



UNIVERSIDAD TÉCNICA DEL NORTE

**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES**

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

**“DESARROLLO DE UN PROTOTIPO IOT PARA LA AUTOMATIZACIÓN
DE CANALES DE RIEGO EN EL BARRIO RÍO FRÍO – COTACACHI”**

Trabajo de titulación previo a la obtención del título de Ingeniera en Telecomunicaciones
--

Línea de investigación: Producción industrial y tecnología sostenible

AUTOR:

Ana Carolina Ramos Tambaco

DIRECTOR:

Msc. Carlos Alberto Vásquez Ayala

Ibarra, Ecuador 2026



UNIVERSIDAD TÉCNICA DEL NORTE BIBLIOTECA UNIVERSITARIA

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

1. IDENTIFICACIÓN DE LA OBRA

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

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1005201049		
APELLIDOS Y NOMBRES:	Ramos Tambaco Ana Carolina		
DIRECCIÓN:	Barrio Río Frío y Esmeraldas		
EMAIL:	acramost@utn.edu.ec / starcarolina_love@hotmail.com		
TELÉFONO FIJO:	062915454	TELÉFONO MÓVIL:	0968467194

DATOS DE LA OBRA	
TÍTULO:	DESARROLLO DE UN PROTOTIPO IOT PARA LA AUTOMATIZACIÓN DE CANALES DE RIEGO EN EL BARRIO RÍO FRÍO – COTACACHI.
AUTOR (ES):	RAMOS TAMBACO ANA CAROLINA
FECHA: DD/MM/AAAA	03/02/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	☒ PREGRADO ☐ POSGRADO
TÍTULO POR EL QUE OPTA:	INGENIERA EN TELECOMUNICACIONES
DIRECTOR:	MSC. CARLOS ALBERTO VÁSQUEZ AYALA
ASESOR:	MSC. LUIS EDILBERTO SUÁREZ ZAMBRANO

2. CONSTANCIAS

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

Ibarra, a los 03 días del mes de febrero del 2026.

EL AUTOR:



.....

Ana Carolina Ramos Tambaco

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 03 de febrero del 2026

MSC. CARLOS ALBERTO VÁSQUEZ AYALA

DIRECTOR DEL PRESENTE TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

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



(f)

Msc. Carlos Alberto Vásquez Ayala

C.C.: 1002424982

DEDICATORIA

Con mucho cariño y gratitud dedico este trabajo: A mi familia, que ha sido el pilar más importante en cada etapa de mi vida. A mis padres, Rosita Tambaco y Rodrigo Ramos, por ser ejemplo de esfuerzo, humildad y amor incondicional. Gracias por enseñarme que los sueños se alcanzan con trabajo constante, fe y valentía. Su apoyo ha sido la fuerza que me impulsó a no rendirme.

A mis hermanos, Geovanny, Samir y Mercedes, por estar siempre conmigo, por su cariño incondicional, sus consejos y por acompañarme en cada paso de este camino con paciencia, ánimo y comprensión.

A mis sobrinas y sobrino, Cadence, Zahir, Tiffany y Mailen, quienes con su inocencia y alegría llenaron mi corazón de esperanza y motivación siendo una inspiración para seguir superándome y construir un mejor futuro.

Y a toda mi familia quienes han sido parte fundamental en mi formación quienes, con su presencia, apoyo moral y cariño, han impulsado cumplir este logro. Cada uno de ustedes vive en estas páginas y en este sueño cumplido.

Ramos Tambaco Ana Carolina

AGRADECIMIENTO

Agradezco primeramente a Dios, por brindarme salud, fortaleza y sabiduría en cada paso de este camino académico. Sin su guía, nada de esto hubiera sido posible (Isaías 41:10).

A mis padres, Rosita y Rodrigo así mismo a mis hermanos por ser mi mayor inspiración, por su amor incondicional, por enseñarme a luchar con determinación y nunca rendirme, incluso en los momentos más difíciles.

A mis amigos de la carrera, Yuli, Jessica, Anahí, Lizbeth, Pablo y David, gracias por su compañía constante, por cada palabra de ánimo, por compartir tantas experiencias y por hacer de esta etapa algo inolvidable. A mis amigas del colegio “Las Dos Toas”, Tania, Evelin y Melanie, por seguir presentes en mi vida, por sus mensajes de apoyo, por creer en mí y por demostrar que la verdadera amistad perdura a lo largo del tiempo.

Al Msc. Carlos Vásquez, director de este trabajo de titulación, por su guía académica, su compromiso y por haberme orientado con responsabilidad y profesionalismo durante todo el proceso y Msc. Luis Suárez, por sus valiosas observaciones como asesor. Al Msc Jaime Michilena, coordinador de la carrera de Ingeniería en Telecomunicaciones, por su apoyo institucional, por facilitar el proceso académico y por impulsar el crecimiento de cada uno de los estudiantes.

Gracias a todos quienes, de una u otra forma, formaron parte de este logro.

Ramos Tambaco Ana Carolina

RESUMEN EJECUTIVO

Los canales de riego del país presentan deficiencias en su gestión y control operativo, derivadas de la ausencia de sistemas tecnológicos que permitan monitorear y regular de forma eficiente la distribución de los recursos de agua. En este contexto, el trabajo de titulación tiene como objetivo desarrollar un sistema basado en Internet de las Cosas (IoT) orientado a la automatización de canales de riego, mediante un prototipo que integra sensores de nivel de agua, actuadores y microcontroladores, considerando la adaptabilidad de los cultivos, las condiciones del suelo y la disponibilidad de riego, bajo la aplicación de la metodología en cascada. Para el desarrollo del prototipo se emplearon placas Arduino Uno como nodos esclavos y una placa ESP32 como nodo maestro, encargada del procesamiento y la comunicación inalámbrica; de este modo, se posibilita la adquisición y transmisión de datos mediante el protocolo MQTT hacia una plataforma de monitoreo, complementada con una interfaz web para la visualización de información y un sistema de notificaciones mediante Telegram. Asimismo, se validó el funcionamiento de los sensores, el control automatizado de las compuertas y la estabilidad operativa del sistema, evidenciando su viabilidad técnica, la reducción de desperdicios innecesarios de agua y su proyección futura.

Palabras clave: Internet de las Cosas (IoT), Automatización de riego, MQTT, Canales de riego, Sensores de nivel de agua, Control automatizado.

ABSTRACT

Irrigation channels in the country present deficiencies in their management and operational control, derived from the lack of technological systems that allow efficient monitoring and regulation of water resources distribution. In this context, this degree project aims to develop a system based on the Internet of Things (IoT) oriented to the automation of irrigation channels, through a prototype that integrates water level sensors, actuators, and microcontrollers, considering crop adaptability, soil conditions, and water availability, under the application of the waterfall methodology. For the development of the prototype, Arduino Uno boards were used as slave nodes and an ESP32 board as a master node, responsible for processing and wireless communication; in this way, data acquisition and transmission are enabled through the MQTT protocol to a monitoring platform, complemented by a web interface for data visualization and a notification system using Telegram. Likewise, the operation of the sensors, the automated control of the gates, and the operational stability of the system were validated, demonstrating its technical feasibility, the reduction of unnecessary water waste, and its future projection.

Keywords: Internet of Things (IoT), Irrigation automation, MQTT, Irrigation channels, Water level sensors, Automated control.

LISTA DE SIGLAS

IoT. Internet of Things (Internet de las Cosas)

MQTT. Message Queuing Telemetry Transport

ESP32. Espressif 32-bit Microcontroller

HTTP. Hypertext Transfer Protocol

URL. Uniform Resource Locator

API. Application Programming Interface (Interfaz de Programación de Aplicaciones)

CPU. Central Processing Unit (Unidad Central de Procesamiento)

RAM. Random Access Memory (Memoria de Acceso Aleatorio)

LAN. Local Area Network (Red de Área Local)

WiFi. Wireless Fidelity

IDE. Integrated Development Environment (Entorno de Desarrollo Integrado)

GPIO. General Purpose Input/Output

PWM. Pulse Width Modulation (Modulación por Ancho de Pulso)

ADC. Analog to Digital Converter (Convertidor Analógico–Digital)

AC. Alternating Current (Corriente Alterna)

DC. Direct Current (Corriente Continua)

USB. Universal Serial Bus

GUI: Graphical User Interface (Interfaz Gráfica de Usuario)

ÍNDICE DE CONTENIDOS

1	Capítulo I: Antecedentes	1
1.1	Tema.....	1
1.2	Problema.....	1
1.3	Objetivos	3
1.3.1	Objetivo General.....	3
1.3.2	Objetivos Específicos.....	3
1.4	Alcance.....	4
1.5	Justificación.....	7
2	Capítulo II: Estado del Arte.....	9
2.1	Sistema de canales de riego.....	9
2.1.1	Importancia del canal de riego en las comunidades.....	10
2.1.2	Distribución de canales de riego en las comunidades.....	10
2.1.3	Condiciones climáticas de riego	11
2.2	Factores que afectan el flujo de agua en los canales	12
2.2.1	Impacto de la temperatura en la evaporación	13
2.2.2	Efectos de la precipitación en el flujo de los canales.....	14
2.2.3	Variabilidad estacional y su gestión	15
2.3	Nivel de agua en canales de riego	16
2.3.1	Sedimento que afecta el paso del agua a los canales de riego	16
2.3.2	Métodos de medición del nivel de agua.....	17

2.4	Estándares y tecnologías aplicadas en canales de riego	19
2.4.1	Arquitectura de sistema IoT para canal de riego	19
2.4.2	Protocolos IoT aplicados al monitoreo y control de riego	20
2.5	Sensores y actuadores en riego automatizado	21
2.5.1	Sensor de temperatura DHT22.....	22
2.5.2	Sensor de humedad SHT10.....	23
2.5.3	Sensor de nivel de agua.....	24
2.6	Placas de desarrollo y microcontroladores	25
2.6.1	Microcontroladores	26
2.6.2	Placa ESP-32.....	26
2.6.3	Comparativa de placas de desarrollo	27
2.6.4	Características de las placas de desarrollo	27
2.7	Base de datos y almacenamiento de información	28
2.7.1	Gestión de datos climáticos mediante metodología en cascada.....	28
2.7.2	Almacenamiento de datos climáticos.....	29
2.8	Metodología de desarrollo.....	30
3	Capítulo III: Diseño del sistema de monitoreo y control de riego.....	33
3.1	Metodología	33
3.2	Etapas de Análisis.....	34
3.2.1	Situación Actual.....	34
3.2.2	Ubicación de la zona	36

3.2.3	Delimitación geográfica.....	37
3.2.4	Propósito del sistema	38
3.3	Etapa de Requerimientos.....	39
3.3.1	Requerimientos	39
3.3.2	Beneficiarios	40
3.3.3	Nomenclatura de los requerimientos	41
3.3.4	Requerimientos de Stakeholders.....	42
3.3.5	Requerimientos del Sistema.....	44
3.3.6	Requerimientos de Arquitectura	47
3.3.7	Selección de hardware y software.....	50
3.3.8	Software	56
3.4	Diseño.....	58
3.4.1	Diagrama de bloque del sistema	58
3.4.2	Arquitectura del sistema	61
3.4.3	Diagrama de adquisición de datos	62
3.4.4	Diagrama de flujo de procesamiento y activación de actuadores	64
3.4.5	Fuerza hidrostática	64
3.4.6	Lectura del nivel de Agua	69
3.4.7	Comunicación y recolección de datos.....	75
3.5	Implementación.....	85
3.5.1	Interfaz Web para la visualización de datos	87

4	Capítulo IV: Análisis de resultados	89
4.1	Diseño de la maqueta y ubicación de los sensores	89
4.1.1	Compuerta	90
4.1.2	Servos	91
4.1.3	Distribución y Armado	91
4.2	Pruebas	92
4.2.1	Validar requerimientos de Arquitectura	92
4.2.2	Test de hardware	94
4.2.3	Test de Software	97
4.2.4	Test de diseño	101
4.2.5	Test lógico	103
4.3	Validar requerimientos de Sistema	105
4.3.1	Validar requerimientos de Stakeholdres	107
4.4	Pruebas de integración del sistema	110
4.4.1	Análisis del caudal disponible y superficie de riego alcanzable	112
4.4.2	Control de precisión diario del canal de riego	113
	Conclusiones	115
	Recomendaciones	117
	Bibliografía	118
	ANEXOS	123
	Anexo 1 (Encuesta a los moradores del barrio)	123

Anexo 2 (Resultados de la encuesta realizada).....	125
Anexo 3 (Código del ESP32 como nodo maestro para comunicación serial con Arduinos y publicación MQTT)	132
Anexo 4 (Código fuente del nodo esclavo Arduino Uno para lectura de nivel, control de compuerta y comunicación).....	135
Anexo 5 (Dashboard de Control de Riego).....	137
Anexo 6 (Levantamiento de servicios del sistema en la nube cloud del bot, broker MQTT y servidor web mediante Docker Compose).....	142
Anexo 7 (Código fuente del prototipo de automatización de canales de riego)	144
Anexo 8. (Código fuente del módulo de monitoreo de nivel de agua y notificación vía Telegram en ESP32)	147

ÍNDICE DE TABLAS

Tabla 1	<i>Características y especificaciones del sensor DHT22</i>	<i>23</i>
Tabla 2	<i>Especificaciones técnicas del sensor de nivel de agua.....</i>	<i>24</i>
Tabla 3	<i>Lista de Stakeholders</i>	<i>40</i>
Tabla 4	<i>Definición de acrónimos.....</i>	<i>41</i>
Tabla 5	<i>Prioridad de los requerimientos del prototipo del canal de riego</i>	<i>42</i>
Tabla 6	<i>Requerimientos de Stakeholders</i>	<i>43</i>
Tabla 7	<i>Requerimientos iniciales del sistema</i>	<i>44</i>
Tabla 8	<i>Requerimientos de Arquitectura.....</i>	<i>47</i>
Tabla 9	<i>Comparación de diferentes placas embebidas</i>	<i>51</i>
Tabla 10	<i>Especificaciones técnicas de ESP32 y placa Arduino uno</i>	<i>52</i>
Tabla 11	<i>Consumo energético del prototipo</i>	<i>54</i>
Tabla 12	<i>Elección del software de programación</i>	<i>57</i>
Tabla 13	<i>Parámetros utilizados para el cálculo.....</i>	<i>65</i>
Tabla 14	<i>Validación del sistema eléctrico.....</i>	<i>92</i>
Tabla 15	<i>Validación de hardware</i>	<i>95</i>
Tabla 16	<i>Demostración de la prueba de software</i>	<i>97</i>
Tabla 17	<i>Validación del requerimiento de diseño</i>	<i>101</i>
Tabla 18	<i>Validación del sistema lógico.....</i>	<i>103</i>
Tabla 19	<i>Test del sistema operacional</i>	<i>107</i>
Tabla 20	<i>Validación de los requerimientos de uso</i>	<i>108</i>
Tabla 21	<i>Comparación de criterios entre ambiente controlado y canal real</i>	<i>111</i>
Tabla 22	<i>Relación entre caudal disponible y superficie de riego alcanzable.....</i>	<i>113</i>
Tabla 23	<i>Control diario de precisión del canal de riego mediante el sistema automatizado.....</i>	<i>114</i>

ÍNDICE DE FIGURAS

Figura 1 <i>Árbol de problemas</i>	2
Figura 2 <i>Formas farmacéuticas según su estado físico</i>	6
Figura 3 <i>Parámetros de la humedad en el suelo</i>	11
Figura 4 <i>Muestras de la evaporación en los días de invierno y verano</i>	14
Figura 5 <i>Salida en el hidrograma de la cuenca</i>	15
Figura 6 <i>Sedimentación en un canal de riego</i>	17
Figura 7 <i>Sistema de almacenamiento y recuperación de agua subterránea</i>	18
Figura 8 <i>Arquitectura de red IoT para monitoreo y gestión de datos climáticos</i>	20
Figura 9 <i>Protocolo MQTT</i>	21
Figura 10 <i>Diagrama de bloque del sistema de control</i>	22
Figura 11 <i>Sensor SHT10 de temperatura</i>	24
Figura 12 <i>Sensor de nivel de agua</i>	25
Figura 13 <i>Sistema de monitoreo y almacenamiento de datos climáticos</i>	30
Figura 14 <i>Metodología en cascada</i>	31
Figura 15 <i>Fases de la metodología en cascada</i>	33
Figura 16 <i>Referencia del terreno de aplicación</i>	35
Figura 17 <i>Ubicación del barrio Rio Frio en Cotacachi</i>	36
Figura 18 <i>Delimitación geográfica con coordenadas de latitud y longitud</i>	38
Figura 19 <i>Diagrama de bloques del sistema</i>	59
Figura 20 <i>Arquitectura general del sistema</i>	62
Figura 21 <i>Diagrama de flujo de adquisición de datos</i>	63
Figura 22 <i>Diagrama de flujo de procesamiento y activación</i>	64
Figura 23 <i>Diagrama de conexión sensor de nivel</i>	70

Figura 24 Código de lectura de sensores.....	71
Figura 25 Conexión y control básico de un servomotor con Arduino Uno	72
Figura 26 Diagrama de flujo del procesamiento local en nodos esclavos	74
Figura 27 Comunicación I2C entre ESP32 y múltiples Arduino uno	75
Figura 28 Código base de nodo esclavo para comunicación I2C con Arduino	77
Figura 29 Lectura y decodificación de datos I2C en ESP32 utilizando MicroPython	78
Figura 30 Secuencia integrada de monitoreo y control del sistema.....	79
Figura 31 Búsqueda e ingreso al bot oficial BotFather en Telegram	80
Figura 32 Comandos disponibles en BotFather para gestión de bots.....	81
Figura 33 Token de autenticación del bot de Telegram para integración con MicroPython.....	82
Figura 34 Envío de notificaciones desde ESP32 a Telegram mediante MicroPython	83
Figura 35 Mensajes del sistema de riego enviados por el bot de Telegram.....	84
Figura 36 Diagrama de integración del sistema	85
Figura 37 Configuración del servidor cloud y reglas de seguridad del sistema IoT.....	87
Figura 38 Interfaz web del sistema de riego y fragmento del código en HTML/CSS.....	88
Figura 39 Diseño físico de la maqueta y distribución de sensores.....	89
Figura 40 Compuertas colocadas en el canal de riego	90
Figura 41 Servomotores rotativos para las compuertas.....	91
Figura 42 Validación de la arquitectura física del sistema IoT	94
Figura 43 Representación de nodo esclavo en Arduino, lectura de sensores y control de servomotores.	96
Figura 44 Validación del sistema de comunicación MQTT y notificaciones por Telegram...	99
Figura 45 Acceso al sistema de monitoreo mediante la plataforma web	100
Figura 46 Acceso al sistema mediante Telegram a través de código QR o usuario configurado.....	100

<i>Figura 47</i>	<i>Estados de operación de la compuerta del canal de riego</i>	<i>105</i>
<i>Figura 48</i>	<i>Visualización de la interfaz del sistema y notificaciones en la aplicación móvil</i>	<i>106</i>
<i>Figura 49</i>	<i>Colaboradores del barrio.....</i>	<i>109</i>

1 Capítulo I: Antecedentes

1.1 Tema

DESARROLLO DE UN PROTOTIPO IOT PARA LA AUTOMATIZACIÓN DE CANALES DE RIEGO EN EL BARRIO RÍO FRÍO - COTACACHI

1.2 Problema

Ecuador es un país donde las exportaciones primarias son fundamentales para su economía con los productos agrícolas contribuyendo significativamente al Producto Interno Bruto (PIB) con un 7.81%, lo que representa ingresos de 8410.8 millones de dólares según estudios financieros de (Carrión, 2021). Es importante destacar que la producción agrícola bajo riego es responsable de casi el 70% de la producción nacional. No obstante, a pesar de su relevancia, varios desafíos como los cambios climáticos, la falta de asistencia técnica y la limitada presencia de la tecnología en los sistemas de riego han llevado a un manejo ineficiente del agua, afectando negativamente al sector campesino.

Una de las prioridades que se ha generado en la producción agrícola ha sido utilizar menos recursos, siendo esencial gestionar adecuadamente la cantidad de agua que requieren los cultivos. Sin embargo, se debe tomar en cuenta que puede generar varias anomalías tanto la ausencia como el exceso de agua debido a que pueden afectar a los cultivos, ya que ambos extremos alteran significativamente el crecimiento y la productividad de estas. En este caso la falta del recurso puede llevar a la deshidratación, marchitez y pérdida de los cultivos, impidiendo la absorción de nutrientes esenciales del suelo. Por otro lado el exceso de agua, puede asfixiar las raíces, erosionar el suelo y provocar la propagación de enfermedades y plagas, resultando en la pérdida parcial o total de las cosechas (Primicias, 2024).

La distribución irregular de agua entre agricultores de haciendas y comunidades específicamente en el barrio Río Frío es debido a la falta de control de los canales de riego cuyo manejo es de forma manual (SANREM, 2022). Además, la acumulación de sedimentos y basura en los canales de riego conduce a la obstrucción, aumentando el riesgo de desbordamiento y desperdicio de agua hacia las calles cercanas generando un desperdicio innecesario.

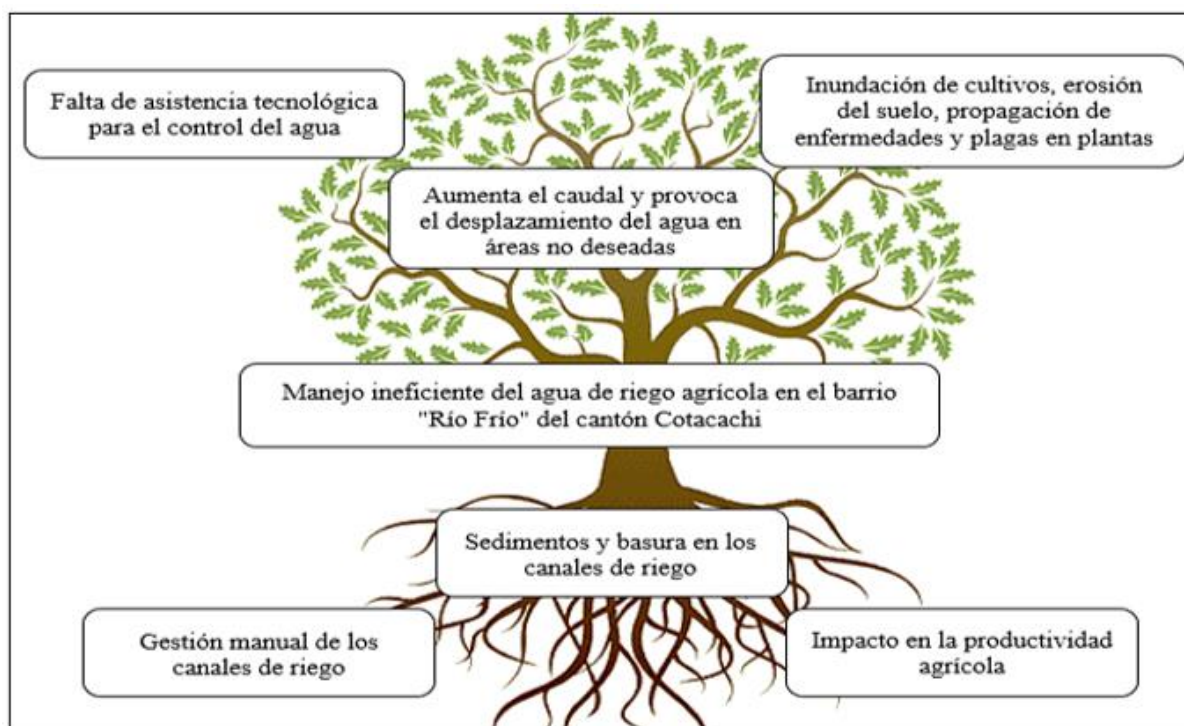
La constante búsqueda de alternativas para un aprovechamiento eficiente de los recursos ha generado la necesidad de aplicar las nuevas tecnologías en el sector rural y que por lo general conlleva a distintos desafíos tanto en el área de ingeniería como las áreas relacionadas con la agricultura y la ganadería (Blacio, 2023).

Esta problemática se puede evidenciar en la Figura 1, donde muestran las causas principales y efectos derivados del manejo ineficiente del agua de riego en el barrio Río Frío.

Por cual, el desarrollo de un prototipo IoT para la automatización de canales de riego permitirá optimizar el uso adecuado del agua en los cultivos, garantizando la gestión y distribución mediante el uso de sensores y actuadores para el control del paso o rechazo del mismo, de la misma manera existirá un aumento de la producción de sembríos, mejora de la gestión del tiempo, ya que elimina la necesidad de supervisión constante considerando que el agua es un recurso limitado.

Figura 1

Árbol de problemas



Nota. La figura describe el árbol de problemas del tema central, abordando las causas y efectos de este.

Fuente: Elaborado por el autor.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema IoT para la automatización de canales de riego en el barrio Río Frío – Cotacachi, mediante un prototipo que integre sensores y actuadores que controle la distribución del agua en cada uno de los sectores de la localidad.

1.3.2 Objetivos Específicos

- Conceptuar la parte teórica del tema propuesto mediante la consulta bibliográfica y el estado del arte que establezcan los criterios para el desarrollo del prototipo basado en la metodología de diseño de proyectos más idónea del caso.

- Establecer los requerimientos de acuerdo con la situación actual del sistema de canales de riego para la selección de los elementos electrónicos necesarios para la detección, corrección de problemas y optimización en los canales de riego.
- Diseñar el sistema integral de riego basado en tecnologías avanzadas que proporcione soluciones sostenibles y eficientes para la comunidad con sus respectivas reglas de uso.
- Realizar las pruebas de funcionamiento a través de la integración de los componentes con el sistema, obteniendo resultados mediante el monitoreo en los canales de riego.

1.4 Alcance

El presente proyecto se centra en el desarrollo de un prototipo IoT para la automatización de canales de riego en el barrio Río Frío del Cantón Cotacachi, provincia de Imbabura. Este sistema tiene como objetivo facilitar el trabajo en el sector agrícola y optimizar el consumo de agua para los cultivos. El proyecto se centra en un entorno controlado, limitándose al diseño y prototipado del sistema, excluyendo la implementación en condiciones reales y análisis complejos que exceden el alcance definido.

La base del desarrollo del presente proyecto seguirá la metodología del Modelo en Cascada, que consiste en un proceso cíclico en el que cada etapa requiere pruebas y ajustes incrementales para asegurar una culminación exitosa (Eby, 2019).

El desarrollo del proyecto consta de 4 etapas que serán cumplidas en base a los objetivos propuestos.

La primera etapa consiste en recopilar la información necesaria mediante la bibliografía referente a casos de estudio y estado del arte enfocados principalmente en el área de la agricultura y los sistemas de riego, tomando en cuenta la adaptabilidad de los cultivos al medio, las condiciones del suelo y la disponibilidad del riego; la segunda área es la investigación

tecnológica que se centrará en los componentes y sistemas adecuados para evitar posibles obstáculos durante el desarrollo.

La segunda etapa reúne la información adquirida en la primera etapa y se la desarrolla de acuerdo con la situación actual del área donde se desarrollará el proyecto según los requerimientos necesarios para la selección de los materiales. Se realizará un breve análisis de la factibilidad del proyecto considerando las características del área donde se desarrollará el prototipado, verificando la disponibilidad de recursos como el acceso a energía eléctrica para llevar a cabo las pruebas y evaluaciones necesarias. También se recopilará información de la comunidad local mediante encuestas para obtener datos relevantes sobre el entorno y las condiciones actuales.

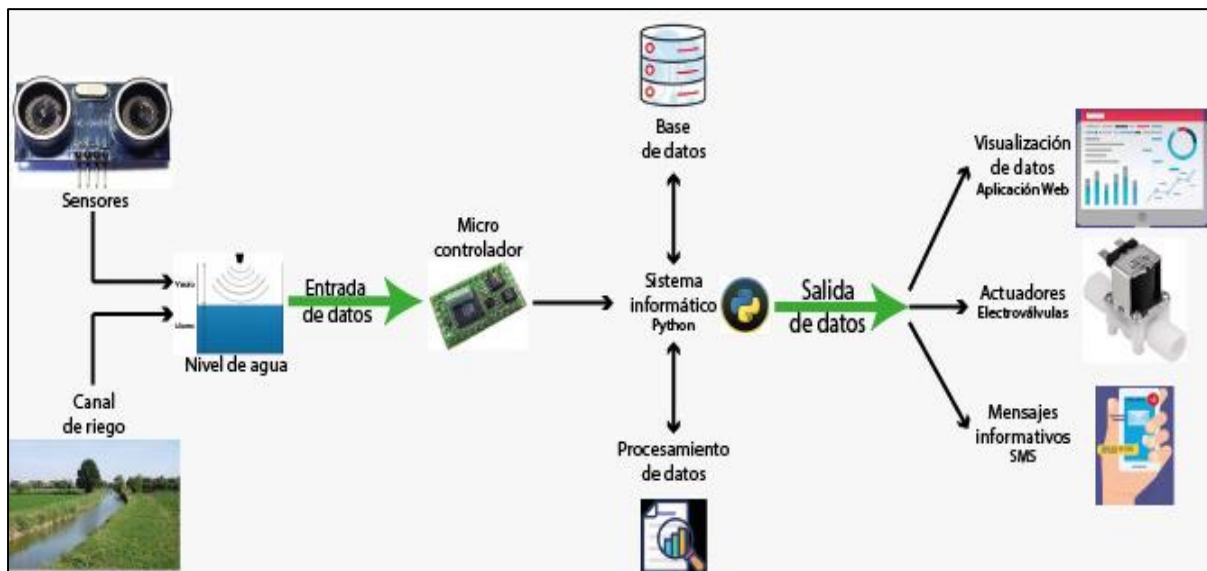
Para el desarrollo de la tercera etapa se requiere construir todo el sistema, tanto la parte del hardware como del software para comenzar con las pruebas de funcionamiento y detectar a tiempo detalles que no se tomaron en cuenta y que serán fáciles de corregir ese momento. Para el correcto funcionamiento de todo el sistema se tomaron en cuenta cuatro procesos: 1) La adquisición de datos mediante sensores colocados a lo largo de los canales de riego, donde se evaluarán las condiciones del medio. 2) La recepción de los datos adquiridos por los sensores conectados a un microcontrolador. 3) El manejo de los datos enviados desde el microcontrolador que serán almacenados y configurados por medio de un sistema desarrollado. 4) El accionar de los diferentes actuadores que se requieran según las órdenes establecidas por el sistema de acuerdo con los parámetros que no se alineen con los requerimientos iniciales.

El proceso descrito puede observarse representado en la Figura 2, donde se muestra el funcionamiento del sistema mediante sensores, actuadores, microcontrolador y su integración con una aplicación.

El prototipo será diseñado y construido con componentes que soporten las condiciones ambientales de la zona, garantizando su eficacia y durabilidad a mediano plazo. Durante un periodo de tiempo, se realizarán pruebas de funcionamiento del sistema en un entorno controlado, integrando los componentes de hardware y software para monitorear y optimizar el flujo de agua en los canales de riego. El sistema proporcionará mediciones confiables y en tiempo real para garantizar una gestión eficiente del agua, mejorando la distribución y el rendimiento de los cultivos.

Figura 2

Formas farmacéuticas según su estado físico



Nota. Funcionamiento del sistema de riego utilizando sensores, actuadores, microcontrolador, sistema y app.

Fuente: Elaborado por el autor.

La cuarta y última etapa del proyecto se centrará en el diseño de un sistema integral de riego que utilice tecnologías avanzadas para garantizar un uso eficiente y sostenible del agua en el entorno agrícola. Esta etapa incluirá la definición de reglas de uso que permitan optimizar el tiempo y el esfuerzo de los usuarios, además de fomentar prácticas sostenibles en la gestión de los recursos. También se evaluarán las características técnicas del hardware, como sensores, actuadores y microcontroladores, asegurando su viabilidad y funcionalidad para futuras aplicaciones.

Durante el diseño y desarrollo, el prototipo será probado únicamente en un entorno controlado, lo que permitirá validar su diseño y funcionamiento sin excederse a análisis más amplios o implementación en la comunidad. Este enfoque asegura que el sistema controlado cumpla con sus objetivos técnicos y proporcione una base sólida para futuros trabajos que busquen implementar el sistema en condiciones reales.

1.5 Justificación

El desarrollo de un sistema IoT para la automatización de canales de riego en el barrio Río Frío – Cotacachi busca abordar el problema del uso ineficiente del agua en dicho sector, causado por prácticas manuales y artesanales. Este proyecto se fundamenta principalmente en optimizar el manejo del agua en los cultivos mediante tecnologías avanzadas, como sensores y actuadores, aplicadas en un prototipo probado en un entorno controlado.

Actualmente, la falta de integración tecnológica en el sector agrícola dificulta la recopilación, almacenamiento y análisis eficiente de datos relacionados con el riego, lo que limita la productividad y sostenibilidad de los cultivos (Tabares, 2020). Este prototipo permitirá validar la funcionalidad de un sistema automatizado que optimice el monitoreo y control del riego, contribuyendo a un uso más eficiente del agua.

La propuesta se alinea con los Objetivos de Desarrollo Sostenible (ODS), específicamente con el objetivo Nro. 6, que promueve el acceso al agua potable, saneamiento y su gestión responsable (Naciones Unidas, 2018). A través de pruebas en un entorno controlado, el prototipo demostrará cómo las válvulas de paso pueden activarse automáticamente según las condiciones detectadas, optimizando el uso del agua en los canales de riego.

Este enfoque permitirá evaluar la eficiencia del sistema para reducir el desperdicio, afectaciones en la productividad y mejorar la distribución del agua. Además, las pruebas proporcionarán datos en tiempos real sobre el desempeño de los sensores, actuadores y microcontroladores, validando la viabilidad técnica del diseño y su capacidad para adaptarse a diferentes condiciones ambientales en sistemas controlados. (Garzón & Torres, 2018).

2 Capítulo II: Estado del Arte

Este capítulo se enfoca en proporcionar una revisión detallada de los principales conceptos teóricos que sirven como base para el análisis y comprensión del tema de investigación. Basándose principalmente en los sensores, microcontroladores y actuadores necesarios para medir el flujo de agua del canal de riego. Los datos recopilados serán almacenados en una base de datos que permita obtener información sobre flujo escaso de agua o en exceso, lo cual será informado mediante mensaje a los interesados.

2.1 Sistema de canales de riego

El sistema de conducción y distribución o red de canales, según (García, 2016) es uno de los componentes esenciales de una zona de riego. Esta red está diseñada y construida para transportar el agua desde su fuente hasta los puntos de consumo, asegurando que se entregue en la cantidad y el momento necesarios para satisfacer las demandas de los cultivos agrícolas, la industria, los asentamientos humanos, la fauna silvestre o acuícola, los sistemas recreativos y otros servicios que requieren este recurso dentro del área de influencia de la zona de riego. Los sistemas de canales se clasifican en tres categorías principales:

- **Sistema de entrega:** El agua se transporta desde una fuente de suministro hasta una serie de puntos de tomas o demanda, que suelen utilizarse en proyectos de riego, industrias y zonas urbanas.
- **Sistema de recolección:** En este aspecto se debe tomar en cuenta el agua recogida de diversas fuentes de suministro, como el desbordamiento de pozos en los canales de riego, tuberías de drenaje que llevan el exceso de agua de lluvia o de riego a los puntos de suministro.
- **Sistema de conexión:** Consiste en transportar el agua de un beneficiario a otro sin traspasar ni recibir demandas o aportes intermedios. Estos sistemas están conectados

por toma de decisiones posterior para la activación de los actuadores, evitando zonas muertas entre presas y canales de riego.

2.1.1 Importancia del canal de riego en las comunidades

Según una investigación realizada por (Munguiá, pág. 1), la agricultura es un pilar fundamental para la producción de productos primarios como el maíz, papa, arroz, fréjol, trigo, arveja, entre otros. Esta actividad depende en gran medida de los sistemas de riego, especialmente ante las dificultades impuestas por los cambios climáticos actuales.

Este problema ocasiona en zonas secas, donde las lluvias son anules e insuficientes para mantener una agricultura de temporada. Por ende, la construcción de un sistema automatizado permite el mantenimiento de canales de riego es crucial para el desarrollo del sector agrícola ya que permiten un control eficiente y notificaciones en tiempo real del mismo, lo que es fundamental para el crecimiento del sector primario y el desarrollo económico del país, dado que la agricultura representa una parte significativa de la economía nacional.

2.1.2 Distribución de canales de riego en las comunidades

Un estudio de la Universidad de Auburn revela que todas las fuentes de agua en Cotacachi están contaminadas con E-Coli y otras bacterias. Los tres principales ríos de la zona andina de Cotacachi, Pichaví, Pitzambiche y Yanayacu, reciben aguas residuales sin tratar de los centros poblados. Además de la contaminación por bacterias fecales, se suman el uso de pesticidas en la agricultura, el incremento de la floricultura y el desarrollo de actividades industriales (textiles, cueros, metalmecánica), que introducen tóxicos en los cuerpos de agua de la región. La mala calidad del agua y los problemas sanitarios son responsables del 80% de todas las enfermedades en los países en desarrollo (Fernández, 2013).

2.1.3 Condiciones climáticas de riego

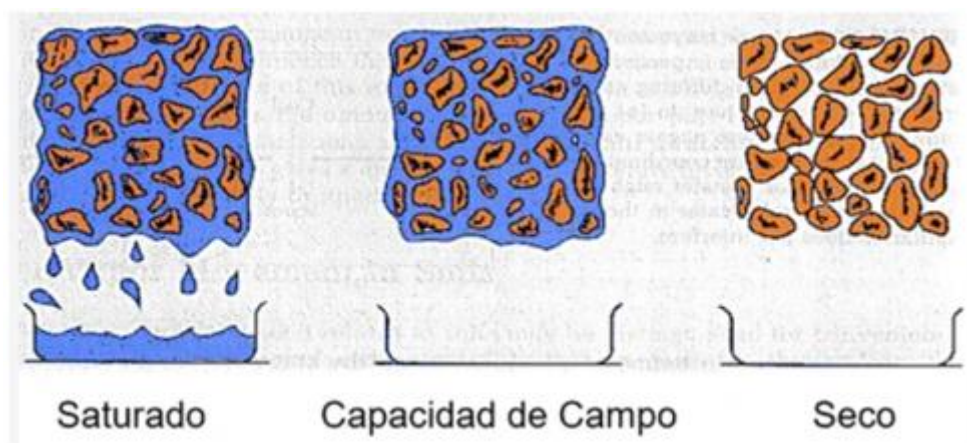
El agua, en su estado líquido, es esencial para la biosfera, ya que constituye la mayor parte de la materia viva y es el medio donde ocurren la mayoría de las reacciones biológicas. Sus propiedades son esenciales con las características fisicoquímicas ya que permiten un monitoreo crucial en sistemas físicos y biológicos de la Tierra.

La cantidad de agua en los tejidos vegetales varía en función de las condiciones ambientales, principalmente del nivel de humedad en el suelo. Una disminución del agua en el suelo donde crecen las plantas puede detener su crecimiento y provocar alteraciones metabólicas y estructurales significativas (Proain, 2020).

La Figura 3 nos da a conocer los parámetros de humedad que existen en el suelo dependiendo de cuanta agua se vaya almacenando para que las plantas 'puedan absorberlo. Estas variaciones son esenciales para la gestión de los canales en cada una de las parcelas, asegurando que el recurso de agua se ajuste a las necesidades específicas de las plantas en función de las condiciones climáticas.

Figura 3

Parámetros de la humedad en el suelo



Nota. Representación de los niveles de humedad en el suelo.

Fuente: Adaptado de (Proain, 2020).

2.2 Factores que afectan el flujo de agua en los canales

El flujo de agua en los canales de riego se ve afectado por diversos factores, entre ellos los sedimentos, las propiedades de la superficie y las condiciones ambientales. En este caso se debe tomar en cuenta estos elementos para la optimización el exceso de recurso por los canales de riego, tanto a gran escala como a pequeña escala. A continuación, se describen los aspectos clave que influyen en el flujo de agua en estos sistemas.

Características físicas

- Geometría del canal: Se debe tomar en cuenta la forma y el tamaño del canal tienen un impacto considerable en la resistencia al flujo con aspectos como los nanotubos circulares presentan coeficientes de fricción más bajos en comparación con otras geometrías (Rey, 2020).
- Vegetación: La disposición y presencia de vegetación en canales abiertos pueden modificar la velocidad del flujo y generar turbulencias. Los patrones de densidad y distribución de la vegetación contribuyen a la formación de estructuras de estela que afectan las características del flujo aguas abajo (Zhang, 2022).

Propiedades de la superficie

- Tensión superficial y fricción: Se puede determinar que, en los canales pequeños, la tensión superficial del agua y la rugosidad del canal son elementos dominantes en el flujo. En este caso la superficie absorbente incrementa la resistencia en las plantas, mientras que las más lisas permiten un flujo más eficiente (Fan, 2023).

- **Revestimientos:** La utilización de materiales como el grafeno puede optimizar el flujo de agua al reducir la fricción interfacial, lo que se traduce en mejoras significativas en los caudales en comparación con otros recubrimientos (Becerra, 2023).

Condiciones ambientales

- **Condiciones de flujo:** Variables como el caudal aguas arriba y la inclinación del canal afectan la dinámica del flujo. Un mayor caudal aguas arriba puede disminuir tanto la velocidad como el impulso en los canales ramificados (Mizhir, 2024).

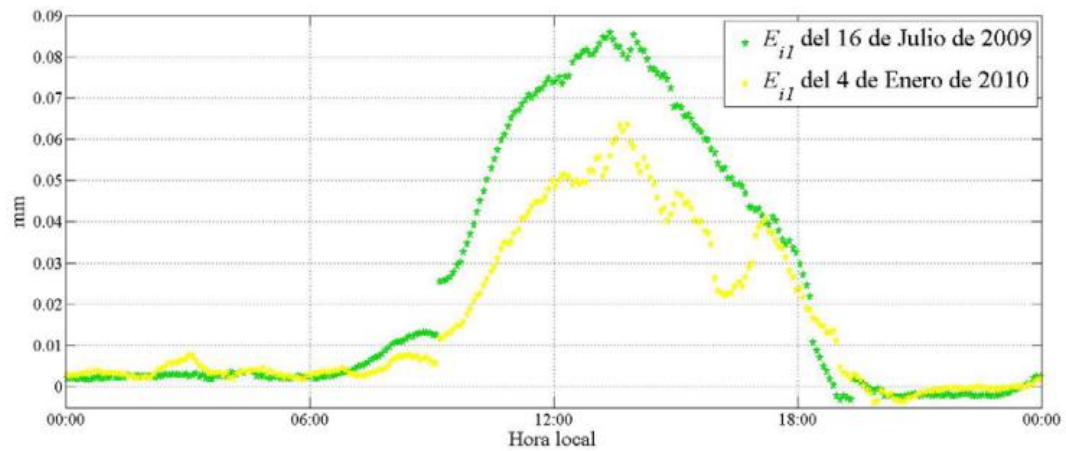
2.2.1 Impacto de la temperatura en la evaporación

La evaporación depende de la cantidad de moléculas de agua que poseen suficiente energía cinética para vencer las fuerzas de atracción en la masa líquida, así como de la capacidad del aire circundante para admitir vapor de agua. Este fenómeno se conoce como el poder evaporante de la atmósfera, que se mide por la cantidad de agua que esta puede absorber.

El poder evaporante de la atmósfera depende principalmente de la radiación solar, seguida por la temperatura del aire, el viento y la humedad. La radiación solar es un factor clave en la evaporación; por ejemplo, durante los días invernales, con menos horas de insolación, la evaporación es menor en comparación con los días de verano, donde el número de horas de insolación es mayor. En la Figura 4 se puede observar que, en un día de invierno, la evaporación (curva amarilla) es menor y más gradual, alcanzando un pico máximo de aproximadamente 0.05 mm alrededor del mediodía y disminuyendo rápidamente después. En contraste, en un día de verano (curva verde), la evaporación es más intensa, alcanzando un valor máximo cercano a 0.08 mm, también alrededor del mediodía, debido a la mayor cantidad de radiación solar y horas de insolación, manteniéndose alta por más tiempo a lo largo del día antes de descender (Velázquez Mendoza, 2010).

Figura 4

Muestras de la evaporación en los días de invierno y verano



Nota. Representación de variación de evaporación en invierno y verano.

Fuente: Adaptado de (Velázquez Mendoza, 2010).

2.2.2 Efectos de la precipitación en el flujo de los canales

La precipitación directa sobre las zonas de captación y las áreas adyacentes a los canales incrementa la cantidad de agua que ingresa al sistema. Este flujo adicional puede ser beneficioso para la recarga de acuíferos y el riego de cultivos, pero también puede ocasionar problemas de control del caudal, especialmente durante lluvias intensas o prolongadas. Según (Ponce, 2014), la esorrentía superficial se clasifica en tres tipos:

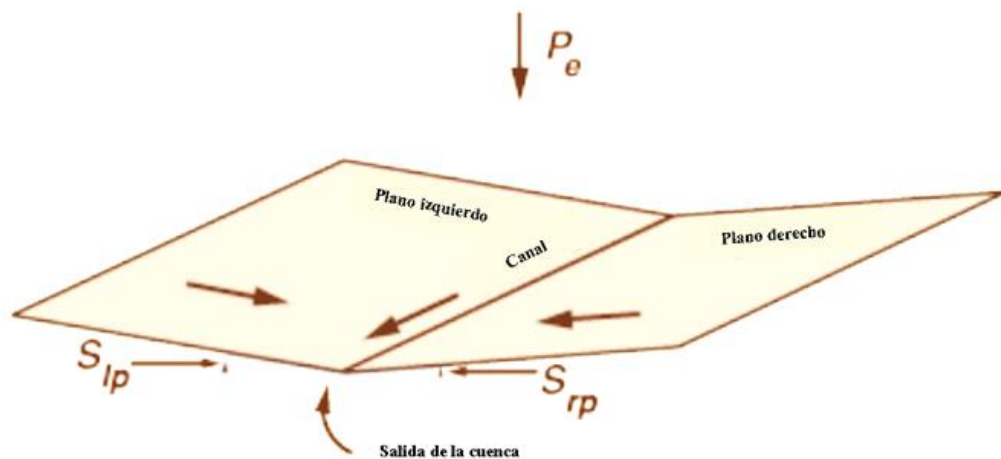
- Flujo concentrado: Se produce cuando la duración efectiva de la precipitación es igual al tiempo de concentración.
- Flujo superconcentrado: En este caso la precipitación supera el límite de concentración, se puede decir que tiene buena duración.
- Flujo subconcentrado: Por otro lado este flujo sucede cuando las lluvias son menores al tiempo de concentración en el canal de riego.

La Figura 5 representa el diagrama de una cuenca en la que la precipitación efectiva Pe cae sobre dos planos inclinados: el plano izquierdo y el plano derecho. El flujo superficial generado en estos planos, denominado como Slp y Srp , se dirige hacia el canal central. Desde allí, el agua fluye hacia la salida de la cuenca. Este modelo simplifica el comportamiento de la escorrentía superficial, permitiendo evaluar cómo la precipitación en áreas adyacentes al canal contribuye al volumen de agua que llega a la salida.

Se puede determinar que estas pruebas son precisas para la planificación de sistemas de drenaje y para prever posibles desbordamientos en canales y ríos, dependiendo de las condiciones de precipitación.

Figura 5

Salida en el hidrograma de la cuenca



Nota: Demostración de libro abierto en la modulación de agua superficial.

Fuente: Adaptado de (Ponce, 2014).

2.2.3 Variabilidad estacional y su gestión

La variación del caudal a lo largo del año es causada por varios factores como las precipitaciones estacionales, los eventos extremos del niño, que provocan exceso y déficit de lluvia en la cuenca. (Llontop Silva & Correa, 2023) recomienda evaluar y analizar registros de

flujo de otros parámetros como cambios en la presión atmosférica, la fuerza del viento, las corrientes de intercambio entre la selva y la costa para encontrar indicadores que predigan los riesgos y peligros asociados con las inundaciones máximas.

2.3 Nivel de agua en canales de riego

El nivel de agua en los canales de riego es crucial para garantizar un suministro adecuado a los cultivos, la medición, el control y la gestión del nivel de agua son aspectos fundamentales en la eficiencia del riego y la prevención de la sedimentación que afecta el paso del agua a los canales de riego.

2.3.1 Sedimento que afecta el paso del agua a los canales de riego

Según (Zavala, Vela , Taípe , & Astete, 2020) determina que la presencia de un dique natural originado a partir de un deslizamiento de tierra representa un peligro continuo, ya que puede presentar fallas o rompedura, generar un desembalse violento no controlado del agua y el material detrítico que compone el dique. La descarga de agua y sedimentos puede formar flujos de detritos, con alta concentración de sólidos, o incluso flujos hiperconectados provocando inundaciones aguas abajo, afectando y erosionando las riberas o laderas inferiores. En este caso estas corrientes pueden afectar estructuras existentes como puentes, terrenos de cultivo y zonas turísticas e incluso afectar zonas agrícolas.

La Figura 6 evidencia un canal de riego donde el flujo de agua no se distribuye de manera equitativa entre dos salidas, en este caso el canal de la derecha transporta más agua debido a la curva que canaliza mayor flujo, mientras que el canal de la izquierda recibe menos agua. Esta desigualdad puede causar a la acumulación de sedimentos, que obstruyen parcialmente el canal de riego, desequilibrando el flujo hacia un lado.

Figura 6*Sedimentación en un canal de riego*

Nota. Representación del flujo de sedimentación en canal de riego.

Fuente: Adaptado de (Abengoa, 2013).

2.3.2 Métodos de medición del nivel de agua

En la actualidad existen muchos métodos y procedimientos para medir el contenido de agua del suelo, algunos de ellos son directos, indirectos, destructivos y no destructivos principalmente los más utilizados y los que tienen mayor potencial de uso.

- Métodos directos: En este aspecto se mide el flujo de agua del suelo, para obtener una muestra en el análisis gravitatorio y evitar desgloses.
- Métodos indirectos: Para este método las mediciones de las propiedades físicas son fundamentales ya que dependen del agua distribuida por el suelo.
- Métodos destructivos: Cambio parcial del suelo en las condiciones del flujo de agua.

- Método no destructivo: Permiten realizar mediciones repetidas a largo plazo no alterarán el suelo y el valor del contenido de humedad en el mismo lugar y a la misma profundidad se puede obtener inmediatamente (Florentino, 2006).

Además de la medición del contenido de humedad en el suelo, en los sistemas de riego es fundamental monitorear el nivel de agua almacenada evitando desgloses en los cultivos, para garantizar un adecuado funcionamiento. Incluso este monitoreo se realiza mediante instrumentos como piezómetros, sensores de presión y sistemas de telemetría.

En la Figura 7 se presenta las fases que contiene la superficie de agua antes de generar el bombeo durante el regadío, acatando toda esta información se puede obtener datos para evitar pérdidas del recurso en los canales y así mismo planificar los tiempos para cada beneficiario.

Figura 7

Sistema de almacenamiento y recuperación de agua subterránea



Nota. Representación del proceso de inyección y recuperación de agua subterránea a través de bombeo, utilizado para almacenamiento temporal y posterior reutilización.

Fuente: Adaptado de (Villatoro, 2024).

2.4 Estándares y tecnologías aplicadas en canales de riego

En Ecuador, los canales de riego están regulados por diversos estándares gubernamentales y organismos especializados en la gestión de recursos del agua. Aquí se detalla el estándar y normativa clave que conlleva el mismo:

- Código Orgánico del Ambiente (COA): Este código nos da a conocer la normativa general para gestionar adecuadamente el recurso de agua en los canales de riego.
- Reglamento para el Aprovechamiento y Uso del Agua: En este caso nos detallan las normas específicas para el uso eficiente del agua en sistemas de riego, incluyendo la infraestructura necesaria, para evitar desbordamientos.
- Manual de Operación y Mantenimiento de Sistemas de Riego: Proporciona directrices técnicas para asegurar la sostenibilidad y eficiencia de los sistemas de riego ya sea por medio de compuertas automatizadas (SENAGUA, 2016).

2.4.1 *Arquitectura de sistema IoT para canal de riego*

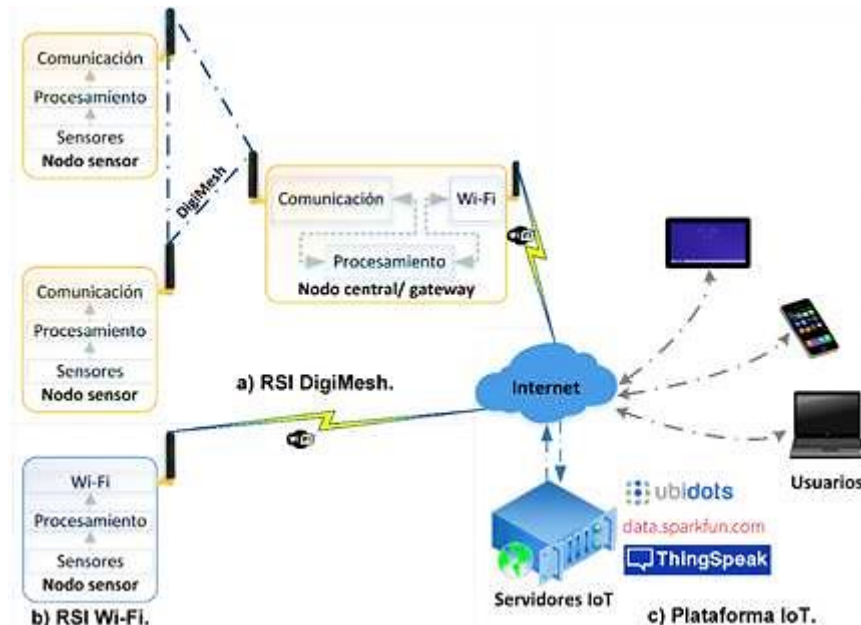
La implementación de un sistema automatizado en el canal de riego permite una gestión inteligente y basada en datos reales para optimizar el uso del agua y la energía en la agricultura. El diseño de una arquitectura de sistema IoT es fundamental para el control de los canales de riego en las comunidades para así evitar desigualdades en los tiempos de regadío ya que nos garantiza una automatización segura y confiable al momento de verificar las notificaciones.

La Figura 8 muestra la arquitectura de un sistema IoT diseñado para el monitoreo y gestión de datos climáticos en canales de riego. En este esquema, se observa cómo los nodos sensores recopilan información ambiental, que se transmite mediante redes inalámbricas hacia

un nodo central, el cual envía los datos a plataformas IoT para su análisis y visualización por parte de los usuarios.

Figura 8

Arquitectura de red IoT para monitoreo y gestión de datos climáticos



Nota. Arquitectura de red IoT compuesta por nodos sensores, nodos centrales y plataformas en la nube, diseñada para el monitoreo de variables climáticas y la gestión de datos en tiempo real.

Fuente: Adaptado de (Quiñones, 2017).

2.4.2 Protocolos IoT aplicados al monitoreo y control de riego

Existen varios protocolos de comunicación en IoT sin embargo, se dará a conocer los más relevantes para los canales de riego.

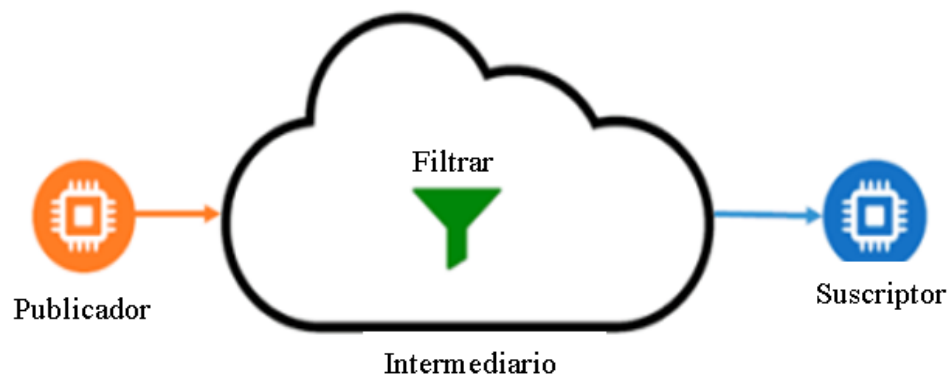
- **Protocolo de comunicación S7COMM:** El protocolo de comunicación S7Comm es un protocolo Siemens y se lleva a cabo entre controladores lógicos programables Siemens S7-300/400/Logo 8 conectados en la misma red.
- **Protocolo MQTT:** Telemetry Transport (MQTT) es un protocolo de comunicación para la transmisión de datos M2M (máquina a máquina). La

operación del protocolo proporciona un servicio de notificaciones push con un modelo de editor/suscriptor (Guambo & Dave, 2022).

La Figura 9 demuestra el funcionamiento del protocolo MQTT, basándose en tres componentes principales: el Publicador que es el encargado del envío de mensaje, el faltado distribuye la información hacia el suscriptor el cual recibe los mensajes permitiendo así un correcto envío de datos al protocolo sin interferencias.

Figura 9

Protocolo MQTT



Nota. Esquema del protocolo MQTT basado en la comunicación entre publicador, intermediario y suscriptor para el intercambio eficiente de datos.

Fuente: Adaptado de (Guambo & Dave, 2022, pág. 32).

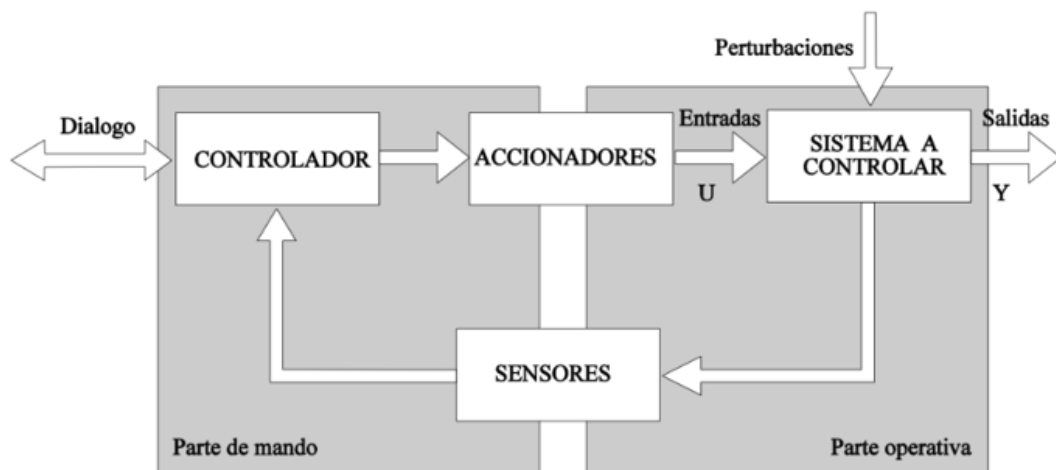
2.5 Sensores y actuadores en riego automatizado

Los sensores y actuadores son una parte importante de cualquier sistema de control de automatización industrial ya que, representan el nivel de campo de la pirámide de automatización. Debido a que los sensores son los sentidos del sistema de control ya que proporcionan información sobre lo que está sucediendo en cambio los actuadores son los brazos del sistema de control ya que permiten cambiar lo que sucede en la planta (Brunete, Herrero, & San Segundo, 2024).

El diagrama de bloques de la Figura 10 muestra la estructura de un sistema de control, dividiéndose en dos partes: Parte de mando y Parte operativa. En la Parte de mando, el Controlador interactúa mediante diálogo y envía señales a los Accionadores, que luego afectan al Sistema a Controlar en la Parte operativa. Este sistema recibe entradas (U) y puede sufrir Perturbaciones externas, generando Salidas (Y) que son monitoreadas por los Sensores. Estos sensores retroalimentan al controlador para ajustar las acciones, cerrando así el lazo de control y manteniendo el sistema bajo supervisión constante.

Figura 10

Diagrama de bloque del sistema de control



Nota. Diagrama de control en lazo cerrado, compuesto por sensores, actuadores y controlador, que regulan un sistema ante variaciones externas.

Fuente: Adaptado de (Brunete, Herrero, & San Segundo, 2024).

2.5.1 Sensor de temperatura DHT22

Este sensor se empleará para tomar mediciones tanto de humedad como de temperatura del aire que rodea al cultivo. El DHT22 es una herramienta ideal para el monitoreo ambiental en aplicaciones agrícolas debido a su alta precisión y estabilidad. Además, la Tabla 1 presenta las características y especificaciones principales del sensor (Chugá Rosero, 2022).

Tabla 1*Características y especificaciones del sensor DHT22*

Características	Especificación
Voltaje de operación	3.3 – 5V
Consumo de corriente	2.5mA
Rango de medición de humedad	0% a 100% RH
Precisión de medición de humedad	±2%
Resolución de humedad	0.1% RH
Histéresis de humedad	<±0.3% RH
Rango de medición de temperatura	-40 A 80°C
Precisión de medición de temperatura	± 0.5°C
Resolución de temperatura	0.1 °C
Tipo de salida	Digital

Nota. Características del sensor DHT22 con sus respectivas especificaciones.

Fuente: Adaptado de (Chugá Rosero, 2022, pág. 33).

2.5.2 Sensor de humedad SHT10

Este sensor será utilizado para medir la temperatura y la humedad del suelo en el área de cultivo, debido a que el sensor SHT10 proporciona datos precisos y confiables, esenciales para gestionar adecuadamente las condiciones del suelo y optimizar el riego y la fertilización del cultivo (Chugá Rosero, 2022). La Figura 11 muestra la apariencia física del sensor.

Figura 11*Sensor SHT10 de temperatura*

Nota. Representación del sensor SHT10.

Fuente: Adaptado de (Chugá Rosero, 2022, pág. 33).

2.5.3 Sensor de nivel de agua

El uso del sensor de nivel de agua es un dispositivo básico de detección de agua / gotas de lluvia, que genera una señal analógica que se ajusta a la humedad registrada, o al nivel de agua recogido. Es un sensor de tamaño reducido diseñado para identificar el agua pluvial o pequeñas variaciones en el nivel del agua para identificar fugas o filtraciones de líquido (Novatronic, 2024).

La Tabla 2 resume las principales especificaciones técnicas del sensor. Así mismo, la Figura 12 muestra la apariencia física del sensor de nivel de agua.

Tabla 2*Especificaciones técnicas del sensor de nivel de agua*

Características	Especificaciones
Tipo de sensor	Analógico
Área de detección	40 mm x 16 mm
	10 °C a 30 °C

Temperatura de funcionamiento	
Humedad	10% - 90% sin condensación
Dimensiones	65 mm x 20 mm x 8 mm
Voltaje de operación	3-5 VDC
Corriente de operación	-20 mA

Nota. Características del sensor de agua con sus especificaciones.

Fuente: Adaptado de (Novatronic, 2024).

Figura 12

Sensor de nivel de agua



Nota. Visualización del sensor de nivel de agua.

Fuente: Adaptado de (Novatronic, 2024).

2.6 Placas de desarrollo y microcontroladores

Las placas de desarrollo, también conocidas como kits de desarrollo, son una herramienta esencial para los diseñadores de sistemas. Estas placas facilitan y aceleran el desarrollo y la prueba de proyectos, ya que incorporan todo el hardware necesario para ejecutar el microcontrolador de destino.

Las placas de desarrollo suelen incluir memoria adicional, puertos de entrada y salida, LED, interruptores, un programador y otras interfaces periféricas, como UART, I2C y SPI. El hardware de estas placas ha sido probado previamente por los fabricantes, por lo que el diseñador puede centrarse en el software de la aplicación, sabiendo que el hardware funciona correctamente (Ibrahim, 2021).

2.6.1 *Microcontroladores*

Placas basadas en una pequeña computadora, generalmente construidas con procesos CMOS. Para que el microcontrolador pueda llevar a cabo sus funciones, es necesario proporcionarle un conjunto de instrucciones precisas que indiquen cómo debe interactuar con los componentes del sistema. Estas instrucciones se codifican en un programa de software que se carga en el microcontrolador para su ejecución (Terán et al., 2022).

A continuación, se presenta los cuatro bloques funcionales básicos: CPU (unidad central de procesamiento), memoria ROM (memoria solo de lectura), memoria RAM (memoria de acceso aleatorio) y periféricos de entrada y salida. Estos bloques están interconectados y trabajan en conjunto para llevar a cabo las tareas específicas que se le asignan dependiendo del modelo y fabrica del producto (Terán et al., 2022).

2.6.2 *Placa ESP-32*

La placa ESP32 está compuesta por chip, el módulo y la placa de desarrollo. Cada componente ofrece diferentes especificaciones lo cual se debe analizar adecuadamente antes de seleccionarlasy e integrarlas en el sistema. A continuación, se presenta una breve comparativa de estos formatos:

- **ESP32 Chips:** Este componente se enfoca en un circuito integrado constituido por microcontroladores y su conexión es inalámbrica vía wifi, sobre todo es ideal para sistemas personalizados que lleven limitaciones como smartwatches.
- **ESP32 Módulos:** Estos módulos son pre-fabricados que contiene un chip ESP32 junto con componentes adicionales como un regulador de voltaje, oscilador de cristal y antena. Son una forma sencilla de añadir conectividad inalámbrica a dispositivos IoT.
- **ESP32 Placas de Desarrollo:** Constituido con una plataforma de hardware , así mismo se integra un chip ESP32, interfaces USB, sensores integrados y reguladores de voltaje siendo ideales para construcción y pruebas rápidas de dispositivos automatizados.

2.6.3 Comparativa de placas de desarrollo

La placa ESP32 ha sido acogida con una gran demanda en la población ya que se ha destacado por ser un componente de gran capacidad inalámbrica, visión clara en las operaciones incluyendo conectividad segura, interfaces periféricas, manejo de datos entiendo real y dimensiones físicas de microcontroladores.

2.6.4 Características de las placas de desarrollo

Las placas de desarrollo basadas en ESP32 ofrecen una amplia gama de características que las hacen adecuadas para diversas aplicaciones en el ámbito del IoT. Estas características pueden incluir capacidades inalámbricas (WiFi, Bluetooth), interfaces periféricas (I2C, SPI, UART), capacidad de procesamiento, capacidad de almacenamiento y soporte para sensores y actuadores. Además, se debe considerar la capacidad de gestión de energía y la facilidad de programación de estas placas.

2.7 Base de datos y almacenamiento de información

La gestión eficaz de los recursos hídricos agrícolas requiere una base de datos sólida para la toma de decisiones, debido a que el método en cascada de canales de riego se centra en la recopilación, almacenamiento y procesamiento de datos climáticos para su posterior uso en la planificación del riego (Cáceres, 2023).

Este enfoque nos ayuda a utilizar de gran manera el recurso de agua evitando desperdicios o desbordamientos del canal, permitiendo identificar patrones climáticos antes de generar pérdidas en los cultivos de los agricultores construyendo un canal de riego eficiente y sostenible.

2.7.1 Gestión de datos climáticos mediante metodología en cascada

La metodología en cascada aplicada en este proyecto sigue un enfoque secuencial para la gestión de datos en canales de riego, organizando las fases de adquisición, almacenamiento, procesamiento y análisis de información climática. Este modelo permite garantizar la fiabilidad e integridad de los datos, asegurando que cada fase del sistema dependa de la finalización de la anterior.

Para desarrollar el prototipo se debe realizar un inventario de información recogida del canal, superficie asociada, distribución del agua, para cada beneficiario y los efectos que causaría el sedimento por los alrededores del canal. Los datos son adquiridos mediante sensores, luego se almacenan en la base de datos posterior a eso se realiza el análisis por medio del protocolo MQTT para evidenciar el comportamiento de los actuadoras y después se visualiza en la página web junto con las notificaciones en tiempo real (Durand Nole, 2021).

2.7.2 Almacenamiento de datos climáticos

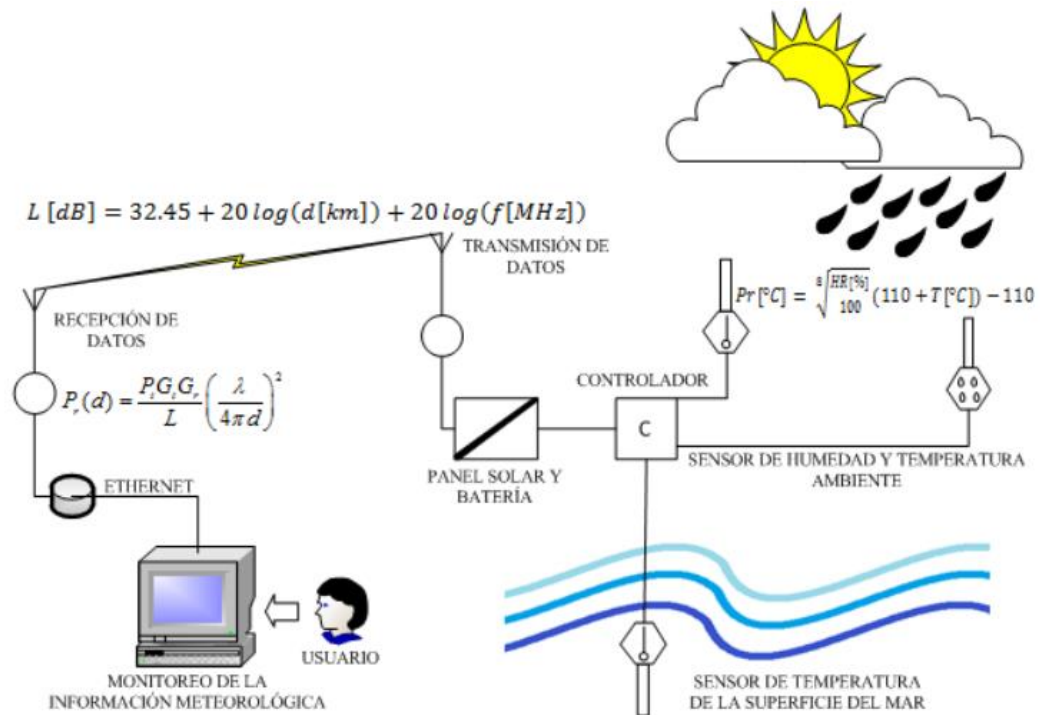
El almacenamiento de datos climáticos es fundamental para la gestión eficaz del riego. La selección de sistemas de almacenamiento de datos climáticos adecuados garantiza la disponibilidad y la integridad de la información utilizada en la toma de decisiones relacionadas con el riego (Aranibar, 2023).

Los modelos climáticos son herramientas cruciales para comprender climas pasados y predecir climas futuros mediante experimentos con variaciones en las concentraciones de gases de efecto invernadero, la eficiencia de los modelos de circulación general ha mejorado gradualmente, evolucionando algunos en modelos del sistema terrestre que integran ciclos biogeoquímicos del carbono, nitrógeno y biogeoquímica oceánica (Carril, 2023).

Como se muestra en la Figura 13, es necesario realizar recopilaciones de los datos climáticos con los componentes electrónicos como los sensores que utilizan en los cultivos para asegurar un monitoreo seguro en la toma de decisiones durante el gestionamiento del sistema automatizado.

Figura 13

Sistema de monitoreo y almacenamiento de datos climáticos

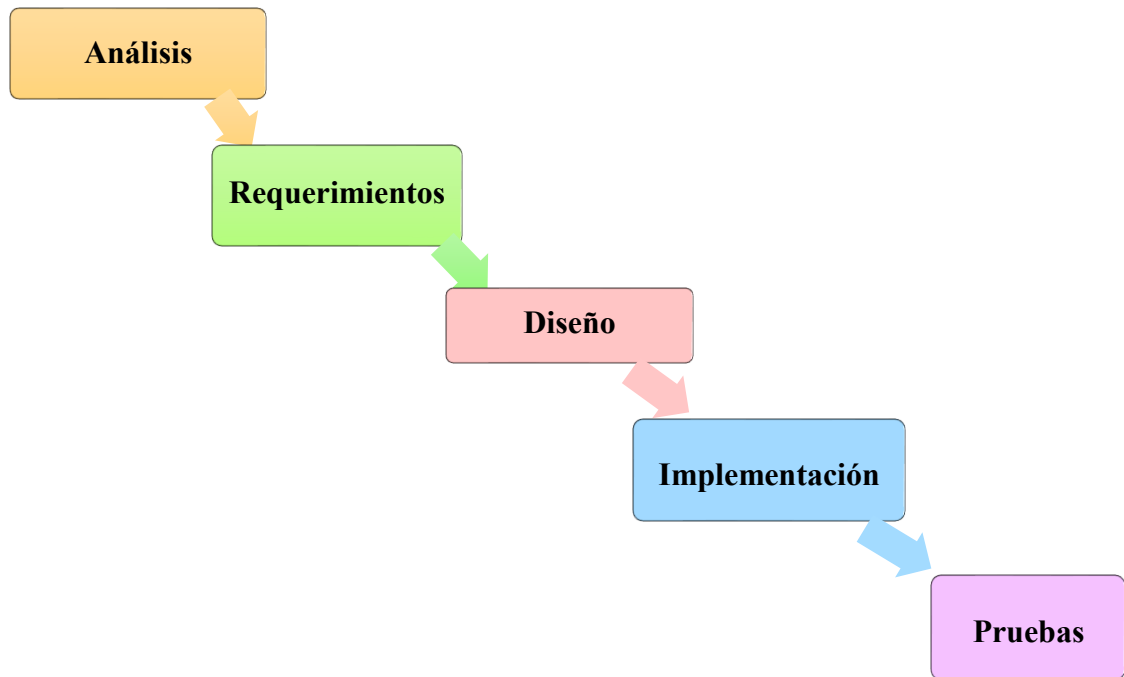


Nota. Esquema del sistema de monitoreo y almacenamiento de datos climáticos mediante sensores ambientales y transmisión de datos a una plataforma de usuario.

Fuente: Adaptado de (Lascano, 2014).

2.8 Metodología de desarrollo

El desarrollo del prototipo se lleva a cabo mediante el modelo en cascada, el cual se caracteriza por su proceso en fases claramente definidas, garantizando un avance ordenado y controlado entre cada etapa. En la Figura 14 se presenta la estructura general del modelo, el cual está conformado por cinco fases: análisis, requerimientos, diseño, implementación y pruebas de validación. A continuación, se detalla el desarrollo correspondiente a cada una de estas fases.

Figura 14*Metodología en cascada*

Nota. El modelo en cascada está conformado por cinco fases que es la fase de análisis, fase de requerimientos, fase de diseño, fase de implementación y la fase de pruebas.

Fuente: Adaptado de (Royce, 2019).

- 1. Análisis:** Esta fase consiste en examinar el entorno donde se desarrollará el proyecto, evaluando la situación actual del sistema de riego, la ubicación geográfica y las delimitaciones del área de estudio. Permitiendo definir el propósito del sistema que se desarrollará en un entorno controlado.
- 2. Requerimientos:** Esta fase consiste en definir los requerimientos tanto hardware como de software para el desarrollo del prototipo, identificando los componentes y herramientas requeridas para su correcto funcionamiento en el sistema, así mismo asegurando que cumplan con los criterios técnicos establecidos para un mejor trabajo.
- 3. Diseño:** Durante esta etapa, se define la arquitectura del sistema, especificando cómo se distribuirán y conectarán los diferentes componentes, lo cual se desarrollará un

esquema del prototipo, evaluando la viabilidad de los materiales seleccionados y su integración con los módulos de programación.

4. **Implementación:** En esta etapa se procede a la incorporación de los componentes físico del prototipo, entre ellos es cargar los códigos que fueron configurando en cada sensor, actuador y microcontrolador permitiendo la operación del sistema. Después se procede a la conexión de sensores, actuadores y módulos de comunicación, garantizando una interacción adecuada entre el hardware y el software, lo que asegura el funcionamiento automatizado y eficiente del sistema de riego.
5. **Pruebas:** En esta etapa se realizan pruebas funcionales del prototipo, el cual se debe evidenciar la adquisición de datos de los sensores de nivel de agua, luego el procesamiento local entre el ESP32 y el Arduino posterior a eso se debe verificar la transmisión inalámbrica por medio del wifi hacia el protocolo MQTT para la validación de los datos en el sistema, una vez almacenada en la base de datos María DB se visualiza en la plataforma web y se observa las notificaciones que van llegando con el accionamiento de las compuertas automatizadas en el canal garantizando un funcionamiento seguro y confiable.

3 Capítulo III: Diseño del sistema de monitoreo y control de riego

El presente capítulo consiste en desarrollar el diseño del sistema de monitoreo y control de riego propuesto, describiendo en detalle la metodología del “Modelo en Cascada” y sus diferentes etapas, lo que permite obtener resultados estructurados y coherente del sistema.

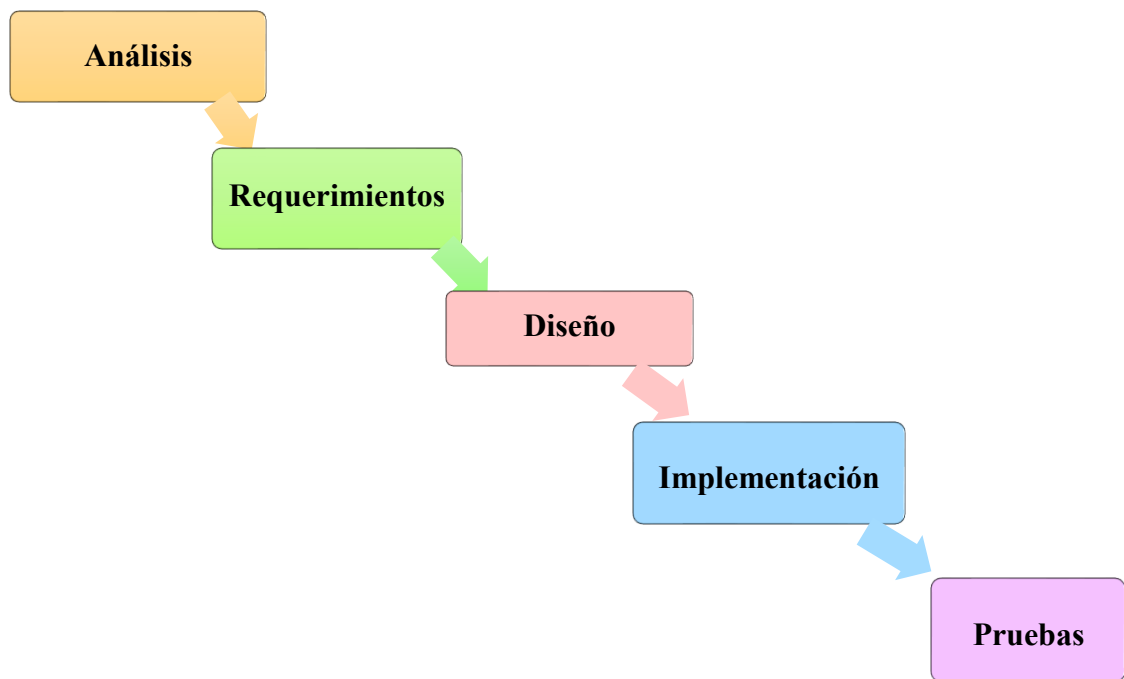
3.1 Metodología

El desarrollo en cascada es una metodología estructurada y secuencial que organiza el proyecto en fases claramente delimitadas, donde cada etapa debe completarse y validarse antes de avanzar a la siguiente, lo cual su enfoque lineal permite un control riguroso de cada fase, garantizando que las pruebas sirvan como insumos directos y verificados para el desarrollo posterior (Royce, 2019).

La Figura 15 demuestra las fases que componen a la metodología del modelo en cascada, en este caso está compuesto por cinco etapas principales: Análisis, Requerimientos, Diseño, Implementación y Pruebas, las cuales permiten estructurar el proceso de forma secuencial, ordenada y alineada con los objetivos planteados en el desarrollo del proyecto.

Figura 15

Fases de la metodología en cascada.



Nota. El modelo en cascada está conformado por cinco fases siendo la fase de análisis, fase de requerimientos, fase de diseño, fase de implementación y la fase de pruebas.

Fuente: Adaptado de (Royce, 2019).

3.2 Etapa de Análisis

Esta etapa consiste en examinar el entorno en el que se lleva a cabo el proyecto incluyendo una evaluación minuciosa del presente estado del sistema, la localización de la zona de estudio, sus fronteras geográficas y los objetivos del sistema sugerido. Esta evaluación preliminar posibilita entender las necesidades concretas del entorno, detectar los problemas existentes y definir las bases técnicas que orientarán las fases siguientes del desarrollo del prototipo.

3.2.1 Situación Actual

Actualmente, el sistema de riego en el barrio Río Frío – Cotacachi se gestiona de forma manual, lo que genera ineficiencias en la distribución del agua y dificulta su control en tiempo

real. La falta de monitoreo adecuado ocasiona pérdidas innecesarias y limita la optimización del uso del agua, afectando la eficiencia en el riego y su disponibilidad.

Ante esta situación, surge la necesidad de desarrollar un sistema de monitoreo y control de riego que permita la recopilación de datos del nivel de agua, facilitando la toma de decisiones y mejorando la eficiencia en su distribución.

La Figura 16 corresponde a una referencia visual del terreno de aplicación, ubicado en el barrio Río Frío – Cotacachi, el cual ha sido considerado únicamente como guía para contextualizar el entorno en el que se enfoca el proyecto.

Figura 16

Referencia del terreno de aplicación



Nota. La presente figura muestra el terreno donde se encuentra ubicado el canal de riego, así como la distribución hacia los diferentes predios aledaños.

Fuente: Adaptado de Google Earth.

3.2.2 Ubicación de la zona

El presente proyecto se desarrollará en la ciudad de Cotacachi, provincia de Imbabura, específicamente en el barrio Río Frío, el cual se encuentra ubicado a tres cuadras y media del Parque Central de Cotacachi "La Matriz".

Esta zona ha sido seleccionada debido a su actividad agrícola predominante y a la necesidad de optimizar el uso del agua en los sistemas de riego. En esta área, los agricultores se dedican principalmente al cultivo de maíz, fréjol, habas, arvejas, papas y aguacates, productos que requieren un suministro adecuado de agua para evitar pérdidas en la producción y garantizar la sostenibilidad del cultivo. Sin embargo, el sistema de riego actual es manual, lo que genera problemas de distribución y uso eficiente del recurso (Tapia & Carrera, 2011).

La Figura 17 muestra un mapa satelital de la ciudad, donde se señala tanto el parque central de Cotacachi como la ubicación del barrio Río Frío, destacando la distancia entre ambos puntos, incluso se observa la infraestructura urbana y la cercanía del área agrícola, resaltando la importancia de mejorar el sistema de riego en la zona.

Figura 17

Ubicación del barrio Río Frío en Cotacachi



Nota. En la figura se muestra la ubicación geográfica del barrio Río Frío en la ciudad de Cotacachi, indicando su cercanía al Parque Central y su contexto urbano circundante.

Fuente: Adaptado de Google Earth.

3.2.3 *Delimitación geográfica*

El área de estudio tiene su enfoque en la ciudad de Cotacachi, provincia de Imbabura, específicamente en el barrio Río Frío, donde se evaluará el funcionamiento del prototipo IoT y la automatización de canales de riego.

Para definir con precisión la zona de estudio, se ha utilizado la herramienta Google Earth, lo que ha permitido obtener las coordenadas geográficas de los puntos más relevantes dentro del área de estudio.

Las coordenadas de referencia del área de estudio son:

- **Centro del área:** 0°18'14"N, 78°16'01"W
- **Límites del área de estudio:**
 - **Norte:** 0°18'12"N – Barrio Diablo Calle
 - **Sur:** 0°18'21"N – Barrio 24 de mayo
 - **Este:** 78°15'55"W – Barrio Morales
 - **Oeste:** 78°16'02"W – Barrio Esmeraldas

En la Figura 18, se presenta la delimitación geográfica del área de estudio, incluyendo las cuadrículas de latitud y longitud facilitando la referencia exacta del barrio Río Frío, asegurando una mejor planificación y posible expansión del sistema en el futuro.

Figura 18

Delimitación geográfica con coordenadas de latitud y longitud



Nota. La figura muestra la delimitación geográfica del canal de riego en el barrio Río Frío, con coordenadas referenciales de latitud y longitud.

Fuente: Adaptado de Google Earth.

3.2.4 Propósito del sistema

El objetivo de este proyecto tiene como finalidad crear un prototipo basado en tecnología IoT que permita automatizar y optimizar la gestión del sistema de riego en el barrio Río Frío, en Cotacachi. Esta iniciativa nace de las necesidades detectadas a través de una encuesta realizada a los habitantes del área, donde se identificaron problemas relacionados con la gestión manual del riego, el desperdicio de agua y la falta de equidad en su distribución.

El sistema está desarrollado por sensores de nivel de agua en los canales de riego. Los datos recolectados serán procesados y transmitidos en tiempo real, lo que permitirá a los responsables del sistema tomar decisiones informadas y gestionar el agua de manera más eficiente mediante la activación automática de compuertas.

Como complemento del prototipo, se utilizará la aplicación de mensajería telegram para enviar notificaciones automáticas a los usuarios sobre la disponibilidad del canal de riego. Esta funcionalidad facilitará una coordinación equitativa del recurso entre los diferentes beneficiarios, optimizando los turnos de riego y evitando conflictos o desperdicios innecesarios.

Por medio de este prototipo se busca reducir el desperdicio de agua, promover un uso más racional del recurso y contribuir a una gestión comunitaria más sostenible, ofreciendo al sector una solución práctica, accesible y segura.

3.3 Etapa de Requerimientos

La etapa de requerimientos tiene como objetivo establecer los elementos esenciales del sistema, tales como los beneficiarios y actores involucrados, las funciones esperadas, las restricciones del entorno y los criterios técnicos que deberán cumplirse en las etapas posteriores del proyecto.

A continuación, se presentan los diferentes tipos de requerimientos que conforman esta etapa, los cuales abarcan aspectos funcionales y no funcionales de Stakeholders, Sistema, Arquitectura, hardware, software y comunicación.

3.3.1 *Requerimientos*

La presente sección establece normas esenciales para el desarrollo del prototipo IoT, definiendo los requisitos de hardware y software necesarios para su correcta operación en un entorno controlado. Por el cual se ha elegido la norma ISO/IEC/IEEE 29148:2018 que nos ayuda a controlar y validar el proceso que se vaya realizando durante el desarrollo del prototipo.

Este análisis permite garantizar la compatibilidad del sistema con las tecnologías de medición, comunicación inalámbrica y procesamiento de datos, asegurando su interoperabilidad y eficiencia. A continuación, se detallan los requerimientos específicos del sistema, organizados en hardware, software y comunicación.

3.3.2 *Beneficiarios*

Los Stakeholders son aquellos actores que tienen un interés directo o indirecto en el desarrollo, implementación y resultados del prototipo automatizado en los canales de riego. En este caso, los beneficiarios directos son los vecinos del barrio Río Frío, quienes utilizan el canal de riego para el suministro de agua en sus terrenos agrícolas.

Por otro lado, la Tabla 3 nos muestra los beneficiarios indirectos quienes pueden verse involucradas en la regulación y optimización del uso del agua en la comunidad. La participación de estos actores es clave para evaluar la viabilidad del sistema y su posible aplicación en futuras mejoras del sistema de riego.

Tabla 3

Lista de Stakeholders

Lista de Stakeholders
1. Dueño del terreno
2. Vecinos del barrio Río frío
3. MSc. Carlos Vásquez (Director del trabajo de titulación)
4. MSc. Luis Suárez (Asesor del trabajo de titulación)
5. Ramos Ana (Desarrolladora del trabajo de titulación)

Nota. Demostración de la lista de las personas involucradas para los Stakeholders.

Fuente: Adaptado de (ISO/IEC/IEEE, 2018).

3.3.3 Nomenclatura de los requerimientos

A continuación, se presentan la nomenclatura y los criterios establecidos para el diseño de las tablas que contienen información clave del proyecto.

La Tabla 4 muestra las siglas utilizadas para facilitar la comprensión y gestión de los términos relacionados con los requisitos. Lo cual esta tabla abarca los diferentes tipos de requerimientos, tanto de hardware como de software, así como aquellos vinculados con los datasheet.

Tabla 4

Definición de acrónimos

Acrónimo	Descripción
StSR	Requerimientos de Stakeholders
SySR	Requerimientos del Sistema
SRSR	Requerimientos de Arquitectura

Nota. Representación de los acrónimos que se van a usar en cada requerimiento.

Fuente: Adaptado de (ISO/IEC/IEEE, 2018).

El modelo adoptado permite clasificar los requerimientos del sistema según su nivel de prioridad, esta clasificación se establece en base a criterios como la relevancia para el usuario, el impacto del sistema y el grado de riesgo que representa para el diseño e implementación del prototipo automatizado.

La Tabla 5 presenta la categorización de los requerimientos del sistema de acuerdo con su prioridad.

Tabla 5*Prioridad de los requerimientos del prototipo del canal de riego*

Prioridad	Descripción
Alta	Corresponde a los requerimientos esenciales para el funcionamiento del prototipo. Su ausencia afecta directamente la automatización y el control del sistema.
Media	Se refiere a requerimientos importantes cuya falta puede reducir el rendimiento o precisión, pero no impide el funcionamiento básico.
Baja	Incluye aquellos requerimientos que añaden valor adicional, pero cuya ausencia no afecta la operatividad general y pueden añadirse posteriormente.

Nota. Definición de prioridades que se van a colocar en los diferentes requerimientos.

Fuente: Adaptado de (ISO/IEC/IEEE, 2018).

3.3.4 Requerimientos de Stakeholders

Se entiende por stakeholders a los individuos o grupos que se ven afectados directa o indirectamente por el desarrollo del proyecto y sus resultados. La definición de los requerimientos de stakeholders (StSR) tiene como propósito identificar las necesidades, expectativas y condiciones que estos actores consideran esenciales para el sistema.

En la Tabla 6 se puede verificar los requerimientos operacionales y de usuario reflejando la interacción tanto de los usuarios y del sistema de riego automatizado sin olvidarse de evaluar las condiciones necesarias para su adecuado funcionamiento dentro del entorno comunitario.

Tabla 6*Requerimientos de Stakeholders*

Requerimientos del Sistema (SiRS)				
Nº	Requerimiento	Prioridad		
		Alta	Media	Baja
Requerimientos operacionales				
SiRS1	Monitorear el nivel de agua en tiempo real.	X		
SiRS2	Operar con bajo consumo energético.	X		
SiRS3	Enviar alertas automáticas a los usuarios.	X		
SiRS4	Garantizar acceso equitativo al agua.	X		
SiRS5	Registrar y almacenar datos recolectados.		X	
SiRS6	Reducir el desperdicio de agua.	X		
Requerimientos de uso				
SiRS8	Notificar sobre disponibilidad del canal.	X		
SiRS9	Fácil de usar para todo tipo de usuario.	X		
SiRS10	Interfaz clara en idioma español.		X	
SiRS11	Plataforma para consultar datos.		X	
SiRS12	Mostar estado de humedad del suelo.	X		
SiRS13	Proteger la privacidad de los datos.	X		

Nota. Representación de los requerimientos de stakeholders con sus respectivas prioridades.

Fuente: Elaborado por el autor.

Los requerimientos establecidos delimitan las funciones fundamentales del prototipo, teniendo en cuenta tanto el entorno operativo como la interacción con el usuario. Se da

prioridad a un sistema que permita el monitoreo en tiempo real, que sea de bajo consumo energético, que envíe notificaciones automáticas y que garantice un acceso equitativo al agua.

Estos lineamientos permiten al desarrollo del proyecto tener un enfoque alineado con la realidad, los cuales han sido respaldados por los usuarios del sector mediante una encuesta que se encuentra anexada en el Anexo 1 y Anexo 2.

3.3.5 *Requerimientos del Sistema*

Los requerimientos del sistema (SySR) son ítems que nos permiten validar el funcionamiento del prototipo, los cuales han sido clasificados en categorías de uso, performance, interfaz, modos/estados y requerimientos físicos, en estrecha relación con los requisitos de Stakeholders previamente definidos.

A continuación, se describen en la Tabla 7 los requerimientos iniciales del sistema (SySR).

Tabla 7

Requerimientos iniciales del sistema

Requerimientos del Sistema (SySR)				
REQUERIMIENTOS INICIALES DEL SISTEMA				
Nº	Requerimientos	Prioridad		
		Alta	Media	Baja
Requerimiento de uso				
SySR1	El sistema deberá monitorear en tiempo real el nivel de agua mediante sensores.	X		

SySR2	Generar y enviar alertas automáticas ante valores críticos detectados.	X
SySR3	Notificar a los usuarios la disponibilidad del canal de riego a través de la app.	X
SySR4	Registro de las lecturas de sensores de nivel en una base de datos segura.	X
SySR5	Acceso de consulta histórica del sensor.	X
Requerimientos de interfaz		
SySR6	Visualizar datos de sensores en una aplicación móvil o plataforma web.	X
SySR7	Utilizar sensores inalámbricos compatibles con el microcontrolador	X
SySR8	Monitorear información en español con terminologías sencillas.	X
SySR9	Emitir alertas visuales y auditivas en la aplicación móvil	X
SySR10	Indicar el estado activo/inactivo de los sensores en la interfaz gráfica.	X
Requerimientos de performance		
SySR11	Operar de forma continua durante 24 horas sin intervención manual.	X
SySR12	Optimizar el consumo energético en periodos de inactividad.	X
SySR13	Actualizar y transmitir datos de sensores cada 10 minutos.	X

SySR14	Asegurar una precisión de $\pm 5\%$ en las mediciones de los sensores.	X
SySR15	Mantener el rendimiento ante variaciones térmicas de 0°C a 50°C .	X
Requerimientos de modos/estados		
SySR16	Alternar entre modo activo y modo de bajo consumo energético.	X
SySR17	Cambiar automáticamente a modo de reposo cuando haya estabilidad.	X
SySR18	Activar alertas al exceder límites preestablecidos en las variables.	X
SySR19	Permitir el reinicio manual del modo de operación vía app.	X
SySR20	Indicar el estado del sistema mediante un LED de funcionamiento.	X
Requerimientos físicos		
SySR21	Proteger sensores contra humedad, polvo y radiación solar.	X
SySR22	Para uso real, el sistema requerirá carcasa resistente. En el prototipo no es necesaria por su uso controlado.	X
SySR23	Mantener dispositivos compactos para facilitar su instalación.	X
SySR24	Asegurar físicamente los módulos de comunicación inalámbrica.	X
SySR25	En futura implementación, instalar sensores en puntos estratégicos para optimizar detecciones.	X

Nota. Representación de los requerimientos iniciales del sistema con sus respectivas prioridades.

Fuente: Elaborado por el autor.

3.3.6 *Requerimientos de Arquitectura*

En los requerimientos de arquitectura (SRSH) se definen las características de hardware, software y del sistema eléctrico necesarias para el desarrollo del prototipo. Estas directrices son esenciales para la selección adecuada de los componentes tecnológicos a emplear en el proyecto. A continuación, se presentan en la Tabla 8 los requerimientos de arquitectura.

Tabla 8

Requerimientos de Arquitectura

SRSH				
REQUERIMIENTOS DE ARQUITECTURA				
N°	Requerimientos	Prioridad		
		Alta	Media	Baja
Requerimientos lógicos				
SRSH1	El sistema deberá transmitir datos de sensores en tiempo real a la plataforma de monitoreo durante pruebas.	X		
SRSH2	El sistema deberá detectar automáticamente niveles críticos de agua.	X		
SRSH3	El sistema deberá registrar eventos críticos de alerta durante las pruebas.	X		

SRSH4	Garantizar que los datos transmitidos no se pierdan durante la operación de prueba.	X
SRSH 5	Utilizar un protocolo de comunicación segura para transmisión de datos (ejemplo, MQTT).	X
Requerimientos de diseño		
SRSH6	El prototipo deberá permitir fácil acceso a sensores y componentes para mantenimiento o ajustes.	X
SRSH7	El diseño deberá ser compacto y modular para facilitar transporte y manejo en ambiente controlado.	X
SRSH8	Los sensores deberán instalarse en ubicaciones estratégicas dentro del prototipo para optimizar mediciones.	X
SRSH9	El sistema deberá estar protegido contra desconexiones accidentales durante las pruebas.	X
SRSH 10	La plataforma de monitoreo deberá tener una interfaz simple y amigable para el usuario.	X
Requerimiento de hardware		
SRSH11	El prototipo deberá utilizar una placa microcontroladora compatible con el sensor de nivel de agua.	X
SRSH12	Se requiere un módulo de comunicación inalámbrica (Wi-Fi) para la transmisión de datos en tiempo real durante pruebas.	X
SRSH13	Los sensores deberán tener protección básica contra humedad para ambiente controlado.	X

SRS14	El prototipo deberá integrar una batería o fuente portátil para asegurar funcionamiento durante las pruebas.	X
SRS15	El hardware deberá soportar la conexión simultánea de múltiples sensores.	X
Requerimiento de software		
SRS16	El software deberá leer datos de sensores y enviarlos automáticamente a la plataforma de monitoreo.	X
SRS17	La plataforma de monitoreo deberá mostrar gráficamente el nivel de agua.	X
SRS18	El sistema deberá generar alertas automáticas al detectar valores críticos en las pruebas.	X
SRS19	El software de monitoreo deberá ser accesible desde navegadores web o dispositivos Android.	X
SRS20	La plataforma deberá almacenar datos de prueba para su posterior análisis.	X
Requerimientos electrónicos		
SRS21	El prototipo deberá operar con voltaje de entre 5V y 12V.	X
SRS22	Se deberá usar una fuente de alimentación estable y protegida contra picos de corriente.	X
SRS23	La conexión eléctrica del prototipo deberá proteger los componentes frente a cortocircuitos accidentales.	X
SRS24	El sistema deberá contar con un regulador de voltaje para asegurar la alimentación constante de los sensores.	X

SRS25	El sistema deberá incluir protección contra polaridad X inversa para evitar daños a los componentes electrónicos.
--------------	---

Nota. Representación de los requerimientos de arquitectura con las prioridades.

Fuente: Elaborado por el autor.

3.3.7 Selección de hardware y software

Se debe tomar en cuenta que la selección de los componentes de hardware y software, se elabora mediante una tabla comparativa basada en los requerimientos de Stakeholders (StRS), Sistema (SySR) y Arquitectura (SRS25). Cada componente es evaluado según su prioridad, el cual si cumple con los requerimientos establecidos, marcando "1" para el cumplimiento y "0" para la falta de cumplimiento. Esta evaluación facilita la identificación de los elementos que ofrecen el mejor desempeño y compatibilidad para el desarrollo del prototipo automatizado.

3.3.7.1 Hardware

La selección del hardware se llevó a cabo teniendo en cuenta los requerimientos establecidos en la Tabla 8 sobre Requerimientos de Arquitectura. En este proceso, se decidió el sistema embebido principal, las placas auxiliares para la gestión de sensores, así como los módulos para la adquisición de datos ambientales.

En un principio, se consideró el uso de módulos ESP8266. Sin embargo, se decidió reemplazarlos por tres placas Arduino Uno, debido a su compatibilidad con sensores

analógicos, su estabilidad en el funcionamiento y la facilidad que ofrecen para la programación. Por otro lado, se eligió la placa ESP32 como unidad central, gracias a su potente capacidad de procesamiento, conectividad integrada y eficiencia energética.

3.3.7.2 Sistema Embebido

La presente placa debe garantizar el procesamiento eficiente de datos, la lectura de sensores y la comunicación inalámbrica en tiempo real. Dado el enfoque IoT del sistema de riego, se prioriza bajo consumo energético, conectividad integrada y compatibilidad con sensores analógicos.

Para una selección adecuada, en la Tabla 9 se comparan diferentes opciones de hardware embebido, evaluadas en función de los requerimientos técnicos definidos previamente.

Tabla 9

Comparación de diferentes placas embebidas

Requisitos	Placa embebida		
	ESP32	Arduino uno	ESP8266(Alternativo)
SHRS1	1	1	1
SHRS2	1	1	1
SHRS3	1	1	1
SHRS4	1	0	1
SHRS5	1	1	1
SHRS6	1	1	0
SHRS7	1	1	0
Total SI	7	6	5

1 Cumple
0 No cumple

Elección: En la Tabla 9 se destaca que la placa ESP32 es más adecuada, gracias a su conectividad WiFi, procesamiento de doble núcleo y eficiencia energética. También la placa de Arduino uno cumple con los requisitos clave para lectura de sensores y control de actuadores.

En cambio, la placa ESP8266 se consideró como alternativa por sus limitaciones en el manejo de múltiples sensores.

Nota. Representación de las placas embebidas que se va a utilizar en el prototipo.

Fuente: Elaborado por el autor.

A continuación, en la Tabla 10 se evidencia las características técnicas principales de las placas seleccionadas.

Tabla 10

Especificaciones técnicas de ESP32 y placa Arduino uno

PLACA ESP32	
ESPECIFICACIONES	PROPIEDADES
Procesador	Tensilica Xtensa 32bits LX6 hasta 240MHz.
Wi-Fi	802.11b/g/n/e/i (802.11n @ 2.4 Ghz hasta 150 Mbit/s)
Bluetooth	v4.2 BR/EDR y bluetooth Low Energy (BLE).
Rom	448 KiB.
SRAM	520 KiB
RTC slow SRAM	8 KiB.
RTC fast SRAM	8 KiB.
eFuse	1 Kbit

Flash embebida	0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips); 2 MiB (ESP32-D2WD chip); 4 MiB (ESP32-PICO-D4 SIP module).
Periféricos compatibles	ADC, DAC, I2C, UART, Interfaz CAN 2.0, SPI, I2S, RMII y PWM entre otros.
Seguridad tipo	IEEE 802.11, WFA, WPA/WPA2 y WAPI.
Criptografía soportada por acelerador de hardware	AES, SHA-2, RSA, ECC, RNG.
Voltaje de trabajo	3.3VDC.
Energía y datos	Vía conector microUSB 5VDC.
PLACA ARDUINO UNO	
Microcontrolador	ATmega328
Voltaje Operativo	5v
Voltaje de Entrada	7 – 12v
Pines de Entradas/Salidas	14 (De las cuales 6 son salidas PWM)
Digital	
Pines de Entradas Análogas	6
Memoria Flash	32 KB (ATmega328) de los cuales 0,5 KB es usado por Bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad del Reloj	16 MHZ

Nota. La placa ESP32 es una placa de alto rendimiento con procesador de 240 MHz y conectividad wi-fi y bluetooth. Incluso Arduino Uno es una placa basada en el ATmega328, con memoria suficiente para tareas de control de sensores y actuadores, operando a 5V y adecuada para proyectos básicos de automatización.

Fuente: Adaptado de (Sigma Electrónica, 2022), (Guerrero, 2014).

3.3.7.3 Consumo energético del sistema

El prototipo IoT opera con una alimentación de 5 V DC, suministrada mediante conexión USB y adaptadores de corriente. El consumo energético total del sistema se estimó considerando los Arduino uno, Esp32, sensores y servomotores utilizados. El mayor consumo corresponde a los servomotores durante los instantes de apertura y cierre de las compuertas, mientras que el ESP32 y los Arduino uno presenta un consumo estable y reducido proporcionando un consumo de 4.75 W, valor adecuado para un prototipo IoT en ambiente controlado, tal como se detalla en la Tabla 11.

Tabla 11

Consumo energético del prototipo

Componente	Cantidad	Voltaje (V)	Corriente aprox. (mA)	Potencia (W)
ESP32	1	5	80	0.40
Arduino uno	2	5	50	0.50
Sensor nivel de agua	2	5	10	0.10
Servomotores	3	5	250	3.75
Total estimado			950 mA (0.95 A)	4.75 W

Nota. El consumo energético total del prototipo se estimó considerando los valores nominales de corriente de los componentes electrónicos utilizados durante la operación normal del sistema en un ambiente controlado.

Fuente: Elaborado por el autor.

➤ Corriente total del sistema

En este apartado se procede a realizar la suma de la corriente consumida por cada componente del sistema, con el fin de determinar la corriente total, mediante la Ecuación (1).

Datos:

- ESP32 (WiFi activo): 80 mA
- Arduino Uno $\times 2$: 50 mA c/u $\rightarrow$ 100 mA
- Sensor de nivel $\times 2$: 10 mA c/u $\rightarrow$ 20 mA
- Servomotores (3): 250 mA c/u $\rightarrow$ 750 mA

Sustituyendo los valores:

$$I_{total} = \sum I_i \quad (1)$$

$$I_{total} = I_{ESP32} + I_{Arduinos} + I_{sensores} + I_{servomotores}$$

$$I_{total} = 80 \text{ mA} + 100 \text{ mA} + 20 \text{ mA} + 750 \text{ mA}$$

$$I_{total} = 950 \text{ mA}$$

➤ Conversión en amperios

Para expresar la corriente total en amperios, se realiza la conversión de miliamperios mediante la Ecuación (2), lo cual permite unificar las unidades empleadas en el cálculo de la potencia eléctrica.

$$I (A) = \frac{I(mA)}{1000 \text{ mA}} \quad (2)$$

$$I (A) = \frac{950 \text{ mA}}{1000 \text{ mA}}$$

$$I = 0.95 \text{ A}$$

➤ Cálculo de la potencia total

Una vez obtenida la corriente total del sistema en amperios, se procede al cálculo de la potencia eléctrica total consumida por el prototipo utilizando la Ecuación (3), considerando el voltaje de alimentación del sistema.

$$P = V \cdot I \quad (3)$$

$$P = 5V \cdot 0.95 A$$

$$P = 4.75 W \text{ (vatios)}$$

3.3.8 *Software*

Una vez seleccionado el hardware, se procede a la selección del software, el cual debe cumplir con los requerimientos de software establecidos en la Tabla 8 sobre Requerimientos de Arquitectura. En esta sección se evalúan las opciones de software que mejor se ajusten a los requisitos del sistema.

Dado que el objetivo principal de este proyecto es el desarrollo de un sistema IoT para el monitoreo y control de canales de riego, es crucial seleccionar una plataforma de programación eficiente, fácil de implementar y que permita la integración con sensores y actuadores. Además, es importante que el software sea compatible con los módulos de comunicación inalámbrica y que cumpla con los requisitos de procesamiento en tiempo real, fundamentales para monitorear datos como el nivel de agua y la humedad del suelo.

3.3.8.1 **Software de programación**

Para la selección del software de programación se evaluaron diferentes opciones, las cuales se adaptan particularmente a los requerimientos del proyecto para la codificación del sistema. En la Tabla 12 se presenta la comparación de tres lenguajes de programación, evaluados según su compatibilidad con sensores y control de actuadores.

Tabla 12*Elección del software de programación*

Lenguaje de Programación	Requisitos					Total
	SiRS4	SiRS5	SHRS16	SHRS19	SHRS23	
Python	1	1	1	1	1	6
C++	1	1	0	1	1	4
Java	1	1	1	0	1	3
1 Cumple						
0 No cumple						

Elección: En la presente tabla se comparan tres lenguajes de programación lo cual Python fue seleccionado por su compatibilidad con ESP32 a través de micropython y su facilidad para el desarrollo rápido de prototipos, también C++ es usado en Arduino Uno ya que es ideal para el control de hardware y manejo de sensores.

Nota. Representación del software de programación que se utilizara para el desarrollo del proyecto.

Fuente: Elaborado por el autor.

3.3.8.2 Software de base de datos

Para el almacenamiento de la información generada por los sensores del sistema IoT se seleccionó una base de datos relacional MariaDB, desplegada en una instancia EC2 de AWS, lo cual permite un control directo sobre el entorno de ejecución y una integración eficiente con el backend del sistema. Esta solución resulta adecuada para un prototipo IoT en ambiente controlado, ya que garantiza la persistencia, organización y consulta histórica de los datos registrados.

3.3.8.3 Software de visualización

La visualización de los datos del sistema IoT permite supervisar en tiempo real el nivel de agua y el estado de las compuertas automatizadas. Para este propósito, se seleccionó una plataforma web desarrollada en HTML y CSS, desplegada en un servidor web Nginx sobre una instancia EC2 de AWS, la cual permite el acceso a la información desde cualquier navegador web. Esta solución facilita la interpretación de los datos y la supervisión del sistema sin incrementar la complejidad ni los costos de implementación.

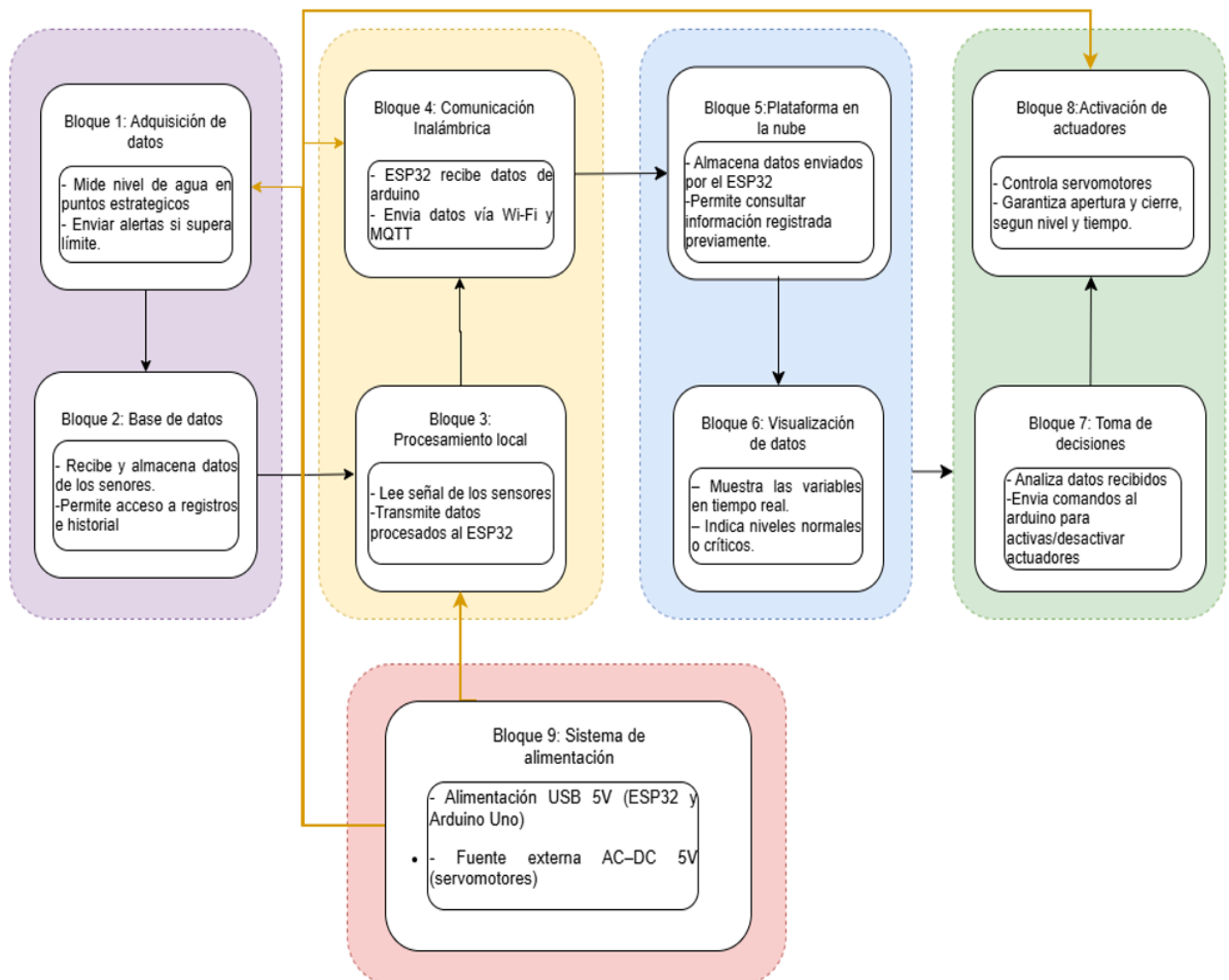
3.4 Diseño

Una vez que se han seleccionado los componentes de hardware y software necesarios para el sistema de automatización de riego, se procede a establecer las pautas de diseño del sistema. Se consideran los criterios previamente evaluados, los cuales facilitarán la adecuada integración del sistema y optimizarán el uso del agua en los canales de riego.

En esta etapa se presenta el funcionamiento del sistema mediante diagramas de bloques y diagramas de flujo que proporcionan una estructura clara para guiar los procesos de programación y la ejecución del sistema final.

3.4.1 Diagrama de bloque del sistema

La Figura 19 presenta el diagrama de bloques del sistema IoT para riego automatizado. El sistema se organiza en ocho bloques funcionales que representan el flujo completo de la información, desde la adquisición de datos hasta la activación de los actuadores. Cada bloque representa la interacción entre sensores, microcontroladores, comunicación inalámbrica, plataforma en la nube y el sistema de toma de decisiones, permitiendo visualizar de manera clara y secuencial el funcionamiento del sistema en un entorno controlado.

Figura 19*Diagrama de bloques del sistema*

Nota: El diagrama muestra la arquitectura funcional del sistema IoT y el flujo lógico de la información. El Bloque sistema de alimentación representa el suministro eléctrico a los microcontroladores y actuadores mediante alimentación USB y una fuente externa de 5 V, sin formar parte del flujo de datos.

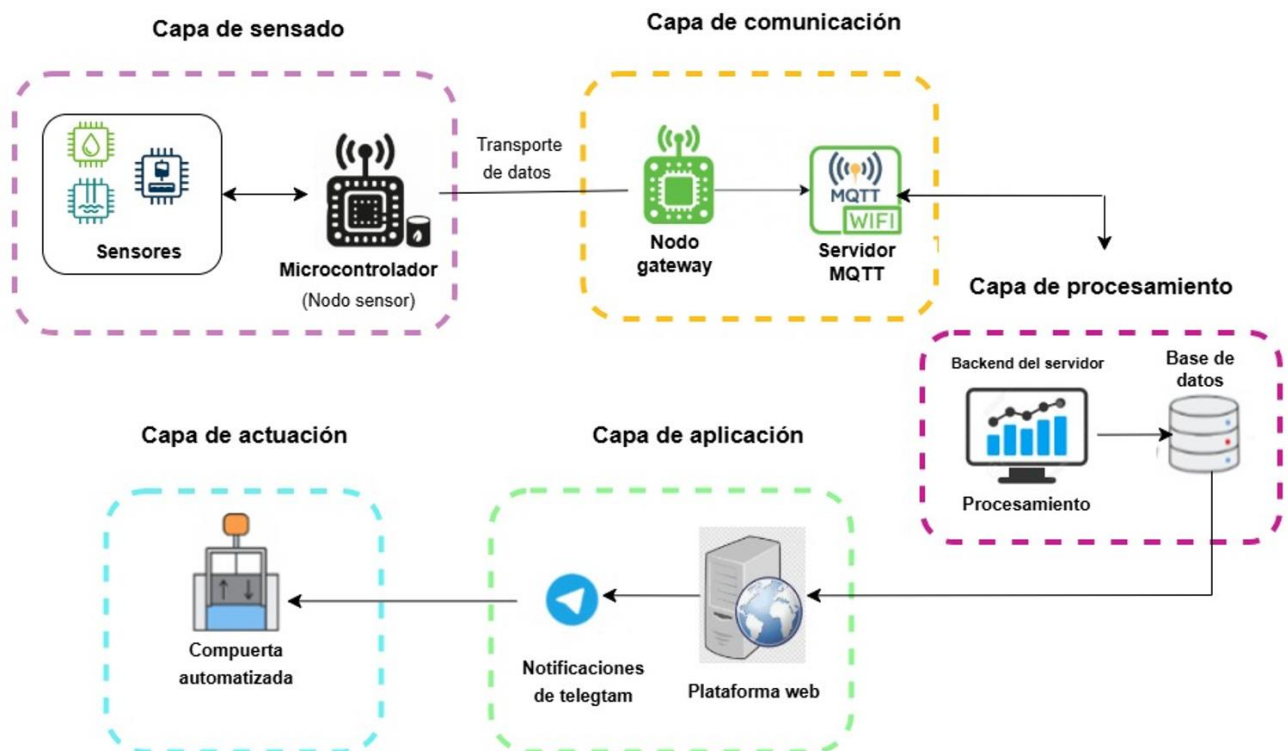
Fuente: Elaborado por el autor.

- **Bloque 1: Adquisición de datos:** Este bloque es responsable de captar y medir las variables físicas relevantes para el control del sistema de riego, principalmente el nivel de agua en puntos estratégicos del canal. Los sensores envían señales digitales o analógicas que representan el estado del nivel de agua para su procesamiento local.
- **Bloque 2: Base de datos:** En este caso los datos obtenidos por los sensores se almacenan temporalmente en la unidad local del sistema y posteriormente se envían de manera estructurada a la plataforma de almacenamiento, permitiendo el registro histórico de las lecturas.
- **Bloque 3: Procesamiento local:** Consiste en microcontroladores organizados en nodos que tienen la función de leer y verificar las señales provenientes de los sensores. Conectan los microcontroladores con la distribución a cada sensor de nivel, las cuales recogen información del entorno de manera modular y que se puede escalar.
- **Bloque 4: Comunicación inalámbrica:** La placa ESP32, que cuenta con conectividad Wi-Fi, transmite los datos procesados a la nube por medio del protocolo MQTT, permitiendo un acceso remoto desde cualquier dispositivo conectado a la red.
- **Bloque 5: Plataforma de monitoreo:** Podemos determinar que la interfaz tanto para web como para dispositivos móviles permite observar en tiempo real las variables del sistema, las alertas y la información histórica, ya que ha sido creada para hacer más sencilla la supervisión y la toma de decisiones, cuenta con notificaciones automáticas que se pueden ajustar.
- **Bloque 6: Visualización de datos:** Muestra en la aplicación móvil o plataforma web los valores actuales de las variables medidas, como el nivel de agua, y señales claras de si se ha sobrepasado algún límite establecido. Esta visualización sencilla permite al usuario conocer el estado en tiempo real sin necesidad de análisis gráfico, facilitando una interpretación rápida y directa.

- **Bloque 7: Toma de decisiones:** En este bloque se analiza los datos en tiempo real para abrir o cerrar las compuertas según el accionar del recurso de agua y el tiempo configurado para así optimizar el uso del agua en el sector.
- **Bloque 8: Activación de actuadores:** Este bloque ejecuta físicamente las órdenes del sistema de control para regular el flujo de agua en los canales. Utiliza servomotores que abren y cierran las compuertas de riego de forma precisa y coordinada, según las instrucciones de la lógica de control, garantizando una distribución equitativa del agua conforme a los turnos establecidos.
- **Bloque 9: Sistema de alimentación:** En este caso el bloque suministra energía eléctrica regulada a los componentes del sistema. El Arduino Uno y el ESP32 operan con una alimentación de 5 V vía USB, asegurando estabilidad en las tareas de control y comunicación. Los servomotores se alimentan mediante una fuente externa AC-DC de 5 V, independiente del sistema de control, lo que permite cubrir mayores demandas de corriente y evita caídas de tensión e interferencias eléctricas, garantizando un funcionamiento confiable del sistema.

3.4.2 *Arquitectura del sistema*

La Figura 20 muestra la arquitectura IoT del sistema distribuida en cinco capas que son: la capa de sensado captura las mediciones del nivel de agua; la capa de comunicación transporta los datos hacia el gateway mediante I2C, Wi-Fi y MQTT; la capa de procesamiento gestiona la lógica del backend y el almacenamiento en la base de datos; la capa de aplicación proporciona la visualización y notificaciones al usuario. Finalmente, la capa de toma de decisiones y actuación ejecuta las órdenes de control, activando la compuerta automatizada según las condiciones detectadas.

Figura 20*Arquitectura general del sistema*

Nota. Representación gráfica del sistema IoT para riego automatizado.

Fuente: Elaborado por el autor.

3.4.3 Diagrama de adquisición de datos

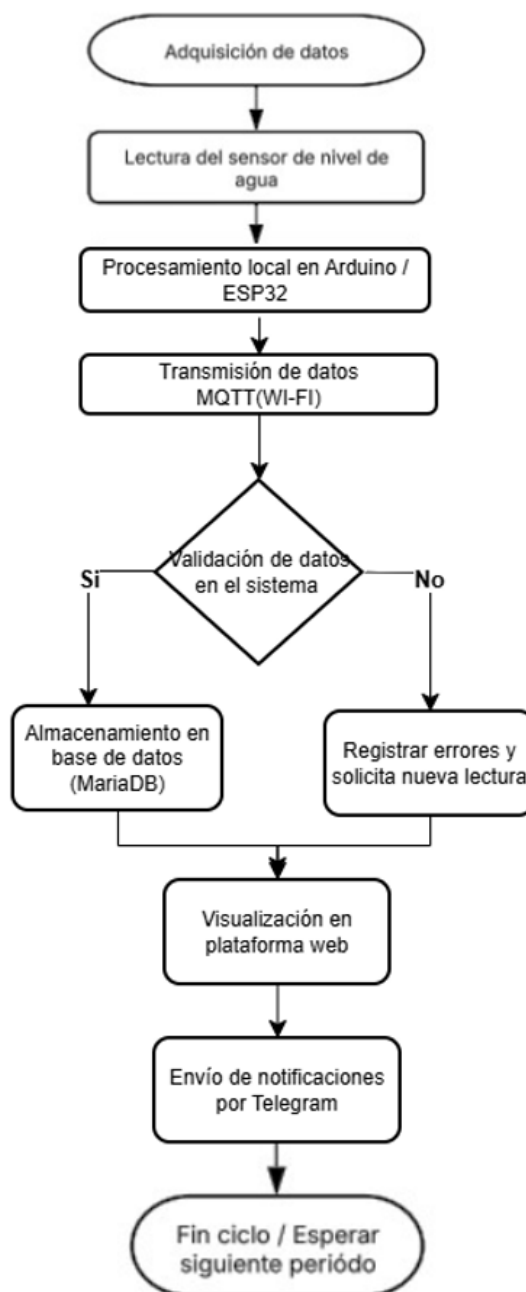
La Figura 21 inicia con la lectura del nivel de agua mediante un sensor específico ubicado en el canal de riego, la señal obtenida por el sensor es transmitida al microcontrolador principal utilizando el protocolo de comunicación I2C. Una vez recibidos, los datos son sometidos a un proceso de validación para verificar su integridad y coherencia. En caso de que los datos sean válidos, se almacenan localmente en una base de datos para su posterior análisis y registro histórico.

Si se detectan errores o inconsistencias en los datos, se registra el incidente y se solicita una nueva lectura para garantizar la fiabilidad del sistema. Finalmente, los datos validados y almacenados son presentados al usuario a través de una interfaz gráfica en una aplicación móvil

o plataforma web, permitiendo el monitoreo en tiempo real del nivel de agua en el sistema de riego.

Figura 21

Diagrama de flujo de adquisición de datos



Nota. Representación del proceso de lectura y recolección de variables ambientales mediante sensores.

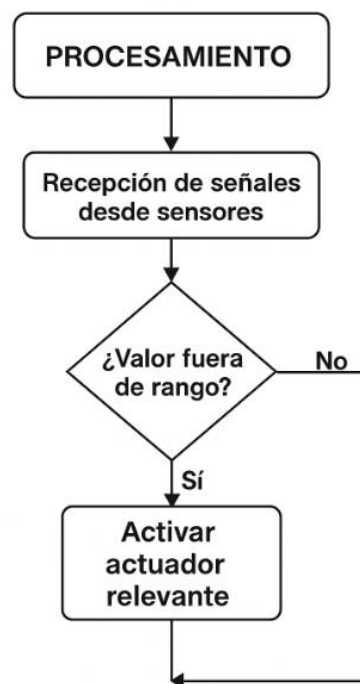
Fuente: Elaborado por el autor.

3.4.4 Diagrama de flujo de procesamiento y activación de actuadores

En este flujo, tras recibir las señales de los sensores, los datos se comparan con valores históricos o umbrales predefinidos. Si se detectan valores fuera de rango o variaciones significativas, se activa el actuador correspondiente para controlar el flujo de agua como se evidencia en la Figura 22 el procedimiento.

Figura 22

Diagrama de flujo de procesamiento y activación.



Nota. Proceso de análisis de señales recibidas, evaluación de condiciones y activación de electroválvulas.

Fuente: Elaborado por el autor.

3.4.5 Fuerza hidrostática

En esta etapa de diseño también se deben considerar los aspectos físicos inherentes al área de desarrollo del prototipo tales como la longitud del canal de riego, ancho, profundidad, etc. Estos parámetros se evidencian en la Tabla 13, ya que permiten realizar el cálculo de la

fuerza hidrostática mediante la cual se pudo dimensionar la capacidad de apertura del sistema y la fuerza requerida por el motor para poder abrir y cerrar las compuertas bajo la presión del flujo del agua que atraviesa el canal de riego.

Tabla 13

Parámetros utilizados para el cálculo

Símbolo	Descripción	Valor utilizado	Unidad
P	Densidad del agua	1000	kg/m ³
G	Aceleración por gravedad	9.81	m/s ²
B	Ancho de la compuerta	0.81	m
H	Altura del agua sobre la compuerta	0.18	m
M	Masa de la compuerta	2.5	Kg
D	Brazo de palanca del peso	0.15	m
M	Coefficiente de fricción	0.3	—
F _r	Fuerza radial sobre el eje	20	N
R	Radio del eje	0.015	M
Θ	Ángulo de giro del eje	90	Grados
T	Tiempo de operación	2	S

Nota. Los valores corresponden a condiciones reales observadas en el diseño del sistema de compuerta para el canal de riego del barrio Río Frío.

Fuente: Elaborado por el autor.

➤ Cálculo del torque hidrostático T_h

Se puede determinar que el torque hidrostático T_h representa el momento de fuerza ejercido por la presión del agua contenida en el canal sobre la superficie de la compuerta. Este valor se calcula mediante la Ecuación (4), la cual esta presión varía con la profundidad y se distribuye de forma desigual según la altura del agua, lo que produce una fuerza que tiende a girar la compuerta sobre su eje.

$$T_h = \frac{1}{6} \cdot \rho \cdot g \cdot b \cdot h^3 \quad (4)$$

Donde:

$$\rho = 1000 \frac{\text{kg}}{\text{m}^3} \rightarrow \text{Densidad del agua}$$

$$g = 9.81 \text{m/s}^2 \rightarrow \text{Aceleración de la gravedad}$$

$$b = 0.81 \text{ m} \rightarrow \text{Ancho de la compuerta}$$

$$h = 0.18 \text{ m} \rightarrow \text{Altura agua sobre la compuerta}$$

Sustituyendo los valores:

$$T_h = \frac{1}{6} \cdot (1000 \text{ kg/m}^3) \cdot (9.81 \text{m/s}^2) \cdot (0.81 \text{ m}) \cdot (0.18 \text{ m})^3$$

$$T_h = \frac{1}{6} \cdot 1000 \cdot 9.81 \cdot 0.81 \cdot 0.005832 \text{ m}^3$$

$$T_h = 7.72 \text{ N} \cdot \text{m}$$

➤ Torque por peso de compuerta

El momento de fuerza generado por el peso de la compuerta se refiere al efecto de rotación que se produce cuando la fuerza gravitacional actúa sobre la masa de la compuerta a cierta distancia del plano de rotación. Esta acción tiende a hacer que la compuerta gire hacia abajo, lo que puede favorecer o contrarrestar el movimiento, dependiendo de si el sistema intenta abrirse o cerrarse. Este valor se calcula mediante la Ecuación (5).

$$T_p = m \cdot g \cdot d \quad (5)$$

$$T_p = 2.5 \text{ kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \cdot 0.15 \text{ m}$$

$$T_p = 3.68 \text{ N} \cdot \text{m}$$

➤ Torque por fricción del eje

En un sistema mecánico real, la fricción entre el eje y los soportes genera una resistencia al giro, la cual debe ser superada por el actuador. Es importante considerar este esfuerzo de rotación, ya que permite evitar subestimaciones en el diseño del actuador, especialmente en ambientes reales donde el desgaste, el polvo y otros factores pueden incrementar la fricción con el tiempo y se calcula mediante la Ecuación (6).

$$T_f = \mu \cdot F_r \cdot r \quad (6)$$

$$T_f = 0.3 \cdot 20 \text{ N} \cdot 0.015 \text{ m}$$

$$T_f = 0.09 \text{ N} \cdot \text{m}$$

➤ Cálculo del torque para abrir y cerrar compuerta

El momento de fuerza total del movimiento depende de la suma (o resta) de los efectos individuales causados por la presión del agua en la puerta, el peso de la estructura y la fricción en el eje de rotación.

Para la apertura de la compuerta, el sistema debe vencer simultáneamente la presión hidrostática, el peso de la compuerta y la fricción mecánica, por lo que el momento de fuerza total requerido se expresa mediante la Ecuación (7).

$$T_{\text{abrir}} = T_h + T_p + T_f \quad (7)$$

$$T_{\text{abrir}} = 7.72 \text{ N} \cdot \text{m} + 3.68 \text{ N} \cdot \text{m} + 0.09 \text{ N} \cdot \text{m}$$

$$T_{\text{abrir}} = 11.49 \text{ N} \cdot \text{m}$$

En cambio, para el cierre de la compuerta se utiliza la Ecuación (8), ya que el peso de la compuerta actúa a favor del movimiento por lo que se resta del momento de fuerza total.

$$T_{\text{cerrar}} = T_h + T_f - T_p \quad (8)$$

$$T_{\text{abrir}} = 7.72 \text{ N} \cdot \text{m} + 0.09 \text{ N} \cdot \text{m} - 3.68 \text{ N} \cdot \text{m}$$

$$T_{\text{abrir}} = 4.13 \text{ N} \cdot \text{m}$$

En base a los valores obtenidos, se puede determinar con precisión el esfuerzo mínimo que debe generar el actuador para garantizar un funcionamiento confiable del sistema bajo condiciones reales.

➤ Velocidad angular de eje (ω)

Para determinar la fuerza ejercida por el actuador durante la apertura y el cierre de la puerta, se debe conocer la velocidad angular del eje de rotación. Esta se calcula mediante la Ecuación (9), basada en el ángulo de apertura deseado y el tiempo de movimiento estimado.

$$\omega = \frac{\theta \cdot \pi}{180 \cdot t} \quad (9)$$

$$\omega = \frac{90 \cdot \pi}{180 \cdot 2 \text{ s}} = \frac{\pi}{4}$$

$$\omega = 0.785 \text{ rad/s}$$

➤ Potencia requerida para abrir y cerrar compuerta

Una vez conocida la velocidad angular del eje, se determina la potencia mecánica necesaria para abrir y cerrar la compuerta multiplicando el torque total por la velocidad. Esta relación se expresa en la Ecuación(10) para la apertura y la Ecuación(11) para el cierre.

$$P_{\text{abrir}} = T_{\text{total}} \cdot \omega \quad (10)$$

$$P_{\text{abrir}} = 11.49 \, \text{N} \cdot \text{m} \cdot 0.785 \, \text{rad/s}$$

$$P_{\text{abrir}} = 9.03 \, \text{W}$$

$$P_{\text{cerrar}} = T_{\text{total}} \cdot \omega \quad (11)$$

$$P_{\text{cerrar}} = 4.13 \, \text{N} \cdot \text{m} \cdot 0.785 \, \text{rad/s}$$

$$P_{\text{cerrar}} = 3.25 \, \text{W}$$

Los valores de potencia calculados permiten establecer los parámetros mínimos de rendimiento que debe cumplir el actuador, garantizando así su capacidad para abrir y cerrar la compuerta de forma eficiente y segura. Este dimensionamiento considera condiciones reales de funcionamiento, en las que pueden presentarse variaciones en la carga, la fricción o condiciones ambientales que afecten el rendimiento del sistema.

3.4.6 *Lectura del nivel de Agua*

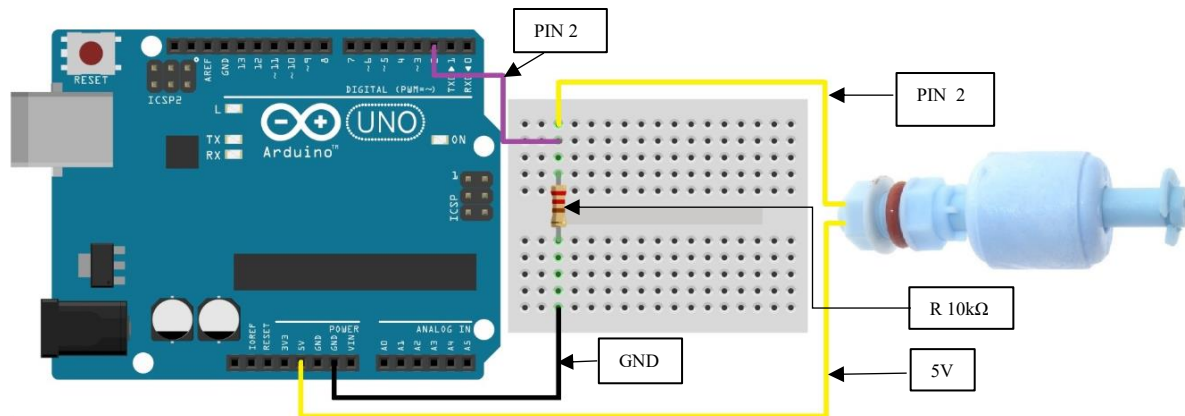
Para la toma de lecturas del nivel de agua se utilizan los sensores del tipo flotador los cuales a través del desplazamiento de una boya integrada permiten conocer si el nivel de agua alcanza cierto límite requerido, para este caso en particular se utiliza una maqueta didáctica de la zona de riego planteada inicialmente, para ello se establecen 4 sensores ubicados a lo largo del canal de riego donde el límite se ubica cercano a la altura máxima del canal.

Primero se establece el diagrama electrónico de los sensores de nivel a utilizar, donde se establece un pin digital 2 del Arduino Uno para la lectura de los estados posibles (0) y (1), donde el estado 1 representara la presencia de máxima de agua en el canal. Como se muestra en la Figura 23 el sensor de nivel cuenta con tres líneas de conexión siendo: alimentación (5V), tierra (GND) y salida de señal se conecta a un extremo de una resistencia de 10 kΩ, cuyo otro

extremo va a tierra, lo que asegura una lectura estable cuando el sensor no está activado. Mediante la variación de estos estados y la coordinación de los sensores en conjunto se podrá conocer el estado actual del canal en un punto dado.

Figura 23

Diagrama de conexión sensor de nivel.



Nota. Representación de la conexión del sensor de nivel de agua al microcontrolador para su lectura.

Fuente: Elaborado por el autor.

Posteriormente en la Figura 24 se establece el código necesario para la toma de lecturas a través del Arduino Uno, donde se establece el pin digital 2 a utilizarse y a manera de ejemplo se muestran los valores en pantalla mediante el puerto de comunicación serial. A partir de este módulo se interconectan los demás sensores con la misma lógica, sin embargo, se debe considerar la ubicación y los actuadores que permitirán abrir o cerrar las compuertas asociadas a cada una de las áreas de riego.

Figura 24

Código de lectura de sensores

```
// Lectura de sensor individual
int PIN = 2;           // Se define el pin digital 2 como la entrada del sensor de nivel de agua
int levelSensor = 0;   // Variable para almacenar el estado lógico leído del sensor (0 o 1)

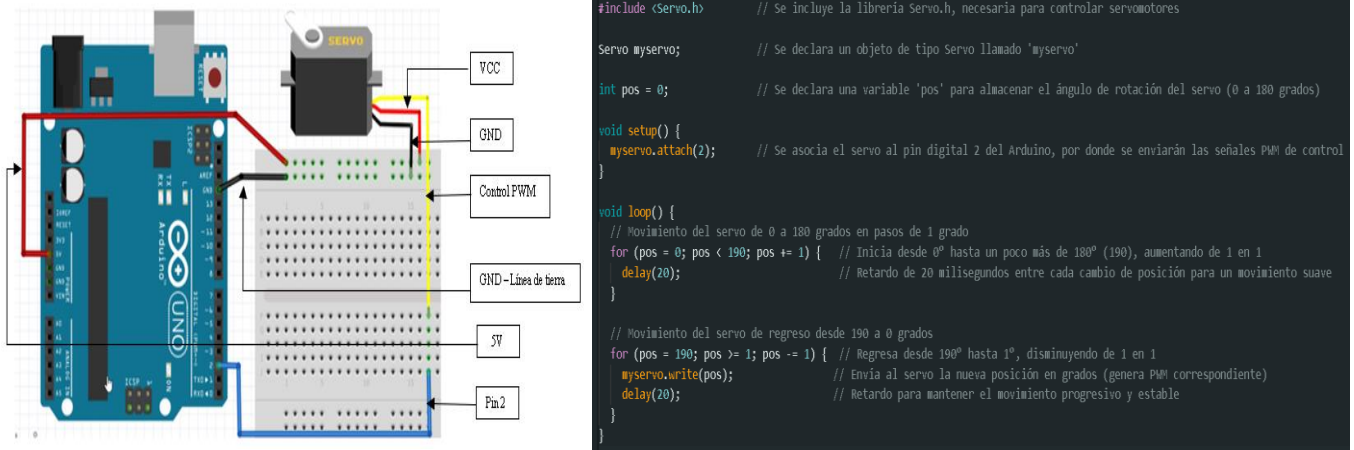
void setup() {
  Serial.begin(9600);   // Inicia la comunicación serial a 9600 baudios para enviar datos al monitor serial
  pinMode(PIN, INPUT);  // Configura el pin 2 como entrada digital (lectura de señal del sensor)
}

void loop() {
  levelSensor = digitalRead(PIN); // Lee el estado actual del pin 2 (sensor activado o desactivado)
  Serial.println(levelSensor);     // Imprime el valor leído (0 o 1) en el monitor serial
}
```

Nota. Código para la lectura de sensores en Arduino Uno, con los valores mostrados en el puerto serial replicando el mismo proceso a otros sensores, considerando actuadores para las compuertas de riego.

Fuente: Elaborado por el autor.

Seguidamente, como se muestra en la Figura 25, se deben configurar los actuadores, que para el caso de la maqueta didáctica planteada se utilizaran servomotores para poder abrir y cerrar las compuertas acordes a la distribución de riego requerida. El diagrama de conexión considera la integración del servomotor con la placa Arduino Uno a través de una protoboard. La señal de control se conecta al pin digital 2 del Arduino, desde donde se emiten señales PWM (modulación por ancho de pulso) que permiten posicionar el eje del motor con alta precisión. El conductor de alimentación se conecta a 5V, mientras que la línea de tierra se conecta al pin GND, garantizando una referencia común y el correcto funcionamiento del sistema. Así mismo podemos evidenciar el código completo en el Anexo 4.

Figura 25*Conexión y control básico de un servomotor con Arduino Uno*

Nota. Conexión básica de un servomotor a Arduino Uno y código para control angular de 0° a 180°.

Fuente: Elaborado por el autor.

Además, es importante considerar que la librería Servo.h debe estar instalada previamente desde el gestor de librerías del IDE de Arduino para habilitar el control del actuador. También se deben definir adecuadamente los ángulos de apertura y cierre, en función de la ubicación del servomotor y la distribución de las compuertas, por lo que estos parámetros deben ser calibrados cuidadosamente para asegurar una operación sincronizada con los sensores previamente instalados.

3.4.6.1 Cálculo de resistencia de entrada digital

Para garantizar una lectura digital estable del sensor de nivel en el microcontrolador, se conecta una resistencia pull-down entre la entrada digital de Arduino y la tierra (GND). Esta resistencia garantiza que el pin mantenga un estado lógico definido cuando el sensor está

abierto, evitando lecturas erráticas causadas por estados flotantes como se muestra en la Ecuación (12) ya que puede modelarse como un divisor de voltaje.

$$V_{IN} = V_{CC} \cdot \frac{R_{pull}}{R_{sensor} + R_{pull}} \quad (12)$$

Donde:

V_{IN} : Voltaje de entrada al pin digital

V_{CC} : Voltaje de alimentación (5V)

R_{pull} : Resistencia pull-down (ohmios)

R_{sensor} : Resistencia del sensor (0 Ω cuando está cerrado).

Sustituyendo los valores:

$$V_{IN} = 5 \cdot \frac{10.000}{0 + 10.000}$$

$$V_{IN} = 5 \cdot \frac{10.000}{10.000}$$

$$V_{IN} = 5 \text{ V}$$

En el caso de un sensor de tipo flotador que funciona como interruptor, se adopta un valor estándar de 10 k Ω , suficiente para mantener un estado Low o bajo cuando el sensor está abierto, sin interferir con el flujo de corriente cuando el sensor está activo.

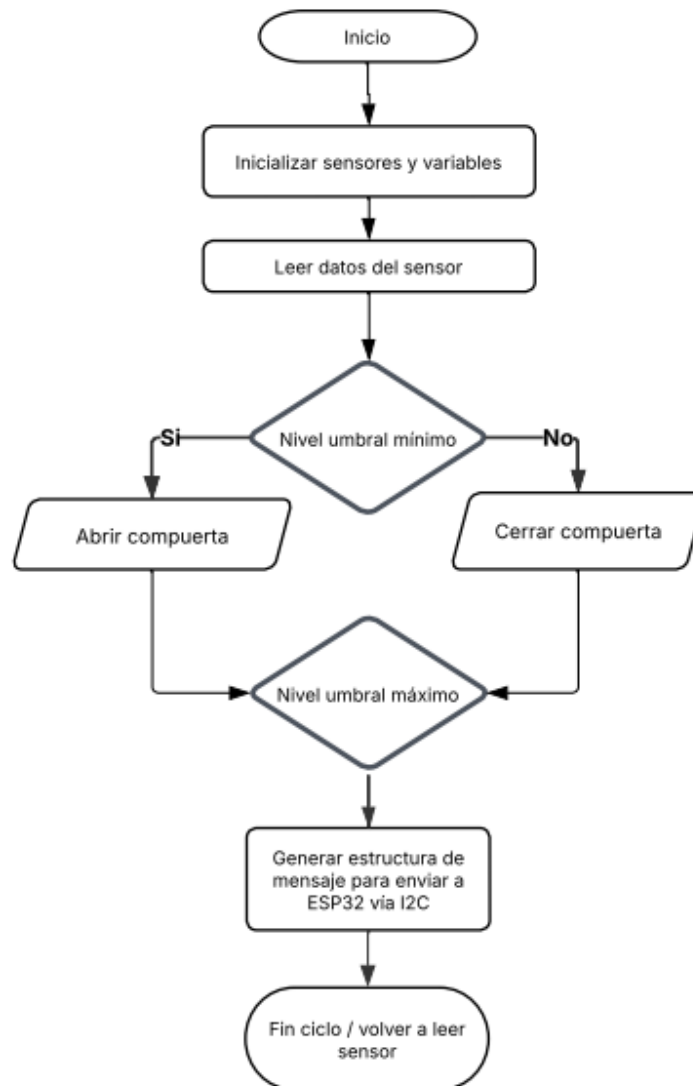
3.4.6.2 Procesamiento local de datos en nodos esclavos

Una vez adquirida la lectura del señor de nivel de agua, el microcontrolador Arduino Uno ejecuta un procesamiento lógico local. Esta lógica compara el valor leído con los umbrales mínimos y máximos previamente definidos, lo que permite decidir si la compuerta debe abrirse, cerrarse o mantenerse en su estado actual. Además, el nodo esclavo estructura un mensaje con

el estado del canal y lo prepara para ser enviado al microcontrolador maestro ESP32 mediante el protocolo I2C como ilustra en la Figura 26.

Figura 26

Diagrama de flujo del procesamiento local en nodos esclavos



Nota. El Arduino Uno evalúa la lectura del sensor de nivel, ejecuta la lógica de control de compuertas, y prepara un mensaje estructurado para ser enviado al nodo maestro vía I2C.

Fuente: Elaborado por el autor.

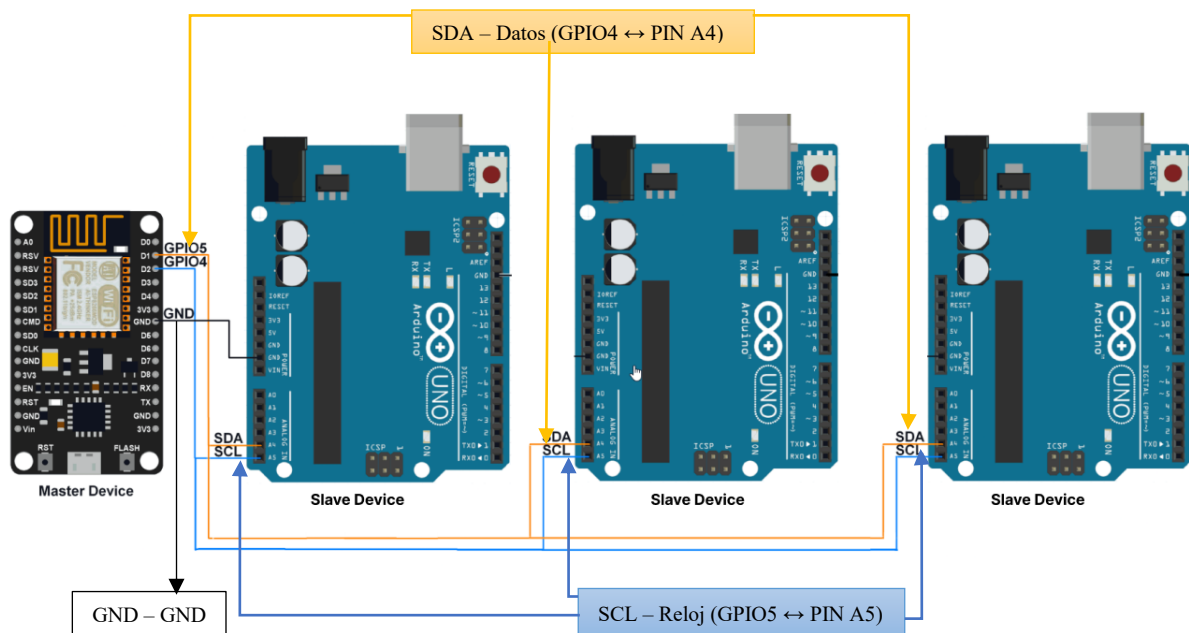
3.4.7 Comunicación y recolección de datos

Una vez se puede detectar los estados de los sensores de manera individual por cada uno de los Arduinos, se debe centralizar dichos datos para la toma de decisiones en base a los valores obtenidos. En esta etapa de comunicación se utilizará el protocolo I2C entre los 3 Arduinos, designados como nodos esclavos y el ESP32 designado como controlador central y Maestro en la arquitectura del protocolo, el cual se encarga de establecer la señal de reloj, iniciar y detener la comunicación conforme a lo establecido en el código.

En la Figura 27, se muestra la interacción entre los dispositivos los cuales a través de los pines SDA y SCL se sincronizan para actuar como un solo sistema en conjunto. Estos pines en cada Arduino corresponden a los pines analógico A4 para la señal SDA y el pin analógico A5 para la señal SCL, para el caso del ESP32 se utiliza los pines denotados físicamente como D1 para la señal SDA y D2 para la señal SCL.

Figura 27

Comunicación I2C entre ESP32 y múltiples Arduino uno



Nota. Interconexión I2C utilizando SDA y SCL para sincronizar la lectura de sensores desde nodos esclavos hacia un controlador maestro ESP32 con MicroPython.

Fuente: Elaborado por el autor.

En el apartado del controlador central (ESP32), se utiliza el firmware de MicroPython el cual es una implementación ligera y optimizada para el uso en microcontroladores del lenguaje de programación Python. MicroPython cuenta con características avanzadas como un prompt interactivo, enteros de precisión arbitraria, cierres, comprensión de listas, generadores, manejo de excepciones y mucho más. Sin embargo, es lo suficientemente compacto como para caber y ejecutarse en sólo 256k de espacio de código y 16k de RAM. Una de las ventajas de utilizar este tipo de firmware es su amplio y generalizado soporte el cual permite integrar características como el protocolo I2C basado en software.

Para poder establecer la comunicación a través del protocolo I2C se debe incluir la librería Wire.h en los nodos sensores que utilizan Arduino, además, se debe establecer una dirección única del dispositivo mediante las cuales se realizara la comunicación para el control y recepción de los datos a enviar. En la Figura 28, se muestra parte de la estructura básica utilizada en cada uno de los nodos sensores. Mediante esta configuración se debe ajustar el tamaño de los datos enviados para así optimizar el uso del canal de comunicación.

Figura 28*Código base de nodo esclavo para comunicación I2C con Arduino*

```

1  #include <Wire.h>           // Librería para comunicación I2C
2
3  #define SLAVE_ADDRESS 0x08  // Dirección del esclavo (misma que definiste en el otro Arduino)
4  char receivedData[32];      // Arreglo para almacenar los datos recibidos del esclavo
5
6  void setup() {
7      Wire.begin();           // Inicia el dispositivo como maestro I2C
8      Serial.begin(115200);    // Comunicación serial para visualizar en el monitor
9      Serial.println("Maestro I2C listo");
10 }
11
12 void loop() {
13     Wire.requestFrom(SLAVE_ADDRESS, 32); // Solicita hasta 32 bytes desde el esclavo
14
15     int i = 0;
16     while (Wire.available() && i < sizeof(receivedData) - 1) {
17         receivedData[i] = Wire.read();    // Lee cada byte recibido
18         i++;
19     }
20     receivedData[i] = '\0';               // Termina la cadena con carácter nulo
21
22     Serial.print("Datos recibidos: ");
23     Serial.println(receivedData);         // Muestra los datos recibidos
24
25     delay(1000);                          // Espera 1 segundo antes de la siguiente solicitud
26 }

```

Nota. Inicialización del esclavo I2C con dirección única y envío periódico de datos procesados desde un sensor analógico.

Fuente: Elaborado por el autor.

Para la programación del nodo maestro ESP32 se empleó MicroPython, una implementación ligera del lenguaje Python orientada a microcontroladores. Por defecto se utilizan los puertos 21 y 22 para las líneas SDA y SCL; sin embargo, dichos puertos pueden modificarse, ya que el protocolo I2C está implementado a nivel de software, lo cual permite una distribución más organizada de la conexión del hardware. En la Figura 29 se muestra el fragmento de código en MicroPython utilizado para la lectura y decodificación de datos provenientes de los nodos esclavos.

Figura 29*Lectura y decodificación de datos I2C en ESP32 utilizando MicroPython*

```

1  from machine import Pin, I2C          Import "machine" clases para controlar pines e interfaz I2C
2  import time                          # Importa funciones para pausas temporales
3
4  # Configura el bus I2C: canal 0, SCL en GPIO22, SDA en GPIO21, frecuencia 100 kHz
5  i2c = I2C(0, scl=Pin(22), sda=Pin(21), freq=100000)
6
7  # Dirección I2C del Arduino esclavo (debe coincidir con la definida en el esclavo)
8  SLAVE_ADDRESS = 0x08
9
10 # Función que solicita y procesa los datos del esclavo
11 def leer_datos():
12     try:
13         # Solicita 26 bytes desde el esclavo en la dirección indicada
14         datos = i2c.readfrom(SLAVE_ADDRESS, 26)
15
16         # Procesa los primeros 2 bytes como entero (little-endian)
17         valor_sensor = int.from_bytes(datos[0:2], 'little')
18
19         # Procesa los siguientes 4 bytes como float [temperatura]
20         temperatura = float.from_bytes(datos[2:6], 'little')
21
22         # Resto de bytes (20) como mensaje tipo string
23         mensaje = datos[6:26].decode('utf-8').strip('\x00')
24
25         # Muestra los valores en el monitor serial
26         print(f"Valor sensor: {valor_sensor}")
27         print(f"Nivel: {temperatura:.2f}°C")
28         print(f"Mensaje: {mensaje}")
29         print("-----")
30
31     except Exception as e:
32         # Captura y muestra errores de comunicación
33         print(f"Error en la comunicación: {e}")
34
35 # Bucle principal que lee datos cada 1 segundo
36 while True:
37     leer_datos()
38     time.sleep(1)

```

Nota. Código en MicroPython para recibir y decodificar datos desde un nodo I2C esclavo mediante pines SDA (21) y SCL (22).

Fuente: Elaborado por el autor.

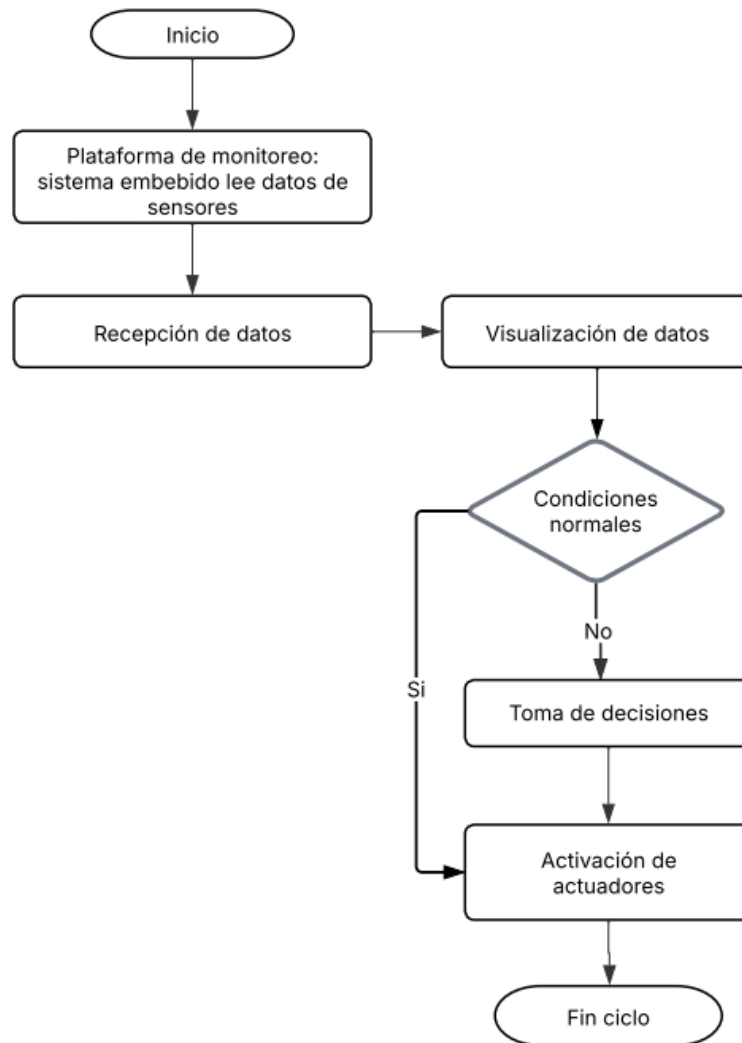
3.4.7.1 Interacción entre plataforma, visualización, decisiones y activación

Básicamente, el diagrama integra los cuatro bloques descritos previamente, mostrando de forma consolidada la secuencia operativa del sistema. La Figura 30 representa la plataforma de monitoreo, continúa con la visualización de datos, prosigue con la toma de decisiones y

culmina con la activación de los actuadores. Este esquema evidencia cómo los módulos interactúan de manera encadenada para garantizar el correcto funcionamiento de la automatización de canales de riego.

Figura 30

Secuencia integrada de monitoreo y control del sistema



Nota. Representación de la secuencia integrada de los bloques 4–7: monitoreo, visualización, toma de decisiones y activación de actuadores en la automatización de canales de riego.

Fuente: Elaborado por el autor.

Una vez se establece el proceso de recolección de los datos y la coordinación del sensado a través del ESP32, es necesario implementar el bloque de notificación para lo cual se

establece un ChatBot mediante Telegram en el cual se enviarán las notificaciones de los diferentes estados de apertura, cierre y riego del terreno. Mediante este chat, en el cual se deben incluir a los usuarios, propietarios o encargados permitirá conocer el estado actual del sistema en cada iteración.

Para la creación del Chatbot inicialmente se debe crear el token de acceso a través de BotFather que es la implementación oficial de Telegram para la creación y gestión de bots en la plataforma. La función principal del Bot es actuar como sistema de notificación por lo cual únicamente se implementa el método de envío de mensajes al grupo establecido como se muestra en la Figura 31, donde se debe especificar el cuerpo del mensaje en formato JSON, junto al ID de Grupo creado. De igual manera se muestra la conexión previa a una red inalámbrica de prueba para la conexión a internet y el envío de peticiones.

Figura 31

Búsqueda e ingreso al bot oficial BotFather en Telegram



Nota. Se muestra la búsqueda del usuario @BotFather desde una cuenta activa de Telegram, punto de partida para la creación de un nuevo bot a través de comandos predefinidos.

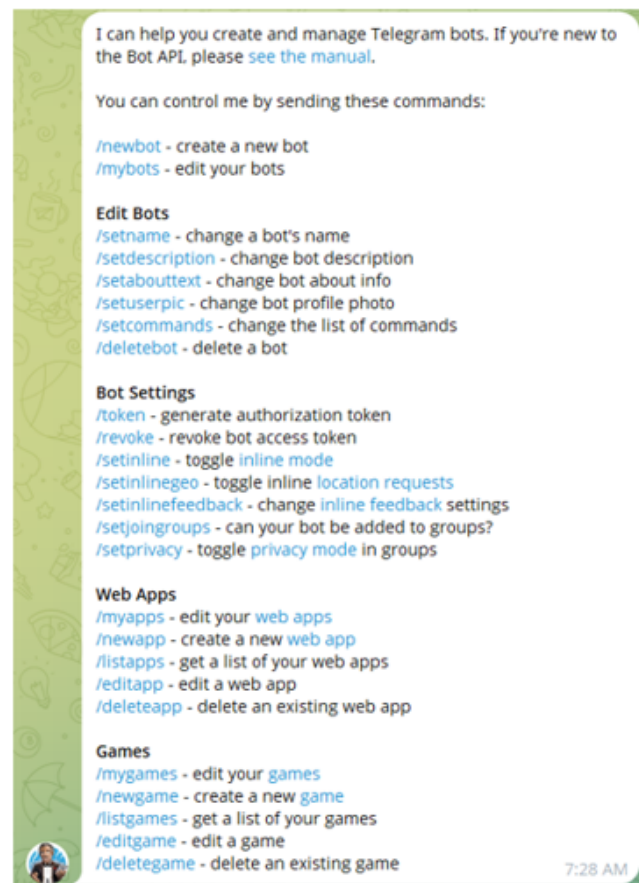
Fuente: Elaborado por el autor.

Para poder generar el chat BOT se lo hace desde una cuenta activa de Telegram, se debe buscar el usuario @BotFather, el cual a su vez es un gestor de bots proporcionado por la

plataforma. Como se muestra en la Figura 32, se debe iniciar el chat y especificar el comando /start para ver las opciones disponibles, para poder crear uno nuevo se utiliza la opción /newbot a través de la cual se asigna un nombre único al nuevo Bot que se desea generar. De igual manera se tiene disponibles varias opciones para el cambio de nombre, descripción y establecimiento de comandos disponibles para los usuarios a partir de los cuales se programa la funcionalidad requerida.

Figura 32

Comandos disponibles en BotFather para gestión de bots



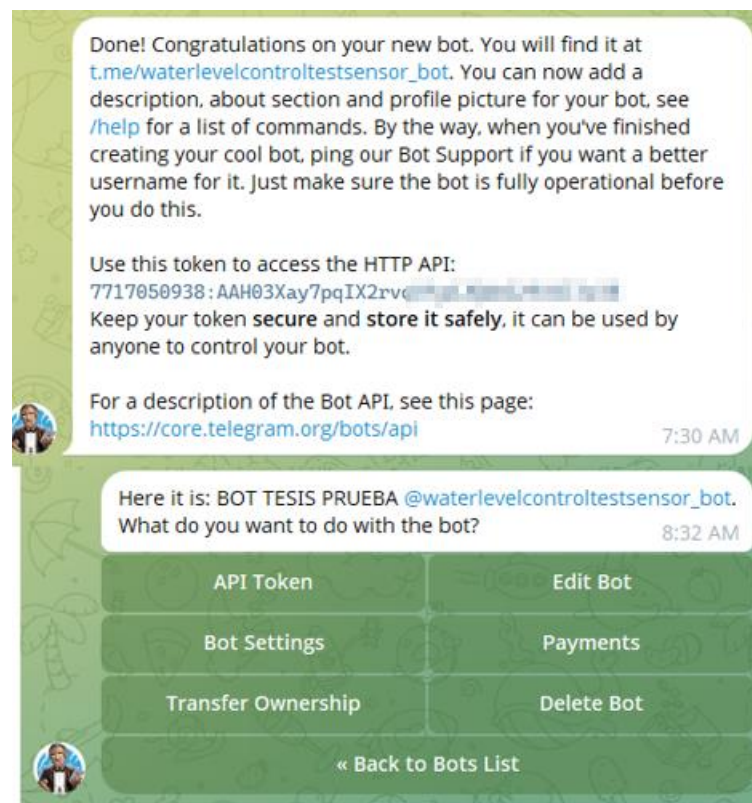
Nota. La figura presenta el conjunto de comandos ofrecidos por BotFather, entre ellos /newbot para la creación de un nuevo bot, y opciones como /setname, /setdescription, y /setcommands para configurar su comportamiento.

Fuente: Elaborado por el autor.

Una vez se finaliza el proceso de creación se genera un token de acceso el cual se debe utilizar para el despliegue del Bot a través de Micropython como muestra la Figura 33. Este código nos permite enviar y recibir mensajes del Bot de acuerdo con su programación mediante la API de Telegram, tal como se ejemplifica en la Figura 34.

Figura 33

Token de autenticación del bot de Telegram para integración con MicroPython



Nota. Se puede evidenciar el token generado por BotFather, necesario para integrar el bot con el sistema mediante MicroPython y la API de Telegram.

Fuente: Elaborado por el autor.

La Figura 34 muestra una versión simplificada del proceso de notificación a través de Telegram desde el microcontrolador ESP32, programado en MicroPython. Permitiendo validar el envío básico de mensajes, pero no representa la lógica final del sistema, para la implementación definitiva, se desarrolló un código más robusto que incluye la lectura y

evaluación de múltiples sensores, el control automatizado de compuertas, y la interacción con usuarios mediante comandos personalizados enviados al bot de Telegram. Este código se presenta íntegramente en el Anexo 3.

Figura 34

Envío de notificaciones desde ESP32 a Telegram mediante MicroPython

```

1  import network          # Manejo de conexión WiFi
2  import urequests        # Envío de solicitudes HTTP
3  import time            # Control de tiempo y retardos
4  from machine import Pin  # Manejo de pines digitales
5
6  # ----- CONFIGURACIÓN DE RED Wi-Fi -----
7  WIFI_SSID = "Red Prueba"      # Nombre de la red Wi-Fi
8  WIFI_PASSWORD = "12345678"    # Contraseña de la red Wi-Fi
9
10 # ----- CONFIGURACIÓN DEL BOT DE TELEGRAM -----
11 BOT_TOKEN = "ABCdefGHIjKlMnOPqRStUvWxYz-123456AbCdE" # Token del bot de Telegram
12 GROUP_CHAT_ID = "1234567890" # ID del grupo o usuario de Telegram
13
14 # ----- SENSOR DE NIVEL DE AGUA -----
15 sensor_nivel = Pin(34, Pin.IN) # Define el pin 34 como entrada digital (para sensor de nivel tipo flotador)
16
17 # ----- CONEXIÓN A Wi-Fi -----
18 def connect_wifi():
19     sta_if = network.WLAN(network.STA_IF) # Configura interfaz Wi-Fi como estación
20     if not sta_if.isconnected():
21         print("Conectando a Wi-Fi...")
22         sta_if.active(True) # Activa el Wi-Fi
23         sta_if.connect(WIFI_SSID, WIFI_PASSWORD) # Intenta conectar con las credenciales
24         while not sta_if.isconnected(): # Espera conexión
25             time.sleep(0.5)
26     print("Conectado a Wi-Fi:", sta_if.ifconfig()) # Muestra IP asignada
27
28 # ----- ENVÍO DE MENSAJE A TELEGRAM -----
29 def send_to_telegram(message):
30     url = "https://api.telegram.org/bot{}/sendMessage".format(BOT_TOKEN)
31     payload = {
32         "chat_id": GROUP_CHAT_ID,
33         "text": message,
34         "parse_mode": "HTML"
35     }
36     try:
37         response = urequests.post(url, json=payload) # Envía el mensaje vía HTTP POST
38         response.close()
39     except Exception as e:
40         print("Error al enviar mensaje:", e)
41
42 # ----- PROGRAMA PRINCIPAL -----
43 connect_wifi() # Conexión inicial a Wi-Fi
44
45 estado_anterior = -1 # Guarda el estado anterior del sensor
46
47 while True:
48     estado_actual = sensor_nivel.value() # Lee el valor lógico del sensor (0 o 1)
49
50     # Si hay un cambio de estado (por ejemplo, de seco a nivel alto o viceversa)
51     if estado_actual != estado_anterior:
52         if estado_actual == 1:
53             mensaje = "🔵 <b>Alerta:</b> El canal ha alcanzado el <b>nivel máximo</b> de agua."
54         else:
55             mensaje = "🟡 <b>Alerta:</b> El canal está por <b>debajo</b> del nivel requerido<b>.</b>"
56         send_to_telegram(mensaje) # Envía mensaje al grupo
57         estado_anterior = estado_actual # Actualiza el estado anterior
58
59     time.sleep(5) # Espera 5 segundos antes de volver a comprobar

```

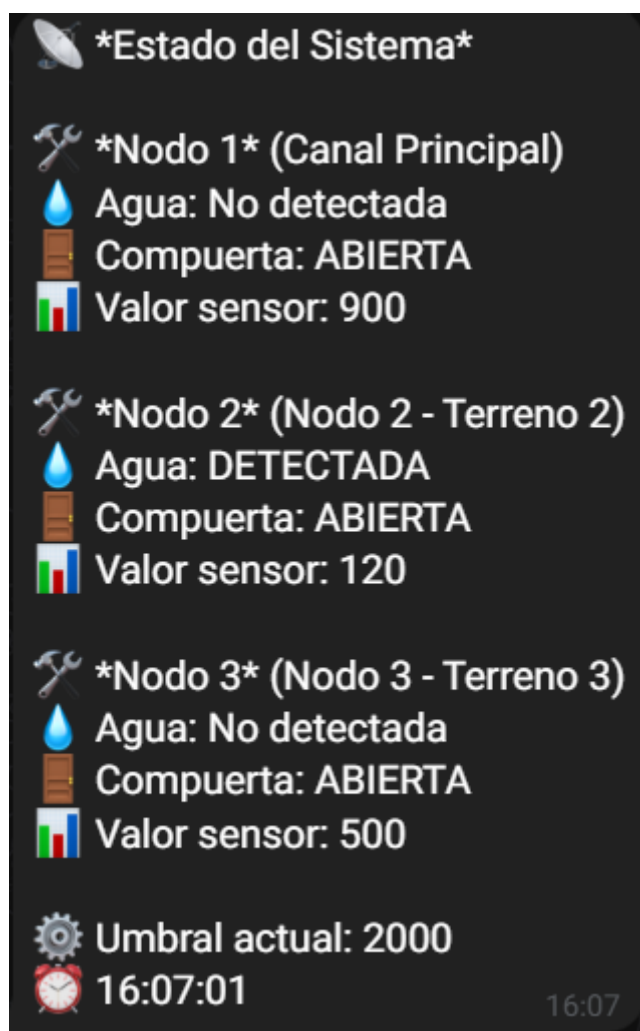
Nota. Código para conectar el ESP32 a WiFi y enviar mensajes formateados en JSON a un grupo de Telegram usando un bot.

Fuente: Elaborado por el autor.

Desde el punto de vista de los usuarios las notificaciones les permiten conocer el estado actual del riego sobre su terreno, y en base al orden establecido verificar en cuanto tiempo les corresponderá el próximo ciclo de riego. En la Figura 35 se ejemplifican los mensajes generados por el sistema a través del bot de Telegram.

Figura 35

Mensajes del sistema de riego enviados por el bot de Telegram



Nota. Notificaciones automáticas enviadas por el ESP32 a través de telegram informan el estado de compuertas y ciclos de riego en tiempo real.

3.5 Implementación

En la Figura 36 se presenta el diagrama de integración final del sistema IoT propuesto, donde se unifican todos los bloques de hardware y software que conforman el prototipo. El proceso inicia con la adquisición de datos mediante los sensores de nivel de agua, los cuales se encuentran conectados a los nodos esclavos Arduino Uno. Estos realizan un procesamiento local y envían la información al nodo maestro ESP32 a través del protocolo de comunicación I2C.

El ESP32 cumple la función de centralizar y validar los datos, además de ejecutar la lógica de control para el manejo de los servomotores encargados de abrir o cerrar las compuertas de los canales de riego. Posteriormente, la información procesada es publicada en el Broker MQTT, el cual actúa como intermediario de la comunicación en tiempo real.

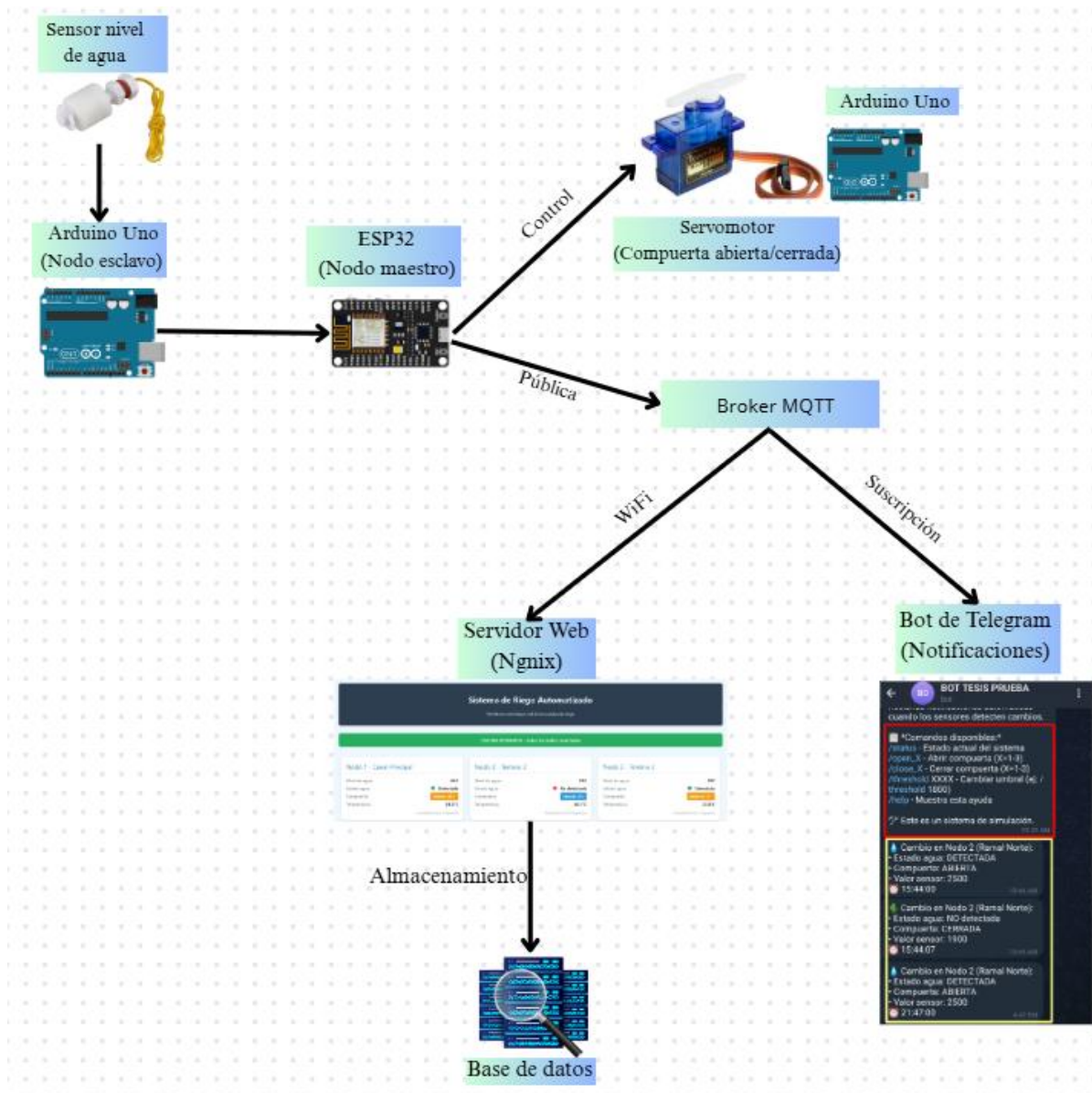
A partir de este broker, se distribuyen los datos hacia dos destinos principales:

- El Servidor Web implementado en Nginx, que permite la visualización de datos y gestión del sistema mediante una interfaz de monitoreo se encuentra enlazado a una base de datos, donde se almacena registros para su análisis posterior.
- El Bot de Telegram, que recibe notificaciones automáticas con el estado de los canales de riego, permitiendo que el usuario final tenga un acceso inmediato a las alertas y condiciones del sistema.

De esta manera, el diagrama integra de forma clara los módulos de adquisición, procesamiento, comunicación, control, almacenamiento y notificación, garantizando una visión global del funcionamiento del sistema IoT en su conjunto.

Figura 36

Diagrama de integración del sistema



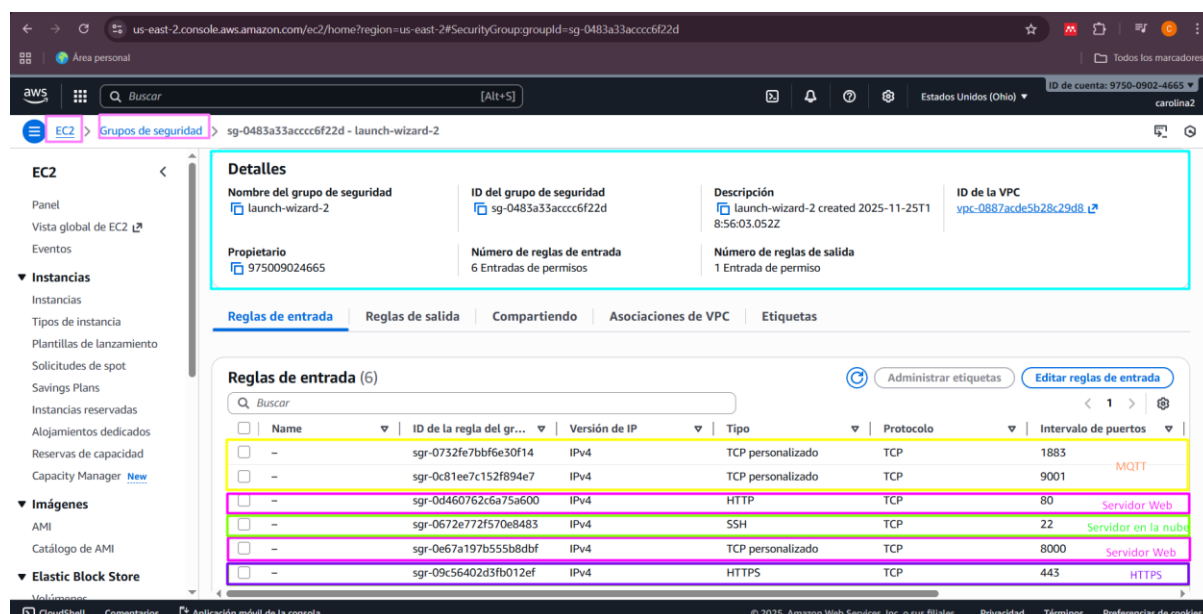
Nota. El diagrama muestra la interacción entre los nodos esclavos (Arduino Uno) con sensores de nivel de agua, el nodo maestro (ESP32), los servomotores de control de compuertas y el sistema de notificaciones mediante Telegram. Todos los módulos operan de forma integrada a través comunicación serial permitiendo la automatización del sistema de riego en ambiente controlado.

Fuente: Elaborado por el autor.

Adicionalmente, para garantizar el funcionamiento del sistema IoT en un entorno accesible y seguro, los servicios de comunicación y visualización fueron desplegados en un servidor alojado en la nube. En este servidor se habilitaron los puertos necesarios para el acceso al broker MQTT, la interfaz web y los canales de administración del sistema. La Figura 37 muestra la configuración del grupo de seguridad y los puertos utilizados en la plataforma cloud.

Figura 37

Configuración del servidor cloud y reglas de seguridad del sistema IoT



Nota. La figura muestra la instancia cloud empleada para el despliegue del sistema IoT, en la cual se encuentran configurados el broker MQTT, el servidor web y los canales de administración. Asimismo, se detallan los puertos de acceso necesarios para garantizar la comunicación entre los nodos locales y la plataforma en la nube.

Fuente: Adaptado de (AWS, 2025).

3.5.1 Interfaz Web para la visualización de datos

Para una mejor visualización de los datos recibidos por los sensores, se realizó una página web desarrollada mediante HTML y CSS, cuyo código completo se presenta en el Anexo 5. Esta plataforma permite al usuario acceder a la información en tiempo real sobre el

nivel de agua y estado de las compuertas en cada nodo de monitoreo. Además, el sistema muestra advertencias en caso de que alguno de los nodos no responda, lo que mejora la capacidad de supervisión remota y facilita la toma de decisiones como se muestra en la Figura 38.

Figura 38

Interfaz web del sistema de riego y fragmento del código en HTML/CSS

Sistema de Riego Automatizado
Monitoreo en tiempo real de los canales de riego

SISTEMA OPERATIVO - Todos los nodos conectados

Nodo 1 - Canal Principal
Nivel de agua: 642
Estado agua: Detectada
Compuerta: Abierta (85%)
Temperatura: 24.5°C
Actualizado hace 5 segundos

Nodo 2 - Terreno 2
Nivel de agua: 312
Estado agua: No detectada
Compuerta: Cerrada (0%)
Temperatura: 26.1°C
Actualizado hace 7 segundos

Nodo 3 - Terreno 3
Nivel de agua: 587

ADVERTENCIA - Nodo 2 no responde

```

<html lang="es">
<head>
<style>
.btn {
}
.footer {
font-size: 0.8rem;
margin-top: 10px;
color: gray;
}
</style>
</head>
<body>
<div class="header">
<h1>Sistema de Riego Automatizado</h1>
<p>Monitoreo en tiempo real de los canales de riego</p>
</div>
<div class="warning">
<div class="container">
<div class="card">
<h3>Nodo 1 - Canal Principal</h3>
<p><strong>Nivel de agua:</strong> 642</p>
<p class="status">Estado agua: <span><strong>No detectada</strong></span></p>
<p><strong>Compuerta:</strong> <button class="btn">Cerrada (0%)</button></p>
<p><strong>Temperatura:</strong> 24.5°C</p>
<p class="footer">Actualizado hace 17 segundos</p>
</div>
<div class="card">
<h3>Nodo 2 - Terreno 2</h3>
<p><strong>Nivel de agua:</strong> 312</p>
<p class="status">Estado agua: <span><strong>No detectada</strong></span></p>

```

Nota. Representación de la plataforma web y visualización de código que genera la visualización de los nodos.

Fuente: Elaborado por el autor.

4 Capítulo IV: Análisis de resultados

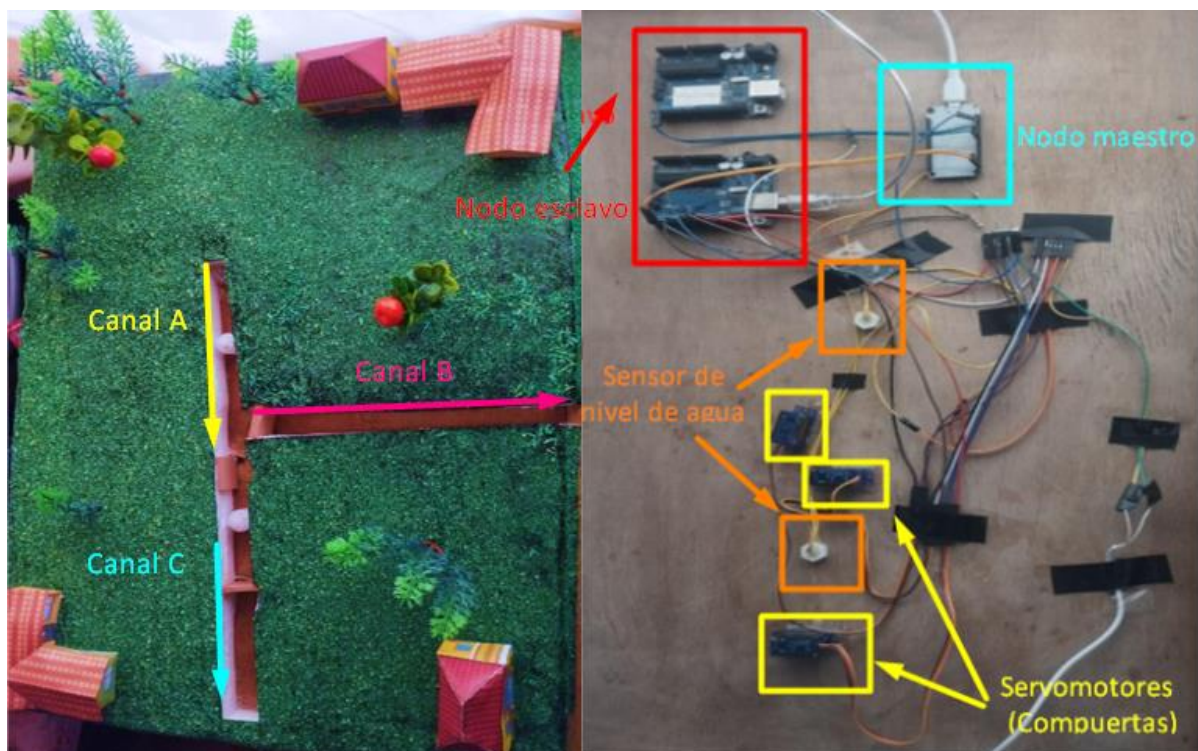
En este capítulo se detallará el desarrollo del proyecto, evaluando la eficiencia del sistema mediante diversas pruebas que permitirán determinar en qué condiciones el funcionamiento es adecuado y en qué aspectos se requiere mejorar para así garantizar que el prototipo pueda operar de manera confiable, segura y facilitando su uso como herramienta práctica para los beneficiarios del sistema de riego.

4.1 Diseño de la maqueta y ubicación de los sensores

En esta sección se desarrolló una representación a escala del canal real, empelando materiales resistentes al agua que facilitaron la fijación de los sensores y servomotores como se evidencia en la Figura 39 se instalaron sensores de nivel en puntos estratégicos a lo largo del canal, identificando las zonas clave para el monitoreo. A su vez, se instalaron servomotores para las compuertas, permitiendo simular la distribución de agua a los diferentes puntos de uso.

Figura 39

Diseño físico de la maqueta y distribución de sensores.



Nota. Demostración del armado de la maqueta y las conexiones realizadas para su correcto funcionamiento.

Fuente: Elaborado por el autor.

4.1.1 Compuerta

Las compuertas fueron diseñadas para replicar el control de flujo de agua hacia los distintos sectores de riego de los canales reales. Se emplearon compuertas móviles de dimensiones reducidas, fabricadas con materiales plásticos livianos y resistentes al agua, lo que permitió su instalación y operación eficiente durante las pruebas como se muestra en la Figura 40.

Estas se ubicaron en las salidas principales del canal y se conectaron a los servomotores, posibilitando su activación automática según los niveles de agua detectados por los sensores.

Figura 40

Compuertas colocadas en el canal de riego



Nota. Representación de las compuertas que serán usadas en el proyecto.

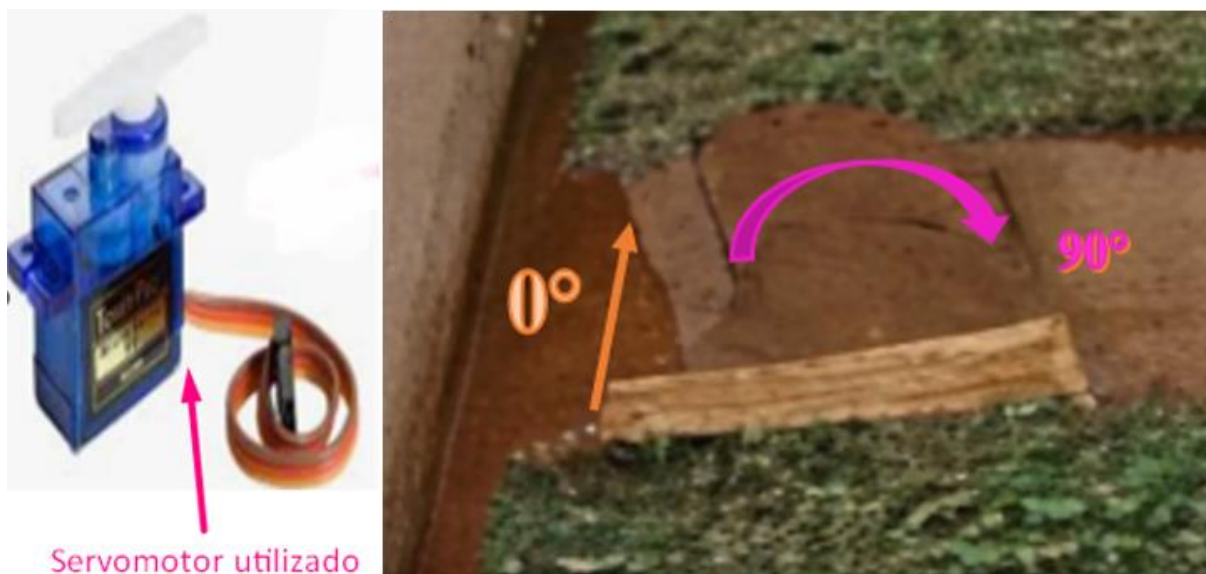
Fuente: Elaborado por el autor.

4.1.2 *Servos*

Se utilizaron servomotores rotativos presentes en la Figura 41, seleccionados por su precisión angular y bajo consumo. Cada uno se acopló a una compuerta, controlando su apertura mediante señal PWM del microcontrolador. Durante las pruebas, respondieron eficazmente a las órdenes del nodo maestro (ESP32), garantizando tiempos de reacción rápidos y estables ante las variaciones de nivel detectadas por los sensores.

Figura 41

Servomotores rotativos para las compuertas



Nota. Representación del movimiento de las compuertas.

Fuente: Elaborado por el autor.

4.1.3 *Distribución y Armado*

La distribución de los componentes se dispuso conforme a la estructura del canal de riego del barrio Río Frío. Definiendo un canal principal con canales secundarios para simular

la distribución de agua hacia las parcelas, los sensores de nivel se instalaron en puntos estratégicos para monitorear zonas críticas, los servomotores y compuertas se ubicaron en las salidas de los canales secundarios, conectados a placas Arduino Uno como nodos esclavos. El sistema se ensambló sobre una base resistente y de fácil acceso, asegurando conexiones eléctricas seguras y pruebas repetitivas sin comprometer la estructura.

4.2 Pruebas

En la presente sección, se detallan las pruebas realizadas para validar el correcto funcionamiento del proyecto. Estas pruebas abarcan aspectos fundamentales del sistema, Arquitectura, Stakeholdres e integración completa del sistema con el fin de asegurar que el sistema cumple con los requerimientos establecidos y funciona de manera eficiente en diversos escenarios de uso.

4.2.1 Validar requerimientos de Arquitectura

En esta ocasión en la Tabla 14 vamos a evaluar la parte del sistema eléctrico.

Tabla 14

Validación del sistema eléctrico

Test eléctrico	Subsistema de prueba	Eléctrico
	¿Prueba de despliegue?	Si/No

Descripción:

Prueba inicial del sistema enfocada en verificar la correcta alimentación eléctrica del prototipo. Se valida que el sistema opere dentro del rango de voltaje de 5V a 12V (SRSH21), que la conexión eléctrica proteja los componentes frente a cortocircuitos y polaridad inversa (SRSH23 y SRSH25). Además, se verifica el funcionamiento del

regulador de voltaje que asegura la alimentación constante de los sensores (SRSH24). Esta prueba garantiza que todos los componentes operen de manera estable sin interrupciones.

Prerrequisitos:

- a) Ensamblaje del circuito de alimentación con batería de litio 3.7V y módulo elevador de voltaje MT3608.
 - b) Conexión de salida de 5V regulada hacia ESP32 y sensores.
 - c) Disponibilidad de multímetro digital para verificación de voltaje y protección de picos.
-

Pasos:

- Verificación de conexión de los módulos de alimentación y regulador de voltaje.
 - Medición del voltaje de salida asegurando 5V constantes y protección ante sobrevoltaje.
 - Prueba de polaridad inversa para protección de componentes.
 - Verificación del encendido correcto de ESP32 y sensores.
-

Resultado esperado:

Todos los componentes deben encenderse correctamente y mantenerse estables, tomando en cuenta que el voltaje debe estar dentro del rango de operación (5V–12V) y la fuente debe proteger contra picos de corriente y polaridad inversa, garantizando un funcionamiento confiable de los sensores y el sistema completo.

Nota. Validación de los requerimientos de arquitectura basados en el sistema eléctrico.

Fuente: Elaborado por el autor.

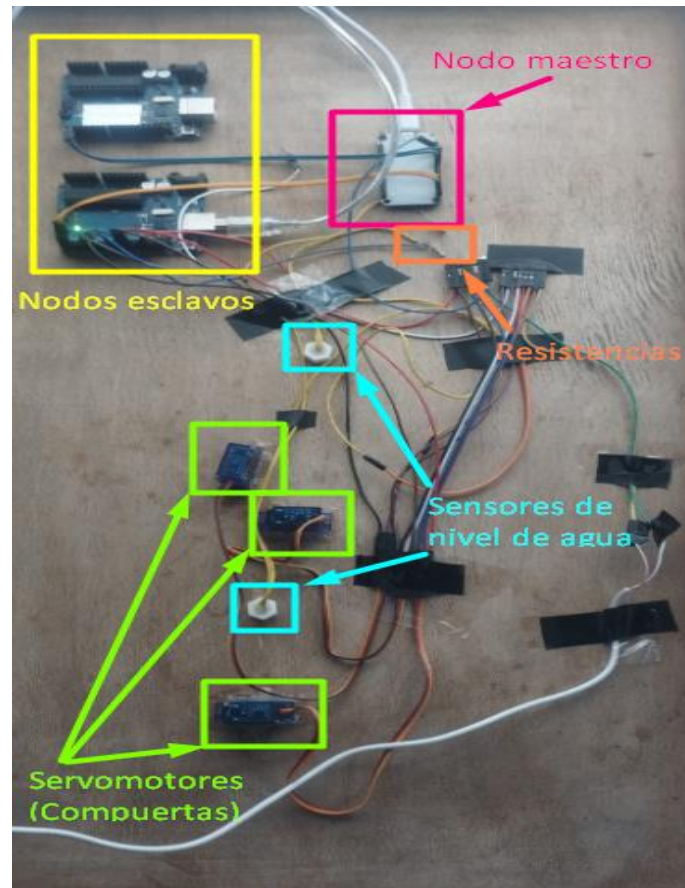
Resultado obtenido

El circuito de alimentación entrega un voltaje estable de $5.0V \pm 0.01V$ a los módulos y sensores, verificando la protección contra sobrevoltaje, cortocircuitos y polaridad inversa,

funcionando correctamente. Los sensores y ESP32 encienden y operan de manera continua sin interrupciones como se muestra en la Figura 42.

Figura 42

Validación de la arquitectura física del sistema IoT



Nota. Validación de la arquitectura implementada evidenciando la integración física de nodo maestro, nodos esclavos basados en el Arduino uno, los sensores de nivel de agua y servomotores encargados del accionamiento de las compuertas.

Fuente: Elaborado por el autor.

4.2.2 Test de hardware

En la Tabla 15 se presenta la validación de los requerimientos de hardware del sistema.

Tabla 15*Validación de hardware*

Test de hardware	Subsistema de prueba	Hardware
	¿Prueba de despliegue?	Si/No

Descripción:

Prueba destinada a verificar el correcto funcionamiento de los componentes de hardware, primero validamos que la placa microcontroladora sea compatible con el sensor de nivel de agua (SRSH11), el módulo de comunicación Wi-Fi funcione para transmisión de datos en tiempo real, los sensores cuenten con protección contra humedad (SRSH13) y que el sistema soporte la conexión simultánea de múltiples sensores (SRSH15). Esta prueba asegura que el prototipo pueda operar de manera confiable en condiciones controladas.

Prerrequisitos:

- Montaje completo del sistema de adquisición y transmisión de datos en el prototipo (microcontroladores, sensores y actuadores).
- Carga de los scripts en el microcontrolador ESP32.
- Establecimiento de conexión Wi-Fi para transmisión de datos en tiempo real.
- Plataforma web habilitada para monitoreo de los datos enviados.
- Conexión simultánea de todos los sensores instalados en los nodos de riego.

Pasos:

- Verificación de compatibilidad de la placa microcontroladora con los sensores de nivel de agua.
- Verificación de la correcta comunicación entre el circuito de obtención de datos y el circuito de transmisión de datos

Resultado esperado:

El microcontrolador debe registrar correctamente los datos de todos los sensores, el módulo Wi-Fi debe transmitir los datos sin interrupciones, los sensores deben mantener su integridad frente a humedad y todos los sensores conectados deben enviar datos simultáneamente sin errores.

Nota. Validación de los requerimientos del hardware del sistema.

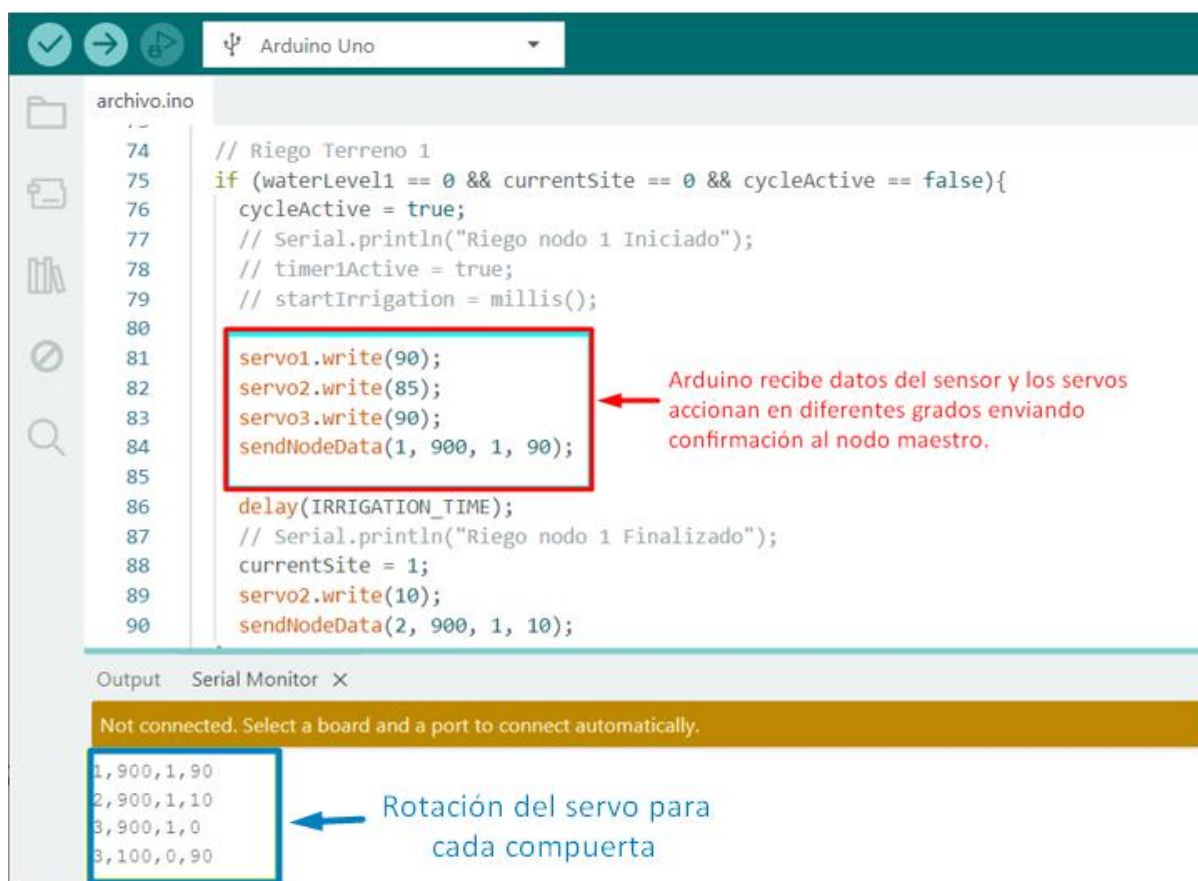
Fuente: Elaborado por el autor.

Resultado obtenido

Durante la prueba de validación de hardware, el prototipo cumplió con todos los requerimientos especificados: La placa microcontroladora funcionó correctamente con todos los sensores de nivel de agua conectados, el módulo de comunicación Wi-Fi transmitió datos en tiempo real hacia la plataforma web sin retrasos mostrando lecturas confiables y transmisión estable de datos como se muestra en la Figura 43.

Figura 43

Representación de nodo esclavo en Arduino, lectura de sensores y control de servomotores.



Nota. Representación del hardware del sistema con los Arduino y conexiones del sistema de control.

Fuente: Elaborado por el autor.

4.2.3 Test de Software

En la Tabla 16 se detallan los requisitos que van a ser evaluados del sistema de software.

Tabla 16

Demostración de la prueba de software

	Subsistema de prueba	Software
Test de software	¿Prueba de despliegue?	Si/No
Descripción:		

Se valida que el software lea automáticamente los datos de los sensores y los envíe a la plataforma, generando alertas automáticas al detectar valores críticos y que los datos de las pruebas se almacenen correctamente para su posterior análisis.

Prerrequisitos:

- a) Plataforma de monitoreo web o aplicación Android disponible y configurada.
 - b) Conexión de la placa microcontroladora y sensores al software de adquisición.
 - c) Scripts cargados correctamente en la microcontroladora para lectura y envío de datos.
 - d) Acceso a la base de datos configurado para registrar los datos.
-

Pasos:

- Verificación de lectura automática de los sensores por el software.
 - Transmisión de datos a la plataforma de monitoreo.
 - Generación de alertas automáticas al detectar valores críticos.
 - Acceso a la plataforma desde navegador web y dispositivo Android.
 - Revisión de almacenamiento de datos para posterior análisis.
-

Resultado esperado:

El software debe capturar correctamente los datos de todos los sensores, transmitirlos automáticamente a la plataforma, mostrar los niveles de agua de forma gráfica, generar alertas al detectar valores críticos, permitir acceso desde cualquier navegador o dispositivo Android y almacenar todos los datos de prueba de manera confiable.

Nota. Validación de los requerimientos del sistema de software.

Fuente: Elaborado por el autor.

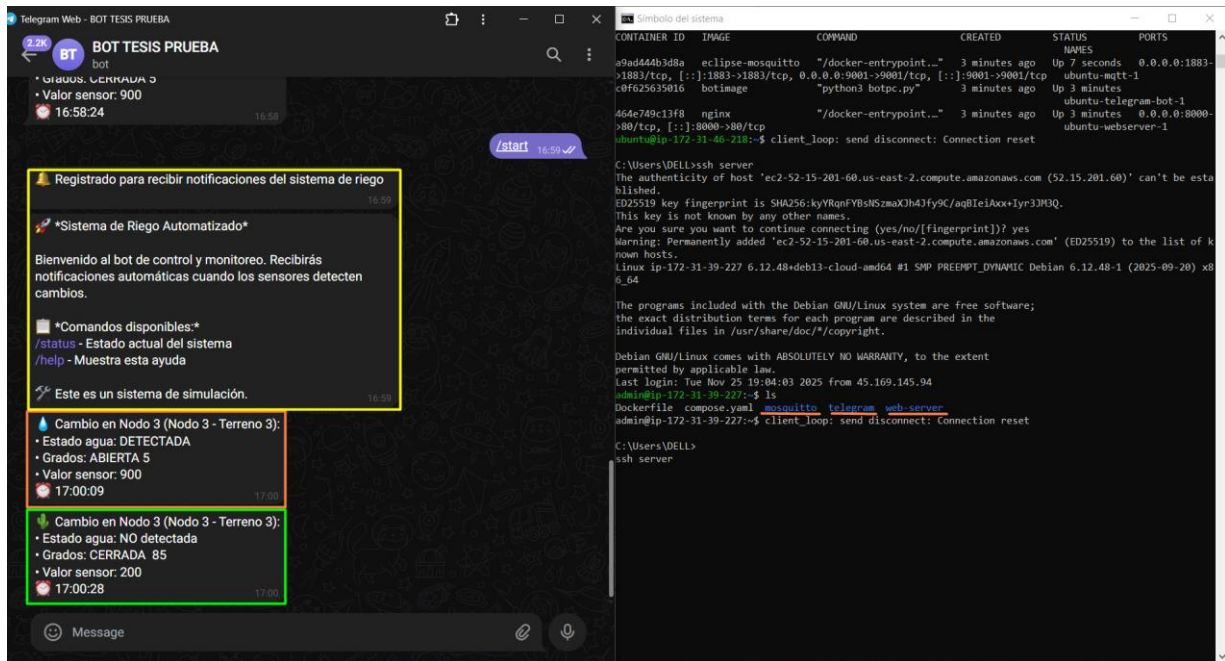
Resultado obtenido

El software del prototipo funcionó correctamente capturando y transmitiendo automáticamente los datos de todos los sensores a la plataforma de monitoreo, presentados en

tiempo real como se muestra en la Figura 44. Incluso se generaron alertas automáticas al detectar valores críticos, lo cual la plataforma fue accesible sin problemas desde navegadores web y dispositivos Android, garantizando la trazabilidad y la usabilidad del sistema.

Figura 44

Validación del sistema de comunicación MQTT y notificaciones por Telegram.



Nota. Demostración efectiva de las conexiones de los sensores y envío de datos a la plataforma de monitoreo.

Fuente: Elaborado por el autor.

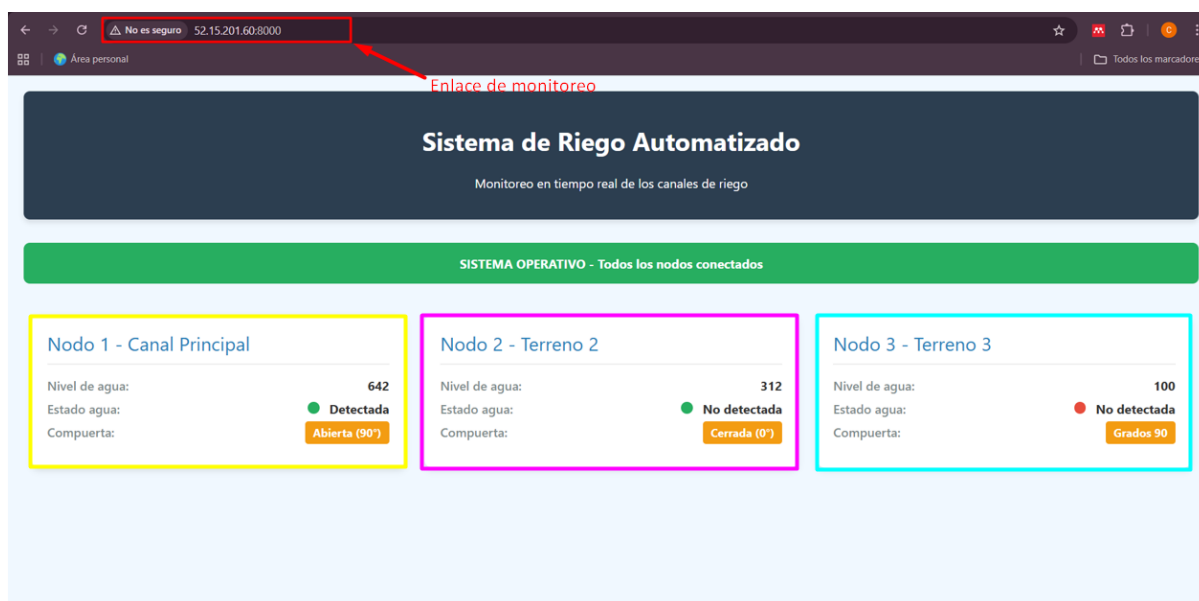
4.2.3.1 Pruebas de registro de usuario

Al sistema de monitoreo se puede acceder a través de diferentes medios. En primer lugar, el usuario puede ingresar directamente a la plataforma web mediante un enlace, permitiendo visualizar en tiempo real la información transmitida por los sensores. También es posible acceder a los datos a través de la aplicación Telegram, ya sea escaneando el código QR proporcionado o buscando el usuario configurado para el prototipo.

En la Figura 45 se muestra la interfaz de la página web del sistema mediante el enlace <http://52.15.201.60:8000/>. Esta plataforma permite visualizar en tiempo real los datos transmitidos por los sensores, ofreciendo al usuario una supervisión remota del nivel de agua y del estado de las compuertas facilitando la toma de decisiones oportunas y garantizando un control eficiente del sistema de riego.

Figura 45

Acceso al sistema de monitoreo mediante la plataforma web



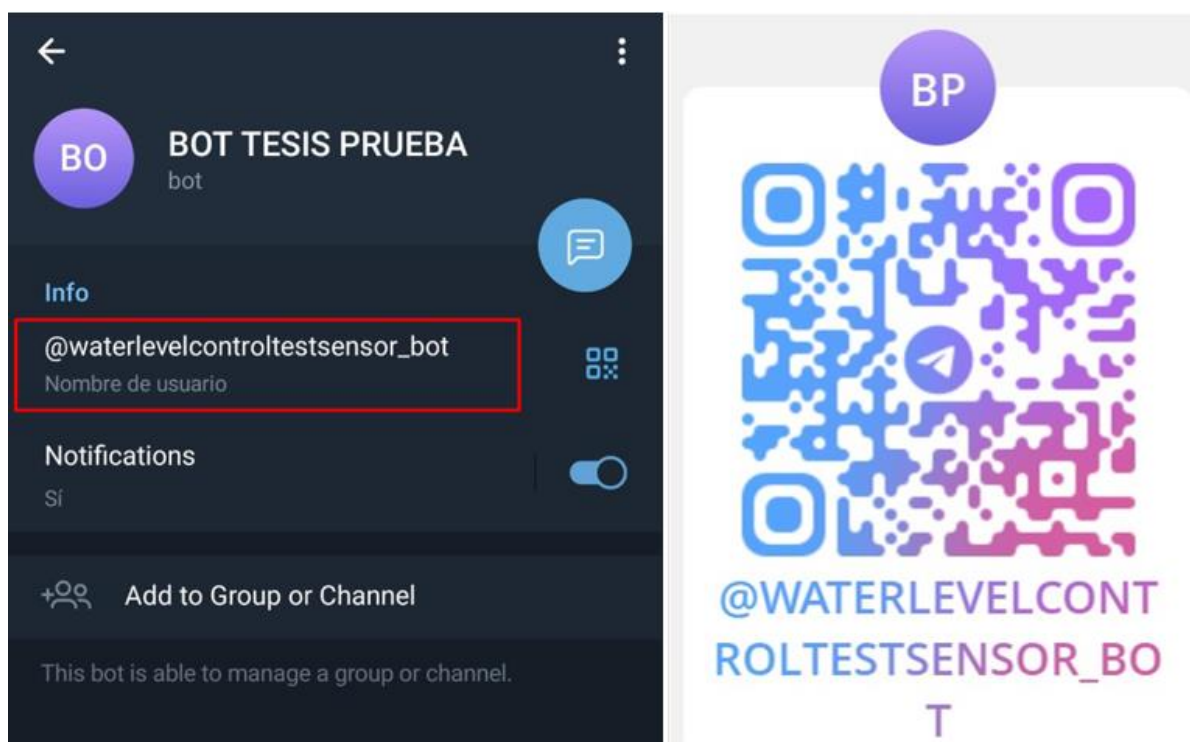
Nota. Al ingresar a la plataforma web con el link presentado se puede monitorear el estado de los nodos con las características y así resolver a tiempo cualquier inconveniente.

Fuente: Elaborado por el autor.

En la Figura 46 se evidencia el acceso al sistema de monitoreo a través de la aplicación Telegram, el cual puede realizarse escaneando el código QR o, de manera alternativa, buscando el usuario @waterlevelcontroltestsensor_bot. Esta opción complementa el acceso web y brinda al usuario una forma rápida y práctica de supervisar los datos en tiempo real.

Figura 46

Acceso al sistema mediante Telegram a través de código QR o usuario configurado



Nota. Representación del usuario y código QR para ingresar a Telegram.

Fuente: Elaborado por el autor.

4.2.4 Test de diseño

Podemos evidenciar en la Tabla 17 la validación de los requerimientos de diseño.

Tabla 17

Validación del requerimiento de diseño

	Subsistema de prueba	Diseño
Test de diseño	¿Prueba de despliegue?	Si/No

Descripción:

Prueba enfocada en validar que el prototipo cumpla con los lineamientos de diseño establecidos, incluyendo accesibilidad de sensores, modularidad de la estructura, ubicación estratégica de los componentes, protección frente a desconexiones accidentales y usabilidad de la interfaz de monitoreo.

Prerrequisitos:

- a) Prototipo completamente ensamblado con sensores y módulos instalados.
- b) Acceso a la plataforma de monitoreo desde un navegador web.
- c) Herramientas básicas (destornillador, multímetro, conectores) para pruebas de acceso y mantenimiento.
- d) Simulación de pruebas de manipulación (conexión/desconexión y transporte del prototipo).

Pasos:

- Verificar accesibilidad a sensores y componentes internos para ajustes o reemplazos.
- Evaluar facilidad de transporte del prototipo en ambiente controlado.
- Comprobar la correcta ubicación de sensores y la precisión de sus mediciones.
- Probar resistencia del sistema frente a desconexiones accidentales.
- Validar que la interfaz web de monitoreo sea simple, clara y de fácil interacción para el usuario.

Resultado esperado:

El diseño debe garantizar acceso rápido a los componentes sin afectar el funcionamiento, manteniendo un formato compacto y modular que facilite el transporte y estabilidad de las conexiones durante el uso. Los sensores deben ubicarse en posiciones estratégicas que optimicen las mediciones, mientras que la plataforma de monitoreo debe presentar una interfaz intuitiva y amigable para el usuario.

Nota. Validación de los requerimientos de diseño.

Fuente: Elaborado por el autor.

Resultado obtenido

El sistema transmitió los datos de los sensores en tiempo real sin pérdidas, empleando el protocolo MQTT para asegurar la comunicación detectando y registrando automáticamente los niveles críticos de agua por medio de alertas almacenadas para su análisis posterior.

4.2.5 Test lógico

En este caso se va a evaluar los requerimientos del sistema lógico evidenciado en la Tabla 18.

Tabla 18

Validación del sistema lógico

	Subsistema de prueba	Lógico
Test lógico	¿Prueba de despliegue?	Si/No

Descripción:

Prueba orientada a validar la lógica funcional del sistema: transmisión en tiempo real de datos de sensores a la plataforma, detección automática de niveles críticos de agua, registro de eventos críticos, garantía de no perder datos durante la operación de prueba y uso de un protocolo de comunicación seguras como MQTT para la transmisión.

Prerrequisitos:

- a) Plataforma de monitoreo operativa con almacenamiento habilitado
- b) Bróker/servidor de mensajería configurado MQTT con credenciales seguras.
- c) Firmware cargado en la microcontroladora con lectura periódica de sensores y envío en tiempo real.
- d) Reglas/umbrales de alerta configurados para niveles críticos de agua.
- e) Conectividad de red estable para la prueba (Wi-Fi).

Pasos:

- Iniciar el sistema y verificar publicación periódica de datos de sensores hacia el bróker.
- Forzar condiciones de nivel crítico (simulación o prueba controlada) y verificar generación automática de alerta.
- Confirmar que cada alerta quede registrada con marca de tiempo y detalle del evento en la plataforma.
- Interrumpir brevemente la red (simular pérdida de conexión) y verificar reintento/reenvío sin pérdida de datos al restablecer.
- Inspeccionar que la comunicación se realice con el protocolo seguro.

Resultado esperado:

Los datos se transmiten en tiempo real y de forma continua a la plataforma ante un nivel crítico, el sistema detecta y genera la alerta automáticamente, lo cual cada alerta queda registrada y no se pierden datos durante cortes breves ya que el sistema reanuda y entrega los mensajes pendientes.

Nota. Validación de los requerimientos lógicos.

Fuente: Elaborado por el autor.

Resultado obtenido

Durante la ejecución de las pruebas lógicas, el sistema logró transmitir los datos de los sensores en tiempo real hacia la plataforma de monitoreo manteniendo la continuidad sin pérdidas de información. Al forzar condiciones de nivel crítico, el sistema detectó automáticamente los valores anómalos y generó las alertas correspondientes, las cuales fueron registradas con marca de tiempo en la base de datos para su trazabilidad. Finalmente, se verificó que la transmisión de información se realizó utilizando el protocolo seguro MQTT con autenticación, cumpliendo con los requerimientos SRS1 a SRS5.

4.3 Validar requerimientos de Sistema

La validación de los requerimientos se realizó mediante pruebas funcionales, inspección técnica y justificación documental, utilizando una maqueta representativa del canal de riego y un prototipo electrónico operativo, cuyo funcionamiento se evidencia en la Figura 47.

En cuanto a los requerimientos de uso, se comprobó que los sensores monitorean el nivel de agua en tiempo real como se evidencia en el requerimiento (SySR1), generación de alertas automáticas ante valores críticos (SySR2), y la notificación de la disponibilidad del canal se muestra mediante la aplicación móvil (SySR3). Por ende las lecturas se registraron en una base de datos segura (SySR4) y se verificó la consulta histórica de registros (SySR5).

Figura 47

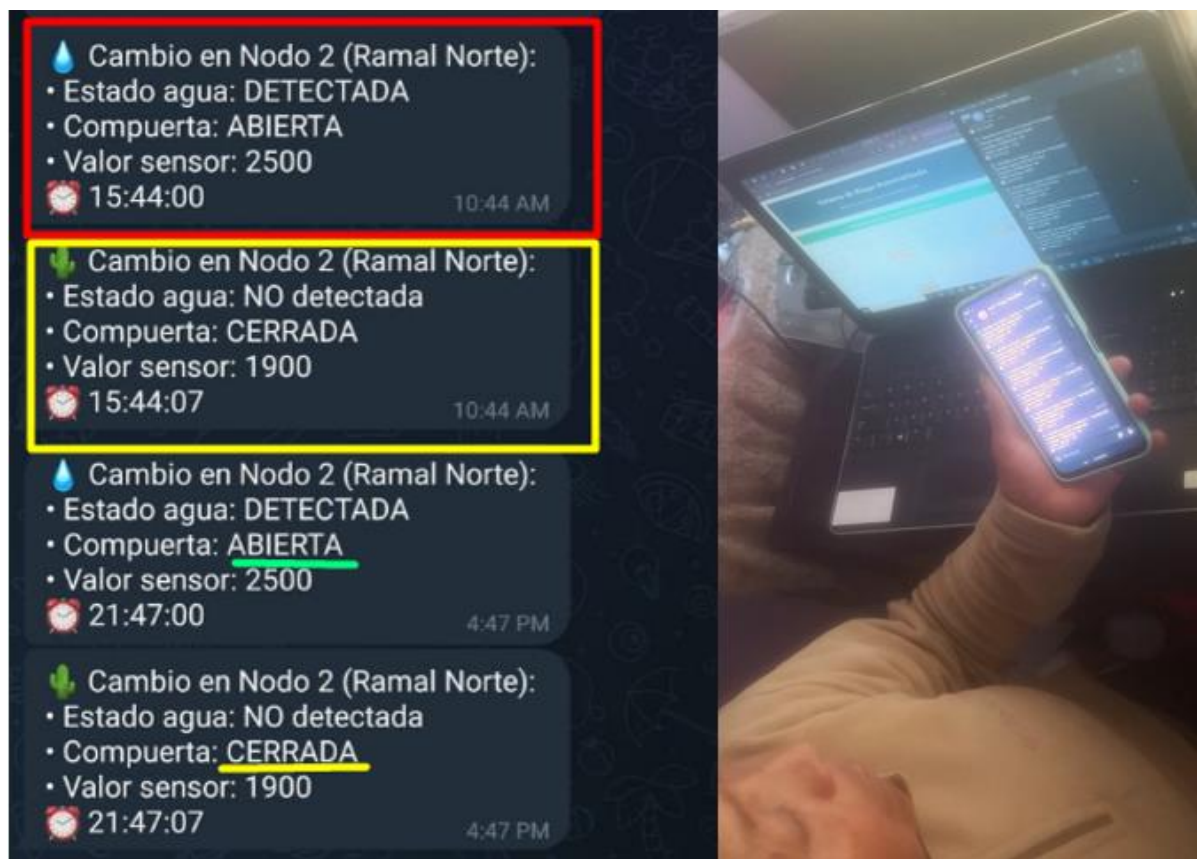
Estados de operación de la compuerta del canal de riego



En los requerimientos de interfaz, se validó la visualización de datos en la aplicación móvil (SySR6), la compatibilidad de los sensores inalámbricos con el microcontrolador (SySR7), el uso de terminología sencilla en idioma español (SySR8), así como la emisión de alertas visuales y auditivas (SySR9) y el estado activo/inactivo de los sensores en la interfaz (SySR10), tal como se observa en la Figura 48.

Figura 48

Visualización de la interfaz del sistema y notificaciones en la aplicación móvil



Respecto a los requerimientos de performance, se justificó la operación continua del sistema (SySR11) y la optimización energética en periodos de inactividad (SySR12), verificando además la transmisión de datos cada 2 segundos (SySR13) y la precisión de los sensores dentro de un margen del $\pm 5\%$ (SySR14). El desempeño en condiciones térmicas de 0°C a 50°C (SySR15) se fundamenta en las especificaciones técnicas de los componentes utilizados.

En cuanto a los modos y estados, se comprobó la alternancia entre modo activo y de bajo consumo (SySR16), el cambio automático a modo reposo en condiciones de estabilidad (SySR17), y la activación de alarmas ante límites excedidos (SySR18). Asimismo, se validó el reinicio manual del sistema mediante la aplicación (SySR19) y la indicación del estado del sistema con un LED de funcionamiento (SySR20).

Finalmente, para los requerimientos físicos, se verificó la protección parcial de los sensores frente al agua en condiciones controladas (SySR21), por lo cual la necesidad de carcasa resistente se consideró solo para implementaciones reales (SySR22), así mismo se constató la viabilidad de los dispositivos para su instalación (SySR23) y la fijación segura de los módulos de comunicación inalámbrica (SySR24). Por último, se instalaron sensores en puntos estratégicos del canal a escala, validando el requerimiento de futura implementación (SySR25).

4.3.1 Validar requerimientos de Stakeholders

A continuación, se va a evaluar los requerimientos de Stakeholders distribuidos por operacional y uso.

En este caso se validará los requerimientos del sistema operacional como se muestra en la Tabla 19.

Tabla 19

Test del sistema operacional

	Subsistema de prueba	Operacional
Test Operacional	¿Prueba de despliegue?	Si/No

Descripción:

Validación del cumplimiento de los requerimientos operacionales, asegurando que el sistema monitoree, registre y gestione los datos de nivel de agua en tiempo real,

Prerrequisitos:

- a) Prototipo armado y en operación.
- b) Sensores de nivel calibrados.
- c) Plataforma de monitoreo activa.

d) Conexión de comunicación inalámbrica estable.

Pasos:

- Verificar la transmisión en tiempo real de los niveles de agua.
 - Medir consumo energético del sistema durante operación.
 - Comprobar envío de alertas automáticas al detectar valores críticos.
 - Validar registro y almacenamiento de datos en la plataforma.
 - Observar distribución equitativa de riego en la simulación.
 - Evaluar reducción de desperdicio de agua en ciclos de prueba.
-

Resultado esperado:

El sistema debe mostrar niveles en tiempo real, enviar alertas, almacenar datos, operar con bajo consumo y garantizar un uso eficiente del agua.

Nota. Validación de los requerimientos operacionales.

Fuente: Elaborado por el autor.

Resultado obtenido

El prototipo cumplió con los requerimientos operacionales evidenciando los niveles en tiempo real, registró y almacenamiento de datos correctamente, así mismo dio a conocer un bajo consumo y generó alertas automáticas, contribuyendo a la optimización y uso equitativo del agua.

A continuación, en la Tabla 20 se presenta la validación de los requerimientos de uso

Tabla 20

Validación de los requerimientos de uso

	Subsistema de prueba	Uso

Test de uso	¿Prueba de despliegue?	Si/No
-------------	------------------------	-------

Descripción:

Evaluación de la experiencia de usuario validando accesibilidad, facilidad de uso y claridad de la interfaz de la plataforma de monitoreo.,

Prerrequisitos:

- a) Plataforma web activa con acceso mediante enlace.
- b) Bot de Telegram habilitado con código QR y nombre de usuario.
- c) Base de datos en funcionamiento.
- d) Usuarios de prueba con distintos niveles de conocimiento tecnológico.

Pasos:

- Acceder al sistema vía web y Telegram.
- Revisar claridad de la interfaz y notificaciones en idioma español.
- Verificar accesibilidad y simplicidad en la navegación.
- Validar protección de la privacidad de los datos recolectados.

Resultado esperado:

La plataforma debe ser intuitiva, notificar disponibilidad y proteger la privacidad de los usuarios.

Nota. Validación de los requerimientos de uso.

Fuente: Elaborado por el autor.

Resultado obtenido

La plataforma fue accesible desde web y Telegram, presentó una interfaz clara en español, notificó la disponibilidad del canal y el estado del suelo, siendo fácil de usar para diferentes perfiles de usuario, garantizando además la seguridad de los datos.

Figura 49

Colaboradores del barrio



Nota. Demostración del funcionamiento del prototipo automatizado.

Fuente: Elaborado por el autor.

4.4 Pruebas de integración del sistema

[RamosCarolina_Trabajo de grado.mp4](#)

Una vez validados los requerimientos en ambiente controlado, se verifica el funcionamiento del sistema frente a condiciones reales de operación en el canal de riego del barrio Río Frío. Para ello, se establecieron criterios comparativos que evidencian las variaciones necesarias en los parámetros de diseño y operación. En la Tabla 21 se resumen los principales aspectos que deben ajustarse al trasladar el sistema al entorno real, considerando la distancia de sensado, los caudales, las condiciones ambientales, la conectividad y gestión energética.

Tabla 21

Comparación de criterios entre ambiente controlado y canal real

Parámetro	Ambiente controlado	Ambiente real	Margen de error	Observación
Tiempo de apertura de compuerta	2 s	10-15 s	8 – 13 s	Mayor presión del agua y torque necesario en el canal real.
Precisión de sensores de nivel de agua (kPa)	1.77 kPa (h = 0.18 m)	4.9 – 5.9 kPa (h = 0.50–0.60 m)	±0.5–1.0 cm	En ambiente controlado la profundidad es baja, por eso la presión es menor. En ambiente real aumenta por la mayor lámina de agua.
Caudal del canal	0.003 – 0.004 m ³ /s (3–4 L/s)	0.10 – 0.15 m ³ /s (100–150 L/s)	±5% por variación en velocidad y sección	Aumento del área de sección y velocidad del flujo.

Transmisión de datos (WiFi/MQTT)	1 s	2 a 5 s	1 a 3 s	Pérdida de señal en áreas rurales.
Notificación a usuarios	≤ 1 s	3 a 5 s	2 a 4 s	Retardo de conexión con servidor o red móvil.
Energía de operación microcontrolado res	Estable a 5V DC	Variable (5V a 12V)	-----	Alimentación por baterías.

Nota. La tabla representativa presenta la comparación del ambiente real y ambiente controlado generados por medio del funcionamiento de la maqueta, permitiendo identificar criterios que requieren ajustes.

4.4.1 Análisis del caudal disponible y superficie de riego alcanzable

Con el fin de evaluar la viabilidad hidráulica del sistema de riego automatizado propuesto, se realiza un análisis del caudal disponible en el canal y su relación con la superficie de terreno que puede ser abastecida. Para este reporte se consideran los valores de caudal obtenidos en condiciones reales de operación del canal, los cuales se encuentran en un rango comprendido entre 100 y 150 litros por segundo (L/s), conforme a los resultados presentados en la Tabla 21.

A partir de estos valores, se asume una distribución equitativa del caudal hacia las tres compuertas automatizadas que conforman el sistema, permitiendo estimar el caudal disponible por compuerta. Asimismo, para el cálculo de la superficie de riego alcanzable se adopta un

requerimiento promedio de 0,8 L/s por hectárea, correspondiente a sistemas de riego por gravedad, criterio acorde a las características agrícolas del sector de estudio.

En función de estos parámetros, se determina la superficie aproximada que puede ser irrigada tanto por cada compuerta como por el sistema en su conjunto. Los resultados obtenidos del análisis de caudal y superficie de riego alcanzable se presentan de forma resumida en la Tabla 22.

Tabla 22

Relación entre caudal disponible y superficie de riego alcanzable

Caudal total (L/s)	Caudal por compuertas (L/s)	Superficie por compuerta (ha)	Superficie total (ha)
100	33	41	125
150	50	62	187

Nota. El cálculo de la superficie de riego se realiza considerando un requerimiento promedio de 0,8 L/s por hectárea para riego por gravedad.

4.4.2 Control de precisión diario del canal de riego

En este caso se realiza un control de precisión diario del canal, considerando el nivel de agua detectado, el caudal estimado y el porcentaje de apertura de las compuertas, lo cual los resultados obtenidos permiten evaluar la capacidad del sistema para mantener una distribución estable y eficiente del recurso de agua a lo largo de cada jornada de riego. Los datos del control diario se presentan en la Tabla 23.

Tabla 23*Control diario de precisión del canal de riego mediante el sistema automatizado*

Día	Nivel de agua detectado (cm)	Caudal estimado (L/s)	Apertura de la compuerta (%)	Estado del sistema	Observaciones
1	45	100	60	Estable	Operación normal
2	48	110	65	Estable	Ajuste automático por incremento de nivel
3	50	120	70	Estable	Mayor demanda de riego
4	46	105	62	Estable	Corrección automática del caudal
5	52	130	75	Estable	Nivel alto controlado
6	47	108	64	Estable	Operación continua
7	49	115	68	Estable	Variación diaria compensada

Nota. Se puede determinar que los valores presentados corresponden a un escenario de operación diaria simulada del sistema de riego automatizado, considerando el monitoreo continuo del nivel de agua y la regulación automática de las compuertas con el fin de mantener un flujo estable y eficiente durante cada jornada de riego.

Conclusiones

El diseño del prototipo cumple con la revisión teórica y el estado del arte, permitiendo establecer criterios funcionales, tecnológicos y metodológicos coherentes con las arquitecturas IoT aplicables a sistemas de riego. Asimismo, la aceptación comunitaria del 97,5% evidencia la factibilidad técnica y operativa del sistema en el barrio, demostrando concordancia entre los requerimientos reales del sector y las capacidades del prototipo desarrollado.

Se puede determinar que el análisis de la situación actual del canal, complementado con los datos obtenidos de la comunidad, permitió determinar con claridad los requerimientos técnicos del sistema dando a conocer que el 87,5% de los usuarios reconoció la necesidad de integrar tecnologías IoT para mejorar la gestión del riego, mientras que el 100% consideró que el uso de sensores facilita una mejor toma de decisiones, lo que respalda la selección de sensores de nivel, actuadores, microcontroladores y sistemas de comunicación utilizados en el prototipo.

Incluso el diseño del sistema integral se estructuró sobre una arquitectura IoT validada teóricamente y respaldada por la aceptación comunitaria, donde el 100% de los usuarios demostraron un interés por conocer y utilizar el sistema automatizado de canales de riego, lo cual confirma la pertinencia del diseño propuesto. Por otro lado, la integración de tecnologías como ESP32, nodos Arduino, comunicación MQTT y sensores de nivel permitió definir un sistema sostenible, eficiente y adaptable al entorno del barrio Río Frío.

Las pruebas de funcionamiento confirmaron que el prototipo opera correctamente, reproduciendo la lógica de control, la transmisión de datos y la activación automática de las compuertas conforme al diseño planteado. El desempeño del sistema demostró estabilidad y

respuesta adecuada en cada una de las etapas del proceso, validando la integración entre sensores, microcontroladores, actuadores y la plataforma de monitoreo, lo que demuestra que el prototipo no solo funciona de manera adecuada en un entorno controlado, sino que también cuenta con aceptación comunitaria para su futura aplicación real.

Recomendaciones

Se recomienda efectuar calibraciones regulares de los sensores instalados en el sistema de riego, con el fin de garantizar lecturas confiables frente a variaciones ambientales propias del entorno agrícola. Un proceso adecuado de calibración evita errores en la detección del nivel del agua y asegura el funcionamiento correcto de las compuertas automatizadas, fortaleciendo la precisión y estabilidad operativa del prototipo.

En base a que la transmisión de datos hacia el servidor MQTT depende de la conectividad inalámbrica del ESP32, se sugiere mantener y supervisar periódicamente la calidad de la red Wi-Fi privada utilizada. Asegurar una señal estable permite que la información proveniente de los sensores llegue al servidor sin interrupciones y garantizando la continuidad del monitoreo y de las notificaciones del sistema sin interferencias.

Considerando que el prototipo opera mediante nodos electrónicos distribuidos a lo largo del canal de riego, se recomienda analizar la posibilidad de incorporar fuentes de energía autónoma, como paneles solares de baja potencia para posibles casos en el cual no tengamos acceso al suministro de energía eléctrica.

También es recomendable fortalecer el sistema de riego automatizado mediante la implementación del registro y almacenamiento histórico de los datos generados durante el control diario del canal, tales como el nivel de agua y el porcentaje de apertura de las compuertas. La disponibilidad de esta información permitirá realizar análisis comparativos a lo largo del tiempo, identificar patrones de comportamiento del caudal y optimizar la toma de decisiones en la gestión del riego, contribuyendo a una operación más eficiente y sostenible del sistema en el mediano y largo plazo.

Bibliografía

- Abengoa. (6 de Marzo de 2013). *Slideshare*. Obtenido de <https://es.slideshare.net/slideshow/manejo-de-sedimentos-en-sistemas-hidroelctricas/16983585>
- Altieri, M. (2008). Los impactos del cambio climático sobre las comunidades campesinas y de agricultores tradicionales y sus respuestas adaptativas. *Agroecología*.
- Aranibar, J. (2023). Development of a system for intelligent irrigation forthe automation of water use. *Contribution_918_a*, 4.
- AWS. (15 de Julio de 2025). Obtenido de AWS Amazon: https://aws.amazon.com/es/free/?trk=3ad9a803-2790-4f66-993c-d7d4648b794f&sc_channel=ps&ef_id=Cj0KCQiAubrJBhCbARIsAHIdxD9Qmweml_S_G2MU8NbD652dTSmvjYtHTmdbiFiAf8IYE0zkRxyv4pUaAtkZEALw_wcB:G:s&s_kwid=AL!4422!3!780600711246!e!!g!!aws!23178730709!190531268154
- Becerra, D. (30 de Septiembre de 2023). Flujo de agua en un canal nanopolimérico con revestimientos de pared de grafeno y nitruro de boro hexagonal: un estudio atomístico. Obtenido de <https://typeset.io/papers/water-flow-in-a-polymeric-nanoslit-channel-with-graphene-and-3ecubkbfb2>
- Brunete, A., Herrero, R., & San Segundo, P. (Marzo de 2024). *Ingeniería diseño industrial*. Obtenido de https://bookdown.org/alberto_brunete/intro_automatica/sensoractuador.html
- Cáceres, G. (22 de Diciembre de 2023). *Universidad Loyola*. Obtenido de <https://www.uloyola.es/blog/doctorado/una-tesis-desarrolla-un-sistema-predictivo-de-riego-que-optimiza-el-uso-de-agua-y-energia>

Carril, A. (3 de Julio de 2023). *Scielo*. Obtenido de

http://www.scielo.org.ar/scielo.php?pid=S0011-67932023000100515&script=sci_arttext

Chugá Rosero, M. (18 de Octubre de 2022). *Repositorio UISEK*. Obtenido de

<https://repositorio.uisek.edu.ec/bitstream/123456789/4876/1/Chug%C3%A1%20Rose-ro%20Mar%C3%ADa%20Alejandra%20.pdf>

Coba, P. (2024). *Pcobau*. Obtenido de

<https://www.pcobau.com.ec/index.php/proyectos/canal-de-riego-quitubi-imantag>

Durand Nole, G. (2021). *Univeridad César Vallejo*. Obtenido de

https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/70923/Durand_NGADP-SD.pdf?sequence=1&isAllowed=y

Eby, K. (2 de Enero de 2019). *smartsheet*. Obtenido de El poder del diseño y proceso

iterativos: <https://es.smartsheet.com/iterative-process-guide#:~:text=El%20modelo%20iterativo%20es%20una,flexibilidad%20para%20incorporar%20cambios%20r%C3%A1pidamente>.

Fan, H. (17 de Marzo de 2023). Análisis del estado actual de la investigación sobre el flujo de

fluidos en microcanales. 37, 380-385. Obtenido de <https://typeset.io/papers/analysis-of-the-current-status-of-research-on-fluid-flow-in-21lwbj0k>

Fernández, Á. (2013). *Fundaciongeneraluclm*. Obtenido de

<https://fundaciongeneraluclm.es/wp-content/uploads/2018/05/24-%C3%81NGEL-FERN%C3%81NDEZ.pdf>

Florentino, A. (2006). Obtenido de

http://saber.ucv.ve/ojs/index.php/rev_venes/article/view/983

García, N. (2016). *IMTA*. Obtenido de [https://www.imta.gob.mx/biblioteca/libros_html/riego-](https://www.imta.gob.mx/biblioteca/libros_html/riego-drenaje/operacion-canales.pdf)

[drenaje/operacion-canales.pdf](https://www.imta.gob.mx/biblioteca/libros_html/riego-drenaje/operacion-canales.pdf)

- Garzón, A., & Torres, E. (Noviembre de 2018). *Dspace*. Obtenido de <https://dspace.ups.edu.ec/bitstream/123456789/16462/1/UPS-ST003826.pdf>
- Guambo, D., & Dave, O. (12 de Septiembre de 2022). *Universidad Politécnica Salesiana*. Obtenido de <https://dspace.ups.edu.ec/bitstream/123456789/24009/1/UPS-GT004151.pdf>
- Guerrero, J. (21 de Septiembre de 2014). *Pluselectric*. Obtenido de <https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/>
- Lascano, A. (2014). *Estación meteorológica autónoma con transmisión de datos inalámbrica en 5GHz 2x2 MIMO*. Universidad Politécnica Salesiana. Guayaquil: Maskana. Obtenido de <https://images.app.goo.gl/KJBmEGNqqLH29f1CA>
- Llontop Silva, M., & Correa, D. (2023). Variabilidad estacional en la cuenca Chancay-Lambayeque: desafíos y oportunidades en la gestión de los recursos hídricos. *South Sustainability*, 3-4.
- Mizhir, J. (29 de Abril de 2024). Una revisión del estudio de las características de flujo en canales abiertos ramificados. *42(2)*, 560-566. Obtenido de <https://typeset.io/papers/a-review-of-studying-the-flow-characteristics-in-branching-rp1inqlmol>
- Munguiá, A. (s.f.). *Scribd*. Obtenido de <https://es.scribd.com/document/237956294/La-Importancia-de-Los-Canales>
- Naciones Unidas. (2018). Obtenido de <https://repositorio.cepal.org/server/api/core/bitstreams/cb30a4de-7d87-4e79-8e7a-ad5279038718/content>
- Novatronic. (2024). Obtenido de <https://novatronicec.com/index.php/product/sensor-de-nivel-de-agua/>

- Ponce, V. (25 de Junio de 2014). Obtenido de https://ponce.sdsu.edu/la_difusion_de_la_escorrentia_presentacion.html
- Proain. (3 de Septiembre de 2020). Obtenido de <https://proain.com/blogs/notas-tecnicas/frecuencia-y-tiempos-de-riego-para-los-cultivos-agricolas>
- Quiñones, M. (24 de Febrero de 2017). Sistema De Monitoreo de Variables Medioambientales Usando Una Red de Sensores Inalámbricos y Plataformas De Internet De Las Cosas. doi:<https://doi.org/10.29019/enfoqueute.v8n1.139>
- Rey, J. (3 de Noviembre de 2020). Flujo de agua dentro de varios canales de nanoconfinamiento geométrico. *Scispace*, 22, 24633-24639. doi:<https://doi.org/10.1039/D0CP04491A>
- Royce, W. (3 de Noviembre de 2019). *IONOS*. Obtenido de Digital Guide: <https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/>
- Sajid, I. (2022). Performance Evaluation and Water Availability of Canal Irrigation Scheme in Punjab Pakistan. *MDPI*.
- SENAGUA. (Septiembre de 2016). Obtenido de https://www.controlsanitario.gob.ec/wp-content/uploads/downloads/2019/05/Estrategia-Nacional-de-Calidad-del-Agua_2016-2030.pdf
- Sigma Electrónica*. (2022). Obtenido de <https://www.sigmaelectronica.net/producto/esp-32/>
- Tabares, D. C. (2020). *PROTOTIPO DE SISTEMA DE CONTROL DE RIEGO PARA UN CULTIVO DE SOYA CON SUPERVISIÓN REMOTA DE HUMEDAD*. SANTIAGO DE CALI: UNIVERSIDAD DE SAN BUENAVENTURA.
- Tapia, C., & Carrera, H. (2011). *Promoción de los cultivos andinos para el desarrollo rural en Cotacachi-Ecuador*. Cotacachi, Imbabura, Ecuador: Instituto Nacional Autónomo

- de Investigaciones Agropecuarias (INIAP). Obtenido de
<https://repositorio.iniap.gob.ec/bitstream/41000/5142/1/iniapsc362.pdf>
- Velázquez Mendoza, J. (20 de Septiembre de 2010). *CICESE*. Obtenido de
<https://cicese.repositorioinstitucional.mx/jspui/bitstream/1007/872/1/184951.pdf>
- Villatoro, B. (2024). Cosecha de agua de lluvia una fuente alternativa de aprovechamiento hídrico en los períodos de estiaje de los ríos. *Cengicana*, 14-17. Obtenido de
<https://cengicana.org/files/20150828053605921.pdf>
- Yangkhurung, J. (2018). Effect of Canal Operation on Sedimentation and Erosion in Irrigation canals. *master2*, 5.
- Zavala, B., Vela , J., Taipe , E., & Astete, I. (18 de Junio de 2020). *Ingemmet*. Obtenido de
https://repositorio.ingemmet.gob.pe/bitstream/20.500.12544/3832/5/Zavala-Origen_respuesta_Colca_Andagua.pdf
- Zhang, J. (2022). Características de flujo de canales abiertos basadas en la distribución de parches de vegetación combinada rígida parcialmente discontinua. *13*.
doi:<https://doi.org/10.3389/fpls.2022.976646>

ANEXOS

Anexo 1 (Encuesta a los moradores del barrio)

1. Actualmente, ¿Cómo es gestionado el riego en su sector?

- a) Manualmente
- b) Con un sistema de riego automatizado
- c) Con apoyo de reservorios
- d) Otro: _____

2. ¿Cree que el riego manual genera desperdicio de agua?

- a) Sí
- b) No

3. ¿Ha tenido dificultades con la distribución del agua para el riego en su comunidad?

- a) Frecuentemente
- b) Ocasionalmente
- c) Nunca

4. ¿Considera que el acceso al agua para riego en la comunidad es equitativo?

- a) Sí
- b) No

5. ¿Cree que un sistema de monitoreo ayudaría a conocer mejor la disponibilidad del agua en los canales de riego?

- a) Sí
- b) No

6. ¿Qué beneficio considera más importante en un sistema de monitoreo y control de riego?

- a) Conocer el nivel de agua en el canal

- b) Distribuir mejor el riego entre los agricultores
- c) Reducir el desperdicio de agua
- d) Otro: _____

7. ¿Considera que el uso de sensores para medir la humedad del suelo y el nivel de agua podría mejorar la toma de decisiones en el riego?

- a) Sí
- b) No
- c) No estoy seguro

8. ¿Cree que la tecnología IoT (sensores, monitoreo en tiempo real) podría aplicarse en el futuro para mejorar la gestión del riego en la comunidad?

- a) Sí
- b) No
- c) No estoy seguro

9. ¿Le interesaría conocer cómo funcionaría un prototipo de monitoreo de riego con sensores y control automatizado?

- a) Sí
- b) No

10. ¿Considera que un sistema como este podría ser útil para futuras mejoras en la gestión del agua en el barrio?

- a) Sí
- b) No
- c) No estoy seguro

Anexo 2 (Resultados de la encuesta realizada)

Pregunta 1: En la primera pregunta realizada, se encontró que el 100% de los encuestados indica que el riego en el barrio Río Frío se lleva a cabo de manera manual. Este resultado evidencia la urgente necesidad de automatizar el proceso de riego para optimizar su eficiencia. La Figura 1 ilustra la tabulación de los resultados obtenidos.

Figura 1

Resultados de la primera pregunta realizada.

1. Actualmente, ¿Cómo es gestionado el riego en su sector? (0 punto)



Nota: En la presente figura se evidencia que la gestión del riego en el barrio Río Frío se realiza de forma manual. Elaborado por el autor.

Pregunta 2: Los resultados obtenidos de la segunda pregunta indican que el 87.5% de los encuestados considera que el riego manual genera desperdicio de agua, mientras que el 12.5% restante opina lo contrario. Lo cual se puede determinar que es importante implementar sistemas que optimicen el uso del recurso hídrico. La Figura 2 muestra la distribución de las respuestas.

Figura 2

Resultados de la segunda pregunta realizada.

2. ¿Cree que el riego manual genera desperdicio de agua? (0 punto)



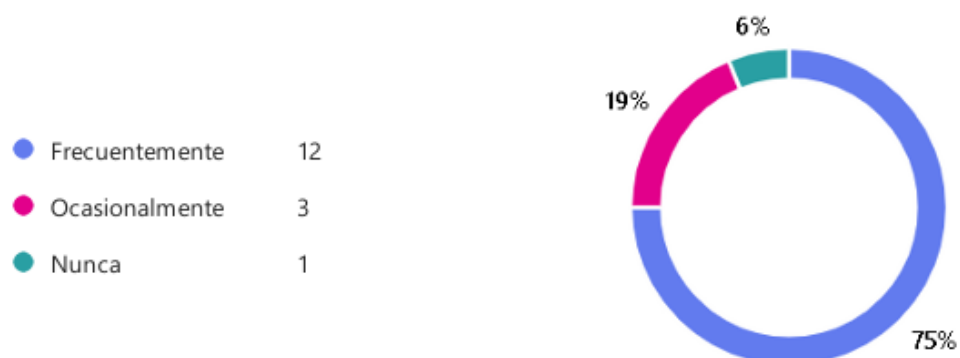
Nota: Mediante la encuesta realizada a los habitantes confirman que si existe un desperdicio de agua con el riego manualmente. Elaborado por el autor.

Pregunta 3: En relación con las dificultades en la distribución del agua el 75% de los encuestados indica haber enfrentado problemas de forma frecuente, mientras que el 18.75% lo experimenta ocasionalmente y solo el 6.25% asegura no haber tenido inconvenientes. Estos hallazgos evidencian la posibilidad de establecer un sistema de monitoreo que garantice un acceso más equitativo a este recurso de agua. En la Figura 3 se muestran los resultados obtenidos.

Figura 3

Resultados de la tercera pregunta realizada.

3. ¿Ha tenido dificultades con la distribución del agua para el riego en su comunidad? (0 punto)



Nota: Se evidencia las dificultades en la distribución del riego. Elaborado por el autor.

Pregunta 4: En la cuarta pregunta, el 81. 25% de los encuestados opina que el acceso al agua en la comunidad no es equitativo, mientras que el 18. 75% sostiene lo contrario. Esta disparidad pone de manifiesto la problemática de distribución que enfrenta el sector. En la Figura 4 se presenta una representación gráfica de estos resultados.

Figura 4

Resultados de la cuarta pregunta realizada.

4. ¿Considera que el acceso al agua para riego en la comunidad es equitativo? (0 punto)



Nota: El presente resultado nos muestra que no es equitativo el acceso al riego de agua.
Elaborado por el autor.

Pregunta 5: En la Figura 5 el 100% de los encuestados considera que la implementación de un sistema de monitoreo ayudaría a conocer mejor la disponibilidad de agua en los canales de riego, reafirmando la pertinencia del proyecto.

Figura 5

Resultados de la quinta pregunta realizada.

5. ¿Cree que un sistema de monitoreo ayudaría a conocer mejor la disponibilidad del agua en los canales de riego? (0 punto)



Nota: En la presente pregunta se evidencia que la mayoría de los habitantes están de acuerdo en que un sistema de monitoreo mejoraría la disponibilidad del agua en el canal de riego.
Elaborado por el autor.

Pregunta 6: En cuanto a los beneficios de un sistema de monitoreo, la Figura 6 nos muestra que el 68.75% de los encuestados considera que el principal beneficio sería reducir el desperdicio de agua, mientras que el 31.25% optó por la mejor distribución del riego entre los agricultores.

Figura 6

Resultados de la sexta pregunta realizada.

6. ¿Qué beneficio considera más importante en un sistema de monitoreo y control de riego? (0 puntos)



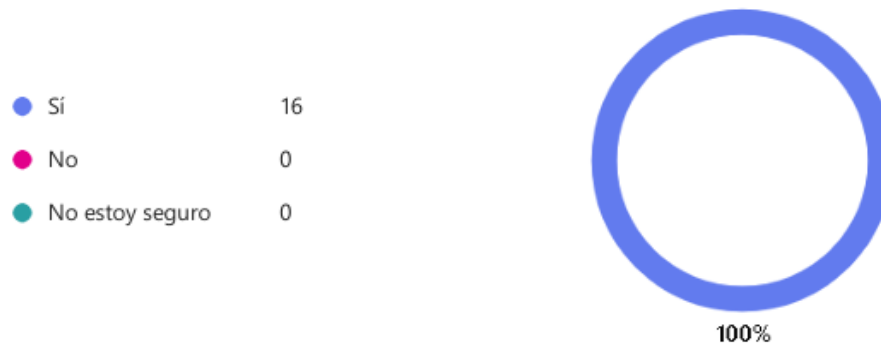
Nota: En este caso se puede determinar que al obtener un sistema de monitoreo en el sector se reduciría el desperdicio de agua, así mismo existiría una mejor distribución de riego entre los agricultores. Elaborado por el autor.

Pregunta 7: Podemos determinar que en la Figura 7 todos los encuestados, es decir, el 100%, opinan que el uso de sensores para medir la humedad del suelo y el nivel de agua contribuiría significativamente a una mejor toma de decisiones en el riego, lo que respalda directamente la funcionalidad del prototipo propuesto.

Figura 7

Resultados de la séptima pregunta realizada.

7. ¿Considera que el uso de sensores para medir la humedad del suelo y el nivel de agua podría mejorar la toma de decisiones en el riego? (0 punto)



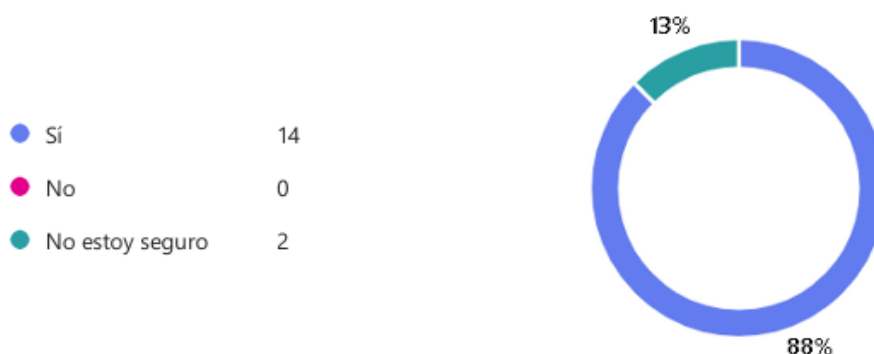
Nota: La mayoría de los encuestados están de acuerdo en que los sensores serían una herramienta fundamental para la toma de decisiones en el canal de riego. Elaborado por el autor.

Pregunta 8: Se puede dar a conocer que el 87.5% de los encuestados considera que la tecnología IoT tiene el potencial de aplicarse en el futuro para optimizar la gestión del riego en la comunidad, mientras que un 12.5% se muestra indeciso al respecto. Estos hallazgos se ilustra en la Figura 8.

Figura 8

Resultados de la octava pregunta realizada.

8. ¿Cree que la tecnología IoT (sensores, monitoreo en tiempo real) podría aplicarse en el futuro para mejorar la gestión del riego en la comunidad? (0 punto)



Nota: Representación de que la tecnología IoT sería esencial en un futuro. Elaborado por el autor.

Pregunta 9: Además, el 100% de los participantes manifestó su interés en comprender el funcionamiento de un prototipo de monitoreo de riego que utilice sensores y control automatizado, lo que subraya la importancia del proyecto. La Figura 9 presenta la distribución de estas opiniones.

Figura 9

Resultados de la novena pregunta realizada.

9. ¿Le interesaría conocer cómo funcionaría un prototipo de monitoreo de riego con sensores y control automatizado? (0 punto)



Nota: La mayoría de los habitantes del barrio quiere saber cómo funcionaría el prototipo.

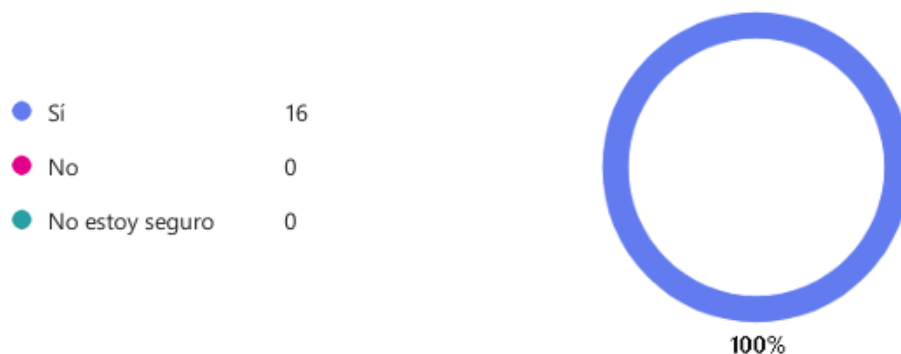
Elaborado por el autor.

Pregunta 10: Por último, en la Figura 10 todos los encuestados coincidieron en que un sistema como el que se propone sería valioso para mejorar la gestión del agua en el barrio Río Frío, lo que confirma el efecto positivo que podría resultar de la implementación del prototipo.

Figura 10

Resultados de la décima pregunta realizada.

10. ¿Considera que un sistema como este podría ser útil para futuras mejoras en la gestión del agua en el barrio? (0 punto)



Nota: Mediante la encuesta se evidencia que el utilizar un sistema de automatización mejoraría la gestión del agua en el barrio. Elaborado por el autor.

Anexo 3 (Código del ESP32 como nodo maestro para comunicación serial con Arduinos y publicación MQTT)

```
import network          # Biblioteca para configurar WiFi en el ESP32
import urequests         # Permite hacer solicitudes HTTP (no usado
directamente aquí)
import utime             # Manejo de tiempos (similar a time)
import time              # Temporizador para pausas en el programa
import json              # Manejo de datos en formato JSON
```

```

from umqtt.simple import MQTTClient # Cliente MQTT para comunicación con el
servidor cloud
from machine import Pin, UART       # Acceso a pines físicos y puertos UART
del ESP32
import random                       # Para generar números aleatorios (pruebas)

# Configuración de puertos UART para comunicación serial con Arduinos
uart1 = UART(1, baudrate=9600, tx=17, rx=16) # UART para Arduino #1
uart2 = UART(2, baudrate=115200, tx=Pin(22), rx=Pin(23)) # UART para Arduino
#2

# Credenciales de conexión WiFi
WIFI_SSID = "One 2.4G FAMILIA_RAMOS" # Nombre de la red WiFi
WIFI_PASSWORD = "1003473087" # Contraseña de la red WiFi

# Configuración del broker MQTT (servidor en la nube)
BROKER = "52.15.201.60" # IP del broker MQTT en la nube
CLIENT_ID = "esp32" # Identificador único del ESP32
TOPIC = b"test/topic" # Tópico MQTT donde se publicarán los mensajes

# Creación del cliente MQTT
client = MQTTClient(CLIENT_ID, BROKER, port=1883)

def mqttclient():
    client.set_callback(onMessage) # Define función para mensajes
    recibidos
    client.connect() # Conexión al broker en la nube
    print("Connected to MQTT broker, waiting for messages...")
    client.subscribe(TOPIC) # Suscripción al tópico MQTT

# LED indicador (GPIO2) para mostrar estados del sistema
led = Pin(2, Pin.OUT)

def connect_wifi():
    sta_if = network.WLAN(network.STA_IF) # Modo estación del ESP32
    if not sta_if.isconnected(): # Si no está conectado a WiFi:
        print("Connecting to WiFi...")
        sta_if.active(True) # Activa WiFi
        sta_if.connect(WIFI_SSID, WIFI_PASSWORD) # Inicia conexión

    # Espera hasta que el ESP32 logre conectarse
    while not sta_if.isconnected():

```

```

        utime.sleep(1)
        print(".", end="")

    # Una vez conectado
    print("\nConnected to WiFi")
    print("IP Address:", sta_if.ifconfig()[0])    # Imprime IP asignada

def sendData(node, level, status, servo):
    # Construcción del mensaje MQTT con los datos recibidos del Arduino
    data = "{status: " + str(status) + ", level: " + str(level) + ", node: " +
    str(node) + ", servo: " + str(servo) + "}"

    # Publicación en el tópico MQTT del servidor cloud
    client.publish(TOPIC, data.encode())
    print("Message send")    # Confirmación de envío

def onMessage(topic, message):
    # Función que se ejecuta si llega un mensaje desde el broker MQTT
    print(f"New message on {topic}: {message}")

def read_arduino(uart, name):
    # Verifica si hay datos disponibles en el UART
    if uart.any():
        try:
            line = uart.readline().decode('utf-8').strip()    # Lee línea serial
            y la decodifica
            print(f"Read from {name}")
            print(line)

            if line:
                # Se espera el formato: node,level,status,servo
                node, level, status, servo = line.split(",")
                sendData(node, level, status, servo)    # Envía datos al
servidor MQTT
        except Exception as e:
            print(f"Error reading {name}: {e}")

def main():
    connect_wifi()    # Conexión inicial a la red WiFi

    # Parpadeo del LED indicando conexión exitosa
    for _ in range(3):
        led.on()

```

```

        utime.sleep(0.3)
        led.off()
        utime.sleep(0.3)

    mqttclient()          # Conexión al broker MQTT

    try:
        while True:
            client.check_msg()          # Revisa si hay mensajes
entrantes MQTT

            read_arduino(uart1, "Arduino 1") # Lectura serial Arduino 1
            time.sleep(1)

            read_arduino(uart2, "Arduino 2") # Lectura serial Arduino 2

            time.sleep(5)                # Pausa entre ciclos

        except KeyboardInterrupt:
            print("Program interrupted by user.") # Si el usuario detiene el
programa
            client.disconnect()            # Cierra conexión MQTT

# Punto de entrada principal del programa
if __name__ == "__main__":
    main()

```

Anexo 4 (Código fuente del nodo esclavo Arduino Uno para lectura de nivel, control de compuerta y comunicación)

```

#include <Wire.h>
#include <Servo.h>
Servo servoMotor;

int waterLevel = 0;

#define I2C_ADDRESS 0x08
String lastCommand = "";
bool responseSent = false;

void setup() {
    Wire.begin(I2C_ADDRESS);
    Wire.onReceive(receiveEvent);
}

```

```

Wire.onRequest(requestEvent);
pinMode(13, OUTPUT);
Serial.begin(115200);
Serial.println("I2C Slave Ready");
servoMotor.attach(9);
}

void loop() {

  //Water level reading
  waterLevel = digitalRead(2);
  Serial.print("Water level: ");
  Serial.println(waterLevel);

  if(waterLevel == 1){
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }

  if (responseSent && Wire.available() == 0) {
    responseSent = false;
  }

  // Servomotor rotation
  for (int i = 0; i <= 90; i++){
    servoMotor.write(i);
    delay(25);
  }

  for (int i = 89; i > 0; i--){
    servoMotor.write(i);
    delay(25);
  }

  delay(1000);
}

void receiveEvent(int howMany) {
  lastCommand = "";
  while (Wire.available()) {
    char c = Wire.read();
    if (c >= 32 && c <= 126) {
      lastCommand += c;
    }
  }
}

```

```

    }
    responseSent = false;
    Serial.print("Received command: ");
    Serial.println(lastCommand);
}

void requestEvent() {
    if (responseSent) {
        Wire.write('\0');
        return;
    }

    String response;
    if (lastCommand.length() > 0) {
        response = "CMD:" + lastCommand;
    } else {
        response = "READY";
    }

    response.trim();

    Serial.print("Sending response: ");
    Serial.println(response);

    Wire.write(response.c_str());
    Wire.write('\0');

    responseSent = true;
}

```

Anexo 5 (Dashboard de Control de Riego)

```
<!DOCTYPE html>
```

```
<html lang="es">
```

```
<head>
```

```
<meta charset="UTF-8" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

```
<title>Sistema de Riego Automatizado</title>
```

```
<style>
```

```
body {
```

```
font-family: Arial, sans-serif;  
  
background: #e9f4ff;  
  
margin: 0;  
  
padding: 0;  
  
}
```

```
.header {  
  
background-color: #1e3a5f;  
  
color: white;  
  
text-align: center;  
  
padding: 20px;  
  
border-radius: 0 0 10px 10px;  
  
}
```

```
.warning {  
  
background-color: #f59e0b;  
  
color: white;  
  
text-align: center;  
  
padding: 10px;  
  
font-weight: bold;  
  
}
```

```
.container {  
  
display: flex;  
  
justify-content: center;
```

```
flex-wrap: wrap;  
  
gap: 20px;  
  
padding: 30px;  
}
```

```
.card {  
  
background-color: white;  
  
border-radius: 12px;  
  
padding: 20px;  
  
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
  
width: 250px;  
}
```

```
.card h3 {  
  
color: #2563eb;  
  
margin-bottom: 15px;  
}
```

```
.status {  
  
display: flex;  
  
align-items: center;  
  
gap: 10px;  
  
font-weight: bold;  
  
color: red;  
}
```

```
.btn {  
    background-color: #3b82f6;  
    color: white;  
    padding: 5px 10px;  
    border-radius: 6px;  
    border: none;  
}  
  
.footer {  
    font-size: 0.8rem;  
    margin-top: 10px;  
    color: gray;  
}  
  
</style>  
</head>  
<body>  
    <div class="header">  
        <h1>Sistema de Riego Automatizado</h1>  
        <p>Monitoreo en tiempo real de los canales de riego</p>  
    </div>  
  
    <div class="warning">  
        ⚠ ADVERTENCIA - Nodo 2 no responde  
    </div>
```

```
<div class="container">
```

```
<div class="card">
```

```
<h3>Nodo 1 - Canal Principal</h3>
```

```
<p><strong>Nivel de agua:</strong> 642</p>
```

```
<p class="status">Estado agua: <span><img alt="red circle icon" data-bbox="531 251 554 269" style="vertical-align: middle;"/> No detectada</span></p>
```

```
<p><strong>Compuerta:</strong> <button class="btn">Cerrada
```

```
(0°)</button></p>
```

```
<p><strong>Temperatura:</strong> 24.5°C</p>
```

```
<p class="footer">Actualizado hace 17 segundos</p>
```

```
</div>
```

```
<div class="card">
```

```
<h3>Nodo 2 - Terreno 2</h3>
```

```
<p><strong>Nivel de agua:</strong> 312</p>
```

```
<p class="status">Estado agua: <span><img alt="red circle icon" data-bbox="531 582 554 600" style="vertical-align: middle;"/> No detectada</span></p>
```

```
<p><strong>Compuerta:</strong> <button class="btn">Cerrada
```

```
(0°)</button></p>
```

```
<p><strong>Temperatura:</strong> 26.1°C</p>
```

```
<p class="footer">Actualizado hace 8 segundos</p>
```

```
</div>
```

```
<div class="card">
```

```
<h3>Nodo 3 - Terreno 3</h3>
```

```
<p><strong>Nivel de agua:</strong> 587</p>
```

```

<p class="status">Estado agua: <span> No detectada</span></p>

<p><strong>Compuerta:</strong> <button class="btn">Cerrada
(0°)</button></p>

<p><strong>Temperatura:</strong> --</p>

<p class="footer">Actualizado hace X segundos</p>

</div>

</div>

</body>

</html>

```

Anexo 6 (Levantamiento de servicios del sistema en la nube cloud del bot, broker MQTT y servidor web mediante Docker Compose).

Los comandos presentados permiten el levantamiento del servicio en la nube, el cual presenta la imagen del bot desplegando automáticamente los contenedores requeridos mediante Docker Compose que es el servidor web, para finalmente asegurar la ejecución persistente y organizada de los servicios del sistema.

```

C:\Users\DELL>ssh server
The authenticity of host 'ec2-18-188-29-87.us-east-2.compute.amazonaws.com (18.188.29.87)' can't be established.
ED25519 key fingerprint is SHA256:XgQsCUrn01uN25sz2/t6/pSPmPw0gPR3znodDuLe1lEc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-188-29-87.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Tue Nov 25 16:36:56 UTC 2025

System load:  0.0           Temperature:   -273.1 C
Usage of /:   35.7% of 6.71GB Processes:      113
Memory usage: 28%          Users logged in: 1
Swap usage:   0%           IPv4 address for ens5: 172.31.46.218

Expanded Security Maintenance for Applications is not enabled.

31 updates can be applied immediately.
24 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Nov 25 16:21:44 2025 from 45.169.145.94

```

```

Seleccionar admin@ip-172-31-39-227: ~
ubuntu@ip-172-31-46-218:~$ sudo docker build -t botimage
[+] Building 12.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                                docker:default
=> => transferring dockerfile: 223B                                              0.1s
=> [internal] load metadata for docker.io/library/python:slim-trixie            1.2s
=> [internal] load .dockerignore                                                 0.1s
=> => transferring context: 2B                                                  0.0s
=> [1/5] FROM docker.io/library/python:slim-trixie@sha256:0a6ecac02dc3d4c5dbb024b753af084cafe41f5416e02193f1ce345 5.3s
=> => resolve docker.io/library/python:slim-trixie@sha256:0a6ecac02dc3d4c5dbb024b753af084cafe41f5416e02193f1ce345 0.0s
=> => sha256:0a6ecac02dc3d4c5dbb024b753af084cafe41f5416e02193f1ce345d671ec966e 10.37kB / 10.37kB 0.0s
=> => sha256:5af4c7f950774a1abf3fd4e8e3fc95f4d0fe684c7fdc1eb10777fc5a017371c7 1.75kB / 1.75kB 0.0s
=> => sha256:88fe63acba12eb55b9f21a121536ad36b2fdbd13c46e08159eaa7a1f3adcfcc 4.95kB / 4.95kB 0.0s
=> => sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 29.78MB / 29.78MB 1.4s
=> => sha256:aebd987125d414f355d50710ebe2c8c1ef82be7f9b295a56c5a75f8a0c4246ea 1.29MB / 1.29MB 0.5s
=> => sha256:20a580af6aa213e326f6c625b68adba4affedcee7ed5002f13160db623e69497 12.17MB / 12.17MB 1.0s
=> => sha256:ec80ae14678f10e720bc3837c21cd58c485f01d08c008d31702a3961595057c5 250B / 250B 0.6s
=> => extracting sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 2.1s
=> => extracting sha256:aebd987125d414f355d50710ebe2c8c1ef82be7f9b295a56c5a75f8a0c4246ea 0.2s
=> => extracting sha256:20a580af6aa213e326f6c625b68adba4affedcee7ed5002f13160db623e69497 0.9s
=> => extracting sha256:ec80ae14678f10e720bc3837c21cd58c485f01d08c008d31702a3961595057c5 0.0s
=> [internal] load build context                                                0.1s
=> => transferring context: 9.60kB                                              0.0s
=> [2/5] WORKDIR /app/                                                         0.2s
=> [3/5] COPY ./telegram/ /app/                                                0.1s
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt                    5.0s
=> [5/5] RUN echo "America/Bogota" > /etc/timezone                           0.3s
=> exporting to image                                                         0.3s
=> => exporting layers                                                         0.3s
=> => writing image sha256:621694a90a2ea22558a5b2c74a5e3f94aa758b7847c3a06ed458586d76d61b63 0.0s
=> => naming to docker.io/library/botimage                                     0.0s
ubuntu@ip-172-31-46-218:~$ sudo docker compose -f compose.yaml up -d
WARN[0000] /home/ubuntu/compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 12/12
  webserver Pulled                                3.8s
  0e4bc2bd6656 Already exists                     0.0s
  b5feb73171bf Pull complete                      3.0s
  108ab8292820 Pull complete                      3.1s
  53d743880af4 Pull complete                      3.1s
  77fa2eb06317 Pull complete                      3.2s
  192e2451f875 Pull complete                      3.2s
  de57a609c9d5 Pull complete                      3.3s
  mqtt Pulled                                    1.5s
  2d35ebdb57d9 Pull complete                      0.7s
  35ae20b23dca Pull complete                      1.0s
  4ab41a84cd6d Pull complete                      1.0s
[+] Running 4/4
  Network ubuntu_default Created                  0.2s
  Container ubuntu-telegram-bot-1 Started         2.1s
  Container ubuntu-mqtt-1 Started                 2.6s
  Container ubuntu-webserver-1 Started             2.5s

```

```

ubuntu@ip-172-31-46-218:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED    STATUS    PORTS
a9ad444b3d8a   eclipse-mosquitto "/docker-entrypoint..." 3 minutes ago Up 7 seconds 0.0.0.0:1883->1883/tcp, [::]:1883->1883/tcp, 0.0.0.0:9001->9001/tcp, [::]:9001->9001/tcp
a9f625635816   botimage      "python3 bottcp.py"      3 minutes ago Up 3 minutes 0.0.0.0:8000->80/tcp, [::]:8000->80/tcp
464e749c13f8   nginx        "/docker-entrypoint..." 3 minutes ago Up 3 minutes 0.0.0.0:8000->80/tcp, [::]:8000->80/tcp
ubuntu@ip-172-31-46-218:~$ client_loop: send disconnect: Connection reset

C:\Users\DELL>ssh server
The authenticity of host 'ec2-52-15-201-60.us-east-2.compute.amazonaws.com (52.15.201.60)' can't be established.
ED25519 key fingerprint is SHA256:kyYRqnfY8NSzmaXJh4Jfy9C/aq8IeIAxx+lyr3JH9Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-15-201-60.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Linux ip-172-31-39-227 6.12.48+deb13-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.12.48-1 (2025-09-20) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 25 19:04:03 2025 from 45.169.145.94
ubuntu@ip-172-31-39-227:~$ ls -la
total 12
drwxr-xr-x 3 ubuntu ubuntu 4096 Nov 25 19:04
-rw-r--r-- 1 ubuntu ubuntu  220 Nov 25 19:04 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3526 Nov 25 19:04 .bashrc
-rw-r--r-- 1 ubuntu ubuntu  589 Nov 25 19:04 .profile
Dockerfile compose.yaml mosquitto telegram web-server
ubuntu@ip-172-31-39-227:~$ client_loop: send disconnect: Connection reset

```

Anexo 7 (Código fuente del prototipo de automatización de canales de riego)

```

// Node 1
// Pin 4 -> level           // El pin 4 se usa para el sensor de nivel del
nodo 1
// Pin 6 -> servo          // El pin 6 se usa para el servomotor del nodo
1

// Node 2
// Pin 8 -> level           // El pin 8 se usa para el sensor de nivel del
nodo 2
// Pin 10 -> servo         // El pin 10 se usa para el servomotor del nodo
2

//Node 3
// Pin 11 -> Servo         // El pin 11 se usa para el servomotor del nodo
3

#include <Servo.h>           // Incluye la librería para controlar
servomotores

Servo servo1;               // Declara el servomotor del nodo 1
Servo servo2;               // Declara el servomotor del nodo 2
Servo servo3;               // Declara el servomotor del nodo 3

int waterLevel1 = 0;        // Variable para almacenar el estado del sensor
de nivel 1
int waterLevel2 = 0;        // Variable para almacenar el estado del sensor
de nivel 2
int servoPin1 = 6;          // Pin físico del servomotor 1
int servoPin2 = 10;         // Pin físico del servomotor 2
int servoPin3 = 11;         // Pin físico del servomotor 3

int SERVO1OPEN = 70;        // Ángulo de apertura del servo 1
int SERVO2OPEN = 90;        // Ángulo de apertura del servo 2
int SERVO3OPEN = 110;       // Ángulo de apertura del servo 3

// realIrrigationTime = 3600000 // 1 Hora en milisegundos (comentado,
referencia real)

```

```

long simulationIrrigationPeriod = 10000; // Tiempo de riego simulado en
milisegundos

long IRRIGATION_TIME = 5000; // Tiempo de riego por cada terreno

long startIrrigation = 0; // Variable para almacenar el inicio del riego
long currentIrrigationTime = 0; // Variable para almacenar el tiempo actual

bool cycleActive = false; // Indica si el ciclo de riego está activo
int currentSite = 0; // Indica el terreno actual (0,1,2,3)

void setup() { // Función que se ejecuta una sola vez al
iniciar

    pinMode(4, INPUT); // Configura el pin 4 como entrada (sensor nodo
1)
    pinMode(8, INPUT); // Configura el pin 8 como entrada (sensor nodo
2)
    Serial.begin(115200); // Inicia comunicación serial para monitoreo
    servo1.attach(servoPin1); // Asocia el servo 1 al pin 6
    servo2.attach(servoPin2); // Asocia el servo 2 al pin 10
    servo3.attach(servoPin3); // Asocia el servo 3 al pin 11

    servo1.write(SERVO10PEN); // Coloca el servo 1 en posición abierta
    servo2.write(SERVO20PEN); // Coloca el servo 2 en posición abierta
    servo3.write(SERVO30PEN); // Coloca el servo 3 en posición abierta

}

void loop() { // Función que se ejecuta continuamente

    //Water leavel reading
    waterLevel1 = digitalRead(4); // Lee el estado del sensor de nivel 1
    waterLevel2 = digitalRead(8); // Lee el estado del sensor de nivel 2

    // Stage 1
    // Código comentado para pruebas anteriores de temporización

    // Riego Terreno 1
    if (waterLevel1 == 0 && currentSite == 0 && cycleActive == false){
        // Si no hay agua en nodo 1, es el primer
terreno

        // y el ciclo no está activo
        cycleActive = true; // Activa el ciclo de riego

        servo1.write(SERVO10PEN); // Abre compuerta del nodo 1
        servo2.write(SERVO20PEN); // Abre compuerta del nodo 2
        servo3.write(SERVO30PEN); // Abre compuerta del nodo 3
    }
}

```

```

sendNodeData(1, 900, 1, 90); // Envía datos del nodo 1 por serial

delay(IRRIGATION_TIME);      // Mantiene el riego activo por un tiempo

currentSite = 1;              // Cambia al terreno 2
servo2.write(10);             // Cierra parcialmente compuerta 2
sendNodeData(2, 900, 1, 10);  // Envía datos del nodo 2
}

//Riego Terreno 2
if (waterLevel1 == 0 && waterLevel2 == 0 && currentSite == 1){
    // Si ambos sensores indican nivel bajo
    // y es el turno del terreno 2
    servo1.write(SERVO1OPEN);  // Mantiene abierta compuerta 1
    delay(500);                // Pequeño retardo de estabilización
    servo2.write(10);          // Mantiene cerrada compuerta 2
    servo3.write(SERVO3OPEN);  // Abre compuerta 3

    delay(IRRIGATION_TIME);    // Tiempo de riego del terreno 2
    currentSite = 2;           // Cambia al terreno 3
    servo3.write(0);           // Cierra compuerta del nodo 3
    sendNodeData(3, 900, 1, 0); // Envía datos del nodo 3
}

//Riego terreno 3
if(waterLevel1 == 0 && waterLevel2 == 0 && currentSite == 2){
    // Si ambos sensores indican nivel bajo
    // y es el turno del terreno 3
    servo1.write(SERVO1OPEN);  // Mantiene abierta compuerta 1
    servo2.write(10);          // Mantiene cerrada compuerta 2
    servo3.write(0);           // Mantiene cerrada compuerta 3

    delay(IRRIGATION_TIME);    // Tiempo de riego del terreno 3
    currentSite = 3;           // Marca final del ciclo
    servo3.write(SERVO3OPEN);  // Abre compuerta 3
    sendNodeData(3, 100, 0, 90); // Envía datos finales del nodo 3
}

if (currentSite == 3 && waterLevel1 == 0 && waterLevel2 == 1 && cycleActive
== true){
    // Condición para finalizar el ciclo
    sendNodeData(2, 100, 0, 85); // Envía datos finales del nodo 2
    sendNodeData(1, 100, 0, 85); // Envía datos finales del nodo 1
    cycleActive = false;         // Desactiva el ciclo de riego
    servo1.write(0);             // Cierra compuerta 1
    delay(1000);                 // Retardo de seguridad
    servo2.write(SERVO2OPEN);    // Abre compuerta 2
    servo3.write(SERVO3OPEN);    // Abre compuerta 3
}

```

```

    delay(IRRIGATION_TIME);    // Espera antes de reiniciar
    currentSite = 0;            // Reinicia el ciclo
}

delay(500);                    // Retardo general del sistema
}

// Send node, level, status, servo
void sendNodeData(int nodeId, int level, int status , int servo){
    // Función para enviar datos por Serial
    String data = String(nodeId) + "," +
                  String(level) + "," +
                  String(status) + "," +
                  String(servo);    // Construye la cadena de datos
    Serial.println(data);          // Envía los datos por el puerto serial
}

```

Anexo 8. (Código fuente del módulo de monitoreo de nivel de agua y notificación vía Telegram en ESP32)

```

----- IMPORTACIÓN DE LIBRERÍAS -----
import network                // Permite la conexión Wi-Fi
import urequests              // Para enviar solicitudes HTTP (API Telegram)
import time                   # Temporizador y funciones de retardo
from machine import Pin      # Manejo de pines GPIO (entrada/salida)

----- CONFIGURACIÓN DE CREDENCIALES Wi-Fi -----
WIFI_SSID = "Red Prueba"      # Nombre de tu red Wi-Fi
WIFI_PASSWORD = "12345678"    # Contraseña de tu red Wi-Fi

----- CONFIGURACIÓN DEL BOT DE TELEGRAM -----
BOT_TOKEN = "ABCdefGHIJKLmNOPQRsTuVWxyZ-123456AbCdE" # Token del bot
# (obtenido de BotFather)
GROUP_CHAT_ID = "1234567890" # ID del chat o grupo de Telegram

----- CONFIGURACIÓN DEL SENSOR -----
sensor_nivel = Pin(34, Pin.IN) # Define el pin 34 como entrada digital
# para leer el sensor de nivel de agua

# ----- FUNCIÓN PARA CONECTARSE A LA RED Wi-Fi -----
def connect_wifi():
    sta_if = network.WLAN(network.STA_IF) # Activa la interfaz Wi-Fi
    # en modo estación
    if not sta_if.isconnected():
        print("Conectando a WiFi...")
        sta_if.active(True) # Habilita la conexión Wi-Fi

```

```

        sta_if.connect(WIFI_SSID, WIFI_PASSWORD)      # Inicia conexión con
SSID y clave
        while not sta_if.isconnected():              # Espera hasta que la
conexión sea exitosa
            time.sleep(0.5)
        print("Conectado a WiFi:", sta_if.ifconfig()) # Muestra IP y datos de
conexión

# ----- FUNCIÓN PARA ENVIAR MENSAJE A TELEGRAM -----
def send_to_telegram(message):
    url = "https://api.telegram.org/bot{}/sendMessage".format(BOT_TOKEN) #
URL de la API para enviar mensajes
    payload = {
        "chat_id": GROUP_CHAT_ID,    # ID del chat de destino
        "text": message,              # Contenido del mensaje
        "parse_mode": "HTML"         # Permite usar formato HTML en el mensaje
    }
    try:
        response = urequests.post(url, json=payload) # Envío del mensaje como
JSON
        response.close()                  # Cierre de la respuesta
    HTTP
    except Exception as e:
        print("Error al enviar mensaje:", e)        # Muestra error si la
petición falla

# ----- BLOQUE PRINCIPAL -----
connect_wifi()                                # Llama a la función de conexión
Wi-Fi
estado_anterior = sensor_nivel.value()        # Lee el estado inicial del sensor
(0 o 1)

while True:
    estado_actual = sensor_nivel.value()        # Lectura actual del sensor

    if estado_actual != estado_anterior:        # Si hay cambio de estado (subida
o bajada del agua)
        if estado_actual == 1:
            send_to_telegram("💧 <b>Alerta:</b> Nivel alto de agua
detectado.") # Si hay agua: alerta
        else:
            send_to_telegram("✅ <b>Actualización:</b> Nivel de agua ha
bajado.") # Si bajó el nivel
            estado_anterior = estado_actual      # Se actualiza el estado
registrado

        time.sleep(2)                          # Espera 2 segundos antes de la
siguiente lectura

```