UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA ELECTRÓNICA

DESARROLLO DE UN DISPOSITIVO IOT EN CLOUD PARA LA ELABORACIÓN DE COMPOSTAJE EN INVERNADEROS CASEROS

Trabajo de titulación previo a la obtención del
Título de Ingeniero Electrónico

AUTORES: Wilmer Patricio Caiza Guallichico
Christian Alejandro Herrera Cadena

TUTOR: Luis Germán Oñate Cadena

Quito-Ecuador

2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Wilmer Patricio Caiza Guallichico, con documento de identificación No. 1721846663 y Christian Alejandro Herrera Cadena, con documento de identificación No. 1721335337; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 14 de febrero del año 2023.

Atentamente,

Wilmer Patricio Caiza Guallichico	Christian Alejandro Herrera Cadena
1721846663	1721335337

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Wilmer Patricio Caiza Guallichico, con documento de identificación No. 1721846663 y, Christian Alejandro Herrera Cadena con documento de identificación No. 1721335337, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del proyecto técnico: "Desarrollo de un dispositivo Iot en cloud para la elaboración de compostaje en invernaderos caseros", el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 14 de febrero del año 2023.

Atentamente,

Wilmer Patricio Caiza Guallichico	Christian Alejandro Herrera Cadena
1721846663	1721335337

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Luis Germán Oñate Cadena con documento de identificación No. 1712157401 docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UN DISPOSITIVO IOT EN CLOUD PARA LA ELABORACIÓN DE COMPOSTAJE EN INVERNADEROS CASEROS, realizado por Wilmer Patricio Caiza Guallichico con documento de identificación No. 1721846663 y por Christian Alejandro Herrera Cadena con documento de identificación No. 1721335337, obteniendo como resultado final el trabajo de titulación bajo la opción proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 14 de febrero del año 2023.

Atentamente,

Ing. Luis Germán Oñate Cadena. MSc.
1712157401

DEDICATORIA

Quiero dedicar este trabajo a toda mi familia. Principalmente, a mis padres que me apoyaron en cada etapa buena y mala de mi vida universitaria. Gracias por enseñarme a afrontar las dificultades a ser la persona que soy hoy, mis principios, mis valores, mi perseverancia y mi empeño. Todo esto con su amor infinito y sin pedir nada a cambio.

También quiero dedicarle este trabajo a Silvana. Por toda su paciencia, fuerza y amor, porque la quiero. Realmente por ayudarme en ese momento decisivo en donde me vi perdido y me ayudo a alcanzar el equilibrio. Nunca dejaré de estar agradecido por esto.

Wilmer Patricio Caiza Guallichico

Le dedico el resultado de este trabajo de titulación a toda mi familia. Especialmente, a mis abuelos Marcelo y Victoria quienes han sido un pilar fundamental en mi vida, los cuales han estado junto a mí en todas las etapas importantes de mi vida. En memoria a mi abuelo Marcelo, quien gracias a sus enseñanzas que me brindo antes de tu partida poder lograr ser la persona que soy hoy, y a mi hijo Julian que fue siempre mi inspiración y darme ese empujón que necesitaba para seguir adelante.

Christian Alejandro Herrera Cadena

AGRADECIMIENTOS

A mis padres Gabriel y Esther por siempre apoyarme para conseguir mis metas personales y académicas. También son los que me han brindado el soporte material y económico para poder concentrarme en los estudios y nunca abandonarlos.
A mi tutor Ing. Luis Oñate. por su dedicación y paciencia, sin sus palabras y correcciones precisas no hubiese podido lograr llegar a esta instancia tan anhelada.
A los docentes por repartir sus conocimientos de manera profesional, gracias por su paciencia, perseverancia y tolerancia.

Wilmer Patricio Caiza Guallichico

Quiero dar gracias a Dios, quien con su bendición ha estado presente en mi vida, por brindarme la salud y la fortaleza necesaria para cumplir todos mis anhelos y manifestarse en cada uno de ellos. De manera especial a mi tutor de tesis y a todos mis docentes, por darme esa guía, no solo en la elaboración de este trabajo, sino a lo largo de toda mi carrera universitaria y haberme brindado el apoyo para mi desarrollo humano y profesional, y a todos los que creyeron desde un principio en mí.

Christian Alejandro Herrera Cadena

Agradecemos en el presente trabajo a la Universidad Politécnica Salesiana y a todos los que la conforman, por ser una guía durante toda la carrera y abrirnos las puertas para estudiar Ingeniería Electrónica. A nuestro tutor el Ingeniero Luis German Oñate Cadena por su tiempo y conocimiento invertido para la realización de este proyecto de titulación.

Los Autores

ÍNDICE

ÍNDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE ANEXOS

RESUMEN

El presente proyecto hace referencia al diseño de un dispositivo IoT enfocado en la elaboración y monitoreo de compostaje en invernaderos a nivel casero, mediante una red inalámbrica se realiza la conexión de sensores tanto de temperatura, humedad y pH para el procesamiento del compost según las condiciones óptimas para tener un producto de mayor calidad a un menor tiempo. La red inalámbrica permite enlazar los sensores con la tarjeta embebida ESP8266 que se comunican con el servidor ya que maneja tecnología Wifi además por medio del software Node-RED se enlaza cada controlador para realizar la programación a nivel de nodos gráficos. El uso de una aplicación permite observar las variables de control en tiempo real por medio de un teléfono inteligente, además el diseño del prototipo es realizado en el software Inventor donde se puede tener las dimensiones reales y gráficos en 3D del dispositivo donde se va implementar todo el sistema inalámbrico.

ABSTRACT

The present project refers to the design of an IoT device for the elaboration and monitoring of composting in greenhouses at the home level, through a wireless network, the connection of both temperature, humidity and pH sensors is made for the processing of the compost according to the conditions. Optimum to have a higher quality product in less time. The wireless network allows the sensors to be linked with the ESP8266 embedded card that communicate with the server since it handles Wi-Fi technology. In addition, through the Node-RED software, each controller is linked to carry out programming at the graphic node level. The use of an application allows to observe the control variables in real time through a smartphone, in addition the design of the prototype is made in the Inventor software where you can have the real dimensions and 3D graphics of the device where it is going to be implemented. the entire wireless system.

INTRODUCCIÓN

El proyecto mostrado a continuación muestra el desarrollo de un dispositivo IoT enfocado en la elaboración y monitoreo de compost para un invernadero casero el cual permitirá conectar de forma inalámbrica los sensores con un servidor MQTT para el monitoreo de las variables de control en este caso temperatura, humedad y pH del suelo instalados en un prototipo. A continuación, se describe los puntos a tratar para el desarrollo del proyecto:

En el capítulo 1, se indica los aspectos generales del proyecto, su justificación, el planteamiento del problema, los objetivos y la metodología utilizada en el desarrollo del mismo.

En el capítulo 2, se describe la fundamentación teórica que es la base en la cual está basado el proyecto, conceptos y caracterización de los dispositivos y elementos que intervienen en el proyecto.

En el capítulo 3, se indica el diseño tanto de la red inalámbrica como la placa donde van a ir conectados los elementos como el ESP8266 y sensores además del diseño del prototipo en 3D.

En el capítulo 4, se procede a indicar cuales fueron las pruebas de funcionamiento realizadas para un correcto intercambio de datos entre los sensores y la aplicación móvil.

En el capítulo 5, se muestran las conclusiones del proyecto basados en la implementación del dispositivo como en las pruebas de funcionamiento realizadas.

En el capítulo 6, se detalla las recomendaciones en base a el conocimiento adquirido durante el desarrollo de todo el proyecto sentando bases para posibles investigaciones que tengan que ver con dispositivos IoT.

CAPÍTULO I
ANTECEDENTES

1.1 Planteamiento del problema

El compostaje es un proceso en el cual los residuos orgánicos son descompuestos, esta práctica es muy común en los invernaderos caseros y ha ido creciendo con el pasar de los años, este proceso convierte la materia orgánica en un tipo de abonos natural que es utilizado en los cultivos aprovechando la materia orgánica que se desperdicia en los hogares para reutilizarla por esto se realizará un proceso de compostaje en un vivero ubicado en el Valle de los chillos sector Conocoto.

El proceso de compostaje se basa en clasificar residuos orgánicos que pueden ser utilizados en el proceso de descomposición, estos van a ser puestos en un recipiente dónde se van a controlar las condiciones de temperatura, humedad y oxigenación. Al pasar el tiempo estos residuos van liberando agua y oxígeno entre otros componentes orgánicos, reduciendo su volumen y convirtiéndose en un abono natural, esto se ha venido realizando en recipientes denominados con postergas que son construidas de forma casera y artesanal, sin embargo, al realizarlo de esta manera el tiempo para generar el compost suele ser largo y no garantiza que sea de calidad, en cada compostador se arroja residuos sin tener en cuenta las condiciones de temperatura, oxigenación, humedad y pH en forma manual.

Por lo anterior mencionado, es necesario automatizar este proceso teniendo en cuenta las variables que pueden ser medidas y controladas desde la nube y de esta manera se pueda interactuar con los responsables del compostaje de forma directa.

1.2 Justificación

El desarrollo de un dispositivo IoT en cloud para la elaboración de compostaje en invernaderos caseros como proyecto técnico en el ámbito tecnológico beneficiará a las personas que realizan procesos de cultivos en invernaderos caseros mejorando la calidad de los cultivos, además con la App se puede monitorear las variables físicas de forma continua para obtener el compostaje en un tiempo más corto y de mejor calidad que al realizarlo de manera tradicional en los invernaderos caseros.

Con este proyecto se busca fortalecer el sistema de compostaje inteligente y mejorar el proceso de conversión de la basura en abono, puesto que un control automatizado de sus parámetros logra reducir el tiempo en que el proceso se tarda, de esta manera se puede obtener mayor cantidad de producto y de mejor calidad, para esto el desarrollo de un dispositivo IoT beneficiara a las personas que generan el compostaje en sus invernaderos caseros pudiendo controlar las distintas variables que intervienen en el sistema como son: la temperatura, humedad y pH y así poder visualizar las mismas en una aplicación móvil en tiempo real.

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar un dispositivo IoT cloud en invernaderos caseros para el monitoreo de las variables temperatura, humedad y pH relacionadas con la elaboración del compostaje.

1.3.2 Objetivos Específicos

- Analizar los requerimientos del proceso de compostaje en un invernadero casero para la determinación de los rangos de temperatura, humedad y pH que generen un compost en menor tiempo y de mejor calidad.

- Implementar el sistema de compostaje mediante una red IoT y cloud para el cumplimiento de los parámetros de temperatura, humedad y pH.

- Desarrollar una aplicación móvil para el monitoreo de los sensores en tiempo real según los parámetros de temperatura, humedad y pH.

- Realizar pruebas del funcionamiento para la comprobación de la comunicación entre los sensores que conforman el sistema y la aplicación móvil.

1.4 Metodología

Metodología analítica: Este método servirá para considerar las variables físicas que permitirán buscar los elementos más adecuados para la elaboración del prototipo ya sean sensores, controladores o actuadores.

Metodología deductiva: Con la información obtenida se creará el dispositivo IoT y la aplicación para el smartphone.

Metodología experimental: se realizará las pruebas necesarias para la verificación del funcionamiento del dispositivo IoT para el proceso de compostaje.

CAPÍTULO II
FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se analiza la fundamentación teórica para la realización de la investigación, para que el lector tenga una idea clara del tema tratado, en base a los conceptos que sustentan la solución del problema.

2.1 Invernaderos Caseros

Los invernaderos son un sistema de producción agrícola o también llamado agro sistema. En muchos hogares se opta por la creación de invernaderos caseros pero con el avance de la tecnología además de los cambios tecnológicos se han hecho no avances en el diseño de las estructuras y en los materiales de cubierta que pueden ser el vidrio o plásticos, sino también en las técnicas de control del medio ambiente como la climatización, fertiirrigación, enriquecimiento en CO2, control integrado de las plagas, se ha optado por la automatización de procesos (Aquino, 2017).

2.2 Compostaje

El proceso de compostaje logra transformar la materia orgánica para la obtención del compost por medio de su descomposición logrando un abono natural para las tierras y suelos que sirven para el cultivo y la agricultura en general. En el sector agrícola se utilizan los residuos orgánicos ya que tienen un alto contenido de componentes fenólicos causados por efectos fitotóxicos, estos provocan una disminución en el crecimiento de las plantas hasta llegar a la muerte. para qué los residuos se puedan utilizar deben pasar por un proceso de degradación donde se disminuye y se elimina los compuestos que causan la toxicidad. siendo un proceso hay Rubio los microorganismos como hongos y bacterias son los encargados de transformar la materia orgánica en un producto libre de patógenos y estable por las altas temperaturas a los que son sometidos, el proceso de grada lentamente el material hasta eliminar la fitotoxicidad, actor importante tomar en cuenta en los hogares sí obtienen al menos un 40% de materia orgánica que podría servir para este propósito (Méndez, 2018).

2.2.1 Interpretación para el análisis del compost

Con respecto a la humedad los valores habituales para obtener un buen producto están entre el 40% y 60% durante las primeras etapas de descomposición de materia orgánica, con respecto a la temperatura se estima un valor de 60 °C a 70 °C así también el porcentaje de materia orgánica en el contenedor debe estar entre el 30% con respecto a la cantidad total que se vaya a poner dentro del prototipo.

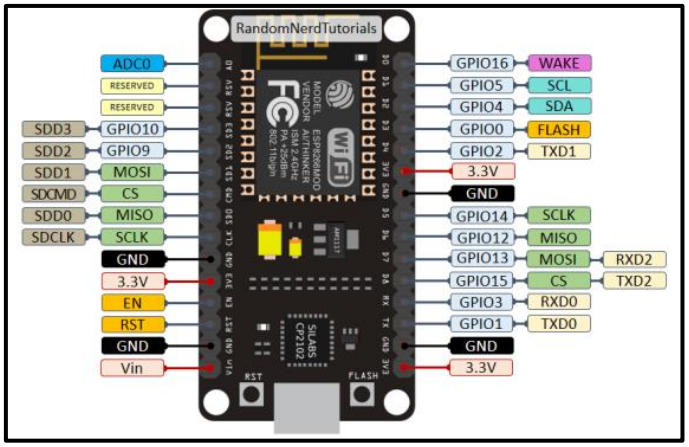
2.3 Node-RED

Esta herramienta permite a la persona que la vaya a usar una facilidad de programar sin tener que escribir una línea de código, pues es un editor de flujo basado en el navegador que añade, elimina o conecta entre sí nodos para establecer comunicación. Mediante el tablero gráfico se controla los dispositivos y se monitorea el estado de los mismos dentro del mismo dominio, los nodos en Node-RED son subrutinas con entradas y salidas que pueden recibir mensajes por MQTT (Clavijo, 2021). Consta de un motor para crear y ejecutar las conexiones entre las aplicaciones de IoT (Arroba, 2021).

2.4 Node MCU ESP8266

El ESP8266 posee una pila TCP/IP completa y capacidad de unidad de microcontrolador (MCU) a bajo costo. El módulo ESP8266 posee dos modos de conexión, la primera funciona con el firmware ESP-AT para la conectividad Wi-fi a microcontroladores de host externos, el segundo modo de conexión trabaja como un microcontrolador independiente con un RTOS SDK que ejecuta aplicaciones de conectividad de forma nativa. Con esto los clientes pueden disfrutar de conectividad en la nube, operación de bajo consumo de energía y soporte para seguridad Wi-Fi (Valencia & Delgado, 2021).

Figura 0.1 Node MCU ESP8266.

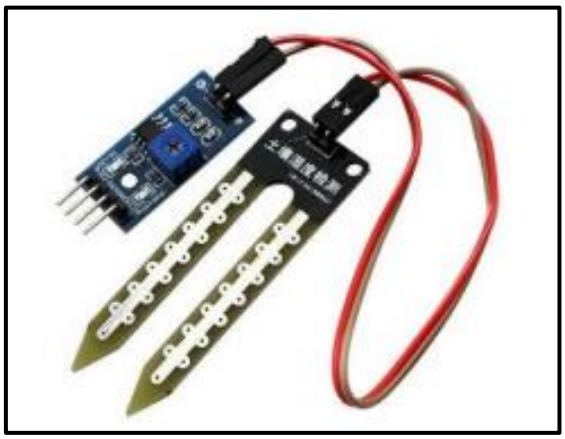


Fuente: (Valencia & Delgado, 2021)

2.5 Sensor de Humedad de Suelo

Los sensores de humedad del suelo FC-28, se utilizan frecuentemente para lograr controlar la humedad del suelo o tierra de las plantas objeto de estudio de invernaderos, jardín botánico, entre otros. Se sabe si la tierra está húmeda, seca o tiene demasiada agua porque el sensor utiliza la tensión proporcional al su nivel de humedad medida (Barrientos, 2020).

Figura 0.2 Sensor de Humedad de suelo FC-28.



Fuente: (Barrientos et al., 2020)

El sensor de humedad posee varias características siendo compatible con Arduino es u sensor ideal para el monitoreo de la humedad funciona a base de dos electrodos que deben ser insertados en el suelo este electrodo ira conectado a una tarjeta de acondicionamiento YL-38 entregando una salida digital con un voltaje de funcionamiento que está en un rango de 3.3 V a 5 V aproximadamente, en la Tabla 2.1 se indica los valores de precisión y respuesta de este sensor.

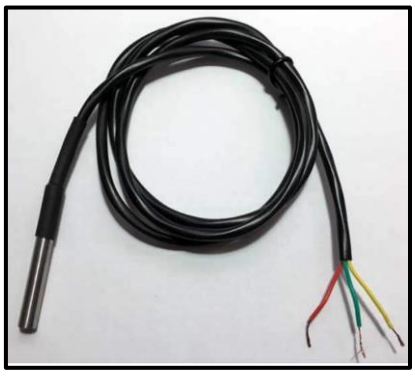
Tabla 0.1 Características del sensor FC-28

Precisión	± 4%
Resolución	1%
Tiempo de sensado	2 seg.

2.6 Sensor de Temperatura DS18B20

El sensor de temperatura DS18B20 tiene la característica de soportar ambientes húmedos es de fácil uso al conectar con los microcontroladores Arduino o ESP8266/ESP32. Este sensor debe ser acondicionado para convertir la señal analógica y permitir tomar los datos de temperatura puede ser usado simultáneamente con otros sensores ya que este tiene un identificador único de fábrica (Valencia, 2021).

Figura 0.3 Sensor de Temperatura DS18B20.



Fuente: (Valencia & Delgado, 2021)

El sensor de temperatura DS18B20 trabaja con un voltaje de operación de 3 V a 5.5 V DC con un rango de operación de -55 ºC hasta los 125 ºC esta versión es a prueba de agua

por lo que es ideal para la implementación en el proyecto ya que posee una cubierta de acero inoxidable de alta calidad en la tabla 2.2 se indica las características del sensor con respecto a la precisión que tiene.

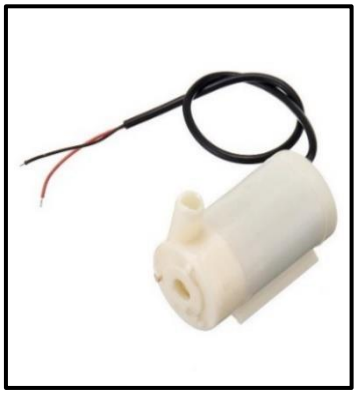
Tabla 0.2 Características del sensor DS18B20.

Precisión	≤± 2%.
Repetibilidad	±1%
histéresis	< ± 0.3%

2.7 Mini bomba 5 V (Arduino)

La bomba a utilizar para el riego del compost es una maquina hidráulica la cual permite generar energía cinética para tener un caudal de agua, estas bombas permiten controlar el flujo de agua siempre y cuando sean accionadas con un control previo para este caso se debe conocer el voltaje de operación el cual va de los 2.5V a los 6V en corriente continua, la bomba puede ser sumergida dentro de un contenedor de agua para ser extraída según los requerimientos, se debe tener en cuenta que si se desea adicionar una bomba más potente es necesario el uso de un relé.

Figura 0.4 Bomba 5V para Arduino



Fuente: (Valencia & Delgado, 2021) Recuperado de: https://naylampmechatronics.com/

2.8 Resistencia Calefactora (Niquelina)

La resistencia de calor o niquelina es un elemento usado para generar los cambios de temperatura, existen un sin número de niquelinas en el mercado sin embargo para el

proyecto se utilizará una que no exceda los 5 A para que no exista ningún problema con el relé, la gran mayoría de resistencias están hechas con alambre de una aleación de nique y cromo capaces de soportar altas temperaturas.

Figura 0.5 Resistencia Calefactora (Niquelina)

Fuente: (Valencia & Delgado, 2021) Recuperado de: https://naylampmechatronics.com/

2.9 Sensor de pH

El sensor de potencial de hidrógeno identifica los valores de alcalinidad presente en una solución acuosa, posee una precisión de 1% pH, permitiendo la construcción de sistemas con grandes rangos de precisión a un costo bajo. Está compuesto de dos partes: el sensor pH posee una placa de conversión que se conecta al Arduino que es la encargada de procesar los datos obtenidos por el sensor pH (Coello & Silva, 2020).

Figura 0.6 Sensor de Potencial de Hidrógeno (pH).

Fuente: (Coello & Silva, 2020)

El sensor de pH posee una tarjeta controladora que muestra un valor analógico de la medición con un potenciómetro que permite calibrar la sonda este de igual manera

funciona con una alimentación de 5V DC y un rango de medición de 0 a 14 pH en la tabla 2.3 se indica las características del sensor con respecto a la precisión.

Tabla 0.3 Características del sensor de pH con respecto a la precisión

Precisión	± 1%
Rango de medición	0-14
Tiempo de respuesta	≤ 1 min

En la tabla 2.4 se observa la dispersión del valor de medida de cada sensor en función de su sensibilidad o factor de escala (F.S) indicado por el fabricante a este dato también se lo conoce como incertidumbre sistemática de medida, por lo que se debe tener presente que el 95% de los datos tomados de una población.

Se debe calcular estos valores haciendo pruebas experimentales en los cuales se usa la media y la desviación estándar para encontrar la incertidumbre aleatoria.

Tabla 0.4 Incertidumbre sistemática y aleatoria para los sensores.

Sensor	B_x	b_x	S_x	s_x
temperatura	≤±2% F.S	≤±1% F. S	±1,73% F. S	±0,57% F. S
humedad	≤±4% F. S	≤±2% F. S	±1,82% F. S	±0,91% F. S
pH	≤±1% F. S	≤±0,5% F. S	±0,457% F. S	±0,228% F. S

Estos sensores fueron utilizados porque cubren con los requerimientos para medir todas las variables siendo compatibles con Arduino y la tarjeta controladora ESP8266 además de tener un costo relativamente accesible son de los más usados en estos prototipos.

2.10 Broker y Node-RED

El software no de red tiene la capacidad de enlazar y permitir la conexión de muchos dispositivos de hardware con API además de brindar servicios de conectividad en línea. La programación es realizada por medio de nodos de subscripción y publicación enfocados a soluciones basados en internet de las cosas, estas conexiones son realizadas mediante una interfaz gráfica que se encuentra instalada en el servidor este tipo de entorno es de fácil manejo para el usuario permitiendo una transportación de datos y almacenamiento sin ocupar demasiado ancho de banda (Pillajo, 2018).

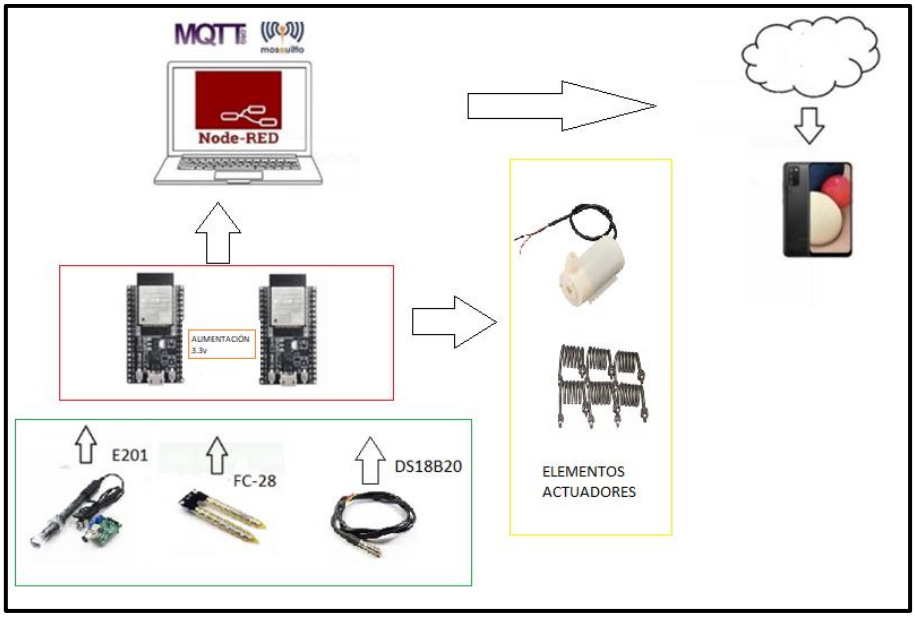
CAPÍTULO III
DISEÑO DE LA RED INALÁMBRICA

En este capítulo se describe a detalle el desarrollo y diseño de un dispositivo IoT para la elaboración de compostaje acompañada de una aplicación móvil, para lograr cumplir este objetivo se procede a realizar los respectivos diseños, esquemáticos y diagramas que intervienen para la implementación del proyecto.

2.11 Diseño del dispositivo IoT

El dispositivo IoT está constituido por sensores de humedad, pH y temperatura los cuales son tomados como variables de control y monitoreo que serán configurados en un rango para tener un compost que trabaja de la mejor manera. Para esto los sensores son conectados a un módulo controlador ESP8266 mismo que posee tecnología Wifi que permite una conexión inalámbrica hacia un servidor MQTT para la administración de los datos, se usa el software NODE-RED para programar los nodos de entrada y salida siendo este una interfaz de programación y gestión de datos para toda la red inalámbrica como se muestra en la Figura 3.1.

Figura 0.1 Diagrama de Bloques para comunicación de los sensores.



Elaborado por: Wilmer Caiza y Christian Herrera.

Los bloques vistos en la Figura 3.1 hacen referencia a la conexión que tienen los sensores tanto de humedad, temperatura y pH con el ESP8266 pues los tres van a ser administrados por un solo controlador. El sensor de pH o E201 va a ocupar el puerto analógico A01, mientras que el sensor FC-28 va ocupar otro puerto analógico y DS18B20 va ser digital. El sensor E201 es usado para medir el pH del suelo mientras que el sensor FC-28 es un sensor de humedad para suelo, su medición la realiza por medio de dos electrodos resistivos además cuenta con un sensor de temperatura DHT22, estos sensores son conectados a la placa ESP8266 la cual mapea los datos para posteriormente ser leídos en servidor MQTT y enviados a un teléfono inteligente para su monitoreo. A continuación, se detallan las características de cada sensor.

El sensor de temperatura será instalado dentro del prototipo, para un monitoreo continuo ya sea desde la computadora o la aplicación instalada en el teléfono. Es de importancia saber las características del sensor en este caso su voltaje de entrada es de 3V a 5.5V DC con un rango de medición de -55 °C a 125 °C suficientes para el rango de valores admisibles en el prototipo se debe tomar en cuenta que este sensor es a prueba de agua por lo que es ideal para el proyecto que se lleva a cabo.

El sensor de humedad de suelo FC-28 mide la resistencia entre 2 electrodos insertados dentro del suelo, la resistencia que existe entre los electrodos siempre será proporcional a la humedad existente en el suelo, si es muy baja (corto circuito) y para un suelo seco será muy alta (circuito abierto) el voltaje de alimentación está en el rango de 3.3V a 5V y una corriente de operación de 35mA.

El medidor de pH analógico se ha diseñado para el uso con Arduino, cuenta con un conector BNC para una conexión instantánea de la sonda con un voltaje de alimentación de 5V DC y una corriente de operación de 10mA.

2.12 Bloque Controlador ESP8266

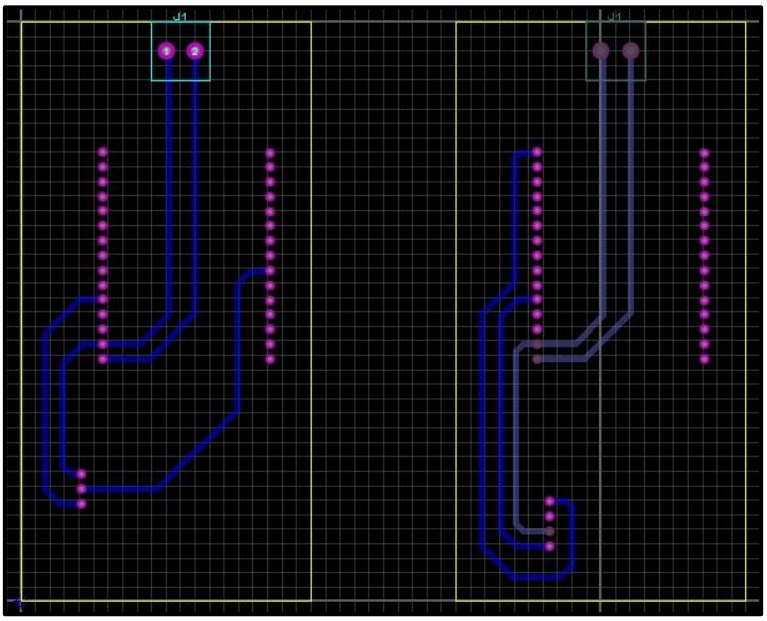
En el desarrollo del proyecto se usará como elemento controlador un módulo ESP8266 el cual permite la integración de redes inalámbricas ya que utiliza tecnología Wifi siendo una gran opción por su bajo consumo de energía e intercambio de datos de manera

bidireccional. Una de las principales características del controlador es su lector de valores análogos pues posee uno llamado A0 por lo que es necesario tener un controlador independiente por cada sensor.

2.13 Diseño de la placa

Para el montaje del ESP8266 se procedió a diseñar una placa en el software Proteus como se muestra en la Figura 3.2 la placa esta provista de pistas que van a conectar el sensor y la alimentación de voltaje y los pines respectivos para la lectura de los sensores por medio del canal analógico.

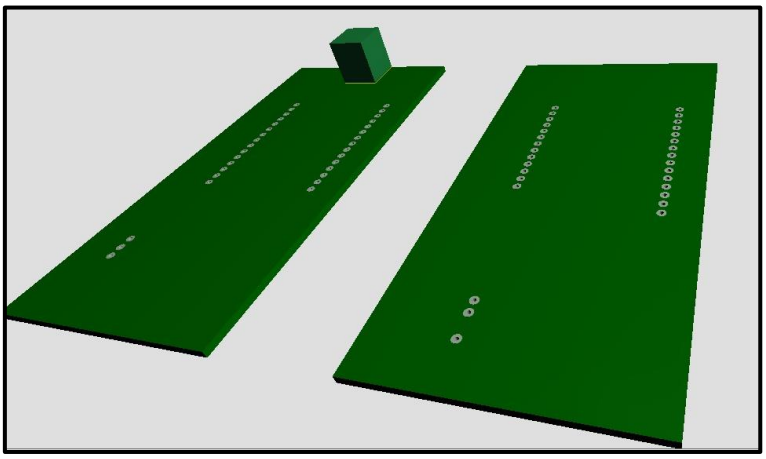
Figura 0.2 Esquemático para la placa ESP8266.



Elaborado por: Wilmer Caiza y Christian Herrera.

En la Figura 3.3 se indica un modelo de la placa en 3D donde se va a colocar el ESP8266 para esto se realiza la impresión del diseño utilizando papel acetato para posteriormente pasarlo a una baquelita y montar los elementos.

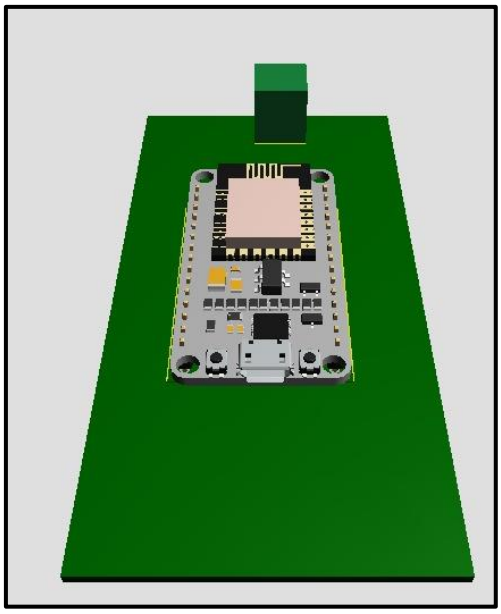
Figura 0.3 Modelo de placa en 3D (Software Proteus).



Elaborado por: Wilmer Caiza y Christian Herrera.

En la Figura 3.4 se muestra el montaje completo del ESP8266 desde una vista superior, se debe verificar que las pistas tengan continuidad pues muchas veces pueden existir saltos o cortes.

Figura 0.4 Montaje de ESP8266 en la placa.



Elaborado por: Wilmer Caiza y Christian Herrera.

2.14 Diseño del Prototipo

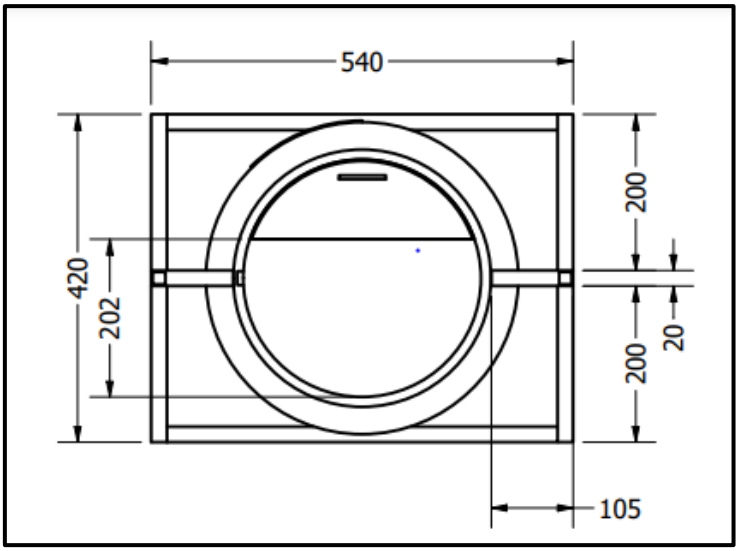
Para el diseño del prototipo se procede a construir un contenedor que sea capaz de almacenar el compost y mantener a los sensores en un lugar donde no sufran ningún daño

por lo tanto se realizó un esquema del prototipo con sus respectivas medidas como se muestra en la Figura 3.5.

El sistema para realizar compost consta de los elementos siguientes:

- Tanque de 20 gal en plástico con tapa superior removible.
- Estructura de acero de tubo cuadrado de 20x20x1,5mm.
- Varilla redonda de 10 mm de diámetro con placa de 50x50mm para instalación de sensores.
- Compuerta de lámina galvanizada 200x200x1.5mm
- 3 bisagras de 1 ½”.
- Aspersor de agua.
- Acoples de tubería (niple, unión, codo).
- Resistencia eléctrica

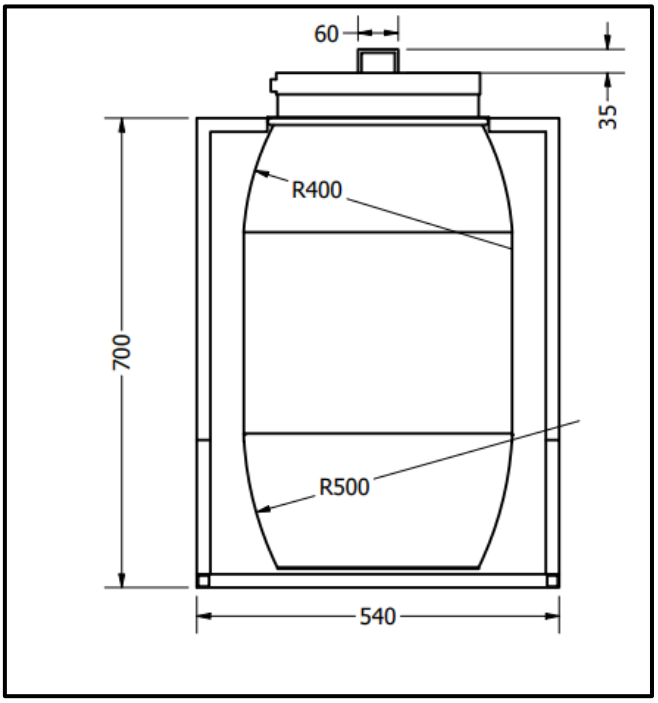
Figura 0.5 Vista superior del prototipo para almacenamiento del compost.



Elaborado por: Wilmer Caiza y Christian Herrera.

El contenedor tiene una altura de 700 mm desde la base y un ancho de 540 mm medidos desde su estructura adicionalmente posee ventanas tanto para el ingreso del material como para la salida colocadas en la parte superior e inferior, la estructura de metal sostiene el contenedor y permite trasladarlo de forma segura como se indica en la Figura 3.6.

Figura 0.6 Vista lateral izquierda del prototipo para el almacenamiento del compost.



Elaborado por: Wilmer Caiza y Christian Herrera.

En el tanque de 20 gal, el cual consta de un volumen aproximado de 0.065 m3, donde se llena a ¾ partes con el material correspondiente al compost, el cual se realiza en primer lugar con una capa de tierra común, luego se añade lo que son los desperdicios vegetales y/o animales que se genera en el día a día de un hogar este proceso se repite hasta crear capas de tierra seguidas de capas de desechos. Ahora como el compost es un proceso de degradación de los desechos donde se debe ir humedeciendo y a la vez calentando para que las bacterias no tengan lugar a expandirse.

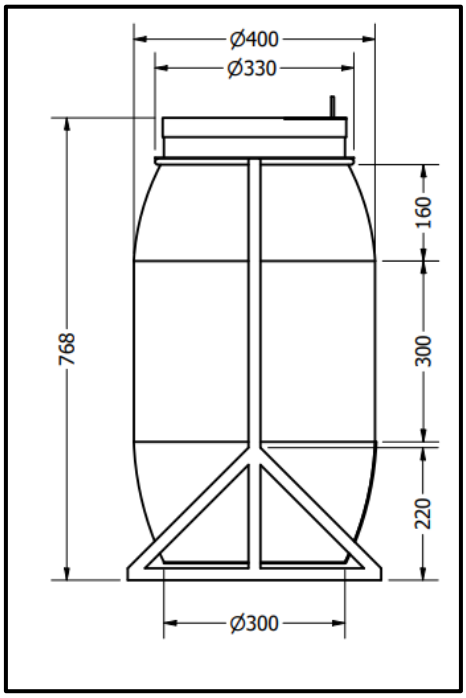
Se tiene los sensores que van dentro del proceso de compost, donde mide la humedad, el pH y la temperatura, así también existe la colocación del aspersor de agua en la parte superior en la tapa fija y la resistencia eléctrica la cual ayuda a mantener a una buena temperatura para el proceso de degradación.

El sistema eléctrico va por fuera anclado a la estructura de acero provista. El cual

monitorea y envía las señales censadas para llevar un registro y/o control de las variables del proceso. Una vez terminado el proceso, se tiene que retirar por la compuerta interior el compost ya procesado.

En la Figura 3.7 se indica las dimensiones que ocupa el contenedor junto con la estructura de metal asignada para sostener y colocar los sensores dentro del prototipo.

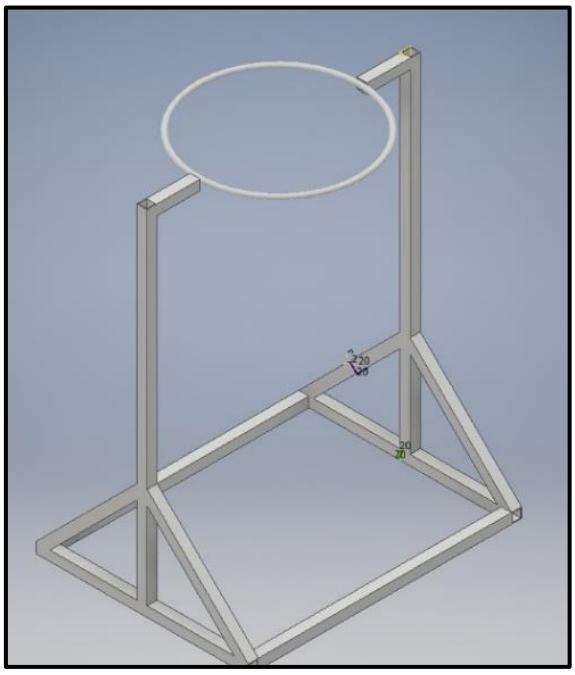
Figura 0.7 Vista lateral derecha del prototipo para almacenamiento del compost.



Elaborado por: Wilmer Caiza y Christian Herrera.

Se realizó una simulación de la estructura que se va a implementar en el software Inventor como se observa en la Figura 3.8 se crea una estructura metálica que sostiene el contenedor con las medidas anteriormente descritas.

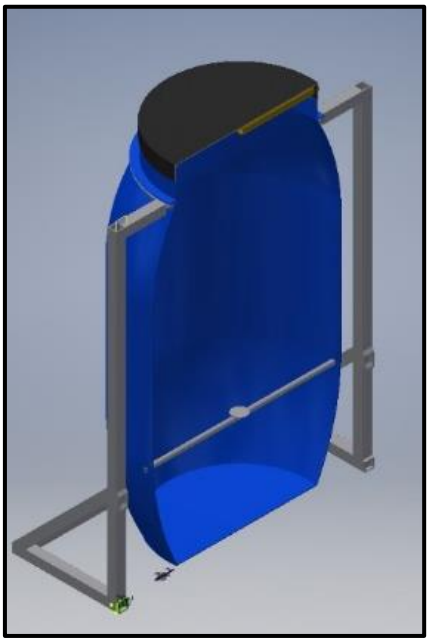
Figura 0.8 Estructura metálica para el contenedor de compost.



Elaborado por: Wilmer Caiza y Christian Herrera.

Se muestra un corte lateral de la estructura donde se va a depositar el compost y mediante los sensores ubicados en el interior lograr medir cada una de las variables que en este caso son temperatura, humedad y pH del suelo para obtener un mejor compost.

Figura 0.9 Corte lateral del prototipo para almacenamiento de compost.



Elaborado por: Wilmer Caiza y Christian Herrera.

En la Figura 3.10 se tiene la vista general exterior del prototipo que va contener el material mismo que será sellado por una tapa en la parte superior y una compuerta en la parte inferior para sacar con facilidad el compost ya procesado.

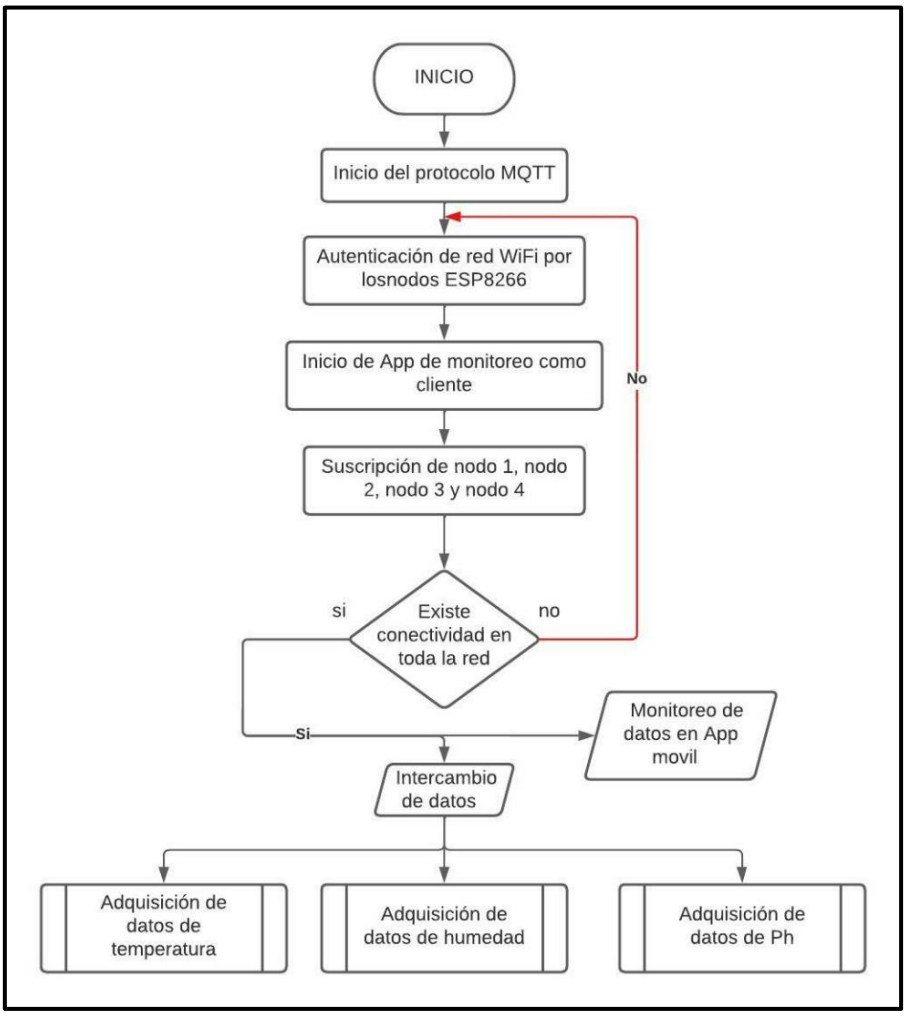
Figura 0.10 Vista general del prototipo para almacenamiento del compost.

Elaborado por: Wilmer Caiza y Christian Herrera.

2.15 Diagrama de flujo

A continuación, en la Figura 3.11 se indica el diagrama de flujo para la adquisición de datos de los sensores desde Node-RED hasta la Aplicación móvil para el monitoreo de las variables, se debe dar acceso al servidor para tener una comunicación por medio del protocolo MQTT mediante el uso de nodos para cada sensor posteriormente se realiza la adquisición e intercambio de datos de forma inalámbrica.

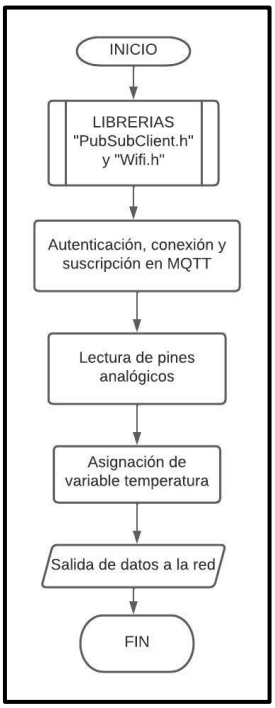
Figura 0.11 Diagrama de flujo para adquisición de datos.



Elaborado por: Wilmer Caiza y Christian Herrera.

Para la lectura general de cada sensor se procede a instalar las librerías PubSubClient.h y Wifi.h que permiten la conexión y suscripción al servidor, estas libreras son instaladas en el IDE de Arduino para posteriormente ser compiladas y cargadas al ESP8266, se procede a la lectura de los pines analógicos para acondicionar la señal a los valores que se requieran tanto de temperatura, humedad y pH.

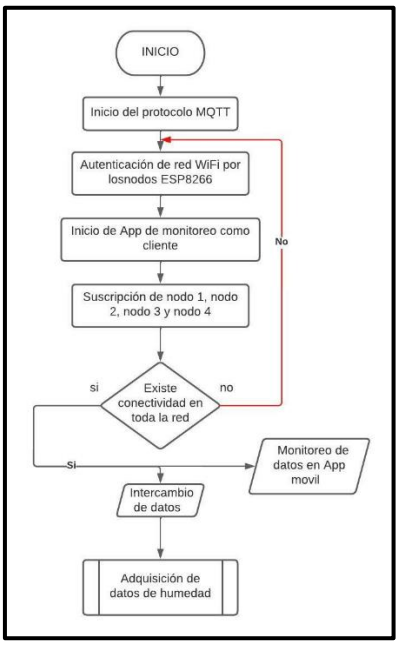
Figura 0.12 Diagrama de flujo para lectura de pin análogo.



Elaborado por: Wilmer Caiza y Christian Herrera.

La lectura del protocolo de comunicación MQTT permite enviar y recibir datos desde el servidor hasta los sensores mismos que mediante una suscripción en cada nodo permiten la conectividad en toda la red y los dispositivos que la conforman.

Figura 0.13 Diagrama de flujo para el sensor de humedad



Elaborado por: Wilmer Caiza y Christian Herrera.

CAPÍTULO IV
IMPLEMENTACIÓN Y PRUEBAS

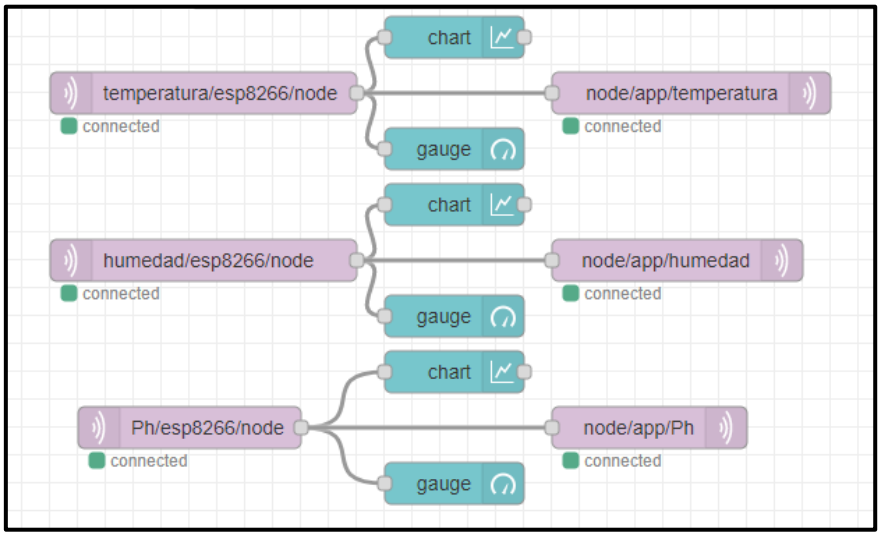
Este capítulo detalla los procesos realizados para obtener los datos de cada variable a ser reflejada en la aplicación móvil mediante el uso de dispositivos IoT previamente configurados y conectados por medio del protocolo de comunicación MQTT.

2.16 Nodos usados en la interfaz Node-red

Para el flujo correcto de datos en la red inalámbrica es necesario el uso de la programación mediante nodos enviados a diferentes etiquetas que identifican a los dispositivos a estos se los conoce como topic´s teniendo la capacidad de enviar datos sin ocupar todo el ancho de banda, fijándose en los requerimientos de acuerdo a los sensores, variables o ambientes a controlar enlazando a los sensores con el entorno grafico en Node-red.

En la Figura 4.1 se indica los tres datos que van a ser enviados desde el módulo ESP8266, así como los nodos chart y gauge que permiten generar un entorno grafico para mostrar los datos obtenidos en el Dashboard.

Figura 0.1 Nodos para recepción y envió de datos.



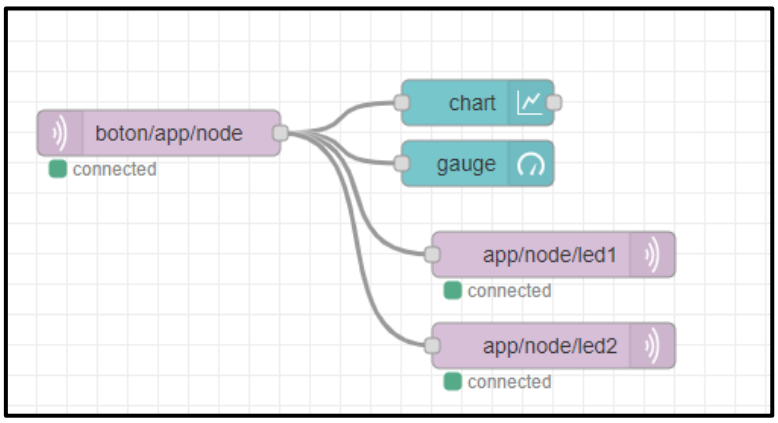
Elaborado por: Wilmer Caiza y Christian Herrera.

Para conectar los nodos se utiliza el protocolo MQTT que permitirá escuchar a través de mensajes enviados a la red todos los datos de entrada o salida. Los nombres que se

utilizaron en los topic son los siguientes: “temperatura/esp8266/node”, “humedad/esp8266/node”, “Ph/esp8266/node”.

En la Figura 4.2 se indica los nodos utilizados para la adquisición y envió de datos que serán usados por la aplicación móvil.

Figura 0.2 Nodos para datos desde la aplicación móvil

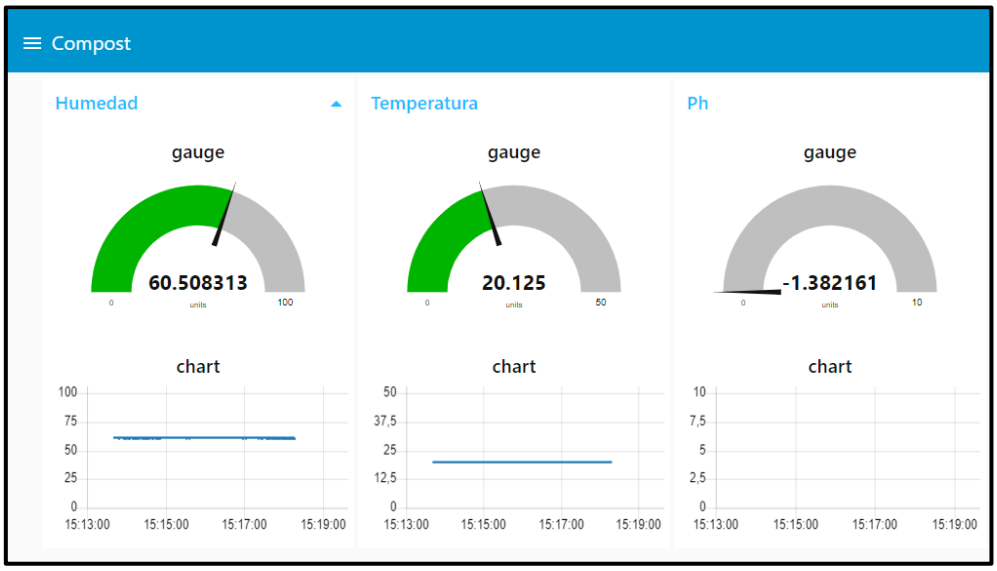


Elaborado por: Wilmer Caiza y Christian Herrera.

En la Figura 4.3 se muestra los valores de temperatura, humedad y pH vistos en la interfaz gráfica del computador que indican los datos generados por los tres sensores instalados en el interior del prototipo. El monitoreo se lo puede hacer mediante el Dashboard en tiempo real, además de poder revisar cada variable en un determinado intervalo de tiempo creando un plano donde la variable física medida dibujada .

Las variables físicas permiten identificar los valores mínimos y máximos con un indicador llamado “GAUGE” según la configuración que se indica en la figura 4.3 junto con sus valores en tiempo real.

Figura 0.3 Interfaz gráfica vista mediante un ordenador

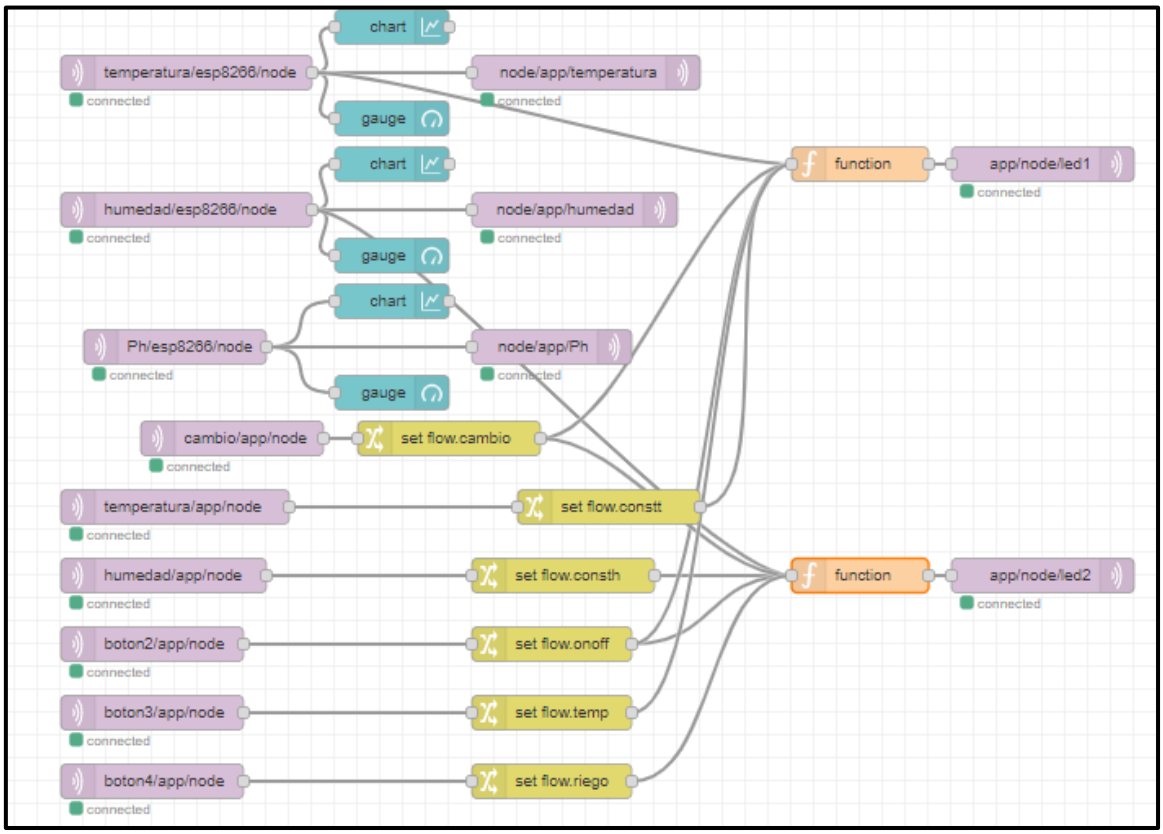


Elaborado por: Wilmer Caiza y Christian Herrera.

En la figura 4.4 se indica todos los nodos generados para el correcto funcionamiento del sistema inalámbrico los nodos temperatura/esp8266/node, humedad/esp8266/node, Ph/esp8266/node son nodos de entrada a la programación, estos recogen los datos generados por los sensores para ser mostrados en la pantalla del ordenador mediante la herramienta Dashboard propia de Node-Red además la salida de los datos siempre va ser escuchada por medio del puerto 1883 teniendo en cuenta la IP que sea generada por la red.

Los nodos app/node/led1, app/node/led2 son salidas de datos para la aplicación previamente el acondicionamiento de los valores mediante el nodo de función, los nodos set Flow son valres que permiten tener un dato de cero o uno para activar los actuadores este también permite el cambio de formato msg.payload a un Flow para programar en Java pues van a ser utilizados para setear los cambios de estado desde la App móvil.

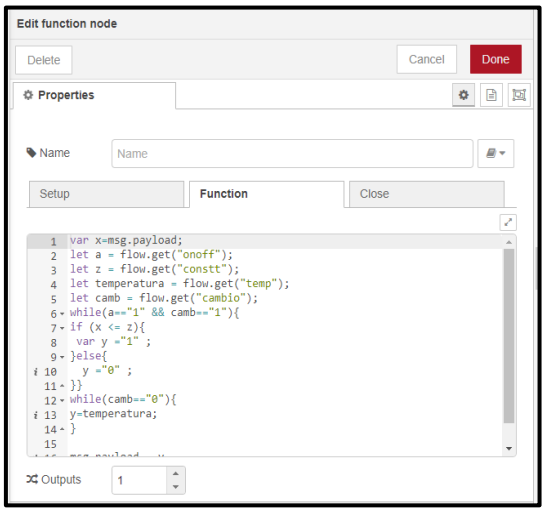
Figura 0.4 Nodos generados para el sistema inalámbrico.



Elaborado por: Wilmer Caiza y Christian Herrera.

Uno de los nodos con mayor importancia es el nodo función en el cual es programado en lenguaje Java para manipular los valores de una variable y transformar el formato de salida a ser utilizado en la aplicación móvil para el cambio de estado de automático a manual.

Figura 0.5 Nodo función.

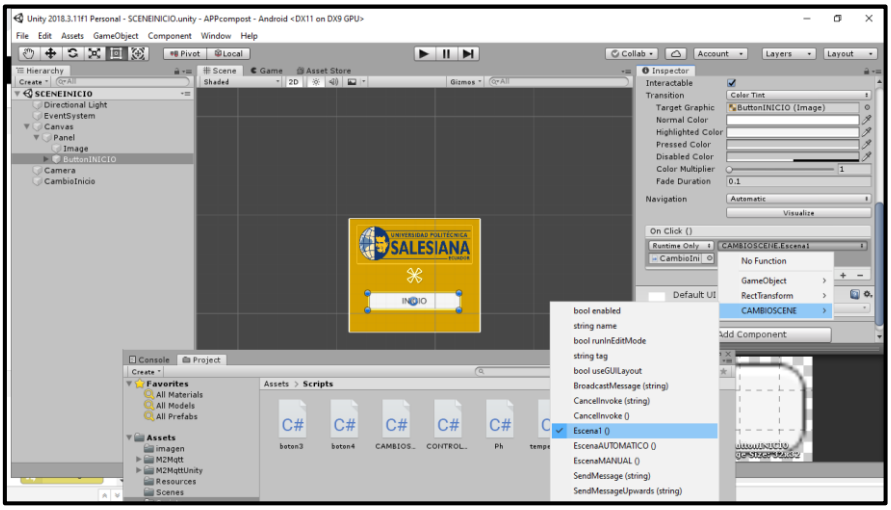


Elaborado por: Wilmer Caiza y Christian Herrera.

2.17 Elaboración de escenarios para la aplicación móvil en el software Unity

En el software Unity se realizaron 4 escenarios que van a ser parte de la aplicación dando un mensaje de bienvenida con un botón de inicio, cada escenario es desarrollado en un script dentro de la opción Canvas ya que esta genera un área de trabajo donde se anidaran todos los elementos que se crean necesarios en el escenario que se esté desarrollando como pueden ser iconos, imágenes o valores que se vayan a incluir en la portada.

Figura 0.6 Desarrollo de la portada en el software Unity.



Elaborado por: Wilmer Caiza y Christian Herrera.

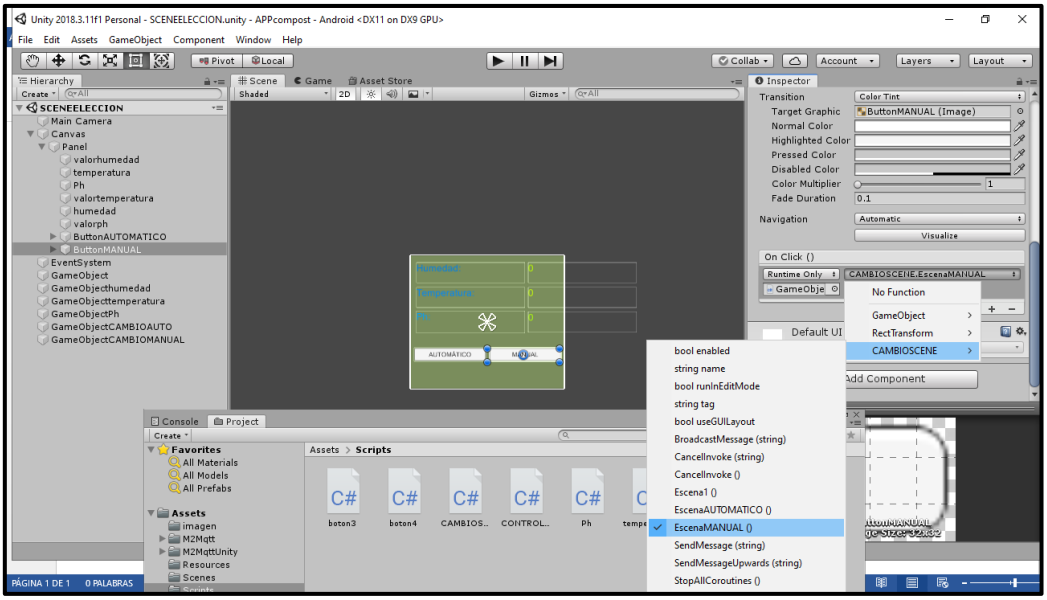
En la Figura 4.7 se muestra la portada generada para la aplicación móvil con una imagen referente a la Universidad Politécnica Salesiana con un botón de inicio el cual mostrara el siguiente escenario.

Figura 0.7 Portada de inicio para la aplicación móvil

Elaborado por: Wilmer Caiza y Christian Herrera.

Para la siguiente escena se adicionará al Canvas los valores de temperatura, humedad y pH medidos por los sensores en tiempo real, además se observan dos botos con un modo automático y manual que posteriormente llevara a otros escenarios teniendo en cuenta los eventos previamente programados en cada botón.

Figura 0.8 Desarrollo de Escena 1 en el software Unity.



Elaborado por: Wilmer Caiza y Christian Herrera.

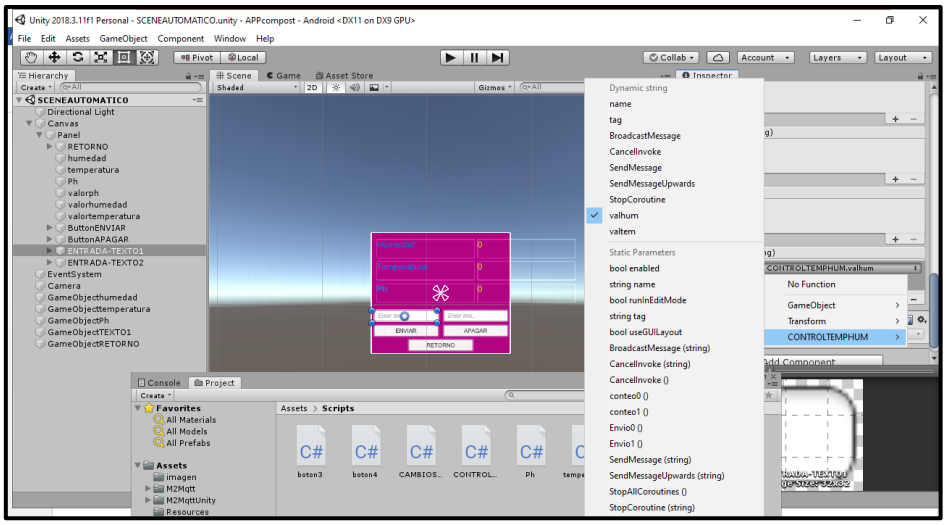
En la Figura 4.9 se indica la pantalla mostrada en la Aplicación móvil con los valores de Humedad Temperatura y pH medidos desde el prototipo hay que recordar que los valores de humedad se dan del 0 al 100 por lo que los valores vistos se van a ir adecuando dependiendo de los valores que sean necesarios para obtener un producto de mayor calidad a un menor tiempo.

Figura 0.9 Escena 1 visualización de valores de las variables en tiempo real.

Elaborado por: Wilmer Caiza y Christian Herrera.

En la escena 2 se despliega al entrar al modo automático el cual muestra los valores de humedad temperatura y pH además de tres botones que son retorno, envió y apagado asi también se tiene casilleros para un texto de entrada para setear los valores de temperatura y humedad siendo accionados los actuadores que generan temperatura y riego dentro del prototipo de manera automática.

Figura 0.10 Creación de escena 2 para valores predeterminados.



Elaborado por: Wilmer Caiza y Christian Herrera.

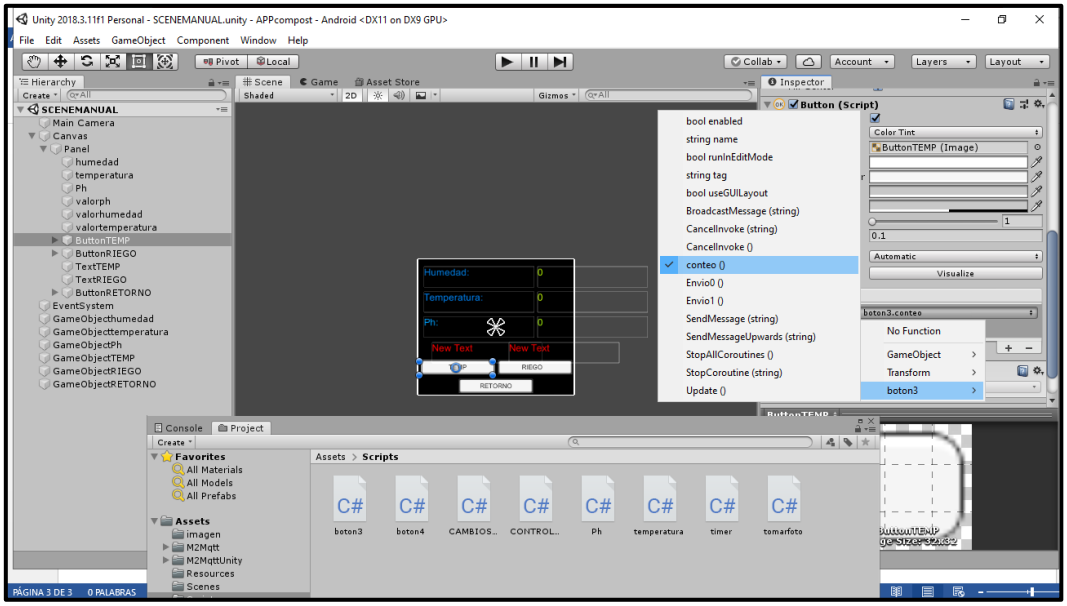
Para la Figura 4.11 se tiene una escena donde se pueden escribir los valores de temperatura y humedad mismos que realizaran el control hasta llegar a los valores introducidos desde la aplicación móvil.

Figura 0.11 Vista de la escena 2 para el modo automático.

Elaborado por: Wilmer Caiza y Christian Herrera.

En la construcción de la escena 3 se tiene los datos de las 3 variables a este modo se ingresa pulsando botón manual, en este escenario se puede prender y apagar los actuadores directamente y se observara los cambios de temperatura y humedad dependiendo de los niveles a los que se necesiten llegar.

Figura 0.12 Elaboración de escenario 3 para el modo manual.



Elaborado por: Wilmer Caiza y Christian Herrera.

La interfaz de monitoreo para el prototipo encuentra un histograma de las variables monitoreadas, los datos mostrados en la Dashboard serán mostrados en la Aplicación móvil como se indica en la Figura 4.13 donde se ven reflejados los valores de las 3 variables. De este modo se muestra que la aplicación está funcionando de manera correcta.

Figura 0.13 Monitoreo de las variables visto desde la aplicación móvil.

Elaborado por: Wilmer Caiza y Christian Herrera.

El proceso de implementación es realizado en la maqueta la cual es capaz de almacenar el compost para producir la transformación de residuos orgánicos y tratarlo para obtener un producto de calidad.

Figura 0.14 Prototipo armado para compost

Elaborado por: Wilmer Caiza y Christian Herrera.

2.18 Latencia con respecto a la aplicación móvil

Con respecto a la latencia se entiende el tiempo que va tardar llegar un paquete de datos desde la aplicación hasta los elementos de accionamientos o sensores, mientras menor sea la latencia la respuesta será más rápida como se observa en la Figura 4.15 mediante el comando ping desde un ordenador se muestra el tiempo que se demora en enviar un mensaje hasta el Gateway y en la Figura 4.16 el tiempo que demora en llegar un mensaje hasta el host que en este caso es el ESP8266.

Figura 0.15 Ping con el Gateway

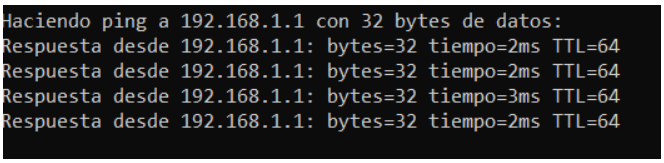
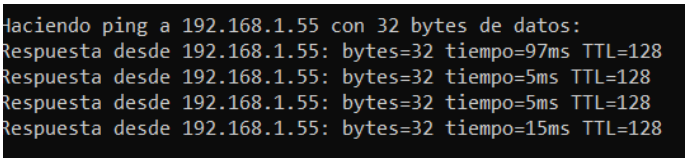


Figura 0.16 Ping con el host



2.19 Análisis de resultados para compost

El proceso de descomposición tiene mucho que ver con el tipo de material que se ingrese en el contenedor para el análisis de resultados, se toma material orgánico y tierra que comúnmente se puede obtener en cualquier casa o invernadero, se utilizaron 10 kilogramos de material para el proceso con una relación 40% de material orgánico y 60% tierra relativamente teniendo en cuenta que el material orgánico idóneo para el proceso de compostaje son resto de verduras y frutas, abonos verdes, cascaras de huevos, hojas secas permitiendo la degradación microbiana controlada.

- En la primera columna se detalla el número de procesos realizados para la recolección de datos.
- La segunda columna hace referencia a los días que el producto está en el contenedor.
- Con respecto al volteo se da 2 veces en la semana pues hay que mezclar la materia orgánica para tener un producto más uniforme.

- La columna temperatura, hace referencia a un intervalo entre (60°C a 70°C) óptimo de trabajo aplicado a la producción de compostaje según (Méndez, 2018).
- Columna humedad, hace referencia a un intervalo entre (40% a 70%) óptimo de trabajo aplicado a la producción de compostaje según (Méndez, 2018).
- Columna pH son los valores dados y recogidos a los que fue sometido el compost dentro del contenedor.

Se debe tener en cuenta que se realiza un análisis para 2 variables (temperatura y humedad) de las cuales se toman diferenciales de 5°C y 10% tanto de temperatura como de humedad respectivamente teniendo en cuenta los intervalos óptimos de trabajo para el compostaje, estos diferenciales se toman pues los valores de cambio deben ser representativos debido a que el valor de pH tiene un cambio mínimo, por lo tanto se realizan 10 procesos de medición en el prototipo cubriendo las diversas variaciones.
Se va tomar los 10 procesos como una población total para esto se realizará el cálculo de muestras que se necesitan para obtener valores de confianza de un 95%.

$$n = \frac{Z^2 * p * q * N}{e^2(N-1) + z^2 * p * q}$$

$$n = 9.77$$

N= Población
n= muestra
p= Probabilidad a favor
q= Probabilidad en contra
z= nivel de confianza
e= error de muestra
El valor para una muestra con un error aproximado al 5% y un nivel de confianza de 95% es de 9.77 por lo que se sugiere tomar los 10 procesos en total para los respectivos cálculos.

Tabla 0.1 Recolección de datos para compostaje.

Proceso	Días	Volteo	Temperatura °C	Humedad %	pH
1	7	2	60	40	3,846
2	7	2	60	50	4,407
3	7	2	60	60	4,615
4	7	2	60	70	5,417
5	7	2	65	40	4,375
6	7	2	65	50	4,423
7	7	2	65	60	4,435
8	7	2	70	40	4,231
9	7	2	70	50	5,217
10	7	2	70	60	4,492

En la Tabla 4.1 se puede observar los datos recogidos en el proceso de compostaje se tomaron a 10 muestras cada uno en una semana a diferentes rangos de temperatura y humedad, el indicador que muestra un mejor producto es cuando el pH a alcanzado los 5.417 ya que el rango de descomposición y mayor higienización idealmente esta entre los 5.8 a 7.2 por lo que se puede decir que a una temperatura de 60 °C con una humedad relativa del 70% se puede obtener un compost de calidad en un corto periodo de tiempo estos datos fueron monitoreados en la aplicación móvil en tiempo real como se indica en el Anexo 12, Anexo 13.

Si el rango mínimo admisible para un producto de calidad es de 5.8 en adelante se propone encontrar el error porcentual según el valor que más se acerca a este siendo de 5.417.

$$\%e = \frac{valor\ ideal - valor\ medido}{valor\ ideal} * 100$$

$$\%e = \frac{5.8 - 5.417}{5.8} * 100$$

$$e = 6.60\%$$

2.19.1 Varianza y desviación estándar, precisión y exactitud del equipo

Se aplicará técnicas de dispersión para encontrar los valores que describan que tanto la exactitud y precisión que se obtuvo en las muestras, hay que tener en cuenta que se está aplicando a una muestra.

Es necesario obtener el valor promedio de los datos por lo que se procede a realizar el cálculo con la siguiente ecuación.

$$\bar{x} = \frac{\sum x}{n}$$

$$\bar{x} = 4.645$$

Dónde:

n= número de intentos.

x= dato medido en cada intento

$\bar{x}$ = media aritmetica

La varianza y desviación estándar mostrará que tanto se han alejado los datos del promedio, indicando la concentración y comportamiento de los mismos.

$$S^2 = \frac{\sum(x - \bar{x})^2}{n - 1}$$

$$S^2 = 0.284$$

$$S = 0.532$$

Dónde:

n= número de intentos.

x= dato medido en cada intento

S = desviaciòn estandar

La muestra de los datos es tomada de los 10 procesos realizados sabiendo que existen rangos de temperatura y humedad que, adecuados para obtener un compost de calidad, por lo que se ha ido variando tanto los niveles de temperatura como los de humedad con el fin de tener una lectura lo más cercana a los valores ideales, teniendo así el valor más cercano en el intento 4.

- Las 4 primeras muestras fueron sometidas a una temperatura de 60°C con una variación de humedad del 40% al 60%.

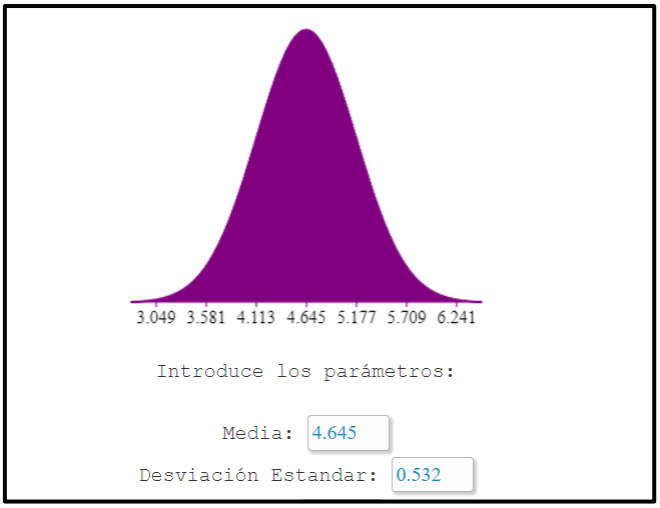
- La recolección de datos del intento 5 al 7 se aplicó una temperatura de 65°C con una variación de humedad del 40% al 60%.

- Los datos recogidos en del intento 8 al 10 fueron realizados con una temperatura aproximada a los 70 °C con una variación de humedad del 40% al 60%.

Durante el tiempo que el compost está en el contenedor se procede a voltear o mezclar para que este distribuido de una manera más uniforme, el prototipo facilita esto pues tiene un eje en el cual puede mover el contenedor haciendo que el material orgánico sea mezclado.

Para la interpretación de los valores de precisión y exactitud fueron necesarios los cálculos de la desviación estándar para indicar que tanto se han dispersado los datos con respecto al promedio indicado dando un valor de 4.645±0.532 con un rango de 68.3% según la campana de Gauss que se muestra en la Figura 4.15.

Figura 0.15 Campana de Gauss con parámetros (Media y desviación estándar)



Elaborado por: Wilmer Caiza y Christian Herrera.

CONCLUSIONES

- Se concluye que los rangos de funcionamiento de un dispositivo de compostaje son los siguientes, para la temperatura es necesario un aproximado de 60 °C a 70°C con una humedad del 40% a 60% y los valores de pH medidos están entre el 4,645 alcanzando un máximo de 5,417.

- Se concluye luego de la implementación del dispositivo de compostaje que es posible controlar la temperatura por medio del accionamiento de la resistencia calefactora y la humedad mediante el riego con una bomba instalada que puede ser accionada de forma manual y automática desde la aplicación móvil.

- Se concluye que la aplicación móvil puede ser instalada en un smartphone teniendo la capacidad de monitorear las variables de temperatura, humedad y pH en tiempo real, la aplicación consta de cuatro escenarios que indican un inicio, modo de selección automático en este se puede dar un valor de entrada tanto de temperatura como de humedad y un modo manual en el que se puede encender o apagar los elementos actuadores.

- Al realizar las pruebas se concluye que el compost al ser sometido a un cierto rango de temperatura y humedad este puede realizar su proceso de descomposición de una manera más rápida, además se podrá obtener un producto de calidad en un tiempo estimado de una semana aplicando una temperatura aproximada a los 60°C y un porcentaje de humedad del 70%.

RECOMENDACIONES

- El prototipo puede ser mejorado dependiendo de la necesidad, cantidad y volumen del producto que se necesite esto depende mucho del lugar donde se vaya a implementar por lo que puede ser realizado con equipos y materiales más robustos si es necesario y se tenga una exigencia mayor a la de un invernadero casero.

- La red inalámbrica es capaz de conectar múltiples sensores a la vez sin embargo las tarjetas controladoras ESP8266 solo poseen una entrada análoga por esto si se tienen varios sensores analógicos se necesitará un ESP8266 por sensor. El manejo remoto puede ser usado en todo el lugar sin embargo dependerá mucho de la intensidad de la señal por lo que a grandes distancias será necesario el uso de repetidores puesto que la red es local.

- La aplicación móvil está compuesta de varios escenarios los cuales se van desplegando dependiendo de la necesidad del usuario, es recomendable verificar el modelo del Smartphone pues puede ser que no cumpla con los requerimientos de la aplicación y por lo tanto no pueda ser instalada.

- Se recomiendo aplicar de 1 a 2 kg de compost por metro cuadrado al año esto depende del tipo de cultivo y el tipo de suelo ya que existen suelos erosionados donde es necesario concentrar el compost en áreas específicas.

REFERENCIAS BIBLIOGRÁFICAS

Aquino, J., Corona, L., & Fernández, C. (2017). *La Otra Mecatrónica Verde…La De Los Invernaderos*. 10. https://www.researchgate.net/profile/Jose-Aquino-10/publication/291972851_La_otra_mecatronica_verde_de_los_invernaderos/links/56a81c2408ae0fd8b3fe46f5/La-otra-mecatronica-verdela-de-los-invernaderos.pdf

Arroba, M. C. (2021). *Implementación de un sistema alterno para monitoreo de datos operativos con enfoque IIoT en la Central Hidroeléctrica Coca Codo Sinclair* [Universidad de las Fuerzas Armadas]. http://repositorio.espe.edu.ec/handle/21000/27325

Barrientos, E., Rico, D., coronel, L., & Cuesta, F. (2020). Análisis y control de humedad del suelo a través de un sistema soportado por sensores en el jardín botánico "Jorge Quintero Arenas." *Respuestas*, *25*(3), 14. https://doi.org/10.22463/0122820x.1796

Cevallos, B. L., & Rubio, S. W. (2021). Desarrollo de una red IoT con tecnología lora para gestión de invernaderos [Universidad Politécnica Salesiana]. In *Tesis*. http://dspace.ups.edu.ec/bitstream/123456789/5081/1/UPS-CYT00109.pdf

Changotasi, F. J., & Lechón, B. A. (2022). Diseño e implementación de un sistema de medición de variables climáticas e hidro-físicas mediante una red LPWAN para aplicaciones IoT de agricultura de precisión. 2005–2003, *8.5.2017*, הארץ.

Clavijo, C. A. (2021). Dosificador inteligente de balanceado con control Fuzzy y herramientas IoT para el galpón de gallinas ponedoras [Universidad Politécnica Salesiana]. In *Tesis*. http://dspace.ups.edu.ec/bitstream/123456789/5081/1/UPS-CYT00109.pdf

Coello, J. I., & Silva, D. A. (2020). *Diseño e implementación de un sistema de monitoreo en tiempo real de sensores de temperatura, turbidez, TDS y pH para la calidad del agua utilizando la tecnología Lorawan* [Universidad Politécnica Salesiana]. https://dspace.ups.edu.ec/bitstream/123456789/7986/1/UPS-CT004855.pdf

Dhall, R., & Solanki, V. (2017). An IoT Based Predictive Connected Car Maintenance Approach. *International Journal of Interactive Multimedia and Artificial Intelligence*, *4*(3), 8. https://doi.org/10.9781/ijimai.2017.433

Méndez, A., Robles, C., Ruiz, J., & Catañeda, E. (2018). Compostaje de residuos agroindustriales inoculados con hongos lignocelulósicos y modificación de la relación C/N. *Revista Mexicana de Ciencias Agrícolas*, *9*(2), 10. https://doi.org/10.29312/remexca.v9i2.1070

Pillajo, Carlos, and Roberto Hincapié, 2018. *WIRELESS NETWORK CONTROL SYSTEMS*. Vol.4.

Ramírez, W. G. (2020). Desarrollo una red de sensores que permita la monitorización de los niveles de contaminación mediante tecnología LPWAN. *Engineering, Construction and Architectural Management*, *25*(1), 1–9. http://dx.doi.org/10.1016/j.jss.2014.12.010%0Ahttp://dx.doi.org/10.1016/j.sbspro.2013.03.034%0Ahttps://www.iiste.org/Journals/index.php/JPID/article/viewFile/19288/19711%0Ahttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.6911&rep=rep1&type=pdf

Valencia, W. G., & Delgado, C. E. (2021). *Diseño e implementación de prototipo IoT para el monitoreo remoto de la calidad del agua para la crianza de tilapias en estanques.* [Universidad Politécnica Salesiana]. http://dspace.ups.edu.ec/handle/123456789/21427

ANEXOS

Anexo 1 Control de temperatura y humedad

CONTROL TEMPHUM

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;
using System.Threading;


public class CONTROLTEMPHUM : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;
    private string dat;
    /// <summary>
    /// ///
    /// </summary>
    // control de temperatura y humedad
    private string encendido = "1";
    private string valorhum = "0";
    private string valortem = "0";
    private int cont = 0;


    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos

    }

    public void valtem(string temperatura)
    {
        valorhum = temperatura;
    }

    public void valhum(string humedad)
    {
        valortem = humedad;
    }
```

```
    public void conteo1()
    {

        cont = 1;
    }
    public void conteo0()
    {
        cont = 0;
    }

    public void Envio1()
    {
        // se registra el mensaje que se resibe
        client.Publish("boton2/app/node",
System.Text.Encoding.UTF8.GetBytes(encendido), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos
        // se registra el mensaje que se resibe
        client.Publish("humedad/app/node",
System.Text.Encoding.UTF8.GetBytes(valorhum), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos
        // se registra el mensaje que se resibe
        client.Publish("temperatura/app/node",
System.Text.Encoding.UTF8.GetBytes(valortem), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos

    }

    public void Envio0()
    {
        // se registra el mensaje que se resibe
        client.Publish("boton2/app/node", System.Text.Encoding.UTF8.GetBytes("0"),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, true);
        // Se subscribe al topico hello y se obtiene los datos
        // se registra el mensaje que se resibe
        client.Publish("humedad/app/node", System.Text.Encoding.UTF8.GetBytes("0"),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, true);
        // Se subscribe al topico hello y se obtiene los datos
        // se registra el mensaje que se resibe
        client.Publish("temperatura/app/node",
System.Text.Encoding.UTF8.GetBytes("0"), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, true);
        // Se subscribe al topico hello y se obtiene los datos

    }


    public void Update()
    {

        // envio de uno y cero
        if (cont == 0)
        {
            Envio0();
            Debug.Log("cero");
        }
```

```
        else if (cont == 1)
        {
            Envio1();
            Debug.Log("uno");
        }
        else if (cont >= 2)
        {
            cont = 0;

        };
        Thread.Sleep(1000);
    }
}
```

```
           else if (cont == 1)
           {
               Envio1();
               Debug.Log("uno");
           }
           else if (cont >= 2)
           {
               cont = 0;

           };
           Thread.Sleep(1000);
       }
   }
```

Anexo 2 Programación para uso de botones

BOTON4

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;
using System.Threading;

public class boton4 : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;
    private string dat;
    /// <summary>
    /// ///
    /// </summary>

    private string encendido = "1";
    private string apagado = "0";
    private int cont = 0;
    public Text textRIEGO;
    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos



    }

    public void conteo()
    {




        cont = cont + 1;
    }
    public void Envio1()
    {
        // se registra el mensaje que se resibe
```

```
        client.Publish("boton4/app/node",
System.Text.Encoding.UTF8.GetBytes(encendido), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos
        textRIEGO.text = "ON";
    }

    public void Envio0()
    {
        // se registra el mensaje que se resibe
        client.Publish("boton4/app/node",
System.Text.Encoding.UTF8.GetBytes(apagado), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos
        textRIEGO.text = "OFF";
    }
    public void Update()
    {

        if (cont == 0)
        {
            Envio0();
            Debug.Log("cero");
        }
        else if (cont == 1)
        {
            Envio1();
            Debug.Log("uno");
        }
        else if (cont >= 2)
        {
            cont = 0;

        };
        Thread.Sleep(1000);
    }
}
```

BOTON3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;
using System.Threading;

public class boton3 : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;
    private string dat;
    /// <summary>
    /// ///
    /// </summary>

    private string encendido = "1";
    private string apagado = "0";
    private int cont = 0;
    public Text textTEMP;
    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos



    }

    public void conteo()
    {

        cont = cont + 1;
    }
    public void Envio1()
    {
        // se registra el mensaje que se resibe
        client.Publish("boton3/app/node",
System.Text.Encoding.UTF8.GetBytes(encendido), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos
```

```
        textTEMP.text = "ON";
    }

    public void Envio0()
    {
        // se registra el mensaje que se resibe
        client.Publish("boton3/app/node",
System.Text.Encoding.UTF8.GetBytes(apagado), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE,
true);
        // Se subscribe al topico hello y se obtiene los datos
        textTEMP.text = "OFF";
    }
    public void Update()
    {

        if (cont == 0)
        {
            Envio0();
            Debug.Log("cero");
        }
        else if (cont == 1)
        {
            Envio1();
            Debug.Log("uno");
        }
        else if (cont >= 2)
        {
            cont = 0;

        };
        Thread.Sleep(1000);
    }
}
```

Anexo 3 Lectura de temperatura

TEMPERATURA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;


public class temperatura : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;
    private string dat;
    /// <summary>
    /// ///
    /// </summary>
    public Text valortemperatura;
    public String temp = "";
    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        client.MqttMsgPublishReceived += client_MqttMsgPublishReceived;
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos
        client.Subscribe(new string[] { "node/app/temperatura" }, new byte[] {
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
    }
    void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
    {
        Debug.Log("Received: " + System.Text.Encoding.UTF8.GetString(e.Message));
        dat = System.Text.Encoding.UTF8.GetString(e.Message);
        // tiempo = Convert.ToInt64(dat);//se convierte en dato a variable int
        temp = dat;
    }


    public void Update()
    {

        valortemperatura.text = temp;
    }

}
```

Anexo 4 Lectura de humedad

HUMEDAD

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;


public class timer : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;
    private string dat;
    /// <summary>
    /// ///
    /// </summary>
    public Text valorhumedad;
    public String humedad = "";
    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        client.MqttMsgPublishReceived += client_MqttMsgPublishReceived;
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos
        client.Subscribe(new string[] { "node/app/humedad" }, new byte[] {
        MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
    }
    void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
    {
        Debug.Log("Received: " + System.Text.Encoding.UTF8.GetString(e.Message));
        dat = System.Text.Encoding.UTF8.GetString(e.Message);
        // tiempo = Convert.ToInt64(dat);//se convierte en dato a variable int
        humedad = dat;
    }


    public void Update()
    {
        valorhumedad.text = humedad;
    }

}
```

Anexo 5 Lectura de PH

PH

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;


public class Ph : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;
    private string dat;
    /// <summary>
    /// ///
    /// </summary>
    public Text valorph;
    public String ph = "";
    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        client.MqttMsgPublishReceived += client_MqttMsgPublishReceived;
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos
        client.Subscribe(new string[] { "node/app/Ph" }, new byte[] {
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
    }
    void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
    {
        Debug.Log("Received: " + System.Text.Encoding.UTF8.GetString(e.Message));
        dat = System.Text.Encoding.UTF8.GetString(e.Message);
        // tiempo = Convert.ToInt64(dat);//se convierte en dato a variable int
        ph = dat;
    }


    public void Update()
    {

        valorph.text = ph;
    }}
```

Anexo 6 Programación para cambios de escena

CAMBIO DE ESCENA

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System.IO;
using System.Net;
using System;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using uPLibrary.Networking.M2Mqtt.Utility;
using uPLibrary.Networking.M2Mqtt.Exceptions;
using System.Threading;

public class CAMBIOSCENE : MonoBehaviour
{
    private const int V = 1883;
    private const string IpString = "192.168.61.177";

    // Declaramos variables privadas para el cliente MQTT
    private MqttClient client;

    // Start is called before the first frame update
    void Start()
    {
        // Creamos la instancia del cliente
#pragma warning disable CS0618 // El tipo o el miembro están obsoletos
        client = new MqttClient(IPAddress.Parse("192.168.61.177"), 1883);
#pragma warning restore CS0618 // El tipo o el miembro están obsoletos
        // se registra el mensaje que se resibe
        string clientId = Guid.NewGuid().ToString();
        client.Connect(clientId);
        // Se subscribe al topico hello y se obtiene los datos
    }
    public void Escena1()
    {
        SceneManager.LoadScene("SCENEELECCION", LoadSceneMode.Single);
    }
    public void EscenaAUTOMATICO()
    {
        SceneManager.LoadScene("SCENEAUTOMATICO", LoadSceneMode.Single);
        client.Publish("cambio/app/node", System.Text.Encoding.UTF8.GetBytes("1"),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, true);
    }
    // Update is called once per frame
    public void EscenaMANUAL()
    {
        SceneManager.LoadScene("SCENEMANUAL", LoadSceneMode.Single);
        client.Publish("cambio/app/node", System.Text.Encoding.UTF8.GetBytes("0"),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, true);
    }
    void Update()
    {

    }
}
```

Anexo 7 Código de programación para ESP8266

ARDUINO TEMPERATURA Y HUMEDAD

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define ALTITUDE 1655.0 // Altitude of SparkFun's HQ in Boulder, CO. in meters

// Update these with values suitable for your network.
const char* ssid = "Bryan's Galaxy A51";
const char* password = "qwgq2017";
const char* mqtt_server = "192.168.61.177";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
char msg3[50];
char msg4[50];
const char* msg1="1";
const char* msg2="0";
int value = 0;
const int led=14;//d5
const int led1=13;//d6
const int analoghumedad = A0; //inicializar analogico
int sensorValue=0 ;
double escalamiento=0;
 double T1,P,p0,a,T;
 const int pinDatosDQ = 12;
OneWire oneWireObjeto(pinDatosDQ);
DallasTemperature sensorDS18B20(&oneWireObjeto);
```

```
void setup() {

 pinMode(BUILTIN_LED, OUTPUT);    // Initialize the BUILTIN_LED pin as an output
 pinMode(led,OUTPUT);
 pinMode(led1,OUTPUT);
//  pinMode(boton1, INPUT);

 Serial.begin(115200);
 setup_wifi();
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);
 sensorDS18B20.begin();

}


void setup_wifi() {
 delay(10);
 // We start by connecting to a WiFi network
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);


 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
```

```
 }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Message arrived [");
 Serial.print(topic);
 Serial.print("] ");
 for (int i = 0; i < length; i++) {
  Serial.print((char)payload[i]);
 }
 Serial.println();

 // Switch on the LED if an 1 was received as first character
 if ((char)payload[0] == '1') {
  digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
  // but actually the LED is on; this is because
  // it is acive low on the ESP-01)
 digitalWrite(led, HIGH);
 Serial.print("   encendido");
 }

 else {
  digitalWrite(BUILTIN_LED, HIGH);// Turn the LED off by making the voltage HIGH
  digitalWrite(led, LOW);
  Serial.print("   APAGADO");
```

```
  }
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP82661")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("esp2", "Enviando el primer mensaje");
      // ... and resubscribe
      client.subscribe("app/node/led1");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}




void loop() {
```

```
 sensorDS18B20.requestTemperatures();

Serial.print("Temperatura sensor 1: ");
Serial.print(sensorDS18B20.getTempCByIndex(0));
Serial.print(" C");
T=sensorDS18B20.getTempCByIndex(0);

 if (!client.connected()) {
  reconnect();
 }
 client.loop();
 long now = millis();
 if (now - lastMsg > 500) {
  lastMsg = now;

  ++value;
 // snprintf (msg, 75, "hello #%ld", value);
  //Serial.print("Publish message: ");
  Serial.println(msg);
  //client.publish("casa/despacho/temperatura", msg);

    sensorValue = analogRead(analoghumedad);
    escalamiento=(-0.0977517*sensorValue)+100;

//sensorValue = 666;
    snprintf (msg3, 75, "%lf", T );
    snprintf (msg4, 75, "%lf", escalamiento );
```

```
Serial.print("Publish message: ");
Serial.println(sensorValue);

 client.publish("temperatura/esp8266/node", msg3);
 client.publish("humedad/esp8266/node", msg4);

//  buttonState = digitalRead(boton1);
//  if (buttonState == HIGH) {
//    // turn LED on:
//   client.publish("casa/despacho/temperatura", msg1);
//
// }
//
// if (buttonState == LOW) {
//    // turn LED on:
//     client.publish("casa/despacho/temperatura", msg2);
//
//   }
  }
}
```

ARDUINO DE PH

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define ALTITUDE 1655.0 // Altitude of SparkFun's HQ in Boulder, CO. in meters

// Update these with values suitable for your network.
const char* ssid = "Bryan's Galaxy A51";
const char* password = "qwgq2017";
const char* mqtt_server = "192.168.61.177";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
char msg3[50];
char msg4[50];
const char* msg1="1";
const char* msg2="0";
int value = 0;
const int led=14;//d5
const int led1=13;//d6
const int pHSense = A0; //inicializar analogico
int sensorValue=0 ;
double escalamiento=0;
int samples = 10;
float adc_resolution = 1024.0;
```

```
void setup() {

 pinMode(BUILTIN_LED, OUTPUT);    // Initialize the BUILTIN_LED pin as an output
 pinMode(led,OUTPUT);
 pinMode(led1,OUTPUT);
//  pinMode(boton1, INPUT);

 Serial.begin(115200);
 setup_wifi();
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);
}


void setup_wifi() {
 delay(10);
 // We start by connecting to a WiFi network
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);


 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
```

```
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Message arrived [");
 Serial.print(topic);
 Serial.print("] ");
 for (int i = 0; i < length; i++) {
  Serial.print((char)payload[i]);
 }
 Serial.println();
 // Switch on the LED if an 1 was received as first character
 if ((char)payload[0] == '1') {
  digitalWrite(BUILTIN_LED, LOW);   // Turn the LED on (Note that LOW is the voltage level
  // but actually the LED is on; this is because
  // it is acive low on the ESP-01)
 digitalWrite(led, HIGH);
 Serial.print("   encendido");
 }


 else {
  digitalWrite(BUILTIN_LED, HIGH);// Turn the LED off by making the voltage HIGH
  digitalWrite(led, LOW);
  Serial.print("   APAGADO");
```

```
 }
}


void reconnect() {
 // Loop until we're reconnected
 while (!client.connected()) {
  Serial.print("Attempting MQTT connection...");
  // Attempt to connect
  if (client.connect("ESP82662")) {
   Serial.println("connected");
   // Once connected, publish an announcement...
   client.publish("esp2", "Enviando el primer mensaje");
   // ... and resubscribe
   client.subscribe("app/node/led2");
  } else {
   Serial.print("failed, rc=");
   Serial.print(client.state());
   Serial.println(" try again in 5 seconds");
   // Wait 5 seconds before retrying
   delay(5000);
  }
 }
}
float ph (float voltage) {
   return 7 + ((2.5 - voltage) / 0.18);
}

void loop() {
```

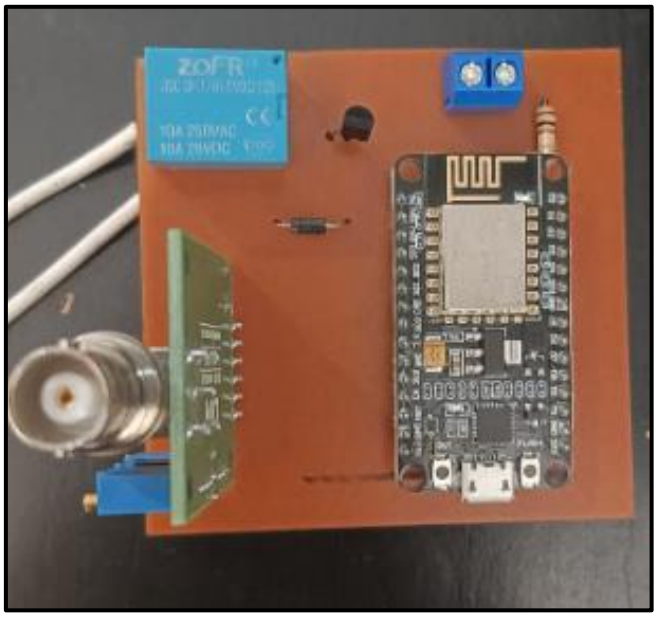
```
if (!client.connected()) {
  reconnect();
}
client.loop();
long now = millis();
if (now - lastMsg > 500) {
  lastMsg = now;

  ++value;
  // snprintf (msg, 75, "hello #%ld", value);

  //Serial.print("Publish message: ");

  Serial.println(msg);
  //client.publish("casa/despacho/temperatura", msg);

   int measurings=0;
  for (int i = 0; i < samples; i++)
  {
     measurings += analogRead(pHSense);
     delay(10);
  }
  float voltage = 5 / adc_resolution * measurings/samples;
//sensorValue = 666;
    snprintf (msg3, 75, "%lf", ph(voltage) );
Serial.print("Publish message: ");
Serial.println(ph(voltage));

client.publish("Ph/esp8266/node", msg3);
```

```
if (!client.connected()) {
  reconnect();
}
client.loop();
long now = millis();
if (now - lastMsg > 500) {
  lastMsg = now;

  ++value;
 // snprintf (msg, 75, "hello #%ld", value);

  //Serial.print("Publish message: ");

  Serial.println(msg);
  //client.publish("casa/despacho/temperatura", msg);

   int measurings=0;
  for (int i = 0; i < samples; i++)
  {
     measurings += analogRead(pHSense);
     delay(10);
  }
  float voltage = 5 / adc_resolution * measurings/samples;
//sensorValue = 666;
    snprintf (msg3, 75, "%lf", ph(voltage) );
Serial.print("Publish message: ");
Serial.println(ph(voltage));

 client.publish("Ph/esp8266/node", msg3);
```

Anexo 8 Elaboración de Placas

Anexo 9 Placa de para control de humedad y temperatura

Anexo 10 Placa de control para pH.



Anexo 11 Conexión de placas de control en el prototipo.

Anexo 12 Medición de variables en tiempo real (4 intento)

Anexo 13 Medición de variables en tiempo real (9 intento)