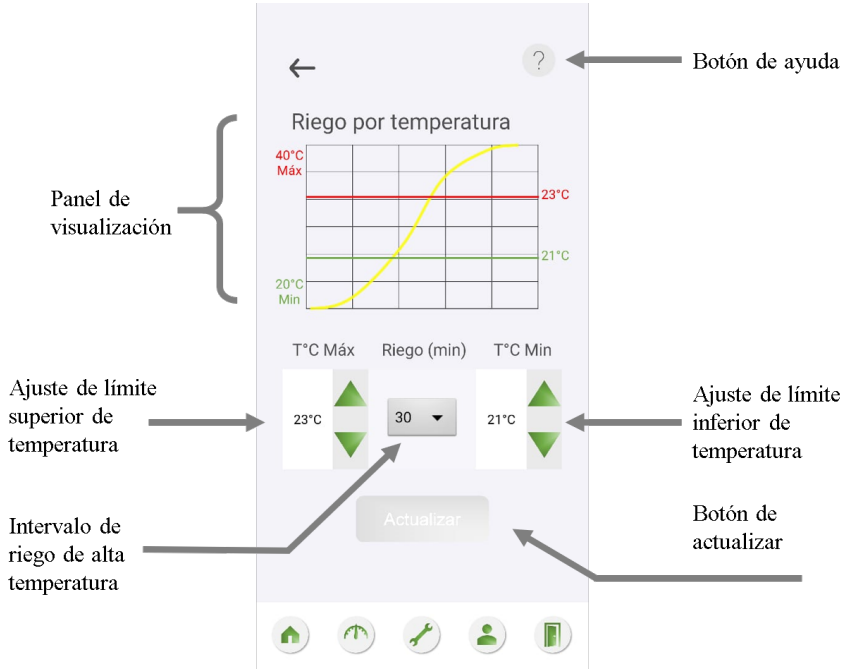


Figura 3.33: Pantalla de ajuste del riego por temperatura.

El botón “Setup” contiene un menú que permite al usuario calibrar la sonda de pH, la sonda de conductividad, ajustar la referencia de conductividad e ingresar parámetros del tanque de la solución. La Fig.3.34 muestra esta pantalla.

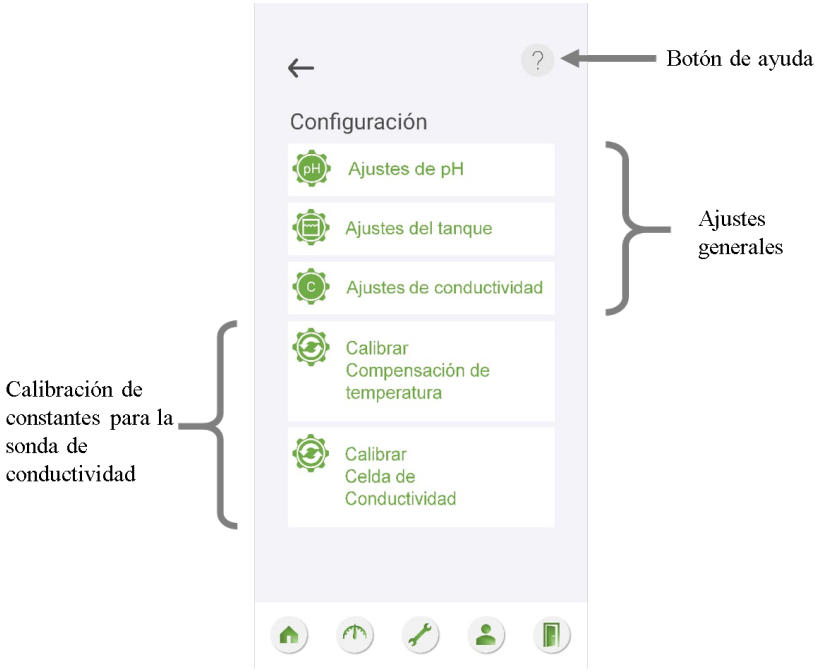


Figura 3.34: Pantalla de Setup.

- **Ajustes de pH. -** Es una ventana que permite calibrar la sonda de pH por medio de una función lineal. La ecuación empleada tiene la forma de y = mx + b, la cual es ingresada en el sistema en forma de los parámetros de ganancia y offset. La interfaz posee una función que permite visualizar el valor del voltaje medido y su equivalencia en pH con los factores de conversión que estén seteados.

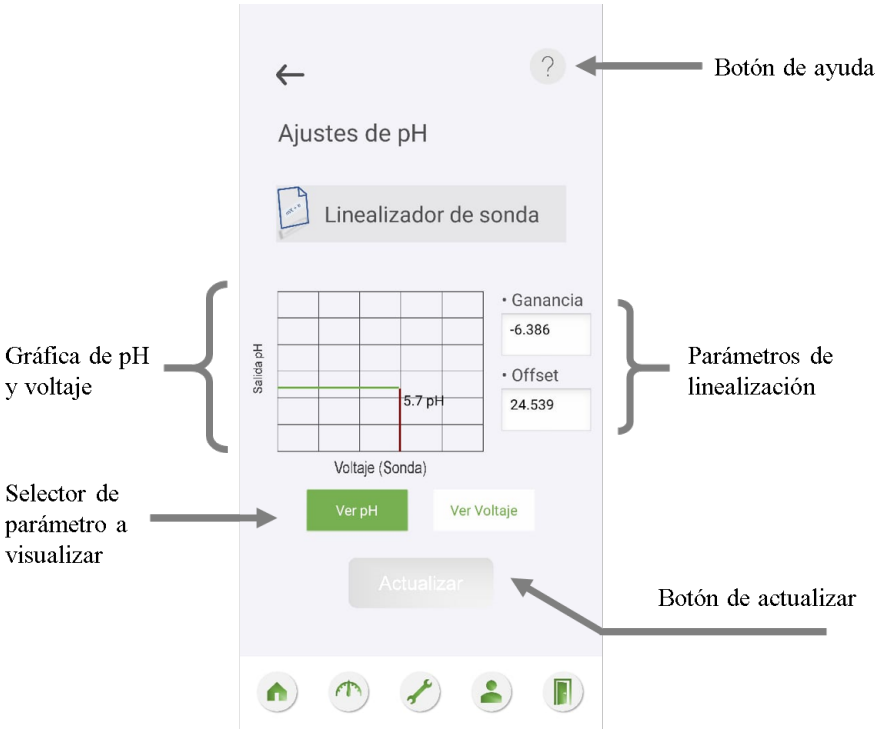


Figura 3.35: Pantalla de ajustes de pH.

- **Ajustes del tanque. -**Ventana que permite ingresar la altura del tanque y la ubicación del sensor para calcular el nivel de solución.

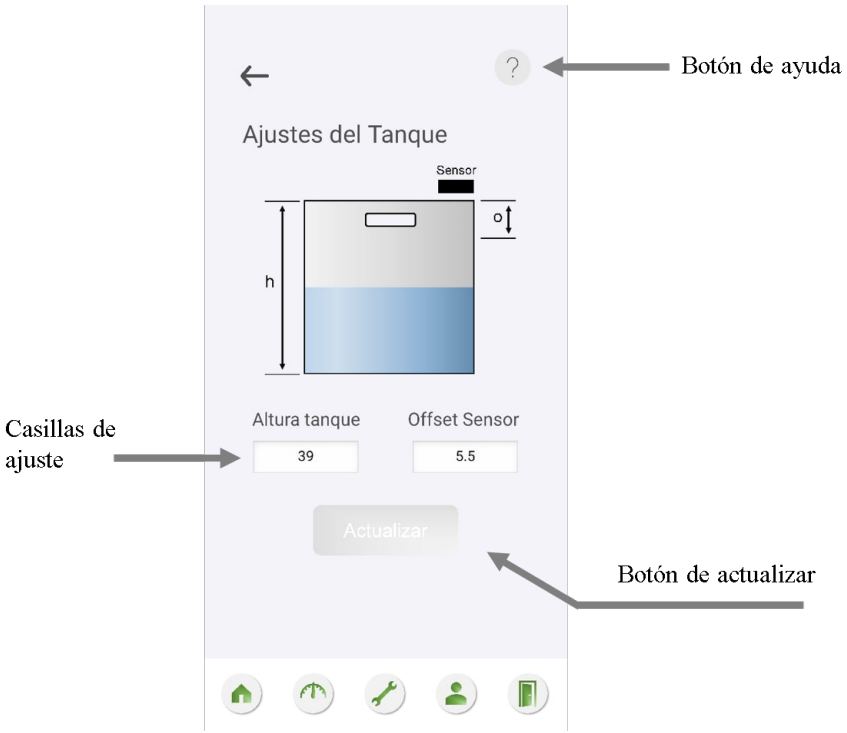


Figura 3.36: Pantalla de ajustes de pH.

- **Ajustes de conductividad. -**Ventana que permite ajustar la conductividad y temperatura de referencia para la solución nutritiva.

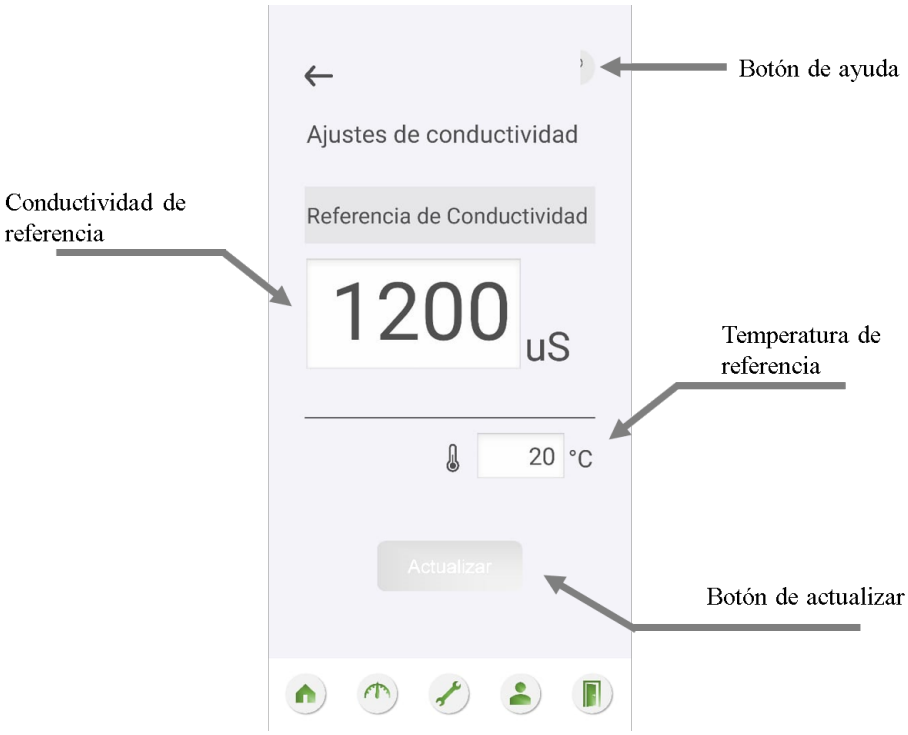


Figura 3.37: Pantalla de ajustes de conductividad.

- **Calibrar compensación de temperatura. -**Es un botón que habilita la función de calibración de la constante de compensación de temperatura en el sistema local. La anterior calibración es reemplazada por lo que no es posible recuperarla una vez ejecutada la función.

- **Calibrar celda de conductividad. -**Es un botón que habilita la función de calibración de la constante de celda de la sonda de conductividad en el sistema local. La anterior calibración es reemplazada por lo que no es posible recuperarla una vez ejecutada la función.

3.4.6. Pantalla de sesión

La pantalla de sesión es una ventana que permite visualizar información de la conexión actual. Presenta la opción de finalización de conexión activa con la cual el dispositivo de supervisión puede conectarse con otro código de vinculación a otro sistema local. La Fig.3.38 se puede observar la interfaz de la pantalla de sesión.

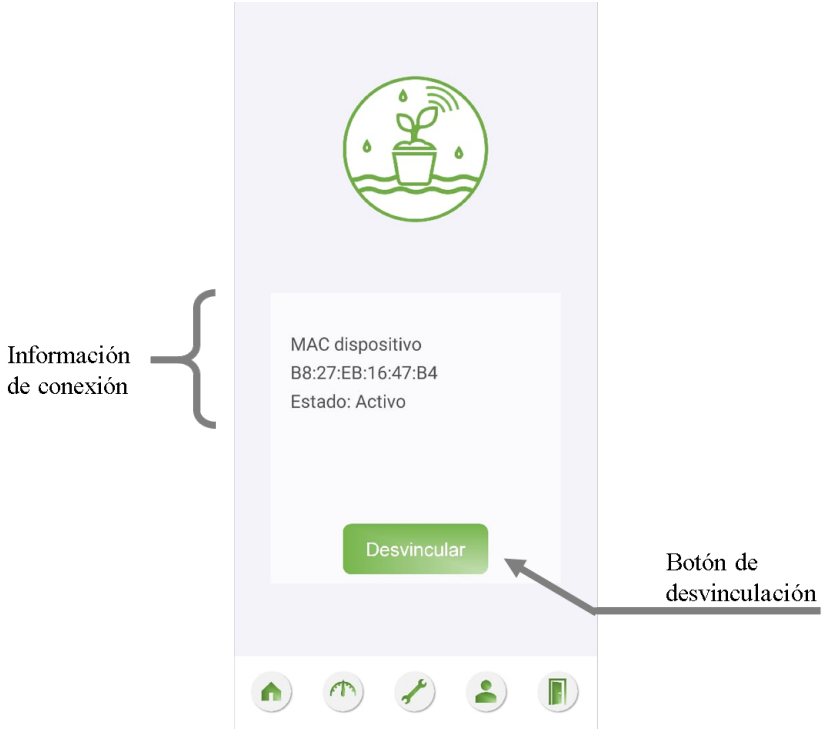


Figura 3.38: Pantalla de Sesión.

3.4.7. Menú de ayuda

El menú de ayuda es una ventana auxiliar cuya tarea es presentar una breve descripción de como funcionan los distintos elementos que componen la interfaz gráfica desde la cual se inició. El acceso a esta ventana se realiza mediante el botón de ayuda que posee el símbolo “?”, el cual se encuentra normalmente en la esquina superior derecha a excepción de la pantalla principal, donde se encuentra en cada función de operación manual de las bombas. La forma que posee esta ventana es la que se puede observar en la Fig.3.39

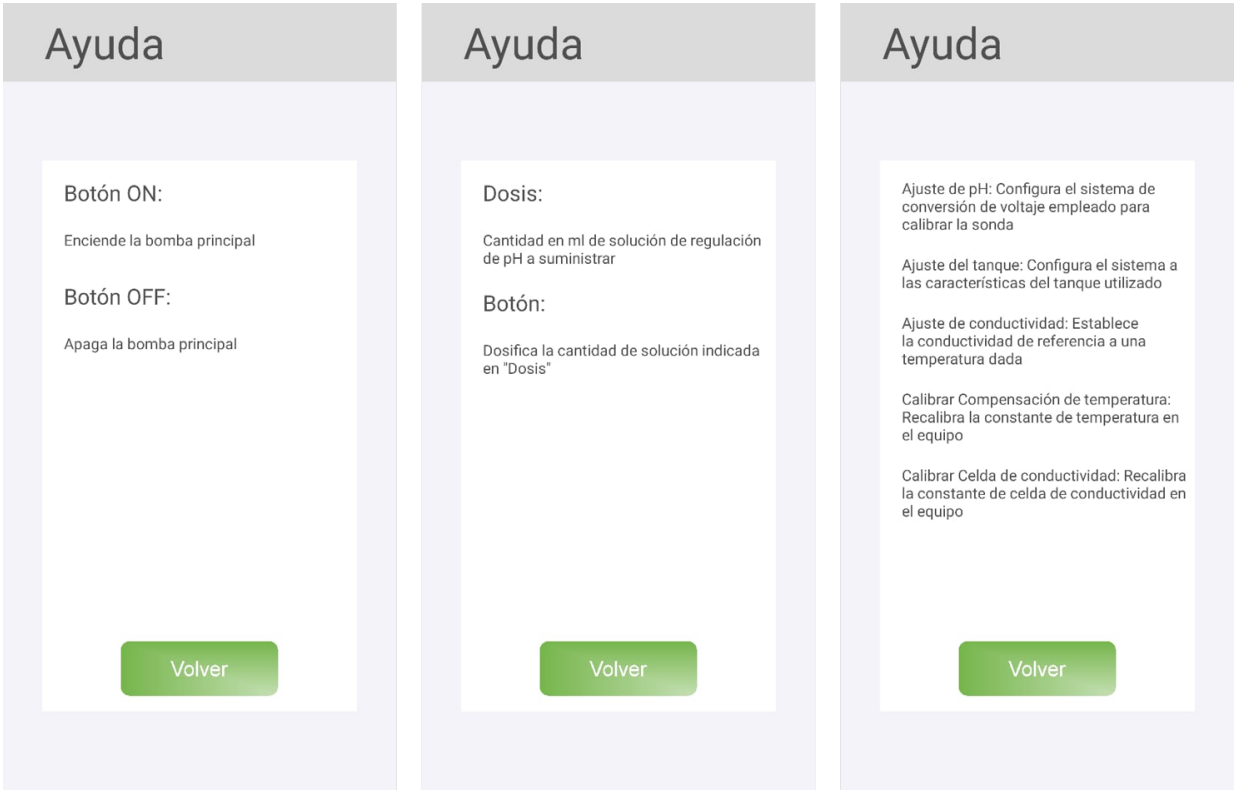


Figura 3.39: Menú de ayuda.

3.5. Evolución de la supervisión hidropónica de cultivo de lechugas

3.5.1. Ciclo de riego

El ciclo de riego empleado para el cultivo de lechuga hidropónica iniciaba a las 7h00 de la mañana y finalizaba a las 22h00 de la noche. La configuración para el riego se realizo con intervalo de riego por defecto de 60 min con un tiempo de riego de 10 min. Además se ajusto el riego por temperatura con intervalos de riego de 30 min cuando la temperatura supere los 23 °C manteniéndose hasta que la temperatura disminuya por debajo de los 21 °C. El resultado de esta configuración se puede observar en la Fig.3.40, el cual representa los cambios en el intervalos del ciclo de riego a lo largo del día y las temperaturas ambientales medidas.

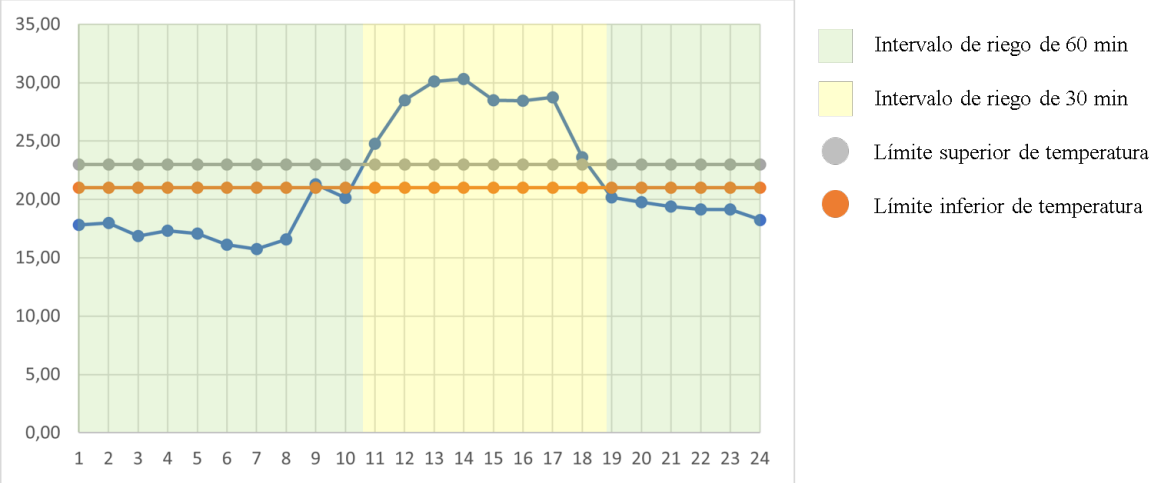


Figura 3.40: Intervalo de riego y temperatura ambiente del cultivo hidropónico del día 15 de enero del 2021.

Se observa un ciclo de riego con intervalos de 60 min durante las primeras horas de la mañana hasta alrededor de las 10h00 u 11h00, donde el intervalo de riego cambia a 30 min. Este intervalo se mantiene durante toda la tarde, momento donde el sol es más intenso. El intervalo de riego de 60 min es restablecido durante la noche, alrededor de las 19h00 y permanece de esta manera hasta finalizar el ciclo de riego.

3.5.2. Caudal de riego

El caudal de riego se ajusto a 1L por minuto por medio de una llave de paso.

3.5.3. Calibración de sonda de pH

La calibración de la sonda de pH se realizó empleando 3 soluciones de pH conocido:

- Solución de 4 pH
- Solución de 6.86 pH
- Solución de 9.18 pH

Se coloco la sonda en cada una de las soluciones y se obtuvo la lectura de voltaje correspondiente a cada una de ellas. A partir de los datos obtenidos, se construyó una tabla la cual se utilizó para realizar una regresión lineal con el objetivo de obtener la ecuación de pH en función de voltaje.

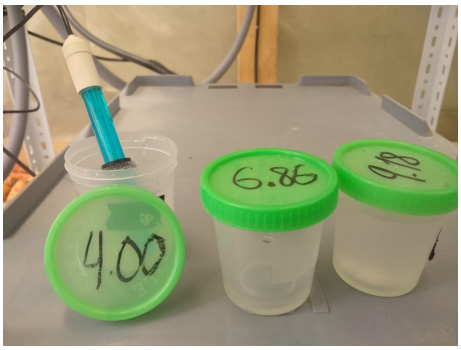


Figura 3.41: Soluciones de calibración para la sonda de pH

Los datos obtenidos se pueden observar en la Fig.3.42. La ecuación de correlación se puede observar en el gráfico y es introducida al sistema a través de la aplicación móvil.

Voltaje (V)	pH
3,22	4
2,76	6,86
2,41	9,18

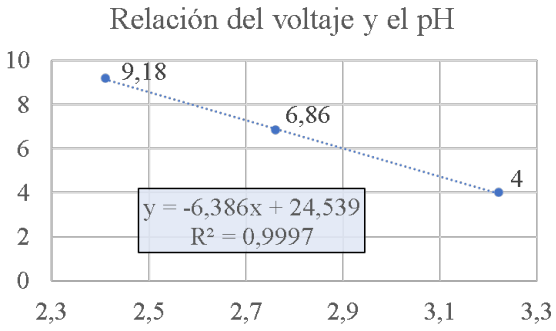


Figura 3.42: Regresión lineal de los valores de voltaje y pH de referencia.

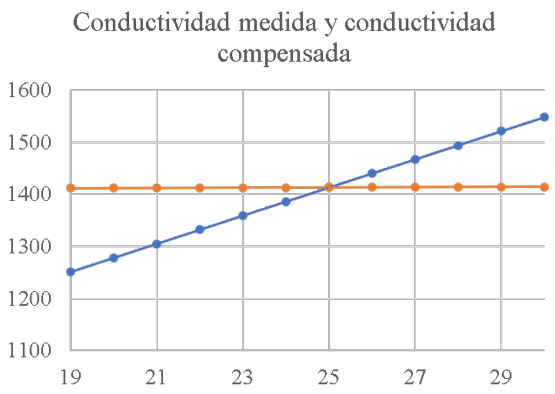
3.5.4. Calibración de sonda de conductividad

La sonda de conductividad es calibrada empleando una solución de referencia de 1413 uS a 25 °C. Para este proceso se introdujo la sonda de conductividad y el sensor de temperatura en el contenedor de solución, se esperó a que la lectura de conductividad se estabilizara y posteriormente se envío la orden de calibración de celda a través de la aplicación android. El procedimiento de calibración de celda de conductividad borra el coeficiente de compensación de temperatura, por lo cual este valor debe ser calibrado posteriormente.

Figura 3.43: Calibración de constante de celda de conductividad.

3.5.5. Conductividad medida y conductividad compensada

La Fig.3.44 muestra un gráfico de la aplicación del algoritmo de compensación de temperatura a la medición de conductividad de la solución de referencia. Como se observa, para un rango de temperatura desde 18 °C a 30 °C la conductividad medida tiene un carácter ascendente. Sin embargo, al aplicar la función de compensación con un coeficiente de 1.90, la conductividad se aproximado al valor esperado cercano a 1413 uS, valor que tendría la conductividad a 25 °C.



Conductividad de referencia (uS)		1413
Temperatura de referencia (°C)		25
Coeficiente de variación de temperatura		1,90
Conductividad	**Temperatura**	**Conductividad compensada**
1251	19	1412
1278	20	1412
1305	21	1412
1332	22	1413
1359	23	1413
1386	24	1413
1413	25	1413
1440	26	1413
1467	27	1413
1494	28	1413
1521	29	1414
1548	30	1414

Figura 3.44: Comparación entre la conductividad medida y conductividad compensada para la solución de referencia 1413 uS, 25° C

3.5.6. Calibración del dosificador de solución reguladora de pH

Para asegurar la correcta dosificación de solución reguladora de pH es necesario calibrar la ecuación de equivalencia entre el volumen a dosificar y los pulsos del sensor de flujo. Para ello se realizó una prueba experimental la cual consistía en activar la bomba de pH para una lista de pulsos de referencia con el fin de medir el volumen dosificado. La tabla 3.1 muestra los datos recolectados de la prueba con la cual se trazo la curva que se observa en la Fig.3.45.

Cuadro 3.1: Pulsos de referencia y volumen medido

Pulsos de referencia	**Primera lectura (ml)**	**Segunda lectura (ml)**	**Tercera lectura (ml)**	**Promedio (ml)**
5	1	1	1	1
10	3	3	3	3
20	5	5	5	5
30	8	8	8	8
40	10	10	10	10
50	13	13	13	13
60	16	16	16	16
70	18	18	18	18
80	21	21	21	21
90	23	23	23	23
100	26	26	26	26

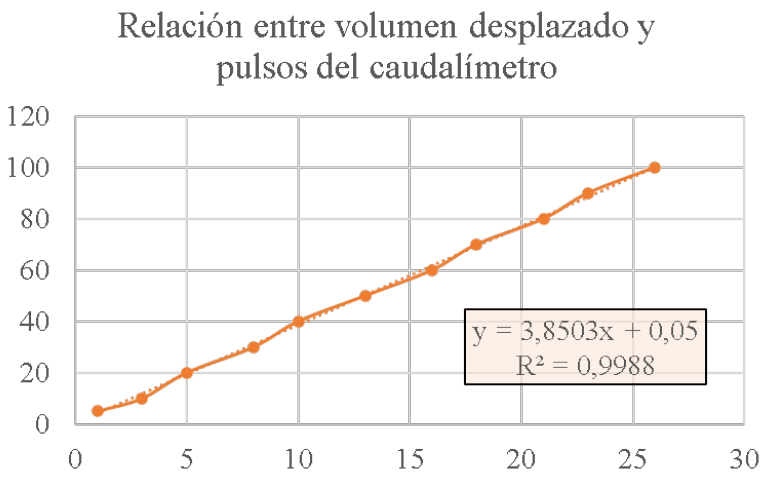


Figura 3.45: Gráfica de distribución de los datos de la prueba experimental.

La ecuación de relación entre volumen y pulsos del sensor es la ecuación 3.1.

$$Pulsos = 3,8503 \times (Volumen_{ml}) - 0,05 \tag{3.1}$$

3.5.7. Control de solución nutritiva y regulación de pH

La regulación de nutrientes y el pH en la solución nutritiva se realizo con intervalos de 2 días. Para la regulación del pH, se tomó como referencia el valor de 5.5 y se empleó una solución diluida de ácido nítrico. Debido a que la solución nutritiva tiende a subir su valor de pH al neutro característico del agua, solo es necesario disminuirla por medio de la solución de control.

Figura 3.46: Solución de regulación del pH.

El control de los nutrientes se realizo por medio de tres soluciones denominadas: solución A, solución B y solución C. Estás se suministraron en proporciones de 5 partes de A, 2 partes de B y 2 partes de C, hasta alcanzar la conductividad de referencia de 1200 uS. La cantidad en ml generalmente empleada es la que se observa en la tabla 3.2.

Cuadro 3.2: Proporciones y cantidades de solución suministradas para el control de nutrientes

Proporción	**Solución**	**Cantidad suministrada**
5 partes	Solución A	40 ml
2 partes	Solución B	16 ml
2 partes	Solución C	16 ml

Figura 3.47: Soluciones nutritivas empleadas en el cultivo hidropónico de lechuga.

3.5.8. Consumo de la solución nutritiva durante el ciclo de cultivo

El consumo de solución nutritiva puede ser visualizado a partir de los datos almacenados en el bloque histórico de la base de datos online. Se promedio el consumo diario de solución y se gráfico en función del tiempo por un período de un mes, iniciando a partir del 22 de noviembre del 2020 y finalizando con la cosecha el 22 de diciembre del mismo año. Como se puede observar en la Fig.3.48, durante los primeros 12 días no existió un consumo importante de solución, manteniéndose en un valor por encima del 90 % del tanque. Es durante estos días que el cultivo de lechuga se encontraba en etapa de plántula, aún desarrollando las primeras hojas. A partir del día 13 se observa un descenso en el nivel diario de solución. Es en esta etapa que la lechuga crece, desarrollando sus hojas y expandiéndose. El consumo de solución es más notorio y el nivel de solución desciende hasta llegar a la fecha de cosecha. A partir de este punto, el nivel permanece casi constante, puesto que no existían plantas que consumieran solución y se detuvo el sistema de riego.

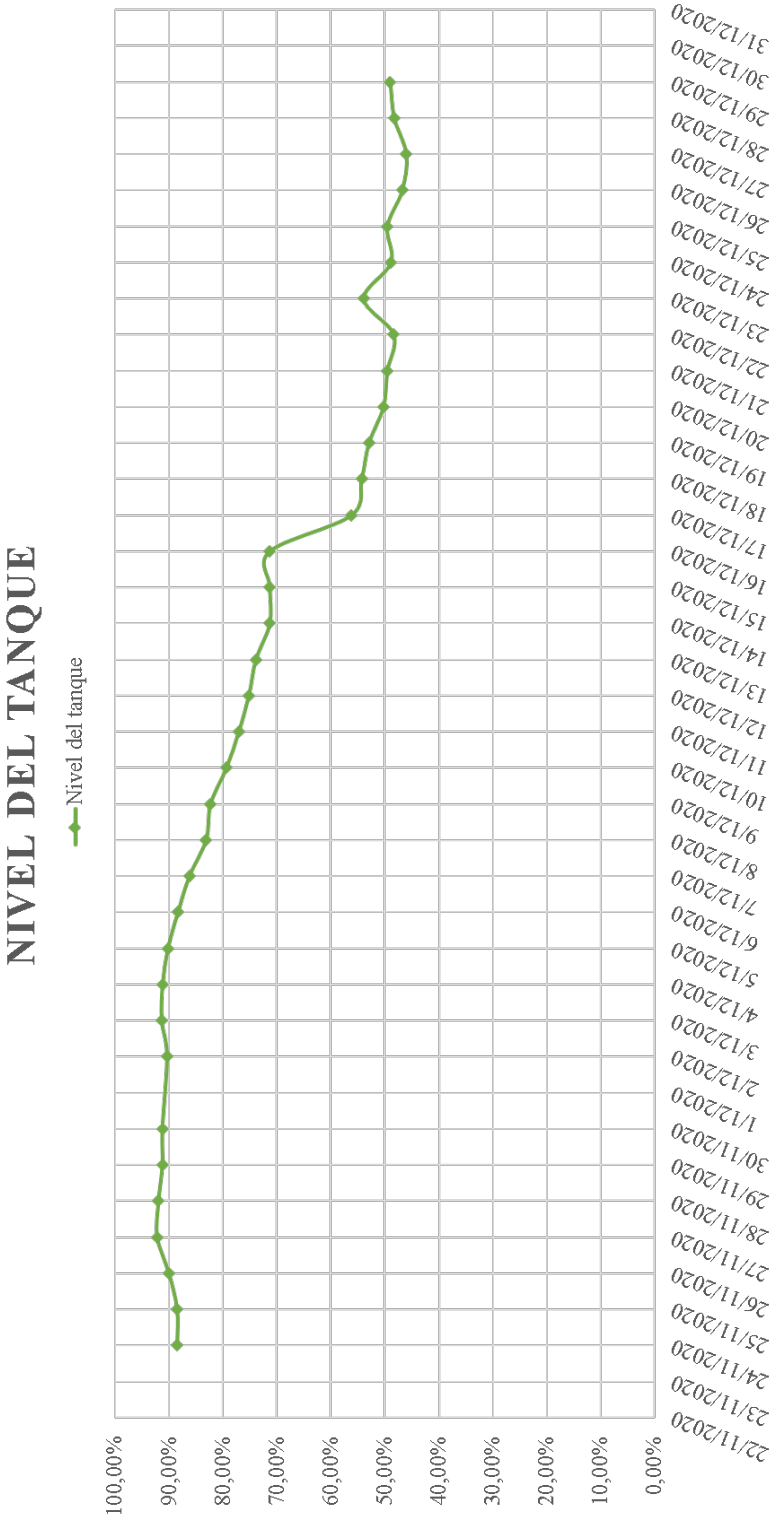


Figura 3.48: Consumo de solución nutritiva durante un ciclo de cultivo.

3.5.9. Precisión en la dosificación de solución reguladora de pH

Para corroborar la correcta dosificación de solución reguladora de pH se realizo un procedimiento el cual consistió en emplear el sistema de dosificación creado para administrar una cantidad determinada de solución y medir la cantidad suministrada por medio de una jeringa graduada. La tabla 3.3 muestra los resultados obtenidos de esta prueba para 10 valores.

Cuadro 3.3: Tabla de dosificación de solución reguladora de pH

Cantidad a dosificar	**Cantidad real**	**Observación**
1 ml	1 ml	Sin burbujas en el tubo ni retroceso en la solución
2 ml	2 ml	Sin burbujas en el tubo ni retroceso en la solución
5 ml	5 ml	Sin burbujas en el tubo ni retroceso en la solución
10 ml	10 ml	Sin burbujas en el tubo ni retroceso en la solución
15 ml	15 ml	Sin burbujas en el tubo ni retroceso en la solución
20 ml	20 ml	Sin burbujas en el tubo ni retroceso en la solución
30 ml	29 ml	Retroceso de solución y burbujas en el tubo
47 ml	47 ml	Sin burbujas en el tubo ni retroceso en la solución
50 ml	50 ml	Sin burbujas en el tubo ni retroceso en la solución
53 ml	52 ml	Retroceso de solución y burbujas en el tubo

Como se puede ver en la tabla 3.3, en la mayor parte de los casos el sistema trabajo de forma correcta y sin problemas en la dosificación. Sin embargo, existe un ligero error en la administración de solución cuando en el sistema físico de dosificación hay burbujas. Este problema también se presenta cuando por acción de la gravedad, la solución tiende a regresar al depósito, el cual se encuentra más bajo que el resto del sistema. Por tal motivo, para garantizar la correcta dosificación de solución, se realizó una activación previa de la bomba de pH con el fin de expulsar cualquier burbuja existente del sistema. La Fig.3.49 muestra el resultado de una dosificación exitosa realizada.

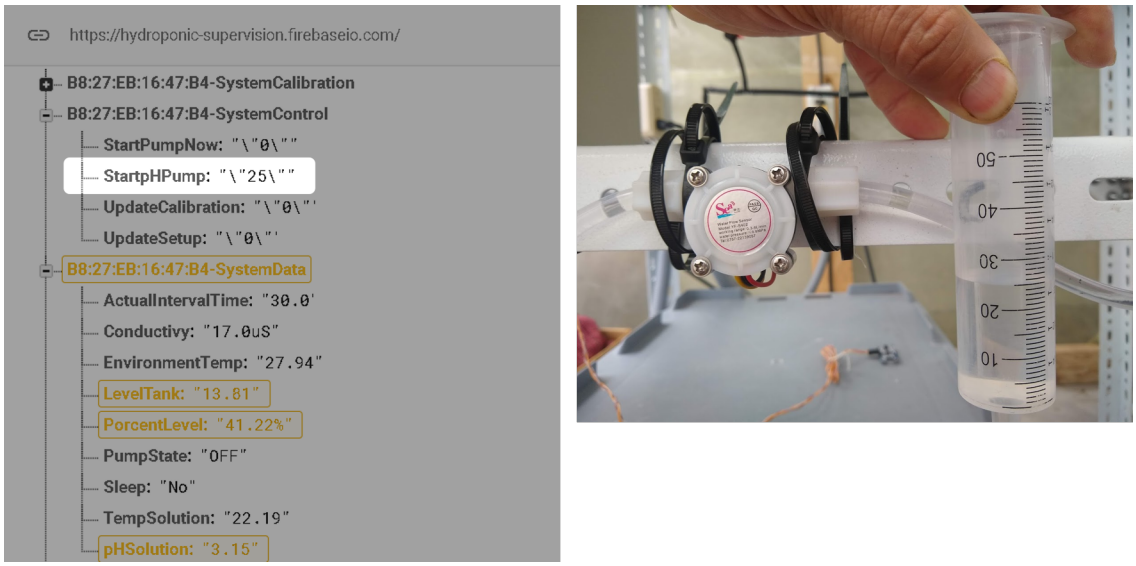


Figura 3.49: Dosificación exitosa de solución reguladora de pH desde la base de datos.

3.6. Resultados del prototipo

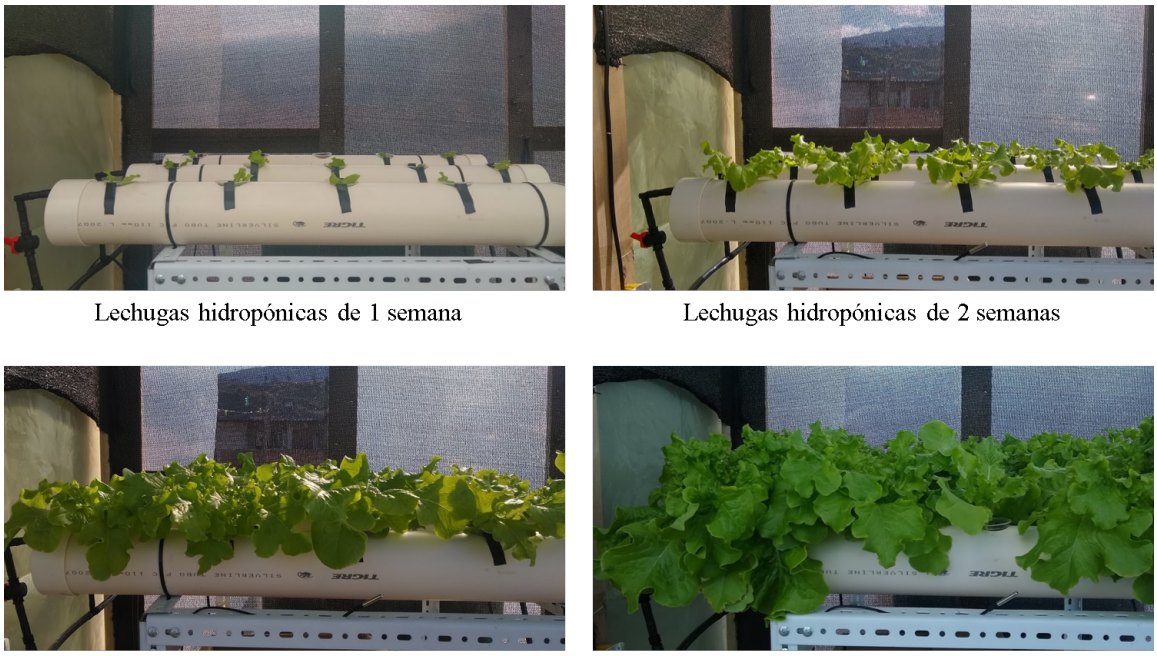
Para calificar que el prototipo cumplió con los objetivos establecidos, se hace uso de la tabla 3.4, en la cual se indica de forma cualitativa el funcionamiento del sistema y se señala las observaciones realizadas.

Cuadro 3.4: Tabla de cumplimiento del sistema de supervisión hidropónico

Descripción	**Función o acción**	**Cumple**	**Observación**
Capacidad de tomar medidas y acciones sobre el cultivo hidropónico	Por medio de las funciones de medición y control del sistema	Sí	No existió problema
Requiere una intervención mínima del operador	Funciones de riego y control automático	Sí	El operador debe intervenir para el ajuste de conductividad y de pH
El sistema es adaptable al tipo de cultivo	Por medio de funciones de ajuste de riego, sistema y de configuración y calibración de sondas	Sí	El operador debe conocer el tipo de cultivo que desea realizar y ser capaz de adaptarlo al sistema
El sistema es independiente a la conexión de red	Existen funciones encargadas de detectar la existencia de una conexión a red y mantener la ejecución del programa	Sí	El sistema no es capaz de guardar información de forma local, por la información correspondiente al período sin conexión se perdería
Almacenamiento e intercambio de información a través de la nube	Por medio de una base de datos online	Sí	Un uso considerable en la descarga de información mensual
Control del sistema de riego de forma manual	Por medio de funciones de control manual	Sí	No existió problema
Funciones de seguridad que mantengan la integridad del sistema	Por medio de funciones de alarma para el operador, función para la reconexión a la red y funciones para detener el ciclo de riego en caso de anomalías en el sistema	Sí	No existió problema
Indicación de estados del sistema de forma local	Por medio de luces indicadoras visuales	Sí	No existió problema
Acceso al sistema de forma remota por medio de una aplicación móvil	A través de la aplicación de supervisión para dispositivos android	Sí	Demoras ocasionales en la respuestas de actuadores

3.7. Resultados sobre el cultivo hidropónico

El sistema de supervisión de cultivos hidropónicos fue instalado en un invernadero con el fin de mantener la mayor cantidad de parámetros controlados. Con respecto al crecimiento de las plántulas de lechuga hidropónica, estas experimentaron un desarrollo acelerado en el transcurso de 4 semanas como se puede observar en la Fig.3.50. De acuerdo a la información recolectada por la base de datos, el ambiente donde se desarrollaron mostraba temperaturas similares a las vistas en la Fig.3.40. En total, se realizaron 3 cosechas con períodos aproximados de un mes y resultados similares entre ellas.



Lechugas hidropónicas de 1 semana

Lechugas hidropónicas de 2 semanas

Lechugas hidropónicas de 3 semanas

Lechugas hidropónicas de 4 semanas

Figura 3.50: Desarrollo de las lechugas hidropónicas.

Se puede apreciar el paso de plántulas durante las primeras dos semana a plantas prácticamente desarrolladas en la cuarta semana.

3.8. Limitaciones

3.8.1. Sistema local sin conexión

Cuando ocurre un problema en la conexión a internet, el sistema local es capaz de seguir su funcionamiento normal hasta que se restablezca la red. Sin embargo, el sistema no posee un almacenamiento de información local, por lo que los datos que se generen durante el período sin conexión se pierden.

3.8.2. Límite de descargas mensuales

A pesar que el proyecto admite el registro de una cantidad indefinida de sistemas locales y dispositivos de supervisión, se ve limitada por las descargas de datos mensuales. Como se observa en la Fig.3.19, al finalizar el mes se tiene un uso de descargas aproximado del 80 % del total disponible para un único sistema de supervisión. Esto implica que para mantener más de un sistema de supervisión, es necesaria una optimización en las funciones de acceso a los datos del sistema local o la adquisición de un plan para el servicio de nube de una mayor capacidad.

3.8.3. Simultaneidad de medición del pH y de conductividad

Al realizar la medición de conductividad de la solución, la sonda aplica un voltaje con el fin de medir la resistividad de la solución. Si se realiza la medición de pH, una parte de la corriente de la sonda de conductividad se desvía hacia la sonda de pH usando la solución como medio conductor. Esta corriente afecta el valor de la medición de pH, por lo que la lectura ya no es válida. Esto imposibilita la medición simultanea de pH y conductividad con el diseño actual del proyecto.

3.8.4. Acceso de múltiples dispositivos de supervisión a un único sistema local

Existe un sistema de seguridad que evita el acceso simultáneo de más de un dispositivo de supervisión al sistema local con el fin de evitar conflictos en la transferencia de información. Para permitir que otro dispositivo acceda al sistema local, se debe desconectar cualquier conexión previa. Sin embargo, esto impide que múltiples usuarios puedan visualizar la información de los sensores y realizar el control de actuadores.

Capítulo 4

Conclusiones y trabajo futuro

4.1. Conclusiones

El trabajo realizado presentó la propuesta de un sistema de supervisión aplicado a los cultivos hidropónicos contando con la capacidad de realizar mediciones y tomar acciones sobre una estructura establecida. El modelo empleado fue capaz de alcanzar los objetivos establecidos a través de sus cuatro bloques: Estructura física, sistema local, almacenamiento y comunicación y dispositivo de supervisión. Cada un desempeñando una función importante en todo el sistema de supervisión. Cabe destacar que la base de este modelo se centra en la nube como mecanismo de gestión, puesto que la información almacenada y los parámetros de control se encuentran en la base de datos online, y son accedidos y usados tanto por el sistema local como por el dispositivo de supervisión.

En el sistema local, emplear un módulo separado dedicado exclusivamente a la lectura de los sensores y control de actuadores resulto en una gran ventaja al trabajar con internet de las cosas. Esto se debe a que siempre existen inconvenientes cuando se usa una red a internet WIFI. Casos como caída de la red, cambios en la configuración del internet, demoras producidas durante la transferencia de información de trabajo, etc., suceden cuando se trabaja a largo plazo y resultan en tiempos de respuestas de la computadora lentos e impredecibles. Con la tarjeta principal, se garantiza que la lectura de sensores se haga de forma correcta, y el control de los actuadores funcione de forma precisa y sin demoras, delegando el trabajo de procesamiento de información e IOT a la computadora. Por otro lado, la tarjeta principal es incapaz de realizar funciones por sí misma, ya que su comportamiento depende de los comandos enviados por la Raspberry, por lo que solo es un periférico a través del cual se obtiene información del sistema. Se puede decir entonces que el sistema local esta centralizado en Raspberry.

Con respecto a la nube, Firebase resulto ser un servicio bastante completo para el proyecto realizado. Para este trabajo, se emplearon la función de autorización para la lectura y escritura de datos y función sin conexión para evitar respuestas inesperadas en el sistema local y el dispositivo de supervisión. Sin embargo, queda trabajo de optimización en la descarga de da-

tos, puesto que mensualmente un solo dispositivo es capaz de alcanzar un aproximado del 80 % de descargas permitidas mensuales, dejando un margen insuficiente para la implementación de nuevos dispositivos en la misma nube.

Como se explicó en capítulos anteriores, el dispositivo de supervisión consistió en una aplicación creada para la plataforma Android. El mayor inconveniente durante su desarrollo fueron las grandes limitaciones que presentaba AppInventor con respecto al diseño de las interfaces, efectos y funcionalidades disponibles. Durante las pruebas, la aplicación móvil funciono de forma correcta como mecanismo de supervisión y control, y las funciones de configuración permitieron reajustar el sistema local a cualquier necesidad que se requería.

Para evaluar el correcto funcionamiento del prototipo, se empleó como referencia las premisas establecidas al inicio del capítulo de metodología y se analizó si existían funciones en el sistema que las cumplían. Los resultados obtenidos fueron satisfactorios y cumplieron con los objetivos planteados, realizándose las debidas observaciones con el propósito de encontrar posibles puntos a mejorar en trabajos futuros.

Por ultimo, aún queda trabajo por delante. Los esquemas, funciones y sistema de control presentados pueden ser mejorados en el sistema de supervisión. o aplicados de distintas formas en nuevos modelos. Como se presento en el apartado de trabajo futuro, el proyecto presenta limitaciones en cuanto a como y que puede hacer. Como punto a favor, cabe destacar el funcionamiento a largo plazo que el sistema ha tenido, con muy pocos inconvenientes. Sin embargo, además del sistema de supervisión, existe un factor clave en el desarrollo del cultivo hidropónico el cual es el aspecto humano. Es el usuario final quien en base a la información mostrada por el sistema tomara las decisiones. El sistema facilita el trabajo y realiza controles en base a lo que el operador haya configurado, por lo que la calidad de las cosechas depende fuertemente del conocimiento del cultivo por parte de la persona.

4.2. Trabajo futuro

4.2.1. Diseño para medición simultanea de pH y conductividad

El problema de la medición simultánea de conductividad y pH se debe a que la solución nutritiva es un medio conductor por la que la corriente que emite la sonda de conductividad puede circular y afectar a la sonda de pH. Esto se debe porque ambas sondas comparten el mismo retorno a tierra, por lo que se forma un circuito cerrado entre ambos sensores. La solución que se plantea para solventar este inconveniente consiste en aislar los circuitos del bloque de conductividad del sistema local a la hora de realizar la medición. Esto se puede realizar de dos maneras. Empleando una fuente de alimentación propia para la medición de conductividad o realizar la medición de la conductividad y pH de forma alternada y rápida, multiplexando la alimentación de voltaje de cada sonda y aparentando simultaneidad. En la Fig.4.1 y Fig.4.2 se observan dos propuestas basadas en fuentes de alimentación aisladas. La base de la propuesta de la Fig.4.1 hace uso de una fuente de alimentación con dos salidas de voltaje independientes y aisladas o dos fuentes de alimentación independientes para alimentar el sistema local y el bloque de medición de conductividad. La Fig.4.2 propone emplear una única fuente y un convertidor DC DC de aislamiento para la fuente alimentación. Para ambos casos, el envío de los datos leídos por el sensor se debe enviar al sistema local también de forma aislada. Para ello se pueden usar dispositivos de aislamiento óptico como son optoacopladores.

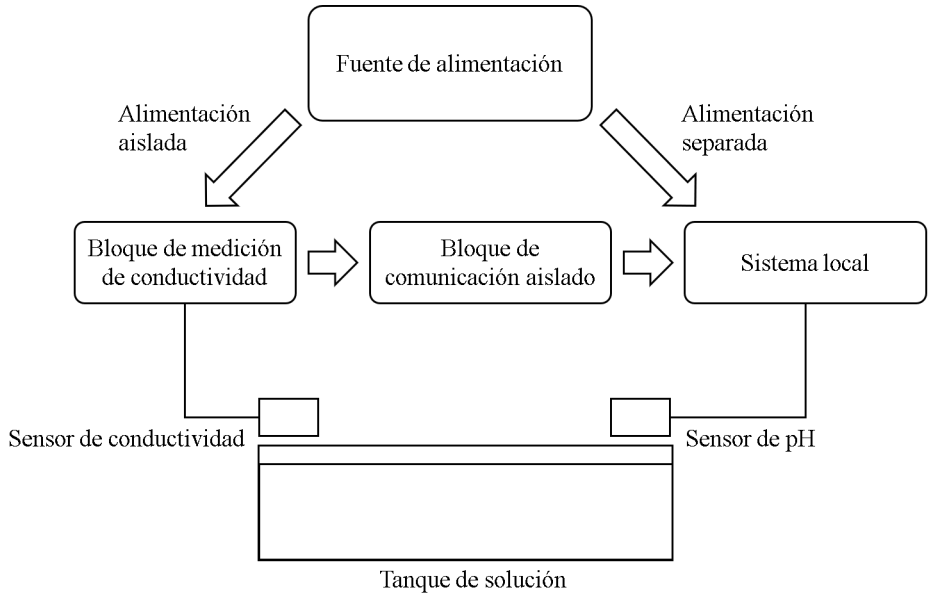


Figura 4.1: Propuesta para la medición de conductividad y pH por medio de fuentes de alimentación independientes.

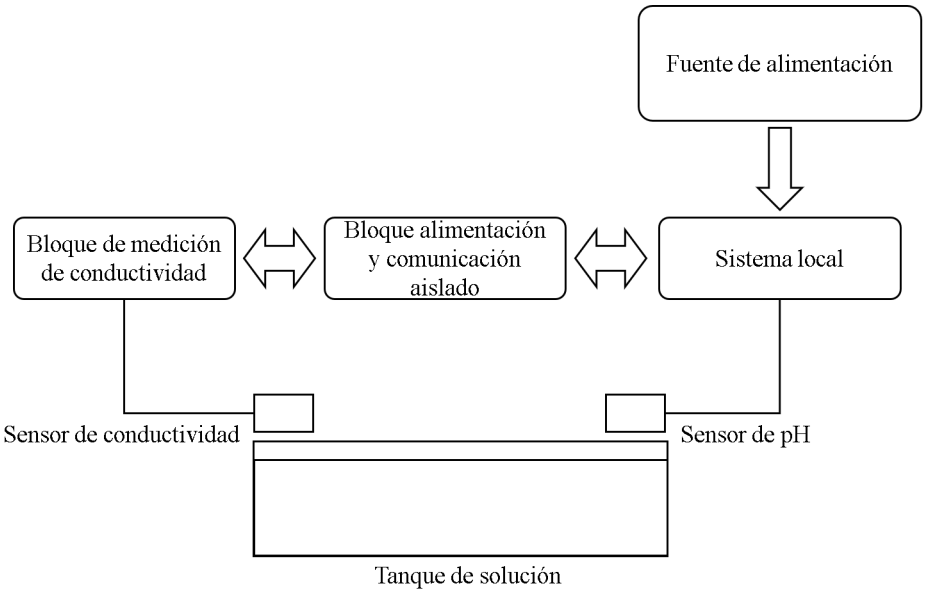


Figura 4.2: Propuesta para la medición de conductividad y pH por medio del aislamiento de la fuente de alimentación.

La Fig.4.3 es una propuesta que se basa en la activación momentánea de los circuitos de medición de la conductividad durante un tiempo T1 mientras los circuitos de medición de pH permanecen desconectados, y posteriormente alternar este estado durante un tiempo T2. Esta multiplexación se realizaría a través de las líneas positiva y negativa de alimentación de voltaje y no necesitaría aislamiento para el envío de datos desde los sensores.

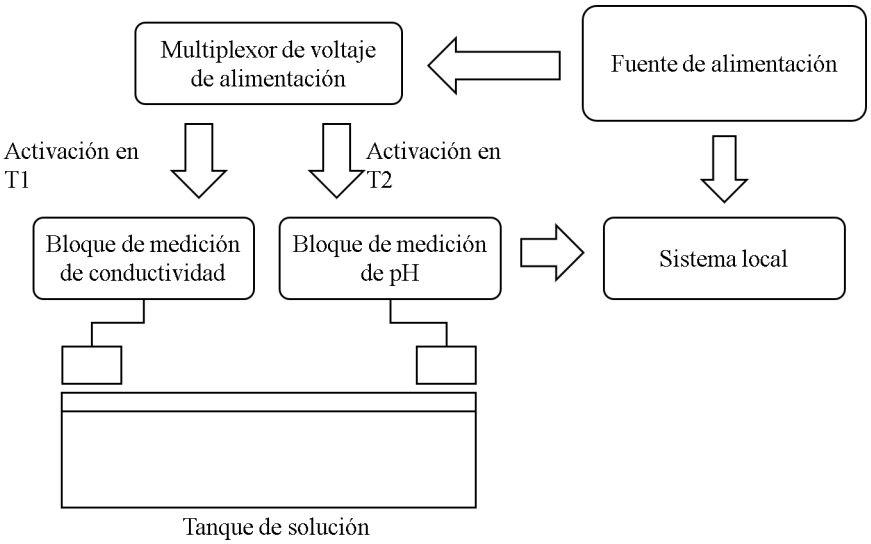


Figura 4.3: Propuesta para la medición de conductividad y pH por medio de multiplexación de la alimentación de voltaje a los bloques de medición.

4.2.2. Base de datos local para almacenamiento de datos sin conexión

Una base de datos local permitiría el almacenamiento de información en el sistema local de la misma forma que se realiza en la nube. La ventaja que presenta es que no existe una dependencia a la conexión a la red para la integridad de los datos, por lo que ya no existiría la pérdida de datos que actualmente ocurre durante un problema de internet. La implementación de este sistema de almacenamiento se puede realizar por medio de librería sqlite3 para python, archivos txt, etc.

4.2.3. Autenticación de dispositivos por correo o cuentas

Actualmente el sistema de supervisión cuenta con un sistema de seguridad para la conexión basado en tokens, los cuales se encuentran integrados en el sistema local y en la aplicación de supervisión. Sin embargo esto permite que cualquier dispositivo que conozca el token puede acceder sin ningún problema a los datos de la nube. Para tener un mejor control de que usuarios tienen acceso a la información almacenada, se puede hacer uso del sistema de autenticación por medio de correo y cuentas, el cual obliga al usuario a autenticar su conexión a través de un correo existente o una cuente a google.

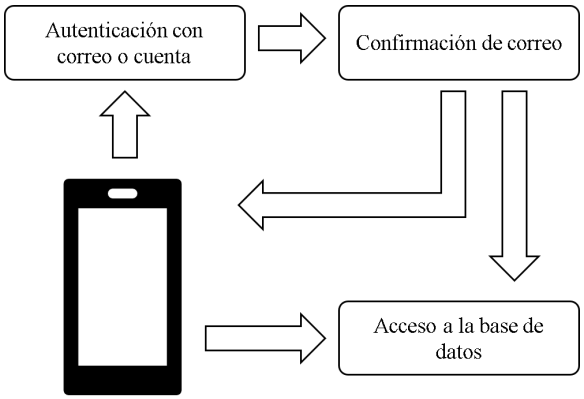


Figura 4.4: Metodología de autenticación por medio de correo o cuenta.

4.2.4. Calibración automática de sonda de pH

Al igual que existen funciones de calibración para la sonda de conductividad que facilitan el reajuste del sensor, se puede integrar una función que realice la calibración automática de la ecuación de la sonda de pH a partir de las soluciones de referencia. El diagrama de flujo de esta función se puede ver en la Fig.4.5

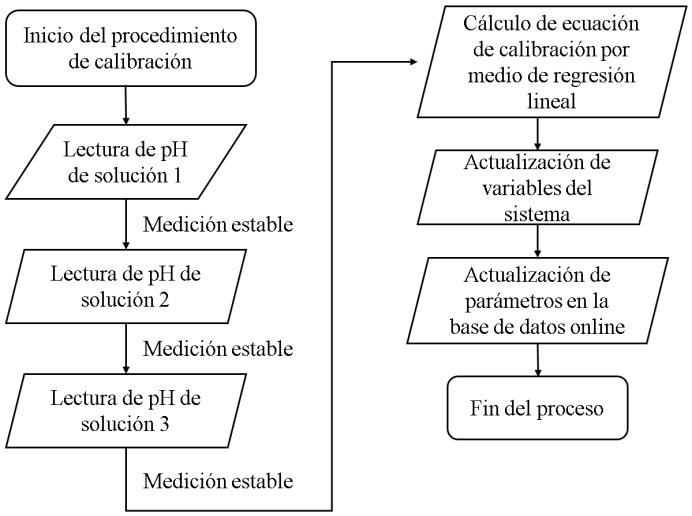


Figura 4.5: Diagrama de flujo para la función de calibración de la sonda de pH.

Esta función se encuentra parcialmente integrada en el programa principal de la Raspberry. Sin embargo no se encuentra en funcionamiento. Las Fig.4.6 y Fig.4.7 muestran los fragmentos del código de esta función.

```
244  # Calcula los coeficientes de la correlacióon lineal de un conjunto de valores x,y ***********
245  def LinealRegresion(X,Y):
246      if len(X) == len(Y):
247          n = len(X)                  # Numero de datos para trabajar
248          Ex  = sum(X)                # Sumatoria de todas las X
249          Ey  = sum(Y)                # Sumatoria de todas las Y
250          Exy = float(sum(numpy.array(X)*numpy.array(Y))) # Sumatoria de X*Y
251          Ex2 = float(sum(numpy.array(X)*numpy.array(X))) # Sumatoria de X^2
252          # Compute lineal regresion coeficients ******************
253          a = (n*Exy-Ex*Ey)/(n*Ex2-Ex*Ex)
254          b = (Ey-a*Ex)/n
255          return (a,b)
256      else:
257          print ('Data lenth is no correct!')
258          return (1,0)    # y = aX + b with a = 1 and b = 0
```

Figura 4.6: Función de regresión lineal para el cálculo de la ecuación de calibración de la sonda de pH.

```
603  # ******************************** Calibrate pH Sensor (No funtional) *************************
604  if phCalibrationEnable == True:
605      SetLed(1,0,1,LedPH,0x02)
606      if (abs(float(pHSolution) - pHLastState) < 0.1)and(abs(pHChange-float(pHSolution)) < 1):
607          # Set led calibration ok **************
608          SetLed(1,0,1,LedPH,0x00)
609          # Save actual voltage *****************
610          if EnableSetCalibration == 1:
611              if phCalSolution == 1:
612                  pHCalMin = float(pHSolution)
613              elif phCalSolution == 2:
614                  pHCalMax = float(pHSolution)
615              elif phCalSolution == 3:
616                  pHCalMed = float(pHSolution)
617                  pH_X = (pHCalMin,pHCalMax,pHCalMed)
618                  pH_Y = (4,9.18,6.86)
619                  a,b = LinealRegresion(pH_X,pH_Y)
620                  print (a)
621                  print (b)
622                  phCalibrationEnable = False
623              pHChange = float(pHSolution)
624              phCalSolution = phCalSolution + 1
625              EnableSetCalibration = 0
626  else:
627      pHLastState = 0
628      phCalSolution = 1
629      pHChange = 0
```

Figura 4.7: Fragmento de la función de calibración de sonda de pH.

4.2.5. Control local del sistema

Durante la etapa de diseño del proyecto se tomo en cuenta la integración del hardware necesario para la implementación de un control local del sistema de supervisión. Para ello, la placa de indicadores LEDs cuenta con entradas para pulsadores los cuales permitirían realizar acciones de control, además, empleando los indicadores luminosos para como mecanismo de retroalimentación para informar al operador que se está realizando. Sin embargo, con respecto al software, todavía no esta implementado.

Figura 4.8: Pulsadores para el control local del sistema.

4.2.6. Notificaciones para el dispositivo de supervisión

La aplicación de supervisión cumple el objetivo de visualizar la información del cultivo hidropónico y tomar acciones sobre los actuadores del sistema local. Sin embargo, realizar el control de nutrientes y pH, tener registro del tiempo que ha transcurrido desde que se inicio el cultivo hidropónico o incluso conocer que ha ocurrido algún fallo en el sistema depende del operador. Es decir, la eficiencia del sistema de supervisión se ve afectada dependiendo de que tan pendiente este el usuario a la aplicación de supervisión y a la evolución del cultivo. Es por eso que se propone un mecanismo de notificación que se ejecute en segundo plano en el Smartphone, de tal manera que al detectar un evento en el sistema, se pueda alertar al usuario por medio de un mensaje en la barra de notificaciones del celular. Las notificaciónes que podrían incluirse en un futuro se listan a continuación:

- **Notificaciones de sistema local desconectado. -** Indicación que avisa al usuario que el sistema local esta desconectado o apagado.
- **Notificaciones de seguimiento del cultivo. -** Alertas que avisan al usuario de realizar control de la solución en ph y nutrientes además de indicar el tiempo transcurrido desde el incio del cultivo.
- **Notificaciones de alarma. -** Advertencias que señalan problemas en sensores, niveles bajos de solución, etc.

Capítulo 5

Anexos

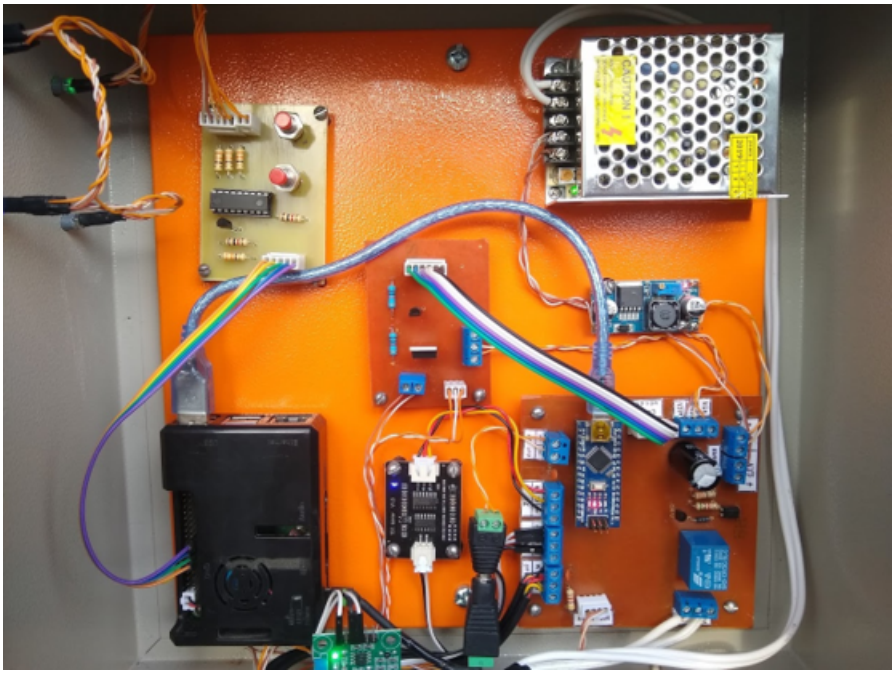


Figura 5.1: Hardware del sistema local.

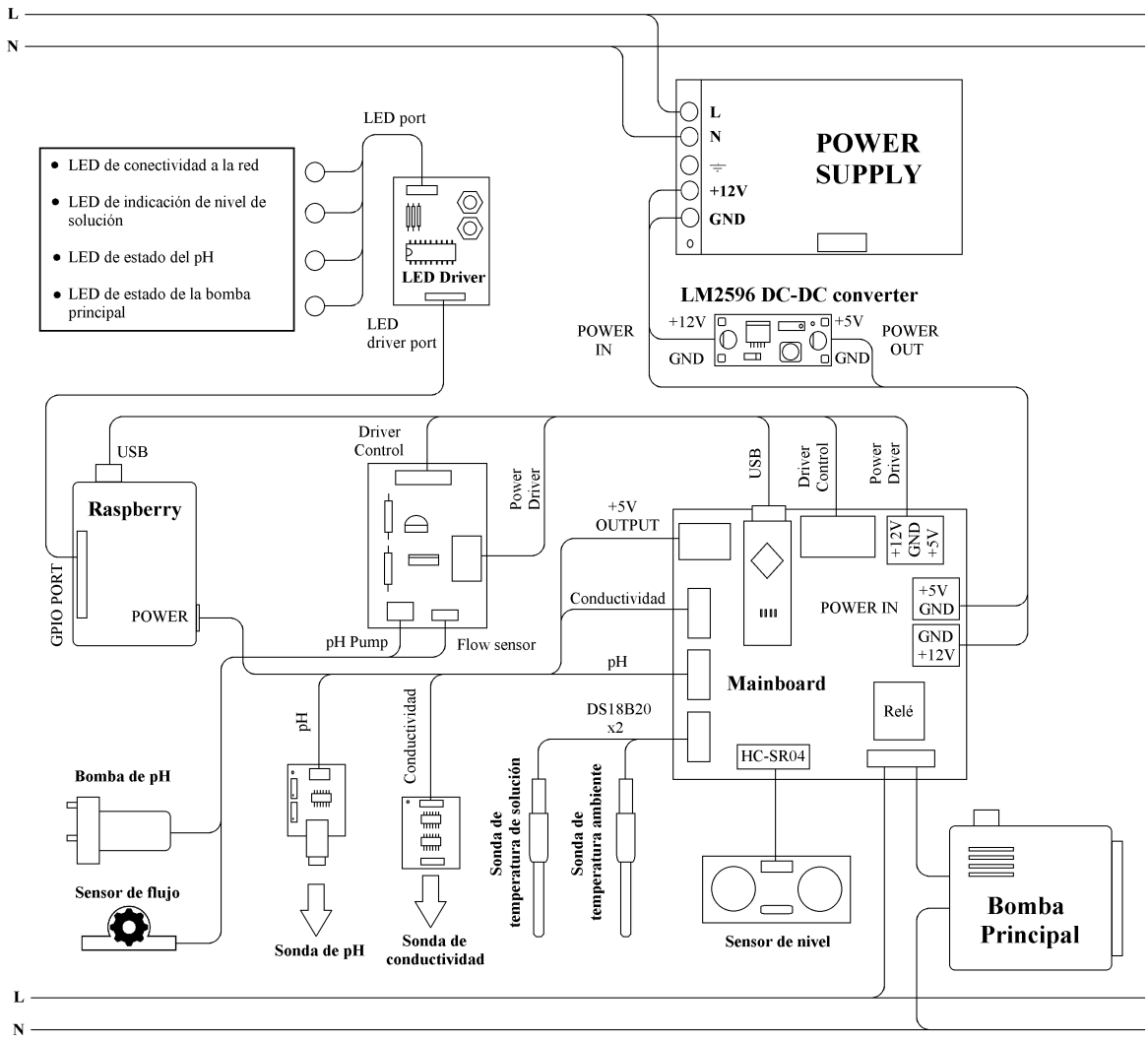


Figura 5.2: Diagrama del hardware del sistema local.

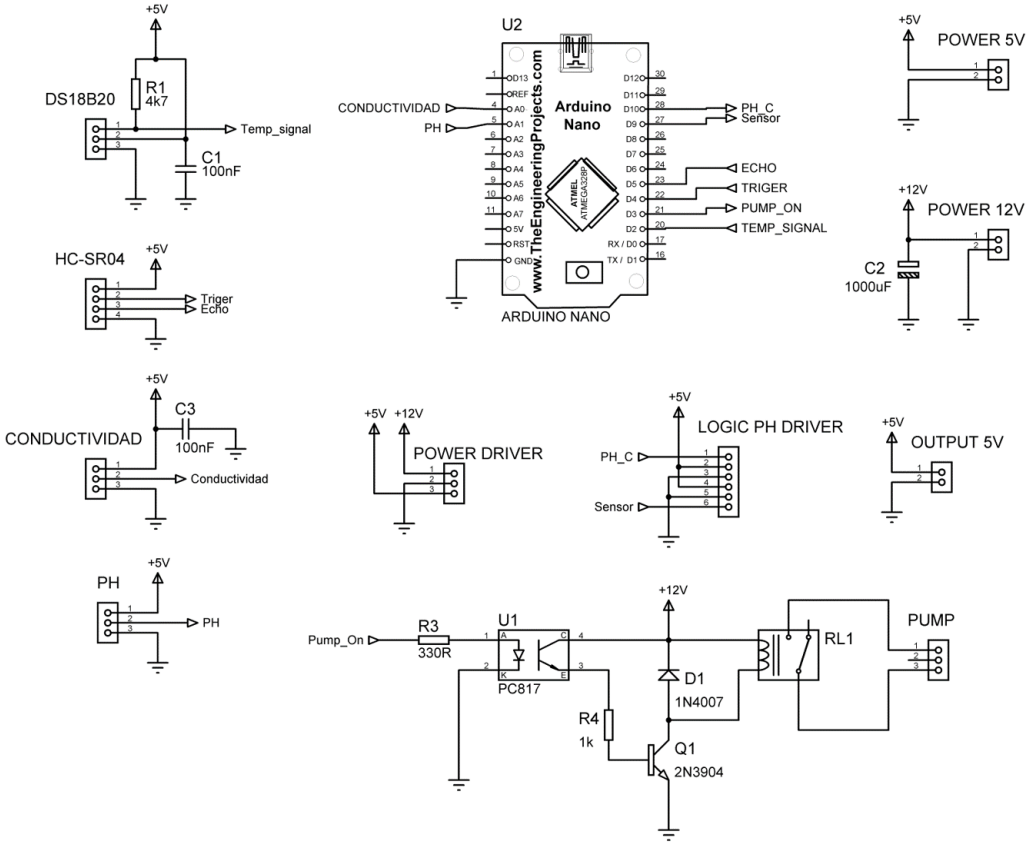


Figura 5.3: Diagrama de circuito de la tarjeta principal.

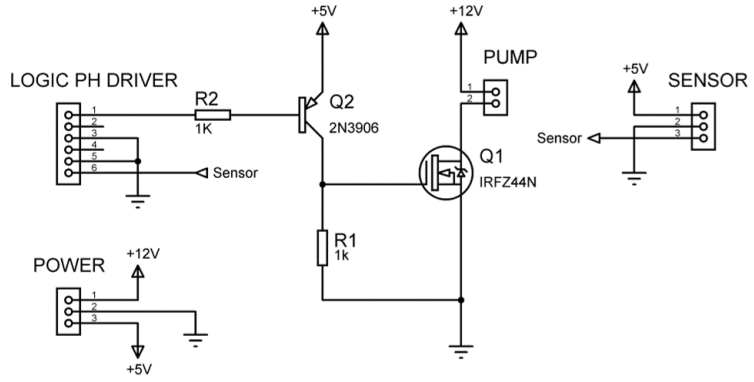


Figura 5.4: Diagrama de circuito de la placa de control de la bomba de pH.

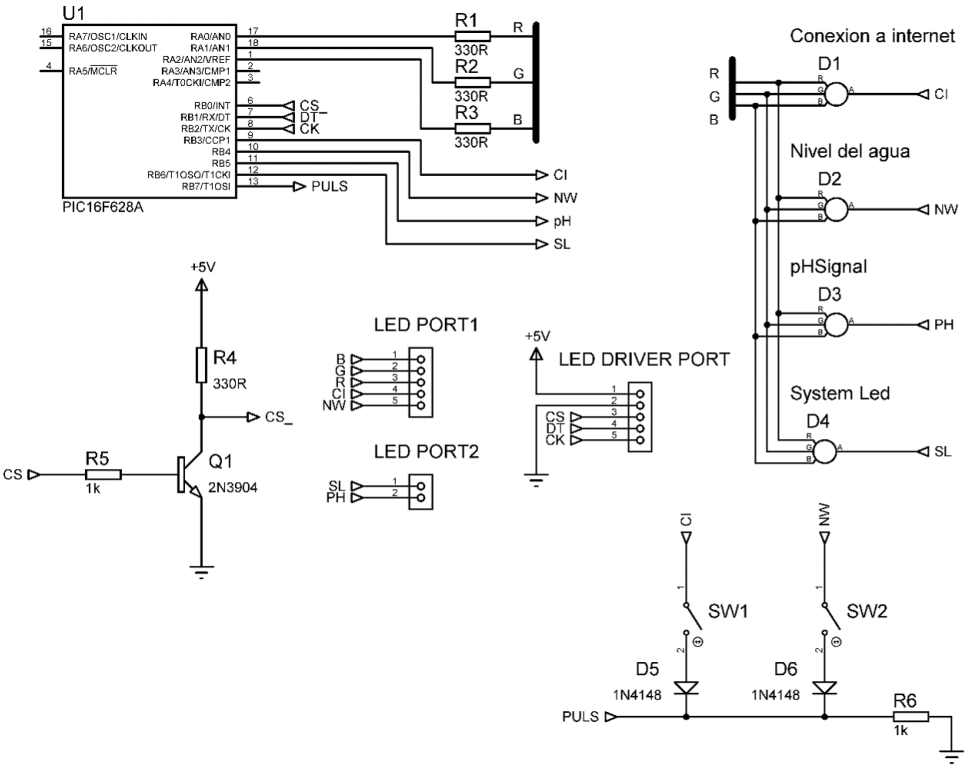


Figura 5.5: Diagrama de circuito de la placa de los indicadores LEDs.

Figura 5.6: Lechugas hidropónicas en la estructura física.

Figura 5.7: Lechuga hidropónica.

Capítulo 6

Listado de códigos

6.1. Código de la raspberry

El código presentado fue desarrollado en python3 empleando la consola de la Raspberry.

```
#!/usr/bin/env python3
import serial
import time
import numpy
import threading
import datetime
import RPi.GPIO as GPIO
from firebase import firebase

# Variables de uso general ********************************
MAC = ""                                    # Almacena la dirección MAC WIFI
de la raspberry (Se emplea como identificador unico del sistema para la comunicac
ión con la nube y el dispositivo de supervisión remoto)
IntervalTime = 0.0                          # Indica el tiempo cada cuanto se
 ejecutara el riego
IntervalTimeBuffer = ""                     # Buffer para calcular el valor e
n segundos del tiempo de riego
WateringTime = 0.0                          # Indica el tiempo que dura el ri
ego
PowerOFFPump = 0                            # Sirve como bandera para indicar
 cuando la bomba esta apagada. 0 = apagada, 1 = encendida. Estado inical apagado
EnableMainProgram = 0                       # Sirve como bandera para dar pas
e a la ejecución del programa principal, despues de haber tomado los datos de set
up de la nube
EnablePump = 2                              # Sirve como bandera para el inte
rvalo de tiempo de sleep de la bomba. Se inicializa en 2 para establecer las cond
iciones iniciales
StartPumpNow = ""                           # Variable para realizar el encen
dido/apagado manual de la bomba
StartTime = ""                              # Señala la hora de inicio del si
stema de riego automatico
SleepTime = ""                              # Señala la hora final del sistem
a de riego automatico
UpdateSetup = ""                            # Señala que se debe actualizar l
a información de setup
UpdateCalibration = ""                      # Señala que se debe actualizar l
a información de calibración
EnableSendDataHistory = 0                   # Bandera para esperar que todos
los datos sean adquiridos antes de enviarlos al historial de la nube
HourBuffer = 24                             # Value random to start system
TempSolution = ""                           # Variable para almacenar el valo
r de la temperatura de la solución
TempEnvironment = ""                        # Variable para almacenar el valo
r de la temperatura del medio ambiente
```

```
pHSolution = "0.0"                              # Variable para almacenar el valo
r del pH de la Solución
pHSolutionBuffer = 2.5                          # Variable que se usa para realiz
ar los calculos de conversión de pH. Valor inicial puede ser aleatorio
ConductivyBuffer = "Nah"                        # Variable que se usa para almace
nar valores temporales de la conductividad
PorcentLevel = ""                               # Varaible para almacenar el nive
l del tanque en porcentaje
LevelTank = ""                                  # Variable para almacenar el nive
l del tanque en cm
TankHeight = ""                                 # Variable para almacenar la altu
ra del tanque
ConductivySolution = ""                         # Variable para almacenar el valo
r de la conductividad enviada
UpdateCellConstant = ""                         # Flag to point when system has t
o update Conductivy Sensor Cell Constant
UpdateTemperatureCompensation = ""              # Flag to point when system has t
o update Solution Temperature Coefficient Variation
ConductivySystemNow = 0.0                       # Variable que almacena el valor
de la conductividad sin compensación de temperatura
OffsetTank = 0.0                                # Variable para almacenar el valo
r de offset para los calculos de nivel del tanque
SystemTemperatureCoefficientVariation = 0.0     # Variable para almacenar el coef
iciente de variacion de temperatura de la solución
SystemCellConstant = 0.0                        # Almacena la constante de celda
de la sonda de conductividad
ReferenceConductivy = 0.0                       # Almacena el valor de la conduct
ividad de referencia
ReferenceTemperature = 0.0                      # Almacena el valor de la tempera
tura de referencia para la compensación de la conductividad
RealConductivy = 0.0                            # Almacena la conductividad medid
a por el sensor sin compensar
mpH = 0.0                                       # Almacena la ganancia o pendient
e de la ecuación de linealización de la sonda de pH
bpH = 0.0                                       # Almacena el offset o valor inde
pendiente de la ecuación de linealización de la sonda de pH
pHMin = 0.0                                     # Almacena el valor de pH mínimo
bajo el cual el pH de solución es considerado crítico
pHMax = 0.0                                     # Almacena el valor de pH bajo el
 cual el pH de solución es considerado normal
pHCalMin = 0.0                                  # Varible de calibración de pH pa
ra el valor minimo (No funcional)
pHCalMed = 0.0                                  # Varible de calibración de pH pa
ra el valor medio  (No funcional)
```

```
pHCalMax = 0.0                                  # Varible de calibración de pH pa
ra el valor maximo (No funcional)
pHLastState = 0.0                               # Last state of pH Voltage (No fu
ncional)
pHChange = 0.0                                  # Used to know when there was a c
hange of solution (No funcional)
LevelLimit = 0.0                                # Almacena el valor de nivel lími
te bajo el cual el nivel de solución es considerado crítico
LevelMin = 0.0                                  # Almacena el valor de nivel bajo
 el cual el nivel de solución es considerado bajo
LevelMax = 0.0                                  # Almacena el valor de nivel bajo
 el cual el nivel de solución es considerado medio. Valores superiores son consid
erados normales
FlagChangeIntervalTime = 2                      # Bandera para gestionar el contr
ol de riego por temperatura. Cuando vale 2, activa el control inmediatamente. Cua
ndo vale 0, habilita el cambio al intervalo de alta temperatura. Cuando vale 1, h
abilita el cambio al intervalo de temperatura normal
WTCTempMax = 0.0                                # Valor de temperatura de superio
r que activa el cambio al intervalo de riego de temperatura alta
WTCTempMin = 0.0                                # Valor de temperatura bajo el cu
al se reestablace el intervalo normal de riego
HighTempIntervalTime = 0.0                      # Almacena el valor del intervalo
 de riego para alta temperatura
phCalibrationEnable = False                     # Habilita la calibración de sond
a de pH local (NO funcional)
EnableSetCalibration = 0                        # Variable de control para la tom
a de datos de calibración de la sonda de pH (No funcional)
StartpHPump = 0.0                               # Variable de control que almacen
a el valor a dosificar de solución de regulación de pH
pHPumpState = 0                                 # Variable de control que evita r
ealizar más de una dosificación hasta actualizar el parámetro "StartpHPump" en la
 nube
PulsesToSend = 0.0                              # Variable que almacena los pulso
s correspondientes a un valor de solución a dosificar


# Variables para ejecución en el tiempo ********************
State1 = 0.0                    # Almacena el valor de tiempo para el encendido d
e la bomba
State2 = 0.0                    # Almacena el valor de tiempo para el apagado de
la bomba
GetSensorData = 0.0             # Almacena el valor de tiempo para la ejecución d
e la adquisición de datos de la tarjeta
GetpH_ConductividadData = 0.0   # Almacena el valor de tiempo para la ejecución d
e la adquisición de la información de pH y conductividad
```

```
Error_Wait = 0.0                # Empleado para esperar a reconexion con la base
de datos

# Pines y variables del panel de control
CS = 19 # Chip Select
DT = 21 # Data pin
CK = 23 # Clock pin

LedConnect = 0x00   # Led of connect signal
LedLevel = 0x01     # Led of level
LedPH = 0x02        # Led of conductivy signal
LedPump = 0x03      # Led of pump state
LastStateLed = [0xFF,0xFF,0xFF,0xFF]    #Save last state of each led and if there
 are a set equal to last state, no set led again
ModeLed = [0x00,0x00,0x00,0x00]         # 0x00: No blink ; 0x01: Blink each 250mS
 ; 0x02: Blink each 500mS ; 0x03: Blink each 1S

def SendSerialData(DataToSend):
    global DT
    global CK
    global CS
    # Enable comunication ***********
    GPIO.output(CS, True)
    # LoopSendData **********************
    for i in range(0,8):
        buffer = DataToSend & 0x01
        if buffer == 0x01:
            GPIO.output(DT, True)
        else:
            GPIO.output(DT, False)
        # One cycle clock *****************
        GPIO.output(CK, True)
        time.sleep(104e-6)
        GPIO.output(CK, False)
        time.sleep(104e-6)
        DataToSend = DataToSend >> 1
    GPIO.output(DT, False)
    GPIO.output(CS, False)
    time.sleep(10e-3)
# Turn off all leds on panel *************
def ClearLed():
    SendSerialData(0x00)
# Set color led and blink mode on panel *************
def SetLed(R,G,B,Position,Mode):
    # Get color data and position ***************
```

```
    RGB = 0x07 & ~(R*0x01|G*0x02|B*0x04)
    PositionBuffer = Position << 3
    Data = 0x20|PositionBuffer|RGB          # computa el comando para setear el c
olor del led especificado
    # Si los datos son iguales al estado actual de los leds, retorna la función
    if (Data == LastStateLed[Position])and(Mode == ModeLed[Position]):      # Si
el color o el modo del led ya esta configurado, retorna
        return
    # Caso contrario, guarda la nueva configuración y continua la ejecución
    LastStateLed[Position] = Data           # Guardar color del led en posicion e
specificada
    ModeLed[Position] = Mode                # Guardar modo de operación del led e
n posición especificada
    # Set led ***********************************
    SendSerialData(Data)
    # Set mode **********************************
    ModeToSend = ModeLed[0]|ModeLed[1] << 2
    Data = 0x40|ModeToSend
    SendSerialData(Data)
    ModeToSend = ModeLed[2]|ModeLed[3] << 2
    Data = 0x60|ModeToSend
    SendSerialData(Data)

# Obtiene la dirección MAC *******
def getMAC(interface='eth0'):
# Return the MAC address of the specified interface
    try:
        str = open('/sys/class/net/%s/address' %interface).read()
    except:
        str = "00:00:00:00:00:00"
    return str[0:17].upper()

# Definición que maneja el encendido de la bomba y la actualización en la nube
def WritePumpON():
    WritePortEncode("P")
    ReadPort()
    WritePortEncode("O")
    ReadPort()
    print ("Pump on")
    print (IntervalTime)
    SendDataBase('/' + MAC + '-SystemData','PumpState','ON')
    SetLed(0,1,0,LedPump,0x02)
# Definición que maneja el apagado de la bomba y la actualización en la nube
def WritePumpOFF():
    WritePortEncode("P")
```

```
    ReadPort()
    WritePortEncode("F")
    ReadPort()
    print ("Pump off")
    print (WateringTime)
    SendDataBase('/' + MAC + '-SystemData','PumpState','OFF')
    SetLed(1,0,0,LedPump,0x00)
# Definición que maneja el envio de datos a la tarjeta de adquisición
def WritePortEncode(DataToSend):
    Mainboard.write(DataToSend.encode())
# Definición que maneja la recepción de datos de la tarjeta de adquisición
def ReadPort():
    InputData = Mainboard.readline().strip()
    Decoding = str(InputData.decode('ascii'))
    return Decoding
# Definición que calcula el nivel actual del tanque en cm y en %
def RealLevelTank(Level):
    global PorcentLevel
    global OffsetTank
    offset = OffsetTank
    RealLevel = round(float(TankHeight)-float(Level),2)
    PorcentLevel = str(round((RealLevel/(TankHeight-offset))*100,2))
    return str(RealLevel)
# Obtiene el valor de la conductividad de referencia (1413uS-
25°C) a la temperatura actual
def GetReferenceCond(Temperature):
    # Compute real conductivy with actual Temperature
    # Note: This funtion only must to use on (1413uS-
25°C) reference solution ***************
    Conductivy = -
0.0017*Temperature**3 + 0.1526*Temperature**2 + 22.541*Temperature + 780.07
    return Conductivy
# Calcula la constante de celda del electrodo
def ComputeCellConstant(InputVoltage,ConductivyRef):
    CellConstant = ConductivyRef/InputVoltage
    return CellConstant
# Lee los valores de calibración de los sensores ********************
def ReadCalibration():
    global SystemTemperatureCoefficientVariation
    global SystemCellConstant
    global pHMin
    global pHMax
    global LevelLimit
    global LevelMin
    global LevelMax
```

```
    Calibrationfile = open("/home/pi/Documents/Calibration.txt","r")
    lines = Calibrationfile.readlines()
    SystemCellConstant = float(lines[0])                        # Lee la constant
e de la celda de conductividad desde archivo de configuración
    SystemTemperatureCoefficientVariation = float(lines[1])     # Lee el coeficie
nte de variación de temperatura desde el archivo de configuración
    pHMin = float(lines[2])                                     # Lee el valor de
 pH minimo para enviar alerta desde el archivo de configuración
    pHMax = float(lines[3])                                     # Lee el valor de
 pH maximo para enviar alerta desde el archivo de configuración
    LevelLimit = float(lines[4])                                # Lee el nivel li
mite minimo del tanque que evita el encendido de la bomba
    LevelMin = float(lines[5])                                  # Lee el valor mi
nimo de nivel del tanque para enviar indicación luminica
    LevelMax = float(lines[6])                                  # Lee el valor su
perior de nivel del tanque sobre el cual se considera un nivel normal
    Calibrationfile.close()
# Calibra la constante de celda de la sonda de conductividad ****************
******
def CellConstantCalibration(TemperatureNow,Voltage):
    ConductivyNow = GetReferenceCond(TemperatureNow)
    print (ConductivyNow)
    print (Voltage)
    NewCalibration = round(ComputeCellConstant(Voltage,ConductivyNow),2)
    print (NewCalibration)
    Calibrationfile = open("/home/pi/Documents/Calibration.txt","r")
    lines = Calibrationfile.readlines()
    Calibrationfile.close()
    lines[0] = str(NewCalibration) + "\n"
    lines[1] = "0\n"            # NO Temperature Coefficient Variation of calibra
tion *************
    Calibrationfile = open("/home/pi/Documents/Calibration.txt","w")
    Calibrationfile.writelines(lines)
    Calibrationfile.close()
    # Set New Values on system ********
    ReadCalibration()
# Calcula el coeficiente de variacion de temperatura empleado para la compensació
n de la conductividad
def ComputeTemperatureCoefficientVariation(ActualConductivy,ReferenceConductivy,A
ctualTemperature,ReferenceTemperature):
    print (ActualConductivy)
    TemperatureCoefficientVariation = ((ActualConductivy-
ReferenceConductivy)*100)/(ReferenceConductivy*(ActualTemperature-
ReferenceTemperature))
    return TemperatureCoefficientVariation
```

```
# Calcula la conductividad con la compensación de temperatura
def ComputeLinearTemperatureCompensation(TemperatureCoefficientVariation,ActualConductivy,ActualTemperature,ReferenceTemperature):
    ConductivyCompensation = ActualConductivy/(1 + (TemperatureCoefficientVariation/100)*(ActualTemperature-ReferenceTemperature))
    return round(ConductivyCompensation,0)
# Calcula el coeficiente de variacion termica de la solución
def TemperatureCoefficientVariationCalibration(ActualConductivy,ReferenceConductivy,ActualTemperature,ReferenceTemperature):
    NewCalibration = round(ComputeTemperatureCoefficientVariation(ActualConductivy,ReferenceConductivy,ActualTemperature,ReferenceTemperature),2)
    Calibrationfile = open("/home/pi/Documents/Calibration.txt","r")
    lines = Calibrationfile.readlines()
    Calibrationfile.close()
    lines[1] = str(NewCalibration) + "\n"
    Calibrationfile = open("/home/pi/Documents/Calibration.txt","w")
    Calibrationfile.writelines(lines)
    Calibrationfile.close()
    # Set New Values on system ********
    ReadCalibration()
# Calcula los coeficientes de la correlacióon lineal de un conjunto de valores x, y ***********
def LinealRegresion(X,Y):
    if len(X) == len(Y):
        n = len(X)                  # Numero de datos para trabajar
        Ex  = sum(X)                # Sumatoria de todas las X
        Ey  = sum(Y)                # Sumatoria de todas las Y
        Exy = float(sum(numpy.array(X)*numpy.array(Y))) # Sumatoria de X*Y
        Ex2 = float(sum(numpy.array(X)*numpy.array(X))) # Sumatoria de X^2
        # Compute lineal regresion coeficients ******************
        a = (n*Exy-Ex*Ey)/(n*Ex2-Ex*Ex)
        b = (Ey-a*Ex)/n
        return (a,b)
    else:
        print ('Data lenth is no correct!')
        return (1,0)    # y = aX + b with a = 1 and b = 0
# Send Data to database with error signal
def SendDataBase(DataBasepatch,DataBaseTag,DataBaseData):
    try:
        if DataBaseTag == None:
            Database.patch(DataBasepatch,DataBaseData)
        else:
            Database.put(DataBasepatch,DataBaseTag,DataBaseData)
        return 0        # If there're not error, return 0
    except:
```

```
        return 1        # If there're error, return 1
# Definición que se encarga de la descarga de datos de la nube y la sincronizació
n de información
def DataBaseComunication():
    global EnableMainProgram
    global IntervalTime
    global WateringTime
    global TankHeight
    global StartPumpNow
    global UpdateSetup
    global UpdateCalibration
    global StartTime
    global SleepTime
    global State1
    global State2
    global UpdateCellConstant
    global UpdateTemperatureCompensation
    global ReferenceConductivy
    global ReferenceTemperature
    global mpH
    global bpH
    global OffsetTank
    global WTCTempMax
    global WTCTempMin
    global HighTempIntervalTime
    global IntervalTimeBuffer
    global MAC
    global FlagChangeIntervalTime
    global StartpHPump
    StartUpdate = "1"
    WateringTimeBuffer = 0.0
    ReadDataBase = 0.0
    SuccessfulReconnect = 0
    while True:
        # Try to conect to database. If an error happened, a catch code will be l
aunched
        try:
            # Get system Setup **********************
            if (time.perf_counter()-ReadDataBase)>= 5: # Update each 5S
                BufferControl = Database.get('/' + MAC + '-SystemControl','')
                StartPumpNow = BufferControl.get('StartPumpNow').replace('"','')
                StartpHPump = float(BufferControl.get('StartpHPump').replace('"',
''))
                UpdateCalibration = BufferControl.get('UpdateCalibration').replac
e('"','')
```

```
                UpdateSetup = BufferControl.get('UpdateSetup').replace('"','')
                # Update Setup to start ***************
                if StartUpdate == "1":
                    StartUpdate = "0"
                    UpdateCalibration = "1"
                    UpdateSetup = "1"
                # Update Setup data flag ********************
                if UpdateSetup == '1':
                    SendDataBase('/' + MAC + '-
SystemControl','UpdateSetup','"0"')
                    BufferSetup = Database.get('/' + MAC + '-SystemSetup','')
                    # Set setup from database *************
                    IntervalTimeBuffer = BufferSetup.get('IntervalTime').replace(
'"','')
                    WateringTimeBuffer = BufferSetup.get('WateringTime').replace(
'"','')
                    StartTime = BufferSetup.get('StartTime').replace('"','')
                    SleepTime = BufferSetup.get('SleepTime').replace('"','')
                    TankHeight = float(BufferSetup.get('TankHeight').replace('"',
''))
                    WTCTempMax = float(BufferSetup.get('WTCTempMax').replace('"',
''))
                    WTCTempMin = float(BufferSetup.get('WTCTempMin').replace('"',
''))
                    HighTempIntervalTime = float(BufferSetup.get('HighTempInterva
lTime').replace('"',''))
                    # Conversiones **********************
                    WateringTime = float(WateringTimeBuffer)*60 # Convert to Seco
nds; Time*60S
                    IntervalTime = float(IntervalTimeBuffer)*60 # Convert to Seco
nds; Time*60S
                    # Reajustar variables del sistema ***************
                    FlagChangeIntervalTime = 2
                    print ("Setup Completed")
                # Update Calibration data flag
                if UpdateCalibration == "1":
                    SendDataBase('/' + MAC + '-
SystemControl','UpdateCalibration','"0"')
                    BufferCalibration = Database.get('/' + MAC + '-
SystemCalibration','')
                    # Get calibration values and conditions ********************+
                    CondCalibrationBuffer = BufferCalibration.get('CondCalibratio
n').replace('"','')
                    OffsetTankBuffer = BufferCalibration.get('OffsetTank').replac
e('"','')
```

```
                    UpdateCellConstant = BufferCalibration.get('UpdateCellConstan
t').replace('"','')
                    UpdateTemperatureCompensation = BufferCalibration.get('Update
TemperatureCompensation').replace('"','')
                    pHCalibrationBuffer = BufferCalibration.get('pHCalibration').
replace('"','')
                    # Get conductivy calibration values ************************
****
                    CondCalibration = CondCalibrationBuffer.split(":")

                    ReferenceConductivy = float(CondCalibration[0])
                    ReferenceTemperature = float(CondCalibration[1])
                    # Tank offset ******************************************
                    OffsetTank = float(OffsetTankBuffer)
                    # pHCalibration ****************************************
                    pHCalibration = pHCalibrationBuffer.split(":")
                    mpH = float(pHCalibration[0])
                    bpH = float(pHCalibration[1])
                EnableMainProgram = 1
                ReadDataBase = time.perf_counter()

                # Reconnect message *********************
                if SuccessfulReconnect == 1:
                    SuccessfulReconnect = 0
                    print ("Successful reconnect!!")
        except:
            print ("An unexpected error has occurred\nFail to get data from Fireb
ase\nThe system will reboot in 5 seconds")
            time.sleep(5)
            print ("Try to conect to database...")
            SuccessfulReconnect = 1
if __name__== '__main__':
    print ("System init...")
    #************** Setup Serial Comunication ***************
    Mainboard = serial.Serial("/dev/ttyUSB0", baudrate=115200)
    #************** Setup panel control *********************
    try:
        GPIO.cleanup()
    except:
        pass
    GPIO.setmode(GPIO.BOARD)
    # Disable comunication while setup ***********
    GPIO.setup(CS, GPIO.OUT)
    GPIO.output(CS, False)
    # Setup of pinout *********************
```

```
    GPIO.setup(DT, GPIO.OUT)
    GPIO.output(DT, False)
    GPIO.setup(CK, GPIO.OUT)
    GPIO.output(CK, False)
    # turn off all leds ***************
    ClearLed()
    time.sleep(5)
    #*************** Read Calibration Values *****************
    ReadCalibration()       # Read all calibration values
    #*************** General Setup *************
    State1 = time.perf_counter()    # Asegura que la bomba empiece apagada
    UpdateSetup = '1'               # Actualizacion de configuracion del sistema
desde la nube
    SuccessfulReconnect = 0         # No connect problems start
    ErrorFlag = 0                   # Reset flag system error
    #*************** Setup FireBase Realtime Database ********
    Database = firebase.FirebaseApplication("https://hydroponic-
supervision.firebaseio.com/",None)
    #*************** Get device MAC ******************
    MAC = getMAC('wlan0')
    #*************** Register Device ******************************************
*******
    # All this data could be modify ***********
    DevicesRegister = Database.get('Devices','')
    if DevicesRegister.get(MAC) == None: # Intenta encontrar la dirección MAC de
la raspberry en la base de datos,
        SendDataBase('Devices',MAC,'True')  # En caso de que no se haya encontrad
o ninguna coincidencia, registra el dispositivo en la Base de datos
        # Crea una nueva instancia para almacenar las controles del sistema *****
************
        DataToSend = {'StartPumpNow':'0','StartpHPump':'0','UpdateCalibration':'0
','UpdateSetup':'0'}
        SendDataBase('/' + MAC + '-SystemControl/',None,DataToSend)
        # Crea una nueva instancia para almacenar las datos del sistema *********
********
        DataToSend = {'ActualIntervalTime':'0','Conductivy':'0','EnvironmentTemp'
:'0','LevelTank':'0','PorcentLevel':'0','PumpState':'OFF','Sleep':'No','TempSolut
ion':'0','pHSolution':'0'}
        SendDataBase('/' + MAC + '-SystemData/',None,DataToSend)
        # Crea una nueva instancia para almacenar los parametros de configuración
 del sistema ******
        DataToSend = {'HighTempIntervalTime':'15','IntervalTime':'30','SleepTime'
:'22h00','StartTime':'7h00','TankHeight':'50','WTCTempMax':'25','WTCTempMin':'23'
,'WateringTime':'10'}
        SendDataBase('/' + MAC + '-SystemSetup/',None,DataToSend)
```

```
        # Crea una nueva instancia para almacenar los parametros de calibración d
el sistema ******
        DataToSend = {'CondCalibration':'1200:25','OffsetTank':'0','SetActualpHVa
lue':'0','UpdateCellConstant':'0','UpdateTemperatureCompensation':'0','UpdatepHCa
libration':'0','pHCalibration':'1:0'}
        SendDataBase('/' + MAC + '-SystemCalibration/',None,DataToSend)
        print ("New device register on Database")
    else:
        print ("This device is already register")    # en caso de que exista, el
dispositivo ya fue registrado con anterioridad
        SendDataBase('/Devices',MAC,'true')
    #************* Start threading task ********************
    print ("Wait for host...")
    threadSetState = threading.Thread(target = DataBaseComunication)
    threadSetState.daemon = True
    threadSetState.start()
    # Wait for Database conection ************
    SetLed(1,0,0,LedConnect,0x02)       # Set Led Connect as red and blink each 5
00mS
    while EnableMainProgram == 0:
        pass
    print ("Conect sucessfull !!")
    time.sleep(1)
    SetLed(0,1,0,LedConnect,0x00)       # Set Led Connect as green en not blink
    WritePumpOFF()                      # Inicia con la bomba apagada
    #************ Main loop ******************************
    while True:
        #****************************************** 1° state, trigged by time = h
our'o clock ****************************************
        # Get time of the system
        TimeNow = datetime.datetime.now()
        HourNow = TimeNow.hour
        MinuteNow = TimeNow.minute

        # Send current Data each hour o clock
        if (HourNow != HourBuffer) and (EnableSendDataHistory == 1):
            if MinuteNow == 0:
                HourBuffer = HourNow
                # Add a "0" if hour is less than 10
                if HourNow < 10:
                    HourToSend = "0" + str(HourNow)
                else:
                    HourToSend = str(HourNow)
                # Add a "0" if minute is less than 10
                if MinuteNow < 10:
```

```
                    MinuteToSend = "0" + str(TimeNow.minute)
                else:
                    MinuteToSend = str(TimeNow.minute)
                # Add a "0" if day is less than 10
                if TimeNow.day < 10:
                    DayToSend = "0" + str(TimeNow.day)
                else:
                    DayToSend = str(TimeNow.day)
                # Add a "0" if month is less than 10
                if TimeNow.month < 10:
                    MonthToSend = "0" + str(TimeNow.month)
                else:
                    MonthToSend = str(TimeNow.month)

                BufferDate = MonthToSend + "-" + DayToSend      # Formato Mes-Día
                BufferTime = HourToSend + "h" + MinuteToSend    # Formato de hora
, 24 horas
                DataToSend = {'Conductivy':ConductivyBuffer,'EnvironmentTemp':Tem
pEnvironment,'LevelTank':LevelTankNow,'PorcentLevel':PorcentLevel + "%",'TempSolu
tion':TempSolution,'pHSolution':str(round(phSolutionBuffer,1))}
                if ErrorFlag == 0:
                    # Guarda el historico desglosando el año y almacenando dentro
 de cada año el mes y el día correspondiente a los datos tomados
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemHistory/' + str(TimeNow.year) + "/" + BufferDate + "/" + BufferTime,None,Da
taToSend)
        # ********************************* 2º State, trigged by time between a r
ange ***********************************************
        # Turn off System pump each Sleep period ************************
        TimePeriodNow = time.strftime("%X")     # Get current time **************
*+
        StartPeriod = StartTime.replace("h",":") + ":00"
        EndPeriod = SleepTime.replace("h",":") + ":00"
        # Convert to variable time
        TPNow = datetime.datetime.strptime(TimePeriodNow, "%X").time()
        SPTime = datetime.datetime.strptime(StartPeriod, "%X").time()
        EPTime = datetime.datetime.strptime(EndPeriod, "%X").time()
        if (TPNow >=SPTime)and(TPNow<=EPTime):
            if EnablePump == 0:
                WritePumpON()
                PowerOFFPump = 1
                State1 = time.perf_counter()
                State2 = time.perf_counter()
                if ErrorFlag == 0:
```

```
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemData','Sleep','No')
            if EnablePump == 2:
                if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemData','Sleep','No')
            EnablePump = 1
        else:
            if EnablePump == 1:
                if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemData','Sleep','Yes')
            EnablePump = 0
            if PowerOFFPump == 0:
                SetLed(1,1,0,LedPump,0x00)
                State1 = time.perf_counter()

        # ********************************** 3° State, triggered by StartPumpNow
 its more than 0  ****************************************
        # Manual Control Pump ********************************
        if StartPumpNow == "1":                                 # Turn on Pump
            WritePumpON()
            PowerOFFPump = 1
            State1 = time.perf_counter()
            State2 = time.perf_counter()
            if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemControl','StartPumpNow','"0"')
        if StartPumpNow == "2":                                 # Turn off Pump
            WritePumpOFF()
            PowerOFFPump = 0
            State1 = time.perf_counter()
            State2 = time.perf_counter()
            if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemControl','StartPumpNow','"0"')


        # ********************************** 4° State, triggered by time >= Inte
rvalTime and PowerOFFPump == 0 ******************
        #Timer pump control ***********
        # Turn on Pump when time elapsed = IntervalTime and pump it's OFF
        if (time.perf_counter()-
State1) >= IntervalTime: # Execute each IntervalTime
            if PowerOFFPump == 0:
```

```
                PowerOFFPump = 1
                State2 = time.perf_counter()
                WritePumpON()

        # ********************************** 5° State, triggered by time >= Wate
ringTime and PowerOFFPump == 1 ******************
        # Turn off Pump when time elapsed = WateringTime and pump it's ON
        if (time.perf_counter()-
State2) >= WateringTime: # Execute each WateringTime
            if PowerOFFPump == 1:
                State1 = time.perf_counter() - WateringTime # Se le resta el tiem
po de riego para sincronizar los tiempos de encendido y apagado
                PowerOFFPump = 0
                WritePumpOFF()

        # ********************************** 6° State, trigged by time elapsed =
 1S ********************************************
        # Get Sensor Data (Temperature, Conductivy solution and level water)
        if (time.perf_counter()-GetSensorData) >= 1:    # Excute each 1S
            GetSensorData = time.perf_counter()
            WritePortEncode("R")
            TempSolution = ReadPort()
            TempEnvironment = ReadPort()
            LevelTank = ReadPort()
            ReadPort()
            LevelTankNow = RealLevelTank(LevelTank)
            DataToSend = {'EnvironmentTemp':TempEnvironment,'LevelTank':LevelTank
Now,'PorcentLevel':PorcentLevel + "%",'TempSolution':TempSolution}
            # Set level Led ********************************
            if float(PorcentLevel) < LevelLimit:
                SetLed(1,0,0,LedLevel,0x02)     # Set Led Level as red, blink 500
mS
                State1 = time.perf_counter()    # Resetea el contador de tiempo d
e encendido, impidiendo que la bomba encienda automaticamente
            elif float(PorcentLevel) < LevelMin:
                SetLed(1,0,0,LedLevel,0x00)     # Set Led Level as red, no blink
            elif float(PorcentLevel)  > LevelMax:
                SetLed(0,1,0,LedLevel,0x00)     # Set Led Level as green, no blin
k
            else:
                SetLed(1,1,0,LedLevel,0x00)     # Set Led Level as yellow, no bli
nk
            if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemData/',None,DataToSend)
```

```
        # ********************************** 7º State, trigged by time elapsed =
 10S ******************************************
        # Get pH and conductivy values ***************************************
**********
        if (time.perf_counter()-
GetpH_ConductividadData) >= 10:  # Excute each 10S
            GetpH_ConductividadData = time.perf_counter()
            pHLastState = float(pHSolution)                 # Save last pH value
***********
            WritePortEncode("H")
            pHSolution = ReadPort()
            ConductivySolution = ReadPort()
            ReadPort()
            # Compute Conductivy ******************
            ConductivySystemNow = SystemCellConstant*float(ConductivySolution)
            # Compute TemperatureCompensation ******
            RealConductivy = ComputeLinearTemperatureCompensation(SystemTemperatu
reCoefficientVariation,ConductivySystemNow,float(TempSolution),ReferenceTemperatu
re)
            ConductivyBuffer = str(RealConductivy) + "uS"
            # Compute pH value *********************
            phSolutionBuffer = mpH*round(float(pHSolution),2)+bpH   # Para una re
solución de salida de 0.1, se necesita un cambio de 0.02 mV (dos decimas)
            DataToSend = {'Conductivy':ConductivyBuffer,'pHSolution':str(round(ph
SolutionBuffer,2))}
            # Set led state ************************
            if phCalibrationEnable == False:
                if round(phSolutionBuffer,1) < pHMin:
                    SetLed(1,0,0,LedPH,0x02)        # Set Led pH as red, blink 50
0mS
                elif round(phSolutionBuffer,1) > pHMax:
                    SetLed(0,0,1,LedPH,0x03)        # Set Led pH as blue, blink 1
S
                else:
                    SetLed(0,0,1,LedPH,0x00)        # Set Led pH as blue, no blin
k
            if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemData/',None,DataToSend)
            EnableSendDataHistory = 1

        # ******************************** 9º State, Calibrate Cell Constant ***
**************************************************
        if UpdateCellConstant == '1':
```

```
            CellConstantCalibration(float(TempSolution),float(ConductivySolution)
)
            UpdateCellConstant = '0'
            if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemCalibration','UpdateCellConstant','"0"')

        # ******************************** 10° State, Calibrate Temperature Coef
ficient Variation ***********************************
        if UpdateTemperatureCompensation == '1':
            TemperatureCoefficientVariationCalibration(ConductivySystemNow,Refere
nceConductivy,float(TempSolution),ReferenceTemperature)
            UpdateTemperatureCompensation = '0'
            if ErrorFlag == 0:
                    ErrorFlag = SendDataBase('/' + MAC + '-
SystemCalibration','UpdateTemperatureCompensation','"0"')

        # ******************************** Calibrate pH Sensor (No funtional) **
************************************************************
        if phCalibrationEnable == True:
            SetLed(1,0,1,LedPH,0x02)
            if (abs(float(pHSolution) - pHLastState) < 0.1)and(abs(pHChange-
float(pHSolution)) < 1):
                # Set led calibration ok **************
                SetLed(1,0,1,LedPH,0x00)
                # Save actual voltage *****************
                if EnableSetCalibration == 1:
                    if phCalSolution == 1:
                        pHCalMin = float(pHSolution)
                    elif phCalSolution == 2:
                        pHCalMax = float(pHSolution)
                    elif phCalSolution == 3:
                        pHCalMed = float(pHSolution)
                        pH_X = (pHCalMin,pHCalMax,pHCalMed)
                        pH_Y = (4,9.18,6.86)
                        a,b = LinealRegresion(pH_X,pH_Y)
                        print (a)
                        print (b)
                        phCalibrationEnable = False
                    pHChange = float(pHSolution)
                    phCalSolution = phCalSolution + 1
                    EnableSetCalibration = 0
        else:
            pHLastState = 0
            phCalSolution = 1
```

```
            pHChange = 0

        # ******************************** 11° State, PH pump control function *
**************************************************
        if StartpHPump > 0:
            if pHPumpState == 0:
                pHPumpState = 1
                PulsesToSend = int(round(3.8503*StartpHPump+0.05,0))
                if PulsesToSend < 10:
                    PulsesToSend = "000" + str(PulsesToSend)
                elif PulsesToSend < 100:
                    PulsesToSend = "00" + str(PulsesToSend)
                elif PulsesToSend < 1000:
                    PulsesToSend = "0" + str(PulsesToSend)
                else:
                    PulsesToSend = str(PulsesToSend)
                print("Dosis: " + str(StartpHPump) + ", Pulsos: " + PulsesToSend)
                WritePortEncode("-")
                WritePortEncode(PulsesToSend)
                ReadPort()
            if ErrorFlag == 0:
                ErrorFlag = SendDataBase('/' + MAC + '-
SystemControl','StartpHPump','"0"')
        else:
            pHPumpState = 0
        # ******************************** 12° State, Watering time control temp
erature ********************************************
        if (float(TempEnvironment) < WTCTempMin) and ((FlagChangeIntervalTime ==
1)or(FlagChangeIntervalTime == 2)):
            IntervalTime = float(IntervalTimeBuffer)*60
            FlagChangeIntervalTime = 0
            if ErrorFlag == 0:
                ErrorFlag = SendDataBase('/' + MAC + '-
SystemData','ActualIntervalTime',IntervalTimeBuffer)
        if (float(TempEnvironment) > WTCTempMax)and((FlagChangeIntervalTime == 0)
or(FlagChangeIntervalTime == 2)):
            IntervalTime = HighTempIntervalTime*60
            FlagChangeIntervalTime = 1
            if ErrorFlag == 0:
                ErrorFlag = SendDataBase('/' + MAC + '-
SystemData','ActualIntervalTime',str(HighTempIntervalTime))
        # ******************************** 13° State, triggered by error to conn
ect to network ***********************************
        if ErrorFlag == 1:          #If ErrorFlag its set, run handler error
```

```
            if SuccessfulReconnect == 0:    # This mean the error is recent, and
send Error message
                print ("An unexpected error has occurred\nFail to send data to Fi
rebase\nThe system will reboot in 5 seconds")
                Error_Wait = time.perf_counter()        # Update Time counter to
reconnect
                SuccessfulReconnect = 1                 # This code execute once,
 flag to point a reconnect
                SetLed(1,1,0,LedConnect,0x01)           # Set Led Connect as yell
ow and blink each 250mS
            if (time.perf_counter()-
Error_Wait) >= 5:   # When 5S wait time elapsed, try to reconnet
                print ("Try to conect to database...")
                Error_Wait = time.perf_counter()        # Update Time counter to
reconnect
                ErrorFlag = 0                           # Clear this flag to try
to reconnect
        else:
            if SuccessfulReconnect == 1:                # If reconnect it´s succe
ssful, send message
                SuccessfulReconnect = 0                 # Clear flag
                SetLed(0,1,0,LedConnect,0x00)           # Set Led Connect as gree
n en not blink
                print ("Successful reconnect!!")
```

6.2. Código del Arduino

Código desarrollado en C/C+ empleando la IDE de arduino.

```
#include <PinChangeInt.h>
#include <DallasTemperature.h>
#include <OneWire.h>


//**************** Definition and pines *****************
#define conductividadSensor 0
#define pHSensor 1
#define DS18B20Bus 2
#define RelePin 3
#define pintrigger 4       // Pin trigger del sensor HC-SR04
#define pinecho 5          // Pin echo del sensor HC-SR04
#define CaudalSensor 9
#define pHPump 10          //
#define LedFlow 13         // LED to point the flow sensor signal

DeviceAddress SensorTempEnvironment = {0x28, 0x9A, 0x09, 0x79, 0xA2, 0x00,
0x03, 0xF3}; // Dirección del sensor de temperatura del medio ambiente
DeviceAddress SensorTempSolution = {0x28, 0x9B, 0xB9, 0x79, 0x97, 0x11, 0x03,
0x48};    // Dirección del sensor de temperatura de la solución

// Variables de uso general ******************************
unsigned char data;                     // Variable para almacenar el dato
recibido
bool Antirebote = false;                // Antirebote para los pulsos del
caudalímetro
int CondTimes = 0;                      // Almacena la cantidad de veces que
se ha sumado el valor de la conductividad
int pHTimes = 0;                        // Almacena la cantidad de veces que
se ha sumado el valor del pH
int PulseConvert = 0;                   // Variable empleada para calcular
el valor de los pulsos de referencia
int ReferencePulses = 0;                // Variable que contiene los pulsos
de referencia del sensor
float Condnow = 0;                      // Valor actual de conductividad
float Conductividad = 0;                // Variable para almacenar el valor
de conductividad
float EchoTime = 0;                     // Variable para almacenar la
duración del eco del sensor
float PH = 0;                           // Variable para almacenar el valor
de pH
float PHnow = 0;                        // Valor actual de pH
float PrompH = 0;                       // Variable para almacenar el
promedio de pH
float PromCond = 0;                     // Variable para almacenar el
promedio de conductividad
float TankLevel = 0;                    // Variable para almacenar el
porcentaje del tanque
float TemperaturaAmbiente = 0;          // Temperatura del ambiente en
grados centigrados
float TemperaturaSolucion = 0;          // Temperatura de la solución en
grados centigrados
volatile int Count = 0;                 // Variable que almacena la cuenta
de pulsos del sensor
volatile bool CountNow = false;         // Habilita el conteo de pulsos del
caudalímetro
```

```
// Variables to time operations *************************
unsigned long currentMillis = 0;
unsigned long resetmillis = 0;
unsigned long state1 = 0;
unsigned long state2 = 0;
unsigned long state3 = 0;

//***************** Inicialización de objetos ******************
OneWire oneWire(DS18B20Bus);                  // Establece el pin del sensor
como bus OneWire
DallasTemperature SensorTemp(&oneWire);       // Inicia objeto OneWire

// Handler of interrupt to sensor flow ************************
void SensorFlow()
{
  digitalWrite(LedFlow, digitalRead(CaudalSensor)); // Indicador visual del
sensor de flujo
  if (Antirebote  == false) {
    Antirebote = true;                              // Evita que se
contabilice más de un pulso cuando se detecta un estado lógico alto
    CountSpinSensor();                              // Call the count
function
  } else {
    Antirebote = false;                             // Se habilita el conteo
una vez la señal regrese a estado lógico bajo
  }
}
// Flow sensor pulse counter **********************
void CountSpinSensor()
{
  if (CountNow == true)
  {
    Count += 1;                    // Incrementa el contador
    if (Count == ReferencePulses) // Si la cuenta de pulsos alcanza el valor
de referencia, se apaga la bomba
    {
      CountNow = false;            // Deshabilita la cuenta de pulsos
      Count = 0;                   // Reinicia el contador
      digitalWrite(pHPump, HIGH); // Apaga la bomba
    }
  }
}
// Turn on the pH pump ***************************
void Turn_On_pH_Pump()
{
  digitalWrite(pHPump, LOW);      // Enciende la bomba de pH
  CountNow = true;                // Habilita el conteo de pulsos
}
// Gets the value of the reference pulses ********
void GetpHPulses() // The pulses must be between 1 to 9999
{
  PulseConvert = 0;
  for (int X = 0; X < 4; X++)
  {
    Wait_for_data();
    data = data - 48; // Convert ascill char to number (Only for ascill char
0-9)
```

```
    switch (X)
    {
      case 0:
        PulseConvert = (int)(data) * 1000;                  // Thousand
        break;
      case 1:
        PulseConvert = (int)(data) * 100 + PulseConvert;    // Hundred
        break;
      case 2:
        PulseConvert = (int)(data) * 10 + PulseConvert;     // Ten
        break;
      case 3:
        PulseConvert = (int)(data) + PulseConvert;          // Unit
        break;
    }
  }
  ReferencePulses = (int)(PulseConvert); // Update reference pulses variable
  Turn_On_pH_Pump();
}

// Get the temperature values **************
void ReadTemperature()
{
  // Temperatura ***************************
  SensorTemp.requestTemperatures();                                          // Se
envía el comando para leer la temperatura
  TemperaturaSolucion = SensorTemp.getTempC(SensorTempSolution);       // Se
obtiene la temperatura en °C del sensor de la solución
  TemperaturaAmbiente = SensorTemp.getTempC(SensorTempEnvironment);    // Se
obtiene la temperatura en °C del sensor del ambiente
}

// Measure the level of solution ***********
void MeasureTankLevel()
{
  digitalWrite(pintrigger, HIGH);     // Send trigger pulse on "TRIGGER" pin
  delayMicroseconds(10);              // Almost 10 uS on high state
  digitalWrite(pintrigger, LOW);      // Set low state on "TRIGGER" pin
  EchoTime = pulseIn(pinecho, HIGH);  // Measure high state time of "ECHO"
pin. This time is proporcional to measure distance
  // LA VELOCIDAD DEL SONIDO ES DE 340 M/S O 29 MICROSEGUNDOS POR CENTIMETRO
  // DIVIDIMOS EL TIEMPO DEL PULSO ENTRE 58, TIEMPO QUE TARDA RECORRER IDA Y
VUELTA UN CENTIMETRO LA ONDA SONORA
  TankLevel = EchoTime / 58;
}

// Calculate the pH value ********
void ReadPH()
{
  // pH ***********************************
  PH = PrompH / pHTimes; // pH promediado
  PrompH = 0;            // Reset PrompH
  pHTimes = 0;           // Reset pHTimes
}

// Makes sum of all sensor pH readings *****
void CalcpHProm()
```

```
{
  // Get pH value
  PHnow = analogRead(pHSensor);
  PHnow = (PHnow * 5) / 1023;       // Conversión
  // pH Promedio ***************************
  PrompH = PHnow + PrompH;              // Sumatoria de cada lectura del
sensor
  pHTimes += 1 ;                        // Contabiliza las veces que se ha
realizado una lectura
}

// Calculate the conductivity value ********
void ReadConductivity()
{
  // Conductividad ************************
  Conductividad = PromCond / CondTimes; // Conductividad promediada
  PromCond = 0;                         // Reset PromCond
  CondTimes = 0;                        // Reset CondTimes
}

// Makes sum of all conductivy sensor readings ***
void CalcCondProm()
{
  Condnow = analogRead(conductividadSensor);
  Condnow = (5 * Condnow) / 1023;
  PromCond = Condnow + PromCond;        // Sumatoria de cada lectura del
sensor
  CondTimes += 1;                       // Contabiliza las veces que se ha
realizado una lectura
}

// Send ambient and solution temperature and level data ******
void ReadStates()
{
  Serial.println(TemperaturaSolucion);
  Serial.println(TemperaturaAmbiente);
  Serial.println(TankLevel);
}

// Send pH and conductivity data *****************************
void SendPH_Conductividad()
{
  Serial.println(PH);
  Serial.println(Conductividad);
}

// Wait for a serial data ******************
void Wait_for_data()
{
  while (Serial.available() == 0)
  {
    data = ' ';
  }
  data = Serial.read();
}
```

```
// Relay control ****************************
void Rele()
{
  Wait_for_data();
  switch (data)
  {
    case 'O':                       // Turn ON pump with "O" command
      Serial.println('o');
      digitalWrite(RelePin, HIGH);
      break;
    case 'F':                       // Turn OFF pump with "F" command
      Serial.println('f');
      digitalWrite(RelePin, LOW);
      break;
  }
}
// Serial command controller ****************
void SerialRequest()
{
  if (Serial.available() > 0)
  {
    data = Serial.read();
    switch (data)
    {
      case 'R':
        ReadStates();
        Serial.println('r'); // Read sensor data ***************
        break;
      case 'H':
        SendPH_Conductividad();
        Serial.println('h'); // Read pH data ******************
        break;
      case 'P':
        Serial.println('p'); // Turn on/off pump ***************
        Rele();
        break;
      case '-':
        GetpHPulses();
        Serial.println('c'); // Write decrement pH control ***************
        break;
      default:
        Serial.println('E'); // In case of error send E ********
        break;
    }
  }
}
void setup() {
  // Start Serial comunication at 115200 bdps *********************
  Serial.begin(115200);
  //******************** Set all inputs and outputs ***************
  pinMode(pinecho, INPUT);
  pinMode(RelePin, OUTPUT);
  pinMode(pintrigger, OUTPUT);
  pinMode(CaudalSensor, INPUT);
  pinMode(pHPump, OUTPUT);
  pinMode(LedFlow, OUTPUT);
```

```
  //******************** Set and Clear initial ********************
  digitalWrite(pintrigger, LOW);
  digitalWrite(RelePin, LOW);     // High turn on relay, Low turn off relay
  digitalWrite(pHPump, HIGH);     // High turn off pump, Low turn on pump
  digitalWrite(LedFlow, LOW);
  //******************** Inital Setup ******************************
  resetmillis = millis();                                    // Stores an
initial time value
  SensorTemp.begin();
  attachPinChangeInterrupt(CaudalSensor, SensorFlow, CHANGE);   // Habilita
interrupción por cambio de estado en el pin SensorFlow
}
void loop() {
  unsigned long currentMillis = millis();

  //actualización al desborde ****************************
  if ((unsigned long)(resetmillis > currentMillis)) {           // Si el
valor de millis almacenado en resetmillis es mayor al valor de currentMillis,
ha ocurrido un desborde
    state1 = currentMillis;
    state2 = currentMillis;
    state3 = currentMillis;
    // Reset millis
    resetmillis = currentMillis;
  }
  // Ejecucion cada 20 milisegundos ************************************
  if ((unsigned long)(currentMillis - state1) >= 20) {
    SerialRequest();          // Serial request
    state1 = currentMillis;
  }
  // Ejecucion cada 500 milisegundos (0.5 segundos) ********************
  if ((unsigned long)(currentMillis - state2) >= 500) {
    ReadTemperature();   // Read Temperature Sensor
    CalcCondProm();      // Calc conductividad promedio
    CalcpHProm();        // Calc pH promedio
    MeasureTankLevel();  // Read Tank Level
    state2 = currentMillis;
  }
  // Ejecucion cada 10000 milisegundos (10 segundos) ******************
  if ((unsigned long)(currentMillis - state3) >= 10000) {
    ReadPH();            // Read pH Sensor
    ReadConductivity();  // Read Conductivity Sensor
    state3 = currentMillis;
  }
}
```

6.3. Código del microcontrolador PIC16F628A

Código desarrollado en Basic empleando el compilador de PicBasicPro.

```
'****************************************************************
'*  Name    : RGB_Led_Controler.BAS                             *
'*  Author  : [Bryan Cañarte]                                   *
'*  Notice  : Copyright (c) 2020 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 19/12/2020                                        *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                   *
'****************************************************************
ONLed           VAR BIT[4]              ; Used to point when turn off a led on
effect mode
ColorLed        VAR BYTE[4]             ; Used to save data color of every led
BlinkLed        VAR BYTE[4]             ; Used to save blink mode of every led
X               VAR BYTE                ; Used to repetion
BufferOption    VAR BYTE                ; Used as buffer to get blink mode from
option_led
Type            VAR BYTE                ; Used to select a correct blink mode
on the subrutine
SelecLed        VAR BYTE                ; Used to select a current led on
mainprogram
Option_Led      VAR BYTE                ; Used to save blink mode of all leds
BufferData      VAR BYTE                ; Used to get a data from serial buffer
BufferCommand   VAR BYTE                ; Used to decode a command recived from
serial buffer
Command         VAR BYTE                ; Used to get a command recived from
serial buffer
Blink_1         VAR WORD                ; Used to count millis of first blink
efect led
Blink_2         VAR WORD                ; Used to count millis of second blink
efect led
Blink_3         VAR WORD                ; Used to count millis of third blink
efect leds
Delay           VAR WORD                ; Used to count a delay between every
led
size            CON 20                  ; Size of the serial buffer
bufferRC        VAR BYTE[size]          ; Buffer of the RC serial data
BSR             VAR BYTE                ; Pointer  register to save recive data
BSRRC           VAR BYTE                ; Pointer register to get a serial data
from buffer
NDT             VAR BYTE                ; Number of data to read
PulsadorBuffer VAR BYTE                 ; Used to save pulsador data
RSCON           VAR RCSTA.4             ; Enable serial Comunication
; ****************** Millis setup ************************************
PIE1.0 = 1                  ; Enable Timer 1 overflow interrupt
TMR1H = $FC                 ; MSB of the timer preset (FC18h = 64536)~ 1mS to
overflow
TMR1L = $18                 ; LSB of the timer preset (FC18h = 64536)~ 1mS to
overflow
T1CON = %00000001           ; Prescaler 1:1, external oscilator turn off,
internal clock source, Timer on
; ****************** Serial comunication setup ************************
TXSTA = %00010000       ; 8 bits data synchronous slave mode
RCSTA = %10000000       ; Serial port enable, disable recive data
PIE1.5 = 1              ; Enable reception interrupt
;******************** Setup ***************************************
TRISA  = 0
```

```
TRISB  = %10000111
OPTION_REG.6 = 0       ; Set falling edge external interrupt
CMCON  = 7
PORTB  = 0
PORTA  = 0
SelecLed = 0
Option_Led = %00000000      ;LD4.1-LD4.0,LD3.1-LD3.0,LD2.1-LD2.0,LD1.1-LD1.0
                            ;LDX.1-LDX.0
                            ; 0      0        No efect
                            ; 0      1        Blink 250 mS
                            ; 1      0        Blink 500 mS
                            ; 1      1        Blink 1000 mS
; Set color ***************
ColorLed[0]   = %111
ColorLed[1]   = %111
ColorLed[2]   = %111
ColorLed[3]   = %111
; Set option **************
GOSUB SetOption
; Reset all serial register ***************
BSR   = 0              ; Clear pointer buffer
BSRRC = 0              ; Clear RC pointer buffer
NDT   = 0              ; Clear all data to send
PAUSE 5                ; Delay before start
; ******************* Interrupts setup *********************************
ON INTERRUPT GOTO ISRF
INTCON = %11010000              ; Global interrups enable, peripherical
interrup
                                ; enable, clear all flags
;******************* Loop ****************************
SelecLed = 0
loop:
    ; Get serial data from port ********************
    IF NDT > 0 THEN
        GOSUB RC_Get_bytes
        Command = BufferData & $E0 ; Get command code *******
        ; If command is clear, then turn off leds and reset mode
        IF Command = $00 THEN
            FOR X = 0 TO 3
                ColorLed[X] = 0
            NEXT
            Option_Led = %00000000
            GOSUB SetOption
        ENDIF
        ; IF is color command, execute this code ************
        IF Command = $20 THEN
            ; Format byte set led color instruction **************
            ; CM2 : CM1 : CM0
            ;  0  :  0  :  1  : LP1 : LP0 : CB : CG : CR
            ; ************************************************
                            ; 0      0   Led 0
                            ; 0      1   Led 1
                            ; 1      0   Led 2
                            ; 1      1   Led 3
            ;(CM3-CM0) Command bits
            ; X don´t care
            ; CB = Color Blue      Set = 1, Clear = 0
```

```
        ; CG = Color Green     Set = 1, Clear = 0
        ; CR = Color Red       Set = 1, Clear = 0
        BufferCommand = BufferData & $18          ; Get led posicion
        BufferCommand = BufferCommand >> 3
        ColorLed[BufferCommand] = BufferData & $07 ; Put led color
    ENDIF
    ; If is a led effect of the first two leds, execute this code ******
    IF Command = $40 THEN
    ; Format byte set led effect instruction **************
        ; CM2 : CM1 : CM0
        ;  0  :  1  :  0  :  X : LD2.1 : LD2.0 : LD0.1 : LD0.0
        ; ************************************************
        ;(CM3-CM0) Command bits
        ; X don´t care
        BufferCommand = BufferData & $0F
        Option_Led = BufferCommand | (Option_Led & $F0)
        GOSUB SetOption
    ENDIF
    ; If is a led effect of the last two leds, execute this code ******
    IF Command = $60 THEN
    ; Format byte set led effect instruction **************
        ; CM2 : CM1 : CM0
        ;  0  :  1 :  1  :  X : LD3.1 : LD3.0 : LD2.1 : LD2.0
        ; ************************************************
        ;(CM3-CM0) Command bits
        ; X don´t care
        BufferCommand = BufferData & $0F
        @swapf _BufferCommand,f
        Option_Led = BufferCommand | (Option_Led & $0F)
        GOSUB SetOption
    ENDIF
ENDIF
;************************** Set actual Led **********************
GOSUB Current_Led
IF SelecLed = 3 THEN
   SelecLed = 0
ELSE
   SelecLed = SelecLed + 1
ENDIF
; handler of no blink effect *
GOSUB State_0_Led
; handler of blink 250 *******
IF  Blink_1 > 250 THEN
    Blink_1 = 0
    Type = 1
    GOSUB ChangeLed
ENDIF
; handler of blink 500 *******
IF  Blink_2 > 500 THEN
    Blink_2 = 0
    Type = 2
    GOSUB ChangeLed
ENDIF
; handler of blink 1000 *******
IF  Blink_3 > 1000 THEN
    Blink_3 = 0
    Type = 3
```

```
        GOSUB ChangeLed
    ENDIF
GOTO loop
;***** Set current Led ***********
Current_Led:
    PORTA = ColorLed[SelecLed]
    GOSUB Multiplexer
    GOSUB Wait_
    PORTB = 0
RETURN
;***** Change State Led **********
ChangeLed:
    FOR X = 0 TO 3
        IF BlinkLed[X] = Type THEN
            IF ONLed[X] = 1 THEN
              ONLed[X] = 0
            ELSE
              ONLed[X] = 1
            ENDIF
        ENDIF
    NEXT
RETURN

;***** Set no effect **********
State_0_Led:
    FOR X = 0 TO 3
        IF BlinkLed[X] = 0 THEN
              ONLed[X] = 1
        ENDIF
    NEXT
RETURN
; ***************** wait 3 mS ***************
Wait_:
    Delay = 0
    loopdelay:
        IF  Delay > 3 THEN RETURN
        GOTO loopdelay

; Subrutine of multiplexer ******
Multiplexer:
   IF ONLed[SelecLed] = 1 THEN
      PORTB = %00000100 << (SelecLed + 1)
   ELSE
      PORTB = 0
   ENDIF
RETURN
; ******************* Get option data led ******************************
SetOption:
    BufferOption = Option_Led
    FOR X = 0 TO 3
        ONLed[X] = 1
        BlinkLed[X] = BufferOption & $03
        BufferOption = BufferOption >> 2
    NEXT
RETURN
;********************** Get data from serial Port ********************
;Subrutine to get data from serial buffer
```

```
RC_Get_bytes:
    IF NDT = 0 THEN                          ; When there aren't any data,
return last recive data
       BufferData = bufferRC[BSRRC]             ; Buffer is last recive data
       RETURN                                ; Go back
    ENDIF
    BufferData = bufferRC[BSRRC]                ; Send data of the buffer
    IF BSRRC = (size-1) THEN
       BSRRC = 0                             ; If buffer is full, BSRTX is
reset
    ELSE
       BSRRC = BSRRC + 1                     ; Go to next data to send
    ENDIF
    NDT = NDT - 1                            ; Sub a data to send
RETURN
; ***************** Handler de interrupción *******************
DISABLE           ; All this code and subrutine are used on interrupts.
Disable interrupts from here
ISRF:
    IF PIR1.0   = 1 THEN GOSUB SetMillisTime    ; SetMillisTime
    IF PIR1.5   = 1 THEN GOSUB SerialDataRecive ; Subrutine to save serial
data from serial port
    IF INTCON.1 = 1 THEN GOSUB CSE_             ; Chip Select enable
    INTCON = %11010000                          ; Enable interrupts again
RESUME
;*********************** Handler of serial buffer *******************
;Used to get bytes to buffer synchronous
SerialDataRecive:
    NDT = NDT + 1                     ; Add a data to read
    bufferRC[BSR] = RCREG             ; Save data on buffer
    IF BSR = (size-1) THEN
       BSR = 0                        ; If buffer is full, BSR is reset. Next
data overwrite buffer
    ELSE
       BSR = BSR + 1                  ; Next adress of the data
    ENDIF
RETURN
;Used to start comunication. Chip Select bit ***********************
CSE_:
    IF PORTB.0 = 0 THEN
       RSCON = 1              ; Enable recive data
       OPTION_REG.6 = 1       ; Set raising edge external interrupt
    ELSE
       RSCON = 0              ; Disable recive data
       OPTION_REG.6 = 0       ; Set falling edge external interrupt
    ENDIF
    PAUSE 5
    INTCON.1 = 0              ; Clear flag
RETURN
;************** Count time system *********************************
SetMillisTime:
    PIR1.0   = 0                          ; Clear Timer 1 overflow flag
    Blink_1  = Blink_1 + 1                  ; Add 1mS to the counter
    Blink_2  = Blink_2 + 1                  ; Add 1mS to the counter
    Blink_3  = Blink_3 + 1                  ; Add 1mS to the counter
    Delay  = Delay + 1                      ; Add 1mS to the counter
    ; Preset timer again ***********
```

```
    TMR1L = $18                 ; LSB of the timer preset (FC18h = 64536)~ 1mS
to overflow
    TMR1H = $FC                 ; MSB of the timer preset (FC18h = 64536)~ 1mS
to overflow
RETURN
ENABLE  ; End of all subrutine of interrupts. Enable code from here
END
```

Bibliografía

[1] El Telégrafo. (07 de septiembre de 2019). La producción del campo mejora con la tecnología. Obtenido de https://www.eltelegrafo.com.ec/noticias/economia/4/produccion-tecnologia-ecuador-ministerio-agricultura?fbclid=IwAR3s-Tlpk2QHuChzmOAX3q_i4K7j_HFVsAnfEujXWAJw2BpcVzywpRoIE1g

[2] ASOBANCA. (21 de enero de 2020). Boletín Macroeconómico. Obtenido de http://www.asobanca.org.ec/publicaciones/boletin-macroeconomico?fbclid=IwAR3pTnVZlgh2nCIOc07k7zacfpflOhfCHMFqmhMx-3Q-ZqR5ZwBr7CwM-Ns

[3] INEC. (2018). Instituto Nacional de Estadística y Censos. Censo Nacional Agropecuario. Obtenido de https://www.ecuadorencifras.gob.ec//documentos/web-inec/Estadisticas_agropecuarias/CNA/Resul_Nac_resu_Prov_CNA.zip.

[4] Rubio, C. (2017). Automatización de un cultivo hidropónico NFT para el control detemperatura, riego y mezcla de la solución nutritiva, ubicada en la zona urbna de Quito. (tesis de pregrado). Universidad Politécnica Salesiana Sede Quito, Quito, Ecuador

[5] Castillo, L. (30 de abril de 2015). Los cultivos hidropónicos toman impulso en Cuenca. El Comercio. Obtenido de https://www.elcomercio.com/actualidad/cultivos-hidroponicos-toman-impulso-cuenca.html?fbclid=IwAR1c8C-KqSMdwoG0O__yG4h-GtA5z_V-W9C5vedW_6bvvchfJc3_ojFIQRM

[6] Texier, W. (2013). Hidroponía para Todos. Todo sobre la horticultura en casa. París: MAMA EDITIONS.

[7] Cacciagioni, L., Eizmendi, J., & Nieva, J. (20 de septiembre de 2019). aHydro: Hidroponía automatizada. Obtenido de http://sedici.unlp.edu.ar/bitstream/handle/10915/71678/Documento_completo.pdfPDFA.pdf?sequence=1&isAllowed=y&fbclid=IwAR1-I9_ 1R4rkWIT8TJtz1Azw34DKmX2PUUJ9wRqP5laxAW9RqiAPVbB8fOM

[8] CONVERGIA. (03 de febrero de 2020). ¿Cuáles son las ventajas de implementar sensores IoT en el sector agrícola? Obtenido de

https://www.convergia.io/es-mx/cuales-son-las-ventajas-de-implementar-sensores-iot-en-el-sector-agricola/?fbclid=IwAR3Dus6LOM5g2IvIW3vjnhSjpEpTqk5g0GX-lEc8nreGDTQu-vap0FBiRvw

[9] Herrera, J. (2016). Diseño de un sistema de control hidropónico para la granja experimental Yuyucocha e implementación de un módulo didáctico (tesis de pregrado). Universidad Técnica del Norte, Ibarra, Ecuador.

[10] Carlozama, G. (2018). SCADA para invernadero sobre software libre. (tesis de pregrado). Universidad Técnica del norte,Ibarra,Ecuador.

[11] Bedón, K.,& Tovar, A. (2016). Implementación de un sistema de control automático con monitoreo a través de la web para la producción de tomate riñón variedad daniela basado en la agricultura hidropónica y control de riego de agua por goteo en el invernadero localizado en el barrio San Gerardo de la ciudad de Latacunga (tesis de pregrado). Universidad de las Fuerzas Armadas ESPE, Latacunga, Ecuador.

[12] R. M. Salinas Arcos, "Diseño de un prototipo de sistema automatizado con Arduino para riego en el cultivo de fresas.,"6 mar 2019. [Online].

[13] Hinojasa, S. (2019). Diseño de una arquitectura IoT para el control de sistemas hidropónicos (tesis de pregrado). Universidad Politécnica de Valencia, Valencia, España.

[14] Gutierrez, E., Montiel, J., Arellano, C., & Menchaca, F. (26 de 07 de 2019). Propuesta de sistema de gestión inteligente basado en IoT para hidroponia. Obtenido de https://rcs.cic.ipn.mx/2019_148_10/Propuesta%20de%20sistema%20de%20gestion%20inteligente%20basado%20en%20IoT%20para%20hidroponia.pdf

[15] Aguirre, E., González, O., Vega, D., & Monje, J. (2019). Prototipo basado en Nutrient Film Technic para el monitoreo y control de un cultivo de hortaliza. Revista colombiana de investigaciones agroindustriales, 6(1), 29-40. doi:10.23850/24220582.1830

[16] Romero, N., & Yánez, V. (2016). Construcción y automazación de un prototipo de invernadero hidropónico (tesis de pregrado).Escuela politécnica Nacional, Quito, Ecuador

[17] Conde, E. (2017). Diseño de un prototipo para el control y automatización de un sistema hidropónico en un invernadero (tesis de grado). Universidad mayor de San Andres, La paz, Bolivia.

[18] Hydro Enviroment. (2020). ¿Qué es el sistem NFT? Obtenido de https://hydroenv.com.mx/catalogo/index.php?main_page=page&id=101&chapter =9sequence=1&isAllowed=y.

[19] GroHo. (19 de junio de 2017). Solución nutritiva. Obtenido de https://www.groho.es/post/solucion-nutritiva-hidroponia

[20] cultivoHIdroponico.info. (febrero de 2019). Cultivo Hidropónico de Lechuga – Guía Para Principiantes! Obtenido de https://cultivohidroponico.info/cultivo-hidroponico-de-lechuga/

[21] F. Martinez,"Obtención de hortalizas a pequeña y media escala con hidroponía,.Esperanza, Santa Fe, Argentina, 2020.

[22] Dallas Semiconductor,"DS18B20 Programmable Resolution 1-Wire Digital Thermometer,"2019. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[23] "DS18B20 Waterproof Temperature Sensor Cable,"[Online]. Available: https://www.terraelectronica.ru/pdf/show?pdf_file=%252Fz%252FDatasheet%252F1%252F1420644897.pdf.

[24] E. Morgan, "HC-SR04 Ultrasonic Sensor,"16 Noviembre 2014. [Online]. Available: https://html.alldatasheet.net/html-pdf/1132203/ETC2/HC-SR04/111/1/HC-SR04.html.

[25] Cytron Technologies, "User's Manual V1.0,"Mayo 2013. [Online]. Available: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit.

[26] Testo Argentina SA, "Los electrodos de pH,"2018. [Online]. Available: http://www.academiatesto.com.ar/cms/los-electrodos-de-ph#: :text=La%20medici%C3%B3n%20del%20pH%20es,hidronio%20en%20una%20tensi%C3%B3n%20el%C3%A9ctrica.&text=En%20el%20electrodo%20de%20medici%C3%B3n,referencia%20de%20potencial%20del%20sistema..

[27] BotShop, "How to use a PH probe and sensor,"[Online]. Available: https://www.botshop.co.za/how-to-use-a-ph-probe-and-sensor/.

[28] D. Ghia, Diseño y construcción de un conductivímetro digital para medición de conductividad en soluciones con conexión a PC, Quito, 2007.

[29] L. Acosta, CONVERTIDOR DE SEÑALES ECL PARALELO SERIAL PARA LA COMPATIBILIDAD DE INTERFACE DE LAS GRABADORAS AMPEX EN EL INSTITUTO ESPACIAL ECUATORIANO, ESTACIÓN COTOPAXI, Ambato, 2014.

[30] E. Lopez, Protocolo RS-232, 2003.

[31] Word Press, "UART y USB en Arduino,"9 11 2016. [Online]. Available: https://aprendiendoarduino.wordpress.com/category/puerto-serie/.

[32] M. Mynor, USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO, 2015.

[33] T. Omar, DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO PARA EL CONTROL Y REGISTRO DE TEMPERATURA A TRAVÉS DE LA COMUNICACIÓN ENTRE LOS SENSORES INTELIGENTES DS18B20 Y SU PROTOCOLO DE COMUNICACIÓN ONE-WIRE, ENTRE UNA TARJETA MAESTRA Y ESCLAVO, ENLAZADOS POR MÓDULOS INALÁ, Guayaquil, 2015.

[34] E. López, Protocolo SPI(Serial Peripherical Interface), 2003.

[35] ATMEL, ATmega 328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash, 2015.

[36] Microchip, PIC16F627A/628A/648A Data Sheet Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technolog, 2007.

[37] ARDUINO STORE, "ARDUINO NANO,"2021. [Online]. Available: https://store.arduino.cc/usa/arduino-nano. [Accessed 11 Abril 2021].

[38] MCIElectronics, "¿Que es Raspberry Pi?,"[Online]. Available: https://raspberrypi.cl/que-es-raspberry/. [Accessed 11 Abril 2021].

[39] Raspberrypi, "Raspberry Pi 3 Model B+,"[Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed 11 Abril 2021].

[40] V. Lima, CONTROLE EMBARCADO DE COLCHÃO DE, Párana, 2017.

[41] J. Gutiérrez, Máquinas de Estados Finitos, 2008.

[42] J. Salazar and S. Silvestre, Internet de las cosas, Erasmus, 2020.

[43] Firebase, "Firebase Realtime Database,"2021. [Online]. Available: https://firebase.google.com/docs/database?authuser=0. [Accessed 11 Abril 2021].

[44] JSON, "Introducing JSON,"[Online]. Available: https://www.json.org/json-en.html. [Accessed 11 Abril 2021].

[45] Firebase, "Explicación de las reglas de Firebase Realtime Database,"2021. [Online]. Available: https://firebase.google.com/docs/database/security?authuser=0. [Accessed 11 Abril 2021].

[46] ElDelCable, "RECONEXIÓN WIFI AUTOMÁTICA EN RASPBERRY PI CON AVISO,"12 2016. [Online]. Available: http://laraspberrypi.blogspot.com/2016/12/reconexion-wifi-automatica-en-raspberry-con-aviso.html.

[47] J. Barron and C. Ashton, The Effect of Temperature on Conductivity Measurement, 2005.

[48] C. Goméz, R. Gonzaléz and R. Viruela, CONDUCTIVIDAD DE LAS DISOLUCIONES ELECTROLITICAS, 2010.

[49] Banggod, “Analog TDS Sensor Conductivity Sensor Tester Monitorización de calidad del agua de detección de agua,”2020. [Online]. Available: https://www.banggood.com/Analog-TDS-Sensor-Water-Conductivity-Sensor-Tester-Liquid-Detection-Water-Quality-Monitoring-p-1601493.html?cur_warehouse=CN. [Accessed 11 Abril 2021].

[50] “High Precision PVC Water Flow Sensor YF-S401,”[Online]. Available: https://www.epitran.it/ebayDrive/datasheet/YF-S401.pdf. [Accessed 11 Abril 2021].