UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"DISEÑO DE UN SISTEMA IOT PARA EL MONITOREO Y CONTROL DEL CULTIVO DE LECHUGAS EN UN INVERNADERO"

PARA OPTAR:

TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR:

Bach. SEBASTIÁN BERRIOS GÓMEZ

TACNA – PERÚ

2022

UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"DISEÑO DE UN SISTEMA IOT PARA EL MONITOREO Y CONTROL DEL CULTIVO DE LECHUGAS EN UN INVERNADERO"

Tesis sustentada y aprobada el 28 de junio de 2022; estando el jurado calificador integrador por:

PRESIDENTE: **Mag. ANIBAL JUAN ESPINOZA ARANCIAGA**

SECRETARIO: **Mtro. MARKO JESÚS POLO CAMACHO**

VOCAL: **Mag. MARCO ANTONIO SEBASTIÁN COLOMA YUNGANINA**

ASESOR: **Ing. HUGO JAVIER RIVERA HERRERA**

DECLARACIÓN JURADA DE ORIGINALIDAD

Yo, Sebastián Berrios Gómez, en calidad de Bachiller de la Escuela Profesional de Ingeniería Electrónica de la Facultad de Ingeniería de la Universidad Privada de Tacna, identificado con DNI 72235848.

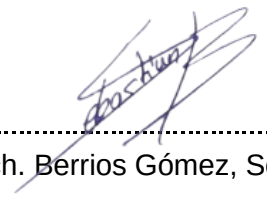
Declaro bajo juramento que:

1. Soy autor de la tesis titulada:
   "Diseño de un sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero"
   la misma que presento para optar el:
   Título Profesional de Ingeniero Electrónico.
2. La tesis no ha sido plagiada ni total ni parcialmente, para la cual se han respetado las normas internacionales de citas y referencias para las fuentes consultadas.
3. La tesis presentada no atenta contra derechos de terceros.
4. La tesis no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
5. Los datos presentados en los resultados son reales, no han sido falsificados, ni duplicados, ni copiados.

Por lo expuesto, mediante la presente asumimos frente a la universidad cualquier responsabilidad que pudiera derivarse por la autoría, originalidad y veracidad del contenido de la tesis, así como por los derechos sobre la obra presentada. En consecuencia, nos hacemos responsables frente a la universidad y a terceros, de cualquier daño que pudiera ocasionar, por el incumplimiento de lo declarado o que pudiera encontrar como causa del trabajo presentado, asumiendo todas las cargas pecuniarias que pudieran derivarse de ello a favor de terceros con motivo de acciones, reclamaciones o conflictos derivados del incumplimiento de lo declarado o las que encontrasen causa en el contenido de la tesis.

De identificarse fraude, piratería, plagio, falsificación o que el trabajo de investigación haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, sometiéndome a la normatividad vigente de la Universidad Privada de Tacna.

Tacna, 10 de julio de 2022



Bach. Berrios Gómez, Sebastián

DNI: 72235848

DEDICATORIA

Le dedico esta tesis a Dios, a mi padre y a mi madre. Sin ellos no hubiese logrado nada de esto. Gracias por haberme convertido en la persona que soy en la actualidad; muchos de mis logros son gracias a ustedes, incluido este.

AGRADECIMIENTO

Mis agradecimientos se dirigen a las personas que siempre confiaron en mí y me apoyaron, sin estas personas nada de esto seria posible, es por eso que le agradezco a:

MI PADRE: Mi ejemplo a seguir, desde pequeño siempre quise ser como tú. Gracias a ti estudie ingeniería y estoy logrando lo que me propuse. Siempre me cuidaste y me diste todo lo que necesite para cumplir mis metas, espero que te sientas orgulloso de mi. Te amo papá.

MI MADRE: Tus consejos y tu apoyo incondicional fueron muy importantes para lograr esto, sin ti no sería como soy. Gracias por siempre escucharme y nunca juzgarme, sé que siempre podre confiar en ti. Te amo mamá.

MIS HERMANOS: Por darme ánimos, siempre a su manera, de ustedes tres aprendí que, sin importar la edad, cualquier persona puede darte el consejo que necesitas en ese momento. Rodrigo, Piero y Betito, gracias por ser los mejores hermanos.

PAPÁ GEORGE Y BIBI: Me cuidaron desde que nací, siempre estuvieron para mí y yo siempre estaré para ustedes, haría cualquier cosa para que se sientan orgullosos de mí.

PAPÁ BETUCO Y MAMÁ MARILU: Gracias por acogerme en su hogar y cuidarme como si fuera su propio hijo en toda mi etapa universitaria, y aun lo siguen haciendo.

MA PETITE: Me motivaste, me aconsejaste y, lo más importante, me ayudaste a encontrar el equilibro entre el trabajo y el estudio, me enseñaste que uno crea el momento perfecto para hacer algo.

MI ASESOR: Mas que un asesor, un amigo. No solo me ayudo en la elaboración de mi tesis, lo hizo durante mis cinco años de carrera. Gracias por todo lo que me enseñó.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

El objetivo de la tesis fue diseñar un sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero. El sistema IoT está conformado por un nodo central, tres nodos sensor, una base de datos y una aplicación móvil. El componente principal de todos los nodos es el microcontrolador ESP32, cuenta con tecnología WiFi y Bluetooth de modo dual integrada y con un microprocesador Tensilica Xtensa LX6. El enfoque principal de la tesis está en el sistema IoT que nos facilita el monitoreo y control del sistema desde cualquier parte del mundo, el único requerimiento es el acceso a internet. Los nodos sensor recolectan las mediciones de temperatura, humedad relativa, humedad del suelo y el nivel del tanque de agua y mediante el protocolo de comunicación ESP-NOW envían la información al nodo central. El nodo central activa y desactiva los actuadores para controlar los parámetros climáticos considerados y sube la información a la base de datos. La aplicación móvil obtiene las lecturas de la base de datos, también es capaz de encender y apagar los actuadores. El sistema está programado para mantener en un rango óptimo la temperatura (Horario Diurno: 15 °C – 20 °C) (Horario Nocturno: 10 °C – 15 °C), humedad relativa (60 % – 70 %) y humedad del suelo (50 % - 75 %) del invernadero, este rango fue determinado por especialistas en agronomía. Para medir la temperatura y humedad relativa se utilizó el sensor DHT22 con rangos de medición de -40 °C a 80 °C, con una precisión de ± 0,5 °C, y 0 % a 100 %, con una precisión de 2 %, respectivamente, para medir la humedad del suelo se utilizó un sensor que está compuesto por dos electrodos (HW-080) y un convertidor (HW-103), este sensor tiene un rango de medición de 0 % a 100 %, y para medir el nivel del tanque se utilizó un sensor HC-SR04 con rangos de medición teórico de 2 cm a 400 cm, con una resolución de 0,3 cm. Los rangos de medición y la precisión de los sensores se obtuvieron de la ficha técnica que brindan los fabricantes. Para controlar estos rangos óptimos se utilizó un ventilador y un calefactor para la temperatura y humedad relativa, y una bomba de agua para la humedad del suelo. Además, la visualización de los datos que recolecten los nodos sensor se puede observar en la base de datos y en la aplicación móvil en tiempo real.

Palabras clave: Aplicación Móvil**;** Internet de las cosas; Sistema de Control**;** Sistema de Monitoreo**;** Sistema IoT.

ABSTRACT

The thesis aimed was to design an IoT system for the monitoring and control of the cultivation of lettuce in a greenhouse. The IoT system consists of a central node, three sensor nodes, a database, and a mobile application. The main component of all nodes is the ESP32 microcontroller, it has integrated dual-mode WiFi and Bluetooth technology and a Tensilica Xtensa LX6 microprocessor. The main focus of the thesis is on the IoT system that facilitates the monitoring and control of the system from anywhere in the world, the only requirement is access to the internet. Sensor nodes collect temperature, relative humidity, soil humidity, and water tank level measurements and, using the ESP-NOW communication protocol, send the information to the central node. The central node turns the actuators on and off to control the climate parameters considered and feeds the information to the database. The mobile application obtains the readings from the database, it is also able to turn the actuators on and off. The system is a program to maintain the optimum temperature range (Daytime: 15 °C – 20 °C) (Nighttime: 10 °C – 15 °C), relative humidity (60 % – 70 %), and soil humidity (50 % – 75 %) of the greenhouse, this range was determined by specialists in agronomy. The DHT22 sensor with measuring ranges from -40°C to 80°C, with an accuracy of ± 0,5°C, and 0 % to 100 %, with an accuracy of 2 %, respectively, were used to measure temperature and relative humidity. A sensor consisting of two electrodes (HW-080) and a converter (HW-103) was used to measure soil moisture, this sensor has a measuring range of 0 % to 100 %, and to measure the level of the tank, an HC-SR04 sensor with theoretical measurement ranges from 2 cm to 400 cm, with a resolution of 0,3 cm, was used. The measurement ranges and accuracy of the sensors were obtained from the technical sheet provided by the manufacturers. To control these optimum ranges, a fan and heater were used for temperature and relative humidity, and a water pump was used for soil moisture. In addition, the display of data collected by sensor nodes can be seen in the database and mobile application in real time.

Keywords: Mobile Application; Internet of Things; Control System; Monitoring System; IoT System.

INTRODUCCIÓN

El mayor problema que tienen los agricultores son las variaciones de clima a lo largo del año, estos cambios climáticos dificultan o a veces hacen imposible cultivar lechugas. La mejor solución para afrontar estos fenómenos son los invernaderos. Un invernadero es un lugar cerrado cubierto con un plástico especial para poder mantener regulada la temperatura y la humedad relativa dentro de él.

El invernadero por sí solo es una buena solución para afrontar los cambios climáticos, pero esto no nos garantiza preservar la temperatura y la humedad relativa optima que necesita la lechuga para crecer y poder ser cosechada. El sistema IoT para el monitoreo y control nos ayudará a cumplir este objetivo, al mantener una temperatura y humedad relativa optima podremos tener más posibilidades para que la lechuga crezca.

La tesis propone solucionar este fenómeno con el diseño de un sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero, este sistema permitirá controlar los parámetros climáticas del invernadero.

La tesis se divide en 5 capítulos:

Capítulo I: Planteamiento del problema: Describe el problema que origina el cambio de clima en las diferentes estaciones del año. Muestra la formulación del problema y da una justificación hacia ella, también se indican los objetivos y la hipótesis.

Capítulo II: Marco Teórico: Resume y concluye sobre tres investigaciones que se tomaron como referencia sobre el tema, señala las bases teóricas y la definición de términos necesarios para comprender la tesis.

Capítulo III: Marco Metodológico: Contiene el tipo y diseño de la investigación y la operacionalización de variables. Además, explica paso a paso la configuración y programación del sistema IoT.

Capítulo IV: Resultados: Realiza distintas pruebas del sistema y se presentan los resultados de la investigación.

Capítulo V: Discusión: Lleva a cabo la discusión con la hipótesis formulada y con los antecedentes del estudio.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1. Descripción del problema

Uno de los problemas que tiene la horticultura es la variación del clima en las diferentes estaciones a lo largo del año. Las heladas agronómicas se miden a 1,50 m de la superficie del suelo, es la disminución de la temperatura del aire a niveles críticos de los cultivos, una vez que la temperatura desciende a 1,5 °C provoca daños al tejido vegetal. Este fenómeno se da con cielo despejado o escasa nubosidad, la disminución de la temperatura se registra en horas de la noche o madrugada. Las heladas alteran la temperatura y la humedad relativa, estos dos parámetros climáticos son fundamentales para el cuidado de cualquier cultivo.

La horticultura es la principal siembra de cultivos de muchos agricultores, la lechuga es uno de los pocos tipos de hortalizas que se pueden sembrar todo el año, pero también es afectada por el problema de las heladas y esto disminuye su producción, esto una gran pérdida para los agricultores por no contar con una solución para afrontar este fenómeno.

Una solución para mejorar el cultivo de lechugas es la implementación de invernaderos, así puede proteger el cultivo del exceso de frio. En la actualidad contamos con tecnología muy eficiente para poder acondicionar el ambiente del interior de los invernaderos, así podremos garantizar la estandarización de la temperatura, humedad relativa y la humedad del suelo del cultivo.

1.2. Formulación del problema

¿El diseño de un sistema IoT permitirá el monitoreo y control del cultivo de lechugas en un invernadero?

1.3. Justificación e importancia

El suelo es el factor más importante al momento de cultivar, varios huertos poseen un suelo muy agradable para la producción de lechugas, pero no siembran todo el año por el cambio de clima, especialmente en la estación de invierno que es donde se producen las heladas. En algunas zonas el efecto de las heladas no es tan grave como en otras zonas que tienen mayor altura sobre el nivel del mar.

Algunos huertos producen grandes cantidades de lechugas al año, pero a veces esa producción no es suficiente para satisfacer su demanda. Este proyecto es una buena alternativa para aumentar la productividad de lechugas en todos los huertos.

Las heladas es el problema que tienen los huertos para poder aumentar la cantidad de lechugas producidas por año, pero implementando este sistema IoT se puede optimizar y mejorar el cultivo de lechugas. Con la implementación de este sistema se podrá controlar la temperatura y la humedad relativa para poder combatir los efectos de las heladas.

El sistema IoT nos ayudará a monitorear y controlar la temperatura, humedad relativa y la humedad del suelo del invernadero desde cualquier lugar, así podremos garantizar un mayor rendimiento de la producción. El sistema nos proporcionará mediante una aplicación móvil los datos en tiempo real para lograr una mejor supervisión de los parámetros considerados.

La aplicación móvil estará diseñada para ejecutarse desde cualquier teléfono inteligente o tablet que cuente con un sistema operativo Android. Esta aplicación facilitará la visualización de los datos que recolecten los sensores que se encontraran dentro del invernadero. También contará con una interfaz muy amigable para que el usuario no tenga inconvenientes al interactuar con ella.

1.4. Objetivos

1.4.1. Objetivo general

- Diseñar un sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero.

1.4.2. Objetivos específicos

a. Identificar los niveles óptimos de temperatura, humedad relativa y humedad del suelo en el cultivo de lechugas.
b. Seleccionar los equipos electrónicos para el diseño del sistema IoT.
c. Implementar el prototipo del sistema IoT.

1.5. Hipótesis

Mediante el diseño de un sistema IoT se monitorea y controla los parámetros del cultivo de lechugas en un invernadero.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes del estudio

Molanes (2019), en la tesis titulada “Diseño e implementación de un sistema electrónico de control basado en FPGA, que optimice las condiciones climáticas de un invernadero para el cultivo de tomate en la ciudad de Tacna, 2017” utilizo un dispositivo FPGA para el monitoreo de temperatura y humedad relativa y para el control de humedad del suelo, se llegó a la conclusión que este diseño es reconfigurable y adaptable a mejoras sin necesidad de hacerle grandes cambios a toda la placa. También, supero la fase de pruebas en escenarios típicos de un invernadero.

Reyna (2015), en la tesis titulada “Sistema automatizado para el monitoreo y control de humedad de un invernadero” desarrolló su algoritmo usando el lenguaje C y el programa AVR Studio 4, realizó su implementación usando el microcontrolador Atmega8 e implementó una la tarjeta de comunicación serial usando el Max232, se llegó a la conclusión que al aplicar el método de control On/Off el sistema otorgaría cierta autonomía al invernadero sin depender constantemente del operario, debido a que el sistema mantendría un valor adecuado de humedad dentro del invernadero.

Hernández (2019), en la tesis titulada “Desarrollo de un Sistema de monitorización y control de un invernadero aplicando Tecnología IoT” uso la placa ESP32-DevKitC para el sistema de monitoreo y control de un invernadero, se llegó a la conclusión que todavía queda mucho trabajo por hacer para conseguir un producto comercializable, por ejemplo, diseñando una placa industrial profesional.

2.2. Bases teóricas

2.2.1. Invernaderos

Iglesias (2006) determinó que un invernadero es un recinto cerrado delimitado por una estructura de madera o metal, recubierta de vidrio u otro material plástico transparente, en el que se suelen cultivar hortalizas y plantas ornamentales en épocas en que las condiciones climáticas exteriores no son adecuadas, y por tanto no se puede obtener el producto deseado. La eficiencia y la funcionalidad son las dos principales características que debe poseer un invernadero para ser apto para cultivar. La eficiencia se entiende como la capacidad para condicionar ciertos

elementos principales climáticos dentro del límite definido según los requerimientos fisiológicos de los cultivos, y la funcionalidad es el conjunto de requerimientos que permiten el mejor efecto invernadero, tanto desde el punto de vista técnico como económico.

2.2.2. Lechugas

Cárdenaz et al. (2012) estableció que la lechuga es una hortaliza que tiene hojas sueltas o acogolladas, listo para ser consumida directamente en ensaladas u otras preparaciones gracias a sus características organolépticas. Esta hortaliza es una planta bienal o anual; permanece a la familia de las compuestas Compositae, subfamilia Chicorioideae.

Las condiciones agroclimatológicas para el cultivo de lechuga se especifican en la Tabla 1.

Tabla 1

Condiciones Agroclimatológicas

Parámetro	Rango óptimo
Temperatura del ambiente	15 °C – 20 °C (Día) 10 °C – 15 °C (Noche)
Humedad relativa	60 % – 70 %
pH del suelo	5.7 – 6.5

Nota. Adaptado de Condiciones agroclimatológicas. (p. 11), Cámara de Comercio de Bogotá (2015).

Cámara de Comercio de Bogotá (2015) concluyó que es muy importante realizar el riego solo cuando sea el momento en el que el suelo y el cultivo realmente lo necesiten; una gran parte de perdida de producción se debe a una mala decisión de riego.

La humedad del suelo nos indicara el momento exacto en el que el riego debe iniciarse y detenerse: “La humedad del sustrato debe aproximarse a la capacidad de campo (estimada entre el 50 y el 75% del punto de saturación), evitando que se sature puesto que asfixiaría las raíces de las plántulas recién nacidas” (Fueyo et al., s.f., p. 3).

En la Tabla 2 se puede apreciar los diferentes grados de humedad y sus porcentajes.

Tabla 2

Grado y Porcentaje de la Humedad del Suelo

Grado de Humedad	Tacto	Porcentaje de Humedad
Seco	Polvo seco	Ninguna (0 %)
Bajo	Se desmorona, pero no se aglutina	25 % o menos
Medio	Se desmorona, pero se aglutina	25 % – 50 %
Aceptable	Se forma bola y se aglutina (presión)	50 % – 75 %
Excelente	Se forma bola, se aglutina (amasable)	75 % – 100 %
Húmedo	Chorrea agua cuando se aprieta	Sobrecapacidad

Nota. Adaptado de Determinación del contenido de humedad del suelo por medio del tacto. (p. 29), Cámara de Comercio de Bogotá (2015).

2.2.3. Sensor

Alciatore (2008) afirma que un sensor es parte de un sistema mecatrónico o metrológico que detecta la magnitud de un parámetro físico y lo convierte en una señal que un sistema pueda procesar. Los sistemas de monitoreo y control requieren sensores para medir cantidades físicas como posición, distancia, fuerza, tensión, temperatura, vibración y aceleración.

Existe una gran variedad de sensores: “En sistemas electrónicos, los sensores son los elementos encargados de obtener información. Son llamados técnicamente transductores, y son capaces de convertir cualquier magnitud física, química o biológica en una magnitud eléctrica” (Guarella et al., 2011. p. 3).

2.2.3.1. Sensor de temperatura y humedad (DHT22)

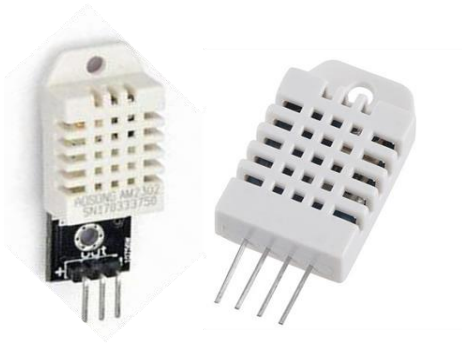
El sensor DHT22, también conocido como AM2302, es un sensor digital de temperatura y humedad relativa. El sensor incluye un sensor capacitivo de humedad y un sensor de temperatura NTC de alta precisión conectado a un chip de 8 bits, muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica).

En la Figura 1 se muestra dos imágenes del sensor de temperatura y humedad, en la imagen izquierda podemos observar solo el sensor mientras que en la imagen

derecha se aprecia el sensor en un pequeño modulo, este módulo facilita sus conexiones eléctricas.

Figura 1

Sensor de Temperatura y Humedad (DHT22)



En la Tabla 3 se encuentran algunos de los parámetros técnicos del sensor de temperatura y humedad.

Tabla 3

Parámetros Técnicos del Sensor DHT22

Parámetros técnicos	
Rango de medición (Temperatura)	-40 °C – 80 °C
Precisión (Temperatura)	± 0,5 °C
Rango de medición (Humedad)	0 % – 100 %
Precisión (Humedad)	± 2 %
Señal de salida	Señal digital a través de un solo bus
Voltaje de alimentación	3,3 V_{DC} – 6 V_{DC}
Dimensiones	14 x 18 x 5,5 mm

Nota. Adaptado de Technical Specification. (p. 2), Aosong Electronicd (2020).

Con respecto a los pinout del sensor, en la Tabla 4 se muestran el orden los pines de izquierda a derecha.

Tabla 4

Pines del Sensor DHT22

Pin	**Función**
1	V_{DC}
2	DATA
3	NC (No conexión)
4	GND

2.2.3.2. Sensor de humedad del suelo

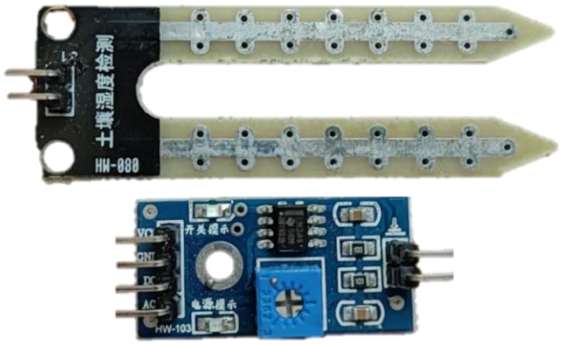
El sensor de humedad del suelo mide la humedad por medio de la variación de la conductividad del suelo en el que se encuentra enterrado. Está conformado por dos electrodos resistivos (HW-080) y un módulo convertidor (HW-103). El convertidor proporciona una señal digital y una señal analógica. La señal digital indica “1” o “0”, es decir, o hay humedad o no hay humedad. La señal analógica da un voltaje de 0 a 5 voltios dependiendo de la cantidad de humedad que detecte el electrodo.

El funcionamiento del sensor es muy sencillo, la medición de la resistencia del suelo y la humedad del suelo depende de los electrodos, en un terreno muy húmedo, habrá un cortocircuito entre los dos electrodos, es decir, una resistencia muy baja o una resistencia igual a cero, y en un terreno muy seco, la resistencia será muy alta. En conclusión, el sensor tiene dos sondas y medimos la resistencia entre ellas, entre menor sea la resistencia mayor la humedad.

La Figura 2 muestra los dos electrodos resistivos y el módulo convertidor.

Figura 2

Sensor de Humedad del Suelo



Con respecto a los pinout del convertidor, la Tabla 5 muestra el orden de los pines de arriba hacia abajo con respecto a la imagen anterior.

Tabla 5

Pines del Convertidor HW-103

Pin	Función
1	V_{DC}
2	GND
3	DO
4	AO

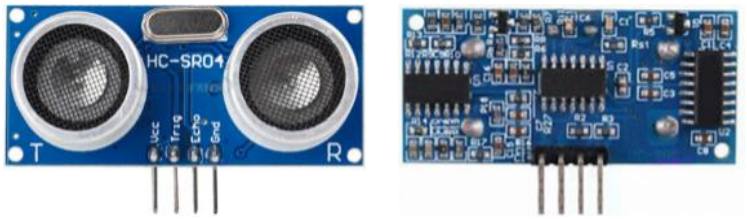
2.2.3.3. Sensor ultrasónico (HC-SR04)

El sensor HC-SR04 es un sensor de distancia de bajo costo. Funciona emitiendo un sonido ultrasónico a través de uno de sus transductores (Trig), el segundo transductor (Echo) se encarga de captar el sonido cuando rebote en algún objeto. La distancia es proporcional al tiempo que transcurre desde que se envía el sonido hasta que se recibe.

La Figura 3 muestra la parte anterior y posterior del sensor ultrasónico.

Figura 3

Sensor Ultrasónico (HC-SR04)



En la Tabla 6 se encuentran algunos de los parámetros técnicos más relevantes del sensor ultrasónico.

Tabla 6

Parámetros Técnicos del Sensor HC-SR04

Parámetros técnicos	
Rango mínimo de medición	2 cm
Rango máximo de medición	400 cm
Angulo de medición	< 15°
Frecuencia de trabajo	40 Hz
Voltaje de alimentación	5 V_{DC}
Corriente	15 mA
Dimensiones	45 x 20 x 15 mm

Nota. Adaptado de Electric Parameter. (p. 1), Elec Freaks (2013).

Con respecto a los pinout del sensor, en la Tabla 7 se muestran el orden los pines de izquierda a derecha.

Tabla 7

Pines del Sensor HC-SR04

Pin	Función
1	V_{DC}
2	Trig
3	Echo
4	GND

2.2.4. Actuador

Alciatore (2008) La mayoría de los sistemas electrónicos o mecatrónicos implican algún tipo de acción o movimiento. Esta acción o movimiento puede aplicarse a cualquier cosa. Los actuadores son los dispositivos utilizador para producir la acción o movimiento.

> Los actuadores constituyen la interfaz entre el procesamiento de la señal (procesamiento de la información) y el proceso (mecánico). Transforman las señales que aportan la información de ajuste, de baja potencia, en señales potentes correspondientes a la energía necesaria para intervenir en el proceso. (Guarella et al., 2011, p. 18)

2.2.5. Microcontrolador

Un microcontrolador es un circuito integrado programable, consta de una unidad central de proceso (CPU), memorias (RAM o ROM) y periféricos de entrada y salida. Está diseñado para realizar el único programa que puede ser guardado en su memoria, los sensores y los actuadores se conectan a los periféricos de entrada y salida para poder controlar el dispositivo.

2.2.6. ESP32

ESP32 es un Sistema de un Chip (System on a Chip – SoC) desarrollado por la compañía Espressif Systems. Esta serie de chips está orientada a una amplia variedad de aplicaciones móviles, dispositivos electrónicos e Internet de las Cosas (Internet of Things – IoT).

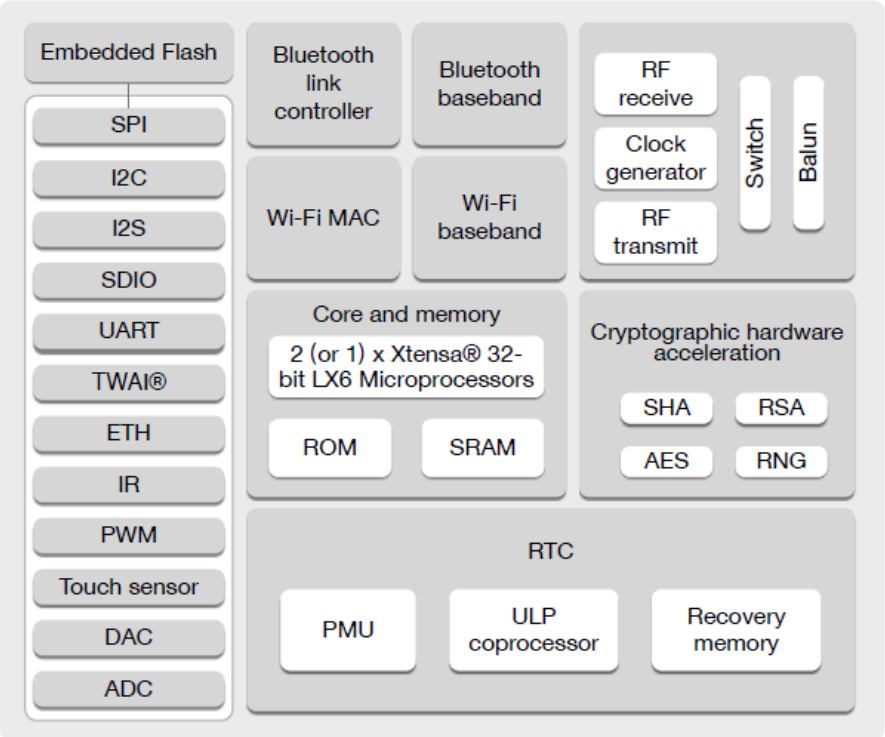
ESP32 es un chip que combina Wi-Fi y Bluetooth de 2,4 GHz, está diseñado con tecnología TSMC de ultra baja potencia de 40 nm. El objetivo de su diseño es

lograr la mejor potencia y rendimiento de RF, mostrando robustez, versatilidad y confiabilidad en diversas aplicaciones y escenarios de potencia.

En la Figura 4 se muestran todos los bloques funcionales del microcontrolador ESP32.

Figura 4

Estructura del ESP32



Nota. Adaptado de Fuctional Block Diagram (p. 12), Espressif Systems (2021).

En la Tabla 8 se muestran las características técnicas principales del microcontrolador ESP32.

Tabla 8

Características Técnicas del ESP32

Características	ESP32
Procesador	Tensilica Xtensa LX6 – 32 bits Dual-Core a 160 MHz (hasta 240 MHz)
SRAM	520 kB
Memoria Flash	Hasta 16 MB
ROM	448
Alimentación	2,3 V a 3,6 V

Continuación de la Tabla 8.

Características	ESP32
Rango de temperaturas	-40 ºC a 125 ºC
Consumo promedio	80 mA, 225 max
Consumo modo Deep Sleep	2,5 µA
WiFi	802.11 b / g / n / e / i (hasta +20 dBm) WEP, WPA
Encriptación	AES, SHA, RSA, ECC
Ethernet	10 / 100 Mbps MAC
Bluetooth	v4,2 BR / EDR y BLE
UART	3
I2C	2
SPI	4
GPIO	34
PWM	16
ADC	2 (12 bits) con pre-amplificador de bajo ruido, hasta 60 dB
DAC	2 (8 bits)
1-wire	por software
I2S	2
CAN bus	1 (2,0)
Sensor de temperatura	Si
Sensor efecto HALL	Si
Sensor capacitivo	10
IR	Si
Temporizador	4 (64 bits)

Nota. Adaptado de Fuctional Description (p. 22), Espressif Systems (2021).

2.2.7. Protocolo ESP-NOW

El protocolo ESP-NOW es una tecnología de comunicación rápida y sin conexión que ofrece transmisión de paquetes cortos. ESP-NOW es ideal para luces inteligentes, dispositivos de control remoto, sensores y otras aplicaciones.

ESP-NOW brinda las siguientes características:

- Comunicación de unidifusión cifrada y no cifrada.
- Dispositivos homólogos encriptados y no encriptados.
- Puede transportar hasta una carga de 250 bytes.
- Función de envío de devolución de llamada que se puede configurar para informar a la capa de aplicación del éxito o fallo de la transmisión.

ESP-NOW tiene las siguientes limitaciones:

- Pares cifrados limitados. El modo de estación admite 10 pares cifrados como máximo; 6 como máximo en modo SoftAP o SoftAP + Station;
- Se admiten varios pares no cifrados, sin embargo, su número total debe ser inferior a 20, incluidos los pares cifrados.

A continuación, en la Tabla 9, se muestra un cuadro con las funciones principales del protocolo ESP-NOW.

Tabla 9

Funciones Útiles del Protocolo ESP-NOW

Nombre de la función	Descripción de la función
esp_now_init()	Inicializa ESP-NOW. Debe inicializar WiFi antes de inicializar ESP-NOW.
esp_now_deinit()	Desinicializa ESP-NOW. Cuando se llama a esta función se eliminará toda la información de los dispositivos emparejados.
esp_now_add_peer()	Esta función sirve para emparejar un dispositivo y pasar como argumento la dirección MAC del par.
esp_now_send()	Envía datos con ESP-NOW.
esp_now_register_send_cb	Registra una función de devolución de llamada que se activa al enviar datos. Cuando se envía un mensaje, se llama a una función: esta función devuelve si la entrega se realizó correctamente o no.
esp_now_register_rcv_cb	Registre una función de devolución de llamada de recepción que se activa al recibir datos. Cuando se reciben datos a través de ESP-NOW, se llama a una función.

2.2.7.1. Tipos de Comunicación

ESP-NOW puede tener dos tipos, comunicación unidireccional o comunicación bidireccional.

2.2.7.1.1. Comunicación unidireccional

Este tipo de comunicación se emplea para enviar datos de una placa hacia otra, es excelente para enviar lecturas de sensores o comandos de encendido o apagado para controlar un actuador. La Figura 5 muestra lo anteriormente descrito.

Figura 5

Comunicación Unidireccional

Existen dos formas de configurar la comunicación unidireccional, de uno a mucho (Un Maestro – Múltiples Esclavos o One Master – Multiple Slaves) o de muchos a uno (Un Esclavo – Múltiples Maestros o One Slave – Multiple Masters).

a. Un maestro, múltiples esclavos

En esta configuración una placa ESP32 envía los mismos o diversos comandos a diferentes placas ESP32. Esta configuración es ideal para construir algo como un control remoto. Puede tener varias placas ESP32 que están controladas por una placa ESP32 principal. La Figura 6 ilustra lo descrito anteriormente.

Figura 6

Un Maestro, Múltiples Esclavos

Nota. Adaptado de ESP-NOW one-way communication protocol (p. 3), Pasic et al., 2020.

b. Un esclavo, múltiples maestros

Esta configuración es ideal si desea recopilar datos de varios nodos de sensores en una placa ESP32. Esto se puede configurar en un servidor web para mostrar datos de todas las demás placas. La Figura 7 muestra lo descrito anteriormente.

Figura 7

Un Esclavo, Múltiples Maestros

Nota. Adaptado de ESP-NOW one-way communication protocol (p. 3), Pasic et al., 2020.

2.2.7.1.2. Comunicación bidireccional

Con este tipo de comunicación cada placa puede ser un emisor y un receptor al mismo tiempo, en consecuencia, se establece una comunicación bidireccional entre las placas ESP32. La Figura 8 ilustra los descrito anteriormente.

Figura 8

Comunicación Bidireccional

Se puede agregar más placas, todas comunicándose entre sí, con este tipo de configuración se puede obtener algo parecido a una red. La Figura 9 muestra lo anteriormente descrito.

Figura 9

Red de Comunicación Bidireccional

Nota. Adaptado de ESP-NOW two-way communication protocol (p. 3), Pasic et al., 2020.

2.2.8. Internet de las Cosas

El Internet de las cosas es la interconexión digital de objetos cotidianos e Internet, esta conexión permite el intercambio automático de información con otros dispositivos o centros de control, todo esto sin intervención humana.

2.2.8.1. Elementos en IoT

Mora y Rosas (2019) afirman que los elementos en IoT son:

- Sensores, actuadores y periféricos.
- Hardware: Los microcontroladores son dispositivos que cumplen la función de realizar la interacción.
- Conectividad: Es el medio de comunicación con el cual los equipos se comunicarán con la red, ya sea inalámbrica o por medio de un cable. Ejemplos: GPRS, Bluetooth BLE, WiFi, LPWAN, 6LoWPAN, Sigfox, Ethernet, etc.
- Protocolos de Comunicación: Son lenguajes de comunicación que utiliza el software para comunicarse con el hardware. Ejemplos: API REST, HTTP, MQTT, etc.
- Plataformas IoT: Están diseñadas para tratar los datos recogidos por los sensores y después almacenarlos. Ejemplos: Adafruit IO, Arduino IoT Cloud, Thingspeak, Thinger, Ubidots, etc.
- Servicios: Son los servicios que ofrecen las plataformas IoT para facilitar el trabajo. Ejemplos: Notificaciones, diagramas, alertas, etc.

2.2.8.2. Arquitectura

Definir una única arquitectura de IoT es complejo porque, aunque se han propuesto diferentes arquitecturas de IoT, no existe un consenso general, de diferentes estudios han sugerido diferentes modelos. La arquitectura recopila más o menos detalles de diferentes aspectos de IoT.

a. Arquitectura de tres niveles

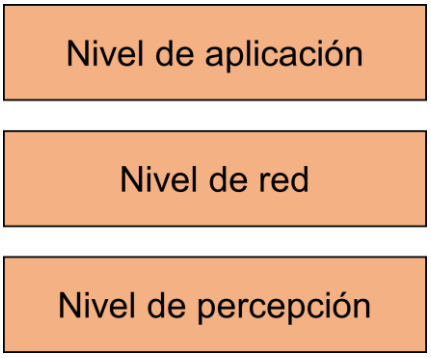
La arquitectura de tres niveles está conformada por el nivel de precepción, el nivel de red y el nivel de aplicación:

> El nivel de percepción es el nivel físico, donde los sensores recogen información del entorno. El nivel de red es el responsable de conectar los sensores y servidores entre sí para transmitir y procesar los datos recogidos por los sensores. El nivel de aplicación es donde IoT puede ser desplegado en diferentes áreas de aplicación. (González, 2017, p. 5)

En la Figura 10 podemos apreciar la jerarquía de la arquitectura de tres de niveles.

Figura 10

Diagrama de la Arquitectura de Tres Niveles



Nota. Adaptado de Arquitectura de 3 niveles (p. 5), de González, A. (2017).

2.2.8.3. Plataformas IoT

La plataforma IoT es un software que conecta sensores, actuadores, dispositivos y equipos industriales en un entorno digital, creando una red para comunicarse y crear información valiosa.

> Al igual que el término IoT, una plataforma IoT es un concepto muy amplio. Puede tratarse de simples plataformas que sirven para almacenar datos y ofrecen interfaces estándares al usuario, hasta sistemas más completos que permiten el uso de herramientas para hacer predicciones, analíticas o para crear interfaces más complejas.
>
> Una plataforma IoT debe de permitir recoger los datos enviados desde los diferentes dispositivos conectados. Por otra parte, debe de facilitar la creación de aplicaciones, tanto móviles como para otros dispositivos, que visualicen de manera clara los datos recibidos de los dispositivos IoT conectados a la plataforma, además de los datos sobre los que se ha trabajado. (Martinez, 2017, p. 6)

A continuación, se nombran algunas plataformas IoT que se encuentran en el mercado:

- Adafruit IO
- Amazon Web Services IoT
- Arduino IoT Cloud
- Azure IoT Hub
- B-scada
- Carriots
- Google cloud platform
- IBM Bluemix
- Machina.IO
- myDevices Cayenne
- Nimbits
- Oracle Internet of Things Cloud Service
- Plot.ly
- Samsung Artik
- Thingspeak
- Ubidots
- Watson IoT
- Xively

La Tabla 10 muestra los lenguajes de programación, protocolos, ventajas y desventajas de las plataformas IoT más conocidas y utilizadas por los usuarios.

Tabla 10

Cuadro Comparativo de Plataformas IoT

Plataforma	SDK/Lenguajes soportados	Protocolos Soportados	Ventajas	Desventajas
AWS IoT	C JavaScript Java Python iOS Android C++	MQTT WebSocket HTTP	Plataforma líder, con casi infinitas posibilidades mediante sus diferentes servicios. Sus tarifas son muy baratas.	Se necesita mucha investigación para vincular diferentes servicios. La plataforma está en constante evolución y se necesita mucha práctica para mantenerse al día.

Continuación de la Tabla 10.

Plataforma	SDK/Lenguajes soportados	Protocolos Soportados	Ventajas	Desventajas
Azure IoT	C Python Node.js Java .NET	MQTT AMQP HTTP	Plataforma bastante completa, con multitud de servicios y una arquitectura por capas muy bien definida. Posee de un sistema de interacción con el dispositivo muy completo.	Los mensajes entre servicios no están incluidos en el precio base. Los precios están agrupados en 4 categorías inflexibles y hay que tener mucho cuidado al usarlos para que no se etiqueten como sobredimensionados.
Oracle IoT	C JavaScript Java SE Windows Mbed Android iOS	MQTT HTTP	Facilita la conexión a dispositivos a través de clientes y puertas de enlace encargadas de gestionar todo el proceso. Herramientas de análisis de datos muy completas.	Plataforma de reciente creación con un número limitado de servicios en comparación con sus competidores directos. Categorías de precios muy difíciles de entender.
Watson IoT	C C# C++ Java Python Node.js	HTTP MQTT	Una plataforma de análisis de datos integral para el aprendizaje automático y la minería de datos.	Falta de una solución interna en otras plataformas para mostrar datos. La cantidad de servicios disponibles es menor que la cantidad de competidores.
Xively	Android iOS	MQTT WebSocke HTTP	Es muy fácil de administrar e iniciar. Gestión completa de dispositivos, gracias a grupos por ubicación o función.	La plataforma proporciona solo servicios básicos para la recopilación, el análisis y la presentación de datos. No tienen una lista de precios con cotizaciones, debe comunicarse con su departamento de ventas.
Samsung Artik	C C++ Node.js	HTTP WebSockets MQTT CoAP	Es fácil conectar las placas de desarrollo del fabricante a la plataforma para facilitar el desarrollo de soluciones de IoT. Muchas integraciones con herramientas de terceros se configuran automáticamente.	Intenta ocultar tanto el proceso de configuración como las opciones, quizás acercándose más a desarrollos con necesidades específicas. Esta es una plataforma mucho más cara y menos flexible que sus competidores.

Continuación de la Tabla 10.

Plataforma	SDK/Lenguajes soportados	Protocolos Soportados	Ventajas	Desventajas
Carriots	Arduino Groovy	MQTT HTTP	Proporciona la capacidad de crear lógica de aplicación básica utilizando un motor de visualización con bloques. Utiliza un lenguaje funcional y moderno como Groovy para desarrollar aplicaciones.	Su sistema de reglas y alertas es mucho más sencillo que el que ofrecen otras soluciones del mercado. No es posible elegir cómo se almacena la información dentro de la plataforma una vez seleccionado el formato de datos, por lo que es necesario exportar los datos a una base de datos externa para realizar el tipo de operación.
Adafruit.IO	Arduino Python Node.js Ruby	MQTT HTTP	Contiene una gran sección de instrucciones. Proporciona la mayoría de los controladores necesarios para los sensores. Es fácil crear cuadros de mando que presenten información de forma gráfica.	La información no se puede exportar fuera de la plataforma. No hay integración con terceros. Las alertas o reglas no se pueden configurar para que funcionen en función de los valores recopilados por el sensor.
Ubidots	Python Java C PHP Node.js Ruby Arduino	MQTT HTTP	Dispone de varias librerías, tanto para placas de desarrollo específicas como para lenguajes de programación. Sus herramientas de gráficos de datos son muy poderosas.	No está permitido administrar dispositivos en grupos, ya que cada dispositivo tiene su propio panel de control. Carece de ofertas flexibles para usar, ya que requiere negociar un plan personalizado con su equipo de ventas.
myDevices Cayenne	Arduino Raspberry Pi LoRa	MQTT HTTP	La plataforma tiene la capacidad de gestionar directamente los conectores de la placa de desarrollo, para evitar tener que programar la interacción del sistema con sensores. La documentación es muy completa con muchos ejemplos.	Soporte limitado para placas de desarrollo y sensores, ya que deben ser compatibles con el software proporcionado en la plataforma. Se basa principalmente en el uso de una aplicación móvil para configurar y administrar dispositivos IoT.

Continuación de la Tabla 10.

Plataforma	SDK/Lenguajes soportados	Protocolos Soportados	Ventajas	Desventajas
Macchina.io	JavaScript	MQTT ModBus COAP	La plataforma es de código abierto. Tiene un sistema de alerta con reglas.	Los usos de la base de datos SQLite son muy limitados para la mayoría de los escenarios industriales. El motor de renderizado es muy restrictivo y requiere que los usuarios implementen componentes de JavaScript.
ThingSpeak	MATLAB Arduino Raspberry Pi	MQTT HTTP	La plataforma es de código abierto. Gran recomendación de integración por parte del desarrollador. Permite la agrupación de dispositivos IoT por función o ubicación a través de sus canales. fácil de usar.	Su sistema de alerta se limita a la integración con Twitter. Para otro tipo de alertas, es necesario utilizar un servicio de terceros. Su sistema de reglas no es intuitivo y difícil de configurar. La instalación de una plataforma en una nube privada es compleja y propensa a fallar. Pocos documentos.

Nota. Adaptado de Comparativa Plataformas IoT (p. 43), de Martinez, R. (2017).

2.2.8.4. Aplicación IoT

El concepto de combinar sensores, actuadores, computadoras y redes para monitorear y controlar distintos dispositivos se remonta a décadas. Sin embargo, la combinación reciente de diferentes tendencias en el mercado tecnológico está acercando a Internet de las Cosas a convertirse en una realidad.

> Desde este punto de vista, la IoT representa la convergencia de una variedad de tendencias en las áreas de la computación y la conectividad que se vienen dando desde hace muchas décadas. En la actualidad, una amplia gama de sectores de la industria —entre ellos el sector automotriz, la salud, la manufactura, la electrónica de consumo y para el hogar— están analizando el potencial de incorporar la tecnología de la IoT en sus productos, servicios y operaciones. (Rose et al., 2015, p. 15)

En la Tabla 11 se aprecian algunos de los muchos entornos en los cuales se pueden desarrollar aplicaciones IoT.

Tabla 11

Entornos para Aplicaciones IoT

Entorno	Descripción	Ejemplo
Cuerpo Humano	Dispositivos que se adjuntan o insertan en el cuerpo humano.	El dispositivo monitorea y mantiene la salud y el bienestar del cuerpo, trata enfermedades, mejora el estado físico y aumenta la productividad.
Hogar	Edificios de vivienda.	Sistemas de seguridad y dispositivos de control para el hogar.
Puntos de venta	Espacios comerciales.	Tiendas, bancos, restaurantes, estadios, cualquier lugar donde los clientes consuman y compren; sistema de pago autoservicio, ofertas de compra directa, optimización de inventarios.
Oficinas	Espacio donde trabajan los trabajadores de cierto sector.	Gestión de energía y seguridad en edificios de oficinas; productividad mejorada, incluso para trabajadores móviles.
Fabricas	Entornos de producción estandarizados.	Lugares con procedimientos de trabajo frecuentes, como hospitales y granjas; eficiencia operativa, optimización de equipos y utilización de inventarios.
Obras	Entornos de producción a medida.	Minería, petróleo y gas y construcción; eficiencia operativa, mantenimiento predictivo, salud y seguridad.
Vehículos	Sistemas dentro de vehículos en movimiento.	Vehículos, incluidos automóviles, camiones, barcos, aviones y trenes; Mantenimiento basado en la condición actual del vehículo, diseño basado en el uso, análisis de preventa.
Ciudades	Entornos urbanos	Espacio público e infraestructura urbana; sistema de control de tráfico adaptativo, contador inteligente, monitorización medioambiental, gestión de recursos.
Exteriores	Entre zonas urbanas y no urbanas.	Los usos al aire libre incluyen ferrocarriles, vehículos autónomos (fuera de ubicaciones urbanas) y aeronáutica; Enrutamiento en tiempo real, navegación conectada y seguimiento de envíos.

Nota. Adaptado de “Entornos” para aplicaciones IoT (p. 16), de Rose et al. (2015).

2.2.9. Protocolo NTP

El Protocolo de Tiempo de Red o Network Time Protocol (NTP) se utiliza para sincronizar relojes en sistemas informáticos. Para sincronizar el reloj del servidor con una precisión de nanosegundos, el protocolo NTP emplea el estándar Tiempo Universal Coordinado o Universal Time Coordinated (UTC).

Características principales del protocolo de tiempo de red:

- NTP emplea un reloj de referencia que actúa como un punto fijo para todas las sincronizaciones, todos los relojes están sincronizados con este reloj de referencia.
- NTP es un protocolo tolerante que busca automáticamente las mejores fuentes de tiempo para realizar la sincronización. Para reducir los errores acumulativos, puede seleccionar y combinar varias líneas. Cuando es posible, NTP detecta fuentes de tiempo que proporcionan valores sesgados temporales o permanentes y los descarta.
- Es un protocolo muy escalable: se puede encontrar un número de reloj de referencia en cada red de sincronización.
- Es muy exacto. Dado que puede elegir la mejor fuente de sincronización, teóricamente debería tener una precisión de un rango de nanosegundos de 2^{-32} segundos (0,233 nanosegundos).
- Puede solucionar problemas temporales de conexión a la red.

2.2.10. Base de datos

La base de datos se denomina al conjunto de datos organizados de una manera que permite un acceso rápido a varios tipos de información.

En términos de informática, una base de datos: “Es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos normalmente predefinida para su posterior organización o consulta” (Marqués, 2011, p. 2).

2.2.10.1. Base de datos relacional – SQL

Se llama base de datos relacional al tipo de base de datos que almacena y proporciona acceso a puntos de datos que se relacionan. Las bases de datos relacionales se basan en el modelo relacional, que es una forma fácil y sencilla de representar los datos en tablas.

> Se denomina base de datos relacional al conjunto de tablas correctamente identificadas en las que se puede acceder a la información de las mismas. Mediante un lenguaje de consultas estructuradas (Structure Query Language), se realiza la interacción entre un software y una base de datos relacional. El concepto básico de una base de datos relacional es el de "Relación". (Palma y Velasquez, 2019, p.11)

2.2.10.2. Base de datos no relacional – NoSQL

Las bases de datos no relacional están diseñadas para varios modelos de acceso a datos, incluidas las aplicaciones de latencia baja. La base de datos de búsqueda no relacional está diseñada para realizar análisis de datos semiestructurados.

> Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Mientras que las tradicionales bases de datos relacionales basan su funcionamiento en tablas, joins y transacciones. Las bases de datos NoSQL no imponen una estructura de datos en forma de tablas y relaciones entre ellas, sino que proveen un esquema mucho más flexible. (Martín et al., 2013, p. 166)

2.2.11. Firebase

Firebase es una plataforma creada por Google. Su principal función es desarrollar y promover la creación de aplicaciones de alta calidad para aumentar la base de usuarios. Firebase está disponible en diferentes plataformas como iOS, Android y Web. Contiene varias funciones, por lo que cualquier desarrollador puede combinar y ajustar la plataforma para satisfacer sus necesidades.

Firebase Realtime Database es una base de datos no relacional (NoSQL) alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado.

2.2.12. Aplicación móvil

Una aplicación móvil, también conocida como app móvil, es una aplicación diseñada para ejecutarse en un teléfono inteligente o una tablet. "El término móvil se refiere a poder acceder a los datos, las aplicaciones y los dispositivos desde cualquier lugar." (Enriquez y Casas, 2013, p. 26).

2.2.13. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones de Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece más funciones para aumentan su productividad al desarrollar aplicaciones de Android, como:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y con muchas funciones.
- Un entorno unificado donde puedes desarrollar aplicaciones para varias versiones de los dispositivos Android.
- Aplicación de cambios para modificar el código y sus recursos a la aplicación en ejecución sin reiniciarla.
- Integración con plantillas de código y GitHub para ayudarte a compilar funciones de aplicaciones comunes y también importar código de muestra.
- Gran variedad de herramientas de prueba y marcos de trabajo.
- Herramientas de Lint para identificar compatibilidad de versiones, usabilidad, problemas de rendimiento, entre otros.
- Compatibilidad con NDK y C++.
- Soporte de integración con Google Cloud Platform, lo que facilita la integración con App Engine y Google Cloud Messaging.

2.2.14. Atenuación por vegetación en un radioenlace

En algunos casos, la atenuación causada por la vegetación puede ser significativa tanto para los sistemas terrenales como sistemas tierra-espacio.

Según Rec. ITU-R P.833-10

> Para un trayecto radioeléctrico terrenal, uno de cuyos terminales está situado en un bosque o en una zona similar de vegetación extensa, la pérdida adicional debida a la vegetación puede describirse en base a dos parámetros:
>
> - El índice de atenuación específica (dB/m) debida fundamentalmente a la dispersión de energía fuera del trayecto radioeléctrico, que se mediría en un trayecto muy corto.
> - La atenuación adicional total máxima debida a la vegetación en un trayecto radioeléctrico (dB) limitada por el efecto de otros mecanismos,

entre ellos, la propagación de ondas de superficie por encima del medio vegetal y la dispersión dentro del mismo. (p. 1)

La atenuación excesiva (A_{ev}) debida a la presencia de la vegetación se halla utilizando la siguiente formula:

$$A_{ev} = A_m \left(1 - e^{\frac{-d*\gamma}{A_m}}\right) \quad (1)$$

Siendo:

d: Longitud del trayecto dentro de la zona boscosa (m)

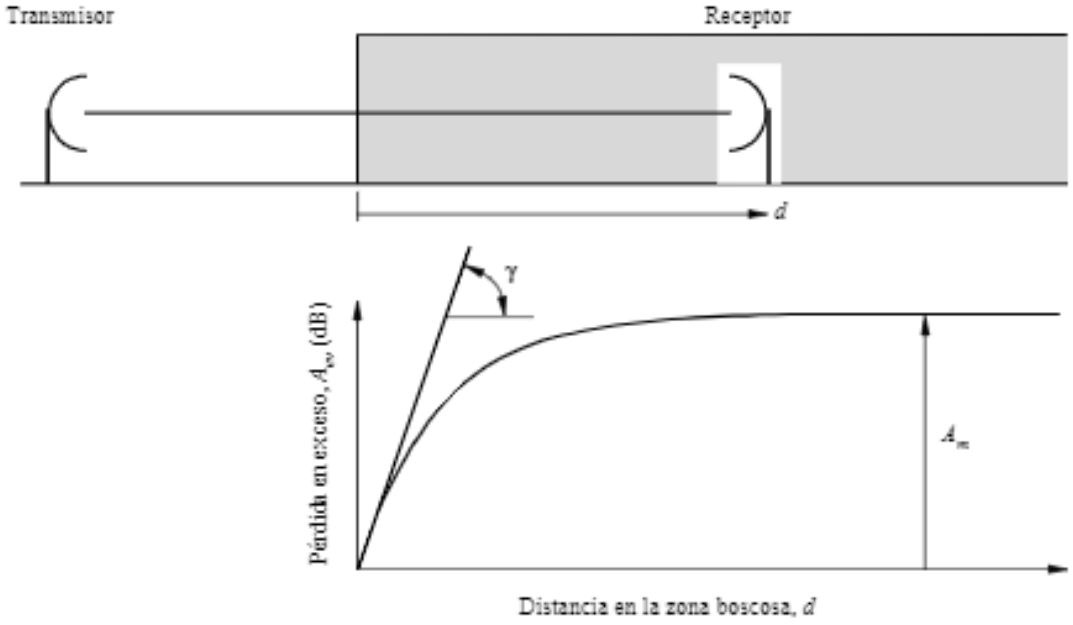
γ: Atenuación específica para trayectos en vegetación muy cortos (dB/m)

A_m: Atenuación máxima cuando el terminal está dentro de una zona de vegetación de un tipo y profundidad específicos (dB)

La Figura 11 muestra la representación gráfica de cómo obtener el valor de las variables para poder hallar la atenuación excesiva.

Figura 11

Trayecto Radioeléctrico Representativo en Zona Boscosa



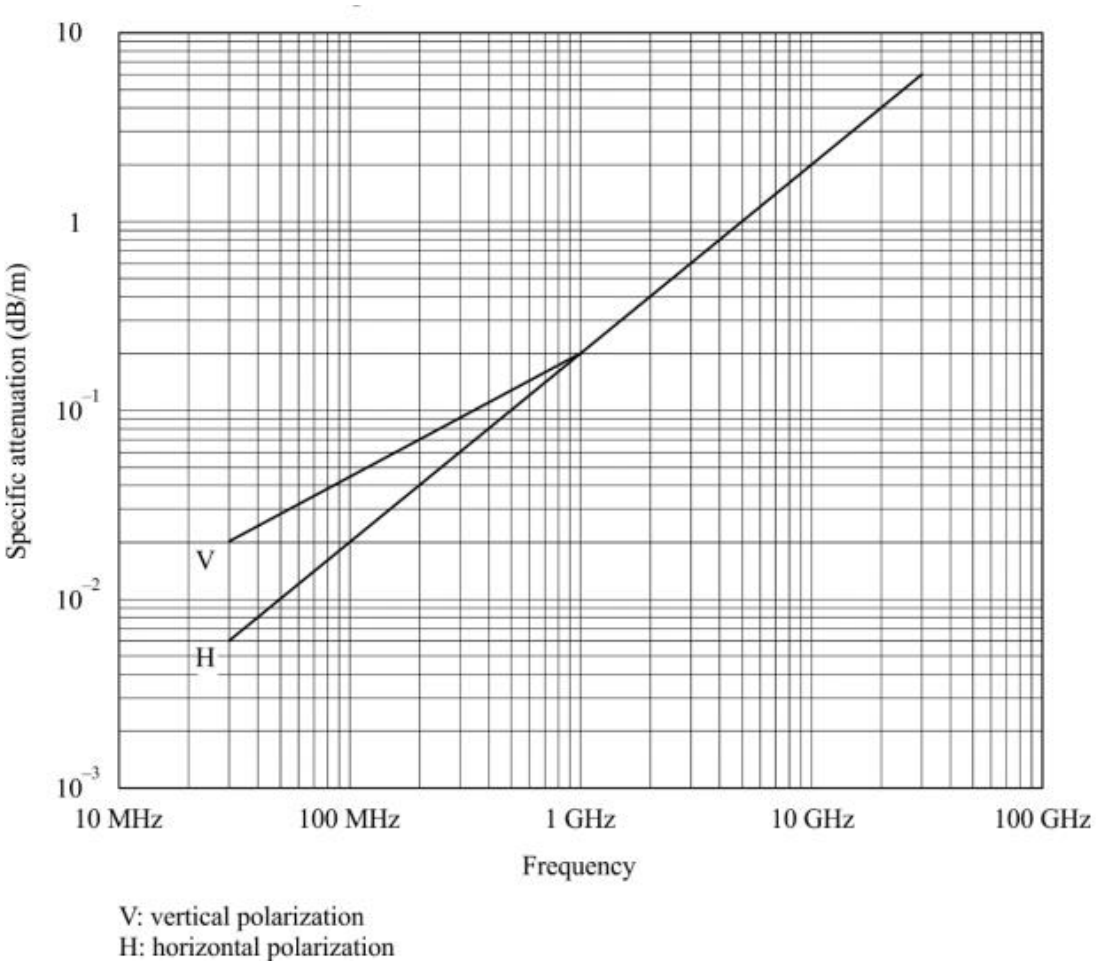
Nota. Adaptado de Representative radio path in woodlan (p. 2), Rec. ITU-R P.833-10 (2021).

Es importante señalar que la definición de atenuación excesiva incluye el exceso debido a todos los demás mecanismos, y no solo a la perdida en el espacio abierto. Por lo tanto, si la forma del trayecto radioeléctrico de la figura anterior fuese de tal forma que el despejamiento total de Fresnel del terreno no existiera, la atenuación excesiva sería el exceso de atenuación producido tanto en el espacio abierto como por la pérdida de difracción. Asimismo, si la frecuencia es lo suficientemente alta como para la absorción gaseosa resultara significativa, A_{ev} también seria el exceso producido por la absorción gaseosa.

El valor de la atenuación específica para trayectos de vegetación muy cortos, γ, depende del tipo y densidad de la vegetación. En la Figura 12 se muestran valores aproximados en función de la frecuencia.

Figura 12

Atenuación Especifica en Zona Boscosa



Nota. Adaptado de Specific attenuation due to Woodland (p. 3), Rec. ITU-R P.833-10 (2021).

La Figura 12, mostrada anteriormente, muestra valores de atenuación específica obtenidos de varias mediciones en el rango de frecuencias aproximadamente de 30 MHz a 30 GHz en zona boscosa. Por debajo de 1 GHz, las señales polarizadas verticalmente tienden a tener una atenuación superior a las señales polarizadas horizontalmente, esto se debe a la dispersión causada por los troncos de los árboles.

El valor de la atenuación máxima, A_m (dB), limitada por la dispersión de la onda de superficie, depende del tipo y la densidad de la vegetación, también del diagrama de radiación de la antena del terminal que se encuentra dentro de la vegetación y de la distancia en vertical entre la antena y el punto más alto de la vegetación.

La dependencia de la frecuencia de A_m (dB):

$$A_m = A_1 f^{\alpha} \tag{2}$$

Donde f es la frecuencia en MHz, esta se ha obtenido realizando distintos experimentos:

> Mediciones en la gama de frecuencias 900 – 1 800 MHz realizadas en un parque con árboles tropicales en Río de Janeiro (Brasil) con una altura media de los árboles de 15 m. Se obtuvieron valores de $A_1 = 0{,}18$ dB y $\alpha = 0{,}752$. La altura de la antena receptora era de 2,4 m.
>
> Mediciones en la gama de frecuencias 900 – 2 200 MHz realizadas en un bosque cerca de Mulhouse (Francia) en trayectos de longitudes diferentes, desde unos pocos cientos de metros hasta 6 km con diversos tipos de árboles de altura media de 15 m. Se obtuvieron valores de $A_1 = 1{,}15$ dB y $\alpha = 0{,}43$. La antena receptora en el bosque era un monopolo de $\lambda/4$ montada sobre un vehículo a una altura de 1,6 m y la antena transmisora era un dipolo de $\lambda/2$ con una altura de 25 m. La desviación típica de las mediciones fue de 8,7 dB. Se observaron unas variaciones según la estación del año de 2 dB a 900 MHz y de 8,5 dB a 2 200 MHz. (Rec. ITU-R P.833-10, 2021, p. 4)

2.3. Definición de términos

2.3.1. Sistema de monitoreo

Un sistema de monitoreo son instrumentos de gestión, responsables de proveer la información sobre el desempeño para alimentar la toma de decisión (Arenas et al., 2021).

2.3.2. Sistema de control

Un Sistema de Control está definido como un conjunto de componentes que pueden regular su propia conducta o la de otro sistema con el fin de lograr un funcionamiento predeterminado, de modo que reduzcan las probabilidades de fallos y se obtengan los resultados buscados (Tumerio, 2016).

2.3.3. Internet de las Cosas (IoT)

El término “Internet de las cosas” (en inglés, Internet of Things, abreviado IoT), hace referencia a todos aquellos objetos o dispositivos cotidianos que se encuentran conectados a Internet y que cuentan con algún tipo de inteligencia (Hernández, 2019).

2.3.4. Aplicación Móvil

Se entiende por apps aquellas aplicaciones de software que funcionan en teléfonos móviles o tablets y que son distribuidos a través de servicios o tiendas como la "iTunes Store" (iOS), "Google Play" (Android); estas pueden ser generadas por desarrolladores de tecnologías móviles o por individuos u organizaciones (Van Velsen et al., 2013).

CAPÍTULO III: MARCO METODOLÓGICO

3.1. Tipo y Nivel de la investigación

El tipo de la investigación es tecnológica porque se utilizan los conocimientos de ingeniería para aplicarlos en el beneficio de la sociedad.

El nivel de investigación es explicativo porque permite determinar los efectos de la implementación de una solución tecnológica.

3.2. Operacionalización de variables

La Tabla 12 muestra la operacionalización de variables.

Tabla 12

Operacionalización de Variables

Variable	Tipo de Variable	Definición Conceptual	Dimensiones	Indicadores	Instrumentos
Sistema IoT para el monitoreo y control.	Variable independiente	El sistema de monitoreo mide la variable en tiempo real y manda una señal al sistema de control para realizar una determinada acción si esta fuera necesaria.	Sistema de monitoreo	Aplicación móvil	ESP32
			Sistema de control	Actuadores	Ventilador Calefactor Sistema de riego
Monitoreo y control de los parámetros del cultivo de lechugas en un invernadero	Variable dependiente	Parámetros de cultivo: • Temperatura: Magnitud física que refleja la cantidad de calor del invernadero. • Humedad relativa: Cantidad de agua en el aire del invernadero en forma de vapor. • Humedad del suelo: Cantidad de agua por volumen de tierra que hay en un cultivo.	Temperatura	Temperatura (°C)	DHT22
			Humedad	Humedad relativa (%)	DHT22
				Humedad del suelo (%)	HW-080

3.3. Descripción del sistema IoT

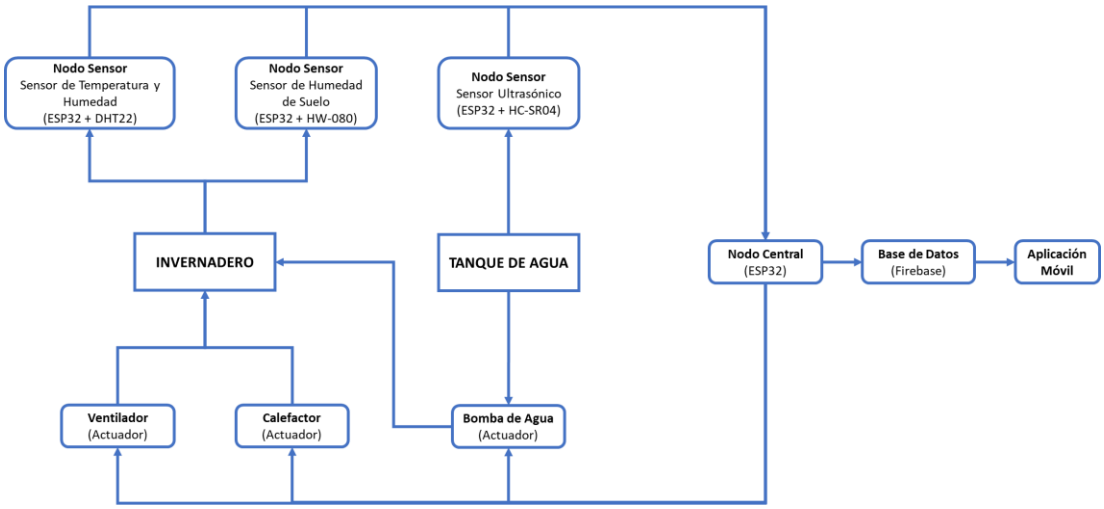
El sistema está conformado por el nodo central, los tres nodos sensor, la base de datos, la aplicación móvil y los actuadores.

Los nodos sensor recolectan la temperatura, la humedad relativa, la humedad del suelo del invernadero y el nivel del tanque de agua. Todos estos datos son enviados al nodo central, este se encarga de enviarlos a la base de datos y activar o desactivar los actuadores cuando sea necesario. La aplicación móvil adquiere la información de la base de datos y la muestra en su interfaz, también permite encender o apagar los actuadores cuando el usuario lo crea conveniente, es importante resaltar que la visualización de datos y control de los actuadores se pueden realizar desde cualquier parte del mundo, solo es necesario el acceso a internet.

En la Figura 13 que se muestra a continuación, se puede observar el diagrama de bloques del Sistema IoT.

Figura 13

Diagrama de Bloques del Sistema IoT

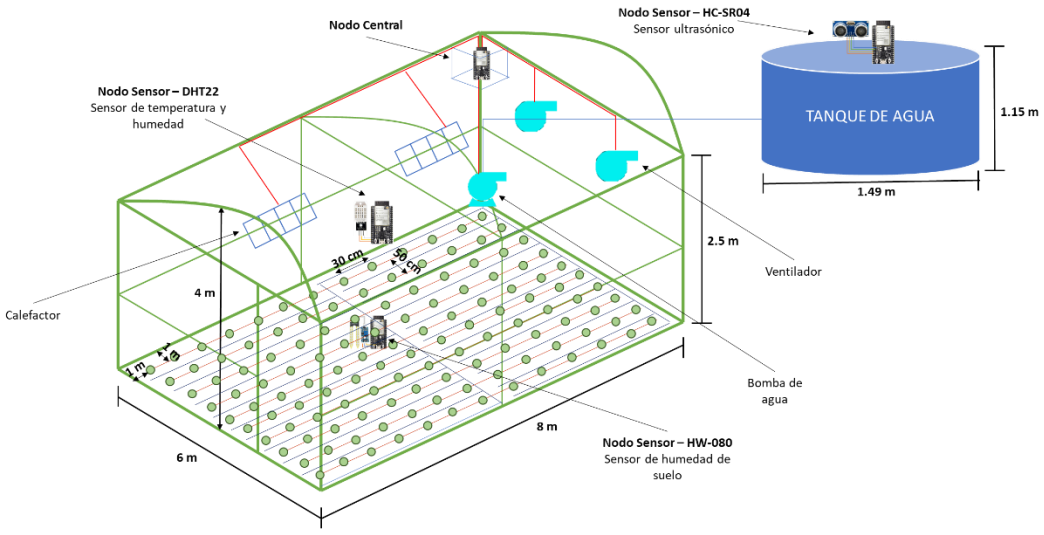


3.4. Disposición del sistema IoT en el invernadero

La Figura 14 muestra la disposición de los nodos y de los actuadores en el invernadero.

Figura 14

Distribución del Sistema IoT en el Invernadero



El invernadero tiene un tamaño de 4 metros de alto, 6 metros de ancho y 8 metros de largo. El tanque de agua tiene un diámetro de 1,49 metros y una altura de 1,15 metros, tiene una capacidad máxima de alrededor de 2 000 litros de agua.

Los nodos de sensor deben estar distribuidos estratégicamente en el invernadero de tal forma que puedan cumplir sus funciones sin ninguna complicación. Todos los nodos están formados por un microcontrolador ESP32 y su componente o componentes adicionales. Existe tres nodos sensor, el Nodo Sensor – DHT22 está compuesto por el sensor de temperatura y humedad y está ubicado a una altura media en el centro del invernadero, el Nodo Sensor – HW-080 con el sensor de humedad del suelo está situado en medio de las plantaciones y el Nodo Sensor – HC-SR04 está conformado por el sensor ultrasónico y se encuentra en la parte superior del tanque de agua. El nodo central se halla en una caja protectora en la parte superior de la esquina de atrás del invernadero, a este nodo van conectado los actuadores.

Los ventiladores están localizados en la parte trasera del invernadero, los calefactores en la parte superior del invernadero y la bomba de agua en la parte inferior de la esquina de atrás del invernadero.

3.5. Proceso de control del sistema IoT

El sistema IoT está diseñado para regular la temperatura, la humedad relativa y la humedad del suelo del cultivo de lechugas. El sistema detecta estos parámetros gracias a los sensores y está programado para activar o desactivar el actuador adecuado cuando se requiera.

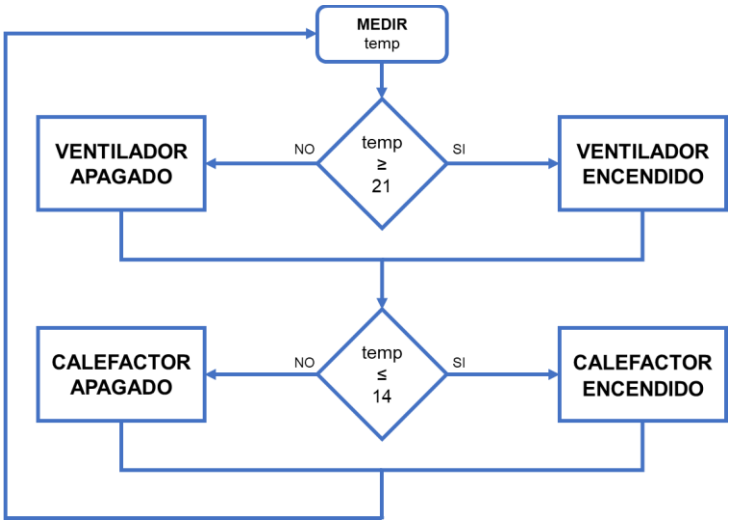
3.5.1. Sistema de ventilación y calefacción

El sistema de ventilación y calefacción controla la temperatura y humedad relativa del invernadero. El sistema diferencia entre las condiciones climáticas del horario diurno y horario nocturno. El rango óptimo de temperatura en el horario diurno es de 15 °C a 20 °C, mientras que en el horario nocturno es de 10 °C a 15 °C. El rango óptimo de la humedad relativa es el mismo para ambos horarios y esta entre 60 % y 70 %.

La Figura 15 muestra, a través de un diagrama de flujo, el funcionamiento del sistema de ventilación y calefacción para el control de temperatura en el día, si la temperatura esta entre 20 °C y 15 °C, el ventilador y el calefactor se mantendrán apagador, pero si la temperatura es mayor o igual a 21 °C, el ventilador se activará y si la temperatura es menor o igual a 14 °C, el calefactor se encenderá.

Figura 15

Diagrama de Flujo del Sistema de Ventilación y Calefacción para el Control de Temperatura – Horario Diurno

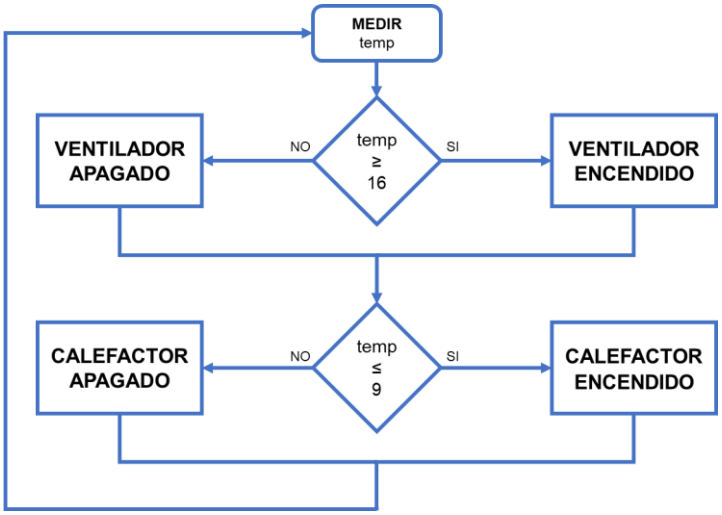


La Figura 16 señala, a través de un diagrama de flujo, el funcionamiento del sistema de ventilación y calefacción para controlar la temperatura durante la noche,

si la temperatura esta entre 15 °C y 10 °C, los dos actuadores se mantendrán apagados, pero si la temperatura es mayor o igual a 16 °C, el ventilador se encenderá y si la temperatura es menor o igual a 9 °C, el calefactor se activará.

Figura 16

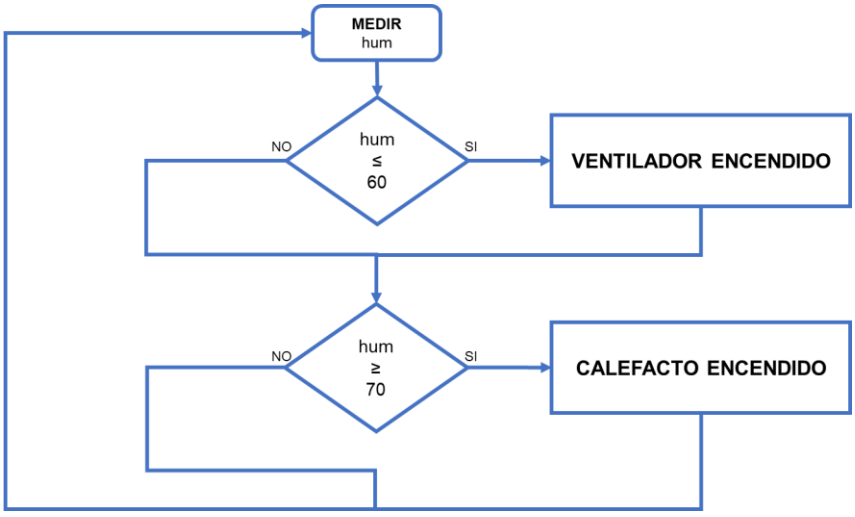
Diagrama de Flujo del Sistema de Ventilación y Calefacción para el Control de Temperatura – Horario Nocturno



La Figura 17, mediante un diagrama de flujo, explica el funcionamiento del sistema de ventilación y calefacción para el control del humedad relativa, el ventilador se encenderá si la humedad relativa es menor o igual a 60 % y el calefactor se activará si la humedad relativa es mayor o igual a 70 %.

Figura 17

Diagrama de Flujo del Sistema de Ventilación y Calefacción para el Control de Humedad Relativa



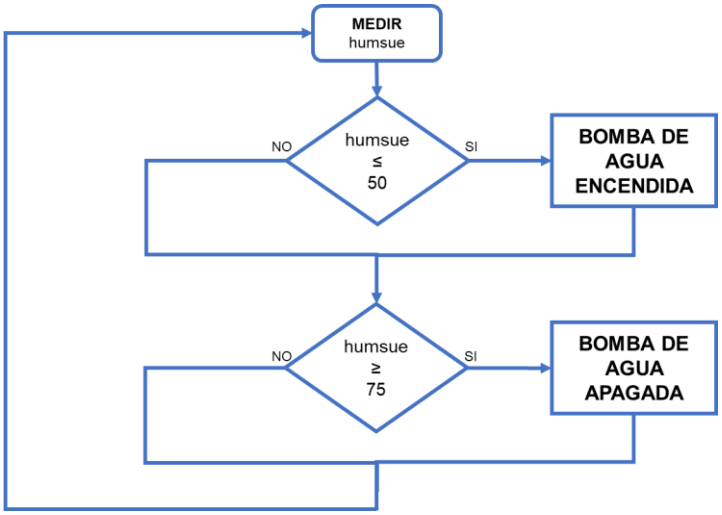
3.5.2. Sistema de riego

El sistema de riego controla la humedad del suelo, el sistema activara la bomba de agua siempre que no tengamos la humedad del suelo suficiente para cumplir con el rango óptimo.

Teniendo en cuenta esto, la Figura 18 explica el funcionamiento del sistema de riego, el sistema de riego activara la bomba de agua siempre que haya humedad del suelo menor o igual a 50 % y la desactivara cuando la humedad del suelo sea mayor o igual a 75 %.

Figura 18

Diagrama de Flujo del Sistema de Riego



3.6. Configuración y programación del sistema IoT

En esta parte de la tesis explicaremos paso a paso la configuración y programación del sistema IoT, para realizar el sistema se necesitó de dos softwares, Arduino IDE y Android Studio, y una base de datos, Firebase.

Arduino IDE es un software multiplataformas, se usa para crear y subir programas en microcontroladores compatibles con Arduino. Arduino IDE admite los lenguajes C y C++ utilizando reglas especiales de estructuración de códigos. Se empleo este software para la programación de todos los nodos.

Android Studio es el IDE oficial para el desarrollo de aplicaciones para Android. Este software admite Kotlin, Java y C++ como lenguajes de programación. Se utilizó Android Studio para el desarrollo de la aplicación móvil.

Firebase es una plataforma ubicada en la nube que utiliza un conjunto de herramientas para crear y sincronizar aplicaciones web o móviles. Firebase nos proporcionó una base de datos en tiempo real que nos permitió que la información de la aplicación móvil sea sincronizada y almacenada en la nube de Firebase.

Los pasos que se realizaron para la configuración y programación del sistema IoT son los siguientes:

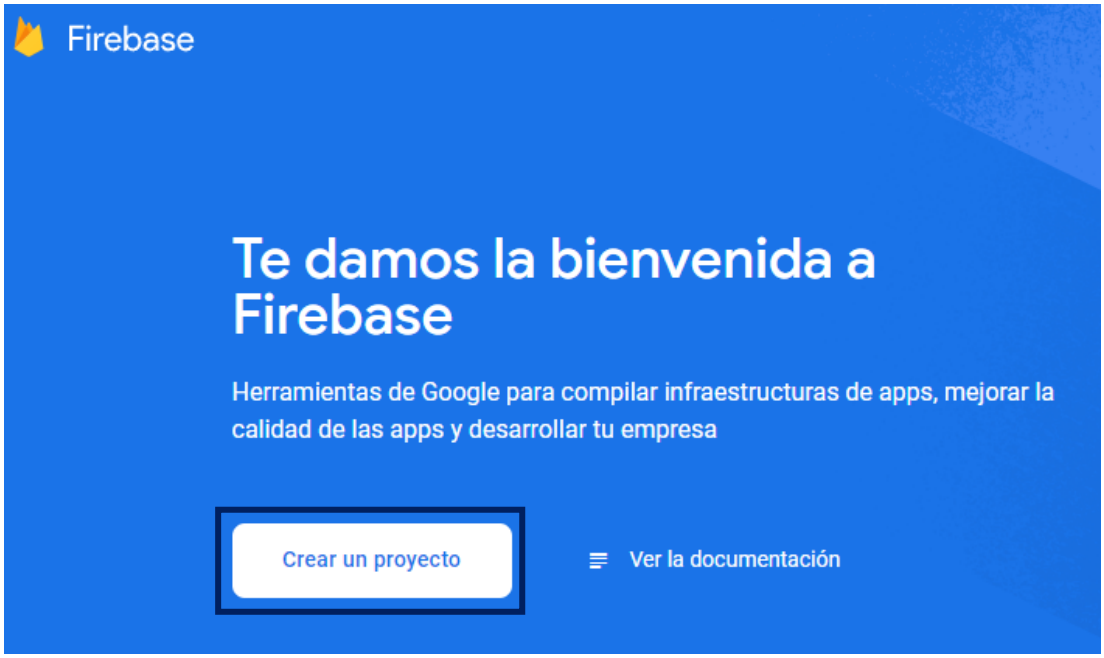
1. Crear un proyecto en Firebase
2. Crear un proyecto en Android Studio
3. Enlazar los dos proyectos
4. Configurar la base de datos en Firebase
5. Configurar la interfaz de la aplicación móvil en Android Studio
6. Programar el funcionamiento de los nodos en Arduino IDE

3.6.1. Creación del proyecto en Firebase

El primer paso se muestra en la Figura 19, fue el ingreso a la página oficial de Firebase, se registró y creó un nuevo proyecto.

Figura 19

Creación del Proyecto en Firebase



El segundo paso que se realizó se puede observar en la Figura 20, este fue nombrar el proyecto y aceptar las condiciones de Firebase.

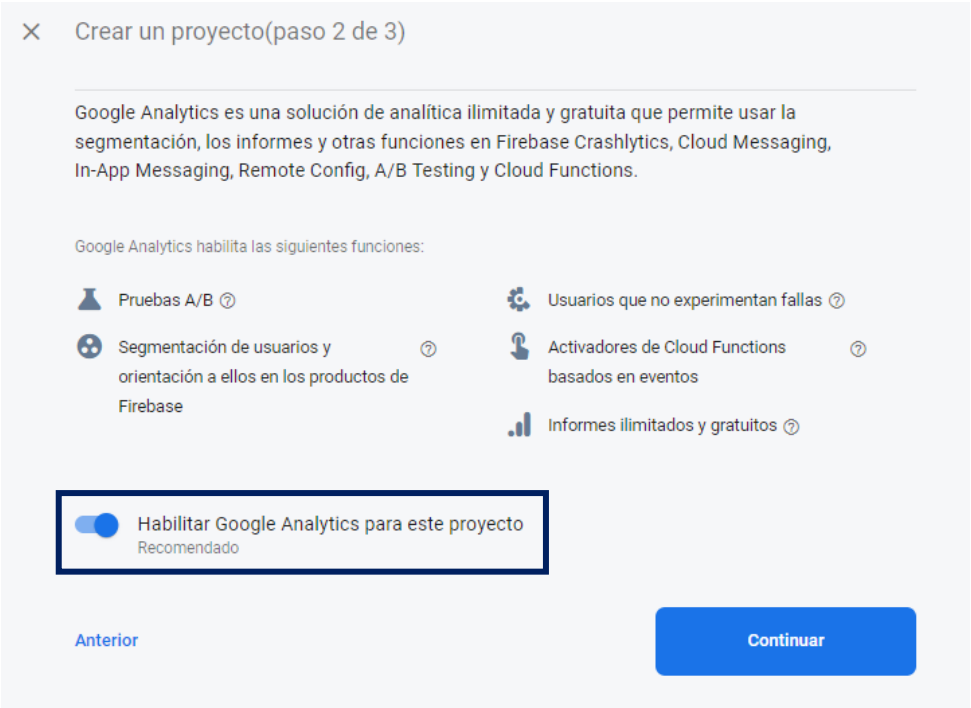
Figura 20

Nombre del Proyecto y Condiciones de Firebase

El paso número tres se puede visualizar en la Figura 21, fue habilitar Google Analytics para el proyecto. Google Analytics es una herramienta de analítica web, ofrece información agrupada del tráfico de datos que llegan a la aplicación.

Figura 21

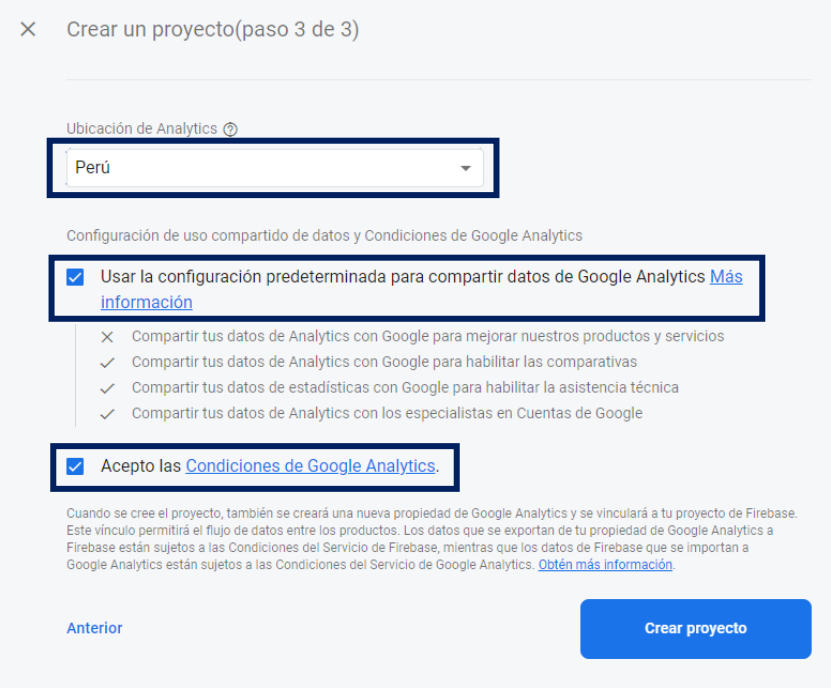
Habilitación de Google Analytics



El cuarto y último paso de la creación del proyecto en Firebase fue la configuración de Google Analytics, se eligió la región en la que nos encontrábamos, se aceptó la configuración predeterminada para compartir datos en Google Analytics y también las condiciones de Google Analytics. Todo este procedimiento se puede percibir en la Figura 22.

Figura 22

Configuración de Google Analytics



Después de esperar unos minutos, una vez que creado el proyecto, apareció la ventana de inicio del proyecto. La Figura 23 muestra lo descrito anteriormente.

Figura 23

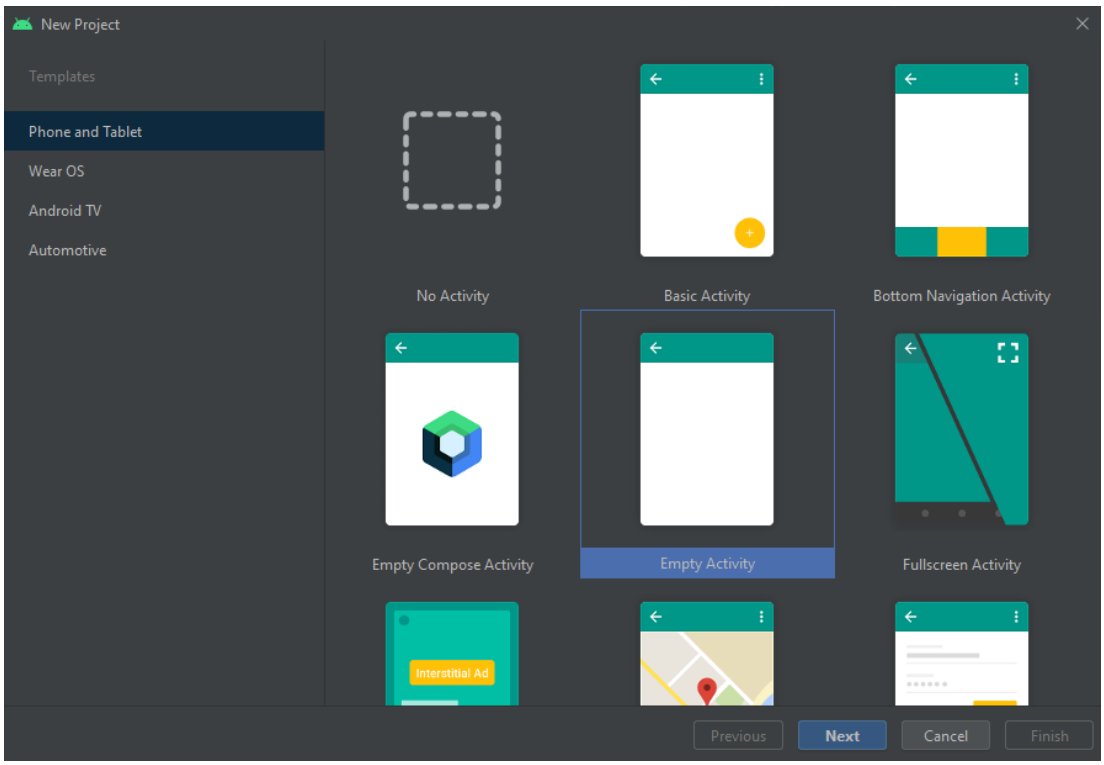
Proyecto Creado – Firebase

3.6.2. Creación del proyecto en Android Studio

Esta parte solo está enfocada en la creación del proyecto. Se abrió el software Android Studio y se seleccionó New Project/Phone and Tablet/Empty Activity. Se escogió Empty Activity porque es la opción que ofrece un mayor espacio de trabajo. La Figura 24 representa el primer paso de la creación del proyecto en Android Studio.

Figura 24

Creación del Proyecto en Android Studio



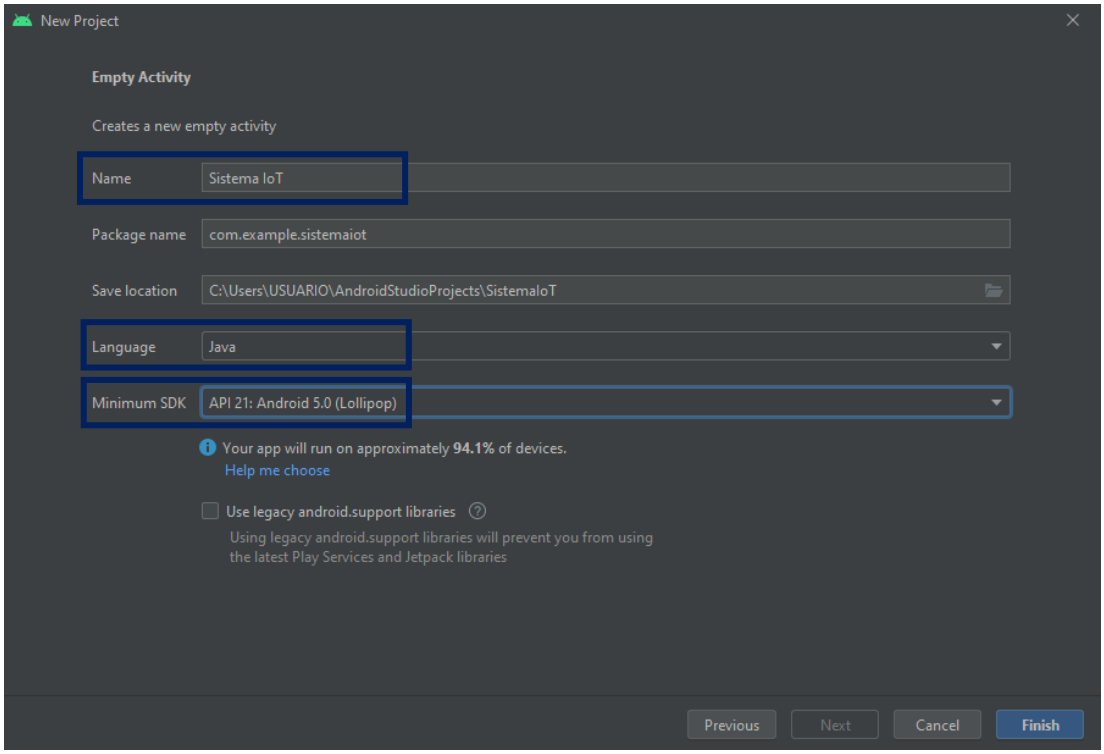
Se continúo nombrando el proyecto, se escogió el lenguaje de programación Java y el API 21 en SDK.

SDK es un conjunto de herramientas y bibliotecas que se requieren para desarrollar aplicaciones Android. API (Interfaz de programación de aplicaciones) 21 tiene el 94.1% de compatibilidad con teléfonos inteligentes con sistema operativo Android, esto se debe a que puede trabajar con la versión Android 5.0 en adelante.

En la Figura 25 se observa resaltado el nombre del proyecto, el lenguaje de programación y el mínimo SDK que se escogió.

Figura 25

Nombre del Proyecto, Lenguaje de Programacion y SDK



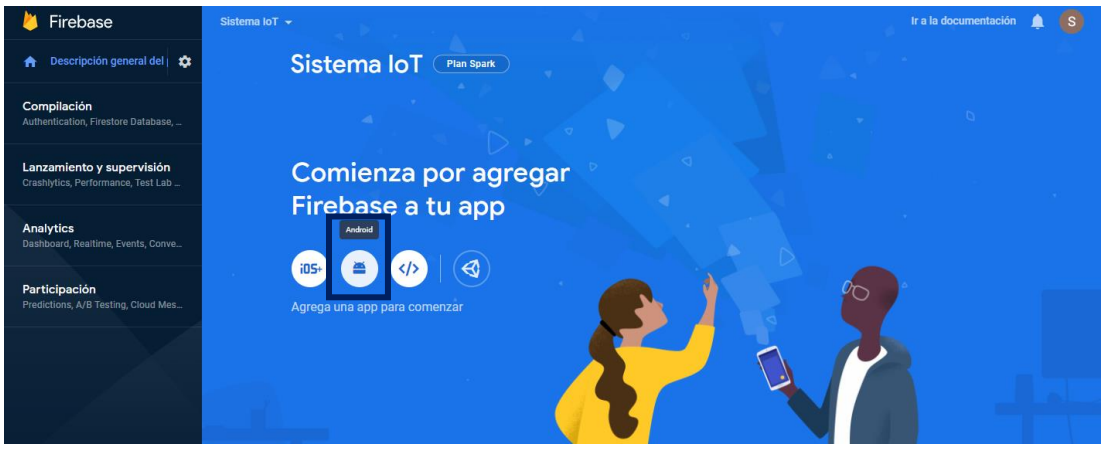
3.6.3. Enlazamiento de proyectos

Para poder trabajar con los dos proyectos simultáneamente, primero se tuvieron que enlazar, esto permitió que la aplicación móvil pueda acceder y hacer cambios en la base de datos.

El primer paso que se realizó fue en página oficial de Firebase, se ingresó a la opción Android para agregar Firebase a la aplicación móvil, tal como se muestra en la Figura 26.

Figura 26

Opción Android en Firebase

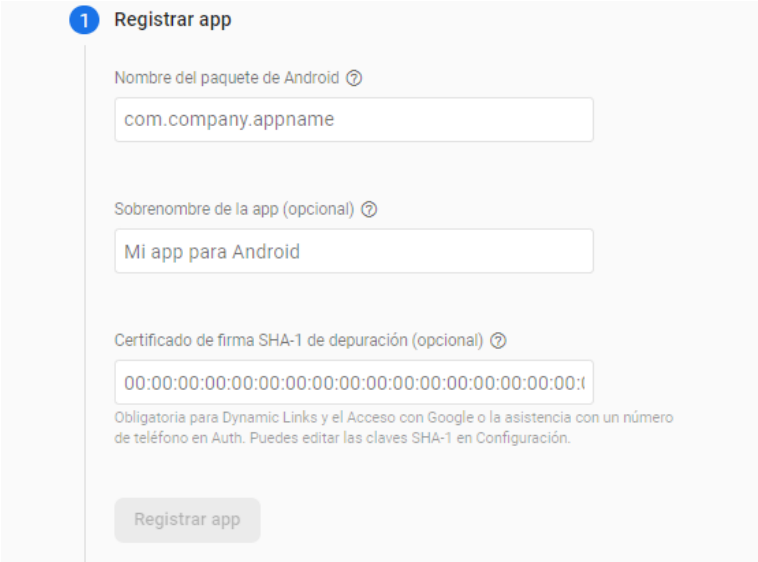


Apareció la ventana que se ve en la Figura 27, en ella se completó el Nombre del paquete de Android y Certificado de firma SHA-1 de depuración. Esta información se obtuvo del proyecto que se creó en Android Studio.

SHA-1 es una clave la cual es necesaria para poder acceder a API de los servicios de Google.

Figura 27

Requisitos para agregar Firebase a la Aplicación Móvil

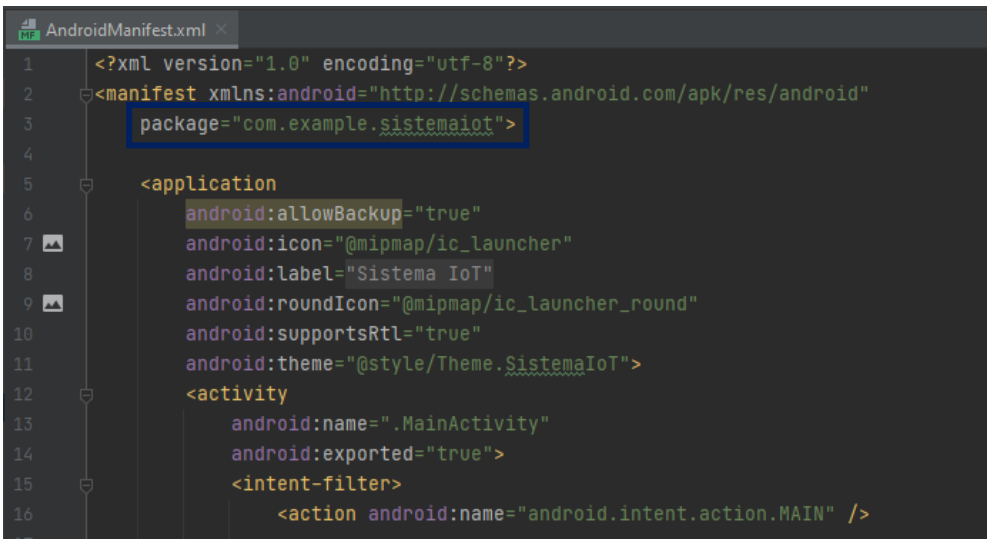


En Android Studio se obtuvo el Nombre del paquete de Android o package ingresando a Project/app/scr/main/AndroidManifest.xml.

En la Figura 28 se puede visualizar como se logró conseguir el Nombre del paquete de Android.

Figura 28

Nombre del Paquete de Android



Luego se ingresó a la opción Gradle/Sistema IoT/Run Configurations/SistemaIoT[signingReport] para conseguir el Certificado de firma SHA-1 de depuración. La Figura 29 muestra lo anteriormente descrito.

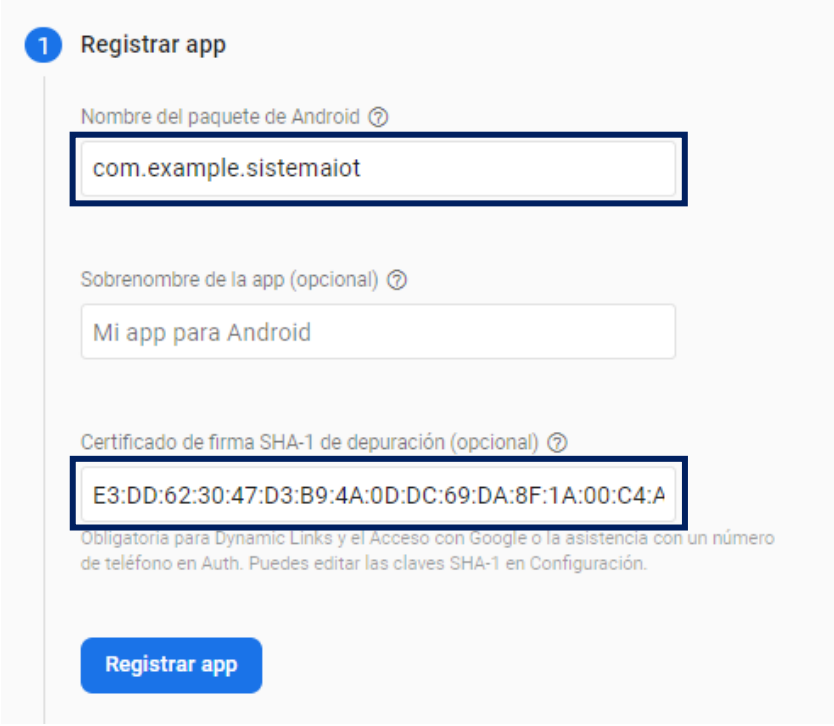
Figura 29

Certificado de Firma SHA-1 de Depuración

A continuación, se completó los datos y se registró la aplicación móvil, tal como se muestra en la Figura 30. El sobrenombre de la aplicación es opcional.

Figura 30

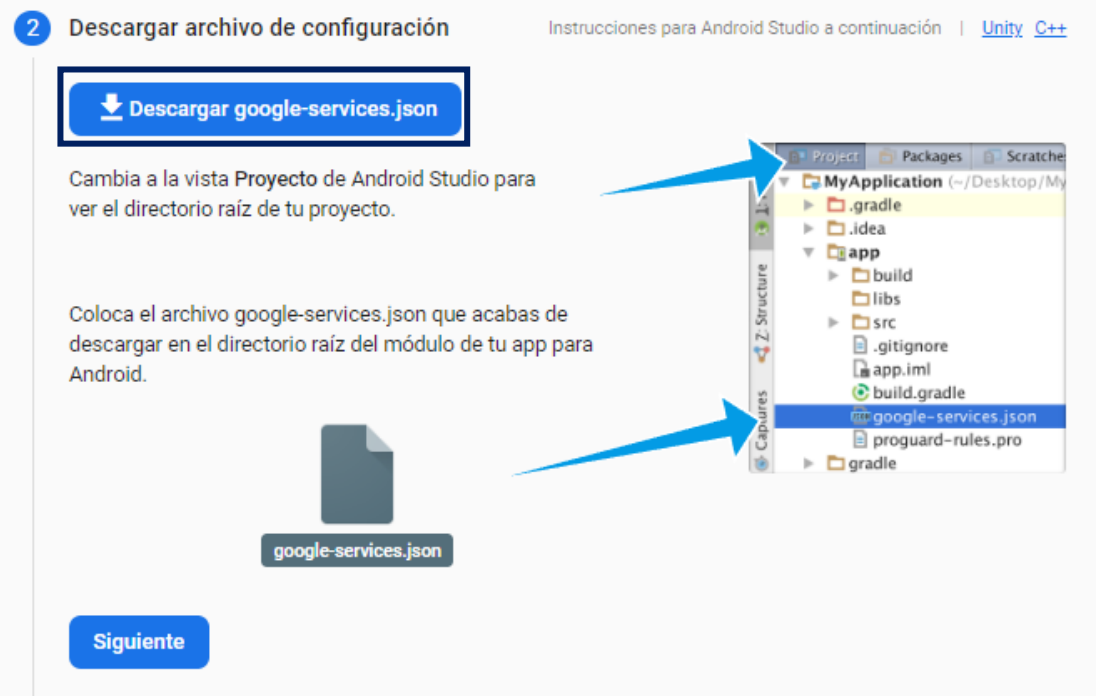
Completamiento de Datos para Agregar Firebase a la Aplicación Móvil



Posteriormente se descargó el archivo google-services.json, como se ve en la Figura 31. Este archivo contiene las credenciales de desarrollo y configuración, estas son necesarias para la verificación durante la conexión con Firebase.

Figura 31

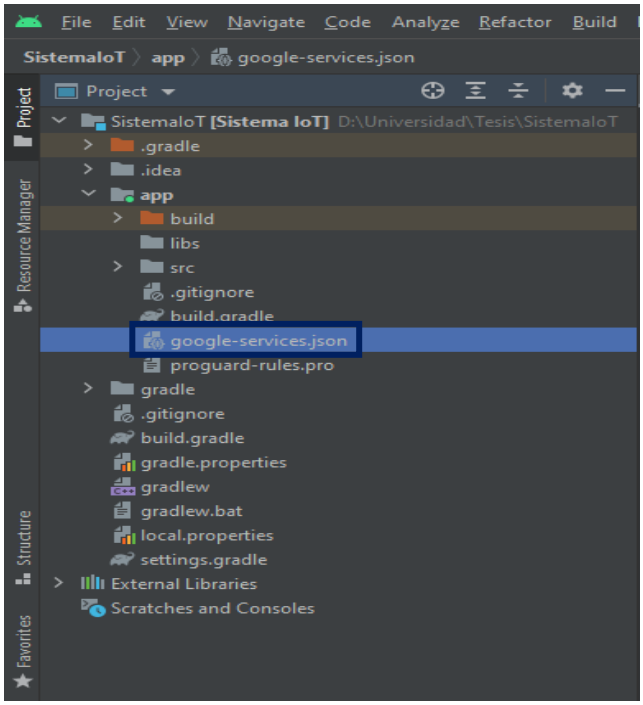
Descarga de Archivo de Configuración



En el proyecto de Android Studio se ingresó a Project/SistemaIoT[Sistema IoT]/app/scr y se agregó el archivo descargado en Firebase, tal como se muestra en la Figura 32.

Figura 32

Agregación del Archivo de Configuración de Firebase a Android Studio



El último paso fue agregar el SDK de Firebase a nuestro proyecto en Android Studio.

Los tres SDK que se requirieron para poder finalizar con la sincronización de los proyectos se visualizan en la Figura 33 y en la Figura 34.

Figura 33

SDK de Firebase – 1

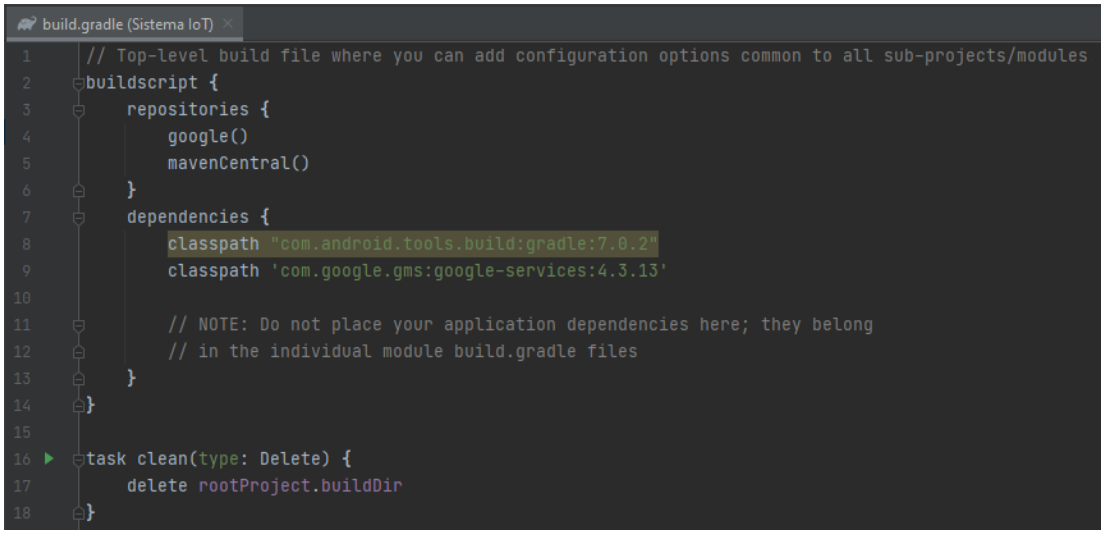
Figura 34

SDK de Firebase – 2

El primer SDK que se agregó fue classpath y se muestra en la Figura 35. Para agregar esta herramienta se ingresó a Android/Gradle Scripts/build.gradle (Project: Sistema_IoT).

Figura 35

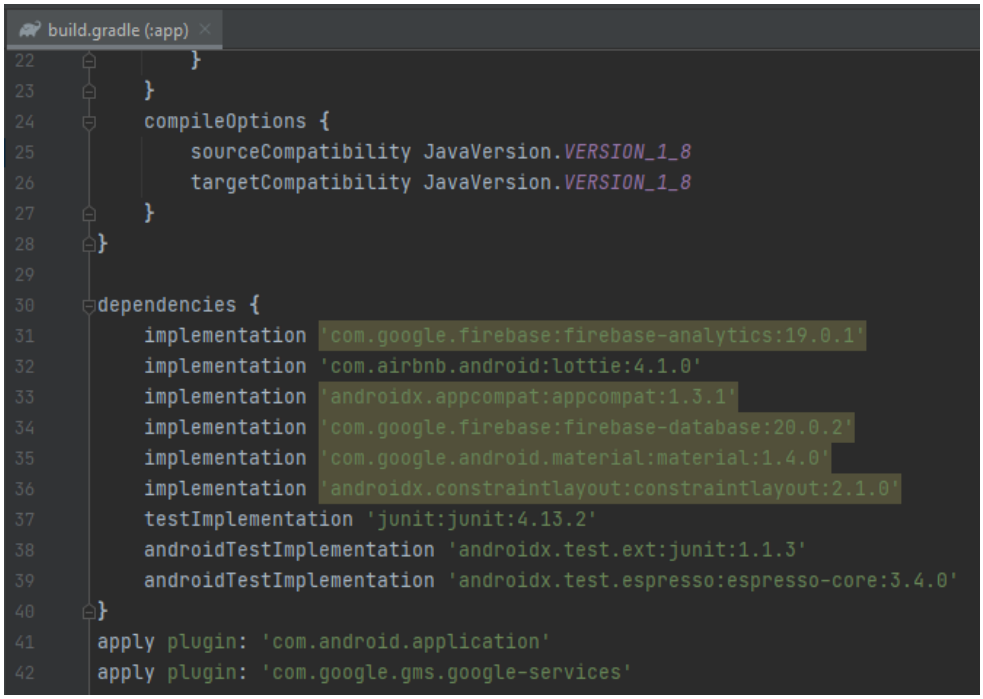
Agregación del SDK de Firebase a Android Studio – 1



Los dos SDK restantes que se agregaron fueron apply plugin e implementation, tal como se observa en la Figura 36. Para agregar esta herramienta se ingresó a Android/Gradle Scripts/build.gradle (Module: Sistema_IoT.app).

Figura 36

Agregación del SDK de Firebase a Android Studio – 2



3.6.4. Configuración de la base de datos en Firebase

Ahora crearemos la base de datos para poder almacenar las lecturas que recolecten los diferentes sensores.

Se ingresó a Realtime Database para crear la base de datos. Después de crear la base de datos, se configuró las reglas. Es importante que ambas estén en true porque estas reglas permiten que la aplicación móvil pueda hacer lectura y escritura en la base de datos.

La Figura 37 muestra la configuración final de las reglas de la base de datos.

Figura 37

Configuración de Reglas en la Base de Datos

El último paso fue la creación de las variables en la base de datos, esto se realizó en Realtime Database/Datos. Se creó cuatro variables, una por cada actuador y una para las lecturas que ingresaran. También se creó subvariables, una por cada actuador y una por cada lectura que ingresara.

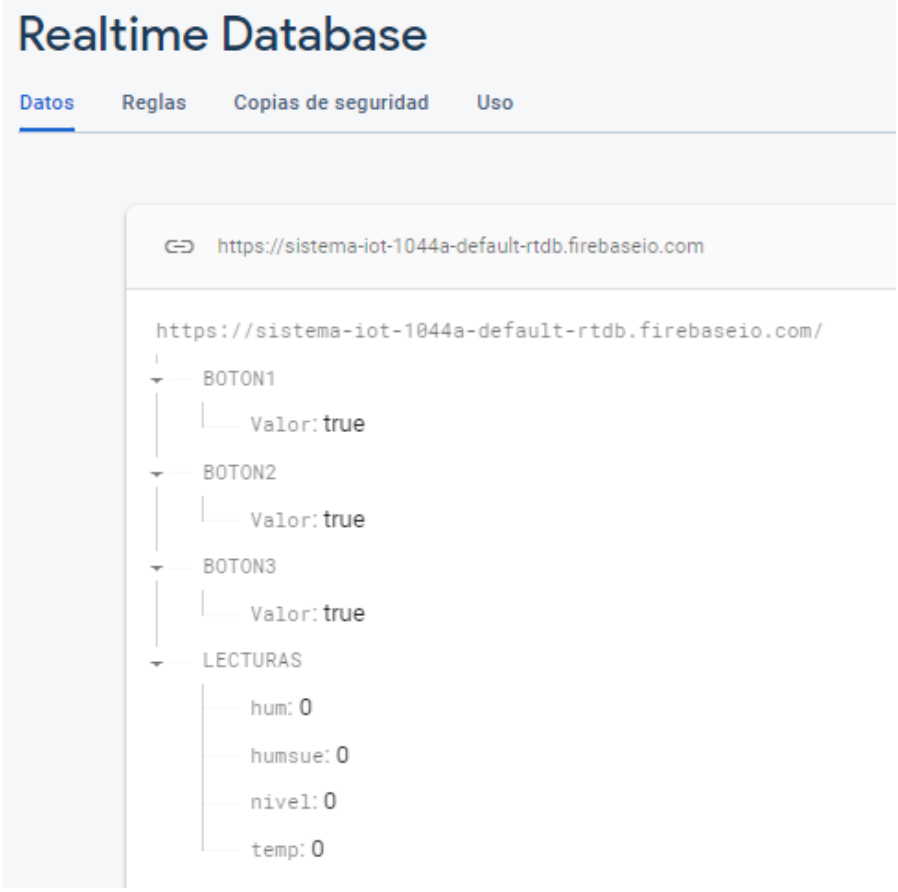
A las subvariables de los actuadores se les coloco un valor de true, cabe resaltar que el valor solo puede ser true o false. True significa que el actuador está encendido y false que esta apagado.

Para la variable de lecturas, se creó cuatro subvariables, una para la lectura de temperatura, una para humedad relativa, una para humedad del suelo y una para el nivel de tanque de agua. El valor que se colocó a estas variables fue de 0.

La Figura 38 muestra lo explicado anteriormente.

Figura 38

Creación de Variables y Subvariables en la Base de Datos



3.6.5. Configuración de la interfaz de la aplicación móvil en Android Studio

En esta sección mostraremos la configuración y programación de la aplicación móvil en el software Android Studio. Para realizar la interfaz, aparte de elementos básicos como lo son los botones y los cuadros de textos, se utilizó animaciones, estas animaciones se descargaron de la página oficial de LottieFiles, como se muestra en la Figura 39.

El primer paso que se llevó a cabo fue la descarga de las animaciones en la página de LottieFiles. En ella se descargó animaciones para acompañar la visualización de los datos de temperatura, humedad relativa, humedad del suelo y nivel del tanque.

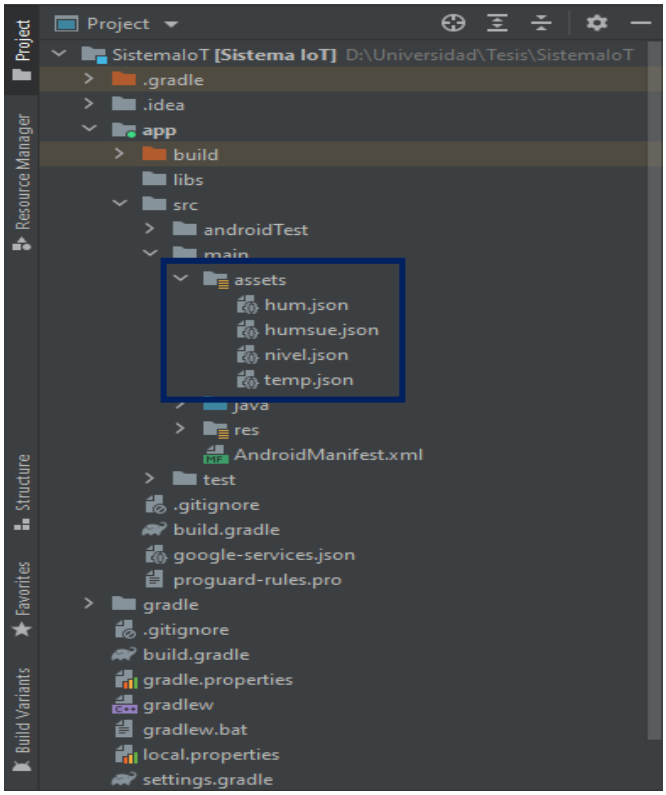
Figura 39

Página principal de LottieFiles

El siguiente paso se aprecia en la Figura 40, este fue crear una carpeta assets en Project/SistemaIoT[Sistema IoT]/app/scr/main, en esta carpeta se guardó las animaciones que se descargaron. Assets es una herramienta de Android Studio que permite generar tus propios iconos para tu aplicación.

Figura 40

Agregación de las Animación a Android Studio

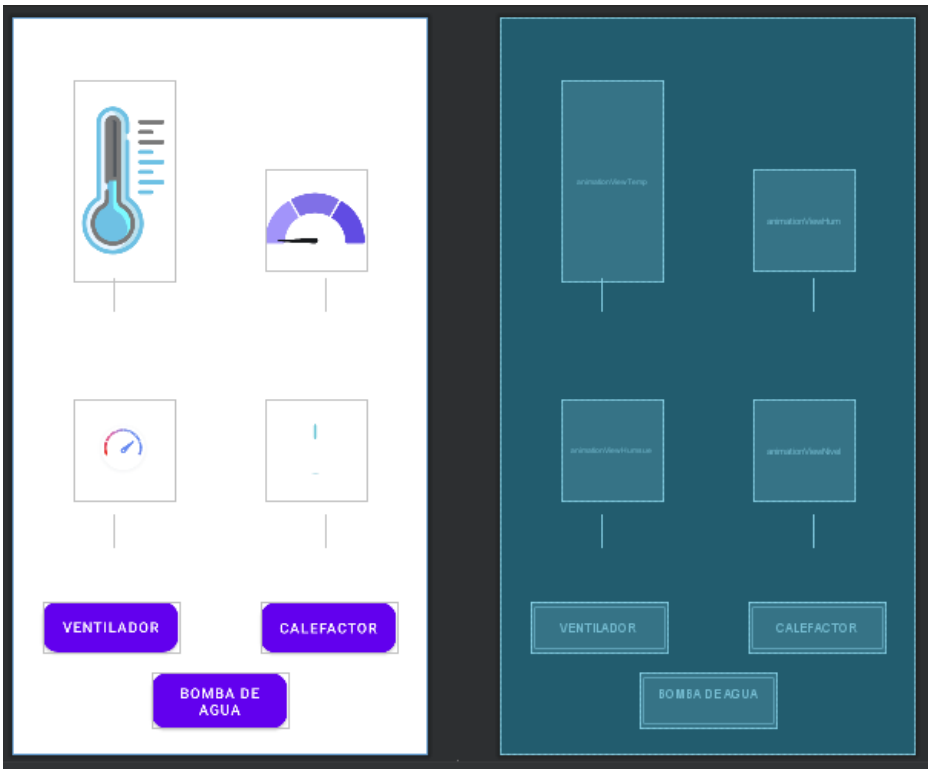


Una vez que se guardaron las animaciones en el proyecto, se procedió a crear la interfaz de la aplicación móvil. Se agregó, nombró y acodó los cuadros de texto, botones y cuadros de animación. En los cuadros de texto se visualizan las mediciones que obtienen los sensores, los botones se utilizan para encender o apagar los actuadores y en los cuadros de animación van las animaciones descargadas.

La Figura 41 muestra en el lado derecho la posición de los botones, cuadros de texto y cuadros de animaciones, mientras que en el lado izquierdo se observa cómo se vería la interfaz de la aplicación móvil si la abriéramos desde un teléfono inteligente.

Figura 41

Interfaz de la Aplicación Móvil



Se continuo con la programación de la interfaz de la aplicación móvil. Primero se creó las variables, se agregó el tipo y el nombre de los elementos que se utilizaron, tal como se muestra en la Figura 42.

Figura 42

Creación de Variables

```
TextView temp, hum, humsue, nivel;
Button button1, button2, button3;
LottieAnimationView animationViewTemp, animationViewHum, animationViewHumsue, animationViewNivel;
```

Después se creó tres variables de tipo boolean, se les dio un valor inicial de false, así como se visualiza en la Figura 43, el contenido de estas variables se imprime en la base de datos.

Figura 43

Variables para la Impresión en la Base de Datos

```
boolean estado1 = false, estado2 = false, estado3 = false;
```

Se creó referencias para poder acceder a la base de datos, tal cual se observa en la Figura 44.

Figura 44

Referencias para Acceder a la Base de Datos

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference referenceEntrada1 = database.getReference( path: "BOTON1");
DatabaseReference referenceEntrada2 = database.getReference( path: "BOTON2");
DatabaseReference referenceEntrada3 = database.getReference( path: "BOTON3");
DatabaseReference referenceEntrada4 = database.getReference( path: "LECTURAS");
```

Con la función findViewById se enlazo los elementos de la interfaz de la aplicación móvil con las variables del código, justo como se ve en la Figura 45.

Figura 45

Enlazamiento de Elementos de la Interfaz con Variables del Código

```
temp = findViewById(R.id.temp);
hum = findViewById(R.id.hum);
humsue = findViewById(R.id.humsue);
nivel = findViewById(R.id.nivel);
button1 = findViewById(R.id.button1);
button2 = findViewById(R.id.button2);
button3 = findViewById(R.id.button3);
animationViewTemp = findViewById(R.id.animationViewTemp);
animationViewHum = findViewById(R.id.animationViewHum);
animationViewHumsue = findViewById(R.id.animationViewHumsue);
animationViewNivel = findViewById(R.id.animationViewNivel);
```

Se utilizó la función addValueEventListener para leer y realizar cambios en la base de datos, esta función junto a una función if permitió cambiar el estado y por lo tanto el encendido o apagado de los botones que controlan los actuadores del sistema IoT. Se realizo este paso para los tres botones, como se muestran en las Figura 46, Figura 47 y Figura 48.

Figura 46

Cambio de Estado del Botón 1

```
referenceEntrada1.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        String estadoButton1 = snapshot.child("Valor").getValue().toString();
        if(estadoButton1.equals("true")){
            button1.setBackground(getResources().getDrawable(R.drawable.botonpulsado));
        } else {
            button1.setBackground(getResources().getDrawable(R.drawable.botonnopulsado));
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Figura 47

Cambio de Estado del Botón 2

```
referenceEntrada2.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        String estadoButton1 = snapshot.child("Valor").getValue().toString();
        if(estadoButton1.equals("true")){
            button2.setBackground(getResources().getDrawable(R.drawable.botonpulsado));
        } else {
            button2.setBackground(getResources().getDrawable(R.drawable.botonnopulsado));
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Figura 48

Cambio de Estado del Botón 3

```
referenceEntrada3.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        String estadoButton1 = snapshot.child("Valor").getValue().toString();
        if(estadoButton1.equals("true")){
            button3.setBackground(getResources().getDrawable(R.drawable.botonpulsado));
        } else {
            button3.setBackground(getResources().getDrawable(R.drawable.botonnopulsado));
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Con los comandos que se aprecian en las Figura 49, Figura 50 y Figura 51, se imprime el contenido de la variable boolean en la base de datos, este cambio se realiza cuando se presiona el botón.

Figura 49

Impresión del Estado del Botón 1 en la Base de Datos

```
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        HashMap<Object, Object> info = new HashMap<>();
        estado1 = !estado1;
        info.put("Valor", estado1);
        referenceEntrada1.setValue(info);
    }
});
```

Figura 50

Impresión del Estado del Botón 2 en la Base de Datos

```
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        HashMap<Object, Object> info = new HashMap<>();
        estado2 = !estado2;
        info.put("Valor", estado2);
        referenceEntrada2.setValue(info);
    }
});
```

Figura 51

Impresión del Estado del Botón 3 en la Base de Datos

```
button3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        HashMap<Object, Object> info = new HashMap<>();
        estado3 = !estado3;
        info.put("Valor", estado3);
        referenceEntrada3.setValue(info);
    }
});
```

La última parte del código está enfocada en la visualización de las medidas que obtienen los sensores, primero se creó variables de tipo String, una por cada parámetro de medición establecido, para almacenar los valores de la base de datos, después se imprimen en el cuadro de texto y se activa la animación en la aplicación móvil. La Figura 52 muestra lo anteriormente descrito.

Figura 52

Visualización de las Medidas y Animaciones en la Interfaz

```
referenceEntrada4.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        String HS = snapshot.child("humsue").getValue().toString();
        humsue.setText(HS);
        String N = snapshot.child("nivel").getValue().toString();
        nivel.setText(N);
        String T = snapshot.child("temp").getValue().toString();
        temp.setText(T);
        String H = snapshot.child("hum").getValue().toString();
        hum.setText(H);
        animationViewHumsue.playAnimation();
        animationViewNivel.playAnimation();
        animationViewTemp.playAnimation();
        animationViewHum.playAnimation();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

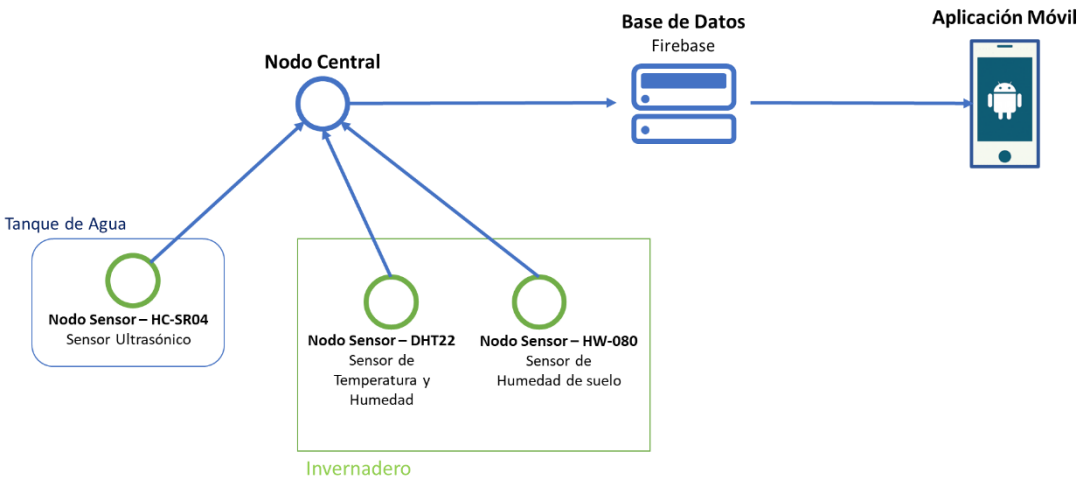
3.6.6. Programación del funcionamiento de los nodos en Arduino IDE

Para concluir con la configuración y programación del sistema IoT, tenemos que programar el funcionamiento de los nodos. El sistema cuenta con cuatro nodos, tres nodos sensor, uno para la temperatura y humedad relativa, uno para la humedad del suelo y uno para el nivel del tanque, y un nodo central.

El funcionamiento de los tres nodos sensor y del nodo central se observa en la Figura 53.

Figura 53

Diagrama Explicativo del Funcionamiento de los Nodos



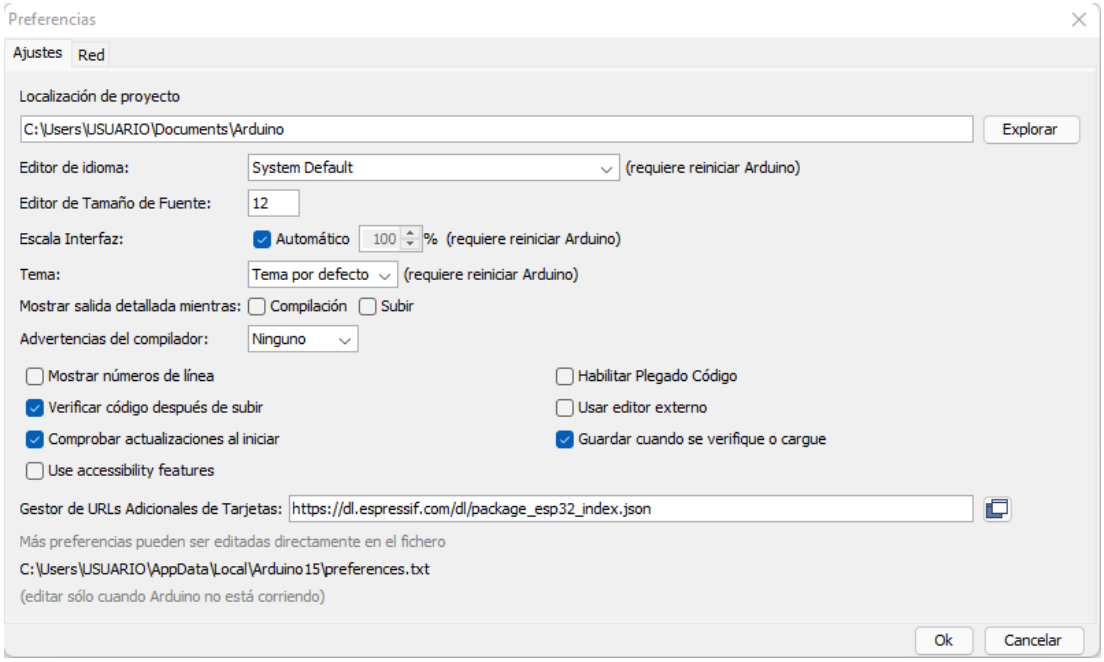
Los nodos sensor mediante el protocolo de comunicación ESP-NOW envían las mediciones que recolectaron al nodo central, el nodo central sube la información recibida a la base de datos, también activa o desactiva los actuadores cuando los parámetros climáticos están por debajo o sobre de lo recomendado.

Antes de empezar con la programación, primero se configuró Arduino IDE para poder trabajar con el microcontrolador ESP32. Para realizar esta configuración se accedió al menú Archivo/Preferencias, en la ventaja Ajustes/Gestor de URLs Adicionales de Tarjetas se agregó https://dl.espressif.com/dl/package_esp32_index.json. Esto permitió trabajar con el microcontrolador y el protocolo de comunicación sin tener ningún problema, no fue necesario descargar alguna librería adicional para utilizar el protocolo ESP-NOW.

En la Figura 54 se puede observar la configuración anteriormente descrita.

Figura 54

Configuración de Arduino IDE para Trabajar con el Microcontrolador ESP32



3.6.6.1. Nodo Central

El nodo central es el nodo principal del sistema IoT, está encargado de recibir la información de los tres nodos sensor y subirla a la base de datos, también cumple la función de mantener los parámetros climáticos considerados en su rango optimo, esto lo hace activando o desactivando los actuadores cuando sea necesario. El nodo

central se conecta a un servidor NTP para poder obtener la hora y así diferenciar entre el horario diurno y nocturno.

El primero paso que se realizo fue la obtención de la dirección MAC del nodo central. La dirección MAC del nodo central se colocó en todos los nodos sensores y es fundamental para la comunicación mediante el protocolo ESP-NOW.

Para obtener la dirección MAC se cargó el código que se muestra en la Figura 55 en el microcontrolador.

Figura 55

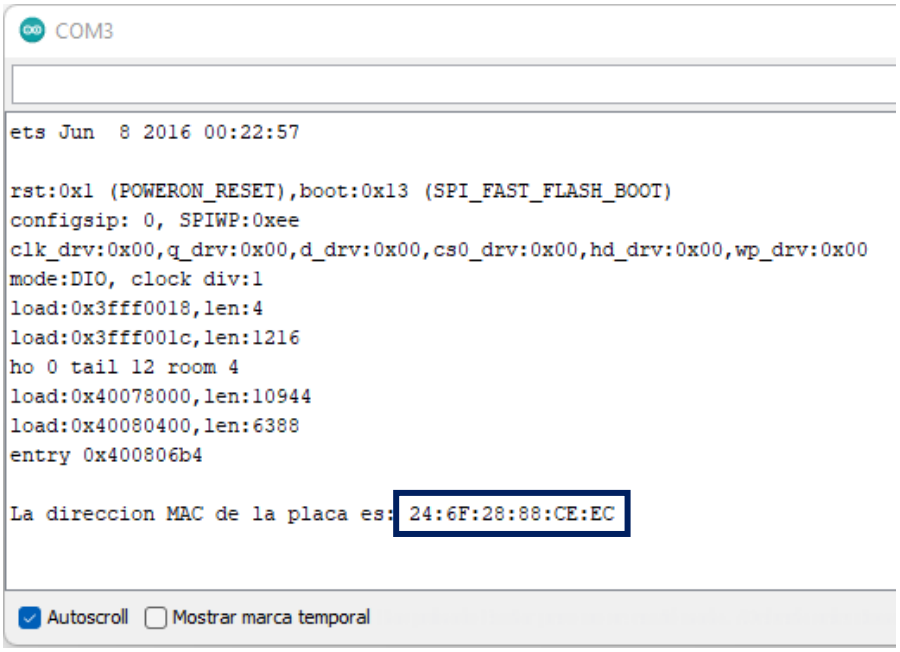
Código para Obtener la Dirección MAC del Nodo Central

```
#include "WiFi.h"

void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.print("La direccion MAC de la placa es: ");
  Serial.println(WiFi.macAddress());
}

void loop() {
}
```

Después de colocar el código anterior, encender el microcontrolador y abrir el monitor serial, se muestra la ventana que se ve en la Figura 56.

Figura 56

Dirección MAC del Nodo Central



El segundo paso que se llevó a cabo para programar el nodo central fue la descarga de las librerías. Para este nodo se necesitó la librería de la base de datos, Firebase ESP32 Client, y la librería para poder utilizar el protocolo NTP, ESP32Time.

Se ingresó al menú Programa/Incluir Librería/Administrar Bibliotecas y se descargaron, tal como se muestran en las Figura 57 y Figura 58.

Figura 57

Descargo de Librería Firebase ESP32 Client

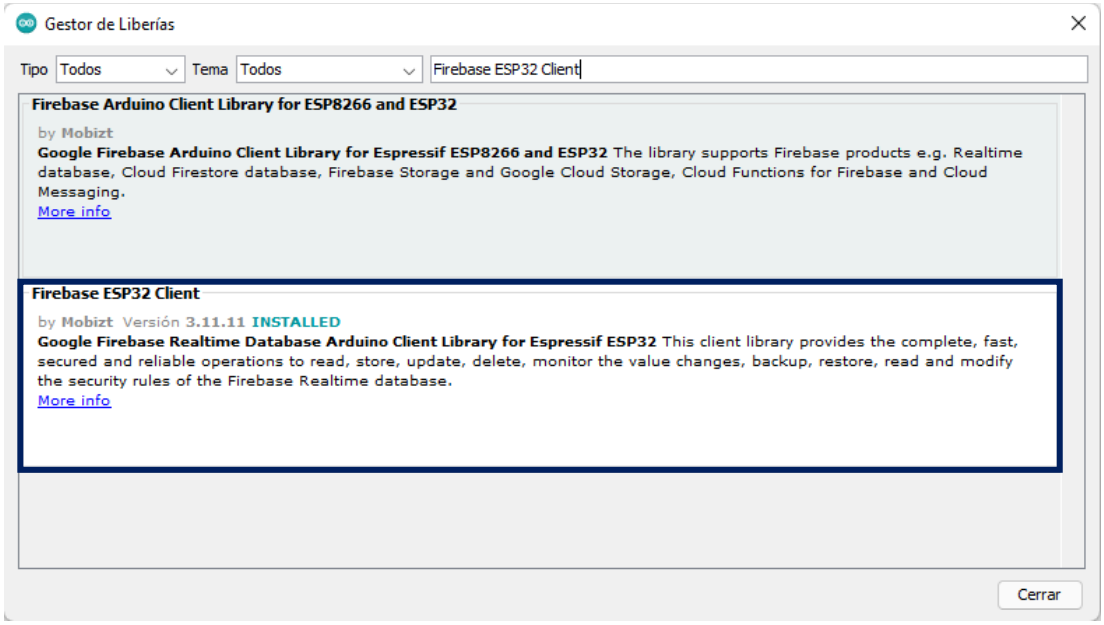
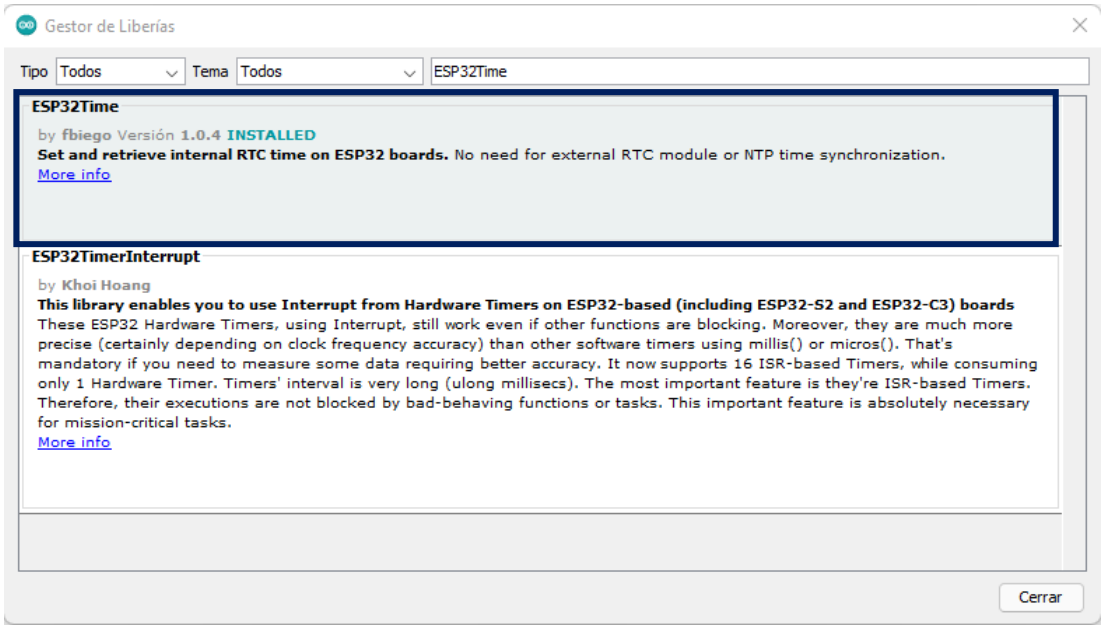


Figura 58

Descargo de Librería ESP32Time



Luego de descargar las librerías se continuo con la programación. Para comenzar se añadieron todas las librerías que se utilizaron. Se agregaron las librerías descargadas y las que utiliza el protocolo ESP-NOW, así como se visualiza en la Figura 59.

Figura 59

Inclusión de Librerías – Nodo Central

```
#include "esp_now.h"
#include "WiFi.h"
#include "FirebaseESP32.h"
#include "ESP32Time.h"
```

Se creo dos variables para el acceso a la red, una que contiene el SSID y la otra su contraseña, tal cual se observa en la Figura 60.

Figura 60

Variables para Acceso de la Red – Nodo Central

```
const char* ssid = "Red WiFi";
const char* password = "SistemaIoT";
```

También se definió el URL y el secreto de la base de datos, justo como se ve en la Figura 61. Estos datos se obtuvieron del proyecto creado en Firebase y son necesarios para poder ingresar y mandar información a la base de datos.

Figura 61

URL y Secreto de la Base de Datos – Nodo Central

```
#define URL "sistema-iot-1044a-default-rtdb.firebaseio.com"
#define secreto "STaotDwJkS6Zd3yHFIzQ9PFBBg2U6Mhl8ebattoY"
```

Se continuo con la configuración del servidor NTP, se creó variables para almacenar la dirección del servidor, el desplazamiento GMT y la compensación de luz diurna, como se muestra en la Figura 62.

GMT (Greenwich Mean Time) o Tiempo Medio de Greenwich es un estándar de tiempo, es equivalente a UTC (Universal Time Coordinated) o Tiempo Universal Coordinado y se utiliza como referencia de la hora internacional. El Perú se encuentra a 5 horas después del GMT (-5 GMT/UTC), por esa razón en el desplazamiento GMT se colocó -5 y se multiplico por la cantidad de segundos que hay en una hora. Como el Perú no posee un horario de verano ni de invierno, se puso 0 en compensación de luz diurna.

Figura 62

Variables para el Servidor NTP – Nodo Central

```
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = -5*3600;
const int daylightOffset_sec = 0;
```

También se creó variables para el servidor NTP, variables para almacenar los datos que nos envían los nodos sensor, variables para los actuadores y variables para cambiar el estado de los actuadores en la base de datos, tal como se visualiza en la Figura 63.

Figura 63

Variables – Nodo Central

```
ESP32Time NTP;

String horaNTP;
int hora;

float temperatura;
float humedad;
int humedadsuelo;
float niveltanque;

int ventilador = 2;
int calefactor = 4;
int bomba = 5;

boolean encendido = true;
boolean apagado = false;
```

A continuación, se creó tres objetos tipo JSON para enviar la información a la base de datos, además, se creó cuatro variables tipo String, estas variables permitieron cambiar el estado del actuador en la base de datos. La Figura 64 muestra la creación de los objetos y variables.

JSON es un formato ligero de intercambio de datos, su principal función es almacenar e intercambiar información de texto.

Figura 64

Variables JSON – Nodo Central

```
FirebaseData myFireBaseData;
FirebaseJson myJson;
FirebaseJsonData myJsonData;

String estados, estado1, estado2, estado3;
```

Acto seguido, se creó una estructura para almacenar los datos del mensaje que se recibirán, así como se observa en la Figura 65. Es importante resaltar que la estructura del mensaje debe ser igual en todos los nodos.

Figura 65

Estructura del Mensaje – Nodo Central

```
typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;
```

Se continúo creando una variable para almacenar los datos de las variables de la estructura del mensaje, tal cual se ve en la Figura 66.

Figura 66

Variable para Almacenar la Estructura del Mensaje – Nodo Central

```
EstructuraMensaje Invernadero;
```

Se creó tres estructuras, una para cada nodo de sensor, de este modo podemos asignar los datos recibidos a la placa correspondiente, también se creó una matriz para contener todas las estructuras, justo como se muestra en la Figura 67.

Figura 67

Estructuras para los Nodos Sensor y Matriz para las Estructuras – Nodo Central

```
EstructuraMensaje placa1;
EstructuraMensaje placa2;
EstructuraMensaje placa3;

EstructuraMensaje boardsStruct[3] = {placa1, placa2, placa3};
```

.

Se prosiguió definiendo la función de devolución de llamada de recepción, como se visualiza en la Figura 68, se llamará a esta función cuando el nodo reciba los datos.

Figura 68

Definición de Función de Devolución de Llamada de Recepción – Nodo Central

```
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
```

Dentro de la función de devolución de llamada de recepción, obtendremos la dirección MAC del nodo sensor que envió el mensaje. En la Figura 69 se puede apreciar el código.

Figura 69

Obtención de la Dirección MAC del Nodo Sensor Remitente – Nodo Central

```
char macStr[18];
Serial.print("Mensaje recibido de: ");
snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x", mac_addr[0],
mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
Serial.println(macStr);
```

También obtendremos la estructura del mensaje recibido, para identificar el nodo usaremos su ID. Entonces, si queremos igualar el valor de la estructura del mensaje recibido con el valor de la matriz creada para contener las estructuras, tenemos que restar 1 porque la matriz empieza en 0. Esto se realizó para poder obtener el dato necesario del nodo.

Para un mejor entendimiento se muestra la Tabla 13.

Tabla 13

Igualación de la Estructura del Mensaje Recibido con la Matriz para Estructuras – Nodo Central

Matriz	boardsStruct[0]	boardsStruct[1]	boardsStruct[2]
Mensaje recibido	placa 1	placa 2	placa 3
ID del nodo	1	2	3

En la Figura 70 se muestra el código.

Figura 70

Igualación de Lecturas Recibidas con la Matriz para Estructuras – Nodo Central

```
memcpy(&Invernadero, incomingData, sizeof(Invernadero));
Serial.printf("Placa N°: %u --- Tamaño del mensaje: %u bytes\n", Invernadero.id, len);

boardsStruct[Invernadero.id-1].temp = Invernadero.temp;
boardsStruct[Invernadero.id-1].hum = Invernadero.hum;
boardsStruct[Invernadero.id-1].humsue = Invernadero.humsue;
boardsStruct[Invernadero.id-1].nivel = Invernadero.nivel;
```

En el Void Setup, se inicializo el monitor serie a 115 200 baudios, tal como se visualiza en la Figura 71. El baudio es una unidad de medida que se utiliza en telecomunicaciones, representa el número de símbolos por segundo en un medio de transmisión.

Figura 71

Inicialización del Monitor Serie – Nodo Central

```
Serial.begin(115200);
```

Después, con ayuda de las variables SSID y password, nos conectamos a la red e imprimimos el canal WiFi que se nos asignó, así como se ve en la Figura 72.

Figura 72

Conexión a la Red e Impresión del Canal WiFi – Nodo Central

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
}

Serial.print("Canal WiFi: ");
Serial.println(WiFi.channel());
```

Con las líneas de código que se observa en la Figura 73 se accedió a la base de datos, para esto se usó el URL y el secreto.

Figura 73

Conexión a la Base de Datos

```
Firebase.begin(URL, secreto);
Firebase.reconnectWiFi(true);     //Reconexion automatica a la base de datos
Serial.println("Conexion con la base de datos exitosa");
```

Se configuró el microcontrolador como un Access Point y una estación WiFi, tal cual se muestra en la Figura 74, de esta forma se pudo conectar a la red y los nodos sensor se pudieron comunicar con el nodo central.

La conexión WiFi es proporcionada por un Access Point (AP), este actúa como un centro para uno o más dispositivos. Se denomina estación WiFi (STA) a cada dispositivo que se conecta a una red WiFi.

Figura 74

Configuración como Access Point y Estación WiFi

```
WiFi.mode(WIFI_AP_STA);
```

Luego, se inicializó el protocolo de comunicación ESP-NOW y se registró la función de devolución de llamada de recepción, esta función se llama cuando se reciben datos, así como se visualiza en la Figura 75.

Figura 75

Inicialización del Protocolo ESP-NOW – Nodo Central

```
if (esp_now_init() != ESP_OK) {
  Serial.println("Error de inicializacion del protocolo ESP-NOW");
  return;
}

esp_now_register_recv_cb(OnDataRecv);
```

Para finalizar con el Void Setup, se configuro el servidor NTP, tal cual se observa en la Figura 76. En esta parte se accede al servidor para obtener la hora y se agrega el desplazamiento GMT y la compensación de luz diurna, y se definió el modo de operación de las variables de los actuadores.

Figura 76

Definición del Servidor NTP y Modo de Operación de los Actuadores – Nodo Central

```
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

pinMode(ventilador, OUTPUT);
pinMode(calefactor, OUTPUT);
pinMode(bomba, OUTPUT);
```

La última parte de la programación es el Void Loop. Esta es la parte del código que se ejecuta una y otra vez.

Primero, se obtuvo el dato necesario del nodo, para ello se igualo la variable con la matriz correspondiente, justo como se ve en la Figura 77.

Figura 77

Obtención de lecturas de los Nodos Sensor – Nodo Central

```
temperatura = boardsStruct[0].temp;
humedad = boardsStruct[0].hum;
niveltanque = boardsStruct[1].nivel;
humedadsuelo = boardsStruct[2].humsue;
```

Segundo, se envió la información a la base de datos. Tenemos que definir la ubicación exacta y agregar la variable, como se muestra en la Figura 78.

Figura 78

Envío de Información a la Base de Datos – Nodo Central

```
Firebase.set(myFireBaseData, "/LECTURAS/temp", temperatura);
Firebase.set(myFireBaseData, "/LECTURAS/hum", humedad);
Firebase.set(myFireBaseData, "/LECTURAS/humsue", humedadsuelo);
Firebase.set(myFireBaseData, "/LECTURAS/nivel", niveltanque);
```

A continuación, se accedió a la base de datos para obtener el estado del actuador. También se creó una condición, si el estado es igual a true, el actuador se encenderá, si no lo es, el actuador se apagará. Se repitió el mismo paso para los tres actuadores, tal como se visualiza en la Figura 79, Figura 80 y Figura 81.

Figura 79

Condición para el Cambio del Estado del Ventilador – Nodo Central

```
Firebase.get(myFireBaseData, "/BOTON1");
estados = myFireBaseData.jsonString();
myJson.setJsonData(estados);
myJson.get(myJsonData, "/Valor");
estado1 = myJsonData.stringValue;

if(estado1 == "true") {
  digitalWrite(ventilador, LOW);
} else {
  digitalWrite(ventilador, HIGH);
}
```

Figura 80

Condición para el Cambio del Estado del Calefactor – Nodo Central

```
Firebase.get(myFireBaseData, "/BOTON2");
estados = myFireBaseData.jsonString();
myJson.setJsonData(estados);
myJson.get(myJsonData, "/Valor");
estado2 = myJsonData.stringValue;

 if(estado2 == "true") {
  digitalWrite(calefactor, LOW);
} else {
  digitalWrite(calefactor, HIGH);
}
```

Figura 81

Condición para el Cambio de Estado de la Bomba de Agua – Nodo Central

```
Firebase.get(myFireBaseData, "/BOTON3");
estados = myFireBaseData.jsonString();
myJson.setJsonData(estados);
myJson.get(myJsonData, "/Valor");
estado3 = myJsonData.stringValue;

if(estado3 == "true") {
  digitalWrite(bomba, LOW);
} else {
  digitalWrite(bomba, HIGH);
}
```

Después, se obtuvo la hora. Se hizo una conversión de variable String a Int para poder trabajar con ella, así como se observa en la Figura 82.

Figura 82

Obtención de la Hora y Cambio de Variable a Int – Nodo Central

```
horaNTP = NTP.getTime("%H");
hora = horaNTP.toInt();
```

Para finalizar, se programó el sistema de calefacción y ventilación y el sistema de riego, tal cual se ve la Figura 83, Figura 84, Figura 85 y Figura 86. Para el sistema de calefacción y ventilación se tuvo en cuenta que los parámetros climáticos varían entre el día y la noche.

Figura 83

Sistema para el Control de Temperatura – Horario Diurno – Nodo Central

```
if (hora >= 6 & hora <= 18){
    if (temperatura >= 21){
      digitalWrite(ventilador, LOW);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", encendido);
    }
    else {
      digitalWrite(ventilador, HIGH);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", apagado);

    }
    if (temperatura <= 14){
      digitalWrite(calefactor, LOW);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", encendido);
    }
    else {
      digitalWrite(calefactor, HIGH);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", apagado);
    }
}
```

Figura 84

Sistema para el Control de Temperatura – Horario Nocturno – Nodo Central

```
if (hora >= 19 & hora <= 5){
    if (temperatura >= 16){
      digitalWrite(ventilador, LOW);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", encendido);
    }
    else {
      digitalWrite(ventilador, HIGH);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", apagado);
    }
    if (temperatura <= 9){
      digitalWrite(calefactor, LOW);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", encendido);
    }
    else {
      digitalWrite(calefactor, HIGH);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", apagado);
    }
}
```

Figura 85

Sistema para el Control de Humedad – Nodo Central

```
if (humedad <= 60){
  digitalWrite(ventilador, LOW);
  Firebase.set(myFireBaseData, "/BOTON1/Valor", encendido);
  digitalWrite(calefactor, HIGH);
  Firebase.set(myFireBaseData, "/BOTON2/Valor", apagado);
}
if (humedad >= 70){
  digitalWrite(calefactor, LOW);
  Firebase.set(myFireBaseData, "/BOTON2/Valor", encendido);
  digitalWrite(ventilador, HIGH);
  Firebase.set(myFireBaseData, "/BOTON1/Valor", apagado);
}
```

Figura 86

Sistema de Riego – Nodo Central

```
if (humedadsuelo <= 50){
  digitalWrite(bomba, LOW);
  Firebase.set(myFireBaseData, "/BOTON3/Valor", encendido);
}
if (humedadsuelo >= 75) {
  digitalWrite(bomba, HIGH);
  Firebase.set(myFireBaseData, "/BOTON3/Valor", apagado);
}
```

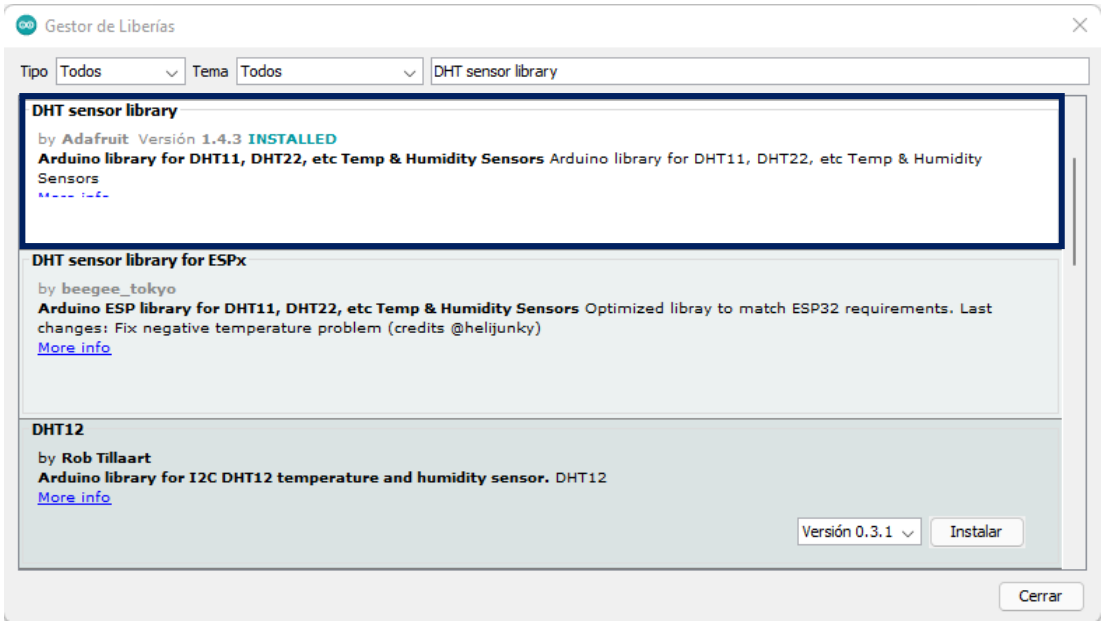
3.6.6.2. Nodo Sensor – DHT22

El Nodo Sensor – DHT22 está encargado de enviar su ID, la temperatura y la humedad relativa al Nodo Central cada vez que las lecturas varíen, esto lo hace mediante el protocolo ESP-NOW.

Lo primero que se realizo fue descargar la librería que se utilizó. Para eso se fue al menú Programa/Incluir Librería/Administrar Bibliotecas y se descargó la librería DHT sensor library de Adafruit para poder utilizar el sensor DHT22, tal como se muestra en la Figura 87.

Figura 87

Descargo de Libraría SHT Sensor Library – Nodo Sensor – DHT22



Después de descargar la librería se procedió con la programación del nodo. Primero se añadieron las librerías que se utilizaron, así como se visualiza en la Figura 88.

Figura 88

Inclusión de Librerías – Nodo Sensor – DHT22

```
#include "esp_now.h"
#include "WiFi.h"
#include "DHT.h"
```

Se siguió con la creación de una variable y asignación de pin para el sensor DHT22, tal cual se observa en la Figura 89, la librería exige establecer una variable en la cual se debe especificar el modelo del sensor para terminar su configuración.

Figura 89

Variables para el Sensor DHT22 – Nodo Sensor – DHT22

```
int sensor = 2;
DHT dht(sensor,DHT22);
```

Se creó una variable para almacenar la dirección MAC de la placa receptora (Nodo Central), justo como se ve en la Figura 90.

Figura 90

Variable para el Almacenamiento de la Dirección MAC – Nodo Sensor – DHT22

```
uint8_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x88, 0xCE, 0xEC};
```

Las cuatro variables que aparecen en la Figura 91 sirven para almacenar las lecturas del sensor, las primeras dos guardan las lecturas enviadas y las últimas dos guardan las nuevas lecturas.

Figura 91

Variables para el Almacenamiento de Lecturas – Nodo Sensor – DHT22

```
//Variables para almacenar las lecturas enviadas
float LecturaAntiguaTemp;
float LecturaAntiguaHum;

//Variables para almacenar las nuevas lecturas del sensor
float temperatura;
float humedad;
```

Luego, se creó una estructura que contiene los datos del mensaje que se enviara, como se muestra en la Figura 92. La estructura del mensaje debe ser igual en todos los nodos, es por eso por lo que incluye todas las variables.

Figura 92

Estructura del Mensaje – Nodo Sensor – DHT22

```
typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;
```

Se continúo creando una variable para almacenar los datos de las variables de la estructura del mensaje, tal como se visualiza en la Figura 93.

Figura 93

Variable para Almacenar la Estructura del Mensaje – Nodo Sensor – DHT22

```
EstructuraMensaje Invernadero;
```

A continuación, se definió la función de devolución de llamada (OnDataSent), así como se observa en la Figura 94. Esta función se ejecuta cuando se envía el mensaje, imprime si el mensaje se entregó correctamente o no.

Figura 94

Definición de la Función de Devolución de Llamada – Nodo Sensor – DHT22

```
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio fallido");
}
```

Se prosiguió con el Void Setup. Primero se inicializo el monitor serie a 115 200 baudios y se configuro el microcontrolador como una estación WiFi, tal cual se ve en las Figura 95 y Figura 96, esto nos permite conectarnos a red del Nodo Central.

Figura 95

Inicialización del Monitor Serie – Nodo Sensor – DHT22

```
Serial.begin(115200);
```

Figura 96

Configuración como Estación WiFi – Nodo Sensor – DHT22

```
WiFi.mode(WIFI_STA)
```

Se inicializo el protocolo de comunicación ESP-NOW y se registró la función de devolución de llamada definida previamente, justo como se muestra en la Figura 97.

Figura 97

Inicialización del Protocolo ESP-NOW – Nodo Sensor – DHT22

```
if (esp_now_init() != ESP_OK) {
  Serial.println("Error de inicializacion del protocolo ESP-NOW");
  return;
}

esp_now_register_send_cb(OnDataSent);
```

Lo siguiente que se realizo fue el emparejamiento como par, primero se tiene que agregar el dispositivo a la lista de dispositivos emparejados antes de enviar datos. Para enviar datos desde el nodo sensor al nodo central, ambos nodos deben de tener el mismo canal de red. En las líneas de código que se muestran en la Figura 98 se puede ver como se definió el canal de red número uno y se decidió no encriptar los mensajes, también hay una función para verificar si se agregó correctamente.

Figura 98

Emparejamiento como par – Nodo Sensor – DHT22

```
esp_now_peer_info_t peerInfo;
memcpy(peerInfo.peer_addr, broadcastAddress, 6);
peerInfo.channel = 1;
peerInfo.encrypt = false;

if (esp_now_add_peer(&peerInfo) != ESP_OK){
  Serial.println("No se puedo agregar el par");
  return;
}
```

Para finalizar con el Void Setup, se inicializo la librería del sensor y se definió su modo de operación, como se ve en la Figura 99.

Figura 99

Definición del Modo de Operación del Sensor – Nodo Sensor – DHT22

```
dht.begin();

pinMode(sensor, INPUT);
```

La última parte de la programación del Nodo Sensor – DHT22 es el Void Loop. Primero, mediante una función que nos ofrece la librería descargada, obtuvimos la medición de la temperatura y humedad, tal como se muestra en la Figura 100.

Figura 100

Obtención de Lecturas del Sensor – Nodo Sensor – DHT22

```
temperatura = dht.readTemperature();  //Temperatura en °C
humedad = dht.readHumidity();         //Humedad en %
```

Después, se estableció los valores de las variables en la estructura del mensaje, así como se visualiza en la Figura 101. Este nodo es el número uno, es por eso por lo que su ID es el número uno.

Figura 101

Igualación de Lecturas con Variables – Nodo Sensor – DHT22

```
Invernadero.id = 1;
Invernadero.temp = temperatura;
Invernadero.hum = humedad;
```

Con la ayuda de la función if se creó una condición para solo enviar el mensaje a través del protocolo de comunicación ESP-NOW cuando hayan variado las lecturas del sensor, tal cual se observa en la Figura 102.

Figura 102

Envío de Lecturas – Nodo Sensor – DHT22

```
if (temperatura != LecturaAntiguaTemp || humedad != LecturaAntiguaHum) {
  Serial.println("La temperatura y/o humedad vario");

  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Invernadero, sizeof(Invernadero));
  if (result == ESP_OK) {
    Serial.println("Envio del mensaje exitoso");
  }
  else {
    Serial.println("Envio del mensaje fallido");
  }
}
```

Por último, se igualó las lecturas a las variables para almacenar las lecturas antiguas, justo como se ve en la Figura 103.

Figura 103

Igualación de Lecturas – Nodo Sensor – DHT22

```
LecturaAntiguaTemp = temperatura;
LecturaAntiguaHum = humedad;
```

3.6.6.3. Nodo Sensor – HC-SR04

El Nodo Sensor – HC-SR04 envía su ID y el porcentaje del nivel del tanque de agua al Nodo Central, solo envía las lecturas si hay una variación en ellas y lo hace utilizando el protocolo de comunicación ESP-NOW.

Igual que el nodo anterior, primero se incluyeron las librerías que se utilizaron, como se ve en la Figura 104, para este nodo solo son necesarias las librerías que utiliza el protocolo de comunicación.

Figura 104

Inclusión de Librerías – Nodo Sensor – HC-SR04

```
#include "esp_now.h"
#include "WiFi.h"
```

Después se continuó creando variables para el transmisor y receptor del sensor, para el cálculo de la distancia entre el sensor y el líquido del tanque, para las medidas del tanque de agua y para el cálculo de los tres volúmenes, también se asignaron los pines para el sensor, tal como se muestra en la Figura 105.

Figura 105

Variables para el Sensor y para el Cálculo del Volumen – Nodo Sensor – HC-SR04

```
//Sensor
int trig = 2;          //Transmisor
int eco = 4;           //Receptor
long duracion;         //Variable para duracion del pulso
float distancia;       //Variable para hallar la distancia

//Tanque de Agua
float pi = 3.1416;     //Variable pi
float r = 74.5;        //Radio del tanque (m)
float h = 115;         //Altura del tanque (m)
float Vt;              //Volumen total del tanque (m3)
float Vv;              //Volumen vacio del tanque (m3)
float Vl;              //Volumen lleno del tanque (m3)
```

Como en el nodo anterior, se creó una variable para almacenar la dirección MAC del Nodo Central, se crearon variables para almacenar las lecturas, se creó una estructura para almacenar los datos del mensaje, también una variable para almacenar la estructura y se definió la función de devolución de llamada, así como se visualiza en la Figura 106.

Figura 106

Definición del Resto de Variables – Nodo Sensor – HC-SR04

```
uint8_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x88, 0xCE, 0xEC};

float LecturaAntiguaNivel;

float niveltanque;

typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;

EstructuraMensaje Invernadero;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio fallido");
}
```

El Void Setup tiene el mismo funcionamiento en los tres nodos sensor. Se habilita el monitor serie a 115 200 baudios, se configura el ESP32 como una estación WiFi, inicializa el protocolo ESP-NOW, se registra la función de devolución de llamada y se empareja como un par y, por último, se define el modo de operación del sensor, tal cual se observa en la Figura 107.

Figura 107

Void Setup – Nodo Sensor – HC-SR04

```
void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error de inicializacion del protocolo ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnDataSent);

  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("No se puedo agregar el par");
    return;
  }

  pinMode(trig, OUTPUT);
  pinMode(eco, INPUT);
}
```

Solo queda el Void Loop para finalizar con la programación del nodo. En esta parte de código se obtiene la distancia entre el sensor y liquido del tanque, con este dato y las medidas del tanque se halló el nivel.

Para hallar la distancia en centímetros, el fabricante del sensor estableció la siguiente formula:

$$\frac{ancho\ del\ pulso\ (\mu s)}{58{,}2} = distancia\ (cm) \qquad (3)$$

En el código que se muestra en la Figura 108 está enfocado en la obtención del ancho del pulso y la aplicación de la formula anterior para poder obtener la distancia en centímetros. El funcionamiento no es muy complicado, primero se le envía al sensor un pulso de 10 µs, el sensor emite 8 pulsos de 40 KHz a través del transmisor (Trig), los pulsos rebotaran en liquido del tanque y serán recibidos por el receptor (Eco), quien generara un pulso de igual duración que el tiempo transcurrido entre la emisión y recepción del pulso.

Figura 108

Obtención del Lecturas del Sensor – Nodo Sensor – HC-SR04

```
void loop() {
  digitalWrite(trig, HIGH);
  delay(1);
  digitalWrite(trig, LOW);

  duracion = pulseIn(eco, HIGH);  //Recepcion del puslo
  distancia = duracion / 58.2;    //Calculo para hallar la distancia en cm
```

Después de conseguir la distancia, se procedió con el cálculo del nivel del tanque de agua. Como el tanque tiene una forma cilíndrica, se utilizó la fórmula para hallar el volumen de un cilindro.

$$V_t = \pi * r^2 * h \quad (4)$$

Reemplazando los datos del tanque en la formula anterior se puede hallar el volumen total del tanque.

$$V_t = \pi * 74{,}5^2 * 115$$

$$V_t = 2\,005\,152{,}693\ cm^3$$

O

$$V_t = 2{,}005\ m^3$$

Como el sensor se encuentra en la parte superior del tanque y nos proporciona la distancia entre él y el líquido, si sustituimos esa distancia en la formula, obtendremos el volumen vacío del tanque.

$$V_v = \pi * r^2 * distancia \quad (5)$$

Al restar el volumen total con el volumen vacío nos da como resultado el volumen lleno del tanque.

$$V_l = V_t - V_v \quad (6)$$

En la Figura 109 se puede apreciar cómo se realizó el cálculo de los tres volúmenes en el código.

Figura 109

Cálculo del Volumen – Nodo Sensor – HC-SR04

```
Vt = pi * pow(r, 2) * h;
Vv = pi * pow(r, 2) * distancia;
Vl = Vt - Vv;
```

Para lograr obtener el porcentaje del nivel del tanque, se utilizó la función map, justo como se ve en la Figura 110. Esta función permite convertir un valor entero de un rango de entrada en un valor correspondiente para otro rango de salida.

Figura 110

Conversión a Porcentaje – Nodo Sensor – HC-SR04

```
niveltanque = map(Vl, 0, Vt, 0, 100);
```

Se estableció los valores de las variables en la estructura del mensaje, justo como se muestra en la Figura 111. Este nodo tiene como ID el número dos.

Figura 111

Igualación de Lecturas a Variables – Nodo Sensor – HC-SR04

```
Invernadero.id = 2;
Invernadero.nivel = niveltanque;
```

Con la función if se creó una condición para solo enviar el mensaje a través del protocolo ESP-NOW cuando hayan cambiado las lecturas del sensor, tal como se visualiza en la Figura 112.

Figura 112

Envío de Lecturas – Nodo Sensor – HC-SR04

```
if (temperatura != LecturaAntiguaTemp || humedad != LecturaAntiguaHum) {
  Serial.println("La temperatura y/o humedad vario");

  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Invernadero, sizeof(Invernadero));
  if (result == ESP_OK) {
    Serial.println("Envio del mensaje exitoso");
  }
  else {
    Serial.println("Envio del mensaje fallido");
  }
}
```

Para finalizar, se igualó las lecturas a las variables para almacenar las lecturas antiguas, así como se observa en la Figura 113.

Figura 113

Igualación de Lecturas – Nodo Sensor – HC-SR04

```
LecturaAntiguaNivel = niveltanque;
```

3.6.6.4. Nodo Sensor – HW-080

El Nodo Sensor – HW-080 envía su ID y la humedad del suelo al Nodo Central, envía lecturas solo si hay una variación entre ellas y también lo hace a través del protocolo ESP-NOW.

De igual manera que en todos los nodos de sensor, primero se incluyeron las librerías que se utilizaron, tal cual se muestra en la Figura 114, solo las librerías utilizadas por el protocolo ESP-NOW son necesarias para este nodo.

Figura 114

Inclusión de Librerías – Nodo Sensor – HW-080

```
#include "esp_now.h"
#include "WiFi.h"
```

Se creo una variable y se asignó un pin para el sensor, así como se visualiza en la Figura 115.

Figura 115

Variable para el Sensor – Nodo Sensor – HW-080

```
int sensor = 33;
```

Del mismo modo que los nodos sensor anteriores, se creó una variable para el almacenamiento de la dirección MAC del Nodo Central, se crearon variables para almacenar las lecturas, se creó una estructura para almacenar los datos del mensaje, también una variable para almacenar la estructura y se definió la función de devolución de llamada, tal como se observa en la Figura 116.

Figura 116

Definición del Resto de Variables – Nodo Sensor – HW-080

```
uint8_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x88, 0xCE, 0xEC};

int LecturaAntiguaHumsue;

int humedadsuelo;

typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;

EstructuraMensaje Invernadero;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio fallido");
}
```

El Void Setup tiene el mismo funcionamiento que los nodos sensor anteriores, la única variación es la definición del modo de operación de la variable, justo como se ve en la Figura 117.

Figura 117

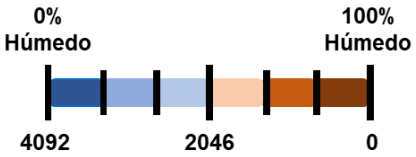
Void Setup – Nodo Sensor – HW-080

```
void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error de inicializacion del protocolo ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnDataSent);

  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("No se puedo agregar el par");
    return;
  }

  pinMode(sensor, INPUT);
}
```

Se continuo con el Void Loop, primero se obtuvo la medida del sensor, para ello se utilizó la función analogRead que nos permitió adquirir el valor analógico del sensor. La Figura 118 muestra los valores del umbral, van desde 4 092 cuando está en el aire o en un suelo muy seco a 0 cuando se encuentra sumergido en agua o en un suelo muy húmedo.

Figura 118

Valores de Humedad del Sensor – Nodo Sensor – HW-080



Después de obtener el valor analógico del sensor se utilizó la función map para conseguir el porcentaje de la humedad del suelo, como se muestra en la Figura 119.

Figura 119

Obtención de Lectura – Nodo Sensor – HW-080

```
humedadsuelo = map(analogRead(sensor), 4092, 0, 0, 100);
```

Se estableció los valores de las variables en la estructura del mensaje, tal como se visualiza en la Figura 120. Este nodo tiene como ID el número tres.

Figura 120

Igualación de Lecturas a Variables – Nodo Sensor – HW-080

```
Invernadero.id = 3;
Invernadero.humsue = humedadsuelo;
```

Utilizando la función if se creó una condición para solo enviar el mensaje mediante el protocolo ESP-NOW cuando hayan variado las lecturas del sensor, así como se observa en la Figura 121.

Figura 121

Envío de Lecturas – Nodo Sensor – HW-080

```
if (humedadsuelo != LecturaAntiguaHumsue) {
  Serial.println("La humedad del suelo vario");

  //Envio del mensaje a traves del protocolo ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Invernadero, sizeof(Invernadero));
  if (result == ESP_OK) {
    Serial.println("Envio del mensaje exitoso");
  }
  else {
    Serial.println("Envio del mensaje fallido");
  }
}
```

Para finalizar, se igualó las lecturas a las variables para almacenar las lecturas antiguas, tal cual se ve en la Figura 122.

Figura 122

Igualación de Lecturas – Nodo Sensor – HW-080

```
LecturaAntiguaHumsue = humedadsuelo;
```

3.7. Conexiones eléctricas del sistema IoT

El último paso para concluir con la implementación del sistema IoT son sus conexiones eléctricas. Las conexiones eléctricas deben respetar los pines que se asignaron en la programación de los nodos.

3.7.1. Nodo Central

La Tabla 14 muestra la asignación de los pines según la programación que se realizó en el Nodo Central.

Tabla 14

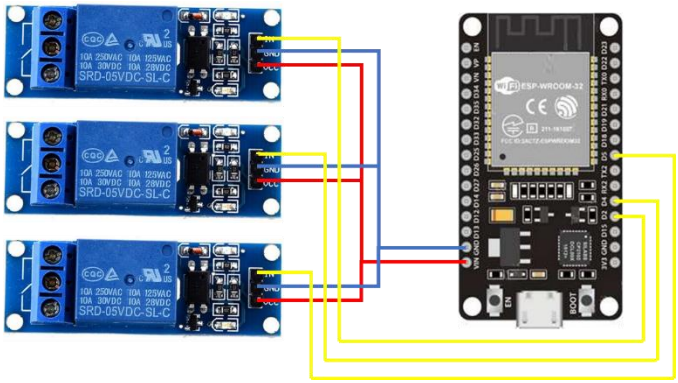
Pines de Conexión Eléctrica – Nodo Central

Pin del microcontrolador	Pin del Relé
GPIO2	IN (Ventilador)
GPIO4	IN (Calefactor)
GPIO5	IN (Bomba de agua)
VIN	V_{DC}
GND	GND

En la Figura 123 se puede observar las conexiones eléctricas desde el microcontrolador ESP32 hacia los módulos relé, respetando los pines designaciones en la Tabla 14.

Figura 123

Diagrama de Conexión Eléctrica – Nodo Central



3.7.2. Nodo Sensor – DHT22

En la Tabla 15 se puede observar la asignación de los pines según la programación del Nodo Sensor – DHT22.

Tabla 15

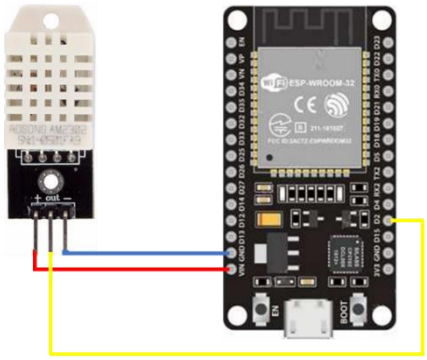
Pines de Conexión Eléctrica – Nodo Sensor – DHT22

Pin del microcontrolador	Pin del sensor
GPIO2	DATA
VIN	V_{DC}
GND	GND

En la Figura 124, puede ver las conexiones eléctricas del microcontrolador ESP32 a el sensor DHT22, de acuerdo con la asignación de pines en la Tabla 15.

Figura 124

Diagrama de Conexión Eléctrica – Nodo Sensor – DHT22



3.7.3. Nodo Sensor – HC-SR04

En la Tabla 16 se puede visualizar la asignación de los pines según la programación que se hizo en el Nodo Sensor – HC-SR04.

Tabla 16

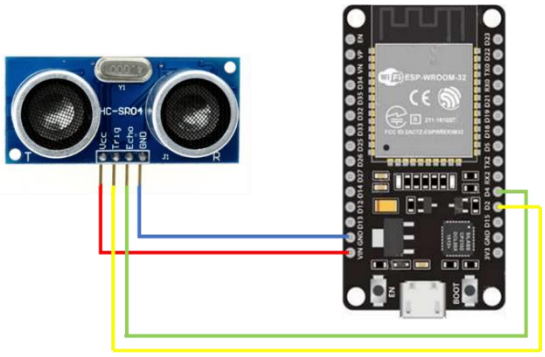
Pines de Conexión Eléctrica – Nodo Sensor – HC-SR04

Pin del microcontrolador	Pin del sensor
GPIO2	Trig
GPIO4	Echo
VIN	V_{DC}
GND	GND

En la Figura 125, se ven las conexiones eléctricas del microcontrolador ESP32 hacia el sensor HC-SR04, de acuerdo con la asignación de pines en la Tabla 16.

Figura 125

Diagrama de Conexión Eléctrica – Nodo Sensor – HC-SR04



3.7.4. Nodo Sensor – HW-080

La Tabla 17 muestra la asignación de los pines según la programación que se realizó en el Nodo Central.

Tabla 17

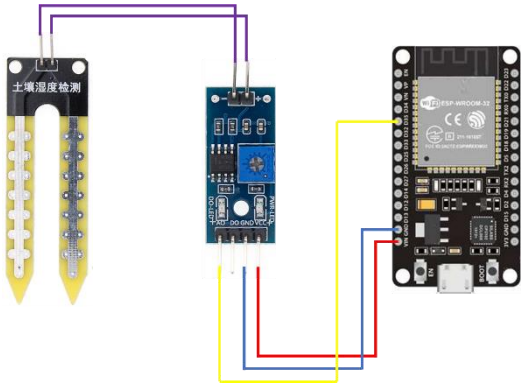
Pines de Conexión Eléctrica – Nodo Sensor – HW-080

Pin del microcontrolador	Pin del sensor
GPIO33	AO
VIN	V_{DC}
GND	GND

En la Figura 126 se puede observar las conexiones eléctricas desde el microcontrolador ESP32 hacia los módulos relé, respetando los pines asignados en la Tabla 17.

Figura 126

Diagrama de Conexión Eléctrica – Nodo Sensor – HW-080



CAPÍTULO IV: RESULTADOS

4.1. Prueba de comunicación entre nodos del sistema IoT

En esta parte del capítulo se puso a prueba la comunicación entre nodos sensor y el nodo central. Se utilizó como referencia la distancia máxima que pueden tener los nodos dentro del invernadero. Para que el nodo central y la laptop puedan tener acceso a internet se utilizó la función Punto de acceso móvil del celular.

Utilizando el teorema de Pitágoras se halló la distancia máxima que habría, en el peor de los casos, entre nodos.

$$c^2 = a^2 + b^2 \tag{7}$$

$$c = \sqrt{a^2 + b^2}$$

Reemplazando con las medidas del invernadero.

$$c = \sqrt{(6\ m)^2 + (8\ m)^2}$$

$$c = \sqrt{36\ m^2 + 64\ m^2}$$

$$c = \sqrt{100\ m^2}$$

$$c = 10\ m$$

Después de haber hallado la distancia máxima, se escogieron al azar tres puntos para comprobar el alcance de la comunicación entre el nodo central y los nodos sensor. Como punto de partida se utilizó el nodo central, se colocó el Nodo Sensor – DHT22 a un metro, el Nodo Sensor – HC-SR04 estuvo a 10 metros y, por último, el Nodo Sensor – HW-080 se ubicó a 15 metro, se utilizó un flexómetro para medir las distancias.

A falta de infraestructura y equipo necesario para realizar las pruebas dentro de un invernadero, se utilizó un ambiente que cuente con un poco de vegetación para poder poner a prueba la comunicación entre nodos.

En la Figura 127 se puede observar cómo se situaron todos los nodos.

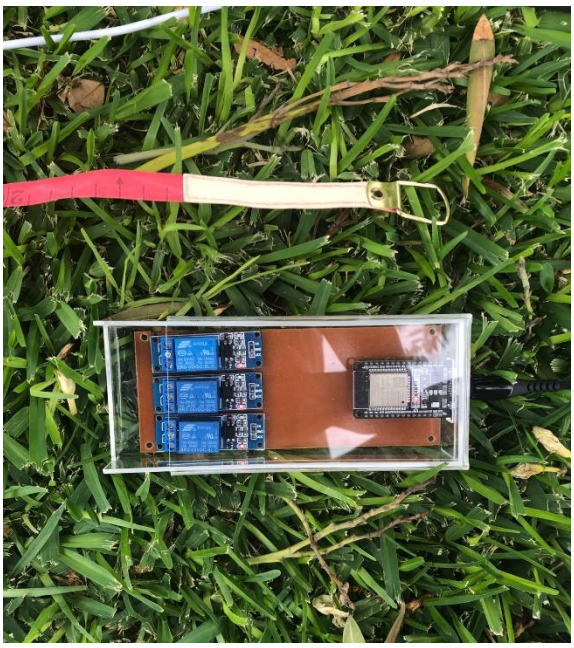
Figura 127

Vista Panorámica de la Situación de Nodos

En la Figura 128 se observa el Nodo Central ubicado en el punto de inicio del flexómetro.

Figura 128

Ubicación del Nodo Central



La Figura 129 muestra el Nodo Sensor – DHT22 en el metro 1 del flexómetro.

Figura 129

Ubicación del Nodo Sensor – DHT22

En la Figura 130 se visualiza en Nodo Sensor – HC-SR04 en el metro 10 del flexómetro.

Figura 130

Ubicación del Nodo Sensor – HC-SR04

En la Figura 131 se ve el Nodo Sensor – HW-080 en el metro 15 del flexómetro.

Figura 131

Ubicación del Nodo Sensor – HW-080

Como se observa en la Figura 132, el Nodo Sensor – HW-080 se ubicó en medio de las plantas, tratando de simular la vegetación.

Figura 132

Ubicación Entre la Vegetación del Nodo Sensor – HW-080

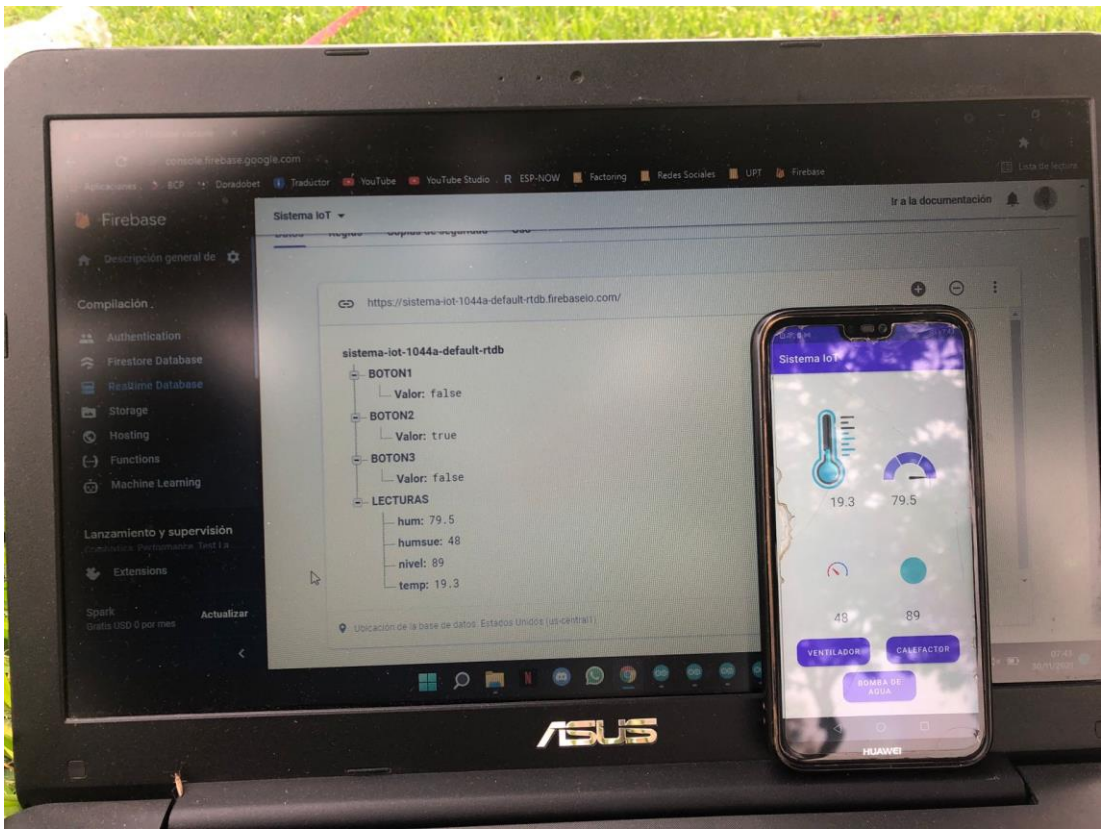
Luego de haber acomodado los nodos, se procedió a encenderlos, como el Nodo Central y el Nodo Sensor – DHT22 están cerca de la laptop, se conectaron a ella, los nodos restantes se alimentaron con un power bank cada uno.

Acto seguido, la base de datos y la aplicación móvil empezaron a recibir las mediciones que captaban los sensores.

En la Figura 133 se puede apreciar la recepción de datos en ambos programas, en la computadora tenemos abierta la base de datos, mientras que en el teléfono inteligente la aplicación móvil.

Figura 133

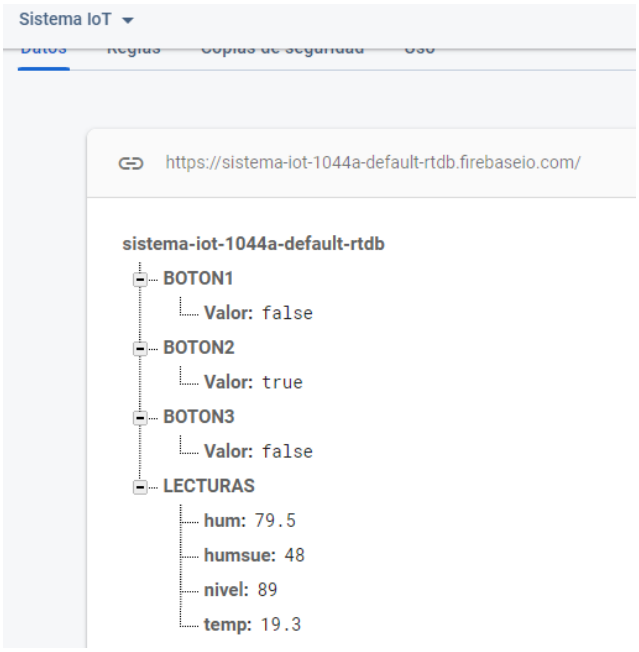
Prueba de Comunicación – Base de Datos y Aplicación Móvil



En la Figura 134 se puede apreciar una captura de pantalla o screenshot de la base de datos.

Figura 134

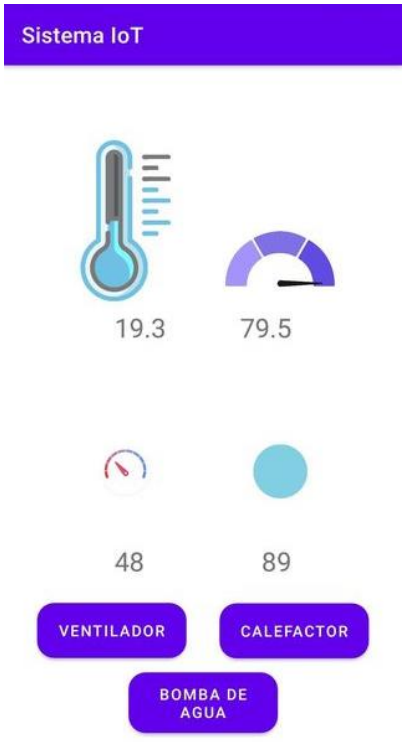
Prueba de Comunicación – Base de Datos



En la Figura 135 se puede observar una captura de pantalla o screenshot de la aplicación móvil.

Figura 135

Prueba de Comunicación – Aplicación Móvil

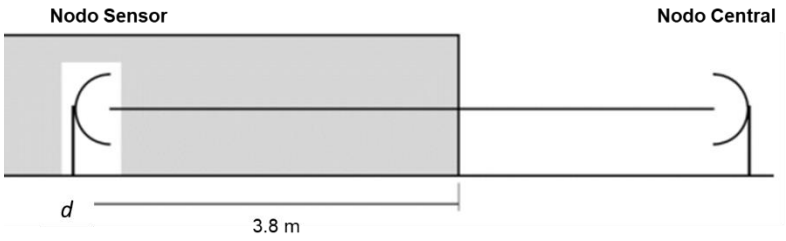


4.2. Cálculo de la atenuación por vegetación en un radioenlace

Se calculó la atenuación excesiva que tuvo la prueba de comunicación del sistema IoT, específicamente del Nodo Sensor – HW-080 al Nodo Central, tal como se muestra en la Figura 136. El nodo sensor actúa como transmisor, mientras que el nodo central como receptor.

Figura 136

Trayecto del Radioenlace entre el Nodo Sensor – HW-080 y el Nodo Central



Para poder calcular la atenuación por vegetación en un radioenlace se utilizó la fórmula de atenuación excesiva.

$$A_{ev} = A_m\left(1 - e^{\frac{-d*\gamma}{A_m}}\right)$$

Antes de aplicar la formula, se halló todas las variables. La longitud del trayecto dentro de la zona boscosa (d) se obtuvo midiendo la distancia de la vegetación. La atenuación específica para trayectos en vegetación muy cortos (γ) se extrajo de la tabla especifica en zona boscosa. La atenuación máxima cuando un terminal esta adentro de una zona de vegetación de un tipo y profundidad específicos (A_m) se halló resolviendo su fórmula con los datos que brinda la Recomendación ITU-R P.833-10.

4.2.1. Longitud del trayecto de la zona boscosa

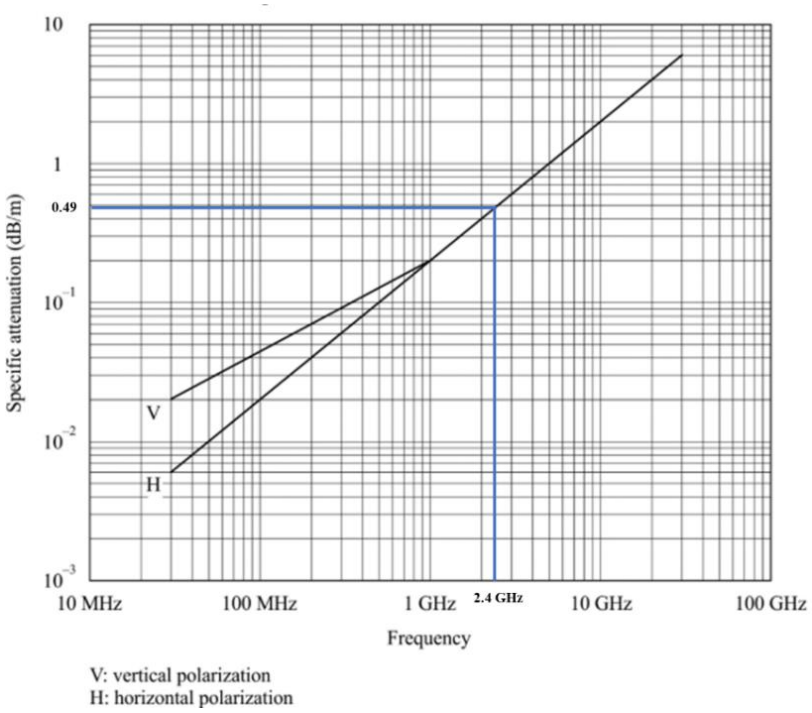
Para obtener d solo se midió la distancia de la vegetación desde el Nodo Sensor – HW-080, el cual nos dio un valor de 3,8 m.

4.2.2. Atenuación específica para trayectos en vegetación muy cortos

La comunicación entre nodos se da en la banda de 2,4 GHz, ubicando dicha frecuencia en la tabla de atenuación especifica en zona boscosa, justo como se observa en la Figura 137, pudimos extraer el valor de γ.

Figura 137

Frecuencia 2.4 GHz en Atenuación Especifica en Zona Boscosa



$$\gamma = 0{,}49\ dB/m$$

4.2.3. Atenuación máxima cuando un terminal esta adentro de una zona de vegetación de un tipo y profundidad específicos

La Rec. ITU-R P.833-10 realizo varios experimentos para poder proporcionar los siguientes datos:

$$A_1 = 1{,}15$$

$$\alpha = 0{,}43$$

El protocolo de comunicación trabaja en una frecuencia de 2,4 GHz, solo se tuvo que realizar la conversión a MHz.

$$f = 2{,}4\ GHz = 2\,400\ MHz$$

Después de haber hallado todas las variables, se procedió a resolver las ecuación brindada por la Rec. ITU-R P.833-10.

$$A_m = A_1 f^{\alpha}$$

$$A_m = 1{,}15(2\,400\ MHz)^{0.43}$$

$$A_m = 32{,}67\ dB$$

4.2.4. Atenuación excesiva

Para finalizar, se reemplazaron todos los datos hallados en la ecuación de atenuación excesiva para obtener la atenuación por vegetación.

$$A_{ev} = A_m\left(1 - e^{\frac{-d*\gamma}{A_m}}\right)$$

$$A_{ev} = 32{,}67\left(1 - e^{\frac{(-3{,}9m)*(0{,}49dB/m)}{32{,}67}}\right)$$

$$A_{ev} = 1{,}86\ dB$$

La distancia y la vegetación que se presenta en la prueba de comunicación no es lo suficientemente alta como para proporcionar una atenuación significativa que interrumpa la comunicación entre los nodos, esto se demostró hallando la atenuación excesiva.

4.3. Prueba de funcionamiento continuo del sistema IoT

La prueba de funcionamiento del prototipo implementado se realizó por tres días seguidos, esto con el propósito de asegurar el funcionamiento continuo del sistema IoT. Para realizar esta prueba solo se utilizaron tres nodos, el Nodo Central, para recibir las mediciones de los nodos sensor y subirlas a la base de datos para poder visualizarlas en la aplicación móvil, y el Nodo Sensor – DHT22 y el Nodo Sensor – HW-080, estos dos nodos miden los parámetros climáticos y envían la información al nodo central. Se decidió no usar el Nodo Sensor – HC-SR04 porque no mide una condición climática y, por lo tanto, su parámetro no variaría constantemente.

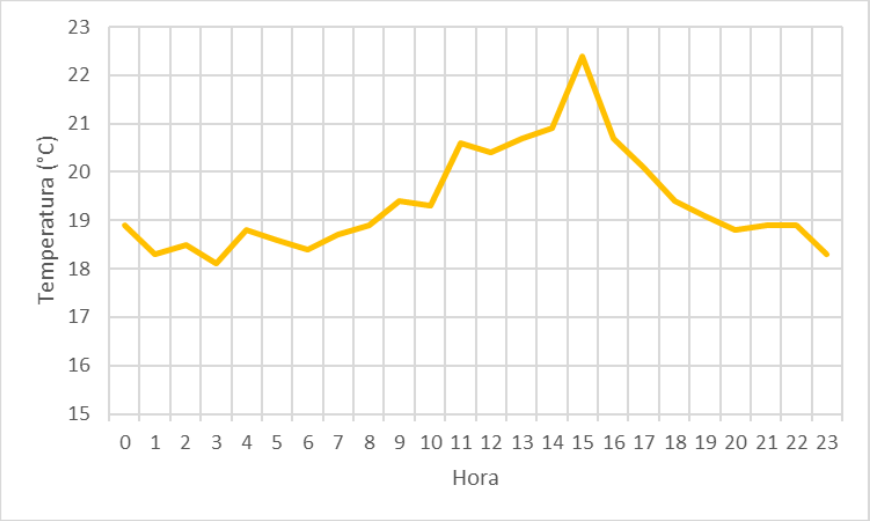
La prueba se realizó en un ambiente con vegetación para tratar de simular el cultivo de lechugas. Se tomaron lecturas de temperatura, humedad relativa y humedad del suelo cada una hora, desde las 00:00 hasta las 23:00, en los tres días se rego la vegetación a las 08:00, después de tomar la lectura, esto se realizó con el fin de poder observar el comportamiento de los parámetros climáticos.

4.3.1. Prueba de funcionamiento – Día 01

La Figura 138 muestra el comportamiento de la temperatura el primer día. Se puede observar que, en la madrugada, de 00:00 a 06:00 para ser más específicos, es cuando se alcanza la temperatura más fría y desde las 09:00 hasta las 17:00 se obtiene la temperatura más alta de todo el día.

Figura 138

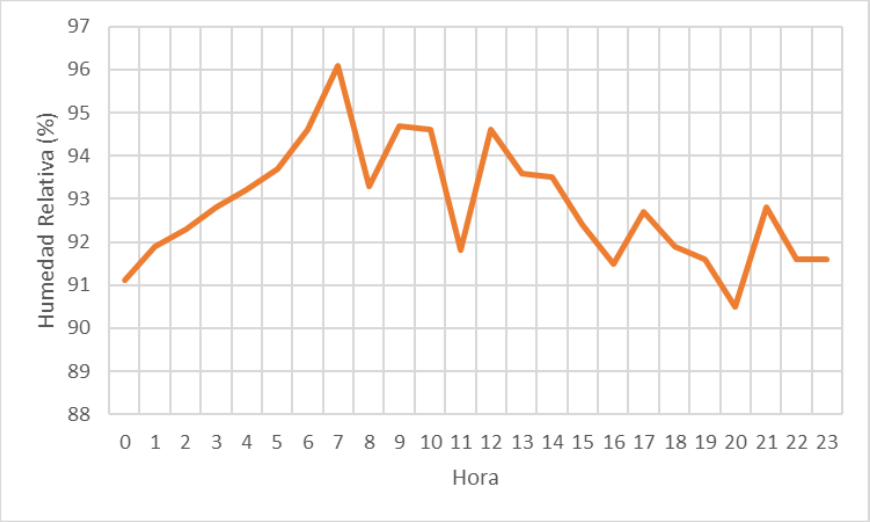
Gráfica del Comportamiento de Temperatura – Día 01



En la Figura 139 se visualiza las medidas conseguidas de la humedad relativa el primer día. El porcentaje de la humedad a inicios del día es bajo, a las 07:00 obtiene la humedad más alta del día, después de eso empieza a disminuir.

Figura 139

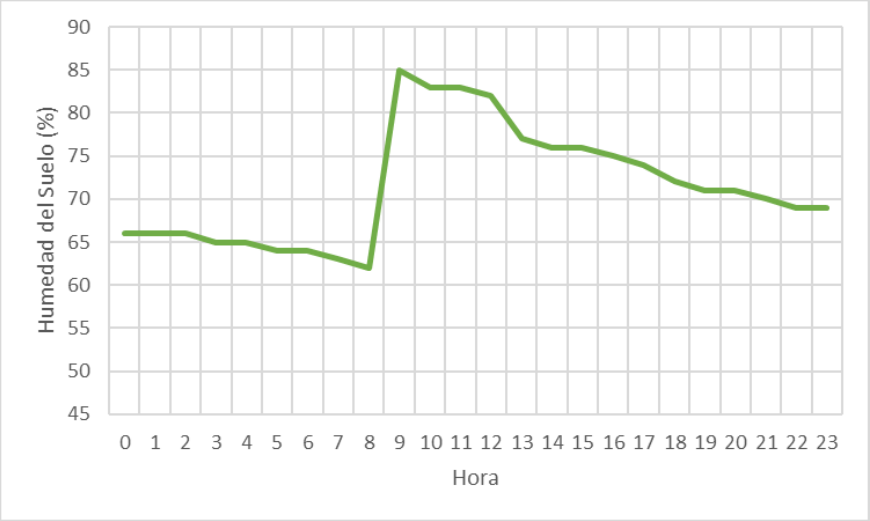
Gráfica del Comportamiento de Humedad Relativa – Día 01



La Figura 140 muestra la variación de la humedad del suelo el primer día. Empieza con una humedad baja, esto se mantiene hasta que se riega la vegetación, a las 08:00, después del riego se obtiene el pico más alto del día, posteriormente, la humedad del suelo empieza a decrecer lentamente hasta el siguiente riego.

Figura 140

Gráfica del Comportamiento de Humedad del Suelo – Día 01

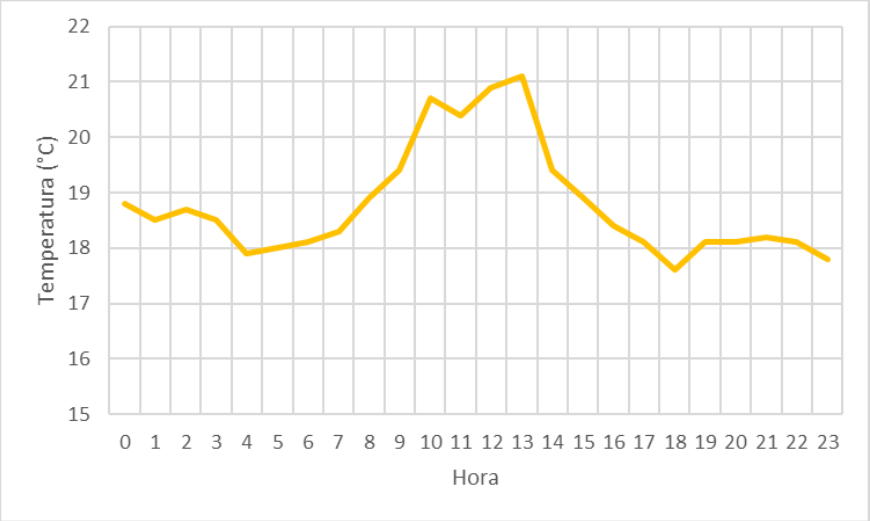


4.3.2. Prueba de funcionamiento – Día 02

En la Figura 141 se observa el comportamiento de la temperatura durante el segundo día. Se puede ver que de 08:00 a 15:00 hay un incremento de temperatura, en comparación a las demás horas, llegando a ser la más alta del día.

Figura 141

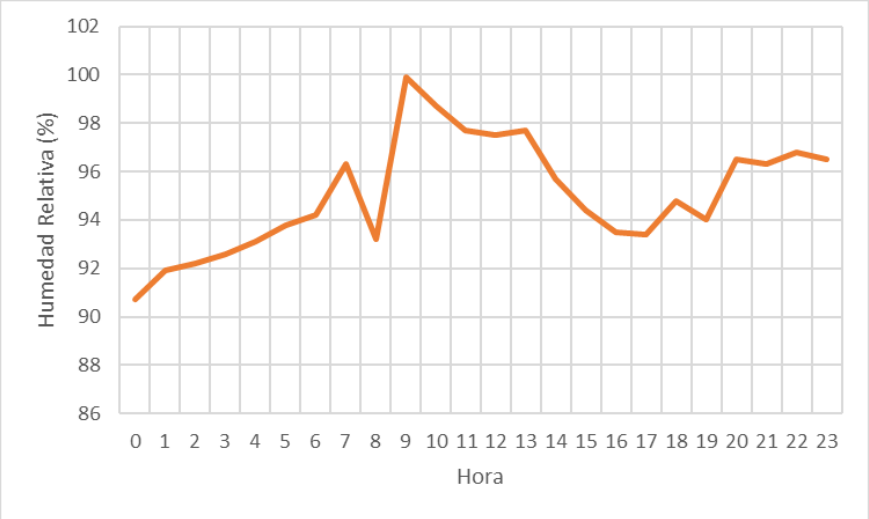
Gráfica del Comportamiento de Temperatura – Día 02



La Figura 142 muestra las medidas de humedad relativa obtenidas el segundo día. La humedad al comienzo del día es baja, a las 09:00 es cuando se logra la humedad más alta del día, y luego disminuye gradualmente hasta las 17:00, después de eso hay un pequeño incremento.

Figura 142

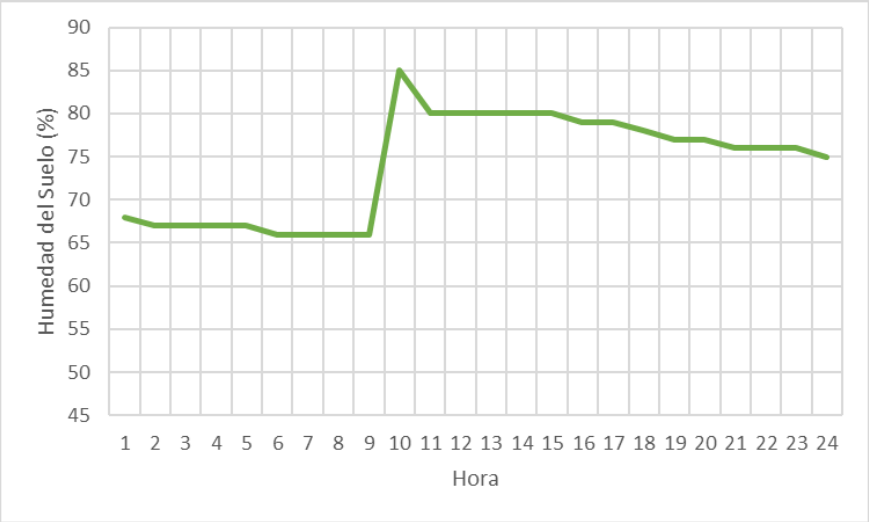
Gráfica del Comportamiento de Humedad Relativa – Día 02



La Figura 143 muestra el cambio en la humedad del suelo en el segundo día. Comienza con poca humedad, continua así hasta que la vegetación ha sido regada, a las 08:00, después del riego alcanza el pico más alto del día, luego la humedad del suelo comienza a disminuir lentamente hasta el próximo riego.

Figura 143

Gráfica del Comportamiento de Humedad del Suelo – Día 02

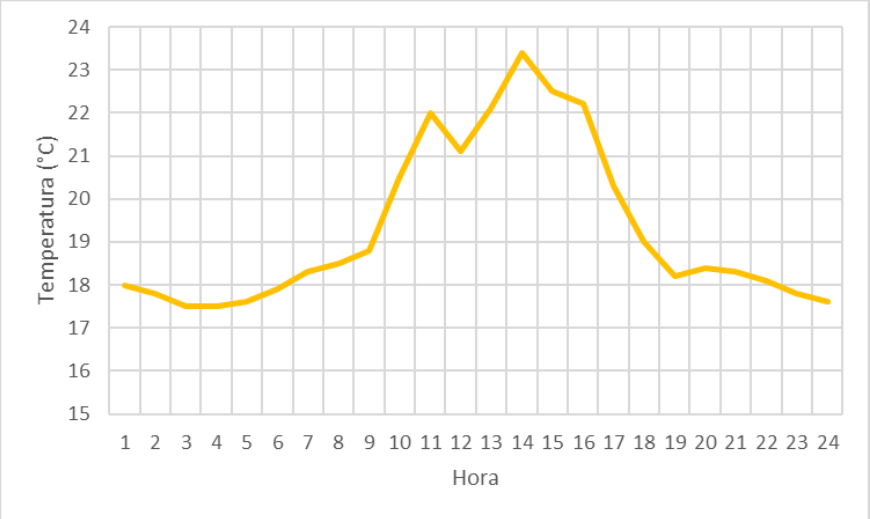


4.3.3. Prueba de funcionamiento – Día 03

En la Figura 144 muestra el cambio de la temperatura durante el tercer día. Se puede observar que desde las 09:00 hasta las 18:00 la temperatura aumenta con respecto a otras horas, convirtiéndose en la más alta del día.

Figura 144

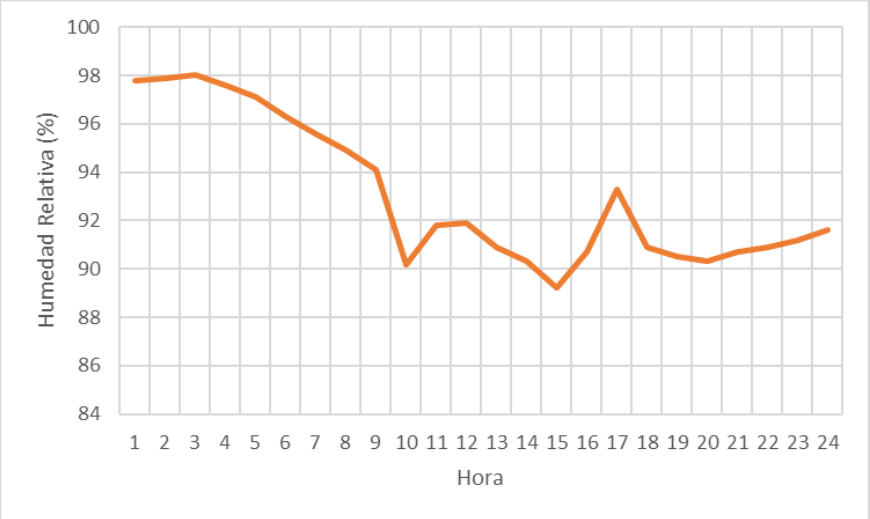
Gráfica del Comportamiento de Temperatura – Día 03



La Figura 145 describe el comportamiento de la humedad relativa en el tercer día. Comenzó con una humedad alta, pero mientras transcurría el día la humedad iba disminuyendo.

Figura 145

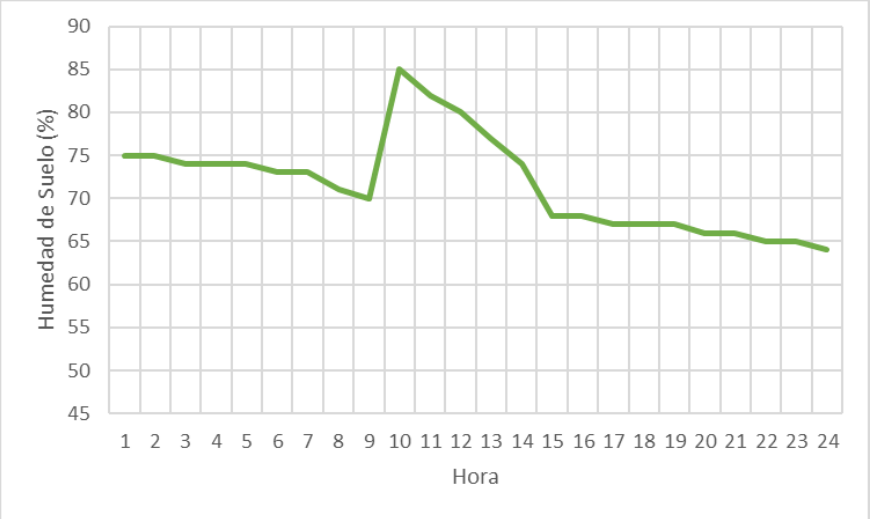
Gráfica del Comportamiento de Humedad Relativa – Día 03



La Figura 146 muestra el cambio en la humedad del suelo el tercer día. Comenzó con una humedad regular, fue disminuyendo poco a poco hasta que la vegetación fue regada, a las 08:00, después alcanzo su pico más alto del día, luego la humedad del suelo comenzó a disminuir lentamente.

Figura 146

Gráfica del Comportamiento de Humedad del Suelo – Día 03



4.4. Prueba de cobertura del protocolo ESP-NOW

Se realizó la prueba de cobertura para obtener el alcance máximo del protocolo de comunicación ESP-NOW, la prueba se llevó a cabo con línea de vista.

Para ejecutar esta prueba se utilizó el Nodo Central y el Nodo Sensor – DHT22, se puso en funcionamiento ambos nodos, se enviaron lecturas desde el nodo sensor hacia el nodo central. Para ello se distribuyó a los nodos con una distancia inicial de 50 m, después se prosiguió a colocarlo a 100 m, se continuó aumentando la distancia en 50 m hasta que se interrumpió la comunicación.

La comunicación fue constante hasta la distancia de 300 m, pero cuando se llegó 350 m, la comunicación cesó, se procedió a ir retrocediendo metro por metro hasta que se restableciera la conexión, lo cual sucedió a llegar a 345 m.

Los resultados obtenidos de esta prueba fueron eficientes, a continuación, se muestra la Tabla 18 que corresponde a la distancia y la cobertura del protocolo ESP-NOW.

Tabla 18

Prueba de Cobertura del Protocolo ESP-NOW

Distancia	Cobertura
50 m	Si
100 m	Si
150 m	Si
200 m	Si
250 m	Si
300 m	Si
345 m	Si
350 m	No

En la Figura 147 muestra el terreno donde se efectuó la prueba de cobertura, se puede observar claramente la línea de vista.

Figura 147

Distancias de la Prueba de Cobertura del Protocolo ESP-NOW

4.5. Resultados

En la Figura 148 se muestra el sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero.

Figura 148

Sistema IoT

El sistema IoT fue puesto a prueba con el propósito de comprobar su funcionamiento y confiabilidad al momento de mantener los parámetros climáticos considerados en su rango optimo.

El proyecto actual muestra un prototipo completo, por lo que se puede concluir que se lograron los objetivos originalmente planteados.

El sistema IoT cumple con las siguientes funciones.

- Recolecta la información de los parámetros climáticos considerados y los envía inmediatamente a la base de datos.
- Se puede visualizar la información de la base de datos en la aplicación móvil.
- Controla los actuadores desde la aplicación móvil.
- Al detectar la temperatura, humedad relativa o humedad del suelo fuera de su rango optimo automáticamente enciende o apaga el actuador adecuado.
- Distingue el horario diurno del horario nocturno.

CAPÍTULO V: DISCUSIÓN

La hipótesis planteaba que: “Mediante el diseño de un sistema IoT se monitorea y controla los parámetros del cultivo de lechugas en un invernadero”, y efectivamente, los resultado muestran que por intermedio de un sistema IoT conformado por un nodo central y tres nodos sensor, los cuales se comunican mediante el protocolo ESP-NOW, se logra realizar el monitoreo y control de la temperatura, la humedad relativa, la humedad del suelo y el nivel del tanque de agua. Adicionalmente, el sistema permite enviar información a una base de datos para después mostrarla al usuario en tiempo real a través de una aplicación móvil.

El sistema IoT desarrollado es similar en cuanto a los resultados hallados por Hernández (2019) en un sistema cableado, utilizando el sensor de temperatura y humedad relativa DHT22, el sensor de humedad de suelo YL69, el sensor ultrasónico HC-SR04, el sensor de pH y una fotorresistencia LDR para luminosidad, concluyó que el sistema diseñado permite controlar los parámetros descritos anteriormente. Igualmente, Molanes (2019) y Reyna (2015) desarrollaron un sistema cableado para el monitoreo y control de sus parámetros climáticos considerados.

Por otra parte, el protocolo de comunicación ESP-NOW demuestra tener una superioridad frente a la comunicación por cable, conforme a las pruebas realizadas corrobora tener un alcance hasta de 345 m, permitiendo la comunicación entre los nodos sensor y nodo central sin la necesidad de una conexión por cable.

El método de control del sistema es similar al utilizado por Molanes (2019), Reyna (2015) y Hernández (2019), se concluyó que utilizando un programa de control usando lógica On/Off los actuadores se activan y desactivan de acuerdo con lo establecido.

La investigación se limita al monitoreo y control de los parámetros considerados y al tipo de cultivo seleccionado, quedando como una futura línea de trabajo, añadir parámetros para un mejor monitoreo y control del cultivo, como la medición de pH del agua o un sistema de fumigación, y un diseño multifuncional para distintos tipos de cultivo

Para concluir, es importante determinar si los parámetros controlados mejoran la eficiencia del cultivo de lechugas o la forma en la que afectan su productividad.

CONCLUSIONES

Se cumplió con el objetivo principal de la tesis. Se diseño un sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero, este sistema nos permite monitorear desde cualquier lugar los parámetros establecidos desde un principio, también nos permite controlar los actuadores. El sistema mantiene automáticamente en su rango optimo la temperatura, humedad relativa y humedad del suelo.

Se determinaron los niveles óptimos de la temperatura, humedad relativa y humedad del suelo que debe tener el invernadero para el crecimiento de la lechuga.

Se seleccionó el microcontrolador y los sensores que nos ayudaron en la adquisición de datos y control del invernadero.

Se implemento un prototipo completo del sistema IoT, este prototipo ayudo en la realización de las pruebas de funcionamiento.

RECOMENDACIONES

En investigaciones posteriores, pueden agregarse especificaciones técnicas al producto final que no han sido consideradas en el alcance del presente trabajo de investigación como el monitoreo y control del sistema IoT desde una computadora mediante una aplicación web Firebase o considerar un diseño multifuncional para distintos tipos de cultivo.

En caso se desee desarrollar un producto comercial, el investigador podría realizar pruebas utilizando sensores más sensibles, midiendo el costo – beneficio del usuario final a fin de determinar su eficacia.

Se propone realizar el estudio de otras tecnologías que se apliquen en la agricultura de precisión en un ambiente de estudio similar al propuesto para medir su impacto y posibles bondades en comparación con la desarrollada en la presente investigación.

Se propone a la Dirección de la Escuela Profesional de Ingeniería Electrónica, incentivar el diseño de sistemas IoT para resolver situaciones problemáticas que puedan observarse a nivel regional a través de la conformación de grupos de investigación conformado por profesores y estudiantes, lo que permitiría acercar a la Escuela hacia la comunidad.

REFERENCIAS BIBLIOGRÁFICAS

Alciatore, D. (2008). *Introducción a la Mecatrónica y los sistemas de medición.*

Arenas, D., Silva, B. y Winchester, L. (2021). *Introducción a los sistemas de monitoreo y evaluación*. Instituto Latinoamericano y del Caribe de Planificación Económica y Social (ILPES) de la Comisión Económica para América Latina y el Caribe, (CEPAL).

Cámara de Comercio de Bogotá. (2015). *Manual lechuga.*

Cárdenaz, G., Ramírez, H., Pulido, S., Gómez de Enciso, C., Henríquez, S., Sáchez, J., Forero, C., Benavides, M., Mora, H., Herrera, A., Escobar, H., Sánchez, G., Flores, L., González, G., Wyckhuys, K., Salamanca, C., Zamudio, A., Jiménez, J., Gil, R.,… Ligarreto, G. (2012). *Manual para el cultivo de Hortalizas.*

Enriquez, J. G., y Casas, S. I. (2013). *Usabilidad en aplicaciones móviles*. Informes Científicos Técnicos-UNPA, 5(2), 25-47.

Espressif Systems (2021). *ESP32 Series Datasheet v.3.5.*

Fueyo, M., Arrieta, A. y Feito, I. (s.f.). *Producción de lechuga* (p. 3).

González, A. (2017). *IoT: Dispositivos, tecnologías de transporte y aplicaciones* (p. 5).

Guarella, J., Heredia, J., Rodríguez, L., y Bagatto, I. (2011). *Sensores y actuadores en motores* (pp. 3-18). Universidad Nacional de la Plata.

Hernández Sanz, E. M. (2019). *Desarrollo de un sistema de monitorización y control de un invernadero aplicando tecnología IoT.*

Iglesias, N. (2006). *Producción de hortalizas bajo cubierta: estructuras y manejo de cultivo para la Patagonia Norte*. INTA (Instituto Nacional de Tecnología Agropecuaria).

Marqués, M. (2011). *Bases de datos* (p. 2). Universitat Jaume I.

Martín, A., Chávez, S. B., Rodríguez, N. R., Valenzuela, A., y Murazzo, M. A. (2013). *Bases de datos NoSQL en Cloud Computing* (p. 166).

Martinez, R. (2017). *Comparativa y estudio de plataformas IoT* (p. 6). Universitat Politècnica de Catalunya.

Molanes Miovich, C. (2019). *Diseño e Implementación de un Sistema Electrónico de Control Basado en FPGA, que Optimice las Condiciones Climáticas de un Invernadero para el Cultivo de Tomate en la Ciudad de Tacna, 2017.*

Mora, H., y Rosas, J. (2019). *Diseño, desarrollo e implementación de una red de sensores inalámbricos (WSN) para el control, monitoreo y toma de decisiones aplicado en la agricultura de precisión basado en internet de las cosas (IoT). – Caso de estudio cultivo de frijol* (Tesis de pregrado). Universidad Ricardo Palma, Lima.

Palma, C., y Velasquez, C. (2019). *Framework para aplicaciones con base de datos relacional orientado a desarrolladores de software*. Universidad Ricardo Palma.

Pasic, R., Kuzmanov, I., y Atanasovski, K. (2020). *Espressif ESP32 development board in WiFi station communication mode.* Temel.

Recommendation ITU-R P.833-10. (2021). *Attenuation in vegetation.*

Reyna Huamán, C. E. (2015). *Sistema automatizado para el monitoreo y control de humedad en un invernadero.*

Rose, K., Eldridge, S. y Chapin, L. (2015). *La Internet de las Cosas – Una breve reseña* (p. 15).

Turmero, P. (2016). *Principios de los Sistemas de Control*.

Van Velsen, L., Beaujean, D. J., & van Gemert-Pijnen, J. E. (2013). Why mobile health app overload drives us crazy, and how to restore the sanity. *BMC medical informatics and decision making*.

ANEXOS

Anexo 1. Matriz de consistencia

Problemática	Formulación de problema	Justificación	Objetivo	Hipótesis	Variables
Uno de los problemas que tiene la horticultura es la variación del clima en las diferentes estaciones a lo largo del año. Las heladas alteran la temperatura y la humedad, estos dos parámetros climáticos son fundamentales para el cuidado de cualquier cultivo.	¿El diseño de un sistema IoT permitirá el monitoreo y control del cultivo de lechugas en un invernadero?	Las heladas es el problema que tienen los huertos para poder aumentar la cantidad de lechugas producidas por año, pero implementando este sistema se puede optimizar y mejorar el cultivo de lechugas. La aplicación móvil facilitará la visualización de los datos que recolecten los sensores que se encontraran dentro del invernadero.	Diseñar un sistema IoT para el monitoreo y control del cultivo de lechugas en un invernadero.	Mediante el diseño de un sistema IoT se monitorea y controla los parámetros del cultivo de lechugas en un invernadero.	Variable independiente: • Sistema IoT para el monitoreo y control. Variable dependiente: • Monitoreo y control de los parámetros del cultivo de lechugas en un invernadero.

Anexo 2. Operacionalización de variables

Variable	Tipo de Variable	Definición Conceptual	Dimensiones	Indicadores	Instrumentos
Sistema IoT para el monitoreo y control.	Variable independiente	El sistema de monitoreo mide la variable en tiempo real y manda una señal al sistema de control para realizar una determinada acción si esta fuera necesaria.	Sistema de monitoreo	Aplicación móvil	ESP32
			Sistema de control	Actuadores	Ventilador Calefactor Sistema de riego
Monitoreo y control de los parámetros del cultivo de lechugas en un invernadero	Variable dependiente	Parámetros de cultivo: • Temperatura: Magnitud física que refleja la cantidad de calor del invernadero. • Humedad relativa: Cantidad de agua en el aire del invernadero en forma de vapor. • Humedad del suelo: Cantidad de agua por volumen de tierra que hay en un cultivo.	Temperatura	Temperatura (°C)	DHT22
			Humedad	Humedad relativa (%)	DHT22
				Humedad del suelo (%)	HW-080

Anexo 3. Código del programa del sistema IoT – Nodo Central

```
//Librerias
#include "esp_now.h"
#include "WiFi.h"
#include "FirebaseESP32.h"
#include "ESP32Time.h"

//SSID y contraseña de la Red
const char* ssid = "WLAN - IoT";
const char* password = "WLAN-IoT";

//URL y Secreto de la Base de la Base de Datos
#define URL "sistema-iot-1044a-default-rtdb.firebaseio.com"
#define secreto "STaotDwJkS6Zd3yHFIzQ9PFBBg2U6Mhl8ebattoY"

//Configuracion de hora y servidor NTP
const char* ntpServer = "pool.ntp.org";     //Servidor NTP
const long gmtOffset_sec = -5*3600;         //Desplazamiento GMT (Perú = -5)
const int daylightOffset_sec = 0;           //Compensacion de luz diurna

//Objeto tipo ESP32Time
ESP32Time NTP;

//Variables para el servidor NTP
String horaNTP;
int hora;

//Variables de medicion
float temperatura;
float humedad;
int humedadsuelo;
float niveltanque;

//Variable para los actuadores
int ventilador = 2;
int calefactor = 4;
int bomba = 5;

//Variables para el estado de los actuadores
boolean encendido = true;
boolean apagado = false;

//Objeto JSON
FirebaseData myFireBaseData;
FirebaseJson myJson;
FirebaseJsonData myJsonData;

String estados, estado1, estado2, estado3;

//Estructura del mensaje
typedef struct EstructuraMensaje {
   int id;          //Numero de placa
   float temp;      //Temperatura
   float hum;       //Humedad
```

```
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;

//Variable para almacenar los valores de los mensajes
EstructuraMensaje Invernadero;

//Estructura para contener las lecturas de cada placa
EstructuraMensaje placa1;
EstructuraMensaje placa2;
EstructuraMensaje placa3;

//Matriz para contener todas las estructuras
EstructuraMensaje boardsStruct[3] = {placa1, placa2, placa3};

//Funcion de devolucion de llamada de recepcion - OnDataRecv()
  //Se llamará cuando se reciban datos nuevos
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
  char macStr[18];
  Serial.print("Mensaje recibido de: ");
  snprintf(macStr,          sizeof(macStr),          "%02x:%02x:%02x:%02x:%02x:%02x",
mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  Serial.println(macStr);

  memcpy(&Invernadero, incomingData, sizeof(Invernadero));
  Serial.printf("Placa N°: %u --- Tamaño del mensaje: %u bytes\n", Invernadero.id,
len);
  //Actualizamos las estructuras con los nuevos datos
  boardsStruct[Invernadero.id-1].temp = Invernadero.temp;
  boardsStruct[Invernadero.id-1].hum = Invernadero.hum;
  boardsStruct[Invernadero.id-1].humsue = Invernadero.humsue;
  boardsStruct[Invernadero.id-1].nivel = Invernadero.nivel;

  Serial.printf("Temperatura: %4.2f \n", boardsStruct[Invernadero.id-1].temp);
  Serial.printf("Humedad: %4.2f \n", boardsStruct[Invernadero.id-1].hum);
  Serial.printf("Humedad del suelo: %d \n", boardsStruct[Invernadero.id-1].humsue);
  Serial.printf("Nivel: %4.2f \n", boardsStruct[Invernadero.id-1].nivel);

  Serial.println();
}

void setup() {
  //Inicializacion del Monitor Serie
  Serial.begin(115200);

  //Configuracion como un Punto de Acceso (Access Point) y una Estacion WiFi
  WiFi.mode(WIFI_AP_STA);

  //Conexion a la red
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
```

```
  Serial.print("Canal WiFi: ");
  Serial.println(WiFi.channel());

  //Conexion a Firebase
  Firebase.begin(URL, secreto);
  Firebase.reconnectWiFi(true);   //Reconexion automatica a la base de datos
  Serial.println("Conexion con la base de datos exitosa");

  //Inicializacion del protocolo ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error de inicializacion del protocolo ESP-NOW");
    return;
  }

  //Registro de la funcion de devolucion de llamada de recepcion
    //Se llamará cuando se reciban datos
  esp_now_register_recv_cb(OnDataRecv);

  //Configuracion del servidor NTP
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  //Definicion del modo de operacion de los actuadores
  pinMode(ventilador, OUTPUT);
  pinMode(calefactor, OUTPUT);
  pinMode(bomba, OUTPUT);
}

void loop() {
  //Igualacion de los datos recibidos a las variables
  temperatura = boardsStruct[0].temp;
  humedad = boardsStruct[0].hum;
  niveltanque = boardsStruct[1].nivel;
  humedadsuelo = boardsStruct[2].humsue;

  //Envio de los datos a la base de datos
  Firebase.set(myFireBaseData, "/LECTURAS/temp", temperatura);
  Firebase.set(myFireBaseData, "/LECTURAS/hum", humedad);
  Firebase.set(myFireBaseData, "/LECTURAS/humsue", humedadsuelo);
  Firebase.set(myFireBaseData, "/LECTURAS/nivel", niveltanque);

  //Actuadores
    //Ventilador
  Firebase.get(myFireBaseData, "/BOTON1");
  estados = myFireBaseData.jsonString();
  myJson.setJsonData(estados);
  myJson.get(myJsonData, "/Valor");
  estado1 = myJsonData.stringValue;

  if(estado1 == "true") {
    digitalWrite(ventilador, LOW);
  } else {
    digitalWrite(ventilador, HIGH);
  }

    //Calefactor
```

```
Firebase.get(myFireBaseData, "/BOTON2");
estados = myFireBaseData.jsonString();
myJson.setJsonData(estados);
myJson.get(myJsonData, "/Valor");
estado2 = myJsonData.stringValue;

 if(estado2 == "true") {
  digitalWrite(calefactor, LOW);
} else {
  digitalWrite(calefactor, HIGH);
}

  //Bomba de Agua
Firebase.get(myFireBaseData, "/BOTON3");
estados = myFireBaseData.jsonString();
myJson.setJsonData(estados);
myJson.get(myJsonData, "/Valor");
estado3 = myJsonData.stringValue;

if(estado3 == "true") {
  digitalWrite(bomba, LOW);
} else {
  digitalWrite(bomba, HIGH);
}

//Obtencion de la hora
horaNTP = NTP.getTime("%H");  //Obtencion de la hora (Variable String)
hora = horaNTP.toInt();       //Conversion de la variable en int

//AUTOMATIZACION
 //Sistema de Calefaccion y Ventilación
  //Dia
if (hora >= 6 & hora <= 18){
    if (temperatura >= 21){
      digitalWrite(ventilador, LOW);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", encendido);
    }
    else {
      digitalWrite(ventilador, HIGH);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", apagado);

    }
    if (temperatura <= 14){
      digitalWrite(calefactor, LOW);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", encendido);
    }
    else {
      digitalWrite(calefactor, HIGH);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", apagado);
    }
}

  //Noche
if (hora >= 19 & hora <= 5){
    if (temperatura >= 16){
```

```
      digitalWrite(ventilador, LOW);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", encendido);
    }
    else {
      digitalWrite(ventilador, HIGH);
      Firebase.set(myFireBaseData, "/BOTON1/Valor", apagado);
    }
    if (temperatura <= 9){
      digitalWrite(calefactor, LOW);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", encendido);
    }
    else {
      digitalWrite(calefactor, HIGH);
      Firebase.set(myFireBaseData, "/BOTON2/Valor", apagado);
    }
  }

  if (humedad <= 60){
    digitalWrite(ventilador, LOW);
    Firebase.set(myFireBaseData, "/BOTON1/Valor", encendido);
  }
  if (humedad >= 70){
    digitalWrite(calefactor, LOW);
    Firebase.set(myFireBaseData, "/BOTON2/Valor", encendido);
  }

   //Sistema de Riego
  if (humedadsuelo <= 50){
    digitalWrite(bomba, LOW);
    Firebase.set(myFireBaseData, "/BOTON3/Valor", encendido);
  }
  if (humedadsuelo >= 75) {
    digitalWrite(bomba, HIGH);
    Firebase.set(myFireBaseData, "/BOTON3/Valor", apagado);
  }
}
```

Anexo 4. Código del programa del sistema IoT – Nodo Sensor – DHT22

```
//Libreiras
#include "esp_now.h"
#include "WiFi.h"
#include "DHT.h"

//Sensor
int sensor = 2;         //Variable y pin del sensor
DHT dht(sensor,DHT22);

//Direccion MAC de la placa receptora (Nodo Central) - 24:6F:28:88:CE:EC
uint8_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x88, 0xCE, 0xEC};

//Variables para almacenar las lecturas enviadas
float LecturaAntiguaTemp;
float LecturaAntiguaHum;

//Variables para almacenar las nuevas lecturas
float temperatura;
float humedad;

//Estructura del mensaje a enviar
typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;

//Variable para almacenar los valores de los mensajes
EstructuraMensaje Invernadero;

//Funcion de devolucion de llamada - OnDataSent()
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio
fallido");
}

void setup() {
  //Inicializacion del Monitor Serie
  Serial.begin(115200);

  //Configuracion como Estacion WiFi
  WiFi.mode(WIFI_STA);

  //Inicializacion del protocolo ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error de inicializacion del protocolo ESP-NOW");
    return;
  }

  //Registro de la funcion de devolucion de llamada
```

```
    //Se llamara cuando se envien datos
  esp_now_register_send_cb(OnDataSent);

  //Agregacion del nodo al mismo nivel
    //Para enviar datos a otra placa, debe emparejarlo como un par.
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;

    //Agregacion del par
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("No se puedo agregar el par");
    return;
  }

  //Inicializacion de la libreria del sensor
  dht.begin();

  //Definicion del modo de operacion del sensor
  pinMode(sensor, INPUT);
}

void loop() {
  //Obtencion de medidas del sensor
  temperatura = dht.readTemperature();  //Temperatura en °C
  humedad = dht.readHumidity();         //Humedad en %

  //Establecimiento de valores a enviar
  Invernadero.id = 1;
  Invernadero.temp = temperatura;
  Invernadero.hum = humedad;

  //Impresion
  Serial.println("---------------");
  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.println(" °C");
  Serial.print("Humedad: ");
  Serial.print(humedad);
  Serial.println("%");

  //Comparacion de lecturas
  if (temperatura != LecturaAntiguaTemp || humedad != LecturaAntiguaHum) {
    Serial.println("La temperatura y/o humedad vario");

    //Envio del mensaje a traves del protocolo ESP-NOW
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Invernadero,
sizeof(Invernadero));
    if (result == ESP_OK) {
      Serial.println("Envio del mensaje exitoso");
    }
    else {
      Serial.println("Envio del mensaje fallido");
    }
```

```
  }

  //Igualdad de lecturas antiguas
  LecturaAntiguaTemp = temperatura;
  LecturaAntiguaHum = humedad;
}
```

Anexo 5. Código del programa del sistema IoT – Nodo Sensor – HC-SR04

```
//Librerias
#include "esp_now.h"
#include "WiFi.h"

//Sensor
int trig = 2;         //Transmisor
int eco = 4;          //Receptor
long duracion;        //Variable para duracion del pulso
float distancia;      //Variable para hallar la distancia

//Tanque de Agua
float pi = 3.1416;    //Variable pi
float r = 74.5;       //Radio del tanque (m)
float h = 115;        //Altura del tanque (m)
float Vt;             //Volumen total del tanque (m3)
float Vv;             //Volumen vacio del tanque (m3)
float Vl;             //Volumen lleno del tanque (m3)

//Direccion MAC de la placa receptora (Nodo Central) - 24:6F:28:88:CE:EC
uint8_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x88, 0xCE, 0xEC};

//Variable para almacenar las lecturas enviadas
float LecturaAntiguaNivel;

//Variable para almacenar las nuevas lecturas
float niveltanque;

//Estructura del mensaje a enviar
typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;

//Variable para almacenar los valores de los mensajes
EstructuraMensaje Invernadero;

//Funcion de devolucion de llamada - OnDataSent()
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio
fallido");
}

void setup() {
  //Inicializacion del Monitor Serie
  Serial.begin(115200);

  //Configuracion como Estacion WiFi
  WiFi.mode(WIFI_STA);
```

```
  //Inicializacion del protocolo ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error de inicializacion del protocolo ESP-NOW");
    return;
  }

  //Registro de la funcion de devolucion de llamada
    //Se llamara cuando se envien datos
  esp_now_register_send_cb(OnDataSent);

  //Agregacion del nodo al mismo nivel
    //Para enviar datos a otra placa (el receptor), debe emparejarlo como un par.
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;

    //Agregacion del par
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("No se puedo agregar el par");
    return;
  }

  //Definicion del modo de operacion del sensor
  pinMode(trig, OUTPUT);
  pinMode(eco, INPUT);
}

void loop() {
  //Obtencion de medidas del sensor
    //Encendido y apagado del transmisor
  digitalWrite(trig, HIGH);
  delay(1);
  digitalWrite(trig, LOW);

  duracion = pulseIn(eco, HIGH);  //Recepcion del puslo
  distancia = duracion / 58.2;    //Calculo para hallar la distancia en cm

  //Calculo del Volumen del Tanque
  Vt = pi * pow(r, 2) * h;
  Vv = pi * pow(r, 2) * distancia;
  Vl = Vt - Vv;

  niveltanque = map(Vl, 0, Vt, 0, 100);

  //Establecimiento de valores a enviar
  Invernadero.id = 2;
  Invernadero.nivel = niveltanque;

  //Impresion
  Serial.println("----------------");
  Serial.print("Nivel del tanque: ");
  Serial.print(niveltanque);
  Serial.println("%");
```

```
  //Comparacion de lecturas
  if (niveltanque != LecturaAntiguaNivel) {
    Serial.println("El nivel del tanque vario");

    //Envio del mensaje a traves del protocolo ESP-NOW
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Invernadero,
sizeof(Invernadero));
    if (result == ESP_OK) {
      Serial.println("Envio del mensaje exitoso");
    }
    else {
      Serial.println("Envio del mensaje fallido");
    }
  }

  //Igualdad de lecturas antiguas
  LecturaAntiguaNivel = niveltanque;
}
```

Anexo 6. Código del programa del sistema IoT – Nodo Sensor – HW-080

```
//Libreiras
#include "esp_now.h"
#include "WiFi.h"

//Sensor
int sensor = 33;         //Variable y pin del sensor

//Direccion MAC de la placa receptora (Nodo Central) - 24:6F:28:88:CE:EC
uint8_t broadcastAddress[] = {0x24, 0x6F, 0x28, 0x88, 0xCE, 0xEC};

//Variable para almacenar las lecturas enviadas
int LecturaAntiguaHumsue;

//Variables para almacenar las nuevas lecturas
int humedadsuelo;

//Estructura del mensaje a enviar
typedef struct EstructuraMensaje {
    int id;           //Numero de placa
    float temp;       //Temperatura
    float hum;        //Humedad
    int humsue;       //Humedad del suelo
    float nivel;      //Nivel del tanque
} EstructuraMensaje;

//Variable para almacenar los valores de los mensajes
EstructuraMensaje Invernadero;

//Funcion de devolucion de llamada - OnDataSent()
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nEstado del ultimo paquete enviado\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio
fallido");
}

void setup() {
  //Inicializacion del Monitor Serie
  Serial.begin(115200);

  //Configuracion como Estacion WiFi
  WiFi.mode(WIFI_STA);

  //Inicializacion del protocolo ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error de inicializacion del protocolo ESP-NOW");
    return;
  }

  //Registro de la funcion de devolucion de llamada
    //Se llamara cuando se envien datos
  esp_now_register_send_cb(OnDataSent);

  //Agregacion del nodo al mismo nivel
```

```
    //Para enviar datos a otra placa (el receptor), debe emparejarlo como un par.
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;

    //Agregacion del par
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("No se puedo agregar el par");
    return;
  }

  //Definicion del modo de operacion del sensor
  pinMode(sensor, INPUT);
}

void loop() {
  //Obtencion de medidas del sensor
  humedadsuelo = map(analogRead(sensor), 4092, 0, 0, 100);  //Humedad del suelo
en %

  //Establecimiento de valores a enviar
  Invernadero.id = 3;
  Invernadero.humsue = humedadsuelo;

  //Impresion
  Serial.println("----------------");
  Serial.print("Humedad del suelo: ");
  Serial.print(humedadsuelo);
  Serial.println("%");

  //Comparacion de lecturas
  if (humedadsuelo != LecturaAntiguaHumsue) {
    Serial.println("La humedad del suelo vario");

    //Envio del mensaje a traves del protocolo ESP-NOW
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Invernadero,
sizeof(Invernadero));
    if (result == ESP_OK) {
      Serial.println("Envio del mensaje exitoso");
    }
    else {
      Serial.println("Envio del mensaje fallido");
    }
  }

  //Igualdad de lecturas antiguas
  LecturaAntiguaHumsue = humedadsuelo;
}
```