El sensor de humedad y temperatura es un dispositivo increíblemente útil tanto para uso industrial como doméstico. Este sensor permite medir la temperatura y la humedad, y tiene una pantalla que muestra la información en tiempo real. Toda la línea de sensores es una gran herramienta para automatizar otros dispositivos.

Un sensor de humedad es un aparato que permite detectar y controlar el porcentaje de agua del aire o de cualquier material o superficie. Su nombre técnico es higrómetro y resulta un elemento indispensable en meteorología.

El sensor de humedad se usa cada vez más en el sector de la técnica de calefacción, ventilación y climatización, así como en los procesos de producción que requieren un control de la humedad. Con frecuencia, además de medir la humedad, también es necesario medir la temperatura.

A continuación se describe un proyecto de aplicación practica

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MONITOREO DE TEMPERATURA, HUMEDAD Y PRESIÓN CON COMUNICACIÓN VÍA RADIO E INTERNET PARA MEJORAR LA PRODUCCIÓN AGRÍCOLA

Objetivo específico

- Medir la temperatura, humedad y presión usando sensores y microcontroladores en un determinado terreno.
- Registrar las medidas obtenidas de los sensores en la base de datos.
- Mostrar a los usuarios la información histórica y actual del dispositivo a través de una página web.

- Posibilitar que otros profesionales realicen cálculos de los datos almacenados y descargados de la página web.

1 Microcontroladores y Sensores

1.1 Microcontroladores

Es un dispositivo electrónico programable capaz de realizar múltiples tareas, tiene integrado un CPU, memorias ROM, RAM, puertos INPUT & OUTPUT (Entrada & Salida), entradas analógicas, entradas digitales, salidas analógicas, salidas digitales, comunicación serial, comunicación I2C, entre otras especificaciones, (Sherlin, 2021), (Smelpro, 2021).

1.2 Aplicaciones de Microcontroladores

Las aplicaciones para los microcontroladores son muy diversas con posibilidades en crecimiento, actualmente los dispositivos digitales satisfacen muchas necesidades lo cual demanda que se hagan avances más rápidos en la tecnología y dispositivos electrónicos, los usuarios y consumidores cada vez son más exigentes y siempre están en busca de innovación.

1.2.1 Industrias con Microcontroladores

Las industrias que trabajan con microcontroladores son, (Microcontraladoressesv, 2021), (Hetpro, 2021):

Dispositivos médicos, Instrumentación, juguetes, sistemas de control, sistemas de medición, robótica, automatización, domótica, automotriz, IoT Internet of Things (Internet de las cosas).

1.3 Sensores

Son dispositivos que convierten magnitudes físicas o químicas en magnitudes eléctricas, es decir, detectan cambios en el campo para después captar esa información y la transforman en un impulso eléctrico que luego pasará a ser manipulado para la aplicación designada.

1.4 Algunos Tipos de Sensores

Los tipos de sensores pueden ser, (Ingeniería Mecafenix, 2021), (Dewesoft, 2021), (Recursostic, 2021):

De contacto, óptico, térmico, humedad, magnéticos, infrarrojos, micrófonos, aceleración.

1.5 Aplicaciones de Sensores

Las aplicaciones de sensores van a depender de la tarea que se requiera realizar, teniendo un sensor de temperatura se puede medir calor, con un sensor de proximidad se pueden medir distancias, con un sensor PIR se puede medir la radiación de luz infrarroja, etc. Gracias a esto se descubren más posibilidades e incluso llegar a acoplar diferentes sensores para cumplir múltiples tareas.

2 Protocolos de comunicación (Microcontroladores)

2.1 Protocolos de Comunicación

2.1.1 Comunicación en Paralelo

Transmite datos bit por bit a través de canales, es decir, un dato para cada hilo, para lograr esta comunicación se requiere el mismo número de entradas como de salidas.

2.1.2 Comunicación en Serie o Serial

Tiene la ventaja de enviar datos a través de un solo canal (“bus” de datos), el método más aplicado es la conexión USB. Gracias a esta comunicación se pueden enviar datos a mayores distancias utilizando menos recursos.

2.1.3 Comunicación I2C (Inter-Integrated Circuit)

Protocolo de comunicación síncrono que funciona mediante el método Maestro & Esclavo cuyo bus de datos va a ser transmitido por un solo canal de modo bidireccional, los dispositivos I2C utilizan 2 canales para transmitir la información y 1 canal para tierra.

2.1.4 Comunicación SPI

Protocolo de comunicación síncrono con la capacidad de enviar y recibir datos por el mismo canal, esta comunicación Maestro & Esclavo se caracteriza por tener un canal llamado SS (Slave Select) el cual se encarga de informar al esclavo cuando debe transmitir la información, (Tapia, 2021), (Concepción, 2021).

3 Protocolos de Comunicación (Redes)

3.1 TCP/IP

Establece la conexión entre los clientes y los servidores después de haber realizado varias solicitudes y se permita el acceso al servidor, para finalizar una conexión se debe enviar un mensaje de Salida o final para cerrar el socket destinado para establecer la conexión, en algunos casos puede existir una sesión abierta por parte del cliente, pero el socket para recibir esté cerrado esto implica que la información va a seguir enviándose y viceversa.

Debido a que TCP sirve a una gran cantidad de protocolos de la capa de aplicación, es fundamental que los datos lleguen correctamente al destinatario, sin errores, y, en orden. Si en la transmisión de los segmentos, se corrompiesen o perdiesen, automáticamente el protocolo TCP inicia la retransmisión, sin intervención de la capa de aplicación. De esta manera, se garantiza que los datos llegan al destinatario sin errores, ya que este protocolo se encarga de solucionar cualquier tipo de problema (De Luz, 2021).

3.2 HTTP

Protocolo de comunicación entre cliente web y servidor web, el cliente solicita acceso al servidor a través del buscador para luego mostrar la página principal con la que podrá interactuar con la información de cabecera.

Para establecer una conexión HTTP se realiza desde un navegador con la respectiva solicitud HTTP y por supuesto se debe esperar una respuesta HTTP. Existen varios lenguajes para desarrollo web, pero la ventaja es que todos van a estar basados en HTTP, por lo tanto, las solicitudes se codifican para que la respuesta sea la requerida (López Jurado, 2021).

3.3 FTP

Protocolo de comunicación que permite compartir archivos entre computadoras o entre cliente web y servidor, del mismo modo se establece una conexión entre el cliente y el servidor y con dicha información se permite el acceso a la descarga de archivos.

3.4 SMTP

Protocolo de comunicación que se encarga del servicio de correo electrónico, para lograr dicha comunicación se establece conexión entre el cliente y el servidor SMTP a través del puerto 25 para permitir el servicio de mensajería entre los 2 puntos.

3.5 UDP

El protocolo UDP permite el envío de datagramas sin necesidad de establecer previamente una conexión, pero con la desventaja de que no se asegura de que la conexión sea completa, por ello se puede perder parte del paquete, tan solo es necesario tener abierto un socket en el destino para que acepte los datagramas del origen.

UDP es un protocolo no orientado a conexión, es decir, no ocurre como en TCP donde hay una fase de establecimiento de la conexión, aquí directamente se envían sin establecimiento previo.

Este protocolo no proporciona ningún tipo de control de flujo, si un equipo es más rápido que otro y envía información, es muy posible que se pierda información debido a que colapsará al más lento, y tendremos que proceder al reenvío de la información. Un detalle importante es que la gestión de reenvío de los datagramas la realiza la capa de transporte, ya que UDP es muy simple y no dispone de mecanismos de control de reenvío de datagramas por haberse perdido.

Se utiliza para sesiones que no requieren de verificación y si se desea obtener la información en tiempo real (Vestertraining, 2021), (Fernandez, 2021), (Mastermoviles, 2021).

3.6 IoT (Internet of Things)

La definición de IoT podría ser la agrupación e interconexión de dispositivos y objetos a través de una red, dónde todos ellos podrían ser visibles e interaccionar. Respecto al tipo de objetos o dispositivos podrían ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos, cualquier cosa que se pueda imaginar podría ser conectada a internet e

interaccionar sin necesidad de la intervención humana, el objetivo por tanto es una interacción de máquina a máquina, o lo que se conoce como una interacción M2M (machine to machine).

Si se piensa en aplicaciones en el sector productivo, es usado ya en muchas plantas donde los dispositivos y sensores conectados a la red permiten analizar los datos y generar alarmas y mensajes que son enviados a los distintos usuarios para que tomen las acciones necesarias o incluso iniciar protocolos de actuación de forma automática, sin interacción humana, para corregir o tratar dichas alarmas.

En aplicación del sector ganadero dónde la monitorización biométrica y la geolocalización es un factor que ayuda a los ganaderos a que sus animales estén siempre controlados.

Las aplicaciones que se le dan son muy diversas mientras se requiera obtener o transferir datos, estos datos luego estarán disponibles para que el usuario los manipule como le parezca ya sea usándolos en una base de datos, usarlos para controlar otros dispositivos incluso de forma remota. El sector ganadero dónde se realiza la monitorización biométrica y la geolocalización es un factor que ayuda a los ganaderos a que sus animales estén siempre controlados (Gracia, s.f.).

4 Comunicación MQTT

4.1 Protocolo MQTT

Es un protocolo de comunicación M2M (Machine to machine) se basa en el modelo cliente/servidor para enviar y recibir mensajes. Los clientes se conectan a un nodo central llamado “Broker” el cual contiene la información requerida.

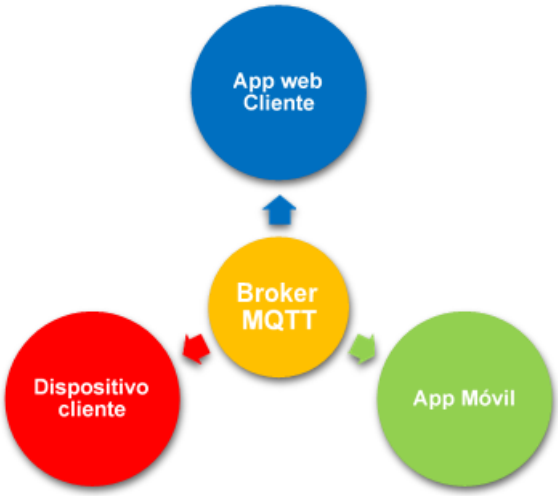

Figura 1. Broker MQTT

Este sistema funciona con jerarquías, gracias a esta estructura se puede filtrar la información que el usuario (cliente o dispositivo) desea recibir. El usuario se subscribe a un “topic” (tema) para poder recibir el mensaje, es decir, que dará prioridad una vez que el usuario se suscribe.

Los usuarios se conectan al broker, esta sesión se mantiene abierta hasta que el usuario la cierra o el tiempo de sesión abierta se agote (Llamas, 2020).

Figura 2. Tópicos MQTT

En cada grupo tenemos diferentes servicios (topics) en los cuales se requiere mantener el control, entonces cada piso va a disponer de un dispositivo diferente que se encargue de monitorear cada servicio, por supuesto se puede disponer de un mismo dispositivo que pueda controlar todos los servicios a la vez. Lo que se busca es obtener y almacenar los datos producidos por el uso de dispositivos para su posterior análisis.

Un usuario se subscribe al tópico, por ejemplo, usuario 1 se subscribe a grupo 1, porque ese usuario requiere de datos que pertenecen a ese grupo, luego deberá estar subscrito a un subtópico, por ejemplo, “seguridad” y finalmente se subscribe a “Ingreso a nivel”

El broker MQTT publica el mensaje en la dirección:

Tópicos/Grupo1/Seguridad/Ingreso_a_nivel

Se debe ser preciso al asignar o subscribirse a la información que se desea recibir ya que pueden existir confusiones tales como:

Caso 1: Estar subscrito a “seguridad” y recibir tanto “ingreso a nivel” como “cámaras de seguridad”.

Caso 2: Estar subscrito a todo el grupo 1 y sólo requerir la información de un solo tópico.

Caso 3: Estar subscrito a todo el grupo 2 y recibir también la información de Grupo 1, ya que se reciben los mismos servicios.

Caso 4: Recibir información de todo el nivel de “seguridad” en ambos grupos tanto “ingreso a nivel” como “cámaras de seguridad”.

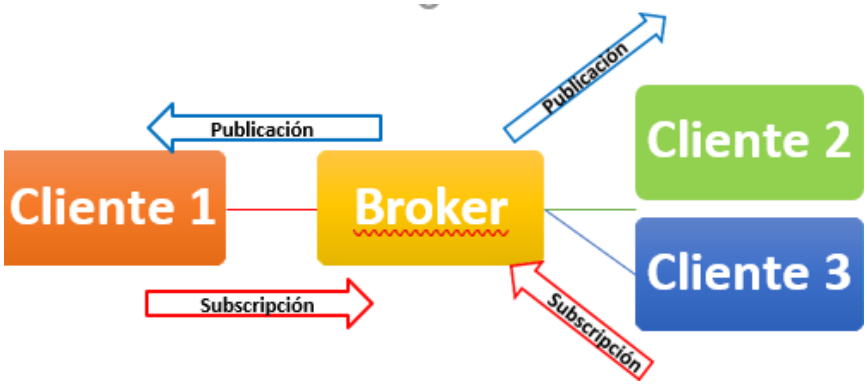

Figura 3. Comunicación Broker MQTT

4.2 MQTT y sensores

Un sensor no tiene la capacidad de computacional para enviar datos a una red o disparar alertas, este enfoque no es ocupación de los sensores ya que por lo general son dispositivos pequeños. El protocolo MQTT complementa las capacidades limitadas de los sensores, incluso de los más avanzados, para poder transmitir la data obtenida por los mismos e incluso llegar a automatizar procesos industriales de forma remota, (Valerie Lampkin, 2012).

Gracias al protocolo MQTT podemos alcanzar niveles de automatización muy avanzados como para acoplar bases de datos y otras herramientas de automatización que facilitan la obtención, guardado y consulta de información.

5 ESP32: Fundamentos y alcances

5.1 ESP32

Placa electrónica de desarrollo de grandes capacidades, el ESP32 ofrece la posibilidad de trabajar con WiFi, Bluetooth, BLE (Low Energy Bluettoth). Internamente posee varios periféricos: sensores táctiles, sensor de efecto Hall, amplificadores de bajo ruido, interfaz para tarjetas SD, Ethernet, SPI de alta velocidad, I2S e I2C, (Naylampmechatronics, 2020).

El ESP32 supera al ARDUINO UNO y a su antecesor el ESP8266 ya que viene a ser una versión más avanzada en la familia de microcontroladores Espressif Systems, con el ESP32 se busca desarrollar proyectos aún más complejos y poder aprovechar mejor las capacidades que se implementaron a este nuevo dispositivo (programarfacil, 2021).

6.1 LoRa

Tecnología inalámbrica de enlace de radio que utiliza modulación en radiofrecuencia. Esta tecnología también conocida como CSS (Chirp Spread Spectrum) se utiliza en la comunicación militar ya que se pueden aprovechar las largas distancias (por lo general kilómetros) para comunicarse.

Algunas ventajas de la tecnología LoRa:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos (-168dB)
- Bajo Consumo (hasta 10 años con una batería*)
- Largo alcance 10 a 20km
- Baja transferencia de datos (hasta 255 bytes)
- Conexión punto a punto
- Frecuencias de trabajo: 868 Mhz en Europa, 915 Mhz en América, y 433 Mhz en Asia

Esta tecnología también trabaja en conjunto con las redes IoT para explotar las utilidades de muchos dispositivos que tengan la capacidad de comunicarse de manera remota (alfaiot, 2021).

6.2 ESP32 LoRa

Es una versión del ESP32 que contiene un módulo LoRa además de las previas mejoras que se implementaron, esto permite tener más posibilidades de transmisión y recepción de información entre dispositivos que se comuniquen de manera remota y/o entre varios ESP32.

Para aplicaciones IoT se puede utilizar el manejo de información del ESP32 y utilizar el módulo LoRa para la transmisión de dicha información la cual luego podrá ser utilizada para otras aplicaciones o guardar dicha información en una nube.

Teniendo la tecnología LoRa se pueden implementar otros métodos de comunicación para no depender únicamente de internet para realizar múltiples tareas aprovechando los demás módulos implementados en el ESP32.

6.3 Bases de Datos

Se pueden definir como gestores de información recopilada y ordenada electrónicamente dicha información es captada y administrada por sistemas computacionales, una base de datos puede almacenar cualquier tipo de valor ya sea numérico, de tipo cadena, de tiempo, etc. Como la información es almacenada puede ser reutilizada para su posterior manipulación, (Olaya, 2021), (Valdés., 2021).

6.4 HeidiSQL

Es un software de código abierto que permite gestionar información con una amplia gama de herramientas como crear tablas, relacionar tablas, conectar múltiples servidores, comunicación SSH, otorgar acceso a usuarios, etc. Una importante característica de esta base de datos es que te permite editar la información guardada e incluso otorgar etiquetas para diferenciar variables, usuarios, datos, entre otros elementos, (Heidisql, 2021), (Aguilar., 2021).

6.5 EMQX

Es un motor de servicio MQTT escalable en tiempo real que permite la administración de miles de usuarios, tiene múltiples herramientas para administrar puertos de comunicación, dar acceso a usuarios y establecer comunicación entre cliente MQTT y el servidor web. En el panel principal podemos configurar a los puertos para siempre estén en modo escucha y estén pendientes de recibir y responder a las alertas o notificaciones de parte de los servicios que requieren comunicarse al servidor MQTT, (Chaparro., 2021), (Docs Emqx, 2021).

Los alcances que puede llegar a tener el ESP32 dependerán de las tareas que se desean realizar, la ventaja de utilizar el ESP32 es que permite realizar múltiples tareas y además se pueden implementar múltiples servicios para trabajar en conjunto tales como: obtener datos de un sensor, guardar información, mostrar información, hacer cálculos, conectar a servidores, entre otros y de ser necesario tener la posibilidad de realizar todo al mismo tiempo.

Fundamentación teórica

Los problemas que han investigado son las complicaciones que hay en supervisión de un terreno agrícola, dado las dificultades por las condiciones del suelo, no tomar medidas correspondientes porque las variables que necesitan ser atendidas no tienen una apreciación cuantitativa más la tienen cualitativa, por este motivo el usuario o trabajador agrícola no puede tomar medidas correctivas a tiempo, como aumentar la cantidad de riego, o saber si sus plantaciones tienden a enfermar.

Por este motivo se desarrolló un prototipo que pueda hacer las mediciones a una distancia de 20m (campo abierto) y de esa manera soluciona la problemática de ir a un lugar difícil de llegar a más de una ocasión y desgastar al usuario por gusto, también se le implemento una medición cuantitativa que se puede verificar en una página web con sus respectivas variables y de esa manera el usuario sabrá las acciones a realizar tanto para prevenir pérdidas como para tener una muestra cuantitativa y ver en la base de datos cómo va el desarrollo de su cultivo.

II. Marco metodológico

Para este proyecto se aplicó la metodología de Scrum, la cual consiste en trabajar en equipo sin llevar una planificación secuencial montando diferentes tareas para completar la meta final. Se planifica según los avances con muchas incertidumbres con una gran cantidad de cambios a última hora.

Para la investigación respectiva se aplicaron los siguientes métodos:

Método analítico

Estudio basado en la experimentación y la lógica empírica, descompone en elementos básicos los fenómenos que se estudian.

Método experimental

Conlleva la observación, manipulación y registro de variables que intervienen en el fenómeno de estudio. El objetivo de este método es descubrir las causas de dicho fenómeno.

Estructura del Proyecto

El proyecto trabaja en 4 niveles:

Nivel 1 Sensores & ESP32: Los módulos ESP32 se van a encargar de enviar y recibir la información, el módulo transmisor capta la información de los sensores y la muestra en su

pantalla integrada para luego transmitir esa información al módulo receptor, el módulo receptor muestra la información recibida en su pantalla para luego subir dicha información a la página web.

Nivel 2 comunicación web: El módulo receptor utiliza la función WiFi para conectarse a una red de internet y con las credenciales MQTT sube y muestra la información obtenida por los sensores en tiempo real en la página web.

Nivel 3 aplicación web: Desde cualquier buscador se accede al dominio, con las credenciales de usuario accedemos al “dashboard” donde se muestran las lecturas de los sensores. Se añadieron gráficos para poder visualizar mejor los cambios que se den en las lecturas. Con instrucciones PHP vamos a permitir el acceso al usuario a al dashboard.

Nivel 4 base de datos: En la programación de la página web está establecido el mensaje que se va a publicar en la base de datos. Las lecturas obtenidas por los sensores se van a guardar en HeidiSQL, específicamente en una tabla, le damos acceso a Heidi para que el servidor pueda almacenar la información. Con instrucciones PHP vamos a permitir la comunicación entre la base de datos y lo mostrado en la página web.

Comunicación entre Sensores & ESP32

Los sensores se comunican a través de I2C, en la programación de los ESP32 se definen qué pines van a ser utilizados para dicha comunicación ya que puede haber conflictos porque la pantalla integrada en el mismo también utiliza comunicación I2C, también se debe definir la frecuencia en la que van a trabajar los ESP32, para el territorio ecuatoriano se usa la banda de 433 MHz.

La comunicación entre Los ESP32 transmisor y receptor va a ser por tecnología LoRa, es decir ellos siempre van a estar comunicados y enviando y recibiendo datos sin depender de internet.

El ESP32 transmisor crea una cadena de caracteres para poder enviar un sólo paquete al ESP32 receptor con la información requerida, es importante establecer variables diferenciadas al momento de enviar y recibir ya que pueden ser confundidas por el código.

ESP32 & Servidor MQTT

Dentro de la programación del ESP32 vamos a encontrar las credenciales para poder acceder al servidor MQTT y poder visualizar los mensajes que llegan a la web y en el panel EMQX.

Registro en Base de Datos

Los datos subidos por el ESP32 receptor están estructurados en una línea de caracteres, este mensaje está siendo registrado en la página web para luego interpretarlo con la codificación correspondiente para ser subido desde la página web a la base de datos.

El proyecto pretende dar la facilidad de visualizar los datos que se registran en cualquier lugar y a cualquier hora con el objetivo de no descuidar los cambios que se den en el campo de trabajo y si se presenta el caso de que ocurra algún inconveniente sea más fácil la toma de decisiones para poder mitigar dichos problemas de la maera más eficiente.

Funcionamiento del Proyecto

El proyecto utiliza los recursos integrados para que sea posible la obtención, la subida, visualización y registro de la información al que está designado.

Para aclarar aún más las dudas que han surgido hasta ahora se procede a explicar con más detalle qué se encarga de qué y cómo se realiza dicha tarea.

Diseño del circuito

Para poder medir la humedad la temperatura y la presión atmosférica se utilizan los siguientes circuitos integrados:

BME280

Es un sensor de temperatura, humedad y presión generalmente utilizado en aplicaciones o en dispositivos inteligentes, mide los parámetros con gran precisión y con respuesta rápidas a los cambios en el entorno.

Rango de operaciones BME280

Los rangos de operaciones en sus diferentes variables son, (Bosch, 2021)

Temperatura: -40 – 85 °C.

Humedad: Humedad relativa: ±3% con tendencia ≤2%.

Presión: 300 – 1100 hPa Interface: I2C & SPI Voltaje: 1.7 – 3.6 V.

DS18b20

Sensor digital de temperatura, utiliza una función de bus de datos para la comunicación. Existen varias presentaciones, pero en este caso se va a enfocar en la versión de sonda. Posee 3 pines, uno de datos, uno para alimentación y uno para tierra, (Naylampmechatronics, 2021).

Rango de operaciones DS18b20

Temperatura: -55 – 125 °C.

Interface: 3 pines. 1 GND, 1 VCC, 1DQ.

Voltaje: 3 - 5V.

Heltec LoRa WiFi 32 V2

Placa electrónica que tiene integrado un ESP32 con pantalla OLED y un chip nodo LoRa. Posee características similares al ESP32 clásico, pero con la posibilidad de conectarse a una red WiFi y comunicarse por tecnología LoRa con otros dispositivos.

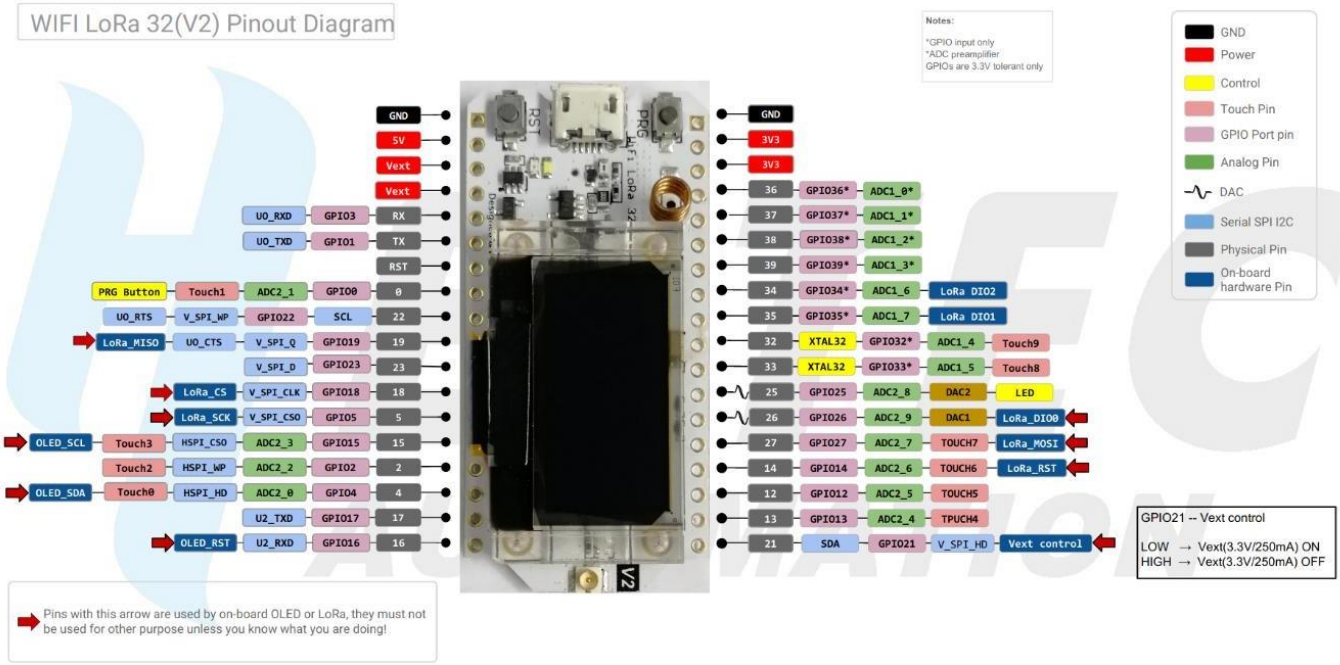

Figura 4. diagrama de pines para Heltec LoRa Wi-Fi (V2)

Fuente: http://www.weargenius.in/wp-content/uploads/2018/12/df.png

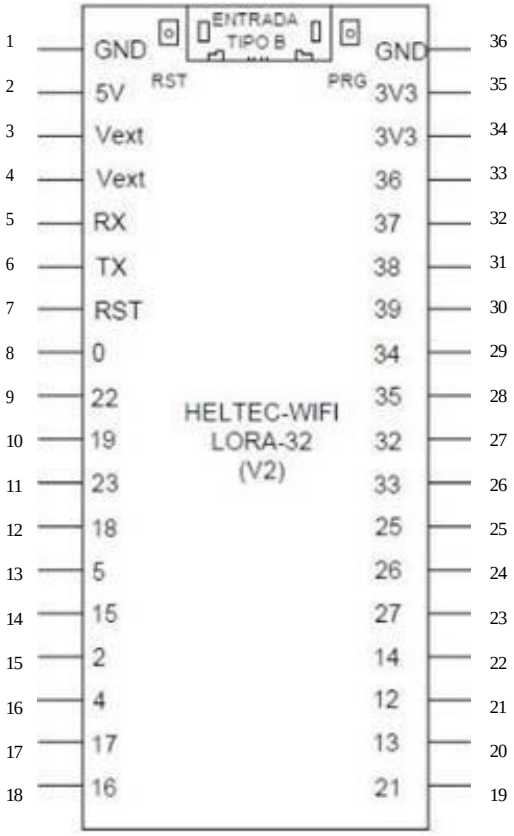

Figura 5. Diagrama de pines para Heltec LoRa Wi-Fi (V2)

¿Qué incluye?

Dependiendo del distribuidor

- Caja.

- 2 juegos de pines.

- Incluye una antena para mayor alcance.

Especificaciones

Chip maestro: ESP32 (240 MHz dual-core)

Comunicación inalámbrica:

WiFi: 802.11 b/g/n (Hasta 150Mbps).

Bluetooth: V4.2 y BLE.

LoRa: Nodo a nodo o LoRaWAN Salida de LoRa (dB): 19dB ± 1dB Memoria FLASH: 8MB.

Pantalla: 0.96 pulgadas 128*64 OLED display

Memoria RAM: 520KB.

Tamaño: 50.2 x 25.5 x 9.74 mm

Tipo de batería: 3.7V Litio (SH1.25 x 2)

Interface: Micro USB x1, Antena LoRa (IPEX) x1, 36 pines.

Características eléctricas

En la siguiente tabla se demuestra las características eléctricas:

Cualidad eléctrica	Condición	Mínimo	Regular	Máximo
Fuente de poder	USB (≥500mA)	4.7V	5	6
	Batería de lítio (≥250mA)	3.3V	3.7	4.2
	Pin 3V3 (≥150mA)	2.7V	3.3	3.5
	Pin 5V (≥500mA)	4.7V	5	6
Consumo (mA)	Escáner WiFi		115	
	WiFi AP		135	
	LoRa 10dB salida		50	
	LoRa 12dB salida		60	
	LoRa 15dB salida		110	
	LoRa 20dB salida		130	
Salida	Pin 3V3			500mA
	Pin 5V (USB)		Igual a la corriente de entrada	
	Vext 3V3			500mA

Tabla 2. Características eléctricas del controlador, (Heltec, 2020).

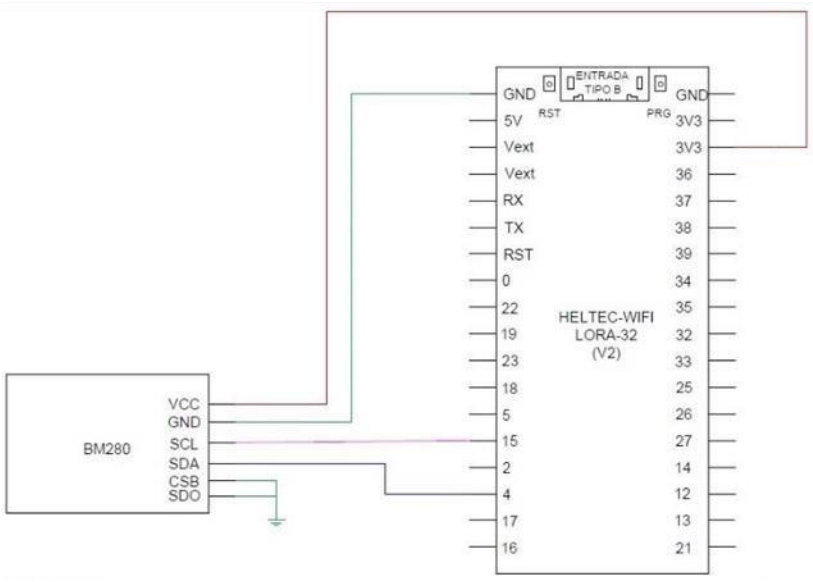

Figura 6. Diseño del circuito para el sensor BME 280.

El sensor tiene 6 pines para su conexión, 4 pines de datos y 2 para voltaje y tierra. SDA y SCL van a ser los pines de datos que se van a comunicar con el ESP32 para comunicación I2C, respectivamente se van a conectar en los pines GPIO4 y GPIO15. Este sensor se va a encargar de medir la humedad y la presión.

VCC	3V3
GND	GND
SDA	GPIO4
SCL	GPIO15

IMPORTANTE

Revisar la imagen referencial 1 y 2 para identificar pines y nomenclaturas

Tabla 3. Especificaciones de conexión adjunta a la figura 4.

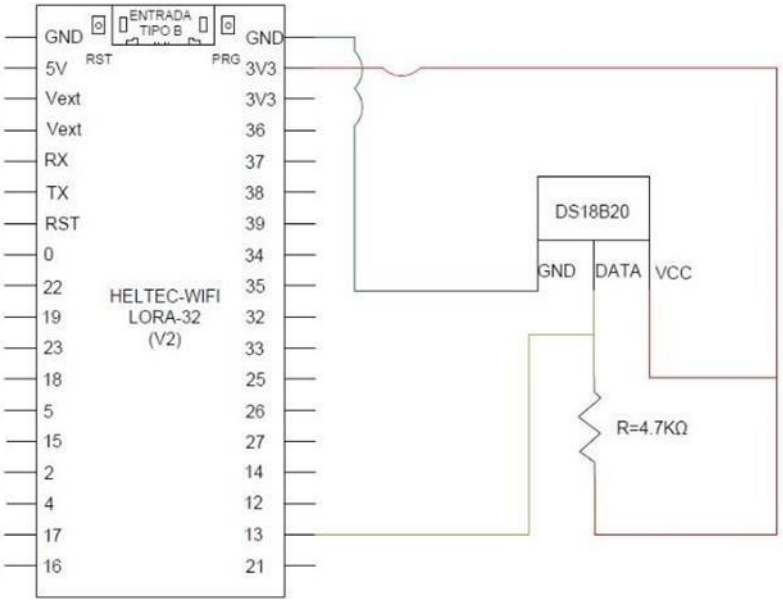

Figura 7. Diseño del circuito para el sensor DS18B20

El sensor tiene 3 Canales para su conexión, 1 VCC, 1 GND, DQ/DATA, con el ESP32 para comunicación I2C, respectivamente se van a conectar en los pines GPIO4 y GPIO15. Este sensor se va a encargar de medir la temperatura.

VCC	3V3
GND	GND
DQ/DATA	GPIO13

IMPORTANTE

Revisar la imagen referencial 1 y 2 para identificar pines y nomenclaturas

Tabla 4. Especificaciones de conexión adjunta a la figura 5.

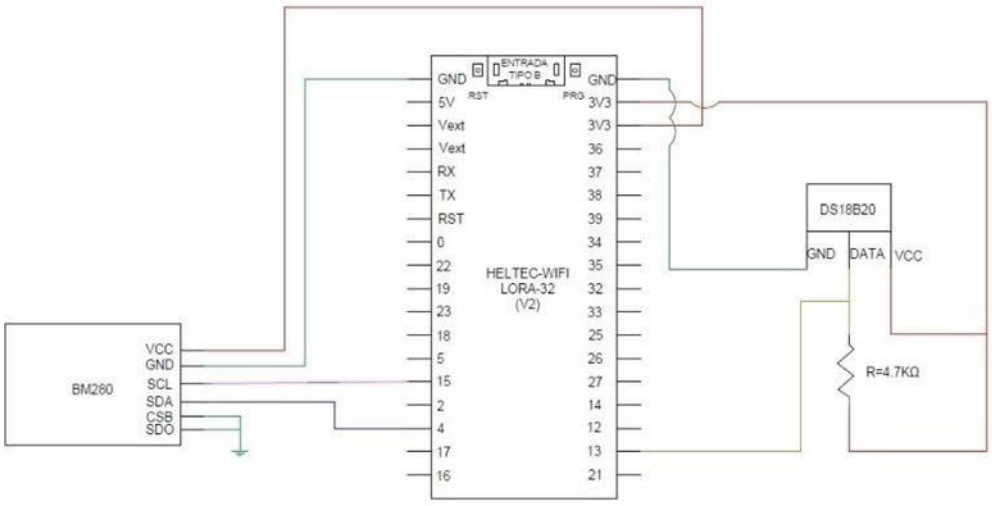

Figura 8. Diseño del circuito completo.

Finalmente quedan los 2 sensores conectados y listos para captar los parámetros, el transmisor se encargará de las lecturas y las enviará al receptor a través de tecnología LoRa. Se designaron los pines para evitar conflictos de comunicación.

BME280	Pin
VCC	3V3
GND	GND
SDA	GPIO4
SCL	GPIO15
DS18b20	Pin
VCC	3V3
GND	GND

IMPORTANTE

Revisar la imagen referencial 1 y 2 para identificar pines y nomenclaturas

Tabla 5. Especificaciones de conexión adjunta a la figura 6.

Codificación

Este proyecto va a ser codificado en lenguaje C++ utilizando el IDE de Atom, los comandos utilizados están en un lenguaje que son de tipo seudocódigo.

Atom

Es un editor de texto que permite programar en una gran cantidad de lenguajes para desarrollo de programas y aplicaciones gráficas.

Atom fue diseñado para ser personalizable, por eso permite el desarrollo de aplicaciones en diferentes lenguajes como C, C++, Node.js, HTML, JavaScript, PHP, entre otros, (Atom, 2021).

Platformio

Es la herramienta de desarrollo específicamente utilizada para programar los ESP32, este software contiene herramientas similares a las de Arduino tales como compilar programas, subir programas a la placa, monitor serial, entre otras. Una gran ventaja de este entorno es que permite conectarse a un servidor y gestionarlo, (Platformio, 2021).

Codificación Transmisor (Tx)

Librerias

El ESP32 Tx va a utilizar las siguientes librerías:

Tx	Función
Arduino	Lenguaje C++
Stdlib	Lenguaje C
Dallas Temperature	BME280
AsyncDelay	BME280
SoftWire	BME280
Adafruit Unified Sensor	BME280
Adafruit BME280 Library	BME280
Heltec ESP32 Dev-Boards	Programación para ESP32 marca HELTEC

Tabla 6. Librerías para la codificación del transmisor (Tx)

```
1   //LIBRERIAS
2   #include <Arduino.h>
3   #include <WiFi.h>
4   #include <stdlib.h>
5   #include <OneWire.h>
6   #include <DallasTemperature.h>
7   #include <AsyncDelay.h>
8   #include <SoftWire.h>
9   #include <Adafruit_Sensor.h>
10  #include <Adafruit_BME280.h>
11  #include "heltec.h"
12  #include "images.h"
```

Figura 9. Librerías incluidas en la codificación del transmisor (Tx)

```
68  #define BAND    433E6
69  #define SDA 4
70  #define SCL 15
```

Figura 10. Codificación para el ESP32 Transmisor (Tx)

ESP32 va a trabaja con frecuencias 300MHz – 3000MHz ya que tenemos un enlace de corto alcance, pero el ESP32 es capaz de transmitir a mayores distancias hasta 2,8km en áreas abiertas, Podemos “setear” las bandas a 433, 868 y 915 MHz, a mayor frecuencia, menor distancia.

```
81  SoftWire sw(SDA, SCL);
82  OneWire oneWireObjeto(pinDato);
83  DallasTemperature sensorDS18B20(&oneWireObjeto);
84  Adafruit_BME280 bme;
85  TwoWire I2Cone = TwoWire(1);//0
```

Figura 11. Codificación del transmisor (Tx), llamado de librerías

La variable pinDato se refiere al Pin designado para datos del sensor DS18b20, fue establecido para el pin GPIO13.

```
void startBME(){
  bool bme_status;
  bme_status = bme.begin(0x76);
  if (!bme_status) {
        Heltec.display->clear();
        Heltec.display->drawString( 0, 0, "No valid BME280 found");
        Heltec.display->drawString( 0, 12, "Please check wiring!");
        Heltec.display->display();
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
  }
}
```

Figura 12. Codificación del transmisor (Tx), Función para identificar BME280

Para iniciar el BME280 necesitamos que el ESP32 lo identifique, por eso creamos una función en donde establecemos por defecto que lo detecta y que comience a medir los parámetros.

```
void getReadings(){
  bme.setSampling(Adafruit_BME280::MODE_FORCED,
                  Adafruit_BME280::SAMPLING_X16,
                  Adafruit_BME280::SAMPLING_X1,
                  Adafruit_BME280::SAMPLING_X1,
                  Adafruit_BME280::FILTER_X16,
                  Adafruit_BME280::STANDBY_MS_0_5 );
  bme.takeForcedMeasurement();
    tempBME = bme.readTemperature();
    hudBME = bme.readHumidity();
    pressBME = bme.readPressure() / 100.0F;

  }
```

Figura 13. Codificación del transmisor (Tx), Función para obtener lecturas BME280

Para que los sensores midan los parámetros se debe crear una función que obtenga los datos, una vez que se active esta función se podrán ver las medidas en la pantalla del ESP32.

Dentro del “Setup” se procede a llamar la función “StartBme”, se la coloca en esta sección para que el programa identifique que existe un BME280 antes de proceder con el resto del programa, es decir, si no identifica al BME280 no se muestran las lecturas. También se procede a llamar al sensor DS18b20 para que también empiece a medir, no se requiere crear una función ya que no necesita de una función para identificarlo.

Figura 14. Codificación del transmisor (Tx), Llamado de función de identificación BME280

Figura 15. Codificación del transmisor (Tx), Llamado de función obtener lecturas DS18b20

En la función “Loop” se va a llamar a las funciones que obtienen los valores de los sensores ya que van a estar obteniendo valores de forma contínua y en tiempo real, para el caso del BME280 se llama a la función “getReadings” y para el DS18b20 se procede a solicitar un “requestTemperatures”.

Figura 16. Codificación del transmisor (Tx), Llamado de función “getReadings” BME280

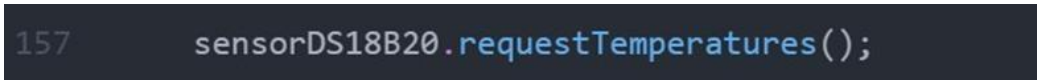

Figura 17. Codificación del transmisor (Tx), Solicitar lecturas del sensor “getReadings” DS18b20

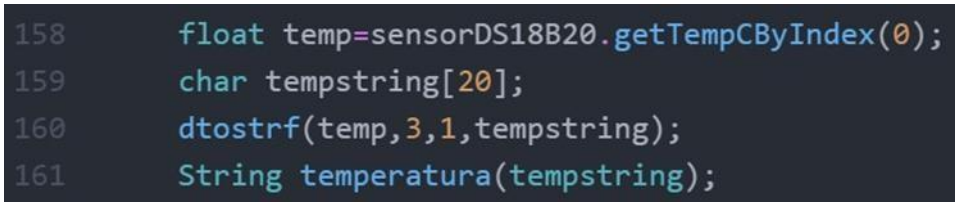

Figura 18. Codificación del transmisor (Tx), Conversión de datos a cadena de caracteres DS18b20

Para el sensor DS18b20 creamos una variable con el comando “sensorDS18B20.getTempCByIndex(0);” y también se crea un arreglo donde se va a guardar ese dato, y para finalizar se transforma en cadena de caracteres ya que vamos a enviar un mensaje con las medidas convertidas a cadena de caracteres.

```
192     LoRa.beginPacket();
193      LoRa.setTxPower(14,RF_PACONFIG_PASELECT_PABOOST);
194      LoRa.print("TempBME:! ");
195      LoRa.print(temperatura);
196      LoRa.print(" HudBME:& ");
197      LoRa.print(hudBME);
198      LoRa.print("PressBME:% ");
199      LoRa.print(pressBME);
200      LoRa.println(MensajeBME);
201     LoRa.endPacket();
```

Figura 19. Codificación del transmisor (Tx), Creación del paquete para comunicación LoRa

Para crear el paquete es necesario empezar con “LoRa.beginPacket();”, luego establecemos la ganancia con la que se va a transmitir, para este caso es 14dB, pero puede ajustarse hasta u máximo de 20dB.

Codificación Recibidor (Rx)

Librerías

El ESP32 Rx va a utilizar las siguientes librerías:

Rx	Función
Arduino	Lenguaje C++
WiFi	Conectarse a una red WiFi
PubSubClient	Conectarse al servidor MQTT
Heltec ESP32 Dev-Boards	Programación para ESP32 marca HELTEC

Tabla 7. Librerias para la codificación del Recibidor (Rx)

```
1  //LIBRERIAS
2  #include <Arduino.h>
3  #include <WiFi.h>
4  #include <PubSubClient.h>
5  #include <stdlib.h>
6  #include <OneWire.h>
7  #include <DallasTemperature.h>
8  #include "heltec.h"
9  #include "images.h"
```

Figura 20. Librerias incluidas en la codificación del Recibidor (Rx)

```
const char* ssid     = "xxxxxxxxxx";
const char* password = "xxxxxxxxxx";

const char *mqtt_server = "prototypealpha.tk";
const int mqtt_port = 1883;
const char *mqtt_user = "xxxxxxxxxx";
const char *mqtt_pass = "xxxxxxxxxx";
```

Figura 21. Credenciales para dar acceso al Recibidor (Rx) a una red WiFi y al servidor MQTT

Se debe otorgar las credenciales necesarias para que el ESP32 se pueda comunicar a través de Internet con la página web y con el servidor MQTT, para conectarse al servidor MQTT se utiliza el puerto 1883.

```
WiFiClient espClient;
PubSubClient client(espClient);

long lastMsg = 0;
char msg[25];
```

Figura 22. Identificación espClient para el acceso del recibidor (Rx)

Llamamos a las funciones “WiFi” y “PubSubClient” y se va a identificar al Recibidor como espCliente, además creamos un arreglo en el cual vamos a guardar 25 caracteres el cual va a ser el mensaje que vamos a ubicar en la página web y n la base de datos.

```
void LoRaData(){
  Heltec.display->clear();
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);
  Heltec.display->setFont(ArialMT_Plain_10);
  Heltec.display->setFont(ArialMT_Plain_16);
  Heltec.display->drawString(5, 5,"T = " + Tempw + " ºC");
  Heltec.display->drawString(5, 20,"H = " + hudBME + "%");
  Heltec.display->drawString(5, 35,"P = " + pressBME + " hPa");
  Heltec.display->setFont(ArialMT_Plain_10);
  Heltec.display->drawString(0, 50,"Receiving Packets From Tx");
  Heltec.display->drawString(110, 0,"(Rx)");
  Heltec.display->display();
}
```

Figura 23. Función" LoRaData" para el recibidor (Rx)

En el recibidor crea una función similar a la del transmisor para que se muestren igual y poder notar las diferencias o que en verdad se están comunicando, si se actualizan las lecturas de Tx también deben actualizarse las Rx.

```
void cbk(int packetSize) {
  packet ="";
  packSize = String(packetSize,DEC);
  for (int i = 0; i < packetSize; i++) { packet += (char) LoRa.read(); }
  rssi = "RSSI " + String(LoRa.packetRssi(), DEC) ;
     MensajeRecibido=packet;
     if (MensajeRecibido!=""){
     int data1=packet.indexOf('!');
     int data2=packet.indexOf('&');
     int data3=packet.indexOf('%');
     int data4=packet.indexOf('#');
     Tempw=packet.substring(data1+1,data1+7);
     hudBME=packet.substring(data2+1,data2+7);
     pressBME=packet.substring(data3+1,data3+8);
     tempBME=packet.substring(data4+1,data4+8);
     LoRaData();
     }
     else{
       Tempw="";
       hudBME="";
       pressBME="";
       tempBME="";
       LoRaData();

     }
     Serial.println(MensajeRecibido);
  }
```

Figura 24. Función "cbk" para el recibidor (Rx)

La función "cbk" se encarga de identificar el mensaje, también se encarga de ordenar el mensaje. El mensaje está separado por símbolos, "!", "&", "%", "#", es decir, el mensaje va a

ubicar las lecturas sobre esos símbolos, luego se construyen nuevas variables en la que se identificará cuántos espacios o cuántos caracteres se van a escribir antes de pasar al siguiente dato para poder visualizar el mensaje “completo”.

```
148  void loop(){
149    if (!client.connected()) {
150      reconnect();
151    }
152    client.loop();
153    long now = millis();
154    if (now - lastMsg > 500){
155      lastMsg = now;
156
157      Serial.println("RSSI ");
158      Serial.println(rssi);
159      String to_send = String(Tempw) + "," + String(hudBME) + "," + String(pressBME);
160      to_send.toCharArray(msg, 25);
161      Serial.print("Publicamos mensaje -> ");
162      Serial.println(msg);
163      client.publish("values", msg);
164      Serial.println(String(packet));
165      Serial.println(Tempw);
166      Serial.println(hudBME);
167      Serial.println(pressBME);
168    }
169    int packetSize = LoRa.parsePacket();
170    if (packetSize) { cbk(packetSize);  }
171    delay(10);
172  }
```

Figura 25. Función “cbk” para el recibidor (Rx)

En la función “Loop” establecemos que el cliente se reconecte en caso de fallas en, luego procederemos a subir las cadenas de caracteres en el lugar designado para mostrar ese mensaje además de visualizar la llegada del mensaje en el servidor MQTT.

```
186   void setup_wifi(){
187     delay(10);
188     // Nos conectamos a nuestra red Wifi
189     Serial.println();
190     Serial.print("Conectando a ");
191     Serial.println(ssid);
192
193     WiFi.begin(ssid, password);
194
195     while (WiFi.status() != WL_CONNECTED) {
196       delay(500);
197       Serial.print(".");
198     }
199
200     Serial.println("");
201     Serial.println("Conectado a red WiFi!");
202     Serial.println("Dirección IP: ");
203     Serial.println(WiFi.localIP());
204   }
```

Figura 26. Función “setup_wifi” para el recibidor (Rx)

La función “setup_wifi” se encarga de conectarse a una red wifi para tener conexión a internet y tener la posibilidad de comunicarse con la página web, el servidor MQTT y con la base de datos.

Página Web y Comunicación Online

Para la creación de la página web se utilizan varios servicios para poder habilitar las características que posee.

Amazon Web Service (AWS)

Es una plataforma de servicios en la nube los cuales pueden ser utilizados para desarrollo de páginas web, aplicaciones, etc. La ventaja de utilizar este servicio es que se no se debe comprar ni poseer centros de datos ni tampoco de servidores físicos y se puede seleccionar el servicio que mejor se adapte al proyecto que se desea realizar, entre las opciones tenemos

servicios de base de datos, almacenamiento informático, inteligencia artificial, análisis de datos, IOT, etc, (Amazon, 2021).

El VPS (Servidor Virtual Privado) del proyecto está alojado en AWS, gracias a esto es posible establecer reglas de entrada y reglas de salida lo cual permite otorgar permisos especiales a usuarios y permitir que se conecten desde cualquier parte del mundo.

Vesta Control Panel

Vesta ofrece servicios de seguridad, servicios de email, servicios de base de datos e incluso servicios de FTP (File Transfer Protocol) permitiendo subir o descargar archivos del servidor, (Vestacp, 2021).

Vesta es el encargado de que el servidor tenga servicios de transferencia de archivos, firewall, certificados SSL o Secure Sockets Layer (capa de sockets seguros) el cual permite la autenticación, encriptación y desencriptación de datos enviados a través de Internet. Incluso se encarga de instalar el motor para que el servidor pueda utilizar una base de datos.

Putty

Permite el acceso a servidores a través de un cliente SSH en modo remoto o serial, se puede aprovechar esta característica para gestionar o administrar un servidor o una máquina remota, (Putty, 2021).

Para acceder al servidor del proyecto se utiliza Putty y con ello se puede administrar el servidor de manera remota o incluso añadir al servidor más herramientas para expandir las utilidades del proyecto.

Plantilla para Desarrollo Web

Para entender cómo funciona la página web se utilizaron plantillas y opciones de Flatkit, Flatkit ofrece una gran variedad de diseños dinámicos y servicios preestablecidos para configurar y personalizar una página web.

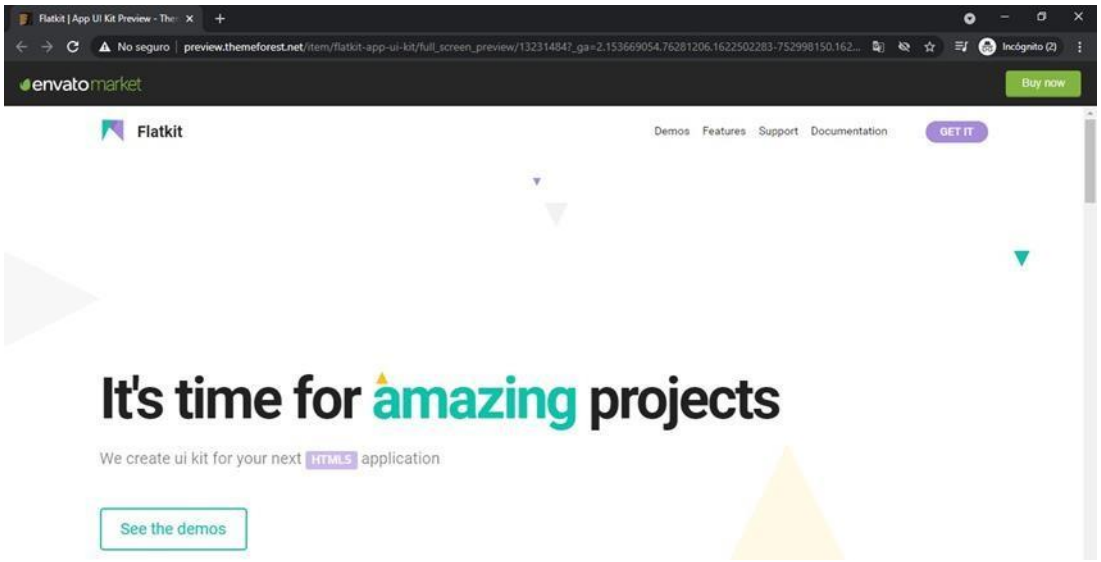

Figura 27. Temas de Flatkit, (Flatkit, 2020).

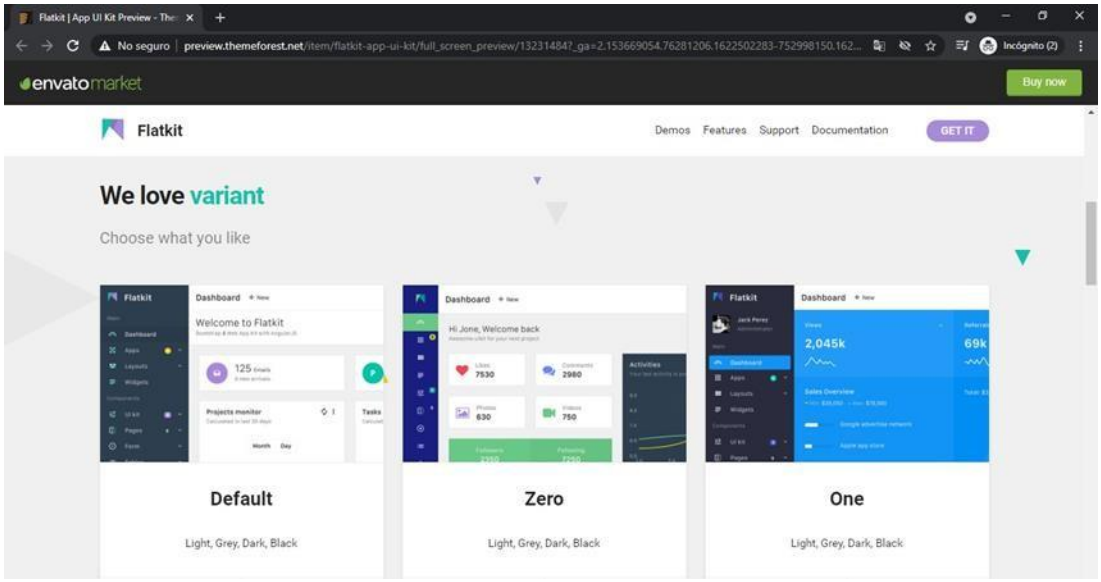

Figura 28. Demos de Flatkit, (Flatkit, 2020).

Se adquirió un paquete de flatkit para construir la página web porque los recursos que flatkit ofrece son muy didácticos y agradables a la vista, también es un entorno bastante amigable y relativamente sencillo de reprogramar.

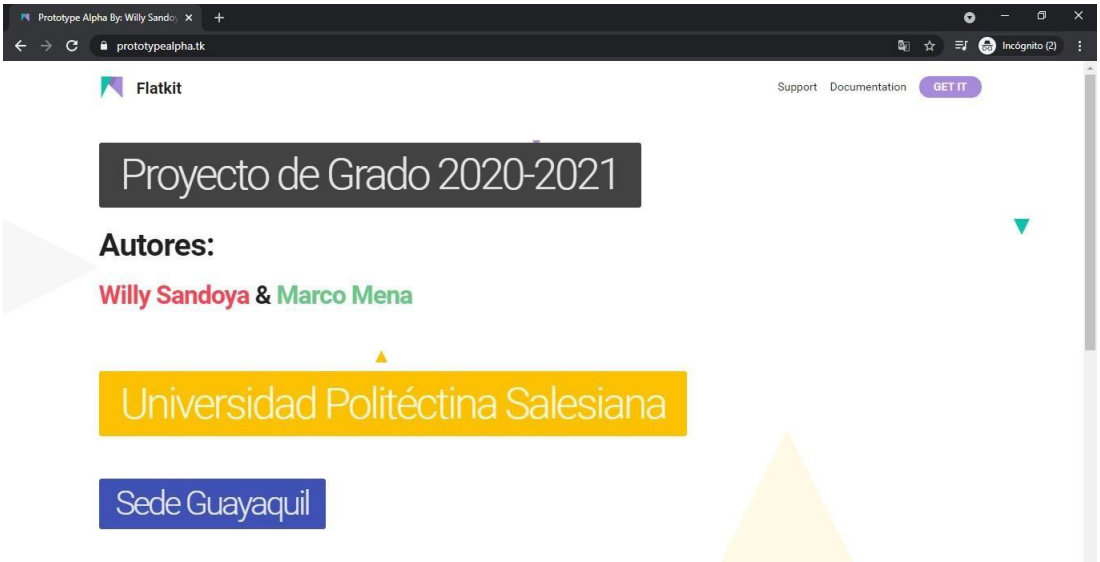

Figura 29. Página principal en la web del Proyecto, (Flatkit, 2020).

En la página principal se colocaron los botones para iniciar sesión y registrarse ya que el proyecto es un prototipo de servicio podemos acceder a nuestro dashboard o panel donde vamos a encontrar la información que se solicita al prototipo.

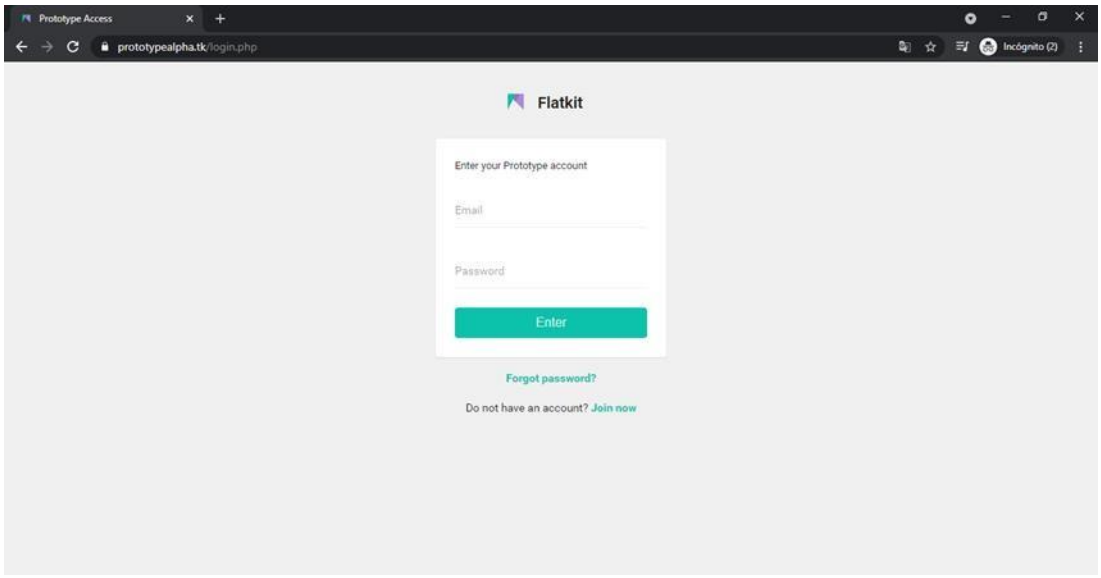

Figura 30. Página Login del proyecto, (Flatkit, 2020).

En la sección “Login” encontramos un formulario donde se va a introducir las credenciales del usuario para que tenga acceso al dashboard.

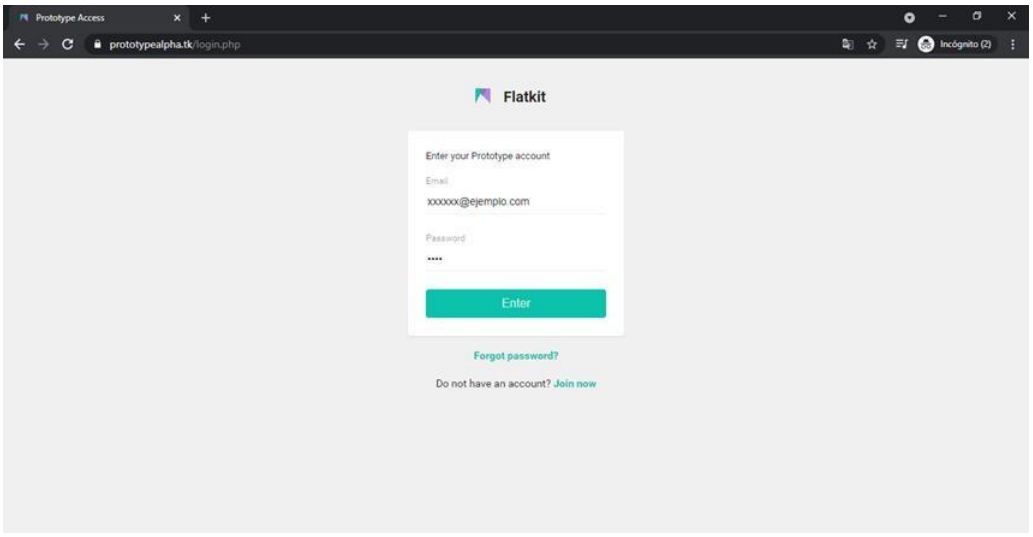

Figura 31. Página Login Introducción de credenciales, (Flatkit, 2020).

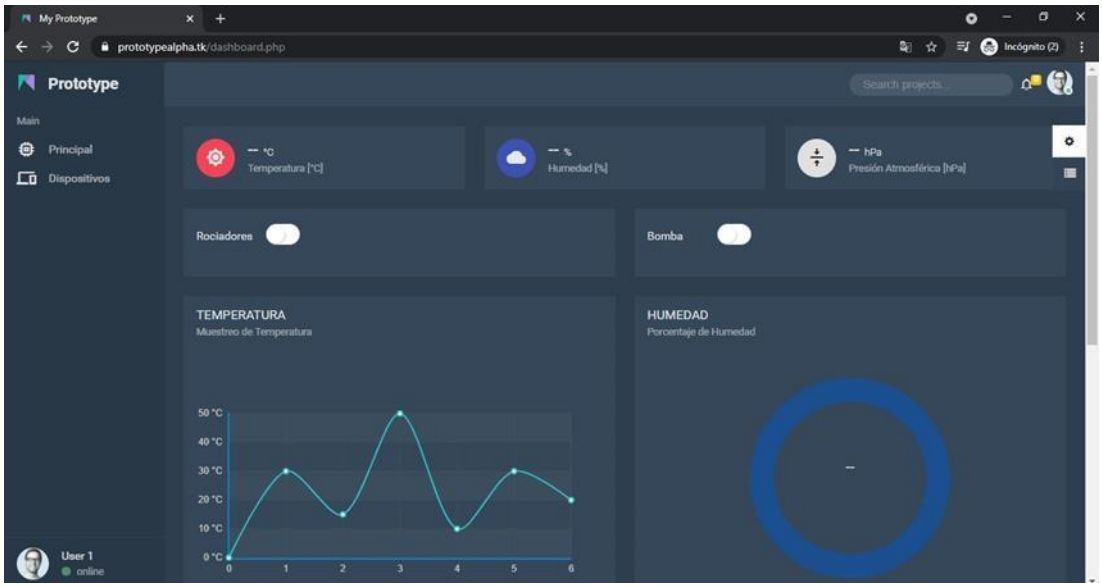

Figura 32. Página Dashboar, (Flatkit, 2020).

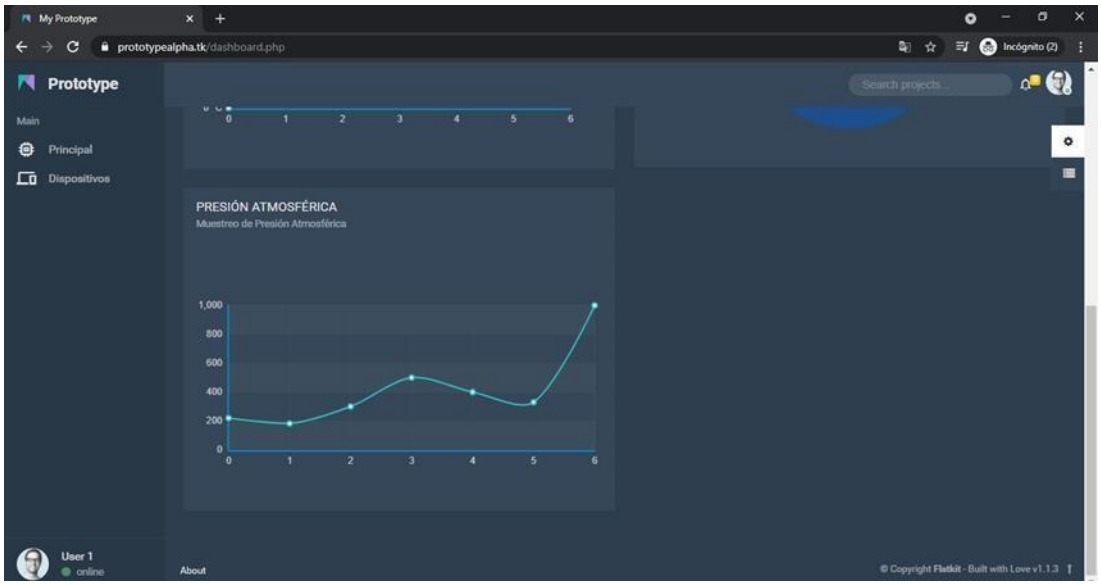

Figura 33. Página Dashboard, (Flatkit, 2020).

En el dashboard se optó por dejar una apariencia predeterminada para poder visualizar cómo se ven los diferentes objetos.

Cuando los ESP32 empiecen a funcionar los datos que se visualizan cambiarán por los datos obtenidos en tiempo real.

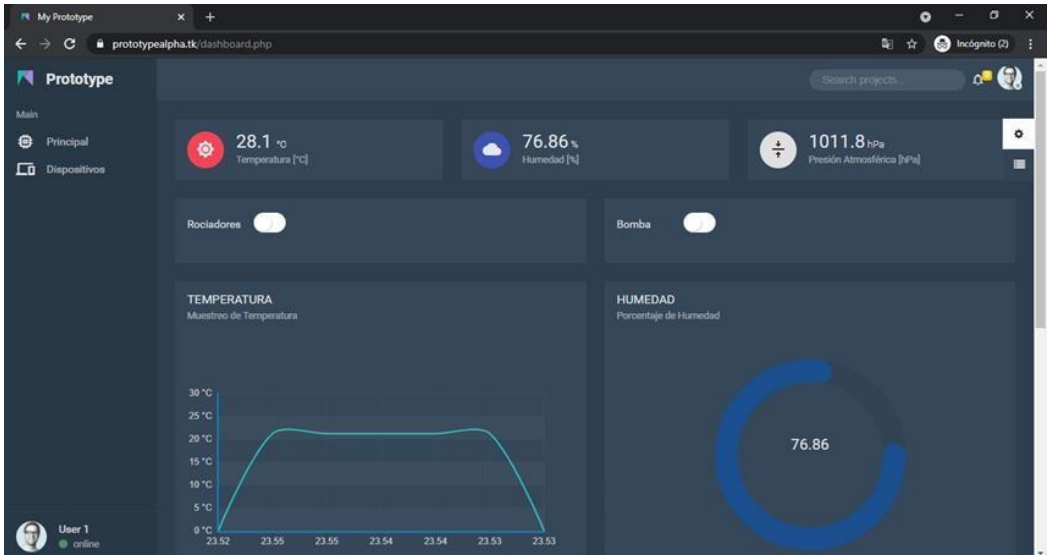

Figura 34. Página Dashboard, actualización de datos, (Flatkit, 2020).

Cuando el recibidor empieza a tomar lecturas del transmisor el gráfico predeterminado desaparece y se actualiza según los parámetros que actualmente se están tomando.

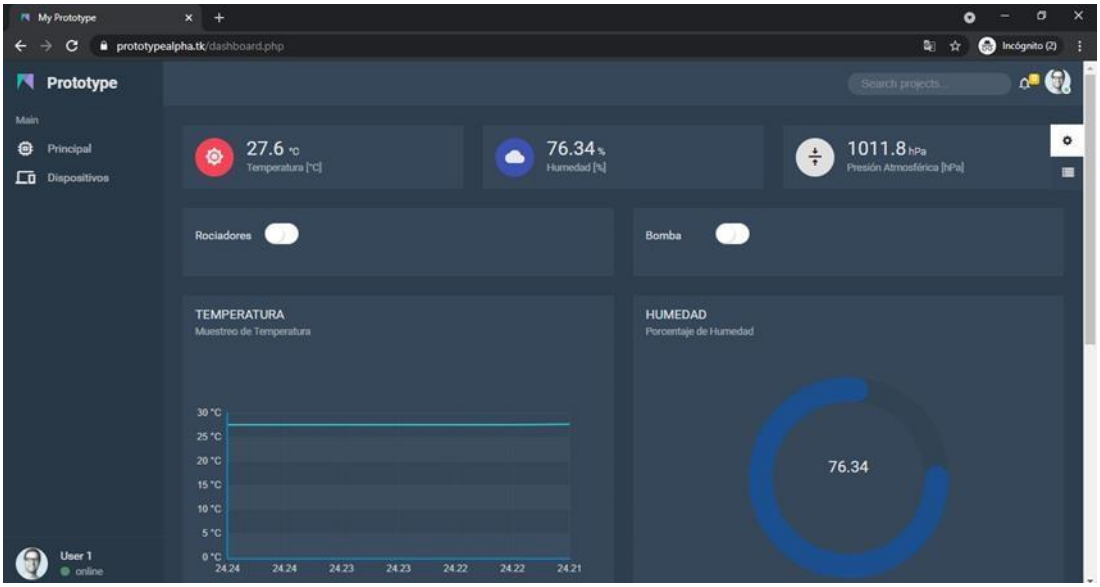

Figura 35. Página Dashboard, estabilidad de datos, (Flatkit, 2020).

Ya que los cambios son muy leves ya que se está hablando de décimas, el gráfico se queda como una función constante.

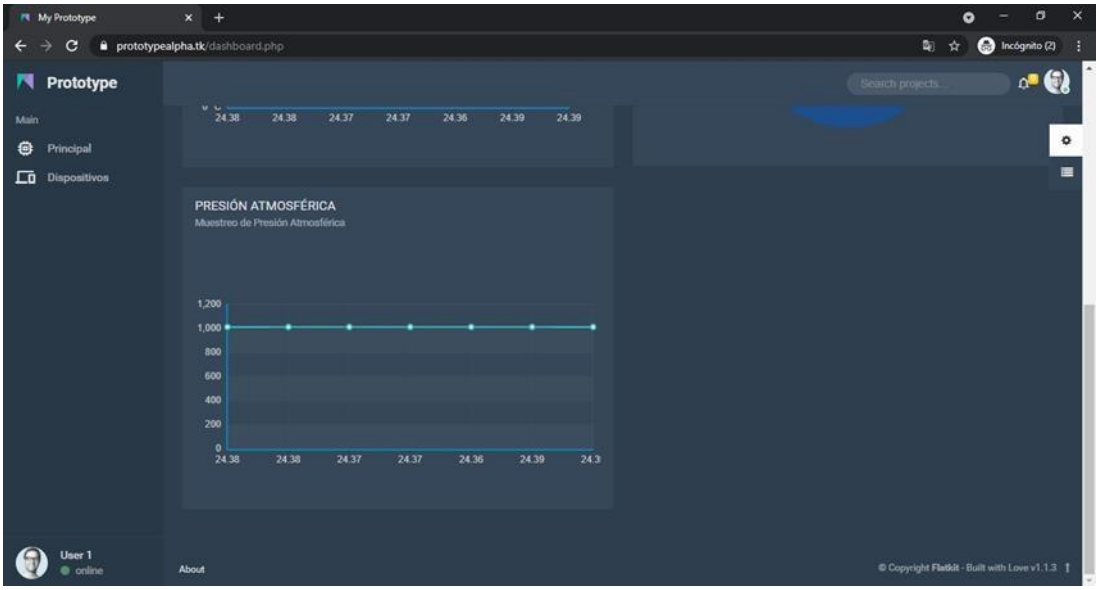

Figura 36. Página Dashboard, estabilidad de datos, (Flatkit, 2020).

Funcionamiento de la Base de Datos

Para tener acceso a la base de datos se llena el formulario con los datos respectivos del servidor.

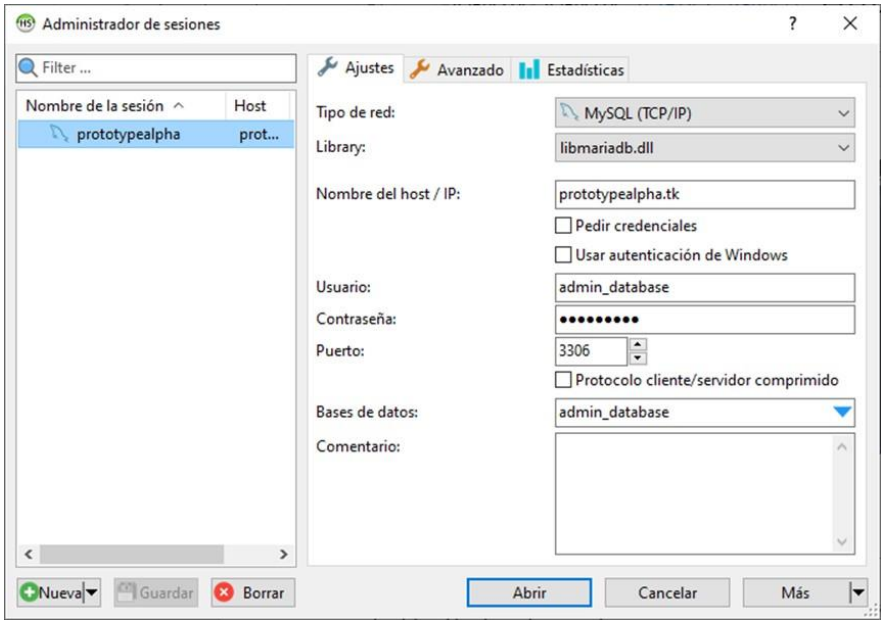

Figura 37. Ventana HeidiSQL, acceso al servidor

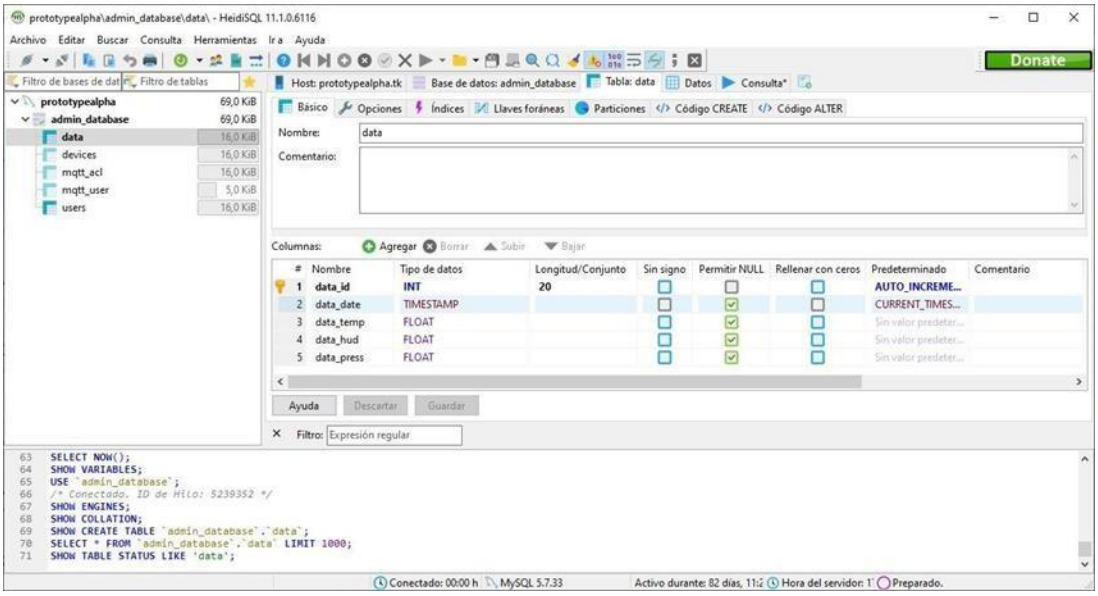

Figura 38. Ventana HeidiSQL, ventana principal del servidor

Después de acceder, aparece nuestro servidor con todas las tablas que se crearon para gestionar el proyecto, tablas para usuarios, administrador MQTT, usuario MQTT, entre otros, cabe mencionar que también está la tabla en la que se va a llenar con los datos que se obtienen de los sensores.

Figura 39, Ventana HeidiSQL, tabla de datos

Para crear la tabla el programa da la facilidad de agregar las variables que se desea introducir, y se pueden crear con herramientas dentro del mismo sin codificar.

data_id: Variable encargada de relacionar al usuario con el dashboard respectivo. data_date: Variable encargada de registrar la fecha y hora de cada registro. data_temp: Variable encargada de registrar la temperatura.

data_hud: Variable encargada de registrar la humedad. data_press: Variable encargada de registrar la presión.

Para permitir el registro de los datos se conecta con el servidor a través de PuTTY, cada vez que se PuTTY adquiera los permisos necesarios y se reciba la información se mostrará el mensaje que se ha agregado una nueva fila en la base de datos.

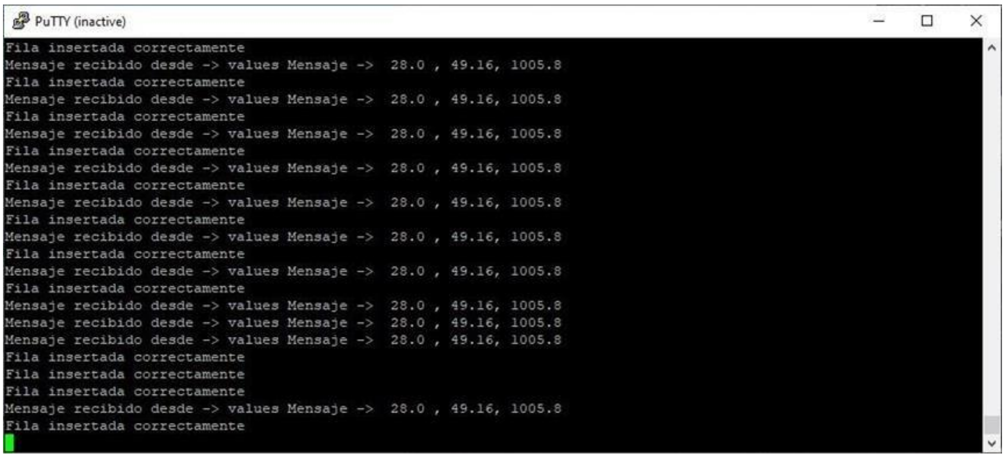

Figura 40. Ventana PuTTY, registro de datos

Cuando se activa el enlace entre el servidor y la base de datos se empiezan a registrar los

datos en la tabla designada para la temperatura, humedad, presión.

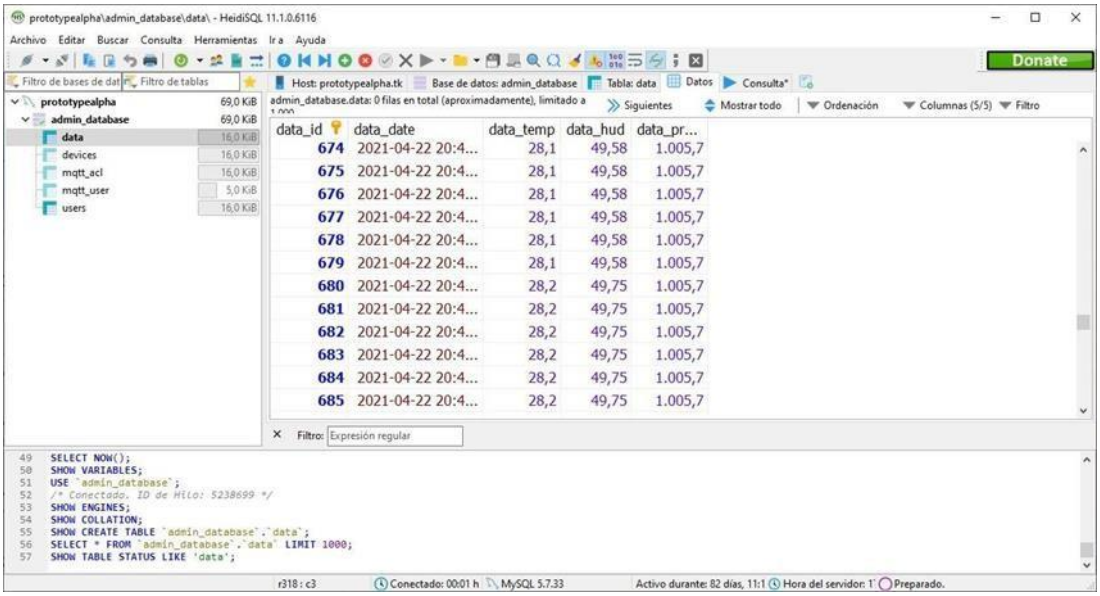

Figura 41. Ventana HeidiSQL, registro en las tablas

Se rellena la tabla, con esto se puede determinar cuándo se realizaron los cambios y determinar las siguientes acciones para mejorar la productividad.