UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO BASADO EN IOT DE UN SISTEMA DE CONTROL PARA LA PRODUCCIÓN DE ORÉGANO EN INVERNADERO, EN EL DISTRITO LA YARADA LOS PALOS 2024"

PARA OPTAR:

EL TITULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR:

Bach. EDWARD ALDO PORTUGAL CUNO

TACNA – PERÚ

2025

UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

“DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO BASADO EN IOT DE UN SISTEMA DE CONTROL PARA LA PRODUCCIÓN DE ORÉGANO EN INVERNADERO, EN EL DISTRITO LA YARADA LOS PALOS 2024”

Tesis sustentada y aprobada el 10 de diciembre de 2025; estando el jurado calificador integrado por:

PRESIDENTE:	**:**	**Dr. ANIBAL JUAN ESPINOZA ARANCIAGA**
SECRETARIO	**:**	**Mtro. MARKO JESÚS POLO CAMACHO**
VOCAL	**:**	**Mtro. CARLOS ARMANDO RODRIGUEZ SILVA**
ASESOR	**:**	**Mag. HUGO JAVIER RIVERA HERRERA**

DECLARACIÓN JURADA DE ORIGINALIDAD

Yo, Edward Aldo Portugal Cuno, egresado, de la Escuela Profesional de Ingeniería Electrónica de la Facultad de Ingenieria de la Universidad Privada de Tacna, identificado con DNI 70889296, así como Hugo Javier Rivera Herrera con DNI 00494293; declaramos en calidad de autor y asesor que:

1. Somos los autores de la tesis titulada: "*Diseño e implementación de un prototipo basado en IoT de un sistema de control para la producción de orégano en invernadero, en el distrito La Yarada los Palos 2024*", la cual presento para optar el *Título Profesional de Ingeniero Electrónico*.
2. La tesis es completamente original y no ha sido objeto de plagio, total ni parcialmente, habiéndose respetado rigurosamente las normas de citación y referencias para todas las fuentes consultadas.
3. Los datos presentados en los resultados son auténticos y no han sido objeto de manipulación, duplicación ni copia.

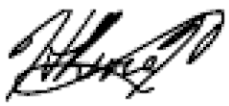
En virtud de lo expuesto, asumimos frente a *La Universidad* toda responsabilidad que pudiera derivarse de la autoría, originalidad y veracidad del contenido de la tesis, asi como por los derechos asociados a la obra.

En consecuencia, nos comprometemos ante *La Universidad* y terceros a asumir cualquier perjuicio que pueda surgir como resultado del incumplimiento de los aquí declarado, o que pudiera ser atribuido al contenido de la tesis, incluyendo cualquier obligación económica que debiera ser satisfecha a favor de terceros debido a acciones legales, reclamo o disputas resultantes de incumplimiento de esta declaración.

En caso de descubrirse fraude, piratería, plagio, falsificación o la existencia de una publicación previa de la obra, acepto toda consecuencia y sanciones que puedan derivarse de mi acción, acatando plenamente la normatividad vigente.

Tacna 10 de diciembre del 2025

Edward Aldo Portugal Cuno
DNI: 70889296



Hugo Javier Rivera Herrera
DNI 00494293

DEDICATORIA

A Dios por darme rayos de esperanza en los momentos más difíciles

A mis padres Gladys y Mariano por el apoyo que me brindaron, por el amor que me tienen, el sacrificio que hicieron para lograr mis metas.

A mi hermano Gustavo por brindarme su conocimiento, guía y comprensión.

A mi mascota Rex por estar siempre a mi lado, brindándome su compañía, dándome fuerzas para seguir.

Edward Aldo Portugal Cuno

AGRADECIMIENTO

Estas palabras de agradecimiento van dirigidas a todos lo que me apoyaron, sin ellos no hubiera sido posible culminar mi tesis.

MI MADRE: La tranquilidad que me otorgas es inmensa, cada cosa que me sale mal, siempre estas con un consejo que darme, nunca paro de aprender de ti.

MI PADRE: El alto conocimiento que tienes, me inspiro a seguir aprendiendo de la vida, a tener más cultura, tengo tanto que aprender del conocimiento que tienes, gracias por enseñarme la cultura del conocimiento.

MI HERMANO: Siempre compartiste tiempo conmigo, me sacabas de dudas que tenía, siempre me apoyaste cada vez que te lo pedía, tenemos que aprender mucho del uno al otro.

Edward Aldo Portugal Cuno

ÍNDICE GENERAL

PÁGINA DE JURADOS ii
DECLARACIÓN JURADA DE ORIGINALIDAD iii
DEDICATORIA iv
AGRADECIMIENTO v
ÍNDICE DE TABLAS ix
ÍNDICE DE FIGURAS x
ÍNDICE DE ANEXOS xiv
RESUMEN xv
ABSTRACT xvi
INTRODUCCION 1
Capitulo I:El PROBLEMA DE INVESTIGACIÓN 2
1.1. Descripción del problema 2
1.2. Formulación del problema 2
1.2.1. Problema general 2
1.2.2. Problemas específicos 2
1.3. Justificación e importancia de la investigación 3
1.4. Objetivos 3
1.4.1. Objetivo general 3
1.4.2. Objetivos específicos 3
CAPITULO II: MARCO TEORICO 4
2.1. Antecedentes de estudio 4
2.1.1. A nivel internacional 4
2.1.2. A nivel nacional 5
2.2. Bases teóricas 6
2.2.1. Concepto de Internet de las cosas (IoT) 6
2.2.2. Protocolos de comunicación IoT 8
2.2.3. Otro tipo de comunicación IoT 11
2.2.4. Tipos de topología utilizables 12
2.2.5. Invernaderos 14
2.2.6. Tipos de invernaderos 15
2.2.7. Invernadero autónomo: 17
2.2.8. Tipos de automatización 18

2.2.9. Arduino 19
2.2.10. IDE Arduino 19
2.2.11. Riego por goteo 19
2.2.12. Orégano 24
2.3. Definición de términos 25
2.3.1. Arduino: 25
2.3.2. NRF24L01 25
2.3.3. ESP32 26
2.3.4. Invernadero automatizado 26
2.3.5. Microcontrolador 26
2.3.6. Actuador 26
2.3.7. Relé 26
2.3.8. Sensor pH 26
2.3.9. Sensor de temperatura y humedad 27
2.3.10. Sensor de humedad del suelo 27
2.3.11. Sensor de CO2 27
2.3.12. Sensor ultrasonido 27
CAPITULO III: MARCO METODOLÓGICO 28
3.1. Diseño de la investigación 28
3.2. Acciones y actividades 28
3.2.1. Planeación 28
3.2.2. Funcionalidad 28
3.2.3. Sistema de control 30
3.3. Materiales e instrumentos 31
3.3.1. Materiales 31
3.3.2. Instrumentos 44
3.4. Operalización de las variables: 44
3.5. Procesamiento y análisis de datos 45
3.5.1. Diagramas de flujo de los sensores 45
3.5.2. Conexiones de los nodos 01-02-03 49
3.5.3. Conexiones del nodo 04 50
3.5.4. Conexiones del nodo 05 51
3.5.5. Conexiones del nodo central: 52
3.5.6. Implementación de software IoT: 52
3.5.7. Análisis y validación del sistema IoT implementado: 56
3.5.8. Base de datos: 57

3.5.9. Revisión general 60
CAPITULO IV: RESULTADOS 61
4.1. Conexión optima bueno recibimiento 61
4.2. Implementación física e incorporación en invernadero: 63
4.2.1. NODO-01 64
4.2.2. NODO-02 65
4.2.3. NODO-03: 67
4.2.4. NODO-04: 68
4.2.5. NODO-05: 70
4.2.6. NODO-CENTRAL: 71
4.3. Parámetros de medidos para un ambiente en prototipo invernadero 73
4.3.1. Temperatura 73
4.3.2. Humedad del aire 74
4.3.3. CO2 74
4.3.4. Humedad del suelo 75
4.3.5. Calidad de líquido para riego 75
4.3.6. Nivel de agua 75
CAPITULO V: DISCUSION 77
CONCLUSIONES 78
RECOMENDACIONES 79
REFERENCIAS BIBLIOGRÁFICAS 80
ANEXOS 83

ÍNDICE DE TABLAS

Tabla 1. Sector agropecuario parte agrícola 25
Tabla 2. Especificaciones técnicas Arduino Mega 2560 32
Tabla 3. Especificaciones técnicas ESP32 33
Tabla 4. Especificaciones técnicas módulo NRF24L01 34
Tabla 5. Especificaciones técnicas SIM7600SA 35
Tabla 6. Especificaciones técnicas de Antena Omnidireccional 36
Tabla 7. Especificaciones técnicas de DHT22 37
Tabla 8. Especificaciones técnicas del sensor capacitivo v1.2 38
Tabla 9. Especificaciones técnicas MH-Z19B 39
Tabla 10. Especificaciones técnicas HC-SR04 40
Tabla 11. Especificaciones técnicas de Sensor pH 41
Tabla 12. Especificaciones técnicas de servomotor 42
Tabla 13. Operalización de variables 45
Tabla 14. Datos con los sensores 56
Tabla 15. Nodos operativos 63

ÍNDICE DE FIGURAS

Figura 1. Arquitectura IoT 7
Figura 2. Topologia LoRaWAN 8
Figura 3. Arquitectura SigFox 9
Figura 4. Arquitectura NB-IoT 10
Figura 5. Arquitectura LTE 10
Figura 6. Topologia estrella 12
Figura 7. Tipo malla 13
Figura 8. Topología bus 14
Figura 9. Orientacion de Norte a Sur 15
Figura 10. Invernadero Tipo túnel 16
Figura 11. Invernadero Asimétrico 16
Figura 12. Invernadero Tipo capilla 17
Figura 13. Control de humedad 18
Figura 14. Cultivo por goteo 20
Figura 15. Riego superficial (estándar) 21
Figura 16. Riego por Exudación 22
Figura 17. Gotero regulable 23
Figura 18. Sector agropecuario parte agrícola 25
Figura 19. Topología estrella 29
Figura 20. Nodos envío y recepción de datos 29
Figura 21. Sistema de control 30
Figura 22. Sistema de control detallado 30
Figura 23. Sistema general 31
Figura 24. Modelo Arduino Mega 2560 32
Figura 25. Microcontrolador ESP32 33

Figura 26. NRF24L01 PA+LNA 34
Figura 27. Módulo 4G SIM7600SA E-L1C 35
Figura 28. Antena omnidireccional 36
Figura 29. Modelo de sensores de temperatura y humedad 37
Figura 30. Modelo de sensor de humedad del suelo 38
Figura 31. Módelo de sensor de MH-Z19B 39
Figura 32. Módulo de sensor HC-SR04 40
Figura 33. Módulo sensor pH 41
Figura 34. Servomotor 42
Figura 35. Interfaz Arduino IDE 43
Figura 36. Diagrama de flujo temperatura 46
Figura 37. Diagrama de flujo humedad del aire relativa 46
Figura 38. Diagrama de flujo humedad del suelo relativa 47
Figura 39. Diagrama de flujo nivel de agua 47
Figura 40. Diagrama de flujo CO2 48
Figura 41. Diagrama de flujo pH 48
Figura 42. Conexiones Nodo-01 49
Figura 43. Conexiones Nodo 02 49
Figura 44. Conexiones Nodo-03 50
Figura 45. Conexiones Nodo-04 51
Figura 46. Conexiones Nodo-05 51
Figura 47. Conexiones nodo central 52
Figura 48. Creaciones de cuenta (Blynk IoT) 53
Figura 49. Llenado de datos (Blynk IoT) 54
Figura 50. Creación de plantilla 54
Figura 51. Creación de parámetros a medir 55
Figura 52. Conexión Proyecto y software 58
Figura 53. Graficas insetadas en el software 58

Figura 54. Graficas insertadas en aplicativo móvil 59
Figura 55. Switches insertados para control 59
Figura 56. Cantidad de mensajes 60
Figura 57. Datos recibidos del nodo 1 en PC 61
Figura 58. Datos recibidos del nodo 2 en PC 61
Figura 59. Datos recibidos del nodo 3 en PC 62
Figura 60. Datos recibidos del nodo 4 en PC 62
Figura 61. Datos recibidos del nodo 5 en PC 62
Figura 62. Datos calculados por el Arduino Mega (promedio) en PC 62
Figura 63. Ordenes de riego, ventiladores y pH en PC 63
Figura 64. Datos de ppm en el Arduino Mega 2560 en PC 63
Figura 65. Parte física Nodo-01 64
Figura 66. Parte interna Nodo-01 64
Figura 67. Instalacion en invernadero Nodo-01 65
Figura 68. Parte física Nodo-02 65
Figura 69. Parte interna Nodo-02 66
Figura 70. Instalacion en invernadero Nodo-02 66
Figura 71. Parte física Nodo-03 67
Figura 72. Parte Interna Nodo-03 67
Figura 73. Instalacion en invernadero Nodo-03 68
Figura 74. Parte física Nodo-04 68
Figura 75. Parte interna Nodo-04 69
Figura 76. Instalacion en invernadero Nodo-04 69
Figura 77. Parte física del Nodo-05 70
Figura 78. Parte interna Nodo-05 70
Figura 79. Instalacion en invernadero Nodo-05 71
Figura 80. Parte fisica del Nodo Central 71
Figura 81. Parte interna del Nodo Central 72

Figura 82. Instalacion en invernadero Nodo Central 72
Figura 83. Instalacion de Nodo Central de cerca 73
Figura 84. Demostración de temperatura promedio 73
Figura 85. Demostración de Humedad promedio 74
Figura 86. Demostración de CO2 (ppm) 74
Figura 87. Demostración de humedad del suelo promedio 75
Figura 88. Demostración de pH 75
Figura 89. Demostracion de nivel de agua (cm) 76

ÍNDICE DE ANEXOS

Anexo 1. Código Del Nodo Nº01 83
Anexo 2. Código Del Nodo Nº02 87
Anexo 3. Código Del Nodo Nº04 91
Anexo 4. Código Del Nodo Nº04 95
Anexo 5. Código Del Nodo Nº05 99
Anexo 6. Código Del Nodo Central 101
Anexo 7. Matriz De Consistencia 120

RESUMEN

El proyecto tiene como objetivo diseñar e implementar un sistema IoT para el control y monitoreo de las condiciones ambientales en un entorno cerrado (invernadero) destinado a la producción de orégano, ubicado en el distrito La Yarada Los Palos, región Tacna. El sistema se basa en una arquitectura de red inalámbrica utilizando módulos NRD24L01 PA+LNA para la comunicación entre nodos ESP32 y un nodo central Arduino Mega 2560 todo esto conectando mediante una red en estrella, encargado de procesar y enviar los datos hacia la nube mediante un módulo SIM7600SA (4G LTE). El prototipo captura variables esenciales como temperatura, humedad del aire, humedad del suelo, pH, nivel de agua y concentración de CO2, a través de sensores distribuidos en el invernadero. Se implemento un software (Blynk IoT) de supervisión que permite la visualización remota de dichas mediciones, posibilitando el accionamiento automático de los ventiladores y riego cuando los valores superan los umbrales definidos. Por último, durante las pruebas, el sistema mostro un rendimiento estable, con trasmisión de datos continua y sin pérdidas significativas de paquetes, garantizando la fiabilidad de la comunicación inalámbrica. Además, se logró reducir el consumo de agua y el uso de insecticidas mediante el control automatizado o manual, obteniendo un ambiente más estable para el cultivo.

Palabras clave: IoT; NRF24L01; invernadero automatizado; orégano; sensores; conexión inalámbrica; software.

ABSTRACT

The project aims to design and implement an IoT-based system for controlling and monitoring environmental conditions in a closed environment (greenhouse) dedicated to oregano production, located in the district of La Yarada Los Palos, Tacna region. The system is based on a wireless network architecture using NRF24L01 PA+LNA modules for communication between ESP32 nodes and a central Arduino Mega 2560 node, all interconnected through a star network topology. The central node processes and transmits data to the cloud through a SIM7600SA (4G LTE**)** module. The prototype captures key variables such as temperature, air humidity, soil moisture, pH, water level, and CO2 concentration using sensors distributed throughout the greenhouse. A Blynk IoT software interface was implemented to allow remote visualization of these measurements, enabling the automatic activation of fans and the irrigation system when the values exceed predefined thresholds. Finally, during testing, the system showed stable performance, with continuous data transmission and no significant packet loss, ensuring reliable wireless communication. In addition, the automated or manual control system reduced water consumption and pesticide use, achieving a more stable and sustainable environment for crop growth.

Keywords: IoT; NRF24L01; automated greenhouse; oregano; sensors; wireless connection; software.

INTRODUCCION

La agricultura contemporánea exige soluciones que integren tecnología y practicas sostenibles para mejorar la productividad y la calidad de los cultivos. En el distrito La Yarada Los Palos en Tacna, la producción de orégano es una actividad relevante para la economía local; sin embargo, se enfrenta a limitaciones como el uso excesivo de insecticidas, manejo ineficiente del agua y falta de adopción de tecnologías de riego y monitorización que garanticen condiciones óptimas de cultivo.

Este trabajo propone el diseño e implementación de un prototipo basado en Internet de las Cosas (IoT) para el control y monitoreo de un invernadero destinado a la producción de orégano. El sistema integra microcontroladores, módulos de comunicación local (NRF24L01) y salida a Internet mediante un módulo celular (SIM7600SA). Además, tambien tiene el uso de sensores de temperatura, humedad ambiente, humedad del suelo, CO2, pH y nivel de agua.

El prototipo autónomo será diseñado e implementado en un invernadero, optando la facilidad de uso que nos brinda tanto para aislarlo de los agentes nocivos para las plantas dándole protección, mantenido estable el interior gracias a la tecnología implementada.

Según entregado por el BCRP en la ciudad de Tacna tenemos una amplia exportación de orégano, según los parámetros dichos por el BCRP, este trabajo de tesis no solo ayudaría a tener una mejor calidad en el cultivo tambien ayudaría a reducir el uso de agroquímicos y reduciendo el consumo de agua, facilitando la siembra en óptimas condiciones con poca intervención humana.

CAPITULO I: EL PROBLEMA DE INVESTIGACIÓN

1.1. Descripción del problema

La demanda de productos es cada vez mayor con el paso de los años, depende de la continuidad de la población. En la actualidad el problema del uso del agua, las plagas afectan en la calidad de los cultivos en los pobladores.

A pesar de todo, no hay un control apropiado de los insecticidas o el uso del agua con una medida controlada afectando al medio ambiente y a la salud, como en la región Tacna, Distrito los Palos. Teniendo un problema principal del uso desmedido de los insecticidas y el agua.

En el transcurso de los años se han desarrollado procedimientos para el menor uso de insecticidas y tener un control de riego de los cultivos mediante el uso de invernaderos con el control automatizados.

En el poblado Yarada los Palo, los agricultores de la zona, no cuentan con la capacitación adecuada en el uso de nuevas tecnologías sobre invernaderos y riego tecnificado y como consecuencia la productividad de sus campos de cultivos no son las más eficientes.

1.2. Formulación del problema

1.2.1. Problema general

¿En qué medida el diseño e implementación de un sistema de control basado en IoT permitirá mejorar la producción de orégano en la Yarada Los Palos, en la región Tacna?

1.2.2. Problemas específicos

a. ¿Cuáles son los parámetros climáticos necesarios para la producción de orégano?
b. ¿Qué tecnología de hardware y software será la más adecuada para el diseño del sistema de control?

1.3. Justificación e importancia de la investigación

BCRP (2024) informo que la producción en la agricultura en Tacna se vio disminuida por las menores áreas cosechadas destinadas al mercado externo, agroindustrial e interno, teniendo una disminución (-30,7 %) en la cosecha de orégano.

Sin embargo, con el paso del tiempo el aumento desmedido de los insecticidas ha ido creciendo, dañando los sembradíos y la salud de la población, provocando un producto poco saludable para la población, con esto teniendo en cuenta se puede hacer un estudio para la mejora en la producción de la región Yarada los Palos.

En este proyecto se justifica la mejora económica, protección de sembradíos, tener un ambiente óptimo para las zonas agrícolas y un producto más conservado con buena calidad para la salud. Con la implementación del diseño aportara una solución que sirva como ejemplo a los demás agricultores y los motive a implementar este diseño en sus zonas de cultivo para mejorar la economía y la seguridad de los pobladores por el uso disminuido de los insecticidas.

Para diseñar e implementar el Diseño e implementación de un sistema de control para el cultivo de orégano, que está basado en un dispositivo denominado controlador, que es un dispositivo embebido (Arduino, compuesto de sensores, actuadores, controladores) contara con una aplicación móvil para estar en todo momento conectado en tiempo real recibiendo datos del invernadero.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar e implementar un prototipo por IoT que tenga un sistema de control que mejore el cultivo de orégano en un invernadero mediante la monitorización clave como la humedad, temperatura, entre otros, en la Yarada Los Palos, en la región Tacna.

1.4.2. Objetivos específicos

a. Determinar los parámetros climáticos necesarios para la producción de orégano.
b. Determinar la tecnología de hardware y software que sea la más adecuada para el diseño del sistema de control basado en IoT.

CAPITULO II: MARCO TEORICO

2.1. Antecedentes de estudio

2.1.1. A nivel internacional

Suarez y Valverde (2023) en su trabajo de tesis “Implementación de un sistema inteligente para invernadero de cultivos de hortalizas utilizando tecnología IoT y energía solar” propusieron como objetivo general “Un sistema inteligente utilizando tecnología IoT y energía solar para un prototipo de cultivos de hortalizas (p.26).

Las conclusiones a la que llegaron en su prototipo “Implementaron un invernadero a dos aguas en el cual, se facilitó la instalación de tuberías, bombas de agua para el riego y adecuación de los respetivos sensores para controlar las condiciones climáticas del invernadero” (Suarez y Valverde, 2023, p.153); su segunda conclusión “mediante la implementación de un algoritmo a través de ecuaciones que describen la posición astronómica del sol implementando en la tarjeta ESP32 se logró tener un control en la orientación de modulo fotovoltaico” (Suarez y Valverde, 2023, p.153).

Pazmiño (2022) en su proyecto de tesis “Repotenciación de un sistema hidropónico convencional para el cultivo de lechugas Lactuca Sativa y Batavia Boinda Di Parigi, mediante automatización y plataformas IoT” tuvo como objetivo general de su investigación “repotencias un módulo hidropónico convencional, mediante sensoria, microcontroladores y plataforma IoT, para el control y monitoreo de los sensores, procesos de suministro de agua, nutrición, circulación y evacuación de la solución nutritiva en el crecimiento de lechugas” (p.20).

Las conclusiones a las que llego “por medio del software inventor se diseñó un modelado 3D del módulo hidropónico tomado en cuenta distintos puntos como el espacio y uso, movilidad, componentes y la arquitectura Lamba para obtener distintos planos de construcción” (Pazmiño, 2022, p.76); continuando con su segunda conclusión “Se evaluaron las distintas graficas que proporciono Ubidots conforme la evolución de las variables independientes pre establecidas, se pudo conocer como avanzaron los datos a través de los 50 días del experimento y como su grafica demostró” (Pazmiño, 2022, p.77).

Satuquinga (2023) en su trabajo de titulación “Sistema de control y monitoreo de riego, purificación de aire y fumigación para la plantación de tomate de riñon en

invernaderos mediante dispositivos IoT en la Agropecuaria San Miguel de Salcedo", tuvo como objetivo general "implementar un control de monitoreo de riego, purificación de aire y fumigación de tomate de riñon" (p.13).

Las conclusiones a la que llego "Para el crecimiento y el desarrollo siempre necesita las condiciones adecuadas como una temperatura que oscila entro los 18 ºC hasta los 25 ºC, humedad de ambiente no mayor a los 70 %, humedad del suelo no menor al 40 % y cantidad de CO2 que oscile entre 700 y 1000 ppm, motivo por el cual estos frutos al no tener estoy parámetros se enferman y disminuye la producción (Satuquinga,2023, p.67); como segunda conclusión, "Los dispositivos que fueron integrados del sistema permiten versatilidad y una fácil configuración en automatización, además se puede configurar las notificaciones y acceso de usuarios al sistema" (Satuquinga,2023,p.67).

Merino (2022) en su proyecto de investigación "Prototipo IoT para un invernadero acuapónico domestico en áreas urbanas", tiene como objetivo general, "implementar un prototipo IoT automatizado para un invernadero acuapónico domestico en áreas urbanas" (p.32).

Las conclusiones a las cuales llego fueron que "las condiciones ambientales bajo las que funciona un sistema acuapónico son de mucho cuidado y delicadas, por lo que la selección de sensores tanto análogos como digitales ayudo en gran medida a obtener las mediciones de estos parámetros para su constante monitoreo, evaluación y control con la ayuda de los actuadores. El ambiente se mantuvo oscilando dentro del rango optimo, es decir, entre 15 ºC y 30 ºC, el PH del agua entre 6 y 7 niveles, la humedad del ambiente entre el 45 % y 60 % HR, la luminosidad entre 400nm. Y 700nm. Y la temperatura del estanque de peces entre 18 ºC y 30 ºC" (Merino, 2022, p.87); en su segunda conclusión, "El sistema acuapónico completamente implementado y funcionando, al transcurso de 6 semanas se logró cosechar los primeros productos vegetales como acelga, lechuga, perejil, nabo y rábanos" (Merino, 2022, p.88).

2.1.2. A nivel nacional

Mendoza y Cordero (2022) en su proyecto realizado "Diseño de un sistema IoT para el monitoreo remoto de calidad del aire en amientes interiores de la facultad de ingeniería de la Universidad privada de Tacna" tiene como objetivo general "Diseñar un sisma IoT para el monitoreo remoto de la calidad del aire en ambientes interiores de la Facultad de Ingenieria de la Universidad Privada de Tacna." (p. 4).

En su trabajo de tesis logro concluir “Que es posible ampliar la cobertura de una red sin la necesidad de instalar APs utilizando una topología de red tipo Malla con la capacidad de ser vinculada a una red WLAN para transmitir la información al exterior teniendo un buen rendimiento tanto en potencia de transmisión, como en velocidad de transmisión.” (Mendoza y Cordero, 2022, p.80).

Ramirez (2024) en su tesis “Diseño de un sistema de monitoreo IoT para mejorar la productividad del agricultor en el Poblado de Cuchuchin-Sayan, 2023” como objetivo general “Conocer el sistema de monitoreo IoT y su relación con la productividad del agricultor del poblado de Cuchuchin – 2023” (p.22.)

En el trabajo de investigación que realizo concluyo “Existe una relación significativamente positiva entre el sistema de monitoreo IoT y la productividad de agricultor del poblado de Cuchuchin – 2023. Obteniendo un coeficiente de correlación de 0.887 equivalente a una magnitud alta.” (Ramirez, 2024, p.75.).

Falcon y Cuya (2022) en su trabajo de investigación “Sistema Web basado en IoT para mejorar el rendimiento de aves finas de un criadero inteligente en la empresa LM Business” como objetivo general tuvo “Determinar en qué medida la implementación de un sistema web basado en IoT mejora el rendimiento de producción de aves finas en un criadero inteligente en la empresa LM Business” (p.21.).

En su tesis concluyo “Que con la implementación del prototipo de la criadora inteligente y el sistema web dio como resultado la reducción significativa de 776 segundos a 226 segundos del tiempo para determinar la temperatura adecuada del criadero en tiempo real” (Falcon y Cuya, 2022, p.115.); como segunda conclusión tuvo “Con el desarrollo del sistema web, el trabajador podrá llevar un monitore diario reduciendo el tiempo de 768 segundos a 2.18 segundos, donde determinara la húmeda de la criadora en tiempo real.” (Falco y Cuya, 2022, p.115.).

2.2. Bases teóricas

2.2.1. Concepto de Internet de las cosas (IoT)

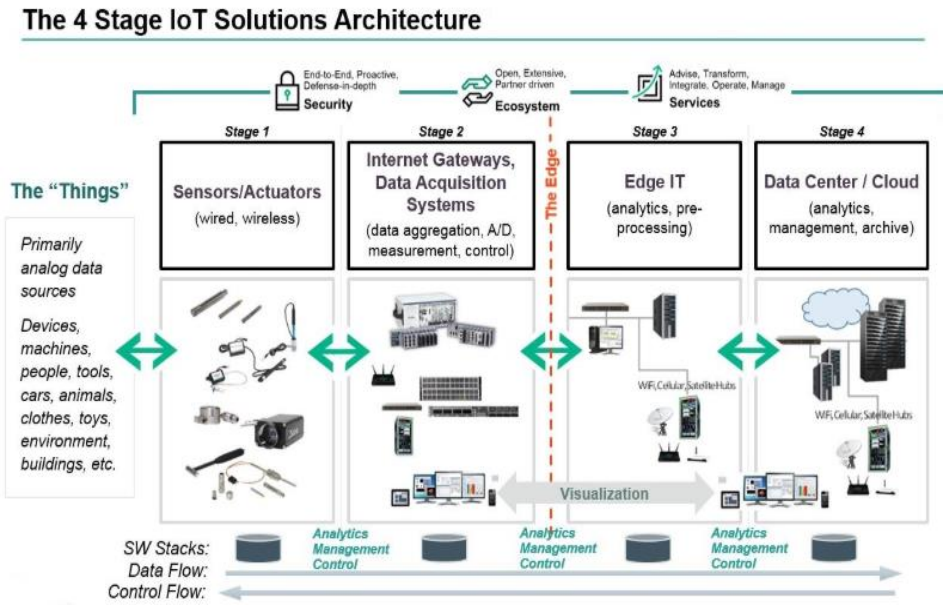
La tecnología IoT, también conocida como el internet de las cosas, ha revolucionado el mundo al transformarlo digitalmente. Esto ha sido posible gracias a la tecnología inalámbrica, los sistemas microelectrónicos, los sistemas electromecánicos y los servicios que ofrece internet. Esta tecnología permite conectar una variedad de dispositivos electrónicos, como sensores, actuadores, electrodomésticos y maquinaria, entre otros. Todos estos dispositivos se conectan de manera inalámbrica a través de

internet, formando una red colectiva que permite la transmisión y recepción de datos. Estos datos pueden ser visualizados mediante un servidor de aplicaciones, lo que facilita el monitoreo y control de procesos (Satuquinga,2023).

Para que un sistema IoT funcione necesitamos de dispositivos configurados para que sean compatible con dicha tecnología. Estos dispositivos denominados “Dispositivos IoT” serán nuestros ojos y oídos cuando físicamente estemos ausentes los cuales tendrán la función de capturar cualquier dato que estén programados para almacenar. También se requiere un sistema Backend que se encargue de almacenar dichos datos, ya sea en un servidor local o en la nube (Mendoza y Ochoa,2022).

Figura 1

Arquitectura IoT



Nota: Tomado de Satuquinga (2023)

Ventajas

- Capacidad de conectar dispositivos electrónicos a través de la red
- Transmisión y recepción de información de manera eficiente en tiempo real.
- Bajo consumo energético.
- Monitoreo remoto.
- Control a larga distancia.
- Comunicación directa.

Desventajas

- La información no está encriptada, lo que la hace vulnerable a hackeos y robos de información.
- Dispositivos carecen de confidencialidad.
- Falta de compatibilidad entre dispositivos.

2.2.2. Protocolos de comunicación IoT

- LoRaWAN (Long Range Wide Area Network)
- Sigfox
- NB-IoT (Narrowband IoT)
- LTE-M (LTE Cat M1)

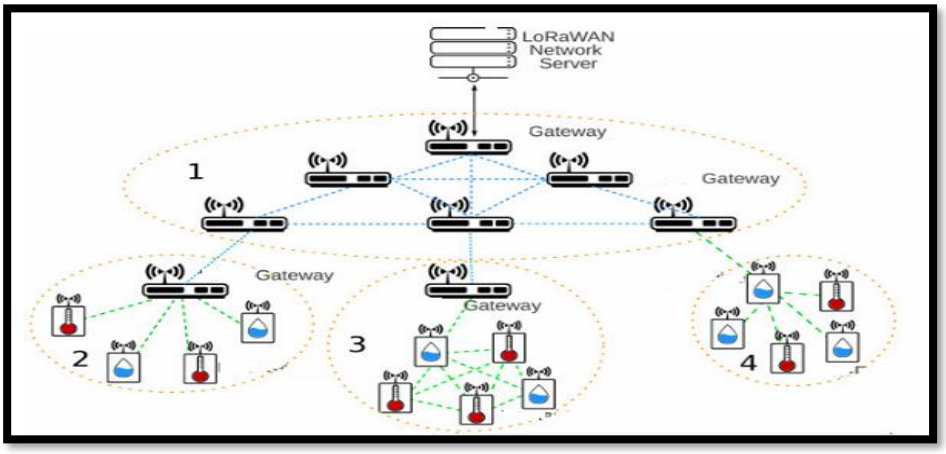
2.2.2.1. Protocolo LoRaWan (Long Range Wide Area Netwok)

Es una tecnología de red de largo alcance, maneja una tasa de datos a un bajo volumen, tiene comunicaciones bidireccionales y está pensada para dispositivos de bajo consumo de energía (Echevarría, 2023).

Es recomendable implementarlo en una tecnología de estrella, como se presenta en la Figura 2, en la que los nodos se comunican directamente con una estación base centralizada, llamado Gateway. Están conectados al servidor a través de conexiones IP y actúan como un puente llegando a una distancia en zonas rurales de 5-15 km y en zonas urbanas 1-5 km (Echevarría, 2023).

Figura 2

Topologia LoRaWAN



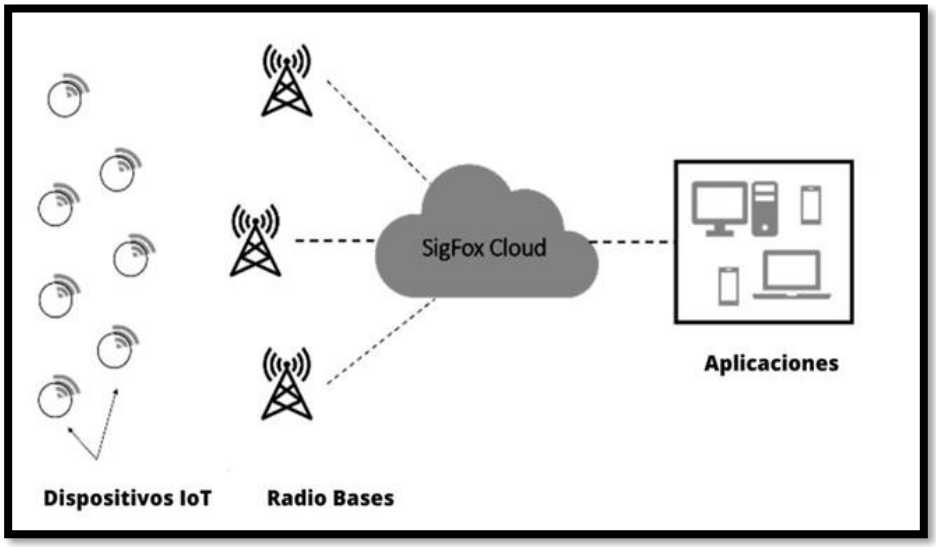
Nota: Tomado de Echevarría (2023)

2.2.2.2. Protocolo Sigfox

Es una red del Internet de las cosas que permite él envió de datos de manera eficiente a una distancia hasta de 10-50 km en campo abierto y de 3-10 km en zonas urbanas sin la necesidad de mantener conexiones de red permanentes. Los dispositivos que utilizan SigFox operan con bajo consumo de energía, lo que la conviene en una tecnología ecológica. Al transmitir paquetes de datos más pequeños, SigFox optimiza el uso de energía en comparación con otras tecnologías, lo que favorece a los dispositivos alimentados por baterías, permitiendo que tengan una mayor duración sin necesidad de recargas frecuentes (Molina,2024).

Figura 3

Arquitectura SigFox



Nota: Tomado de Molina (2024)

2.2.2.3. Protocolo NB-IoT (Narrowband IoT):

Diseñado con la red móvil 3GPP loque deriva a casi una conexión global para dar respuesta a las necesidades de comunicación IoT, en lo que denominan extended Machine Type Comunnications (eMTC). Utiliza como tecnología el Low Power Wide Area Networks (LPWAN) como lo son tecnologías como LoRaWAN, Telensa SigFox. Los LPWAN se caracterizan por ser tecnologías de largo alcance, dando una distancia de cobertura hasta 35 km en áreas rurales y 1-10 km en áreas urbanas (Rojas y Chate, 2022).

Figura 4

Arquitectura NB-IoT

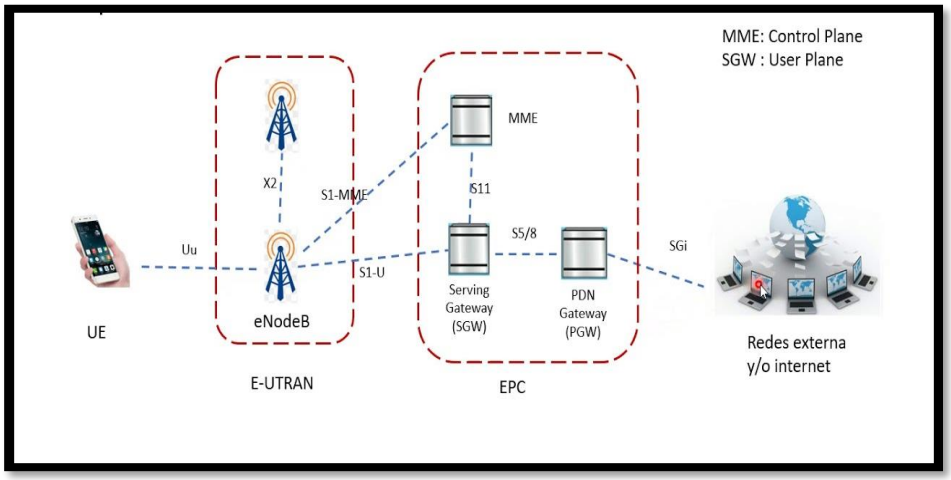
Nota: Tomado de Incebe-cert.

2.2.2.4. Protocolo LTE-M (LTE Cat M1)

La tecnología LTE-M (Lon Term Evolution Category M1) o EPS (Envolved Packet System) que contribuye a la reducción de costos y largo plazo de duración de batería, adicional al LTE-M acapara la mayoría de las especificaciones de redes LTE, acerca de seguridad y confidencialidad de información aprovechando la red 4G. Dicha red tiene como distancia en zonas rurales de 30 km y en zonas urbanas 1-10 km (Garzón y Reyes, 2023).

Figura 5

Arquitectura LTE



Nota: Tomado de YT- Arquitectura de una red LTE

En comparación con tecnologías como SigFox, LoRaWAN o NB-IoT, el uso combinado de NRF24L01 para comunicación local y SIM7600SA (4G) para la conexión en nube y servidor redujeron costos manteniendo una baja latencia en la trasmisión de datos.

2.2.3. Otro tipo de comunicación IoT

Existen otros tipos de conexión IoT aparte de las mencionadas que también aplican el sistema IoT que vendría a ser las siguientes

2.2.3.1. Redes de Area local (LAN)

- Wi-Fi: Utiliza el estándar IEEE 802.11. Utilizado para aplicaciones donde hay acceso a red Wi-Fi disponible.
- Zigbee: Basado en el estándar IEEE 802.15.4, Zigbee es un protocolo de comunicación para rede de área personal (PAN).

2.2.3.2. Red de área amplia (WAN)

- 5G: Al ser la quinta generación de redes móviles ofrece alta velocidad, baja latencia y gran capacidad para conectar varios dispositivos.
- GSM/3G/4G: Tecnología celulares que proporcionan conectividad en redes móviles. 4G LTE y sus predecesoras como GSM y 3G.

2.2.3.3. Conexión por radiofrecuencia (RF)

- Hz RF (433MHz, 868MHz, 915Mhz): Usado para comunicación de largo con bajo consumo en frecuencias no licenciadas. Con alcance favorable de 1km a 10km según el entorno
- Hz RF de corto alcance (2,4GHz) Al igual que la conexión de largo alcance, pero con la limitante de comunicación más corta, pero con más tasa de envió de información. Alcance de 100 a 300 metros.

2.2.3.4. Comunicación en la Nube

- MQTT (Message Queuing Telemetry Transport): Protocolo ligero de mensajería basado en publicación/suscripción, ideal para redes con ancho de banda limitado y alta latencia.
- CoAP (Constrained Application Protocol): Diseñado para dispositivos con recursos limitados y redes con alta latencia

2.2.4. Tipos de topología utilizables

- *Topología en estrella*

“Una topología en estrella es un sistema en el que los mensajes se transmiten en un solo salto y todos los nodos sensores se comunican directamente con la pasarela a una distancia de 30 a 100 metros. Todos los nodos sensores son los mismos nodos finales, y la pasarela recibe datos de todos ellos. Las pasarelas, también, se utilizan para el transporte de datos y la supervisión de la red. Los nodos finales no intercambian mensajes entre sí, sino que emplean pasarelas para comunicar la información que necesitan.” (Fierro,2023).

Figura 6

Topologia estrella

Nota. Topologías de redes-Alvaro Chirou

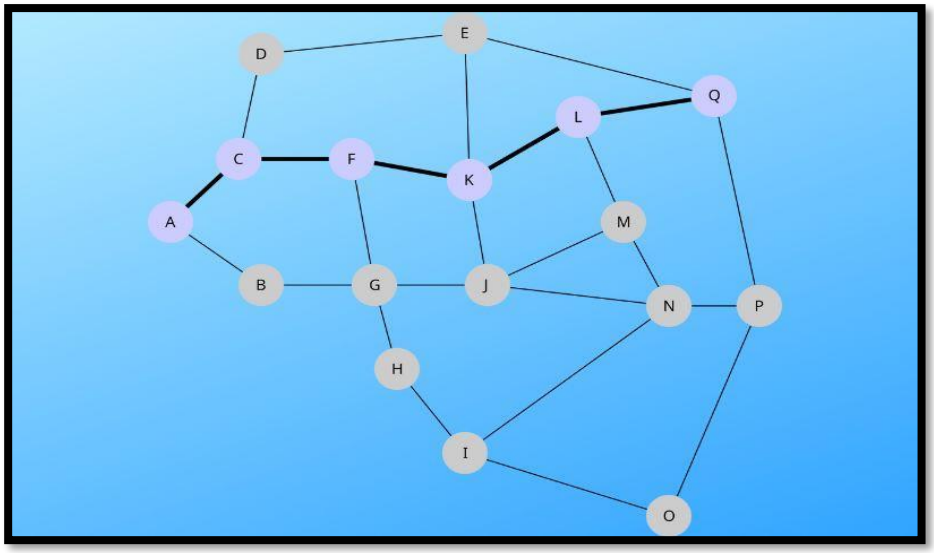
La topología en estrella es la que requiere menos energía, pero está limitada por la distancia de transmisión por radio entre cada nodo y la pasarela. Además, carece de un método de comunicación alternativo en caso de que una de las vías de comunicación de uno de los nodos se vea obstaculizada, con la consiguiente pérdida de información de ese nodo (Fierro, 2023).

- *Topologia en malla*

Cada nodo puede comunicarse y recibir datos de otros nodos y de la pasarela. A diferencia de la arquitectura en estrella, que sólo permite a los nodos conectarse con la pasarela, los nodos, también, pueden enviarse mensajes entre sí.

Figura 7

Tipo malla



Nota. Tomado de Lifeder - Kjerish, CC BY-SA 4,0

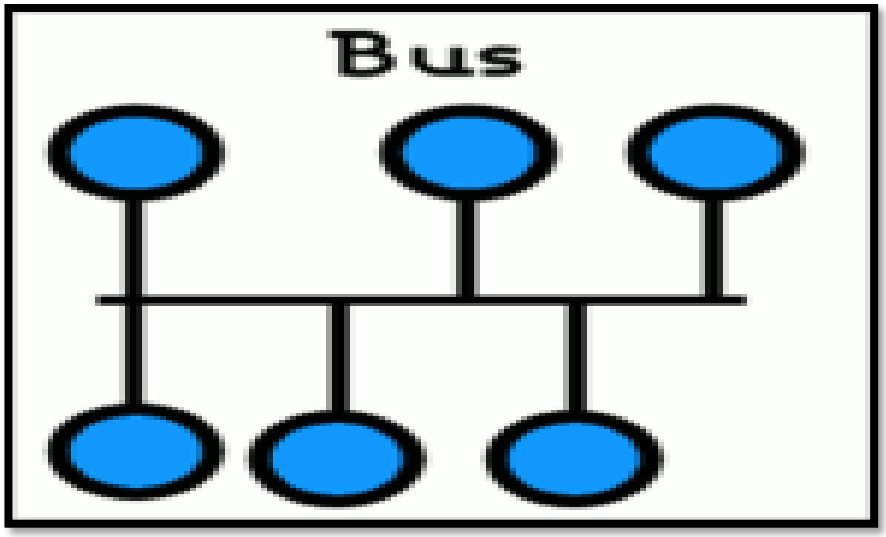
En principio, la difusión de datos de los nodos a las pasarelas permite a las redes escalar sin fin. Esta arquitectura es, también, muy resistente a los fallos porque cada nodo utiliza una ruta única para interactuar con la pasarela. Cuando falla un nodo, la red se reconfigura inmediatamente a su alrededor y, según el número de nodos y su distancia, la red puede tener tiempos de espera prolongados al enviar mensajes (Fierro, 2023)

- *Topología de bus:*

“Todos los dispositivos finales están conectados a un cable primario. se llama Bus porque los datos viajan por este cable. El inconveniente radica en que, cuando existía esta topología, usábamos el cable coaxial a velocidades de 10 Mbps, lo que se considera lento utilizando los estándares actuales” (Cardenas, 2024).

Figura 8

Topología bus



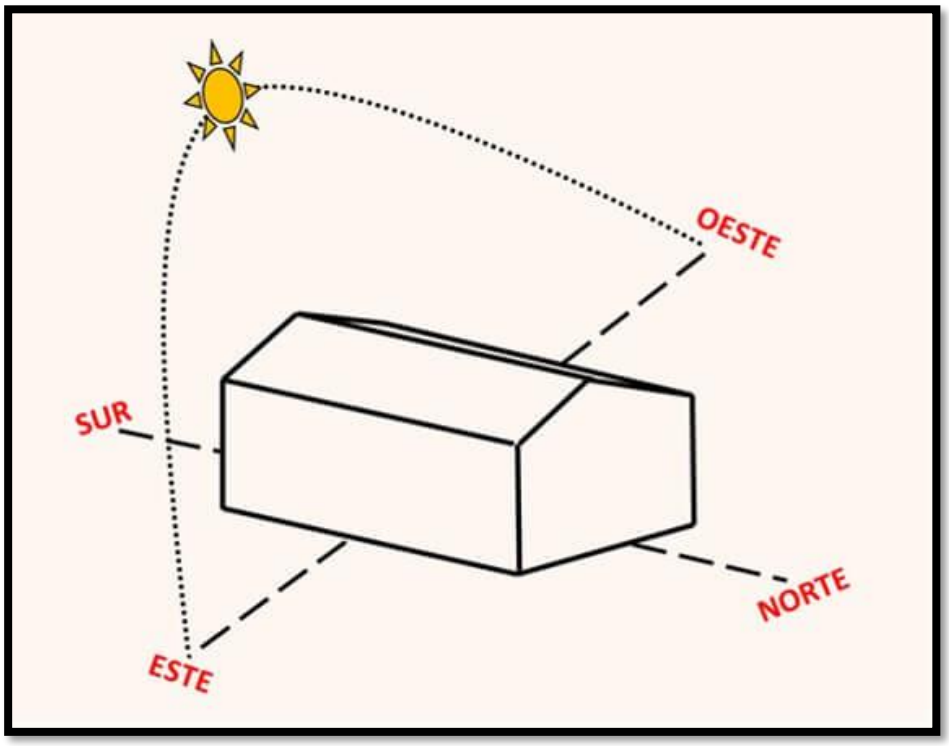
Nota: Tomado de redes inalámbricas y cableadas

2.2.5. Invernaderos

Comúnmente se conocen a los invernaderos de manera estándar con un gran terreno llano, agrupados, de forma de túnel y extensos colocados en columnas, existen varios tipos de invernaderos, va a depender de ciertos factores como: el terreno en donde será ubicado, el presupuesto para su elaboración, los materiales, los sistemas de control de temperatura, riego, humedad del aire y suelo, Ph del suelo y principalmente el producto que se va a sembrar ya que esto depende del tipo de invernadero (López y Toscano,2023). También hay que tener en cuenta la orientación de un invernadero para aprovechar la transmisión de la radiación dentro del invernadero, es recomendable orientarlo de Norte a Sur, permitiendo una distribución homogénea de la radiación solar (Figura 9).

Figura 9

Orientacion de Norte a Sur



Nota: Tomado de intagri.

2.2.6. Tipos de invernaderos

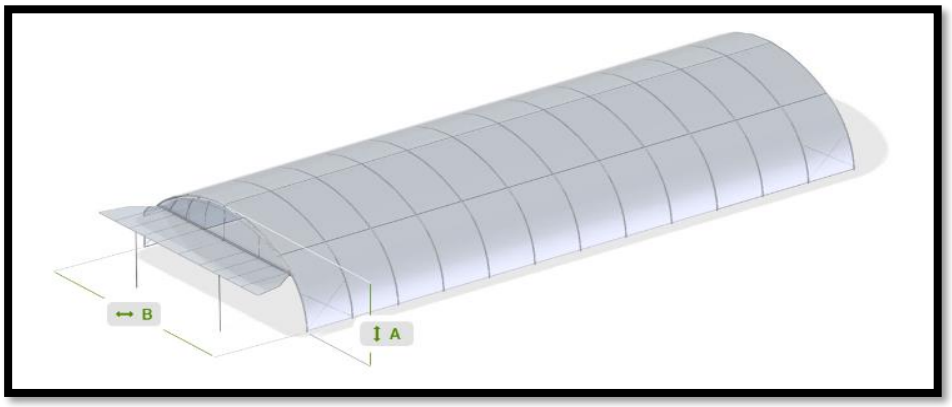
Cuando se refiere a tipos de invernaderos el factor resaltante que más se utiliza es la forma de su estructura, más específicamente en el techo, otros datos que puede variar son el material del que estará compuesto la estructura como metálicas o de madera y del material que cubre el invernadero ya sea de diferentes categorías de plásticos o de vidrio (López y Toscano,2023). Algunas de las estructuras más utilizadas en invernaderos son:

- *Tipo túnel*

El invernadero tipo túnel en la Figura 10, es un invernadero que se caracteriza por su estructura curva que va desde el suelo hasta la cumbrera, formado por arcos galvanizados. Es ideal para pequeñas superficies y cultivos de tamaño reducido, pues ofrece resistencia a los vientos y una alta transmisión de luz solar, lo que favorece el crecimiento de las plantas.

Figura 10

Invernadero Tipo túnel



Nota: Tomado de Ulmaagricola

- *Tipo asimétrico*

Este invernadero se diferencia del resto al tener una de los lados de la cubierta más inclinada que el otro, y su uso se centra en regiones de clima tropical (Figura 11).

Figura 11

Invernadero Asimétrico

Nota: Tomado de Novagric

- *Tipo capilla*

Se les conoce como capilla por su estructura similar a la de una capilla, este invernadero se distingue por su cubierta atrapando en un mayor espacio el clima obtenido por el sol

para su cubertura tiene un ángulo de 15 º o 35 º y una estructura metálica. Es una opción económica y no presenta dificultades en cuanto a la ventilación. Por lo tanto, es apto para materiales de cobertura flexible o rígida (Figura 12).

Figura 12

Invernadero Tipo capilla

Nota: Tomado de Serres-lafrancaise

2.2.7. Invernadero autónomo:

Un invernadero es una construcción en la agricultura. Usualmente, consiste en una estructura metálica cerrada, recubierta con una película plástica translucida (agrofilm), que permite el paso de la luz solar, bloqueando la lluvia, el viento y otros factores climáticos adversos que podrían afectar la producción. El propósito principal de un invernadero es recrear lo más fielmente posible las condiciones climáticas y óptimas para el cultivo de una planta especifica (Santander, 2023; url: https://goo.su/z2JDCNC). Además, los invernaderos presentan varias ventajas.

- *Eficiencia en los recursos:* Los invernaderos inteligentes eliminan tareas repetitivas como el riego o la ventilación, que requieren mucho tiempo y esfuerzo cuando se realizan manualmente, lo que también reduce considerablemente el costo de mano de obra.
- *Personalización según usuario o total:* Los programas que controlan los invernaderos autónomos se pueden ajustar completamente a las

características del cultivo y las necesidades del agricultor, sin versiones estándar, sino adaptadas a cada usuario.

- *Capacidad de respuesta:* Los sensores monitorean constantemente los datos, el agricultor siempre estará informado sobre el estado de sus cultivos, permitiéndole actuar de inmediato ante cualquier problema o dificultad.
- *Rentabilidad:* Al mantener condiciones óptimas para el crecimiento de los cultivos, se obtiene un producto de mayor calidad.

2.2.8. Tipos de automatización

- *Automatización del riego*

Los invernaderos autónomos se encargan de esta tarea por sí mismos, conllevando a que los agricultores valoran enormemente, ya que realizarla de forma manual requiere mucho tiempo y esfuerzo. Sin embargo, no solo se puede automatizar el riego, sino también aplicar fertilizantes y nutrientes (Santander, 2023; url: https://goo.su/z2JDCNC).

- *Control de humedad*

Un invernadero autónomo puede incorporar o no un sistema de nebulización para aumentar el nivel de humedad relativa en el ambiente. Esta es otra tarea que los invernaderos autónomos realizan de manera automática (Santander, 2023; url: https://goo.su/z2JDCNC).

Figura 13

Control de humedad

Nota: Sistema de nebulización en interior de invernadero

- *Sistemas de CO2:*

  El dióxido de carbono es crucial para que las plantas lleven a cabo la fotosíntesis. En un invernadero cerrado, los cultivos no pueden obtener la cantidad adecuada de este gas de forma natural, por lo que es necesario proporcionarlo de manera artificial (Santander, 2023; url: https://goo.su/z2JDCNC).

  Los invernaderos autónomos tienen la capacidad de monitorear los niveles de dióxido de carbono en el ambiente y, de acuerdo con los datos obtenidos, pueden liberar más de este gas o activar la ventilación para disminuir su concentración si es necesario (Santander, 2023; url: https://goo.su/z2JDCNC).

2.2.9. Arduino

Arduino es una plataforma de código abierto, con el fin de diseñar distintos tipos de proyectos o prototipos con los distintos modelos que existen en el mercado, como UNO, DUE, NANO, YUM, LEONARDO, entre otros. Cada modelo tiene distintos y potentes microcontroladores diseñados e integrados en sus placas que permiten la conexión de sensores, actuadores (Chasi, 2023)

2.2.10. IDE Arduino

En la página de Arduino se encuentra el software gratuito llamado IDE Arduino, el cual permite escribir, depurar, editar y grabar los programas llamados “bocetos” o “sketch”, de una forma muy sencilla (López y Toscano, 2023).

Como en la página de Arduino en la tienda de tarjetas, nos dice “Las posibilidades de realizar desarrollos basados en Arduino tienen como limite la imaginación (Arduino.cl s.f.).

2.2.11. Riego por goteo

Este tipo de riego es el más eficiente método de suministro de agua y nutrientes a los cultivos (Figura 14). Entrega el agua y fertilizantes directamente a la zona radicular de cultivo, en una cantidad correcta y en el momento adecuado, cada planta recibe exactamente lo que necesita, cuando lo necesita para desarrollarse óptimamente. Este tipo de riego logra que los productores pueden tener mejores rendimientos mientras ahorran agua, así como fertilizantes, energía e incluso agroquímicos.

Figura 14

Cultivo por goteo

Nota: Tomado de Compo Expert

El agua y los nutrientes se reparten en el campo usando unas tuberías llamadas “mangueras de goteo”, que llevan pequeños dispositivos conocidos como “goteros”. Estos goteros sueltan pequeñas gotas de agua mezcladas con fertilizantes, lo que garantiza que cada planta reciba justo lo que necesita, directamente en sus raíces, de manera uniforme en todo el campo.

- *Estas son algunas de las ventajas del riego por goteo:*
    - Mayor rendimiento y calidad constante: Las cosechas son más abundantes y la calidad del producto se mantiene alte de manera constante.
    - Ahorro significativo de agua: No se pierde agua por evaporización, escurrimientos o desperdicios.
    - Riego homogéneo en cualquier terreno: No importa si el suelo o el terreno son irregulares, el riego se distribuye de manera uniforme.
    - Menor consumo de energía: El sistema de riego por goteo requiere menos presión, lo que ahorra energía.
    - Uso optimizado de fertilizantes y agroquímicos: estos se aprovechan al máximo sin filtrarse a las capas freáticas del suelo.
    - Previene la acumulación de sales: Evita que se acumulen sale en el suelo, algo que suele ocurrir cuando se usan demasiados fertilizantes.
    - Protección del follaje: Como el follaje no se moja, se reducen las probabilidades de enfermedades causadas por hongos.

2.2.11.1. Tipos de riego por goteo

El sistema por goteo por definición es un riego localizado que, como su nombre lo sugiere, consiste en liberar agua gota a gota (Figura 15). Es un método muy eficiente que aplica pequeñas cantidades de agua de manera regular, directamente cerca de las raíces de las plantas.

- *Riego por goteo, superficial*

El funcionamiento de este sistema se encarga de suministrar agua directamente a la base de las plantas en pequeñas cantidades, lo cual se realiza mediante goteros instalados en las tuberías que distribuyen el agua en tu jardín o huerto. Este sistema es muy eficiente porque utiliza solo la cantidad exacta de agua que las plantas necesitan, evitando desperdicios.

Figura 15

Riego superficial (estándar)

Nota: Tomado de Rivulis

Ideal para: Lugares donde el agua es un recurso escaso, como zonas o regiones de sano, áridas, o en las que hay sequía. Supone una inversión, pero a largo plazo un sistema de riego por goteo puede ser más económico si ahorras en agua y tiempo.

- *Ventajas:*
    - Es el más eficiente en su bajo gasto de agua
    - Sistema fácil de automatizar

- Reduce la probabilidad de encharcamiento y plagas porque no genera exceso de humedad
- No alimenta a las malas hierbas

- *Desventajas:*
  - Requiere un poco de conocimientos técnicos para ser montado
  - Inversión inicial de dinero, la cual puede ser recuperada
  - Los goteros pueden obstruirse y causar problemas
  - Una vez lo instalas es más difícil manipular el terreno

- *Riego por exudación:*

  El agua se bombea a las raíces de las plantas a través de una tubería o manguera porosa que se entierra a una profundidad de aproximadamente 8 a 14 centímetros. Este sistema de riego que permite humedecer sin problemas grandes superficies, y además evitar encharcamiento.

Figura 16

Riego por Exudación

Nota: Tomado de Agro Huerto

Ideal para: Quienes buscan un sistema de riego eficiente, más fácil de instalar y que no dificulte tanto la manipulación del terreno antes de la instalación, debido a que la manguera se encuentra bajo tierra.

Ventajas:

- Tiene la posibilidad de cambiar plantas y cultivos con facilidad aun posteriormente a la instalación del sistema de riego por exudación.
- Mas resistencia a dañarse debido a estar bajo la tierra, pero con el tiempo límite de vida del material.

Desventajas:

- Inversión inicial, con el tiempo recuperable.
- Mantenimiento, ya que el material se encuentra enterrado, ocurre ciertos bloqueas por estar bajo tierra debido a depósitos de sales.
- El costo de reemplazo es mayor, por su extracción e instalación.

- *Gotero regulable*

Es un sistema que permite ajustar fácilmente el flujo de agua girando la tapa (Figura 17). Estos goteros están diseñados con topes inferiores que evitan que se desenrosquen accidentalmente, lo que segura que el riego se adapte a las necesidades específicas de cada planta. Al suministrar el agua, también se distribuyen los nutrientes esenciales directamente a las raíces, lo que resulta en cultivos más productivos y de mejor calidad.

Figura 17

Gotero regulable

Nota: Tomado de Jaén Clima

*Ideal para***:** La agricultura como para la jardinería, y se puede utilizar para regar árboles, arbustos, macetas y flores.

Ventajas:

- Debido a que regula el caudal de agua permite usar solo la cantidad necesaria, reduciendo el desperdicio.
- Al ajustar el flujo de agua, también se controla la cantidad de nutrientes que llegan a las raíces, lo que mejora la calidad y cantidad de los cultivos.
- Tiene una buena adaptabilidad, lográndose ajustar para distinto tipos de plantas, lo que lo hace ideal tanto para jardines como para huertos, permitiendo n riego específico para cada planta.

Desventajas:

- Costo inicial un poco más alto, por la adición de los goteros regulables a la manguera de riego y mayor cuidado de no dañar el material.
- Requiere una revisión periódica para asegurarse de que los goteros no estén obstruidos por agentes externos.
- Necesita de una presión de agua adecuada para funcionar correctamente, si la presión es baja, el riego puede ser ineficiente.

2.2.12. Orégano

El orégano aparte de ser una fuente principal de ingresos en el sector de Tacna (Tabla 1) también tiene otros distintos tipos de uso para la salud, transformación del orégano en aceite, entre otros. Por ende, se debe tener un cuidado meticuloso para su correcto cultivo y factible producción natural, con la ayuda de un invernadero no solo facilitara su cultivo si no obtendremos de buena calidad con poco uso de pesticidas. Para su correcto cuidado necesitaremos tener en cuenta estos puntos.

- Tiene que tener una temperatura ideal entre 15 a 30 grados, debido a que es una planta tolerable a climas difíciles logrando ser tolerable de los 0 a 41 grados reduciendo su crecimiento.
- La calidad del suelo en Ph entre 5 y 8 puede tolerar la alcalinidad.
- También necesitara la humedad relativa ambiental entre 40% y 70% y en la humedad del suelo necesitara alrededor de entre 60% y 70%.

- La tasa de CO2 óptimas para distintas plantas incluyendo el orégano son de concentración de 400 – 800 ppm (partes por millón) y cuando la planta se encuentra en un estado maduro oscila entre 800 – 1000 ppm favoreciendo la fotosíntesis son las estimadas para un buen crecimiento generalmente se mantiene en un rango de 400 a 1500ppm.

Tabla 1

Sector agropecuario parte agrícola (Miles de toneladas)

Subsectores	Estructura % 2022 2/	Abril				Ene-Abr			
		2023	2024	Var. %	Contribución al crecimiento	2023	2024	Var. %	Contribución al crecimiento
Sector agrícola	81,8	—	—	-21,5	-17,5	—	—	-20,8	-15,9
Orientada al mercado externo y agroindustria 2/	65,2	—	—	-29,6	-22,7	—	—	-32,2	-20,2
Cebolla	1,0	2,9	6,0	106,5	3,0	7,8	15,6	100,4	2,6
Granada	0,0	0,0	0,0	-	-	0,2	0,2	-17,6	0,0
Olivo	53,8	10,3	2,0	-80,7	-31,1	20,4	3,5	-83,0	-21,5
Orégano	8,1	2,2	3,2	49,3	9,4	3,3	4,0	21,6	2,2
Sandía	0,8	1,7	1,3	-21,2	-0,2	10,6	15,3	44,2	1,1
Vid	1,3	0,1	0,4	196,0	0,6	6,7	4,4	-33,7	-1,9

Nota. Tomado de Banco central de reserva del Perú

2.3. Definición de términos

2.3.1. Arduino

Arduino es una placa basada en un microcontrolador Atmel, para su funcionamiento se tendrá que ver un lenguaje de programación que pueda ser utilizado en su software Arduino (Arduino, 2024).

2.3.2. NRF24L01

Es un módulo transceptor RF (emisor y receptor) de 2,4GhZ NRF24L01 es ideal para comunicar proyectos de forma inalámbrica, opera en la banda de 2,4GHz (Industrial, científica y Medica), (Nordic Semiconductor. (2008)).

2.3.3. ESP32

¡Es un sistema robusto con en un chip SoC (System On Chip) con Wi-Fi y modo dual con Bluetooth! En el fondo, hay un microprocesador Tensilica Xtensa LX6 de doble núcleo o de un solo núcleo con una frecuencia de reloj de hasta 240MHz (Espressif Systems, 2023).

2.3.4. Invernadero automatizado

Al estar automatizado un invernadero tiene la capacidad de monitorear y controlar de forma automática las condiciones ambientales internas, con el fin de optimizar el crecimiento de las plantas (FAO. (2020)).

2.3.5. Microcontrolador

Circuito integrado que contiene un procesador, memoria y periféricos de entrada/salida, capaz de ejecutar programas para controlar dispositivos electrónicos (Mazidi & Naimi, 2024).

2.3.6. Actuador

Dispositivo mecánico o eléctrico que recibe una señal proveniente de un controlador (Arduino) y ejecuta la acción física según lo programado, como abrir una válvula, encender un ventilador o activar un motor (Jazar, 2023).

2.3.7. Relé

Interruptor electromecánico que permite una señal de baja potencia (salida de un Arduino o microcontrolador) controle dispositivos de alta potencia, como bombas de agua o sistemas de iluminación (Platt, 2024).

2.3.8. Sensor pH

Dispositivo utilizado para saber la calidad del líquido a medir según sea alcalino o ácido, 7 es neutro, 8 a 14 a medida que sube se hace más alcalino y de 6 a 1 según baje es más ácido (Karimi, 2023).

2.3.9. Sensor de temperatura y humedad

Dispositivo utilizado para medir la temperatura y humedad en el ambiente. Envía datos al controlador para que se puedan tomar decisiones sobre la activación de los sistemas de ventilación o riego (Espressif Systems, 2024).

2.3.10. Sensor de humedad del suelo

Como su nombre lo indica mide la humedad del suelo. Este dato es crucial para controlar el sistema de riego de las plantas dentro del invernadero (Saini & Marques, 2023).

2.3.11. Sensor de CO_2

Este dispositivo es el encargado de medir la cantidad de CO2 en el ambiente para que la planta se encuentre en buen estado y tenga una buena condición ambiental favoreciendo el cultivo (Winsen Electronics, 2025).

2.3.12. Sensor ultrasonido

Utiliza el ultrasonido para determinar la distancia de un objeto o liquido dependiendo de la distancia a la que este del dispositivo. Además, permite medir niveles de agua o detectar obstáculos sin contacto físico, ofreciendo alta precisión y fiabilidad en distintas aplicaciones (Hughes, 2023).

CAPITULO III: MARCO METODOLÓGICO

3.1. Diseño de la investigación

La investigación desarrollada es del tipo aplicativa. Para su desarrollo se tomó el sistema IoT habilitado por Arduino, el cual se identifican los requerimientos del proceso, el uso de los conocimientos investigados y los resultados que se obtuvieron para el diseño de un invernadero inteligente que permita aumentar los índices de productividad mediante un mejor control de los recursos (agua, electricidad).

Para la parte tecnológica se seleccionaron sensores compatibles con dichos microcontroladores, se utilizaron relés para controlar la potencia de los actuadores. Se creo una interfaz de usuario a manera de consola en donde los usuarios podrán visualizar los datos que envíen los sensores y así poder manipular los actuadores.

3.2. Acciones y actividades

3.2.1. Planeación

- Se desarrollo el diseño de un prototipo basado en microcontroladores, que permita la adquisición de datos de un sensor, a través de canales de comunicaciones analógico y digitales para su procesamiento.
- El prototipo está constituido de hardware (sensores, actuadores) y de un software (nivel lógico, procesamiento y almacenamiento) para su funcionamiento, se utilizó el software Arduino Cloud para recibir y emitir datos del invernadero.

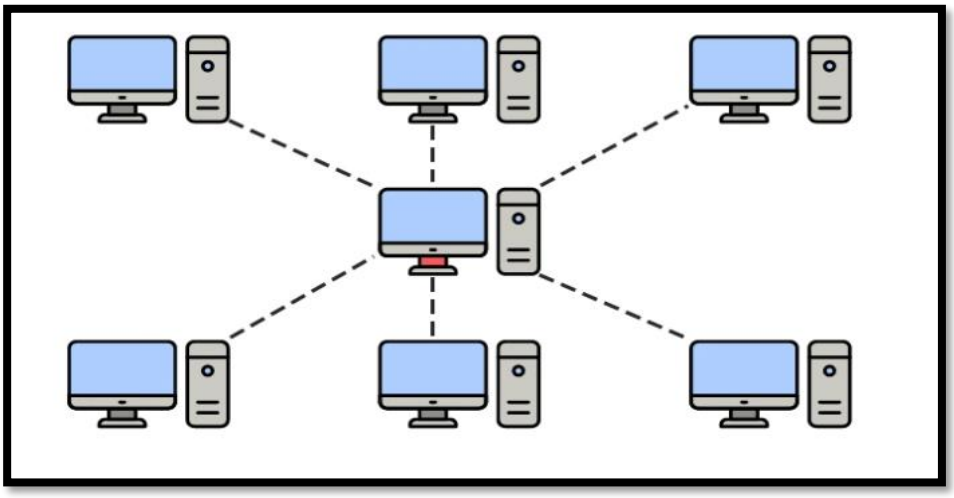
3.2.2. Funcionalidad

- El prototipo tiene una conexión red mediante la nube utilizando el modelo estrella (Figura 18), para la monitorización de los datos enviados por el microcontrolador y ser manipulados de ser necesario.
- La conexión y envió de datos entre los sensores con el Arduino será vía nodos (Figura 19) para la recepción de datos de temperatura, humedad ambiente, humedad del suelo, nivel del agua, CO2, alcalinidad o acidez (Ph).

- Las variables medibles del invernadero son humedad (aire y suelo, HR), temperatura (ºC), Dióxido de carbono (CO2), nivel del agua (cm), alcalinidad (Ph).

Figura 18

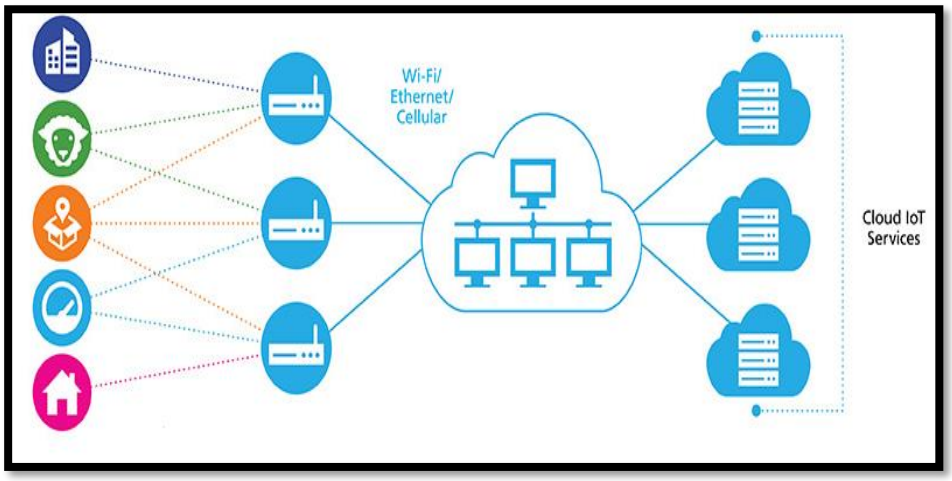
Topología estrella



Nota: Tomado de topologiasdered

Figura 19

Nodos envío y recepción de datos



Nota: Tomado de learn.semtech

3.2.3. Sistema de control

En las figuras 20 al 22 se observan los sistemas de control.

Figura 20

Sistema de control

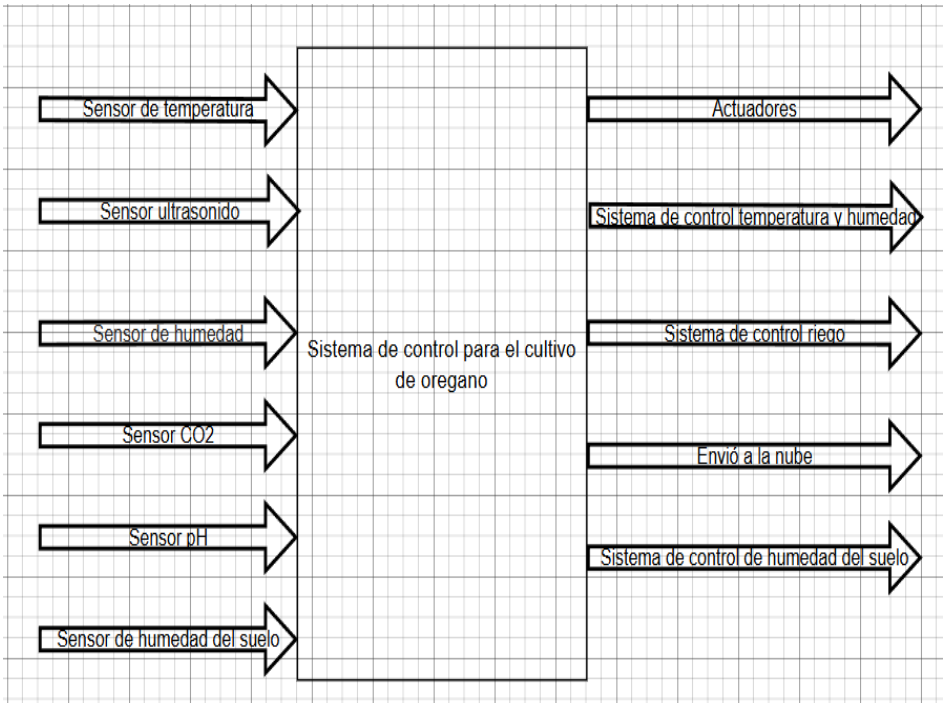


Figura 21

Sistema de control detallado

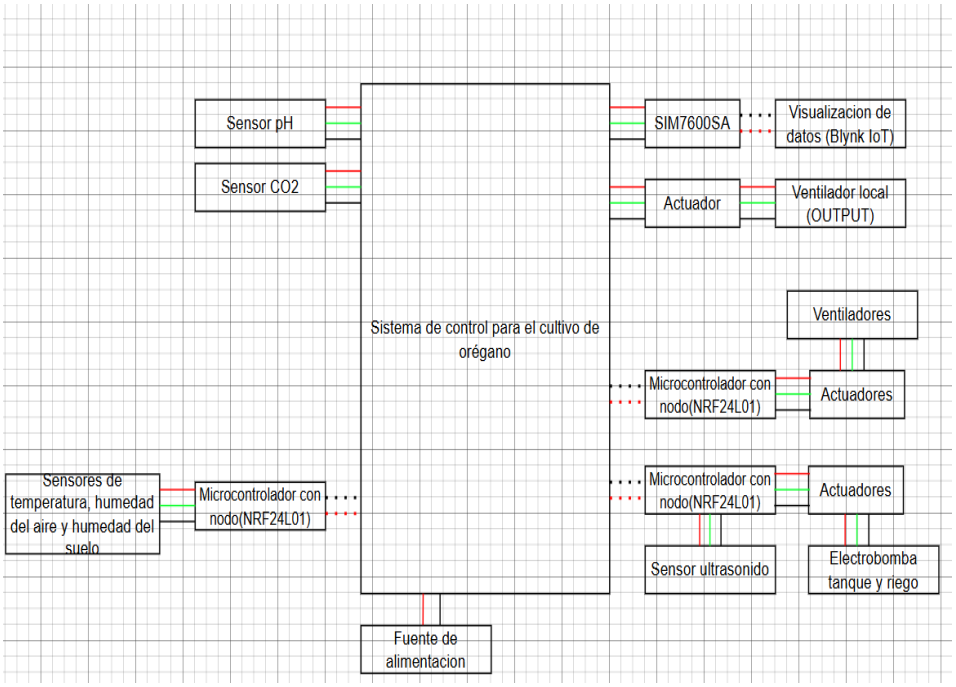
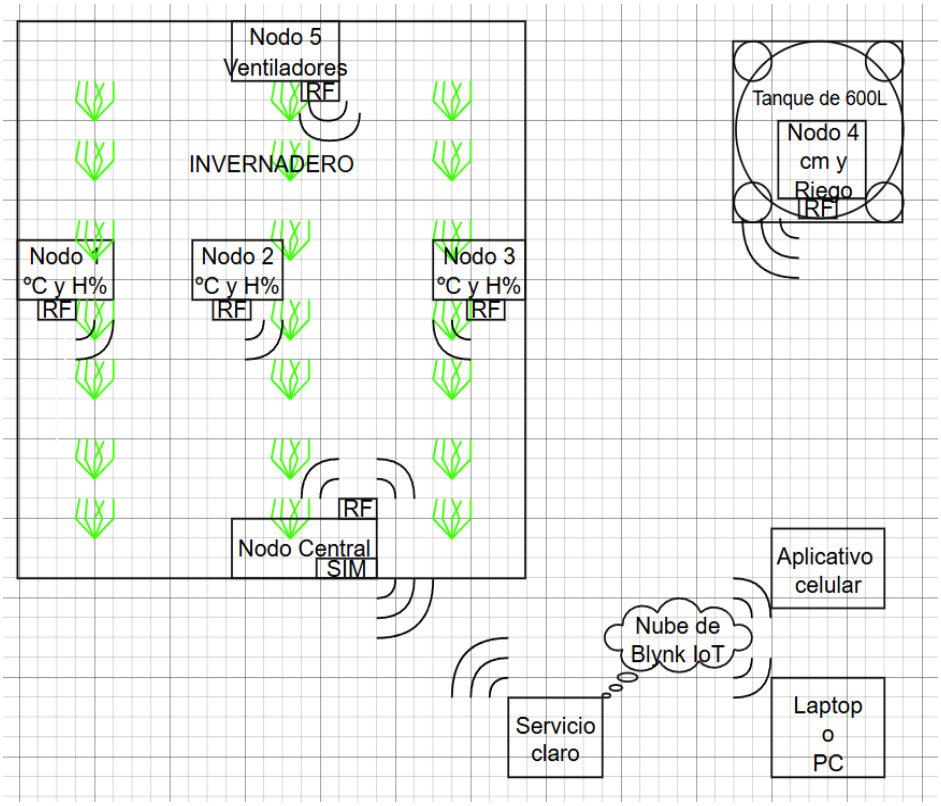


Figura 22

Sistema general



3.3. Materiales e instrumentos

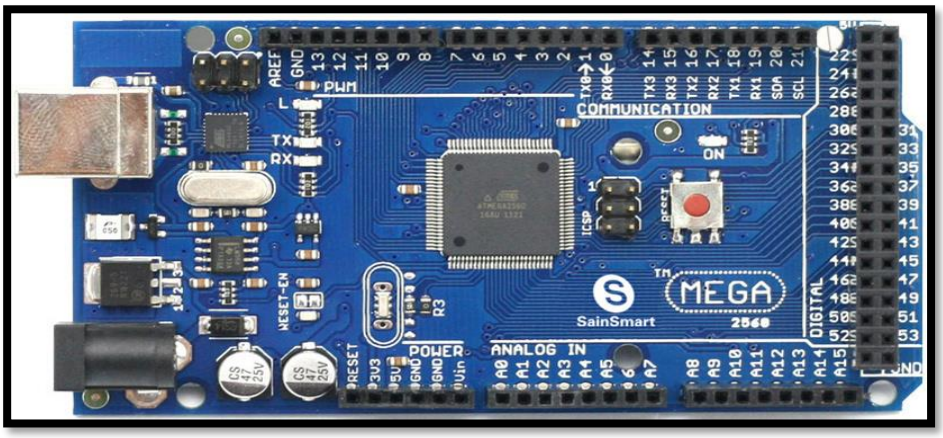
3.3.1. Materiales

3.3.1.1. Arduino Mega 2560

Es una placa de microcontrolador encargado de procesar datos de las salidas analógicas. Tiene pines de entrada/salida digital, entradas analógicas, tiene un modelo bastante robusto, lo que hace capaz de recibir mucha información (Figura 23) según la robustes del proyecto e indicaciones que se requieran para la aplicación, será el encargado de acoger los datos y enviarlos al usuario destinado.

Figura 23

Modelo Arduino Mega 2560



Nota: Tomado de SAISAC Mecatrónica

Tabla 2

Especificaciones técnicas Arduino Mega 2560

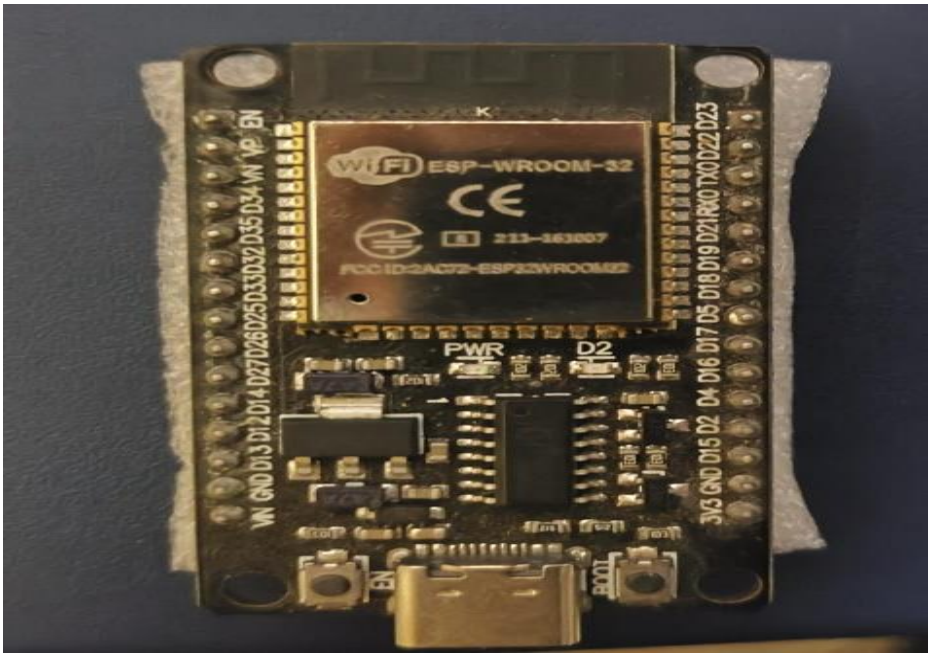
Arduino Mega 2560	
Voltaje de operación	5V – 20V DC
Pines digitales I/O	54 (15 salidas PWM)
Entradas analógicas	16 (ADC 10-bit)
Memoria FLASH	256KB
Memoria SRAM	8KB
Frecuencia de reloj	16MHZ
Consumo de corriente	50 – 75 mA

3.3.1.2. Microcontrolador ESP32

Es un potente módulo que viene integrado con WI-FI y Bluetooth (Figura 24), es ideal para desarrollar IoT, tiene una compatibilidad con Arduino al 100 %, logrando funcionar con el software Arduino IDE y el lenguaje de programación “C++”, tiene un reloj de frecuencia ajustable de 50MHz a 240MHz, esto nos ayuda a poder conectarle diversos sensores o dispositivos, sin que logre saturarse.

Figura 24

Microcontrolador ESP32



Nota: Tomado de fuente propia

Tabla 3

Especificaciones técnicas ESP32

Microcontrolador ESP32	
Voltaje de operación	4V – 12V DC
Pines digitales GPIO	24
Pines PWM	16
Pines analógicos	18
Desempeño	600 DMIPS (millones de instrucciones por segundo)
Frecuencia de reloj	50 – 240 MHz
Consumo de corriente	50 – 240 mA

3.3.1.3. Modulo RF 2.4GHz PA+LNA con antena

El chip Nordic nRF24L01+ (Figura 25) integra un completo transceptor de 2,4Ghz, Sintetizador RF y lógica con el mejorado ShockBurst™ (acelerador de protocolo por

hardware para comunicación por SPI con microcontrolador). Esta versión de módulo posee además un circuito amplificador de potencia (PA), un circuito amplificador de bajo ruido (LNA) además de una antena SMA que le permite lograr un rango de hasta 1000m en campo de vista.

Figura 25

NRF24L01 PA+LNA

Nota: Tomado de naylampmechatronics

Tabla 4

Especificaciones técnicas módulo NRF24L01

Modulo RF 2.4GHz +PA+LNA con antena	
Voltaje de operación	1.9 - 3.6V DC
Potencia de transmisión	250 kbps (-104 dBm)
Sensibilidad de recepción	≤-95dBm (a 1 Mbps)
Consumo de corriente	115 mA

3.3.1.4. Módulo de conexión a red SIM7600SA (4G)

Es una placa Shield 4G (Figura 26) que tambien puede utilizar las tecnologías 3G y 2G con GSM la cual se encarga de conectar los proyectos con Arduino a la nube y enviarlo

a la red celular, de esta forma poder enviar y recibir datos, con la conexión 4G podrá conectarse a internet adentrándose al mundo del Internet de las cosas (IoT).

Con esta tecnología se puede acceder a la red con una SIM, sin la necesidad de un enrutador, haciéndolo factible en términos de bajo costo y movilidad.

Figura 26

Módulo 4G SIM7600SA E-L1C

Nota: Tomado de aliexpress

Tabla 5

Especificaciones técnicas SIM7600SA

SIM7600SA	
Voltaje de operación	5 – 10 V DC
LTE-FDD	B1 / B3 / B5 / B7 / B8 / B20
LTE-TDD	B40
Tecnología 4G	(Movistar/Claro/Entel/Bitel)
Transmisión GSM	850/900/1800/1900MHz
Temperatura de trabajo	-40ºC a +85ºC
Consumo de corriente pico (red)	2A
Consumo de corriente típica	200mA
Nivel de potencia CSQ	+31 (-51 dBm)

El SIM7600SA junto a una antena omnidireccional imantada (Figura 27) amplificando su señal, logrando llegar a 30 dBm (1W de potencia) en potencia de transmisión Tx Power y Rx lo que sería favorable para la conexión y con buena señal para la transmisión y recepción de datos.

Figura 27

Antena omnidireccional

Nota: Tomado de Aliexpress

Tabla 6

Especificaciones técnicas de Antena Omnidireccional

Antena Omnidireccional	Datos
Ganancia	12dB
Ancho de banda	2000 MHz
Rango de frecuencia	700 – 2700 MHz / 3G / 4G
5G	5100 – 5850 MHz
Conector	SMA macho

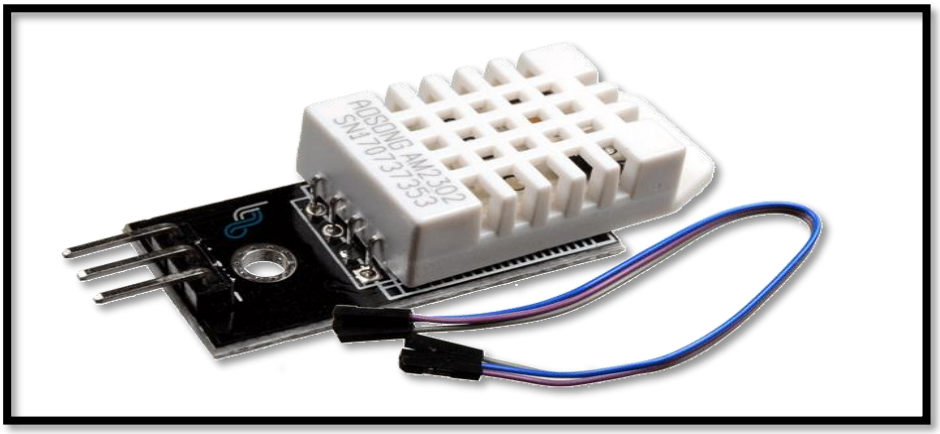
3.3.1.5. Sensor de temperatura y humedad:

El sensor DHT22 utilizado en el proyecto (Figura 28) es utilizado por su alto nivel de precisión, confiabilidad, capacidad y durabilidad. Este sensor será el encargado de

brindar la temperatura ambiente (ºC) y la humedad ambiente (%HR) en el interior del invernadero.

Figura 28

Modelo de sensores de temperatura y humedad



Nota: Tomado de nanoparuro

Tabla 7

Especificaciones técnicas de DHT22

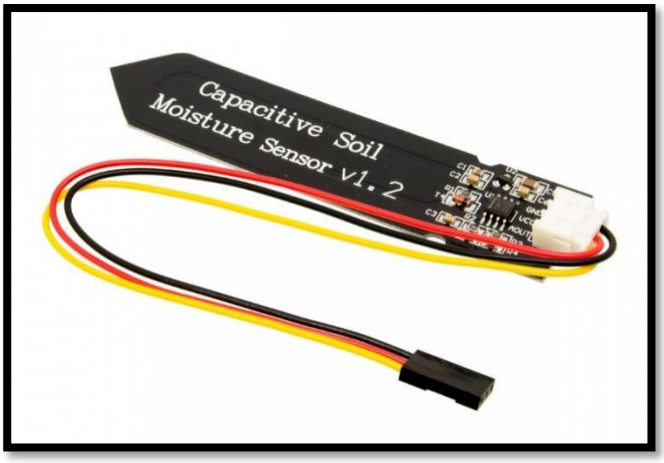
DHT22	Datos
Voltaje de operación	3 – 6 V DC
Rango de medición de temperatura	-40ºC a 80ºC
Precisión de medición de temperatura	<±0.5ºC
Rango de medición de humedad	0 a 100% HR
Precisión de medición de humedad	2% HR
Temperatura de trabajo	-40ºC a +85ºC
Corriente de operación	2.5mA

3.3.1.6. Sensor de humedad del suelo – Capacitivo

El Sensor de humedad del suelo o también llamada Higrómetro (Figura 29) es utilizado para medir la humedad relativa del suelo (HR) será el encargado de enviar datos que tan húmedo este el suelo para la irrigación del cultivo o el cuidado del mismo evitando que las raíces se corroan o se sobrecarguen de agua.

Figura 29

Modelo de sensor de humedad del suelo



Nota: Tomado de SAISAC Mecatrónica

Tabla 8

Especificaciones técnicas del sensor capacitivo v1,2

Sensor de humedad del suelo capacitivo v1.2	Datos
Voltaje de operación	3.3 – 5 V DC
Consumo de corriente	5mA
Voltaje de la señal de salida	0 – 5V (Analógico)
Temperatura de trabajo	-20 ºC a 60 ºC

3.3.1.7. Sensor de CO2-MH-Z19B:

Es un sensor de gases peligrosos (Figura 30) utilizado para el control de la calidad del aire y es adecuado para la detección de CO2. No proporciona valores absolutos, sino que simplemente proporciona una salida analógica que se debe de monitorear y

comparar con los valores de umbral requeridos, para asi saber el estado en el que se encuentra en aire, tanto para personas y otros seres vivos incluyendo a las plantas.

Figura 30

Módelo de sensor de MH-Z19B

Nota: Tomado de Electronic Clinic

Tabla 9

Especificaciones técnicas MH-Z19B

Sensor de Gases – CO2	
Voltaje de operación	3.6 – 5.5 V DC
Consumo de corriente	<18mA
Rango de medición	0-5000ppm
Precisión de medición	5% (50ppm)
Temperatura de trabajo	0 – 50ºC
Principio de medición	NDIR

3.3.1.8. Sensor ultrasonido – HC-SR04

Mide con precisión la distancia de objetos o fluido (agua) (Figura 31), con un rendimiento bastante estable. Módulo de alta precisión, tiene puntos ciegos (2cm) super cerca. Tiene emisor y receptor piezoeléctricos cada piezoeléctrico emite 8 pulsos de ultrasonido (40KHz). Debido a que trabaja a 40KHz no puede traspasar el agua, logrando una medición exacta y para poder hallar la distancia en dicho componente se hace mediante una formula.

$$(duracion * 0.0343)/2.0 \quad (1)$$

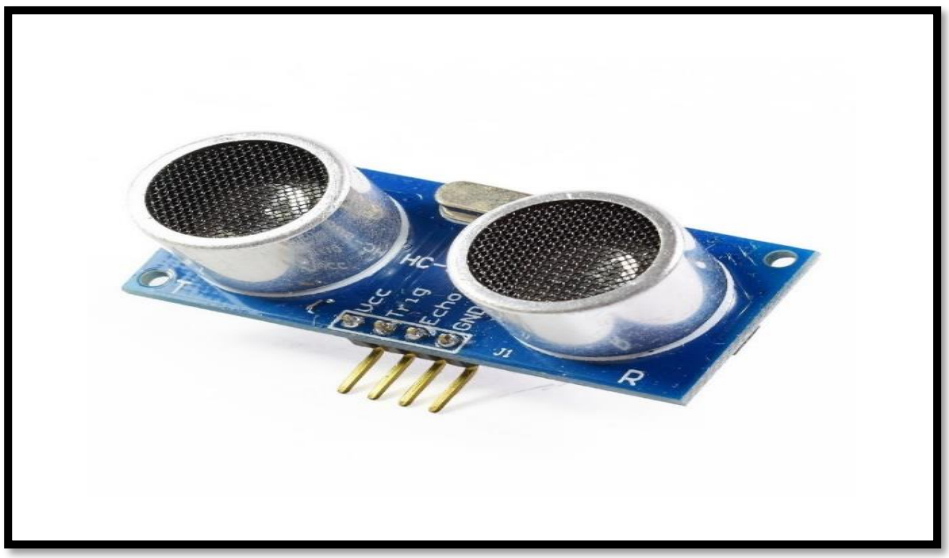
Duración: Es el tiempo que tarda la señal ultrasónica en ir y volver, medida en microsegundos (us)

Los 0.0343 están medidos en cm/us, debido a que la velocidad real del sonido es a 343 m/s

Este sensor será utilizado como sensor de nivel de agua, debido a que los rayos ultrasónicos no traspasan el agua.

Figura 31

Módulo de sensor HC-SR04



Nota: Tomado de naylampmechatronics

Tabla 10

Especificaciones técnicas HC-SR04

Sensor ultra sonido HC-SR04	Datos
Voltaje de operación	5 V DC
Consumo de corriente	15mA
Rango de medición	2 – 450 cm (4.5m)
Precisión de medición	±3mm
Angulo de apertura	15º
Frecuencia de ultrasonido	40KHz

3.3.1.9. Sensor Ph

Este sensor se basa en la medición del grado de acidez o alcalinidad de una sustancia o una solución (Figura 32). Se puede medir en una escala de 0 a 14. Un valor pH de 7 es neutro, lo que significa que la sustancia o solución no es acida ni alcalina. Este sensor nos indicará el nivel de alcalinidad que tendrá el líquido para mejorar lo suministrado al cultivo, debido a que es un sensor es sensible, no puede estar sumergido por mucho tiempo.

Figura 32

Módulo sensor pH

Nota: Tomado de Tesla Electronic EIRL

Tabla 11

Especificaciones técnicas de Sensor pH

Sensor pH – alcalinidad o acidez del agua	Datos
Voltaje de operación	5 V DC
Consumo de corriente	5 - 10 mA
Rango de medición	0 – 14 pH
Precisión de medición	0 – 60ºC
Temperatura de trabajo	±0.1 pH (25ºC)

3.3.1.10. Servomotor

El actuador utilizado MG996R (Figura 33) destaca por su buen torque (11Kg), engranajes metálicos y gran robustez. Es utilizado principalmente en proyectos de robótica, como brazos robóticos y robots bípedos. Puede rotar aproximadamente 180 grados (90° en cada dirección). Tiene la facilidad de poder trabajar con diversidad de plataformas de desarrollo como Arduino, PICs, Raspberry Pi, o cualquier microcontrolador.

Figura 33

Servomotor

Nota: Tomado de naylampmechatronics

Tabla 12

Especificaciones técnicas de servomotor

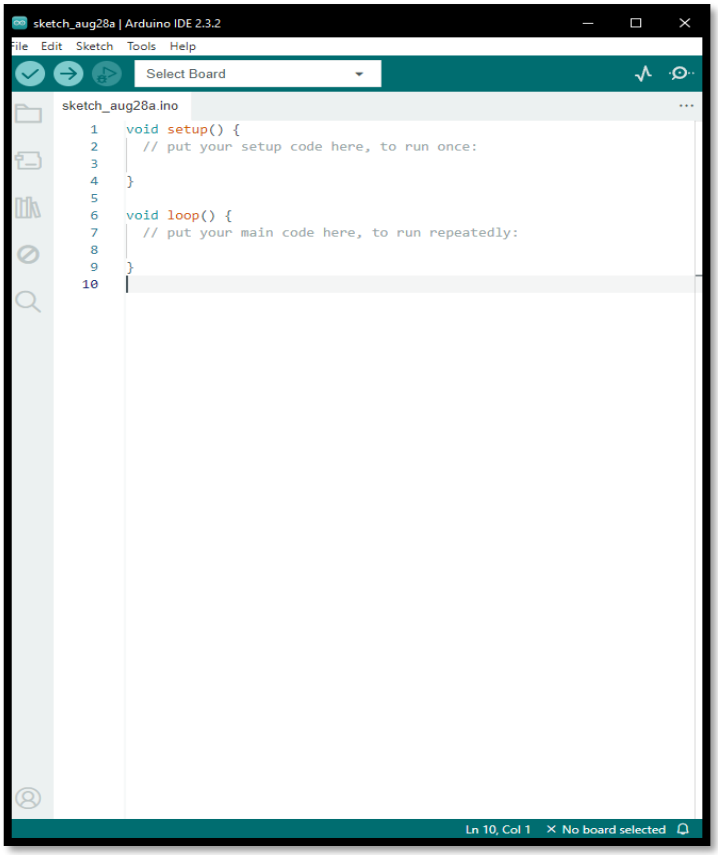
Servomotor	Datos
Voltaje de operación	6 – 7.2 V DC
Consumo de corriente	500 - 900 mA
Ángulo de rotación	0 – 180º
Torque	9.4 kg-cm(4,8V) – 11kg-cm (6V)
Temperatura de trabajo	-30 a + 60 ºC

3.3.1.11. Software de programación

El software (Figura 34) será el utilizado para este proyecto es el de Arduino IDE (Entorno de desarrollo integrado) debido a la versatilidad y los usos que se puede dar al software junto a su hardware. Con este software podremos dar vida al proyecto dando indicaciones al microcontrolador mediante código. El lenguaje que utiliza el software es “C y C++”, debido a que es un software open-source la única limitante es la imaginación.

Figura 34

Interfaz Arduino IDE



Nota: Tomado del Software de Arduino

En esta parte se encuentra el software de programación de Arduino IDE (C y C++) es el inicio de todo el proyecto donde se dará las ordenes al central (Arduino Mega 2560) y a los nodos (ESP32) para el correcto funcionamiento y diseño del código según lo mostrado anteriormente.

3.3.2. Instrumentos

a. Cautín

Usado junto al estaño para la conexión de los nodos, sensores para conectarlo al pcb.

b. Multímetro

Para la verificación de la correcta continuidad del cable entre los sensores, los microcontroladores y poder verificar la correcta alimentación de cada sensor.

c. Pistola de silicona

Usado para la protección de los sensores y los cables, para evitar la entrada de polvo o algún agente corrosivo o dañino que pueda afectar a los microcontroladores.

d. Computador y laptop

Usado para la implementación del software con el hardware del proyecto, para la elaboración del código e implementarlo con el hardware.

e. Taladro portable

Usado para la perforación de las cajas de paso haciendo posible la entrada o salida de los cables de conexión con los sensores con los microcontroladores.

f. Placa PCB

Usado para la inserción del microcontrolador con sus salidas en sus pines para facilitar la conexión de los sensores y actuadores.

3.4. Operalización de las variables:

La tabla 13 muestra la operacionalización de las variables.

Tabla 13

Operalización de variables

Variable	Definición Conceptual	Dimensiones	Indicadores
Independiente Sistema IoT	Es un sistema electrónico que nos permite monitorizar valores como la calidad en un ambiente con un sistema de datos controlado a distancia.	• Red Estrella • Almacenamiento • Visualización • Control	• Procesamiento y trasmisión de la data • Base de Datos • Interfaz Grafica • Automático/Manual
Dependiente Monitoreo de calidad ambiental en invernadero	Se encarga de medir la calidad de un ambiente cerrado	• Monitoreo	• Cantidad de gas CO2 (ppm) • Temperatura • Humedad • Humedad del suelo • pH • Distancia (cm)

3.5. Procesamiento y análisis de datos

3.5.1. Diagramas de flujo de los sensores

En este bloque se verá el proceso que se realizó, para la implementación del código mediante diagramas de flujo (Figura 35, Figura 36, Figura 37, Figura 38, Figura 39) para tener una correcta implementación del código y guía del proyecto.

Figura 35

Diagrama de flujo temperatura

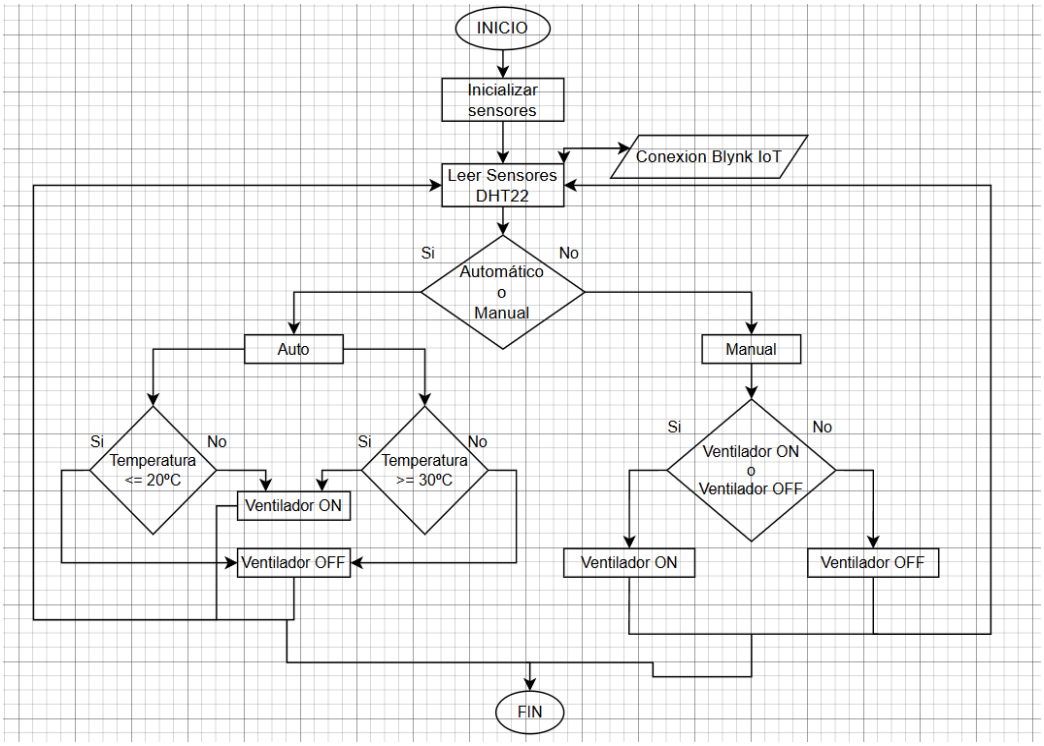


Figura 36

Diagrama de flujo humedad del aire relativa

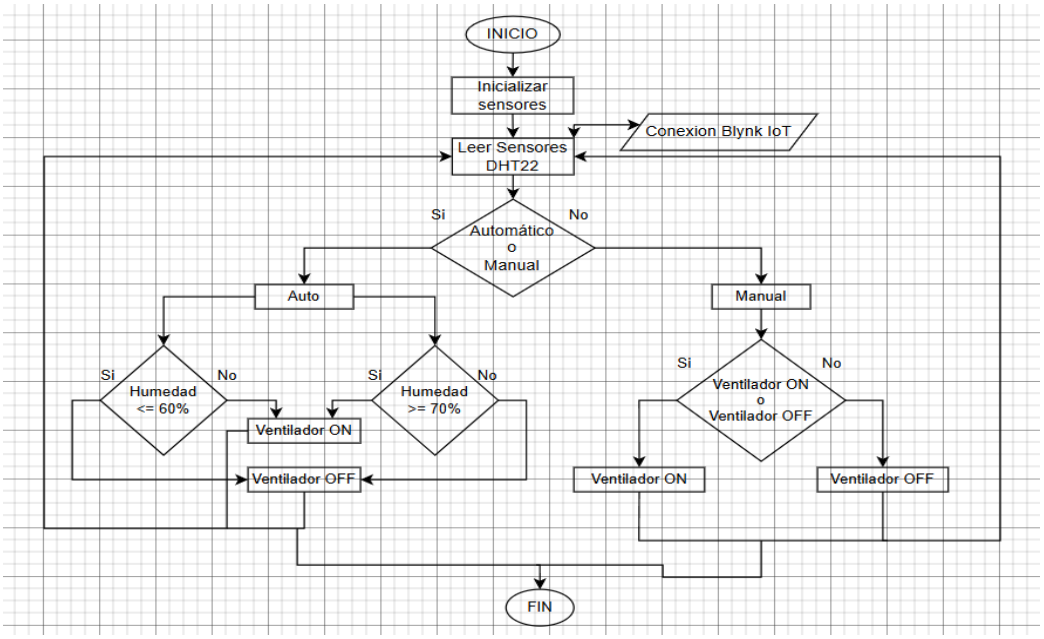


Figura 37

Diagrama de flujo humedad del suelo relativa

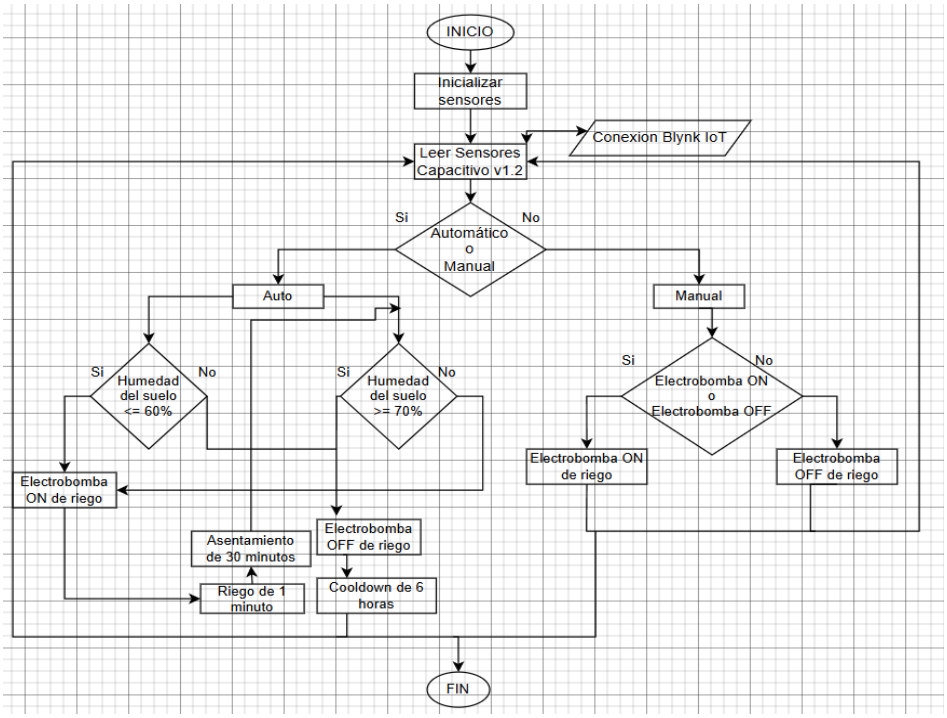


Figura 38

Diagrama de flujo nivel de agua

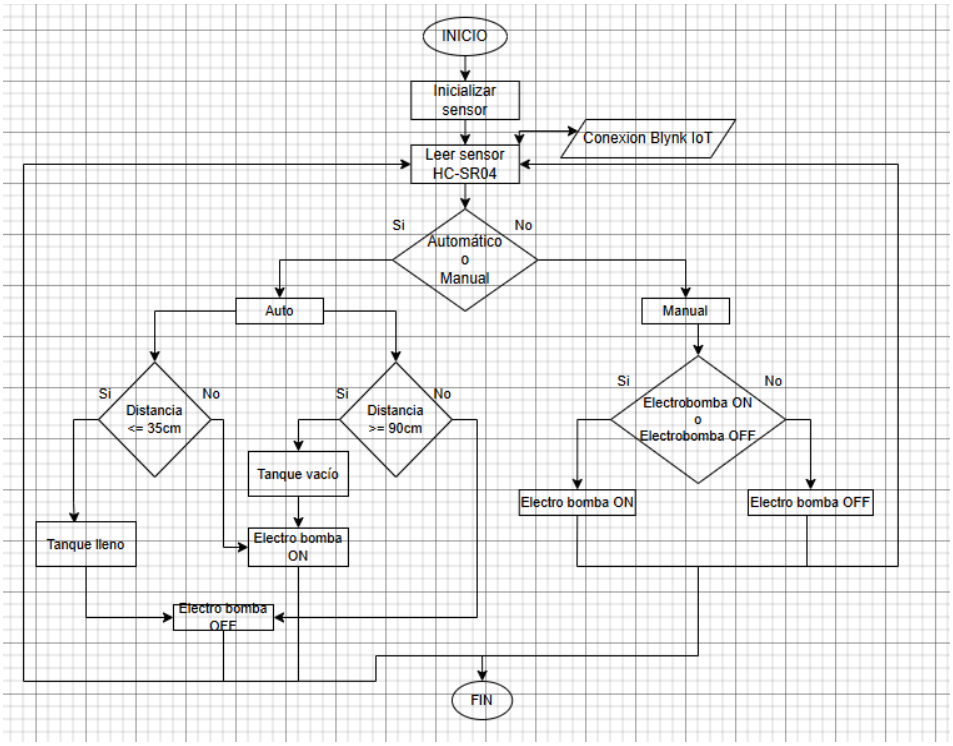


Figura 39

Diagrama de flujo CO2

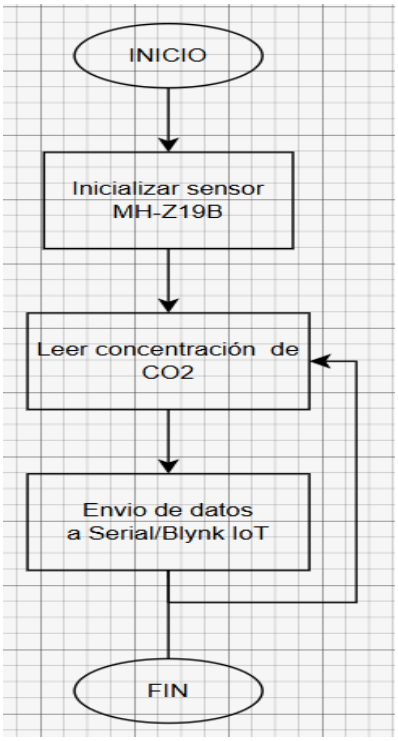
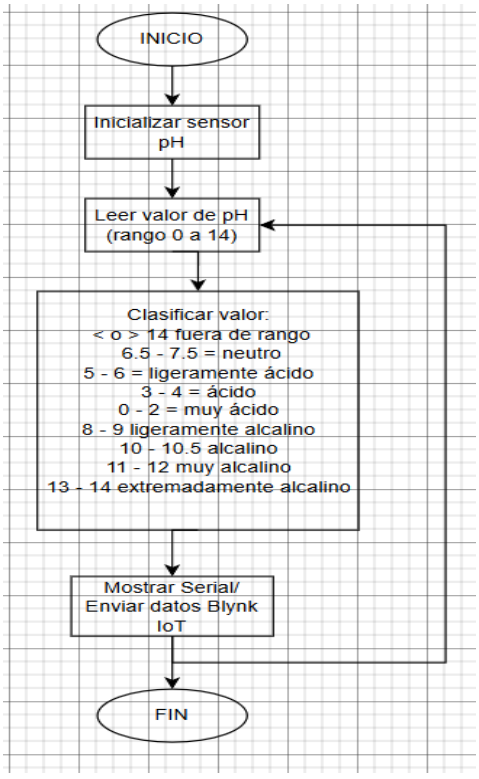


Figura 40

Diagrama de flujo pH



3.5.2. Conexiones de los nodos 01-02-03

En este bloque se empleará las conexiones emuladas (Figura 41, Figura 42, Figura 43) que tendrán los componentes con los microcontroladores, ayudando a facilitar las ubicaciones de pines y correcto funcionamiento del microcontrolador para evitar errores al momento de recibir los datos de temperatura, humedad del aire y humedad del suelo relativas enviadas al nodo central (Arduino Mega 2560).

Figura 41

Conexiones Nodo-01

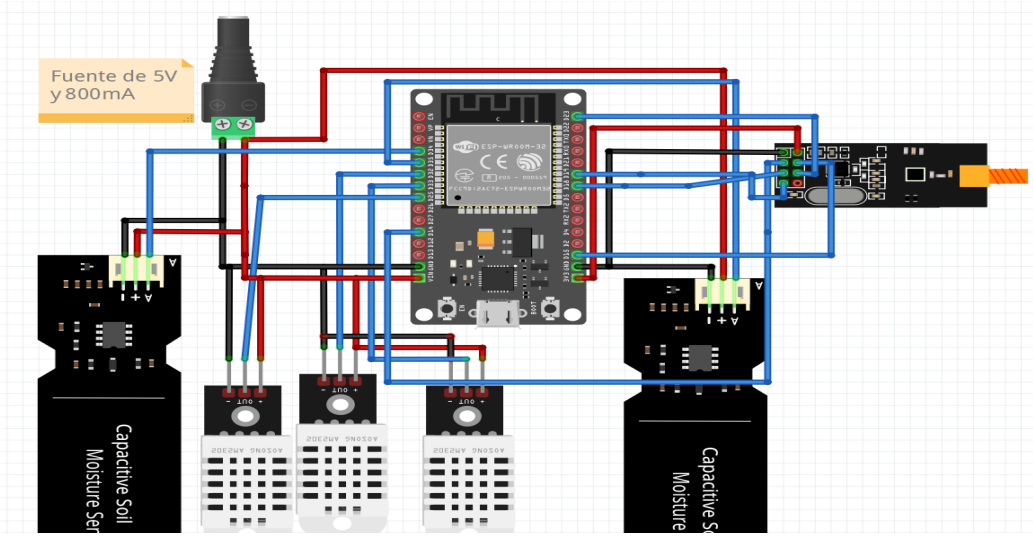


Figura 42

Conexiones Nodo 02

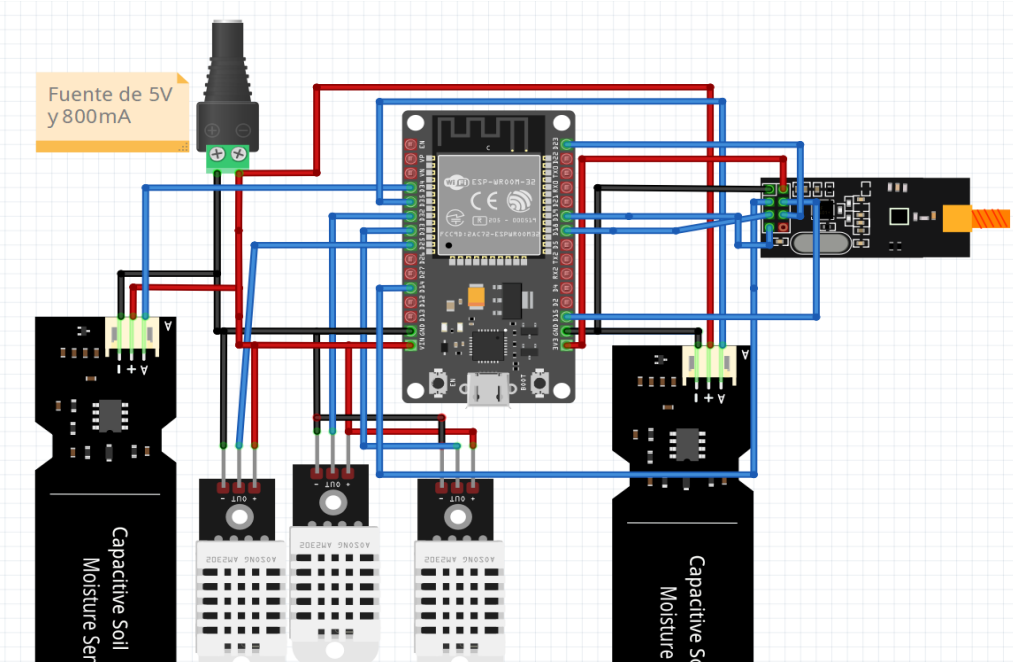
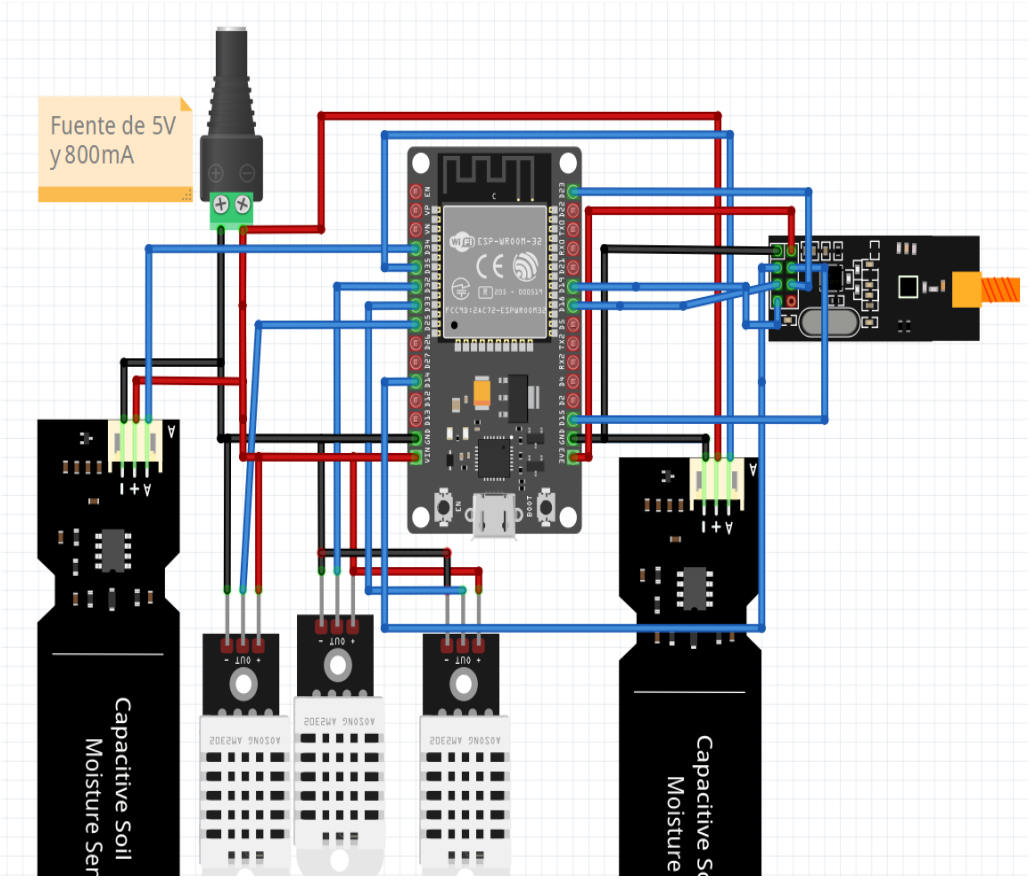


Figura 43

Conexiones Nodo-03



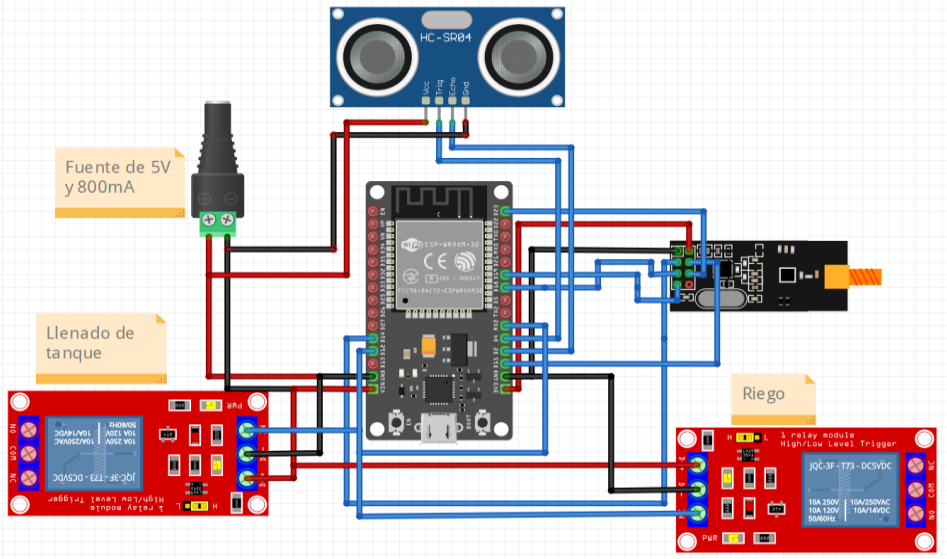
3.5.3. Conexiones del nodo 04

En esta parte se empleará el uso de un sensor ultrasonido como sensor de nivel para medir la distancia de llenado del tanque y ver como esta su estado del tanque lleno o vacío, tambien recibirá comandos del nodo central (Arduino Mega 2560) si necesita riego y activar el modo manual para el llenado y riego, según la preferencia a usar del usuario.

Aquí se podrá aprecia lo implementado en el software en el ESP32 con el sensor ultrasonido junto a los actuadores (reles) para el riego y llenado de tanque (Figura 44).

Figura 44

Conexiones Nodo-04

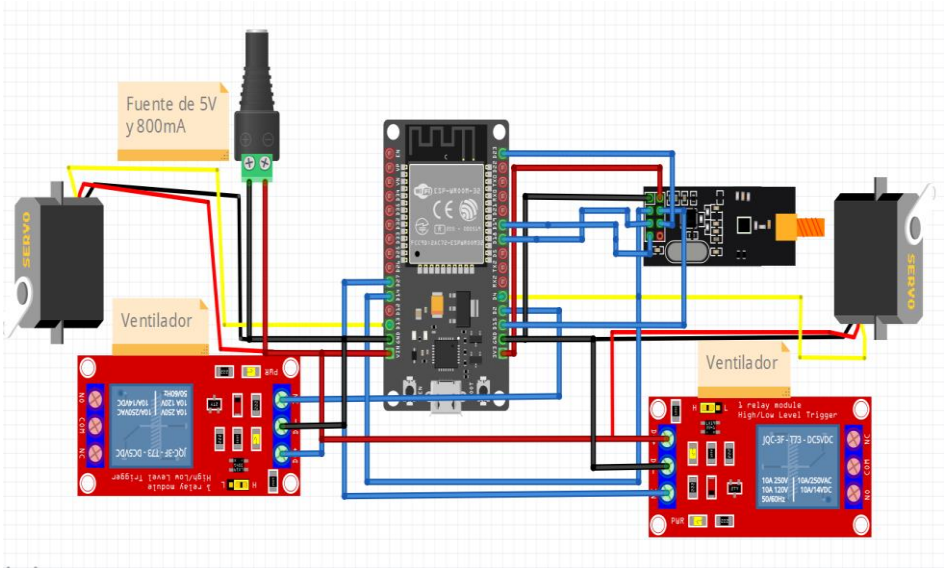


3.5.4. Conexiones del nodo 05

Aquí se empleará el uso del ESP32 con el NRF24L01 para él envió de datos al nodo principal que vendría a ser el Arduino Mega 2560 con el SIM7600SA, en esta parte el ESP32 se encargara de los ventiladores de salida de aire para tener un ambiente interno optimo en el invernadero junto a unos servomotores para sellar las salidas de aire, con la demostración de sus conexiones respectivas (Figura 45).

Figura 45

Conexiones Nodo-05

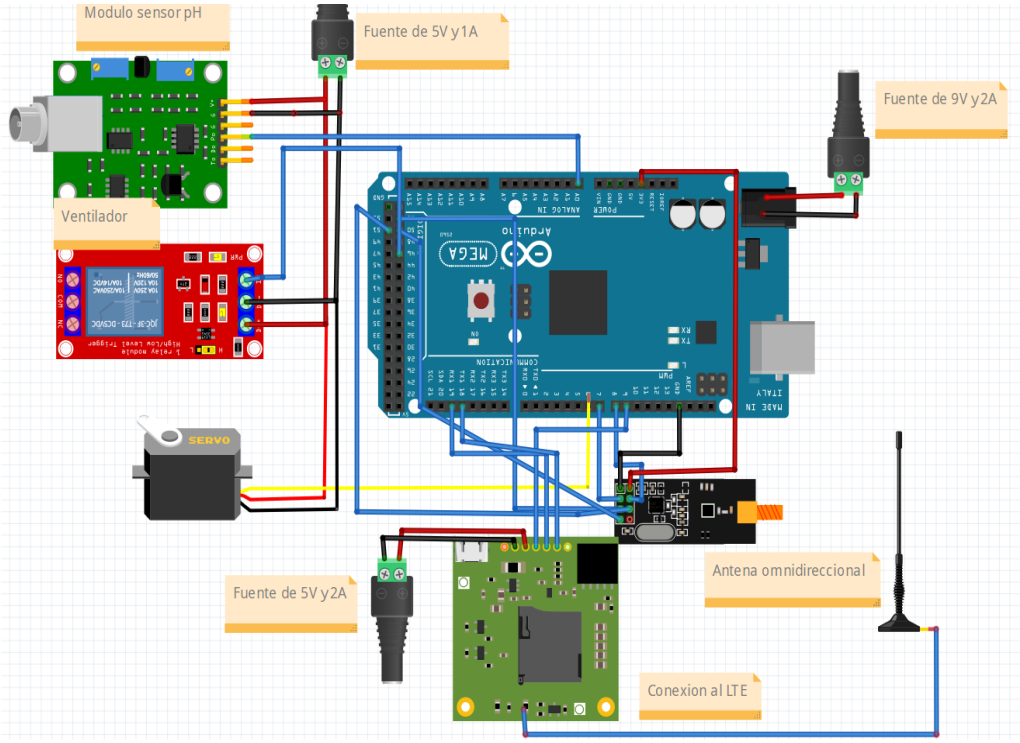


3.5.5. Conexiones del nodo central:

Aquí es donde se encuentra el cerebro de todo el proyecto prototipo de invernadero, donde albergara todos los datos enviados por los nodos, así mismo hace como transmisor y receptor, optando la vía de IoT, será el encargado de enviar a la nube, recibir 5 nodos, junto a ello medir el pH, CO2 y usar un ventilador junto al servo para un ventilador extra de entrada de aire, implementados en el emulador para una correcta guía (Figura 46).

Figura 46

Conexiones nodo central



3.5.6. Implementación de software IoT:

En esta parte de la tesis se explica paso a paso la configuración y la programación del sistema IoT para el invernadero. Para la implementación del software del sistema IoT se usó Arduino IDE y la plataforma Blynk IoT.

El software de Arduino IDE nos facilita elaborar y enviar programas a microcontroladores compatibles con Arduino. El lenguaje por el cual trabaja es C y C++.

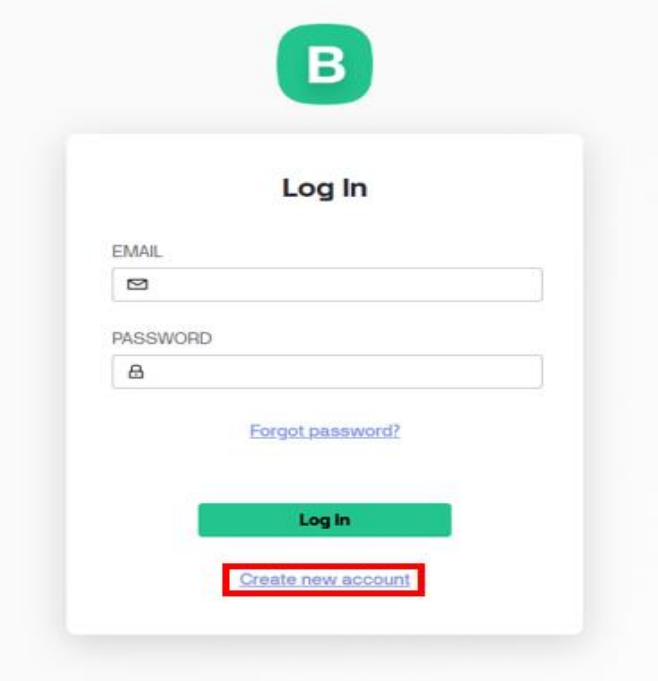
La plataforma de Blynk IoT especializado para IoT con sus servidores en la nube, proporciona datos mediante la nube a cualquier dispositivo que tenga acceso a la KEY que proporciona Blynk de manera privada.

3.5.6.1. Configuración de Blynk IoT:

Para iniciar con la implementación del Blynk se necesitará crear una cuenta (Figura 47) y llenar los datos de correo para la verificación (Figura 48) para poder acceder a sus configuraciones de IoT que se mostrará por gadgets.

Figura 47

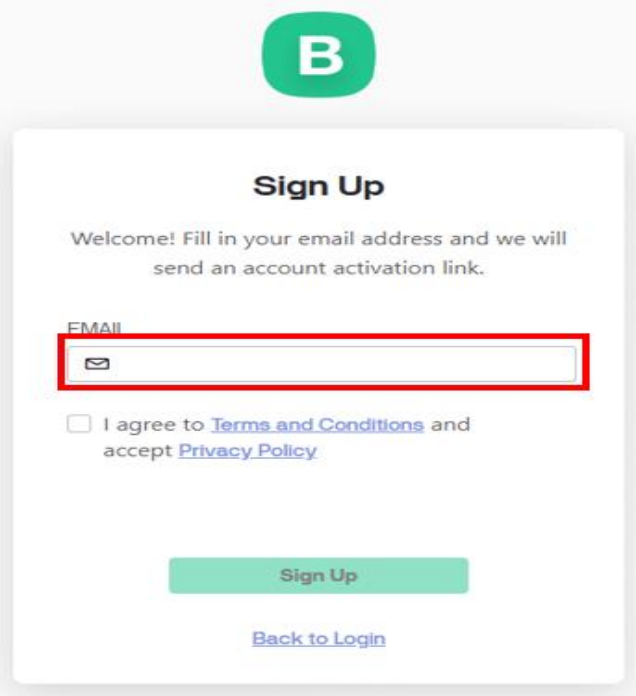
Creaciones de cuenta (Blynk IoT)



Nota: Tomado del software Blynk IoT

Figura 48

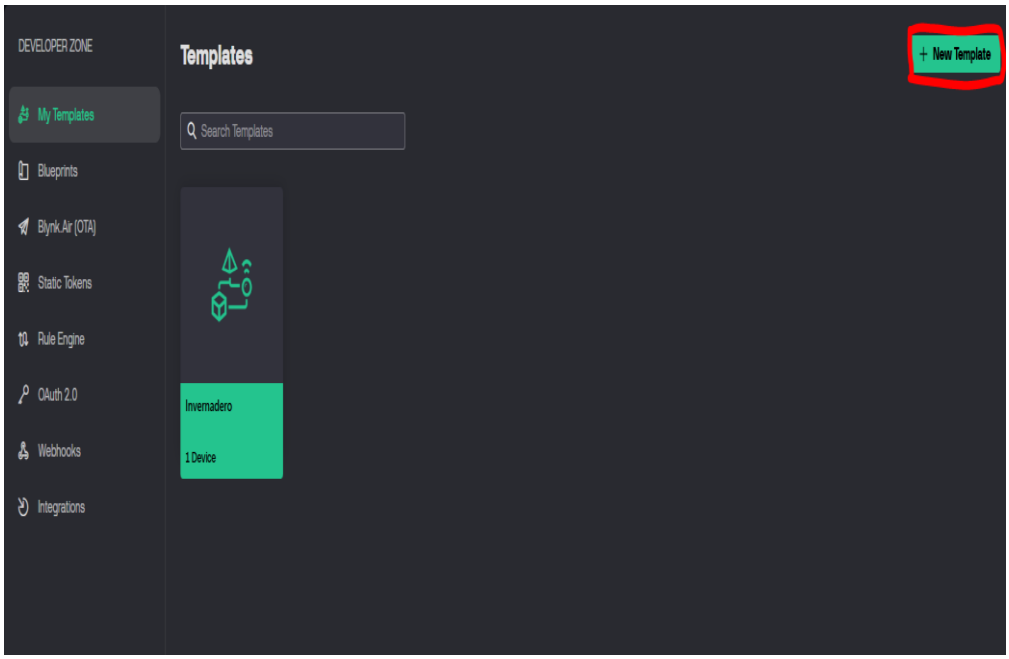
Llenado de datos (Blynk IoT)



Nota: Tomado del software Blynk IoT

Figura 49

Creación de plantilla

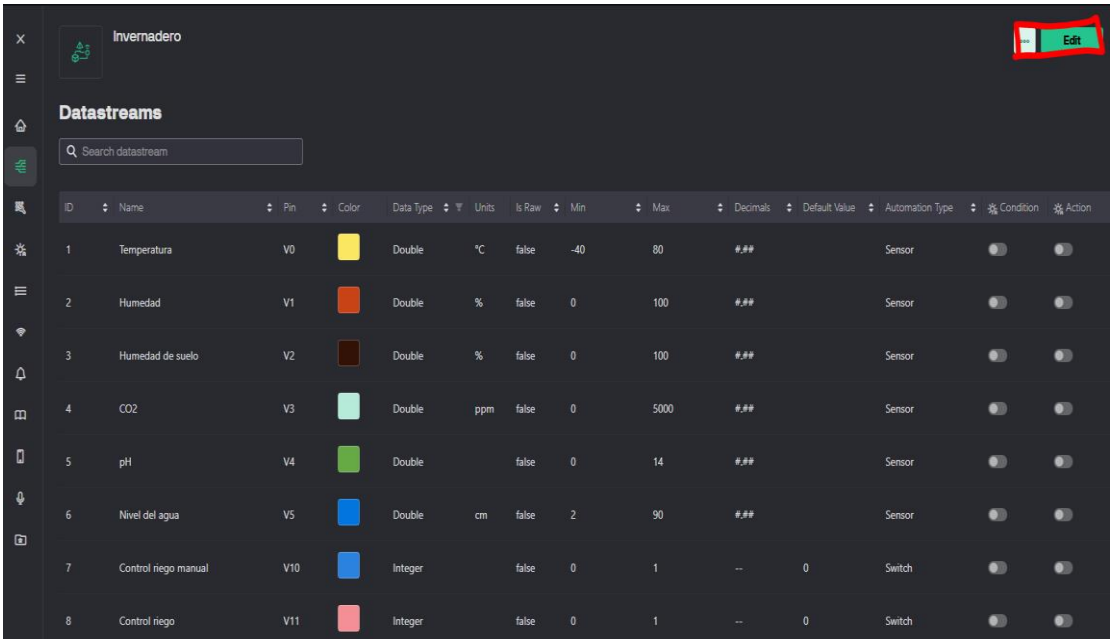


Nota: Tomado del software Blynk IoT

En esta parte se tendrá que crear una plantilla (template) para asi poder asignar cada sensor y control (Figura 49) que tendrá conexión con el Arduino más el SIM7600SA.

Figura 50

Creación de parámetros a medir



ID	Name	Pin	Color	Data Type	Units	Is Raw	Min	Max	Decimals	Default Value	Automation Type	Condition	Action
1	Temperatura	V0		Double	°C	false	-40	80	#.##		Sensor		
2	Humedad	V1		Double	%	false	0	100	#.##		Sensor		
3	Humedad de suelo	V2		Double	%	false	0	100	#.##		Sensor		
4	CO2	V3		Double	ppm	false	0	5000	#.##		Sensor		
5	pH	V4		Double		false	0	14	#.##		Sensor		
6	Nivel del agua	V5		Double	cm	false	2	90	#.##		Sensor		
7	Control riego manual	V10		Integer		false	0	1	--	0	Switch		
8	Control riego	V11		Integer		false	0	1	--	0	Switch		

Nota: Tomado del software Blynk IoT

Aquí se crearán todo lo que se haya puesto en el código como datos a recibir y enviar para cada tipo de sensor (Figura 50) y tambien se creará en el mismo Datastreams la parte de control para cada asignación que se deba realizar.

El software Blynk IoT tiene una limitante de mensajes por mes lo que nos incluye un total de 30 000 mensajes al mes teniendo que hacer cambios en el código para poder hacer valer esos mensajes limites que nos da haciendo un cálculo simple.

i. Envíos en serie de 6 datos por hora

Cada envió ocurre cada 10 minutos y en una hora

$$\frac{60}{10} = 6 \text{ envios por hora} \qquad (2)$$

ii. Cada envío manda 6 datos

$$6\frac{\text{envios}}{\text{hora}} \text{x } 6\frac{\text{datos}}{\text{envios}} = 36\frac{\text{mensajes}}{\text{hora}} \qquad (3)$$

iii. Mensajes por día

$$36\,\frac{\text{mensajes}}{\text{hora}}\,\text{x}\,24\,\frac{\text{horas}}{\text{dia}} = 864\,\frac{\text{mensajes}}{\text{dia}} \qquad (4)$$

iv. Un mes completo

$$864\,\frac{\text{mensajes}}{\text{dia}}\,\text{x}\,30\ \text{dias} = 25\,920\ \text{mensajes mensuales} \qquad (5)$$

3.5.7. Análisis y validación del sistema IoT implementado:

El sistema IoT diseñado fue evaluado bajo condiciones controladas dentro del invernadero prototipo ubicado en el distrito la Yarada los palos.

Se midieron las variables de temperatura, humedad del aire, humedad del suelo, nivel de agua y concentración de CO2, con el fin de determinar la precisión y confiabilidad de la red de sensores, asi como la capacidad de respuesta de los actuadores.

3.5.7.1. Validación de sensores

Ese compara las lecturas obtenidas por los sensores DHT22, MH-Z19B, capacitiva v1,2 y HC-SR04 con valores de referencia obtenidos de instrumentos de medición manual. Se observo una variación máxima de $\pm$2% para la humedad y $\pm$0.5ºC en temperatura, lo cual se considera aceptable para aplicación agrícolas.

Tabla 14

Datos con los sensores

Sensor	Parámetro	Valor medido (promedio)	Valor de referencia	Error %
DHT22	Temperatura	20.5 ºC	20 a 30 ºC	1,1 %
DHT22	Humedad	60 %	62 %	1,5 %
Capacitivo v1,2	Humedad suelo	68 %	70 %	1,6 %
HC-SR04	Nivel agua	35 cm	35 cm	0 %
MH-Z19B	Dióxido de carbono (CO2)	1270	1300	2,5 %
Sensor pH	Alcalinidad - acidez	7,3	7	0,2 %

Nota: Esto demuestra que los sensores ofrecen una precisión suficiente para la toma de decisiones automáticas de riego y ventilación.

3.5.7.2. Estabilidad de comunicación

Durante las pruebas, los nodos basados en ESP32 con módulos NRF24L01 mantuvieron una comunicación estable con el nodo central Arduino Mega 2560 a una distancia de 20 m, sin interferencias ni perdida de paquetes significativas.

El tiempo promedio de transmisión fue de 35 ms por paquete con una tasa de éxito del 95 %, lo que valida la eficiencia del protocolo de comunicación inalámbrica empleada.

3.5.7.3. Evaluación del control automático

El sistema respondió correctamente a los cambios ambientales:

- Cuando la humedad del suelo fue menor al 60 %, se activó automáticamente el relé de riego.
- Cuando la temperatura supero los 30 ºC, los ventiladores se encendieron, manteniendo el rango entre 21 ºC y 29 ºC.

3.5.7.4. Comparación con el sistema tradicional

Para validar la mejora propuesta, se comparó el consumo de agua entre el riego tradicional y el sistema IoT automatizado.

En una simulación de 5 días de riego, el sistema IoT redujo el consumo total en aproximadamente 25 % de un tanque de 600 L con llenado de 500 L resguardando 125 L para un uso propio, debido al encendido automático únicamente cuando haya pasado el tiempo dicho por el código (6 horas de espera) comprobando siempre el rango de humedad.

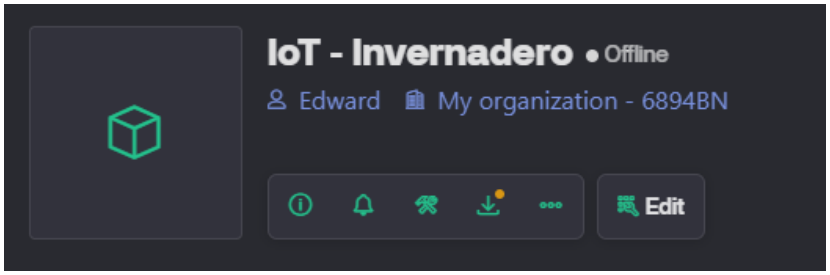
3.5.8. Base de datos:

Para la base de datos se tuvo que almacenar con un software Blynk IoT que esté conectado a la red (Figura 51) para tener registro de los sensores dicho registro será mediante la visualización grafica de cada sensor (Figura 52). Otra particularidad importante es que cada grafica se le debió configurar el tipo de dato y si recibirá de manera “decimal” o “entera” los datos enviados por el SIM7600SA que tambien podrá

recibirlo el teléfono celular (Figura 53) gracias al uso del software de Blynk IoT haciendo aún más sencillo el manejo del sistema. Por otro lado, además, de importante son las señalas se recibirá en SIM7600SA junto al microcontrolador para la parte manual y automática que tendrá este proyecto (Figura 54).

Figura 51

Conexión Proyecto y software



Nota: Tomado del software Blynk IoT

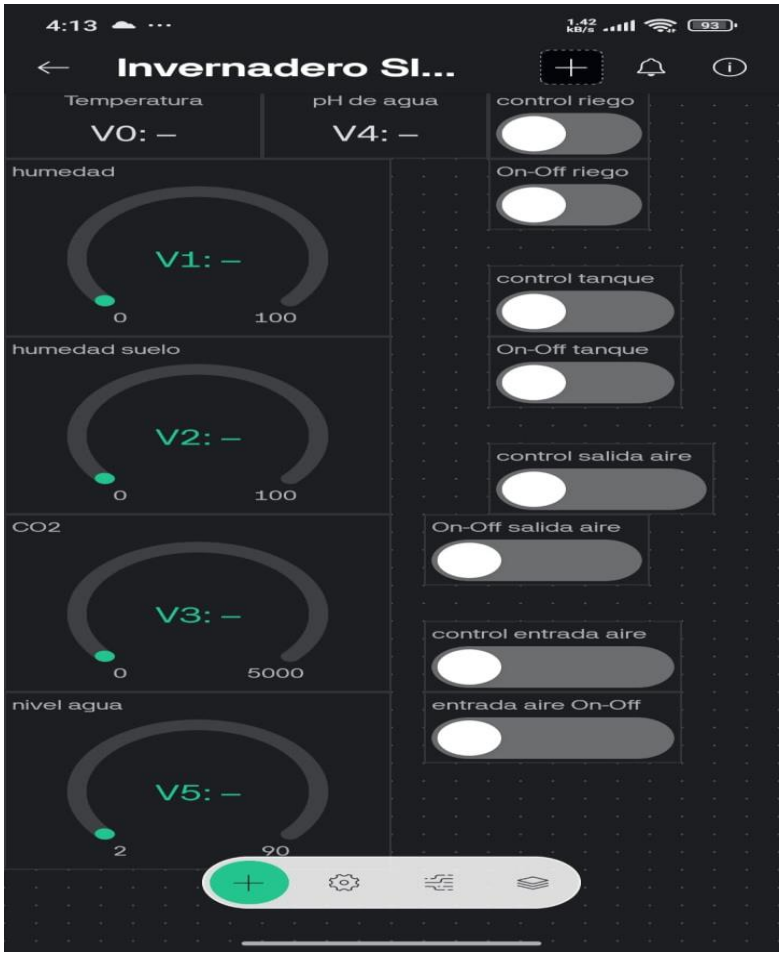
Figura 52

Graficas insetadas en el software

Nota: Tomado del software Blynk IoT

Figura 53

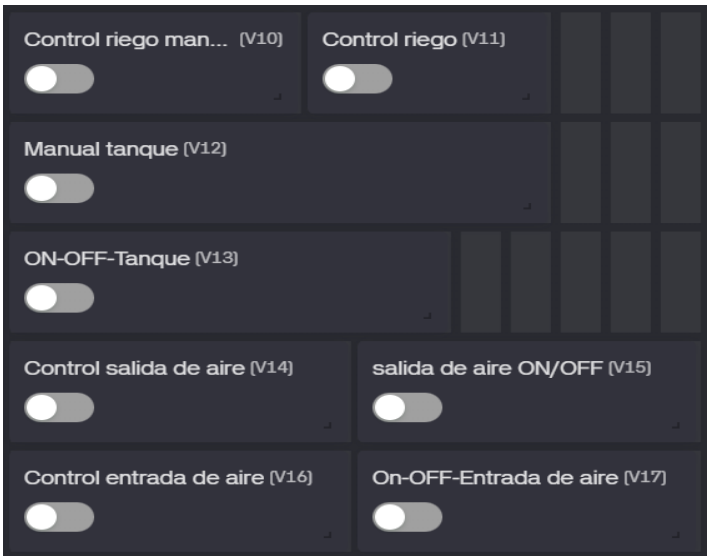
Graficas insertadas en aplicativo móvil



Nota: Tomado del software Blynk IoT

Figura 54

Switches insertados para control

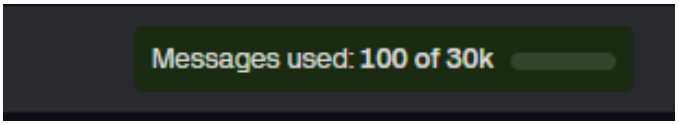


Nota: Tomado del software Blynk IoT

Tambien el almacenamiento de datos enviados al servidor IoT nos da un límite de almacenamiento de 30 000 mensajes (Figura 55) lo que hace suficiente para el funcionamiento de este proyecto.

Figura 55

Cantidad de mensajes



Nota: Tomado del software Blynk IoT, tener precaución con los mensajes enviados.

3.5.9. Revisión general

Los resultados obtenidos confirman que el sistema IoT propuesto cumple los objetivos específicos de la investigación, al proporcionar un control automatizado eficiente, de bajo consumo y alta confiabilidad. El sistema demostró capacidad de operación continua, estabilidad en la red de nodos con el nodo central y correcta comunicación con la nube mediante el módulo SIM7600SA.

CAPITULO IV: RESULTADOS

4.1. Conexión optima bueno recibimiento

La calidad de la conexión se comprobó por la buena recepción del Arduino Mega 2560 con NRF24L01 una velocidad de 250kbps de los nodos logrando recibir y mandar señales según dictaminado en el código elaborado en el software IDE.

Figura 56

Datos recibidos del nodo 1 en PC

```
====== Ciclo completado ======

 Datos recibidos del Nodo 1:
Temperatura: 36.33 °C
Humedad: 43.87 %
Humedad del suelo (promedio): 50.50 %
Estado de los sensores DHT:
  DHT1: OK
  DHT2: OK
  DHT3: OK
```

Figura 57

Datos recibidos del nodo 2 en PC

```
 Datos recibidos del Nodo 2:
Temperatura: 34.83 °C
Humedad: 43.27 %
Humedad del suelo (promedio): 56.50 %
Estado de los sensores DHT:
  DHT1: OK
  DHT2: OK
  DHT3: OK
```

Figura 58

Datos recibidos del nodo 3 en PC

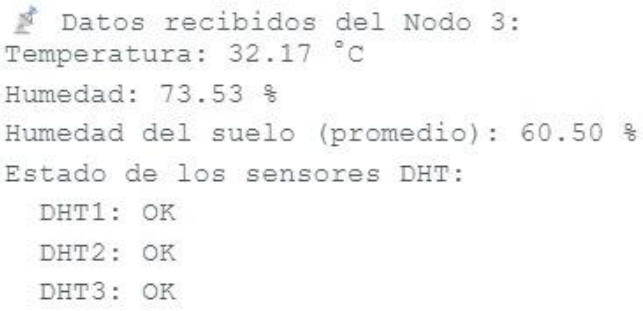


Figura 59

Datos recibidos del nodo 4 en PC

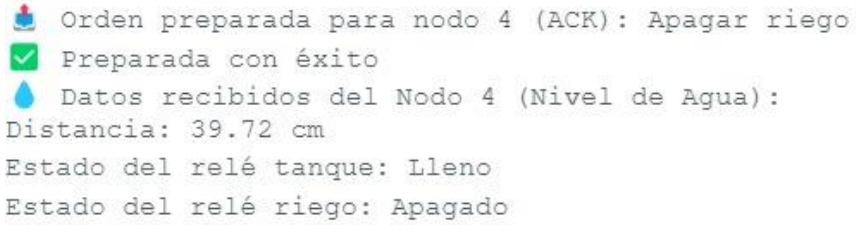


Figura 60

Datos recibidos del nodo 5 en PC

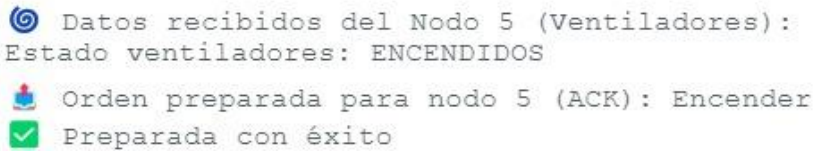


Figura 61

Datos calculados por el Arduino Mega (promedio) en PC

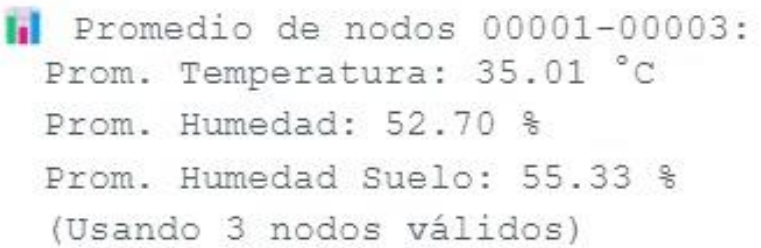


Figura 62

Ordenes de riego, ventiladores y pH en PC

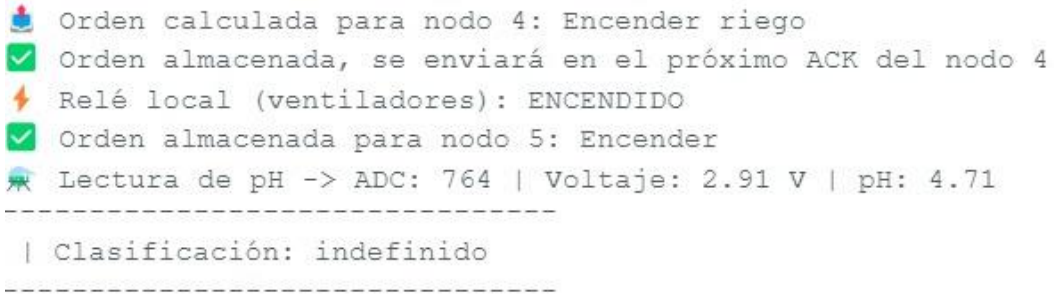


Figura 63

Datos de ppm en el Arduino Mega 2560 en PC

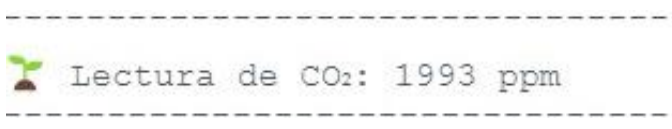


Tabla 15

Nodos operativos

N.º de nodo	Nodo ID	Recepción
1	00001	Correcta (-94 dBm)
2	00002	Correcta (-94 dBm)
3	00003	Correcta (-94 dBm)
4	00004	Correcta (-94 dBm)
5	00005	Correcta (-94 dBm)

Nota: Se pudo comprobar la recepción de datos con éxito en las partes de prueba antes de ser llevado a campo.

4.2. Implementación física e incorporación en invernadero:

Aquí se verá todo lo relacionado con los nodos físicos junto con la instalación en el interior del invernadero.

4.2.1. NODO-01

El nodo 01, se observa en las figuras 64 al 66.

Figura 64

Parte física Nodo-01

Figura 65

Parte interna Nodo-01

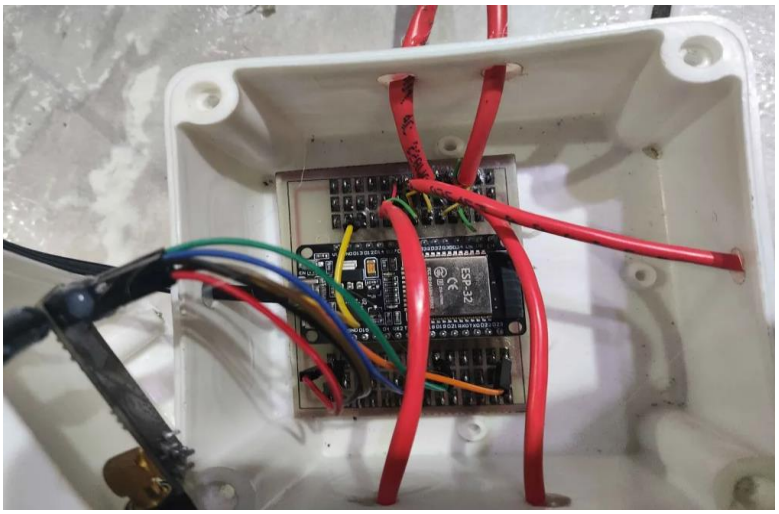


Figura 66

Instalacion en invernadero Nodo-01

4.2.2. NODO-02

El nodo 02, se observa en las figuras 67 al 69.

Figura 67

Parte física Nodo-02

Figura 68

Parte interna Nodo-02

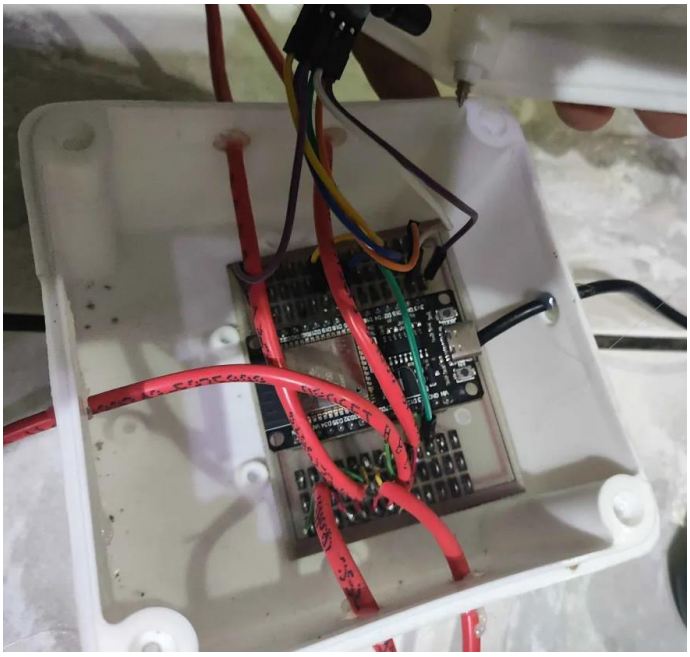


Figura 69

Instalacion en invernadero Nodo-02

4.2.3. NODO-03:

El nodo 03, se observa en las figuras 70 al 72.

Figura 70

Parte física Nodo-03

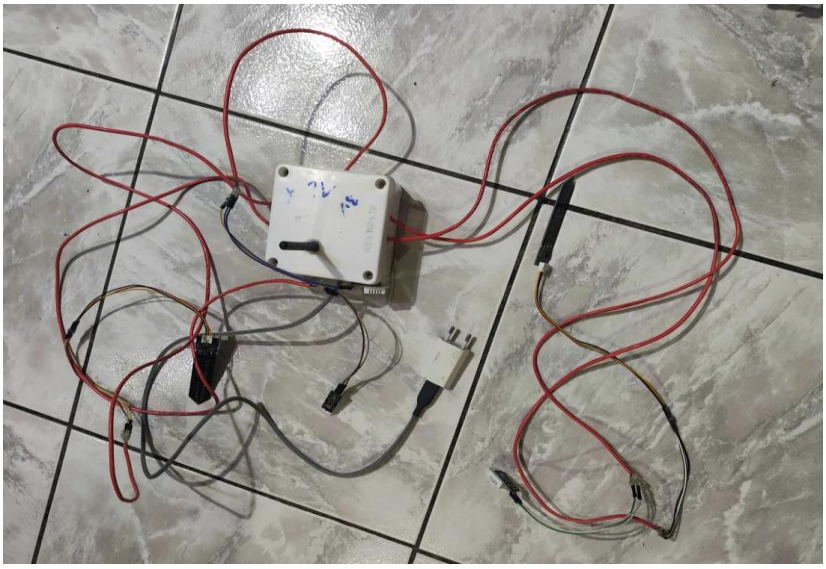


Figura 71

Parte Interna Nodo-03

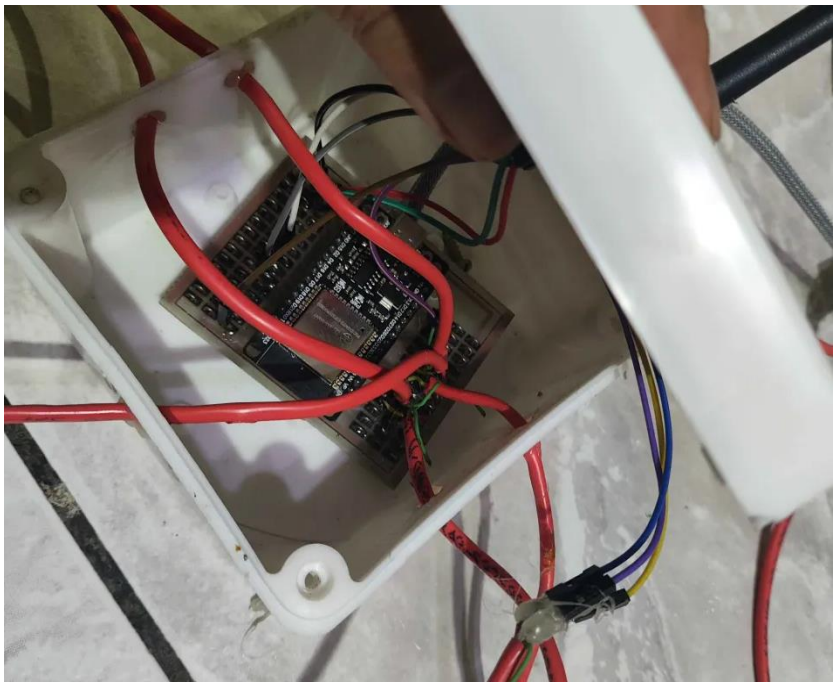


Figura 72

Instalacion en invernadero Nodo-03

4.2.4. NODO-04:

El nodo 04, se observa en las figuras 73 al 75.

Figura 73

Parte física Nodo-04

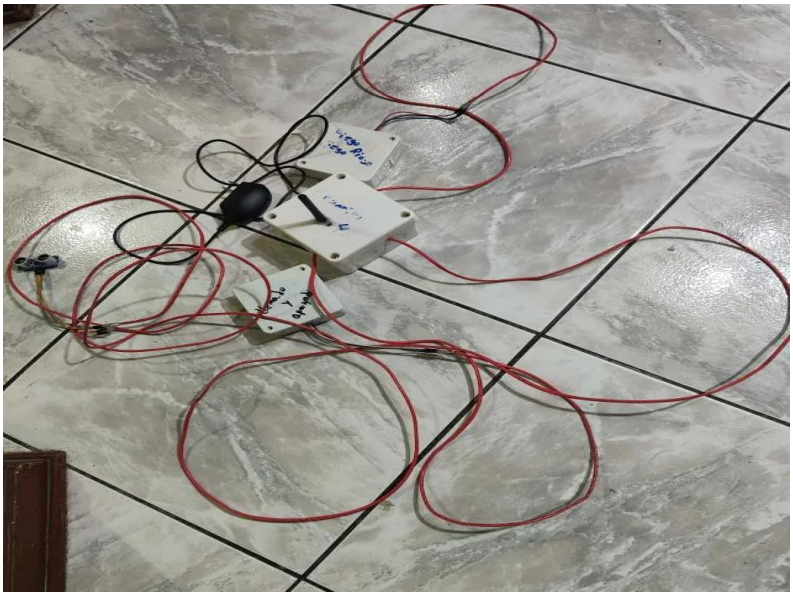


Figura 74

Parte interna Nodo-04

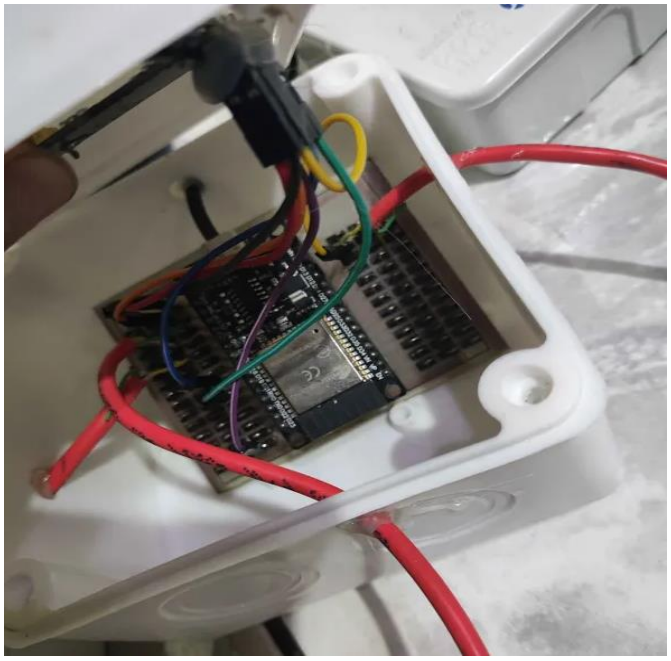


Figura 75

Instalacion en invernadero Nodo-04

4.2.5. NODO-05:

El nodo 05, se observa en las figuras 76 al 78.

Figura 76

Parte física del Nodo-05

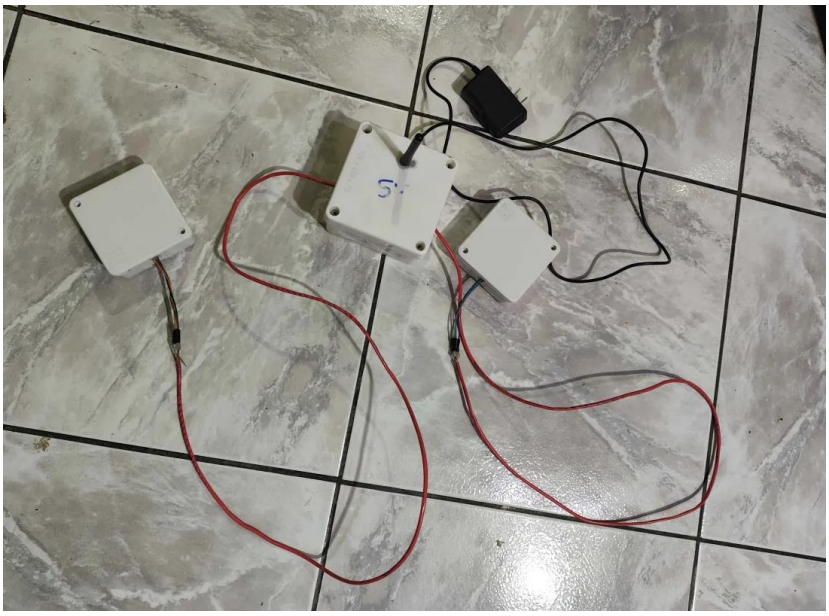


Figura 77

Parte interna Nodo-05

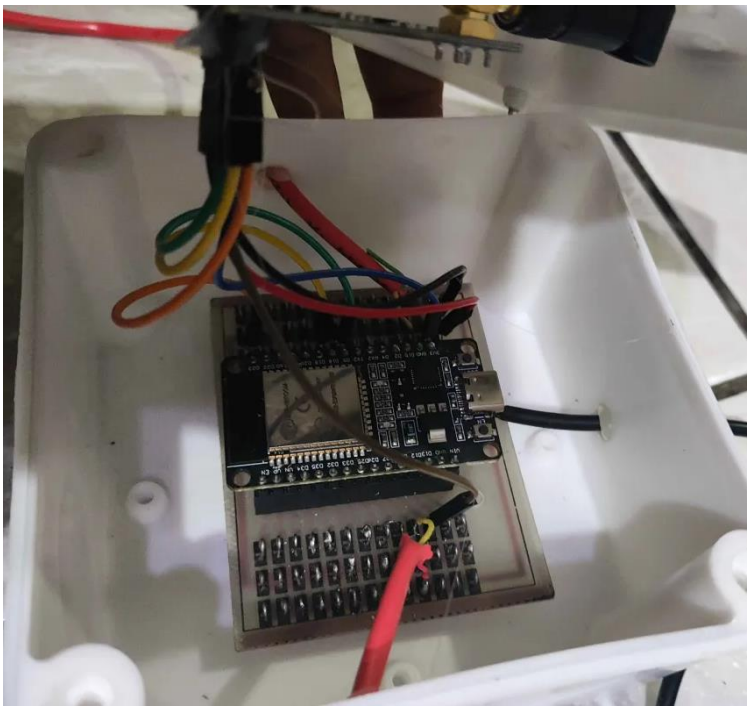
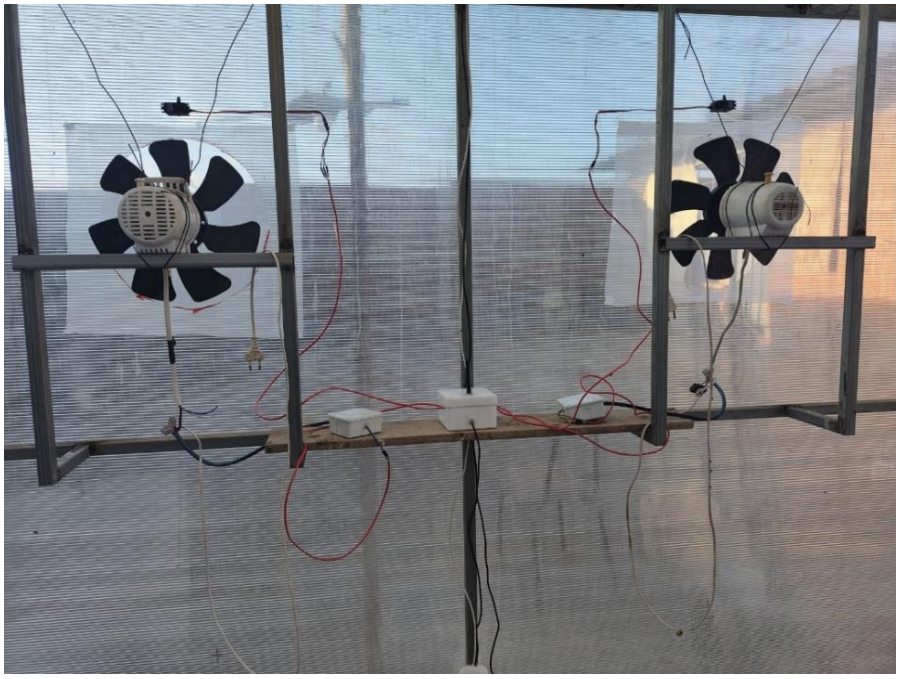


Figura 78

Instalacion en invernadero Nodo-05



4.2.6. NODO-CENTRAL:

El nodo central, se observa en las figuras (Figura 79; Figura 80; Figura 81; Figura 82)

Figura 79

Parte física del Nodo Central

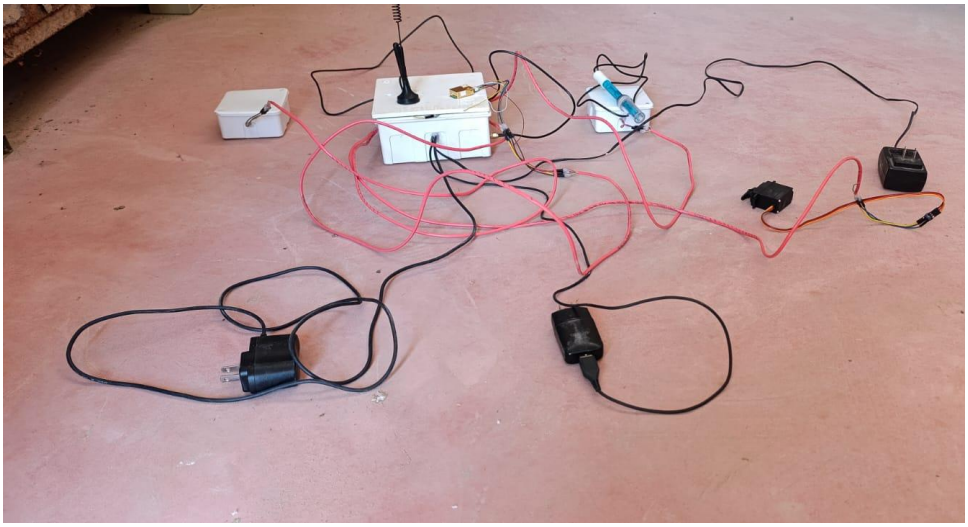


Figura 80

Parte interna del Nodo Central

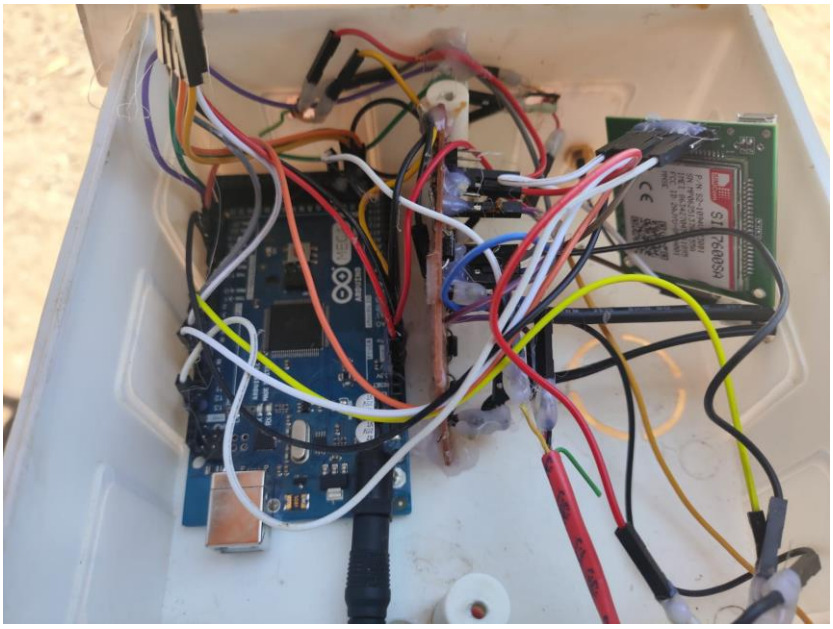


Figura 81

Instalacion en invernadero Nodo Central

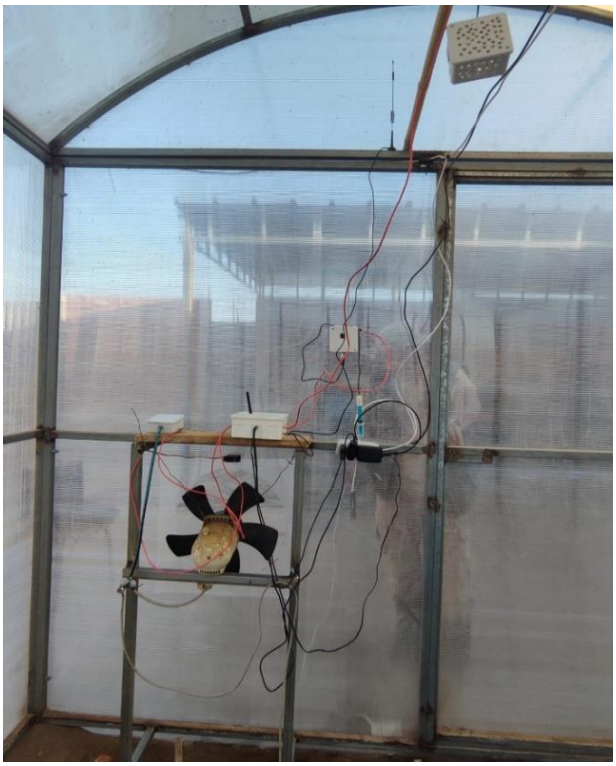
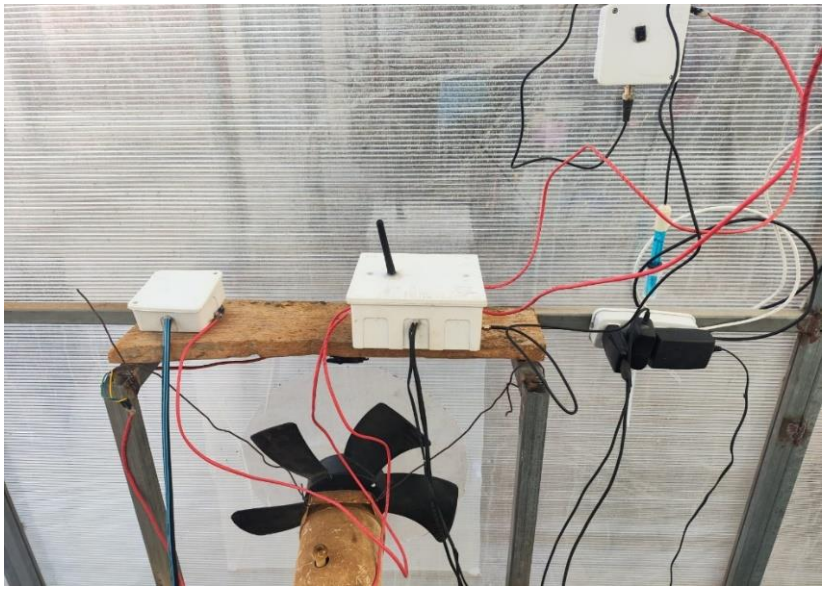


Figura 82

Instalacion de Nodo Central de cerca



4.3. Parámetros de medidos para un ambiente en prototipo invernadero

Para facilitar la visualización de los datos de una mejor manera gráfica y factible, dando a que cualquier usuario pueda comprenderlo.

4.3.1. Temperatura

La temperatura medida fue en grados Celsius (ºC) el promedio de los tres primeros nodos que fueron recepsionados por el nodo central (Arduino Mega 2560). A continuación, se muestra un gráfico indicador (Figura 83).

Figura 83

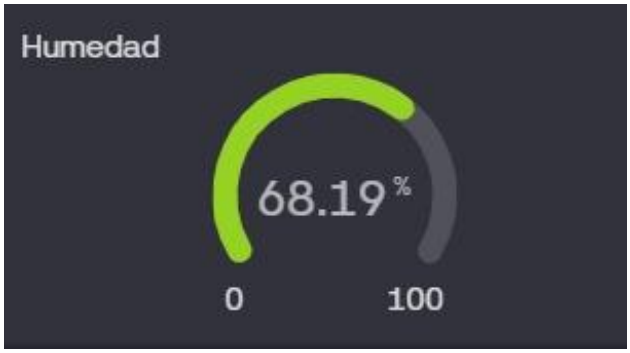
Demostración de temperatura promedio

4.3.2. Humedad del aire

La humedad medida fue en porcentaje (%) promediada al igual que la temperatura de los tres primeros nodos que recepciono el nodo central en comparación con la temperatura al estar seco el aire, la humedad bajara al disminuir el aire se vuelve húmedo. Por consiguiente, se ve una figura (Figura 84).

Figura 84

Demostración de Humedad promedio



4.3.3. CO2

El dióxido de carbono (CO2) fue medido por partes por millón (ppm). El comportamiento de este sensor variara con el pasar del día teniendo un poco de variación con la temperatura y la humedad ambiente relativa, si se mantiene una temperatura de 20-25ºC y una humedad de 60%-70%, tendrá una desviación leve menor a 3% considerado factible para aplicación agrícola. A continuación, una imagen (Figura 85).

Figura 85

Demostración de CO2 (ppm)

4.3.4. Humedad del suelo

La humedad del suelo fue medida en porcentajes (%) al igual que el aire, esta variable tendrá una constancia según lo requiera el cultivo, manteniendo con intervalos de tiempo para evitar encharcamientos (Figura 86).

Figura 86

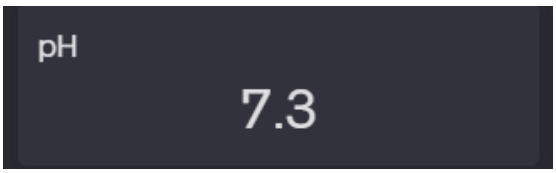
Demostración de humedad del suelo promedio

4.3.5. Calidad de líquido para riego

El nivel óptimo con el que se puedo medir el pH antes de las pruebas fue encontrar la calibración correcta y comprobarla con soluciones exactas de 4pH y 7pH. Mostrada en la siguiente imagen (Figura 87).

Figura 87

Demostración de pH



4.3.6. Nivel de agua

En esta variable importante se mide en distancia (cm) por el sensor, se usó como nivelador de agua según la distancia a la que se encuentre con el sensor, esta variable determinara el uso adecuado del agua ahorrando innecesariamente el regado continuo, balanceando los tiempos de riego, con intervalos de tiempo. A continuación, se vera la distancia en la que se encuentra el regado de tanque (Figura 88).

Figura 88

Demostracion de nivel de agua (cm)

CAPITULO V:DISCUSION

Según los resultados hallados, aceptamos el objetivo general a lo que se apuntaba este proyecto el cual era planteado a: “Diseñar e implementar un prototipo por IoT que tenga un sistema de control que mejor el cultivo de orégano en un invernadero mediante la monitorización clave como la humedad, temperatura, entre otros, en la Yarada Los Palos, en la región Tacna”.

Por otro lado, se logró el correcto funcionamiento del software Blynk IoT, logrando abarcar el 86.4% del total gratuito que nos brinda el software, teniendo el funcionamiento las 24 horas. Comprobando que es factible para un usuario el fácil uso del software, sin tener que recurrir a complejos gráficos, difíciles de entender para un usuario, de acuerdo a Pazmiño (2022) con su software de datos utilizado, “Esta plataforma permite tener una gestión de todo el conjunto de datos obtenidos al igual que un registro histórico” (p. 52.), validando el uso de softwares amigables para usuarios que no tengan tanto conocimiento en lo tecnológico.

En el transcurso del proyecto se comprobó la fiabilidad del prototipo de invernadero con IoT, obteniendo las variables ambientales optimas usando esta tecnología según Satuquinga (2023) según dice “permite tecnificar el proceso de cultivar el tomate de riñon y mejorar las condiciones que perjudican el crecimiento y desarrollo del fruto para evitar pérdidas en el cultivo” (p. 19.).

CONCLUSIONES

Se concluyo el diseño e implementación del prototipo invernadero con IoT, con un sistema de control que mejora el cultivo de orégano mediante la correcta monitorización de la humedad, temperatura, humedad del suelo, CO2, en la Yarada Los palos, Tacna. Además, que se puede medir la calidad de agua con la que se va a regar el cultivo mediante pH si es alcalino o ácido, de cierto modo la distancia para un correcto uso del agua.

Se comprobó los parámetros correctos y climáticos en el invernadero para la producción de orégano logrando estabilidad para el cultivo de 15-30 ºC de temperatura, 40-70 % Humedad del aire, 60-70 % Humedad del suelo, dándonos valores favorables para su cultivo.

Se logro utilizar la tecnología de microcontroladores (ESP32, Arduino Mega 2560) para el funcionamiento del prototipo junto a su software de programación (Arduino IDE en C++) y los sensores usados que controlan el prototipo (DHT22, Capacitivo 2,0, HC – SR04) se encargan de controlar el ambiente y el uso de agua, los sensores de datos (Sensor pH, Sensor MH-Z19B), que nos envían datos para corroborar con el agua que se irriga y la calidad de aire (CO2).

Se concluyó que es factible el uso de RF (radio frecuencia), a 250kbps mediante los módulos NRF24L01 puede ser óptimo para IoT sin problemas de conectividad gracias al amplio espacio.

RECOMENDACIONES

Para futuras investigaciones se recomiendo en el uso del NRF24L01 (PA+LNA) poner en los pines de energía un condensador de 47uF o uno de 100uF para mejorar la estabilidad eléctrica, evitando caídas de voltaje y favoreciendo la señal cuadrada enviada por el RF

Una recomendación crucial al momento de la instalación, tener en cuenta el suministro de energía con el que será alimentado el prototipo, y pueda ser funcional sin problemas, evitando algún cortocircuito.

Al momento de utilizar el software de Blynk IoT es recomendable utilizar envíos intermites, debido a que el software tiene envio por mensajes limitados, pasado el límite, ya no se podrá recibir mensajes del prototipo.

Se recomienda tener la orientación de Norte a Sur, para tener una buena distribución de la radiación más uniforme dentro del invernadero, además de evitar generar zonas con sombra, como ocurre en el sentido de Este a Oeste.

REFERENCIAS BIBLIOGRÁFICAS

Satuquinga (2023) Sistema de control y monitoreo de riego, purificación de aire y fumigación para la plantación de tomate de riñón en invernaderos mediante dispositivos iot en la agropecuaria san miguel de salcedo. Obtenido de t2210ec.pdf (uta.edu.ec)

Pazmiño (2022) Repotenciación de un Sistema Hidropónico Convencional Para el Cultivo de Lechugas Lactuca Sativa y Batavia Boinda Di Parigi, Mediante Automatización y Plataformas IoT. Obtenido de Pazmiño Jarrin José Francisco.pdf (uisek.edu.ec)

Merino (2022) Prototipo iot para un invernadero acuapónico doméstico en áreas urbanas. Obtenido de t1974ec.pdf (uta.edu.ec)

Mendoza y Cordero (2022) Diseño de un sistema IoT para el monitoreo remoto de la calidad del aire en ambientes interiores de la facultad de ingeniería de la Universidad Privada de Tacna. Obtenido de Mendoza-Merma-Cordero Ochoa.pdf

Echevarría (2023) Implementación de un prototipo de sensado para el proceso de administración de basura utilizando LORAWAN en el contexto de ciudades inteligentes. Obtenido de Tesis_Maestria_Katherine Echeverria_v1 (epn.edu.ec)

Molina (2024) Diseño e implementación de una red de área amplia de baja potencia (LPWAN), para el monitoreo y optimización del tiempo de registro de consumo, corte y reconexión del suministro eléctrico monofásico residencial. Obtenido de Molina M.,Alberto D. (2024) Diseño e implementación de una red de área amplia de baja potencia (LPWAN), para el monitoreo y optimización del tiempo de registro de consumo, corte.pdf (unach.edu.ec)

Rojas y Chate (2022) Generación de gemelo digital de un aula inteligente para el monitoreo de CO2, temperatura y radiación UV orientado a la prevención del covid-19 y otras patologías asociadas al medio ambiente empleando tres nodos basados en la tecnología NB-IOT. Obtenido de content (udistrital.edu.co)

Garzón y Reyes (2023) Estudio de mecanismos de ciberseguridad para asegurar las comunicaciones en internet de las cosas industrial IoT. Obtenido de Estudio de mecanismos de ciberseguridad para asegurar las comunicaciones en internet de las cosas industrial IoT. (ups.edu.ec)

Ramirez (2024) Diseño de un sistema de monitoreo IoT para mejorar la productividad del agricultor en el Poblado de Cuchuchin-Sayan, 2023. Obtenido de Tesis Aldair Ramirez.pdf

Falcón y Cuya (2022) Sitema web basado en IoT para mejorar el rendimiento de aves finas de un criadero inteligente en la empresa LM Business Falcón Magallan, Gian Franco y Cuya Delesma, Naim Victor.pdf

López y Toscano (2023) Implementación de sistemas de iluminación, medición de temperatura y automatización del sistema de ventilación para un invernadero ubicado en la comunidad de Mollepamba. Obtenido de Manual de Estilo

Chasi (2023) Sistema de riego para invernaderos con interfaz web utilizando un Arduino Yum. Obtenido de CD 13827.pdf (epn.edu.ec)

BCRP (2024) Síntesis de actividad Económica abril 2024. Obtenido de INTRODUCCION

Suarez y Velarde (2023) Implementación de un sistema inteligente para invernadero de cultivos de hortalizas utilizando tecnología IoT y energía solar. Obtenido de content

Mendoza y Ochoa (2022) Diseño de un sistema IoT para el monitorea remoto de la calidad del aire en ambientes interiores de la facultad de ingeniería de la Universidad Privada de Tacna. Obtenido de Mendoza-Merma-Cordero Ochoa.pdf

Cardenas (2024) Diseño de red de datos mediante la metodología top down en el instituto nacional de estadictica e informática sede Junin 2024. Obtenido de content

Fierro (2023) Diseño de una red inalámbrica de banda ancha para proveer internet a la escuela Padre Celestial del distrito de Ahuac, probicnia de Chupaca -2023. Obtenido de IV_FIN_113_TE_Fierro_Tito_2024.pdf

[Productos electrónicos y ventas]. (s.f.). Naylamp Mechatronics - Perú

[Componentes electrónica y robotica]. (s.f.). SAI SAC – MECATRONICA – Somos importadores de impresoras 3D y sistemas CNC

Aspectos da considerar en la orientación de los invernaderos. (s.f.). Intragri. Aspectos a Considerar en la Orientación de los Invernaderos | Intagri S.C.

Goteros regulables para riego por goteo y sacabocados. (s.f.). Jaen Clima Tipos de Goteros regulables para riego por goteo y sacabocados - Jaén Clima

[ArduinoDocs información completa del microcontrolador]. (s.f.) Mega 2560 Rev3 | Arduino Documentation

ANEXOS

ANEXO 1. Código del nodo Nº01

```
#include <DHT.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

// Pines y configuración para los sensores DHT22
#define DHTPIN1 32
#define DHTPIN2 33
#define DHTPIN3 25
#define DHTTYPE DHT22

DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
DHT dht3(DHTPIN3, DHTTYPE);

// Pines para sensores de humedad del suelo
#define SOIL_SENSOR1 34
#define SOIL_SENSOR2 35
#define SOIL_SENSOR3 36 // Sensor fantasma, no usar

// Pines para NRF24L01
#define CE_PIN 14
#define CSN_PIN 15

RF24 radio(CE_PIN, CSN_PIN);
const byte address[6] = "00001";  // Address implementado

struct __attribute__((packed)) DataPacket {
  float temperature;
  float humidity;
  float soilMoistureAverage;
  bool dht1_status;
  bool dht2_status;
  bool dht3_status;
  uint8_t node_id;
};

DataPacket data;

unsigned long lastSend = 0;
const unsigned long interval = 4000; // Tiempo entre envíos en ms

void setup() {
```

```
  Serial.begin(9600);

  dht1.begin();
  dht2.begin();
  dht3.begin();

  pinMode(SOIL_SENSOR1, INPUT);
  pinMode(SOIL_SENSOR2, INPUT);
  pinMode(SOIL_SENSOR3, INPUT);  // Sin usar

  if (!radio.begin()) {
    Serial.println(" Error al inicializar el NRF24L01.");
    while (1);
  }

  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MAX);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(100);
  radio.setRetries(15, 15);
  radio.stopListening();
  radio.enableDynamicPayloads();

  Serial.println(" Nodo listo y transmitiendo...");
}

void loop() {
  unsigned long now = millis();
  if (now - lastSend >= interval) {
    lastSend = now;

    float tempSum = 0, humSum = 0;
    int validDHTs = 0;

    // Lectura DHT1
    float t1 = dht1.readTemperature();
    float h1 = dht1.readHumidity();
    if (!isnan(t1) && !isnan(h1)) {
      tempSum += t1;
      humSum += h1;
      validDHTs++;
      data.dht1_status = true;
    } else {
      data.dht1_status = false;
    }

    // Lectura DHT2
    float t2 = dht2.readTemperature();
```

```
float h2 = dht2.readHumidity();
if (!isnan(t2) && !isnan(h2)) {
  tempSum += t2;
  humSum += h2;
  validDHTs++;
  data.dht2_status = true;
} else {
  data.dht2_status = false;
}

// Lectura DHT3
float t3 = dht3.readTemperature();
float h3 = dht3.readHumidity();
if (!isnan(t3) && !isnan(h3)) {
  tempSum += t3;
  humSum += h3;
  validDHTs++;
  data.dht3_status = true;
} else {
  data.dht3_status = false;
}

if (validDHTs > 0) {
  data.temperature = tempSum / validDHTs;
  data.humidity = humSum / validDHTs;
} else {
  data.temperature = -99.0;
  data.humidity = -99.0;
}

// Leer humedad del suelo
int soil1 = analogRead(SOIL_SENSOR1);
int soil2 = analogRead(SOIL_SENSOR2);
int soil3 = analogRead(SOIL_SENSOR3);  // No usa

// Mapeo de 0-4095 a 100-0 (humedad %)
soil1 = map(soil1, 0, 4095, 100, 0);
soil2 = map(soil2, 0, 4095, 100, 0);
soil3 = map(soil3, 0, 4095, 100, 0);  // No se usa

// Calcular el promedio de los sensores solo dos
data.soilMoistureAverage = (soil1 + soil2) / 2.0;

data.node_id = 1;  // El id de cada nodo, cambiar

// Mostrar por Serial
Serial.print("Temperatura: ");
Serial.print(data.temperature);
```

```
    Serial.print(" °C | Humedad: ");
    Serial.print(data.humidity);
    Serial.print(" % | Hum. Suelo: ");
    Serial.print(data.soilMoistureAverage);
    Serial.println(" %");

    // Enviar datos
    bool success = radio.write(&data, sizeof(data));
    Serial.println(success ? " Datos enviados correctamente." : " Error al
enviar los datos.");
  }
}
```

ANEXO 2. Código del nodo Nº02

```
#include <DHT.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

// Pines y configuración para los sensores DHT22
#define DHTPIN1 32
#define DHTPIN2 33
#define DHTPIN3 25
#define DHTTYPE DHT22

DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
DHT dht3(DHTPIN3, DHTTYPE);

// Pines para sensores de humedad del suelo
#define SOIL_SENSOR1 34
#define SOIL_SENSOR2 35
#define SOIL_SENSOR3 36 // Sensor fantasma, no usar

// Pines para NRF24L01
#define CE_PIN 14
#define CSN_PIN 15

RF24 radio(CE_PIN, CSN_PIN);
const byte address[6] = "00002";   // Address implementado para el envió entre NRF24L01

struct __attribute__((packed)) DataPacket {
  float temperature;
  float humidity;
  float soilMoistureAverage;
  bool dht1_status;
  bool dht2_status;
  bool dht3_status;
  uint8_t node_id;
};

DataPacket data;

unsigned long lastSend = 0;
const unsigned long interval = 2700; // Tiempo entre envíos en ms

void setup() {
  Serial.begin(9600);

  dht1.begin();
```

```
  dht2.begin();
  dht3.begin();

  pinMode(SOIL_SENSOR1, INPUT);
  pinMode(SOIL_SENSOR2, INPUT);
  pinMode(SOIL_SENSOR3, INPUT);  // Sin usar

  if (!radio.begin()) {
    Serial.println(" Error al inicializar el NRF24L01.");
    while (1);
  }

  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MAX);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(100);
  radio.setRetries(15, 15);
  radio.stopListening();
  radio.enableDynamicPayloads();

  Serial.println(" Nodo listo y transmitiendo...");
}

void loop() {
  unsigned long now = millis();
  if (now - lastSend >= interval) {
    lastSend = now;

    float tempSum = 0, humSum = 0;
    int validDHTs = 0;

    // Lectura DHT1
    float t1 = dht1.readTemperature();
    float h1 = dht1.readHumidity();
    if (!isnan(t1) && !isnan(h1)) {
      tempSum += t1;
      humSum += h1;
      validDHTs++;
      data.dht1_status = true;
    } else {
      data.dht1_status = false;
    }

    // Lectura DHT2
    float t2 = dht2.readTemperature();
    float h2 = dht2.readHumidity();
    if (!isnan(t2) && !isnan(h2)) {
      tempSum += t2;
```

```
  humSum += h2;
  validDHTs++;
  data.dht2_status = true;
} else {
  data.dht2_status = false;
}

// Lectura DHT3
float t3 = dht3.readTemperature();
float h3 = dht3.readHumidity();
if (!isnan(t3) && !isnan(h3)) {
  tempSum += t3;
  humSum += h3;
  validDHTs++;
  data.dht3_status = true;
} else {
  data.dht3_status = false;
}

if (validDHTs > 0) {
  data.temperature = tempSum / validDHTs;
  data.humidity = humSum / validDHTs;
} else {
  data.temperature = -99.0;
  data.humidity = -99.0;
}

// Leer humedad del suelo
int soil1 = analogRead(SOIL_SENSOR1);
int soil2 = analogRead(SOIL_SENSOR2);
int soil3 = analogRead(SOIL_SENSOR3);  // No usa

// Mapeo de 0–4095 a 100–0 (humedad %)
soil1 = map(soil1, 0, 4095, 100, 0);
soil2 = map(soil2, 0, 4095, 100, 0);
soil3 = map(soil3, 0, 4095, 100, 0);  // No se usa

// Calcular el promedio de los sensores solo dos
data.soilMoistureAverage = (soil1 + soil2) / 2.0;

data.node_id = 2;  // El id de cada nodo, cambiar

// Mostrar por Serial
Serial.print("Temperatura: ");
Serial.print(data.temperature);
Serial.print(" °C | Humedad: ");
Serial.print(data.humidity);
Serial.print(" % | Hum. Suelo: ");
```

```
    Serial.print(data.soilMoistureAverage);
    Serial.println(" %");

    // Enviar datos
    bool success = radio.write(&data, sizeof(data));
    Serial.println(success ? " Datos enviados correctamente." : " Error al
enviar los datos.");
  }
      }
```

ANEXO 3. Código del nodo Nº04

```
#include <DHT.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

// Pines y configuración para los sensores DHT22
#define DHTPIN1 32
#define DHTPIN2 33
#define DHTPIN3 25
#define DHTTYPE DHT22

DHT dht1(DHTPIN1, DHTTYPE);
DHT dht2(DHTPIN2, DHTTYPE);
DHT dht3(DHTPIN3, DHTTYPE);

// Pines para sensores de humedad del suelo
#define SOIL_SENSOR1 34
#define SOIL_SENSOR2 35
#define SOIL_SENSOR3 36 // Sensor fantasma, no usar

// Pines para NRF24L01
#define CE_PIN 14
#define CSN_PIN 15

RF24 radio(CE_PIN, CSN_PIN);
const byte address[6] = "00003";   // Address implementado para el envió entre NRF24L01

struct __attribute__((packed)) DataPacket {
  float temperature;
  float humidity;
  float soilMoistureAverage;
  bool dht1_status;
  bool dht2_status;
  bool dht3_status;
  uint8_t node_id;
};

DataPacket data;

unsigned long lastSend = 0;
const unsigned long interval = 4000; // Tiempo entre envíos en ms

void setup() {
  Serial.begin(9600);

  dht1.begin();
```

```
  dht2.begin();
  dht3.begin();

  pinMode(SOIL_SENSOR1, INPUT);
  pinMode(SOIL_SENSOR2, INPUT);
  pinMode(SOIL_SENSOR3, INPUT);  // Sin usar

  if (!radio.begin()) {
    Serial.println(" Error al inicializar el NRF24L01.");
    while (1);
  }

  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MAX);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(100);
  radio.setRetries(15, 15);
  radio.stopListening();
  radio.enableDynamicPayloads();

  Serial.println(" Nodo listo y transmitiendo...");
}

void loop() {
  unsigned long now = millis();
  if (now - lastSend >= interval) {
    lastSend = now;

    float tempSum = 0, humSum = 0;
    int validDHTs = 0;

    // Lectura DHT1
    float t1 = dht1.readTemperature();
    float h1 = dht1.readHumidity();
    if (!isnan(t1) && !isnan(h1)) {
      tempSum += t1;
      humSum += h1;
      validDHTs++;
      data.dht1_status = true;
    } else {
      data.dht1_status = false;
    }

    // Lectura DHT2
    float t2 = dht2.readTemperature();
    float h2 = dht2.readHumidity();
    if (!isnan(t2) && !isnan(h2)) {
      tempSum += t2;
```

```
  humSum += h2;
  validDHTs++;
  data.dht2_status = true;
} else {
  data.dht2_status = false;
}

// Lectura DHT3
float t3 = dht3.readTemperature();
float h3 = dht3.readHumidity();
if (!isnan(t3) && !isnan(h3)) {
  tempSum += t3;
  humSum += h3;
  validDHTs++;
  data.dht3_status = true;
} else {
  data.dht3_status = false;
}

if (validDHTs > 0) {
  data.temperature = tempSum / validDHTs;
  data.humidity = humSum / validDHTs;
} else {
  data.temperature = -99.0;
  data.humidity = -99.0;
}

// Leer humedad del suelo
int soil1 = analogRead(SOIL_SENSOR1);
int soil2 = analogRead(SOIL_SENSOR2);
int soil3 = analogRead(SOIL_SENSOR3);  // No usa

// Mapeo de 0-4095 a 100-0 (humedad %)
soil1 = map(soil1, 0, 4095, 100, 0);
soil2 = map(soil2, 0, 4095, 100, 0);
soil3 = map(soil3, 0, 4095, 100, 0);  // No se usa

// Calcular el promedio de los sensores solo dos
data.soilMoistureAverage = (soil1 + soil2) / 2.0;

data.node_id = 3;  // El id de cada nodo, cambiar

// Mostrar por Serial
Serial.print("Temperatura: ");
Serial.print(data.temperature);
Serial.print(" °C | Humedad: ");
Serial.print(data.humidity);
Serial.print(" % | Hum. Suelo: ");
```

```
    Serial.print(data.soilMoistureAverage);
    Serial.println(" %");

    // Enviar datos
    bool success = radio.write(&data, sizeof(data));
    Serial.println(success ? " Datos enviados correctamente." : " Error al
enviar los datos.");
  }
      }
```

ANEXO 4. Código del nodo Nº04

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#define echoPin 2
#define trigPin 4
#define relayPin 16   // Rele de llenado del tanque
#define relayPin2 12  // Rele de riego

#define CE_PIN 14
#define CSN_PIN 15

RF24 radio(CE_PIN, CSN_PIN);

// Dirección para este nodo
const byte address[6] = "00004";

//Para enviar datos
struct __attribute__((packed)) WaterLevelData {
  float distance;
  bool relayState;   // true = llenando, false = lleno
  bool relay2State;  // estado del rele de riego
};

// Para recibir órdenes
struct __attribute__((packed)) RelayControl {
  bool manualModeRelay1;  // true = manual, false = automático
  bool relay1Command;     // ON/OFF en manual
  bool manualModeRelay2;  // true = manual, false = automático para riego
  bool relay2Command;     // ON/OFF en manual riego
};

WaterLevelData waterData;
RelayControl controlData = {false, false, false, false}; // En automático

// Variables para control
unsigned long previousMillis = 0;
const unsigned long interval = 1500; // 1.5 segundos

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(relayPin, OUTPUT);
  pinMode(relayPin2, OUTPUT);

  digitalWrite(relayPin, LOW);
```

```
  digitalWrite(relayPin2, LOW);

  if (!radio.begin()) {
    Serial.println(" Error al inicializar el NRF24L01.");
    while (1);
  }

  if (!radio.isChipConnected()) {
    Serial.println(" NRF24L01 no está conectado correctamente");
  } else {
    Serial.println(" NRF24L01 conectado correctamente");
  }

  radio.enableDynamicPayloads();
  radio.enableAckPayload();
  radio.setPALevel(RF24_PA_MAX);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(100);
  radio.setRetries(15, 20);
  radio.openWritingPipe(address);
  radio.openReadingPipe(1, address); // ACK payload
  radio.stopListening();

  Serial.println(" Nodo 4 listo para enviar y recibir datos.");
}

void loop() {
  unsigned long currentMillis = millis();

  // ENVÍO DE DATOS
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Medición ultrasónico
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    float duracion = pulseIn(echoPin, HIGH, 25000);

    if (duracion > 0) {
      waterData.distance = (duracion * 0.0343) / 2.0;

      // Control del rele 1 (llenado)
      if (!controlData.manualModeRelay1) {
        if (waterData.distance >= 90.0) {
```

```
          digitalWrite(relayPin, HIGH);
          waterData.relayState = true;
        } else if (waterData.distance <= 35.0) {
          digitalWrite(relayPin, LOW);
          waterData.relayState = false;
        }
      } else {
       // Modo manual, obedece comando, pero con límites para una mejor seguridad
      if (controlData.relay1Command && waterData.distance >= 35.0) {
        digitalWrite(relayPin, HIGH);   // encender solo si no está lleno
        waterData.relayState = true;
       } else {
         digitalWrite(relayPin, LOW);    // forzar apagado si lleno o si orden es OFF
        waterData.relayState = false;
  }
}

      // Control del rele 2 (riego)
      if (controlData.manualModeRelay2) {
      // Modo manual: obedece el comando
      digitalWrite(relayPin2, controlData.relay2Command ? HIGH : LOW);
      waterData.relay2State = digitalRead(relayPin2);

      } else {
      //  Modo automático: obedece la orden calculada en el receptor
      digitalWrite(relayPin2, controlData.relay2Command ? HIGH : LOW);
      waterData.relay2State = digitalRead(relayPin2);
      }


      // Mostrar en monitor serie
      Serial.print("Distancia: ");
      Serial.print(waterData.distance, 2);
      Serial.print(" cm | Estado: ");
        Serial.print(waterData.relayState ? "llenando tanque" : "Tanque lleno");
     Serial.print(controlData.manualModeRelay1 ? " (MANUAL)" : " (AUTO)");
      Serial.print(" | Riego: ");
      Serial.print(waterData.relay2State ? "Encendido" : "Apagado");
        Serial.println(controlData.manualModeRelay2 ? " (MANUAL)" : " (AUTO)");

      // Enviar por radio
      radio.stopListening();
      bool success = radio.write(&waterData, sizeof(waterData));
      Serial.println(success ? " Dato enviado con éxito." : "...Esperando respuesta");
```

```
      // Vía ACK payload
      if (success && radio.isAckPayloadAvailable()) {
        RelayControl incoming;
        radio.read(&incoming, sizeof(incoming));
        controlData = incoming;

        if (controlData.manualModeRelay1) {
          digitalWrite(relayPin, controlData.relay1Command ? HIGH : LOW);
          waterData.relayState = digitalRead(relayPin);
        }
        if (controlData.manualModeRelay2) {
          digitalWrite(relayPin2, controlData.relay2Command ? HIGH : LOW);
          waterData.relay2State = digitalRead(relayPin2);
        }
        Serial.println("Orden recibida vía ACK.");
      }
      radio.startListening();

    } else {
      Serial.println("Lectura inválida del sensor.");
    }
  }
}
```

ANEXO 5. Código del nodo Nº05

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#define RELAY1 2    // Pin D2
#define RELAY2 27   // Pin D27

#define CE_PIN 14
#define CSN_PIN 15

RF24 radio(CE_PIN, CSN_PIN);

const byte address[6] = "00005"; // Dirección del nodo 00005 (ventiladores)

// Estructura
struct __attribute__((packed)) FanControl {
  bool fanCommand; // true = encender, false = apagar
};

// Enviado al central
struct __attribute__((packed)) FanStatus {
  bool fanState; // true = encendidos, false = apagados
};

FanStatus statusData;
FanControl controlData;  // lo recibirá vía ACK

unsigned long previousMillis = 0;
const unsigned long interval = 4000; // cada 4 segundos enviar estado

void setup() {
  Serial.begin(9600);
  Serial.println("Nodo 00005 (Ventiladores) iniciando...");

  pinMode(RELAY1, OUTPUT);
  pinMode(RELAY2, OUTPUT);
  digitalWrite(RELAY1, LOW);
  digitalWrite(RELAY2, LOW);

  if (!radio.begin()) {
    Serial.println(" No se pudo iniciar NRF24L01");
  } else {
    radio.enableDynamicPayloads();
    radio.enableAckPayload();
    radio.setPALevel(RF24_PA_MAX);
    radio.setDataRate(RF24_250KBPS);
```

```
    radio.setChannel(100);
    radio.openWritingPipe(address);
    radio.openReadingPipe(1, address);
    radio.startListening();
    Serial.println(" NRF24L01 listo (ACK activado en 00005)");
  }
}

void loop() {
  unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;                    // Envío constate

    statusData.fanState = (digitalRead(RELAY1) == HIGH);

    // Enviar estado por orden ACK
    // Vaciar buffer antes de leer
    radio.flush_rx();
    radio.stopListening();
    bool ok = radio.write(&statusData, sizeof(statusData));
    radio.startListening();

    Serial.print(" Estado enviado → ");
    Serial.print(statusData.fanState ? "ENCENDIDOS" : "APAGADOS");
    Serial.println(ok ? " " : "  (fallo envío)");

    // Leer la orden que viene en el ACK
    if (radio.isAckPayloadAvailable()) {
      radio.read(&controlData, sizeof(controlData));

      if (controlData.fanCommand) {
        digitalWrite(RELAY1, HIGH);
        digitalWrite(RELAY2, HIGH);
        Serial.println(" Orden recibida (ACK) - ENCENDER ventiladores");
      } else {
        digitalWrite(RELAY1, LOW);
        digitalWrite(RELAY2, LOW);
        Serial.println("Orden recibida (ACK) - APAGAR ventiladores");
      }
    }
  }
}
```

ANEXO 6. Código del nodo Central

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <MHZ19.h>
#include <SoftwareSerial.h>
#include <Servo.h>


#define CE_PIN 7
#define CSN_PIN 8

// PIN DE REINICIO SIM7600SA
#define PWRKEY_PIN 9

//Definicion de blynk
#define BLYNK_TEMPLATE_ID   "TMPL2Dg8tbLrb"
#define BLYNK_TEMPLATE_NAME "Invernadero SIM7600"
#define BLYNK_AUTH_TOKEN    "Y4GP4wanv49umJZklOJc9Wb916-sp_pX"

//Tipo de conexion
#define TINY_GSM_MODEM_SIM7600

// SIM7600 SERIAL 2
#define SIM7600 Serial2    // pines 16 (TX2) y 17 (RX2)

//Libreria de red
#include <TinyGsmClient.h>
#include <BlynkSimpleTinyGSM.h>

// RElE LOCAL
#define RELAY_LOCAL_PIN 46

Servo servoVentilador;
#define SERVO_PIN 6   // pin de servomotor local

// Sensor de pH
#define PH_PIN A0
int sensorValue = 0;
float voltage = 0;
float phValue = 0;

// Sensor MH-Z19B (CO₂)
#define RX_PIN 19    // RX1 en Mega, pin 6)
#define TX_PIN 18    // TX1 en Mega, pin 5)
#define BAUDRATE 9600
```

```
MHZ19 myMHZ19;
HardwareSerial& mySerial = Serial1;

unsigned long previousMillisCO2 = 0;
const long intervalCO2 = 2000; //lee MH-Z19B

RF24 radio(CE_PIN, CSN_PIN);

// Direcciones de los nodos
const byte addressNode1[6] = "00001"; // Nodo de Temperatura y humedad de
aire
const byte addressNode2[6] = "00002"; // Nodo de Temperatura y humedad de
aire
const byte addressNode3[6] = "00003"; // Nodo de Temperatura y humedad de
aire
const byte addressNode4[6] = "00004"; // Nodo de nivel de agua
const byte addressNode5[6] = "00005"; // Nodo de ventiladores

// Estructura para nodos 00001, 00002 y 00003
struct __attribute__((packed)) DataPacket { // Estructura de paquete
  float temperature;
  float humidity;
  float soilMoistureAverage;
  bool dht1_status;
  bool dht2_status;
  bool dht3_status;
  uint8_t node_id;
};

// Estructura para nodo 00004 (nivel de agua + estado de ambos reles)
struct __attribute__((packed)) WaterLevelData {
  float distance;
  bool relayState;   // true = llenando, false = lleno
  bool relay2State;  // estado del rele de riego
};

// Estructura para enviar orden del segundo rele a nodo 00004
struct __attribute__((packed)) RelayControl {
  bool manualModeRelay1; // true = manual, false = automático (tanque)
  bool relay1Command;    // control de tanque
  bool manualModeRelay2; // true = manual, false = automático (riego)
  bool relay2Command;    // control del riego
};

// Estructura para control de ventiladores nodo 00005
struct __attribute__((packed)) FanControl {
  bool fanCommand; // true = encender, false = apagarlos
};
```

```
// Estructura para recibir confirmación del nodo 5
struct __attribute__((packed)) FanStatus {
  bool fanState; // true = encendidos, false = apagados
};

DataPacket data;
WaterLevelData waterData;
FanStatus fanStatus;

// Tiempo de escucha por nodo en ms
const unsigned long tiempoEscucha = 6000; // 6 segundos

// Variables para promedios
float lastTemp[3];
float lastHum[3];
float lastSoil[3];
bool lastValid[3];

// -------- Variables globales  --------
float avgT = 0;    // promedio de temperatura
float avgH = 0;    // promedio de humedad
float avgS = 0;    // promedio de humedad del suelo
int CO2_ppm = 0;   // concentración de CO₂
// -------- Variables globales  --------

// --- Control de reconexión SIM7600 ---
int intentosFallidos = 0;              // contador de intentos fallidos
unsigned long lastAttempt = 0;         // tiempo del último intento

void reiniciarSIM7600() {
  Serial.println(" Reiniciando SIM7600 con PWRKEY...");
  digitalWrite(PWRKEY_PIN, HIGH);
  delay(1500);                  // pulso de 1.5s en PWRKEY
  digitalWrite(PWRKEY_PIN, LOW);
  delay(10000);                 // esperar al arranque del modulo
}

unsigned long ultimoEnvioBlynk = 0; // para controlar envío de sensores


// --- Control de riego de temporización + cooldown ---
unsigned long tiempoInicioRiego = 0;
bool riegoActivo = false;
const unsigned long DURACION_RIEGO =  1UL * 60UL * 1000UL; // 1 min

const unsigned long TIEMPO_ASENTAMIENTO = 30UL * 60UL * 1000UL;  // 30 min
bool esperandoAsentamiento = false;
unsigned long inicioAsentamiento = 0;
```

```
bool cooldownEnCurso = false;
unsigned long inicioCooldown = 0;
const unsigned long DURACION_COOLDOWN = 6UL * 60UL * 60UL * 1000UL; // 6 h

// buffer para la última orden al nodo 4 y 5
RelayControl lastCtrl4 = {false, false, false, false};
FanControl lastCtrl5 = {false};  // buffer de última orden para nodo 5

  // Variables de control que luego seran modificados por la app (Blynk IoT)
  static bool modoManual = false; // modo manual para el riego/SIM7600
  static bool ordenRiego = false; // ON/OFF para el SIM7600

   static bool manualModeRelay1 = false;   // modo manual para el tanque/SIM7600
  static bool relay1Command    = false;  // ON/OFF para el SIM7600

  static bool modoManualFan = false;   // modo manual ventiladores/SIM7600
  static bool ordenFan = false;        // ON/OFF para el SIM7600

   static bool modoManualLocal = false; //   modo manual del ventilador local/SIM7600
  static bool ordenLocal = false;      //  ON/OFF para el SIM7600

// Inicialización primaria del SIM7600
void inicializarSIM7600() {
  Serial.println(" Inicializando SIM7600...");
  SIM7600.begin(115200); // Funcionamiento del SIM7600
  static TinyGsm modem(Serial2);

  modem.restart();

  // Funcionamiento solo en LTE + data
  modem.sendAT("+CNMP=38"); modem.waitResponse();
  modem.sendAT("+CMNB=2");  modem.waitResponse();

  if (!modem.waitForNetwork(30000)) {
    Serial.println(" No se registró en la red");
  } else {
    Serial.println(" Red detectada");
  }

  if (!modem.gprsConnect("claro.com.pe", "", "")) {// operadora a usar
    Serial.println(" Error conectando APN");
  } else {
    Serial.println(" Datos conectados");
  }
```

```
  Serial.print("IP asignada: ");// IP dictada por Claro
  Serial.println(modem.localIP());

  Serial.print(" Señal (CSQ): ");// Nivel de señal
  Serial.println(modem.getSignalQuality());

  Serial.print(" Registro (CREG): ");// Comando AT 0.1-> registro ok 0.2-
> buscando red
  Serial.println(modem.getRegistrationStatus());

  // Conexión a Blynk con SIM7600
    Blynk.begin(BLYNK_AUTH_TOKEN,   modem,   "claro.com.pe",   "",   "",
"blynk.cloud", 8080);

  if (Blynk.connect(20000)) {
    Serial.println(" Conectado a Blynk!");
  } else {
    Serial.println(" Fallo conexión a Blynk");
  }
}

void setup() {

  Serial.begin(9600);



  pinMode(PWRKEY_PIN, OUTPUT);
  digitalWrite(PWRKEY_PIN, LOW);  // inicio sin actividad

  // Encender SIM7600 al inicio con PWRKEY
  Serial.println(" Encendiendo SIM7600 desde Arduino...");
  digitalWrite(PWRKEY_PIN, HIGH);
  delay(1500);              // pulso de 1.5s
  digitalWrite(PWRKEY_PIN, LOW);
  delay(10000);             // esperar de arranque

    // Inicializar primaria del SIM7600
    inicializarSIM7600();

  for (int i = 0; i < 3; ++i) {
    lastTemp[i] = NAN;
    lastHum[i] = NAN;
    lastSoil[i] = NAN;
    lastValid[i] = false;
  }

  if (!radio.begin()) {
```

```
    Serial.println(" Error al inicializar el NRF24L01.");
    while (1);
  }

  radio.setPALevel(RF24_PA_MAX); //  Potencia al maximo
  radio.setDataRate(RF24_250KBPS); // Velocidad aplicada (alcance)
  radio.setChannel(100);
  radio.setRetries(15, 30);
  radio.enableDynamicPayloads();
  radio.enableAckPayload();   //  Recepcion por ACK

  // Inicializar MH-Z19B
  mySerial.begin(BAUDRATE);
  myMHZ19.begin(mySerial);
  myMHZ19.autoCalibration(false);

  // Inicializar rele local
  pinMode(RELAY_LOCAL_PIN, OUTPUT);
  digitalWrite(RELAY_LOCAL_PIN, LOW);

  //Inicia Servomotor local
  servoVentilador.attach(SERVO_PIN);
  servoVentilador.write(0); // posición inicial 0 grados


  // Serial Monitor de la PC
  Serial.begin(9600);
  // SIM7600 módulo para funcionamiento
  SIM7600.begin(115200);  // velocidad del  SIM7600
  static TinyGsm modem(Serial2);

// Segunda inicialización del SIM7600
Serial.println(" Inicializando SIM7600...");
modem.restart();

// Comprobar conexion doble
modem.sendAT("+CNMP=38"); modem.waitResponse();
modem.sendAT("+CMNB=2");  modem.waitResponse();

if (!modem.waitForNetwork(30000)) {
  Serial.println(" No se registró en la red");
} else {
  Serial.println(" Red detectada");
}

if (!modem.gprsConnect("claro.com.pe", "", "")) {
  Serial.println(" Error conectando APN");
} else {
```

```
  Serial.println(" Datos conectados");
}

Serial.print(" IP asignada: ");
Serial.println(modem.localIP());

Serial.print(" Señal (CSQ): ");
Serial.println(modem.getSignalQuality());

Serial.print(" Registro (CREG): ");
Serial.println(modem.getRegistrationStatus());

// Conexión a Blynk
Blynk.begin(BLYNK_AUTH_TOKEN,    modem,    "claro.com.pe",    "",    "",
"blynk.cloud", 8080);

if (Blynk.connect(20000)) {
  Serial.println(" Conectado a Blynk!");
} else {
  Serial.println(" Fallo conexión a Blynk");
}

   Serial.println("  Receptor  listo.  Escuchando  nodos  00001,  00002,
00003,  00004 y 00005...");

  // 🩹 Forzar riego manual
lastCtrl4.manualModeRelay2 = false;
lastCtrl4.relay2Command    = false;
}

void loop() {
  if (!Blynk.connected()) {
    if (millis() - lastAttempt > 10000) {   // cada 10s intenta reconectar
a la app Blynk IoT
      Serial.println(" Intentando reconectar...");
      if (Blynk.connect(5000)) {
        Serial.println(" Reconectado!"); // Reconexión confirmada
        intentosFallidos = 0;
      } else {
        intentosFallidos++;
        Serial.print(" Fallo intento ");
        Serial.println(intentosFallidos);

        if (intentosFallidos >= 3) {
          reiniciarSIM7600();
          inicializarSIM7600();  // Llama a inicio por falla de intentos
          intentosFallidos = 0;
        }
```

```
      }
      lastAttempt = millis();
    }
  }
  Blynk.run();

  escucharNodo(addressNode1, 1); // ID nodo 1 (Temperatura y humedad)
  escucharNodo(addressNode2, 2); // ID nodo 2 (Temperatura y humedad)
  escucharNodo(addressNode3, 3); // ID nodo 3 (Temperatura y humedad)
  escucharNodo4(addressNode4, 4);// ID nodo 4 (Nivel de agua y riego)
  escucharNodo5(addressNode5, 5);  // ID nodo 5 (Ventiladores)

  // Después de leer todos los nodos, calcular promedios y enviar orden
  calcularYMostrarPromedios();

  // Lectura de sensores locales
  leerSensorPH();
  leerSensorCO2();

  Serial.println("====== Ciclo completado ======\n");

  // Ordena para el riego
  if (modoManual) {
  lastCtrl4.manualModeRelay2 = true;
  lastCtrl4.relay2Command    = ordenRiego;
} else {
  lastCtrl4.manualModeRelay2 = false; // Vuelve a automatico

}

// Orden para el tanque
if (manualModeRelay1) {
  lastCtrl4.manualModeRelay1 = true;
  lastCtrl4.relay1Command    = relay1Command;
} else {
   lastCtrl4.manualModeRelay1  =  false;  //  Vuelve  llenado  de  tanque
automatico

}
// Para los ventiladores nodo 5
if (modoManualFan) {
   lastCtrl5.fanCommand = ordenFan;// usar la orden que venga del Blynk
IoT/SIM7600
  Serial.print("[MANUAL] Orden para nodo 5: ");
    Serial.println(ordenFan   ?   "Encender   ventiladores"   :   "Apagar
ventiladores");
}
```

```
// Para el rele local
if (modoManualLocal) {
  digitalWrite(RELAY_LOCAL_PIN, ordenLocal ? HIGH : LOW);
  Serial.print(" [MANUAL] Relé local: ");
  Serial.println(ordenLocal ? "ENCENDIDO" : "APAGADO");

  // Servo sigue al modo manual
  if (ordenLocal) {
    servoVentilador.write(90);   // abrir
    } else {
      servoVentilador.write(0);    // cerrar
  }
}
//  si  NO  está  en  manual,  lastCtrl5  se  actualizará  normalmente  en
calcularYMostrarPromedios()

if (millis() - ultimoEnvioBlynk >= 600000UL) {   // cada 10 minutos envio
a Blynk ioT
  ultimoEnvioBlynk = millis();

  if (Blynk.connected()) {
    Blynk.virtualWrite(V0, avgT); // Envio a Blynk promedio temperatura
    Blynk.virtualWrite(V1, avgH); // Envio a Blynk promedio humedad
     Blynk.virtualWrite(V2, avgS); // Envio a Blynk promedio humedad del
suelo
    Blynk.virtualWrite(V3, CO2_ppm); // Envio Blynk CO2
    Blynk.virtualWrite(V4, phValue); // Envio Blynk pH
     Blynk.virtualWrite(V5, waterData.distance); // Envio Blynk nivel de
agua(distancia)

    Serial.println(" Datos enviados a Blynk");
  }
}

  delay(1000);
}
// Escuchar NODO 1, 2 y 3
void escucharNodo(const byte address[6], uint8_t idNodo) {
  radio.stopListening();
  radio.openReadingPipe(1, address);
  radio.startListening();

  unsigned long startTime = millis();
  bool received = false;
  byte pipeNum;

  while (millis() - startTime < tiempoEscucha) {
    if (radio.available(&pipeNum)) {
```

```
      uint8_t len = radio.getDynamicPayloadSize();

      if (len == sizeof(DataPacket)) {
        radio.read(&data, sizeof(data));
        received = true;

        Serial.print(" Datos recibidos del Nodo ");
        Serial.print(data.node_id); // ID con la que se identifica un NODO
        Serial.println(":");

        Serial.print("Temperatura: ");
        Serial.print(data.temperature);
        Serial.println(" °C");

        Serial.print("Humedad: ");
        Serial.print(data.humidity);
        Serial.println(" %");

        Serial.print("Humedad del suelo (promedio): ");
        Serial.print(data.soilMoistureAverage);
        Serial.println(" %");

      //Comprobar el estado de los dht
        Serial.println("Estado de los sensores DHT:");
        Serial.print("  DHT1: ");
        Serial.println(data.dht1_status ? "OK" : "Fallo");
        Serial.print("  DHT2: ");
        Serial.println(data.dht2_status ? "OK" : "Fallo");
        Serial.print("  DHT3: ");
        Serial.println(data.dht3_status ? "OK" : "Fallo");

        Serial.println("--------------------------------");

        uint8_t idx = (idNodo - 1);
        if (idx < 3) {
          lastTemp[idx] = data.temperature;
          lastHum[idx] = data.humidity;
          lastSoil[idx] = data.soilMoistureAverage;
          lastValid[idx] = true;
        }

        break;
      } else {
        radio.flush_rx();
          Serial.println("Paquete descartado (tamaño desconocido)."); //
evitar fallas de transito
      }
    }
```

```
  }

  radio.stopListening();

  if (!received) {
    Serial.print(" No se recibieron datos del Nodo ");
    Serial.println(idNodo);
    Serial.println("--------------------------------");
    uint8_t idx = (idNodo - 1);
    if (idx < 3) lastValid[idx] = false;
  }
}
// Escuchar Nodo 4 (Riego y tanque)
void escucharNodo4(const byte address[6], uint8_t idNodo) {
  radio.stopListening();
  radio.openReadingPipe(1, address);

    // Cargar la orden pendiente en el ACK
  bool ackOK = radio.writeAckPayload(1, &lastCtrl4, sizeof(lastCtrl4));
  Serial.print(" Orden preparada para nodo 4 (ACK): ");
   Serial.println(lastCtrl4.relay2Command  ?  "Encender  riego"  :  "Apagar
riego");
   Serial.println(ackOK  ?  "  Preparada  con  éxito"  :  "  Error  al  preparar
ACK");

  radio.startListening();

  unsigned long startTime = millis();
  bool received = false;
  byte pipeNum;

  while (millis() - startTime < tiempoEscucha) {
    if (radio.available(&pipeNum)) {
      uint8_t len = radio.getDynamicPayloadSize();

      if (len == sizeof(WaterLevelData)) {
        radio.read(&waterData, sizeof(waterData));
        received = true;

        Serial.print(" Datos recibidos del Nodo ");
        Serial.print(idNodo);
        Serial.println(" (Nivel de Agua):");

        Serial.print("Distancia: ");
        Serial.print(waterData.distance, 2);
        Serial.println(" cm");

        Serial.print("Estado del relé tanque: ");
```

```
        Serial.println(waterData.relayState ? "Llenando" : "Lleno");
        Serial.print("Estado del relé riego: ");
        Serial.println(waterData.relay2State ? "Encendido" : "Apagado");

        Serial.println("--------------------------------");
        break;
      } else {
        radio.flush_rx();
         Serial.println(" Paquete descartado (tamaño desconocido - nodo
4).");
      }
    }
  }

  radio.stopListening();

  if (!received) {
    Serial.print(" No se recibieron datos del Nodo ");
    Serial.println(idNodo);
    Serial.println("--------------------------------");
  }
}

// Escuchar nodo 5 (ventiladores)
void escucharNodo5(const byte address[6], uint8_t idNodo) {
  radio.stopListening();
  radio.openReadingPipe(1, address);
  radio.startListening();

  unsigned long startTime = millis();
  bool received = false;
  byte pipeNum;

  while (millis() - startTime < tiempoEscucha) {
    if (radio.available(&pipeNum)) {
      uint8_t len = radio.getDynamicPayloadSize();

      if (len == sizeof(FanStatus)) {
                bool   ackOK   =   radio.writeAckPayload(1,   &lastCtrl5,
sizeof(lastCtrl5));

        radio.read(&fanStatus, sizeof(fanStatus));
        received = true;

        Serial.print(" Datos recibidos del Nodo ");
        Serial.print(idNodo);
        Serial.println(" (Ventiladores):");
```

```
        Serial.print("Estado ventiladores: ");
        Serial.println(fanStatus.fanState ? "ENCENDIDOS" : "APAGADOS");


        Serial.print(" Orden preparada para nodo 5 (ACK): ");
        Serial.println(lastCtrl5.fanCommand ? "Encender" : "Apagar");
       Serial.println(ackOK ? " Preparada con éxito" : " Error al preparar
ACK");

        Serial.println("--------------------------------");
        break;
      } else {
        radio.flush_rx();
         Serial.println(" Paquete descartado (tamaño desconocido del nodo
5).");
      }
    }
  }

  radio.stopListening();

  if (!received) {
    Serial.print(" No se recibieron datos del Nodo ");
    Serial.println(idNodo);
    Serial.println("--------------------------------");
  }
}

void calcularYMostrarPromedios() {
  float sumT = 0, sumH = 0, sumS = 0;
  int count = 0;

  for (int i = 0; i < 3; ++i) {
    if (lastValid[i]) {
      sumT += lastTemp[i];
      sumH += lastHum[i];
      sumS += lastSoil[i];
      ++count;
    }
  }


  if (count == 0) {
    Serial.println(" No hay lecturas válidas de los nodos 00001-00003 para
promediar.");
    return;
  }
```

```
  avgT = sumT / count;
  avgH = sumH / count;
  avgS = sumS / count;

  Serial.println(" Promedio de nodos 00001-00003:");
  Serial.print("  Prom. Temperatura: ");
  Serial.print(avgT, 2);
  Serial.println(" °C");

  Serial.print("  Prom. Humedad: ");
  Serial.print(avgH, 2);
  Serial.println(" %");

  Serial.print("  Prom. Humedad Suelo: ");
  Serial.print(avgS, 2);
  Serial.println(" %");

  Serial.print("  (Usando "); // Cantidad de nodos en escucha
  Serial.print(count);
  Serial.println(" nodos válidos)"); // Validacion de nodos escuhables
  Serial.println("--------------------------------");

  // --- Enviar orden al nodo 4 (riego y tanque) ---
  RelayControl controlData;
  controlData.manualModeRelay1 = false;
  controlData.relay1Command = false;
  controlData.manualModeRelay2 = false;
  controlData.relay2Command = false;

  unsigned long ahora = millis();

  if (lastCtrl4.manualModeRelay2) {
    controlData.manualModeRelay2 = true;
    controlData.relay2Command   = lastCtrl4.relay2Command;

    Serial.print(" [MANUAL] Orden para nodo 4: ");
    Serial.println(controlData.relay2Command ? "Encender riego" : "Apagar
riego");
      // Enviar la orden manual riego
  radio.writeAckPayload(1, &lastCtrl4, sizeof(lastCtrl4));
  Serial.println("📤 [MANUAL INMEDIATO] Orden enviada al nodo 4 (riego).");

  } else {
    // Lógica automática de riego
    controlData.manualModeRelay2 = false;

    if (!riegoActivo) {
      if (esperandoAsentamiento) {
```

```
        // Estamos en asentamiento → no regar
        if (ahora - inicioAsentamiento >= TIEMPO_ASENTAMIENTO) {
          esperandoAsentamiento = false;
          if (avgS >= 70.0) {
            controlData.relay2Command = false;
            Serial.println(" Humedad ≥70% después del asentamiento. Ciclo
finalizado.");

              // Iniciar cooldown de 6h
            cooldownEnCurso = true;
            inicioCooldown = ahora;
            Serial.println(" Iniciando cooldown de 6 horas...");

          } else {
            riegoActivo = true;
            tiempoInicioRiego = ahora;
            controlData.relay2Command = true;
             Serial.println(" Humedad <70% tras asentamiento → iniciando
otro bloque.");
          }
        } else {
          controlData.relay2Command = false;
        }

      } else if (cooldownEnCurso) {// Cooldown, se tiene que esperar

        if (ahora - inicioCooldown >= DURACION_COOLDOWN) {
          cooldownEnCurso = false;
         Serial.println(" Fin del cooldown → iniciando bloque de riego.");
        } else {
          controlData.relay2Command = false;
          Serial.println(" En cooldown → esperando 6 horas.");
        }

      } else {
        // Arranque Inicial, si ≤60% → regar directo
        if (avgS <= 60.0) {
          riegoActivo = true;
          tiempoInicioRiego = ahora;
          controlData.relay2Command = true;
          Serial.println(" Humedad ≤60% → iniciando riego inmediato.");
        } else {
          controlData.relay2Command = false;
          Serial.println(" Humedad >60% → no se riega.");
        }
      }

    } else if (riegoActivo) {
```

```
      // Riego en curso bloque de 1 minutos
      if (ahora - tiempoInicioRiego >= DURACION_RIEGO) {
        riegoActivo = false;
        esperandoAsentamiento = true;
        inicioAsentamiento = ahora;
        controlData.relay2Command = false;
         Serial.println(" Fin de bloque de riego. Esperando que el suelo
absorba agua...");
      } else {
        controlData.relay2Command = true;
      }
    }
  }

  // Se Guarda la orden en el buffer que se mandará en el próximo ACK del
nodo 4
// Solo actualizar si no esta manual
if (!modoManual) {
  lastCtrl4 = controlData;
}

  Serial.print(" Orden calculada para nodo 4: ");
   Serial.println(lastCtrl4.relay2Command  ?  "Encender  riego"  :  "Apagar
riego");
  Serial.println(" Orden almacenada, se enviará en el próximo ACK del nodo
4");



  // Enviar orden al nodo 5 (ventiladores)
  FanControl fanData;
  fanData.fanCommand = false;

  if (avgT >= 30.0 || avgH >= 70.0) {
    fanData.fanCommand = true;
  } else if (avgT <= 20.0 || avgH <= 60.0) {
    fanData.fanCommand = false;
  } else {
    return;
  }

  // Relé local
  if (!modoManualLocal) {
  digitalWrite(RELAY_LOCAL_PIN, fanData.fanCommand ? HIGH : LOW);
  Serial.print("Relé local (ventiladores): ");
  Serial.println(fanData.fanCommand ? "ENCENDIDO" : "APAGADO");
  // Servo sigue al relé local en automático
  if (fanData.fanCommand) {
```

```
    servoVentilador.write(90);   // abrir
  } else {
    servoVentilador.write(0);    // cerrar
    }

  } else {
  Serial.println(" Relé local en modo manual ");
}

  // Envío al nodo 5 y guardar en el buffer para ACK
  lastCtrl5 = fanData;
  Serial.print("Orden almacenada para nodo 5: ");
  Serial.println(lastCtrl5.fanCommand ? "Encender" : "Apagar");
}

// Función para clasificar el pH
const char* clasificarPH(float pH) {
  if (pH < 0 || pH > 14) return "fuera de rango";
  if (pH >= 6.5 && pH <= 7.5) return "neutro";
  if (pH >= 5.0 && pH < 6.0) return "ligeramente ácido";
  if (pH >= 3.0 && pH < 4.0) return "ácido";
  if (pH >= 0.0 && pH < 2.0) return "muy ácido/corrosivo";
  if (pH > 8.0 && pH <= 9.0) return "ligeramente alcalino";
  if (pH > 10.0 && pH < 10.50) return "alcalino";
  if (pH >= 11.0 && pH <= 12.0) return "muy alcalino/corrosivo";
  if (pH >= 13.0 && pH <= 14.0) return "extremadamente alcalino/corrosivo";
  return "indefinido";
}

// Función para leer sensor de pH
void leerSensorPH() {
  long sum = 0;
  const int muestras = 20; // número de muestras

  for (int i = 0; i < muestras; i++) {
    sum += analogRead(PH_PIN);
    delay(10);
  }

  sensorValue = sum / muestras;

  // Convertir a voltaje ajustar según fue medido, 5V o menor segun medida)
  voltage = sensorValue * (3.90 / 1023.0);

  // Fórmula aproximada ajustar con la calibración fisica
  phValue = 7 + ((2.5 - voltage) / 0.18);

  Serial.print(" Lectura de pH -> ADC: ");
```

```
  Serial.print(sensorValue);
  Serial.print(" | Voltaje: ");
  Serial.print(voltage, 2);
  Serial.print(" V | pH: ");
  Serial.println(phValue, 2);
  Serial.println("--------------------------------");

    const char* categoria = clasificarPH(phValue);
  Serial.print(" | Clasificación: ");
  Serial.println(categoria);
  Serial.println("--------------------------------");
}
void leerSensorCO2() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillisCO2 >= intervalCO2) {
    previousMillisCO2 = currentMillis;

    CO2_ppm = myMHZ19.getCO2();
    Serial.print(" Lectura de CO₂: ");
    if (CO2_ppm > 0) {
      Serial.print(CO2_ppm);
      Serial.println(" ppm");
    } else {
      Serial.println(" Error al leer CO₂");
    }
    Serial.println("--------------------------------");
  }
}

BLYNK_WRITE(V10) { modoManual = param.asInt(); } // Mono manual o automtico
riego lectura enviada del Blynk
BLYNK_WRITE(V11) { ordenRiego = param.asInt(); } // ON-OFF riego enviada
por Blynk
BLYNK_WRITE(V12) { manualModeRelay1 = param.asInt(); }// Mono manual o
automatico tanque lectura enviada del Blynk
BLYNK_WRITE(V13) { relay1Command = param.asInt(); }// ON-OFF tanque enviada
por Blynk
BLYNK_WRITE(V14) { modoManualFan = param.asInt(); }// Mono manual o
automatico ventiladores lectura enviada del Blynk
BLYNK_WRITE(V15) { ordenFan = param.asInt(); }// ON-OFF ventiladores
enviada por Blynk
BLYNK_WRITE(V16) { modoManualLocal = param.asInt(); }// Mono manual o
automatico ventilador local lectura enviada del Blynk
BLYNK_WRITE(V17) { ordenLocal = param.asInt(); }// ON-OFF ventilador local
enviada por Blynk
```

ANEXO 7. Matriz de consistencia

PROBLEMA	OBJETIVOS	VARIABLE	DIMENSIÓN	INDICADOR
Problema general ¿En qué medida el diseño e implementación de un sistema de control basado en IoT permitirá mejorar la producción de orégano en la Yarada Los Palos, en la región Tacna?	**Objetivo general** Diseñar e implementar un prototipo por IoT que tenga un sistema de control que mejore el cultivo de orégano en un invernadero mediante la monitorización clave como la humedad, temperatura, entre otros, en la Yarada Los Palos, en la región Tacna.	Es básica y aplicada, ya que, partiendo de un problema y unos objetivos definidos por los gestores del proyecto, utilizan metodologías para la recolección, interpretación de la información.	Sistema IoT	• Procesamiento y trasmisión de la data • Base de Datos • Interfaz Gráfica • Control Automático/Manual

Problemas específicos ¿Cuáles son los parámetros climáticos necesarios para la producción de orégano? ¿Qué tecnología de hardware y software será la más adecuada para el diseño del sistema de control?	**Objetivos específicos** Determinar los parámetros climáticos necesarios para la producción de orégano. Determinar la tecnología de hardware y software que sea la más adecuada para el diseño del sistema de control basado en IoT.		Monitoreo de calidad ambiental en invernadero	• Cantidad de gas CO2 (ppm) • Temperatura • Humedad • Humedad del suelo • pH • Distancia