UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA:

INGENIERÍA ELECTRÓNICA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO ELECTRÓNICO

PROYECTO TÉCNICO

REPOTENCIACIÓN DE UN MÓDULO INTERACTIVO A TRAVÉS DE ARDUINO CON COMUNICACIÓN DE RADIOENLACE A IOT Y NODE RED PARA INTEGRACIÓN A DJANGO EN LA VISUALIZACIÓN Y ALMACENAMIENTO DE DATOS (SQL SERVER).

AUTOR:

JOEL ENRIQUE TOVAR TAPIA

TUTOR:

ING. GENARO DIAZ SOLIS.MSIG

GUAYAQUIL- ECUADOR

2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Joel Enrique Tovar Tapia con documento de identificación N° 0941649592 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo 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.

Guayaquil, de marzo de 2024

Atentamente,

.....

Nombre: Joel Enrique Tovar Tapia

CI: 0941649592

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

Yo, Joel Enrique Tovar Tapia con documento de identificación N° 0941649592, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del: Proyecto Técnico: "REPOTENCIACIÓN DE UN MÓDULO INTERACTIVO A TRAVÉS DE ARDUINO CON COMUNICACIÓN DE RADIOENLACE A IOT Y NODE RED PARA INTEGRACIÓN A DJANGO EN LA VISUALIZACIÓN Y ALMACENAMIENTO DE DATOS (SQL SERVER)", el cual ha sido desarrollado para optar por el título de: Ingeniero electrónico mención en telecomunicaciones, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

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

Guayaquil, marzo de 2024

Atentamente,

.....

Nombre: Joel Enrique Tovar Tapia

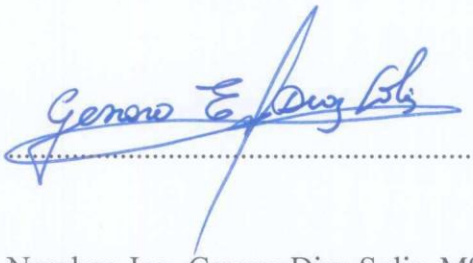
CI: 0941649592

CERTIFICADO DE DIRECCIÓN DE TRABAJO DE TITULACIÓN

Yo, Genaro Diaz Solís, con documento de identificación N° 0912186467, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: realizado por Joel Enrique Tovar Tapia con documento de identificación N° 0941649592, 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.

Guayaquil, marzo de 2024

Atentamente,



Nombre: Ing. Genaro Diaz Solis. MSIG

CI: 0912186467

DEDICATORIA

Este proyecto de trabajo se la dedico en primer lugar a Dios porque siempre me guio por el buen camino durante todo este proceso y darme la fuerza necesaria para seguir adelante a pesar que se presentaban dificultades que sucedían en el largo camino de la vida.

A mis queridos padres por todo su amor, apoyo, comprensión, que me brindaron en mi inicio de la carrera y haber confiado en mí durante todo este tiempo apoyándome en los momentos altos y bajos. Mis padres son esa luz de un final del túnel y son mi inspiración por resto de mi vida.

A mi esposa e hija por convertirse en una parte importante de mi vida, por siempre estar en muchos momentos de mis días, confiar en mí y darme ánimo para continuar estudiando lo que me apasiona porque son mi alegrías e tristeza de mi vida. Les dedico mi amor y respeto para estas dos mujeres que son importantes en mi vida y le entrego todo mi corazón.

AGRADECIMIENTOS

El agradezco el presente trabajo a Dios, por otorgarme la inteligencia, sabiduría y entendimiento en cada etapa de mi vida estudiantil. Dios me bendijo con la perseverancia para poder superar cada obstáculo y desafío que encontré durante este largo recorrido que se veía muy lejano pero que está próximo a concluir.

Mi corazón se llena de profundo agradecimiento hacia mis padres quienes sin duda han sido mis pilares fundamentales desde mis inicios y que continúan aquí en la culminación de esta etapa universitaria, decirles también que les debo una gratitud eterna por su sacrificio y aliento incondicional, gracias por haber sido el motor que me impulsó a seguir adelante en este viaje y cada logro que he alcanzado también les pertenece.

Quiero extender mi más sincero agradecimiento para la universidad, porque todo el tiempo estuvieron de alguna manera para que sus estudiantes a la meta de la mejor manera, le agradezco sinceramente por compartir sus conocimientos y consejos de aparte de los docentes, por darme críticas constructivas y sobre todo por la paciencia infinita que fue esencial para que yo pueda darle forma a esta tesis.

A mi esposa que estuvo todo el tiempo brindándome apoyo moral para no rendirme en ningún momento, aquella que compartió conmigo la montaña rusa de emociones que atravesé y que sin su ayuda no hubiese logrado llegar al final. Agradecimiento especial a mi hija por ese abrazo pequeño y diciéndome papá y con una siempre palabra y gesto me alegra mis días.

Ellas son mi inspiración total para continuar con mis estudios y a su vez mi proyecto, me apoyaron incondicionalmente con su ánimo para no rendirme y que continúe a pesar de las diversas adversidades que tuve, gracias por ser mi paño de lágrimas y alegrías.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

RESUMEN

El proyecto técnico de titulación se enfoca en la modernización de un módulo interactivo mediante la implementación de tecnologías actualizadas. Se utilizará ESP32 para la mejora para obtener los datos, Blynk IoT para la interfaz de usuario y control remoto, y antenas de radioenlace Ubiquiti para establecer comunicación inalámbrica eficiente. Los sensores proporcionados por la universidad se integrarán para la recolección de datos y guardarlas en una base de datos SQL.

Django será la base del sistema, desarrollando una aplicación web para la visualización en tiempo real y visualización de datos de una base de datos SQL. Node-RED, alojado en una Raspberry Pi, gestionará el almacenamiento de datos. Finalmente, la página Django se desplegará en un servidor para su acceso a través de un dominio web. El proyecto busca así mejorar la funcionalidad, control y visualización del módulo interactivo, aprovechando una variedad de tecnologías para lograr una solución integral.

Palabras claves: IoT, ESP32, NODE-RED, DJANGO, PYTHON.

ABSTRACT

The titulation project focuses on the modernization of an interactive module through the implementation of updated technologies. ESP32 will be used to improve data acquisition, Blynk IoT for the user interface and remote control, and Ubiquiti radio link antennas for efficient wireless communication. University-provided sensors will be integrated for data collection, and the data will be stored in an SQL database.

Django will serve as the foundation of the system, developing a web application for real-time visualization and data display from an SQL database. Node-RED, hosted on a Raspberry Pi, will manage data storage. Finally, the Django page will be deployed on a server for access through a web domain. The project aims to enhance the functionality, control, and visualization of the interactive module, leveraging a variety of technologies to achieve a comprehensive solution.

Keywords IoT, ESP32, NODE-RED, DJANGO, PYTHON.

INTRODUCCIÓN

El presente proyecto técnico se enfoca como objetivo en la modernización de un módulo interactivo mediante la implementación estratégica de tecnologías de vanguardia. El enfoque principal implica aprovechar ESP32 para mejorar la adquisición de datos, utilizando Blynk IoT para establecer una interfaz amigable con capacidades de control remoto, y empleando antenas de radioenlace Ubiquiti para lograr una comunicación inalámbrica eficiente. Como parte integral del proyecto, se llevará a cabo la integración de sensores proporcionados por la universidad para facilitar la recolección completa de datos, que posteriormente se almacena en una base de datos SQL.

Se implementará en los laboratorios de la Universidad Politécnica Salesiana el tablero de sensores con las mejoras para una conectividad extra a una página web obteniendo una red de internet usando Antenas Ubiquiti que darán una mejor conectividad sin importar el lugar de implementación a distancia siempre y cuando estén a un rango visible de conectividad y contemplando las distancias de enlace que las antenas proveen.

Blynk IoT permitirá enviar los datos al internet mediante ESP32, y gracias a esta plataforma se podrá accedes a los datos mediante API REST, conectando así al servidor de la base de datos y a la página web para su presentación en tiempo real.

El aspecto fundamental del sistema se establecerá a través de Django, con el desarrollo de una aplicación web diseñada para la visualización en tiempo real y presentación de datos desde una base de datos SQL. Node-RED, alojado en una Raspberry Pi, se encargará de la gestión del almacenamiento de datos.

En última instancia, la página Django se desplegará en un servidor para su acceso a través de un dominio web. El proyecto tiene como objetivo mejorar la funcionalidad, el control y la visualización del módulo interactivo, aprovechando una variedad de tecnologías para lograr una solución integral.

1. EL PROBLEMA

1.1 Planteamiento del problema

En la actualidad, el módulo interactivo existente carece de las capacidades tecnológicas necesarias para una recolección eficiente de datos, así como para una visualización y control remoto efectivos. La falta de una infraestructura moderna y la obsolescencia de los sistemas actuales limitan la funcionalidad y la utilidad práctica del módulo.

Adicionalmente, la ausencia de una plataforma integral para la gestión de datos y la interconexión con tecnologías emergentes como IoT y comunicación inalámbrica, impide aprovechar al máximo el potencial del módulo interactivo. Esto genera una brecha en término de eficiencia, control y capacidad de respuesta a tiempo real, aspectos cruciales para su aplicación en diversos contextos.

En este contexto, surge la necesidad de repotenciar el módulo interactivo mediante la implementación de tecnologías como ESP32, Blynk IoT, Ubiquiti radioenlace, Django, y Node-RED, se obtiene como objetivo de superar las limitaciones existentes y proporcionar una solución integral que mejore significativamente la funcionalidad y la versatilidad del módulo.

1.2 Delimitación del problema

El Proyecto de Titulación se desarrolló en la Universidad Politécnica Salesiana sede Guayaquil, Ecuador en el período 2024.

1.3 OBJETIVOS

1.3.1 Objetivo General

Diseñar e implementar una red de comunicación de radio enlace con antenas Ubiquiti para proveer de internet al módulo de sensores para repotenciar la tecnología existente en una tecnología integradora usando bases de datos, una Raspberry Pi, servidor en tiempo real Node-RED, y una página web de tipo dashboard usando Framework DJANGO basado en programación Python, HTML.

1.3.2 Objetivos Específicos

1. Programación de los sensores del módulo a repotenciar para interacción con el medio físico.

2. Establecer una interfaz de usuario intuitiva y funcional usando Blynk IoT para permitir el monitoreo remoto y adquirir la bondad de comunicación API REST para leer mediante método GET los datos de los sensores programados en la plataforma.

3. Configurar e implementar las antenas de radioenlace para lograr una comunicación eficiente entre el módulo de sensores con Internet y otros dispositivos.

4. Desarrollar una aplicación web con Django que permita visualizar los datos en tiempo real e interactuar con una base de datos SQL y crear informes de la recopilación de datos en Excel.

5. Configurar Node-RED en una raspberry para gestionar de forma eficiente el almacenamiento de datos en la base de datos SQL.

1.4 Justificación del proyecto

La justificación para este proyecto radica en la necesidad de mejorar las capacidades y funcionalidades de un módulo interactivo existente mediante la aplicación de tecnologías avanzadas. A continuación, se detallan algunas razones clave que respaldan la realización de este proyecto:

- Optimización de la Adquisición de Datos:

La implementación de ESP32 y sensores permitirá una adquisición de datos más eficiente, mejorando la precisión y la velocidad de recolección de información relevante.

- Interconexión y Control Remoto:

La incorporación de Blynk IoT facilitará la creación de una interfaz de usuario amigable, posibilitando el monitoreo remoto y el control del módulo desde dispositivos móviles en tiempo real.

- Comunicación Inalámbrica Eficiente:

La utilización de antenas de radioenlace Ubiquiti garantizará una comunicación inalámbrica eficiente, posibilitando la transmisión rápida y confiable de datos entre el módulo interactivo y otros dispositivos.

- Visualización y Almacenamiento Avanzados:

La implementación de Django y Node-RED junto con una base de datos SQL ofrecerá una plataforma avanzada para la visualización en tiempo real y el almacenamiento de datos, mejorando significativamente la capacidad de gestionar y analizar la información recopilada.

- Versatilidad y Aplicaciones Prácticas:

Al lograr una solución integral que abarque desde la adquisición de datos hasta la visualización y almacenamiento avanzados, el módulo interactivo mejorado se volverá más versátil y aplicable en una variedad de ambientes y aplicaciones, desde la educación hasta la investigación y la monitorización de variables específicas.

En conjunto, estas mejoras buscan superar las limitaciones actuales del módulo interactivo, ofreciendo una solución tecnológicamente avanzada y adaptable a diversos contextos, lo que respalda la pertinencia y necesidad de llevar a cabo este proyecto.

1.5 Hipótesis

Implementar la repotenciación del módulo interactivo mediante la integración de tecnologías como ESP32, Blynk IoT, Ubiquiti radioenlace, Django y Node-RED, podría resultar en una mejora significativa en la eficiencia de adquisición de datos, la interactividad remota, la comunicación inalámbrica, así como en la visualización y almacenamiento avanzados de datos.

Esta mejora integral podría generar un módulo interactivo más versátil, adaptable y funcional, con aplicaciones prácticas en diversos escenarios, desde la educación hasta la monitorización en tiempo real en un ambiente específicos implementado a futuro ciencia de datos, inteligencia artificial, entre otras.

1.6 Variable e Indicadores

1.6.1 Variables

Variable Independiente:

- Implementación de la Repotenciación del Módulo Interactivo.

Variables Dependientes:

- Adquisición de Datos Mejorada.
- Interactividad Remota (a través de Blynk IoT).
- Comunicación Inalámbrica Eficiente (utilizando antenas de radioenlace Ubiquiti).
- Visualización en Tiempo Real (mediante Django).
- Almacenamiento Avanzado de Datos (a través de Node-RED y una base de datos SQL).

1.6.2 Indicadores

Adquisición de Datos Mejorada:

- Frecuencia de muestreo.
- Precisión de los datos recopilados.
- Velocidad de transmisión de datos.

Interactividad Remota:

- Tiempo de respuesta en el control remoto.
- Estabilidad de la conexión remota.
- Funcionalidades disponibles en la interfaz Blynk.

Comunicación Inalámbrica Eficiente:

- Velocidad de transmisión de datos inalámbricos.
- Fiabilidad de la comunicación inalámbrica.

Visualización en Tiempo Real:

- Tiempo de respuesta en la visualización de datos.
- Claridad y accesibilidad de la interfaz de visualización.

Almacenamiento Avanzado de Datos:

- Eficiencia en el manejo y almacenamiento de datos por parte de Node-RED.
- Integridad y seguridad de la base de datos SQL.
- Capacidad de escalabilidad del sistema de almacenamiento.

1.7 METODOLOGÍA

1.7.1 Método Inductivo

Este tipo de método se pondrá a prueba los conocimientos de Redes de comunicaciones, programación de microcontroladores, telemática, sistemas operativos, programación de software.

1.7.2 Método Teórico y Sistemático

- Funcionamiento de ESP32.
- Funcionamiento de Raspberry Pi.
- Funcionamiento de Sensores.
- Funcionamiento de antenas de Radioenlace.
- Software de programación y software de configuración.
- Investigación de información referente al tema en internet.
- Búsqueda de alojamientos para página web.

Estos puntos considerados fueron tomados para la realización de este proyecto para implementarlo de la forma más rápida y eficiente.

1.7.3 Técnica Documental

Se recopila datos clave del proyecto, así como la búsqueda de las bases académicas, análisis de artículos del proyecto a repotenciar, y análisis de los datos recopilados para documentar este proyecto.

1.8 Población y Muestra

1.8.1 Población

El grupo analizado abarca a todos los estudiantes de la carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana.

1.8.2 Muestra

De acuerdo a la muestra, están los estudiantes de la carrera de Ingeniería Electrónica que cursan materias técnicas que involucran las Redes y comunicaciones, programación de microcontroladores, programación básica e intermedia, automatización Industrial.

1.9 Descripción de la propuesta

Este proyecto busca la modernización de un módulo interactivo mediante la implementación de tecnologías avanzadas. La integración de ESP32, Blynk IoT, y antenas de radioenlace Ubiquiti optimizará la adquisición de datos y permitirá el control remoto eficiente.

Los sensores proporcionados por la universidad se utilizarán para la recolección de datos, que se almacenarán en una base de datos SQL mediante Django y Node-RED en una Raspberry Pi. El objetivo es mejorar la funcionalidad y versatilidad del módulo, permitiendo su acceso y visualización a través de un dominio web. Este enfoque integral aprovecha diversas tecnologías para una solución avanzada y adaptable.

1.10 Beneficiarios

Los beneficiarios de la repotenciación de un módulo de sensores serán los estudiantes de la carrera Ingeniería Electrónica de la Universidad Politécnica Salesiana.

1.11 Impacto

Los alumnos tendrán la oportunidad de interactuar con los equipos y adquirir nuevas habilidades académicas en áreas como redes de comunicación, desarrollo web y protocolos de red.

2. MARCO TEÓRICO

2.1 ESP32

El ESP32 es un microcontrolador de bajo consumo de energía y alta potencia de procesamiento, desarrollado por Espressif Systems. Este dispositivo se destaca por sus capacidades inalámbricas y su arquitectura dual-core, lo que lo hace ideal para una variedad de aplicaciones en el ámbito de la electrónica y la Internet de las cosas (IoT) (Espressif Systems (Shanghai) Co., 2016).

- Arquitectura Dual-Core:

El ESP32 presenta dos núcleos de procesamiento, lo que permite la ejecución simultánea de tareas y mejora el rendimiento general del sistema. Un núcleo se utiliza para la ejecución del programa principal, mientras que el otro puede encargarse de tareas específicas como la conectividad Wi-Fi y Bluetooth.

- Conectividad Inalámbrica:

El ESP32 integra módulos Wi-Fi y Bluetooth, ofreciendo una conectividad inalámbrica versátil. Esto facilita la comunicación con otros dispositivos, la conexión a redes locales y la implementación de soluciones IoT.

- Bajo Consumo de Energía:

El diseño del ESP32 incorpora modos de bajo consumo de energía, permitiendo un funcionamiento eficiente en aplicaciones alimentadas por batería. Esto es especialmente valioso para dispositivos IoT que deben operar durante períodos prolongados.

- Amplia Variedad de Periféricos:

El ESP32 cuenta con una variedad de periféricos, incluyendo puertos GPIO, UART, SPI, I2C, ADC, y más. Esto brinda flexibilidad para conectar sensores, actuadores y otros componentes a la placa.

- Programación y Desarrollo:

El desarrollo de aplicaciones para ESP32 se realiza comúnmente utilizando el entorno de desarrollo integrado (IDE) Arduino, lo que facilita la programación con su extensa comunidad y abundancia de bibliotecas.

- Soporte para Microporgramas OTA (Over-the-Air):

Una característica destacada es la capacidad de actualizar el firmware de manera inalámbrica a través de OTA, lo que simplifica la gestión de dispositivos distribuidos.

- Comunidad y Documentación Abundante:

La popularidad del ESP32 ha llevado a una comunidad activa y a una amplia documentación, proporcionando recursos valiosos para desarrolladores y diseñadores que buscan aprovechar al máximo las capacidades del dispositivo.

A continuación, se observa en la imagen el microcontrolador:

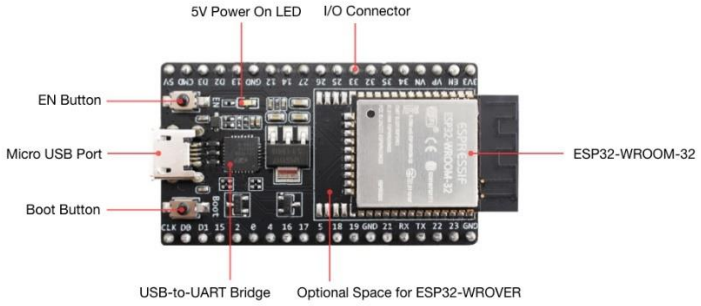


Figura 1: ESP32 (Espressif Systems (Shanghai) Co., 2016)

2.1.1 Especificaciones técnicas ESP32

Las características técnicas del ESP32 incluyen una variedad de especificaciones que lo hacen adecuado para aplicaciones en el campo de la electrónica y la Internet de las cosas (IoT).

A continuación, se especifica algunas de las características de las técnicas clave del ESP32:

- Arquitectura:

Doble núcleo Xtensa 32-bit, que permite la ejecución simultánea de tareas y mejora el rendimiento general del sistema.

- Frecuencia de Reloj:

Hasta 240 MHz, proporcionando una capacidad de procesamiento rápida.

- Memoria RAM:

Varía según el modelo, comúnmente entre 520 KB y 4 MB, facilitando el almacenamiento temporal de datos y programas.

- Conectividad Inalámbrica:

Integración de módulos Wi-Fi 802.11 b/g/n y Bluetooth 4.2 BR/EDR y BLE, ofreciendo conectividad versátil.

- Periféricos:

Amplia variedad de periféricos, incluyendo GPIO, UART, SPI, I2C, PWM, ADC, DAC, y más, que permiten la conexión de sensores, actuadores y otros dispositivos.

- Modos de Bajo Consumo de Energía:

Modos de bajo consumo, como el modo de espera profundo (Deep Sleep), para optimizar el consumo de energía en aplicaciones alimentadas por batería.

- Seguridad:

Soporte para Secure Boot y Flash Encryption, brindando características de seguridad avanzadas.

- Sistema de Alimentación:

Puede ser alimentado por un rango amplio de voltajes (generalmente de 2.2V a 3.6V), proporcionando flexibilidad en la selección de fuentes de energía.

- Puertos de Entrada/Salida (GPIO):

Número variable de puertos GPIO, dependiendo del modelo específico.

- Soporte OTA (Over-the-Air):

Capacidad para actualizar el firmware de manera inalámbrica, simplificando la gestión de dispositivos distribuidos.

- Entorno de Desarrollo:

Compatibilidad con el IDE Arduino y otras plataformas de desarrollo, facilitando la programación y la implementación de aplicaciones.

A continuación, se observa en la imagen el microcontrolador:

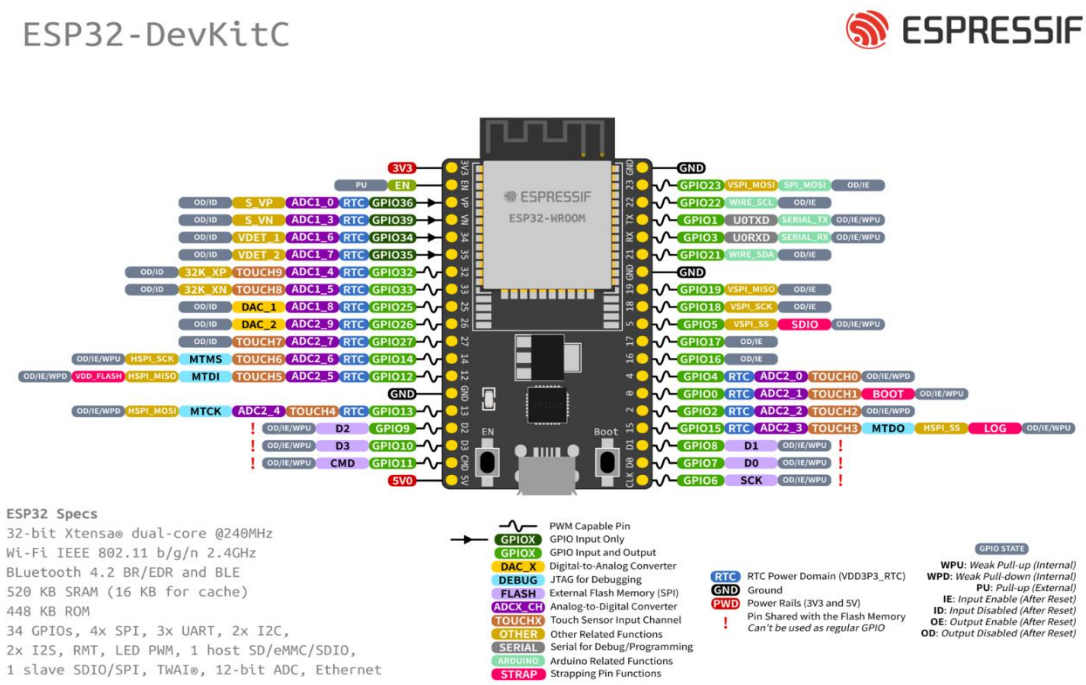


Figura 2: ESP32 GPIO (Espressif Systems (Shanghai) Co., 2016)

2.1.2 Software de programación Arduino IDE.

El Arduino IDE (Integrated Development Environment) es una plataforma de desarrollo de código abierto ampliamente utilizada para programar microcontroladores, y el ESP32 es compatible con el microcontrolador.

- Compatibilidad del ESP32 con Arduino IDE:

El Arduino IDE es compatible con el ESP32 mediante la instalación de un conjunto de herramientas específicas, lo que facilita la programación del microcontrolador usando el mismo entorno familiar de Arduino.

- Placa ESP32 en el Arduino IDE:

La comunidad Arduino proporciona un paquete para el ESP32, permitiendo que el IDE reconozca y soporte la placa ESP32. Este paquete incluye definiciones de pines, bibliotecas y archivos de configuración específicos para el ESP32.

- Lenguaje de Programación:

La programación en el Arduino IDE se realiza principalmente en lenguaje C++ con un conjunto de funciones y bibliotecas simplificadas. Esto facilita la creación de programas para el ESP32 incluso para aquellos que no son expertos en programación de bajo nivel.

- Uso de Bibliotecas de Arduino:

El Arduino IDE permite la utilización de una amplia variedad de bibliotecas desarrolladas por la comunidad Arduino. Esto simplifica el desarrollo, ya que estas bibliotecas abordan tareas comunes y permiten una programación más modular.

- Herramientas de Compilación y Carga Simplificadas:

El Arduino IDE proporciona botones de compilación y carga (upload) directa a la placa ESP32. Esto simplifica el proceso de desarrollo y carga de programas, especialmente para aquellos nuevos en el mundo de la programación de microcontroladores.

- Entorno Gráfico y Serial Monitor:

El Arduino IDE incluye un entorno gráfico intuitivo y un monitor serial que facilita la depuración y el monitoreo de datos en tiempo real durante la ejecución del programa.

- Soporte para OTA (Over-the-Air) Programming:

Se puede implementar la programación inalámbrica (OTA) utilizando herramientas específicas para ESP32 en el Arduino IDE, lo que permite actualizaciones remotas del firmware sin la necesidad de cables.

- Comunidad y Recursos Abundantes:

La comunidad Arduino es activa y proporciona una amplia gama de recursos, tutoriales y ejemplos para programar el ESP32 con el Arduino IDE.

A continuación, se observa en la imagen del software de programación de Arduino IDE:

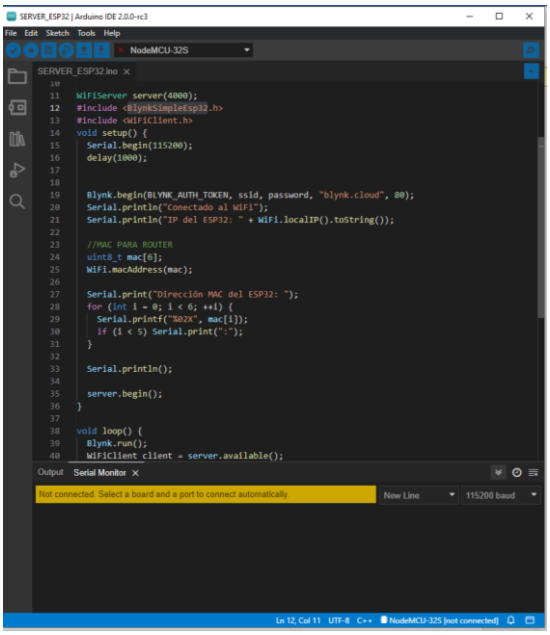


Figura 3: Software de programación Arduino IDE

2.2 Raspberry Pi

Raspberry Pi es una serie de computadoras de placa única (SBC) desarrollada por la Raspberry Pi Foundation. Estas computadoras compactas son versátiles y se utilizan en una variedad de aplicaciones, desde proyectos educativos hasta soluciones industriales (Pi, 2022).

- Arquitectura y Especificaciones:

Raspberry Pi utiliza una arquitectura de CPU ARM y ofrece diferentes modelos con variadas especificaciones, incluyendo memoria RAM, puertos USB, conexiones HDMI, entre otros. La versión y especificaciones seleccionadas dependerán de los requisitos específicos del proyecto.

- Sistema Operativo:

La Raspberry Pi es compatible con varios sistemas operativos, siendo Raspbian (ahora llamado Raspberry Pi OS) el sistema operativo oficial basado en Linux más comúnmente utilizado. Otros sistemas operativos, como Ubuntu, también son compatibles.

- GPIO (General Purpose Input/Output):

Raspberry Pi incluye pines GPIO que permiten la interconexión con sensores, actuadores y otros dispositivos electrónicos. Esto facilita la integración de la Raspberry Pi en proyectos de hardware.

- Interfaz de Usuario:

La Raspberry Pi puede ser operada desde una interfaz gráfica de usuario (GUI) o mediante la línea de comandos, proporcionando flexibilidad para diferentes aplicaciones y niveles de experiencia del usuario.

- Conectividad:

Raspberry Pi incluye puertos USB, Ethernet, Wi-Fi y Bluetooth, lo que facilita la conexión a periféricos, redes y dispositivos externo.

- Desarrollo de Aplicaciones:

Python es el lenguaje de programación predeterminado en Raspberry Pi y es ampliamente utilizado para el desarrollo de aplicaciones y proyectos. También se admiten otros lenguajes como C++.

- Node-RED:

Node-RED es una herramienta de programación visual basada en flujos que se ejecuta en Raspberry Pi. Facilita la conexión de dispositivos y servicios a través de nodos programables, lo que simplifica el desarrollo de aplicaciones IoT.

- Almacenamiento y Node.js:

Raspberry Pi puede utilizarse como un servidor para almacenamiento local y ejecución de aplicaciones web basadas en Node.js, permitiendo la creación de servicios web y aplicaciones interactivas.

- Proyectos y Comunidad:

La comunidad de Raspberry Pi es activa y ofrece una amplia variedad de proyectos y recursos. La Raspberry Pi Foundation promueve la educación y proporciona materiales educativos y tutoriales.

A continuación, se observa en la imagen el Raspberry Pi 4:

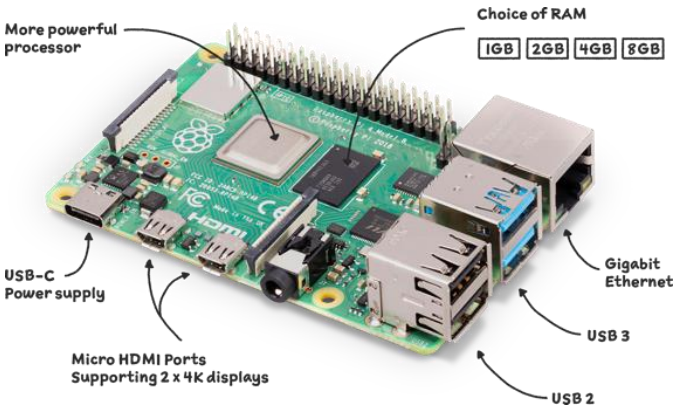


Figura 4: Raspberry Pi 4 (Pi, 2022)

A continuación, se observa en la imagen el Raspberry Pi 4 Pinout:

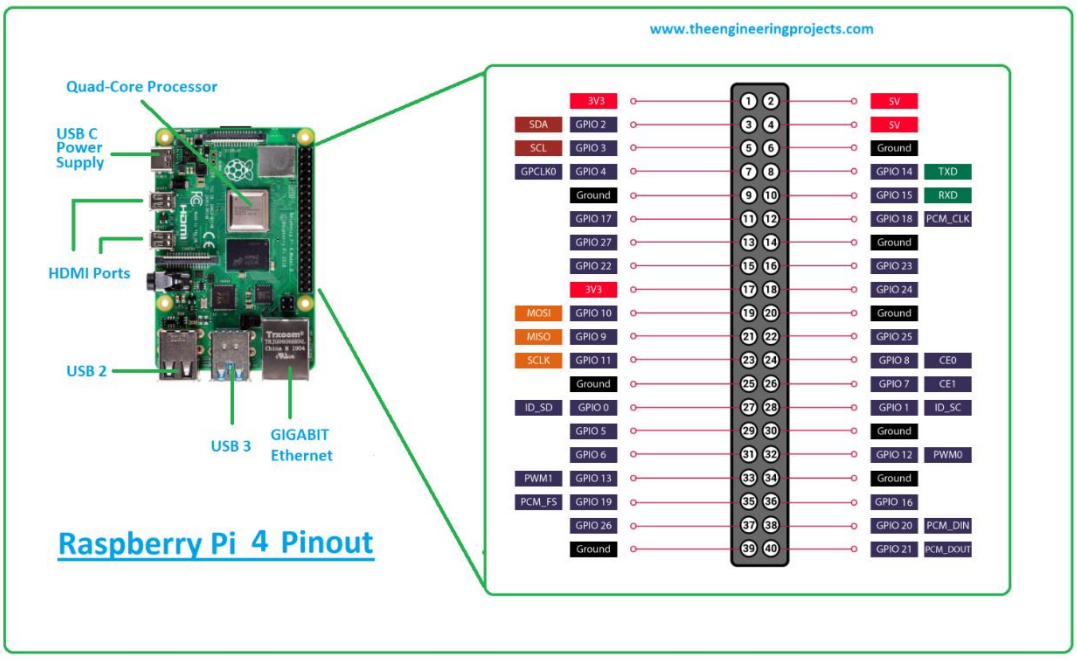


Figura 5: Raspberry Pi 4 Pinout (Pi, 2022)

2.3 Framework DJANGO

Django es un framework de desarrollo web de código abierto escrito en Python, diseñado para facilitar la creación eficiente de aplicaciones web robustas y seguras. Basándose en la arquitectura Modelo-Vista-Controlador (MVC), Django promueve el modularidad y la separación de las capas de datos, presentación y lógica de la aplicación.

Este framework incorpora un Mapeo Objeto-Relacional (ORM), permitiendo la interacción con la base de datos mediante objetos de Python en lugar de consultas SQL directas. Su sistema de plantillas simplifica la generación de contenido HTML dinámico. Las rutas y vistas definen cómo se manejan las solicitudes del usuario, estableciendo un patrón claro para el desarrollo web.

Django integra un sistema de administración automático que simplifica la gestión de modelos de datos sin requerir código adicional. Prioriza la seguridad, incluyendo funciones para prevenir vulnerabilidades comunes, como Cross-Site Request Forgery (CSRF) e inyecciones SQL.

El framework facilita la creación y validación de formularios, simplificando la obtención de datos del usuario. Ofrece sistemas integrados para autenticación y autorización, así como herramientas para el desarrollo basado en pruebas (TDD).

Django es altamente extensible, permitiendo la creación y reutilización de aplicaciones para mejorar el empaquetado del código. Su comunidad activa y la abundante documentación contribuyen a su robustez y facilidad de aprendizaje, convirtiéndolo en una opción popular para el desarrollo web (Django, 2024).

A continuación, se observa en la imagen Django:

Figura 6: Django (Django, 2024)

La estructura de un proyecto Django se compone de un directorio principal que contiene el proyecto Django en sí mismo, junto con varias aplicaciones Django dentro de él. Cada aplicación Django se enfoca en realizar una función específica dentro del proyecto.

```
myproject/
├── manage.py
└── myproject/
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

- manage.py: Es un script de utilidad que permite interactuar con un proyecto Django. Este script se utiliza para diversas tareas, como iniciar el servidor de desarrollo, crear nuevas aplicaciones dentro del proyecto, ejecutar migraciones de la base de datos, entre otras funciones.
- myproject/: Es el directorio principal del proyecto Django. Contiene todos los archivos y carpetas relacionados con el proyecto, incluyendo las configuraciones y las aplicaciones. Es el punto de entrada principal para administrar el proyecto.

- __init__.py: Es un archivo vacío que se utiliza para indicar a Python que el directorio debe considerarse un paquete. Esto permite importar módulos desde ese directorio.
- settings.py: En este archivo se encuentran todas las configuraciones del proyecto Django, como las claves secretas, las aplicaciones instaladas, las bases de datos, entre otras opciones de configuración importantes.
- urls.py: Este archivo define las URL del proyecto y cómo se mapean a las vistas. Aquí se especifican las rutas URL que la aplicación debe manejar y cómo deben ser procesadas por las vistas correspondientes.
- wsgi.py: Proporciona un punto de entrada compatible con WSGI (Web Server Gateway Interface) para la aplicación web. WSGI es una especificación que describe cómo los servidores web pueden comunicarse con aplicaciones web escritas en Python, y este archivo proporciona la interfaz necesaria para que la aplicación Django pueda ser servida por un servidor web compatible con WSGI.

Una aplicación Django es una colección de código Python que sirve para realizar una función específica. Una aplicación Django puede contener modelos de base de datos, vistas, URL, plantillas, archivos estáticos, etc. La estructura típica de una aplicación Django es la siguiente:

```
myapp/
├── __init__.py
├── admin.py
├── apps.py
├── migrations/
│   └── ...
├── models.py
├── tests.py
└── views.py
```

- __init__.py: Indica a Python que este directorio es un paquete.

- admin.py: Define la configuración de la interfaz de administración de Django para los modelos de esta aplicación.
- apps.py: Define la configuración de la aplicación, como el nombre legible para humano de la aplicación.
- migrations/: Directorio que contiene las migraciones de la base de datos para los modelos de la aplicación.
- models.py: Define los modelos de base de datos de la aplicación.
- tests.py: Contiene las pruebas unitarias para la aplicación.
- views.py: Define las vistas de la aplicación, es decir, las funciones o clases que manejan las solicitudes HTTP y devuelven las respuestas correspondientes.

2.4 Base de datos phpMyAdmin

El phpMyAdmin es una herramienta de administración de bases de datos MySQL basada en web, escrita en PHP. Su objetivo principal es proporcionar una interfaz gráfica intuitiva y fácil de usar para gestionar bases de datos MySQL (phpMyAdmin, 2024).

- Interfaz Gráfica para MySQL:

El phpMyAdmin ofrece una interfaz gráfica que permite a los usuarios interactuar con bases de datos MySQL a través de un navegador web. Esto facilita la administración de bases de datos sin necesidad de utilizar comandos SQL directos.

- Gestión de Bases de Datos:

Permite la creación, modificación y eliminación de bases de datos, proporcionando una visión general de la estructura y contenido de cada base de datos.

- Tablas y Relaciones:

Facilita la gestión de tablas, la creación de nuevas tablas, y la definición de relaciones entre ellas. Ofrece una vista visual de la estructura de las tablas y sus relaciones.

- Consultas SQL:

Además de la interfaz gráfica, phpMyAdmin permite a los usuarios ejecutar consultas SQL directamente, brindando flexibilidad para usuarios avanzados que prefieren comandos SQL.

- Importación y Exportación de Datos:

Proporciona herramientas para importar y exportar datos desde y hacia diferentes formatos, facilitando la migración de datos y la realización de copias de seguridad.

- Usuarios y Privilegios:

Permite la administración de usuarios y la asignación de privilegios a nivel de base de datos y tabla. Esto controla el acceso y los permisos de los usuarios en la base de datos.

- Operaciones en la Base de Datos:

Ofrece una amplia gama de operaciones, como la optimización de tablas, la reparación de tablas corruptas, y otras operaciones de mantenimiento de la base de datos.

- Compatibilidad con Caracteres y Collation:

Permite la configuración de conjuntos de caracteres y collation a nivel de base de datos, tabla y columna, asegurando la coherencia en el manejo de caracteres.

- Seguridad:

El phpMyAdmin incluye características de seguridad, como el control de acceso basado en roles y la capacidad de habilitar la autenticación de dos factores, garantizando la protección de la base de datos.

A continuación, se observa en la imagen phpMyAdmin:

Figura 7: phpMyAdmin (phpMyAdmin, 2024)

2.5 Internet de las cosas (IoT)

El Internet de las Cosas (IoT) es un paradigma tecnológico que conecta objetos físicos a la red, permitiéndoles recopilar y compartir datos. A continuación, se presenta un marco teórico para entender el concepto y las características clave del IoT:

- Definición del IoT:

El IoT se refiere a la interconexión de dispositivos físicos, vehículos, electrodomésticos y otros objetos mediante sensores y tecnologías de red. Estos objetos pueden recopilar y compartir datos para facilitar la automatización, la toma de decisiones y la mejora de la eficiencia.

- Sensores y Dispositivos Conectados:

En el contexto del IoT, los dispositivos están equipados con sensores y actuadores que les permiten interactuar con su entorno. Los sensores recopilan datos, como temperatura, humedad, ubicación, etc.

- Comunicación Máquina a Máquina (M2M):

La comunicación M2M es esencial en el IoT. Los dispositivos envían datos entre sí o a una plataforma central sin intervención humana directa. Esto permite la creación de sistemas autónomo y eficientes.

- Conectividad:

La conectividad en el IoT puede ser inalámbrica (Wi-Fi, Bluetooth, Zigbee, LoRa) o por cable. La elección de la tecnología depende de los requisitos específicos de la aplicación y del alcance de la red.

- Plataformas IoT:

Existen plataformas IoT que facilitan la gestión, recopilación y análisis de datos provenientes de dispositivos conectados. Estas plataformas suelen ofrecer servicios como almacenamiento en la nube, análisis de datos y paneles de control.

- Seguridad en el IoT:

La seguridad es una preocupación fundamental en el IoT debido a la gran cantidad de datos sensibles que se manejan. Se implementan medidas de seguridad como cifrado, autenticación y control de acceso para proteger la integridad y la privacidad de los datos.

- Estandarización:

Diversas organizaciones trabajan en estándares para el IoT, facilitando la interoperabilidad entre dispositivos y sistemas. Ejemplos incluyen MQTT, CoAP y el Protocolo de Mensajes Constrained Application (CoAP).

- Aplicaciones del IoT:

El IoT tiene aplicaciones en una amplia variedad de sectores, como la salud (dispositivos médicos conectados), agricultura (monitoreo de cultivos), ciudades inteligentes (gestión de tráfico y energía), y hogar inteligente (automatización residencial).

- Evolución del IoT:

El IoT sigue evolucionando con la incorporación de tecnologías emergentes como la inteligencia artificial, el aprendizaje automático y el 5G, lo que amplía aún más sus posibilidades y aplicaciones.

El Internet de las Cosas ha transformado la forma de interactuar con el entorno físico y ha dado lugar a innovaciones significativas en diversas industrias. La capacidad de conectar y recopilar datos de dispositivos en tiempo real ofrece oportunidades para mejorar la eficiencia, tomar decisiones informadas y crear soluciones más inteligentes y conectadas (Boreal, 2024).

A continuación, se observa en la imagen Internet de las cosas (IoT):

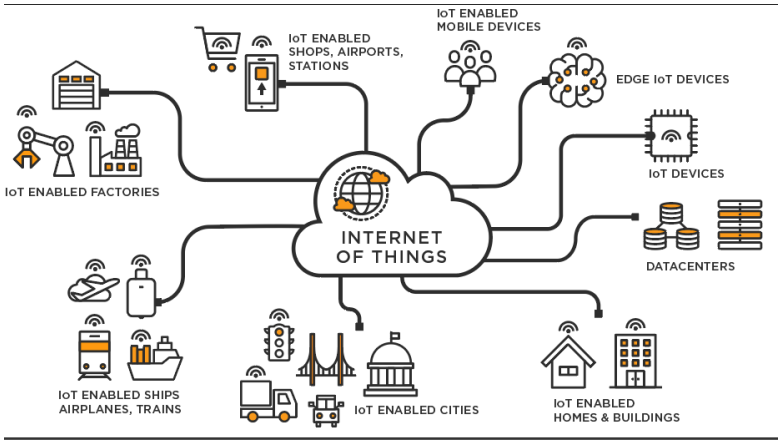


Figura 8: Internet de las cosas (IoT) (Boreal, 2024).

2.6 Digital Ocean

El DigitalOcean es una plataforma de nube que ofrece una variedad de servicios para alojamiento de aplicaciones y servidores virtuales. Con un enfoque en simplicidad y eficiencia, DigitalOcean se ha convertido en una opción popular para desarrolladores y empresas (Digital Ocean, 2024).

- Servidores Virtuales (Droplets):

El componente fundamental de DigitalOcean es el "droplet" o servidor virtual. Los droplets son instancias de máquinas virtuales que se pueden configurar y escalar según las necesidades del usuario.

- Sistemas Operativos y Stacks de Aplicaciones:

El DigitalOcean ofrece una variedad de sistemas operativos, siendo Ubuntu, CentOS y Debian opciones comunes. Los usuarios pueden seleccionar stacks de aplicaciones

preconfigurados para simplificar la implementación de software como servidores web, bases de datos, entre otros.

- Espacios de Almacenamiento y Volúmenes:

Además de los droplets, DigitalOcean proporciona servicios de almacenamiento como Spaces (almacenamiento de objetos) y Volúmenes (almacenamiento de bloques) para cubrir diversas necesidades de almacenamiento.

- Redes y Conectividad:

El DigitalOcean ofrece opciones flexibles de red, incluyendo direcciones IP flotantes, firewalls y balanceadores de carga para garantizar la conectividad y la seguridad de las aplicaciones desplegadas.

- Base de Datos:

La plataforma permite implementar y gestionar bases de datos, con opciones para PostgreSQL, MySQL, y Redis, facilitando la gestión de datos para aplicaciones.

- Gestión de Dominios y DNS:

El DigitalOcean proporciona herramientas para la gestión de dominios y la configuración de registros DNS, facilitando la vinculación de dominios a los recursos desplegados en la plataforma.

- Seguridad y Certificados SSL:

La seguridad es prioritaria en DigitalOcean, con opciones para la implementación de firewalls, monitoreo de seguridad y la integración sencilla de certificados SSL a través de Let's Encrypt.

- Despliegue y Escalabilidad:

La plataforma permite un despliegue rápido y escalabilidad flexible. Los usuarios pueden aumentar o disminuir recursos según las demandas de sus aplicaciones.

- Monitoreo y Registro:

El DigitalOcean ofrece herramientas de monitoreo y registro para que los usuarios puedan supervisar el rendimiento de sus recursos y aplicaciones.

- Backups y Snapshot:

El DigitalOcean facilita la realización de backups automáticos y la creación de snapshots (instantáneas) para garantizar la disponibilidad y recuperación de datos.

A continuación, se observa en la imagen la plataforma de la compañía Digital Ocean:

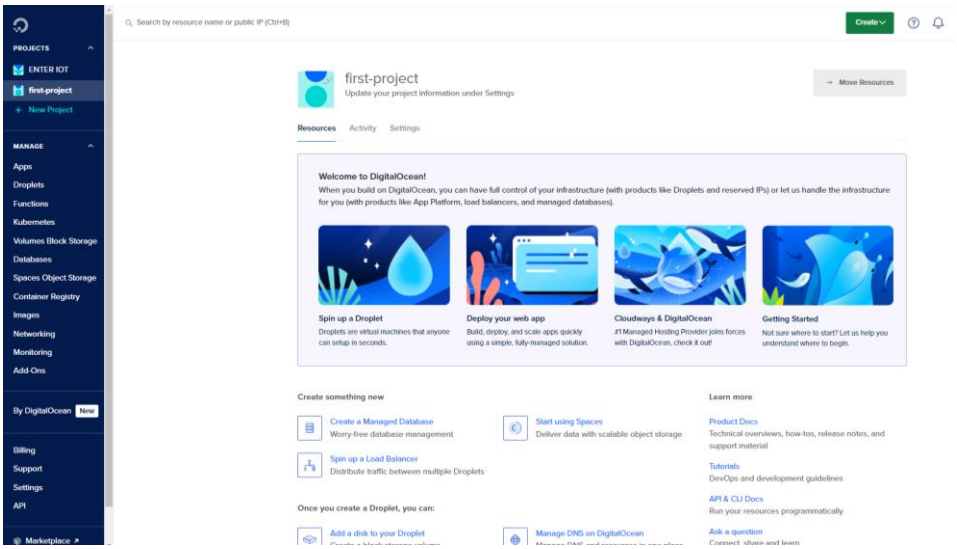


Figura 9: DigitalOcean

2.7 API REST

Una API REST, o Interfaz de Programación de Aplicaciones basada en Transferencia de Estado Representacional, es un conjunto de reglas y convenciones que permiten la comunicación entre sistemas a través del protocolo HTTP (Hypertext Transfer Protocol). El término "REST" fue acuñado por Roy Fielding en su tesis doctoral en 2000.

- Transferencia de Estado Representacional (REST):

REST es un estilo arquitectónico que se basa en principios predefinidos y restricciones para el diseño de sistemas distribuidos. Proporciona una interfaz uniforme para la

interacción entre componentes, lo que facilita la escalabilidad y la simplicidad en la comunicación.

- Principios REST:

1. Arquitectura Cliente-Servidor:

Separación clara entre el cliente (quien realiza las solicitudes) y el servidor (quien maneja las solicitudes y proporciona recursos).

2. Sin Estado (Stateless):

Por cualquier solicitud del cliente al servidor se debe contener la información en su totalidad para procesar dicha solicitud. Por ende, el servidor no retiene datos sobre el estado del cliente entre peticiones.

3. Cacheabilidad:

Las respuestas del servidor pueden ser marcadas como cacheables o no cacheables. La cacheabilidad mejora la eficiencia y la escalabilidad al permitir que las respuestas se almacenen en caché en diferentes capas del sistema.

4. Interfaz Uniforme:

Se define una interfaz uniforme que simplifica y generaliza las interacciones. Incluye la identificación de recursos, la manipulación de recursos a través de representaciones y la navegación a través de hipermedios.

5. Sistema de Capas:

Un sistema puede estar compuesto por varias capas (como firewalls, servidores intermedios, etc.), y cada capa solo debe conocer la capa adyacente, lo que mejora el modularidad y la escalabilidad.

6. Representación:

Los recursos son representados en formatos estándar (como JSON o XML) y se envían al cliente para su manipulación. La representación puede ser modificada y enviada de vuelta al servidor.

- Recursos y URI (Identificadores de Recursos Uniformes):

En REST, todo es un recurso (datos o servicios). Cada recurso es identificado por una URI única (Uniform Resource Identifier). Las operaciones (GET, POST, PUT, DELETE) se aplican a estos recursos para realizar acciones específicas.

- Formatos de Representación:

REST utiliza formatos de representación estándar como JSON o XML para intercambiar datos entre el cliente y el servidor. JSON es especialmente común debido a su simplicidad y legibilidad.

- Operaciones CRUD:

Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) son mapeadas a las operaciones HTTP estándar (POST, GET, PUT, DELETE) para interactuar con los recursos.

- Ventajas de las API REST:

1. Escalabilidad:

La arquitectura REST facilita la escalabilidad al permitir una distribución eficiente de los recursos.

2. Simplicidad y Flexibilidad:

La simplicidad de las operaciones HTTP y la representación de recursos en formatos estándar ofrecen flexibilidad y facilidad de implementación.

3. Independencia del Lenguaje:

Al utilizar formatos estándar como JSON, las APIs REST son independientes del lenguaje de programación y pueden ser consumidas por diversos clientes.

4. Mejora del Rendimiento:

La cacheabilidad y la transferencia eficiente de datos contribuyen a un mejor rendimiento.

5. Escalabilidad y Desacoplamiento:

La separación entre el cliente y el servidor permite una mayor escalabilidad y un menor acoplamiento entre los componentes del sistema.

Las API REST han ganado popularidad por su simplicidad, eficiencia y flexibilidad en la comunicación entre sistemas distribuidos en la web. Su enfoque en la representación de recursos y la interfaz uniforme ha llevado a su amplia adopción en el diseño de servicios web.

2.8 Antena Ubiquiti Loco M5

Ubiquiti Loco M5 es un dispositivo de la familia AirMax de Ubiquiti Networks, diseñado específicamente para aplicaciones de enlace punto a punto (PtP) y punto a multipunto (PtMP). Aquí se presenta un marco teórico para entender el Ubiquiti Loco M5:

- Tecnología AirMax:

Ubiquiti Loco M5 utiliza la tecnología AirMax de Ubiquiti Networks, que se centra en mejorar la eficiencia y el rendimiento de las conexiones inalámbricas, especialmente en zonas con alta interferencia.

- Frecuencia de Operación:

El Loco M5 opera en la banda de frecuencia de 5 GHz, lo que permite un mayor ancho de banda y una menor interferencia en comparación con la banda de 2.4 GHz.

- Antena Integrada:

Loco M5 tiene una antena integrada de doble polarización, lo que facilita la instalación y alineación para establecer conexiones PtP o PtMP.

- Ganancia y Potencia de Transmisión:

La antena del Loco M5 tiene una ganancia específica que ayuda a concentrar la energía de la señal en una dirección deseada. Además, el dispositivo tiene opciones de ajuste de potencia de transmisión para adaptarse a diferentes escenarios de despliegue.

- Modo de Operación:

El Loco M5 puede operar en varios modos, incluyendo Access Point (AP), Station (STA), y modo repetidor. Esto permite su uso en diferentes configuraciones de red.

- Protocolo TDMA (Time Division Multiple Access):

Ubiquiti Loco M5 implementa un protocolo TDMA para mejorar la eficiencia del enlace inalámbrico al asignar tiempos específicos para la transmisión de datos. Esto reduce la posibilidad de colisiones y mejora el rendimiento general.

- Seguridad y Encriptación:

Para garantizar la seguridad de las comunicaciones, el Loco M5 admite encriptación WPA2 y WPA3, proporcionando capas adicionales de protección.

- Interfaz de Configuración:

El dispositivo se configura a través de una interfaz basada en web que facilita la configuración y supervisión del enlace inalámbrico. Los usuarios pueden ajustar parámetros como frecuencia, potencia y modulación.

- Durabilidad y Resistencia a la Intemperie:

Diseñado para su uso en exteriores, el Loco M5 presenta una carcasa resistente a la intemperie que protege el dispositivo contra condiciones climáticas adversas.

- Despliegue Versátil:

El Loco M5 es versátil y puede ser utilizado en una variedad de aplicaciones, como enlaces punto a punto de larga distancia, distribución de Internet en áreas remotas o la extensión de redes inalámbricas existentes.

El Ubiquiti Loco M5 es una solución inalámbrica compacta pero potente, diseñada para ofrecer conectividad eficiente y confiable en ambientes donde se requiere un rendimiento superior y una instalación sencilla. Su diseño robusto y características avanzadas lo convierten en una elección popular para despliegues de red inalámbrica de alta calidad.

A continuación, se observa en la imagen Antena Ubiquiti Loco M5:

Figura 10: Antena Ubiquiti Loco M5 (Air Max, 2020)

2.9 Protocolos de comunicación (TCP, MQTT y HTTP)

TCP es un protocolo de comunicación de red que proporciona una transmisión de datos confiable y orientada a la conexión. Forma parte de la suite de protocolos de Internet (TCP/IP) y se utiliza para la transferencia de datos en numerosas aplicaciones, incluyendo la navegación web, correo electrónico y transferencia de archivos.

- Ventajas de TCP:

Fiabilidad: TCP garantiza la entrega ordenada y fiable de datos.

Detección de Errores: Utiliza mecanismos de detección y corrección de errores para garantizar la integridad de los datos.

Control de Flujo: Gestiona el flujo de datos para evitar congestiones en la red.

Orientado a la Conexión: Establece una conexión antes de la transmisión de datos, proporcionando un canal confiable.

- Desventajas de TCP:

Overhead: La garantía de entrega y los mecanismos de control añaden overhead a la transmisión.

Latencia: Puede introducir cierta latencia debido a la confirmación de recepción de datos.

Ineficiente para Datos Pequeños: Puede no ser eficiente para la transmisión de pequeñas cantidades de datos debido a la configuración de la conexión.

2.9.2 Protocolo MQTT (Message Queuing Telemetry Transport)

MQTT es un protocolo de mensajería ligero y eficiente diseñado para dispositivos con recursos limitados. Es ampliamente utilizado en el Internet de las Cosas (IoT) para la comunicación entre dispositivos conectados (HUAWEI, 2023).

- Ventajas de MQTT:

Bajo Consumo de Ancho de Banda: Es eficiente en término de uso de ancho de banda, ideal para dispositivos con conectividad limitada.

Publicación/Suscripción: Utiliza un modelo de publicación/suscripción que permite la comunicación eficiente entre múltiples dispositivos.

Calidad de Servicio (QoS): Ofrece niveles de QoS que permiten la gestión de la entrega de mensajes.

Mensajes Retenidos: Permite retener mensajes para nuevos suscriptores.

- Desventajas de MQTT:

Seguridad: Puede requerir capas adicionales de seguridad para proteger las comunicaciones.

Complejidad en Implementación: En comparación con protocolos más simples, la implementación de MQTT puede ser más compleja.

A continuación, se observa en la imagen MQTT:

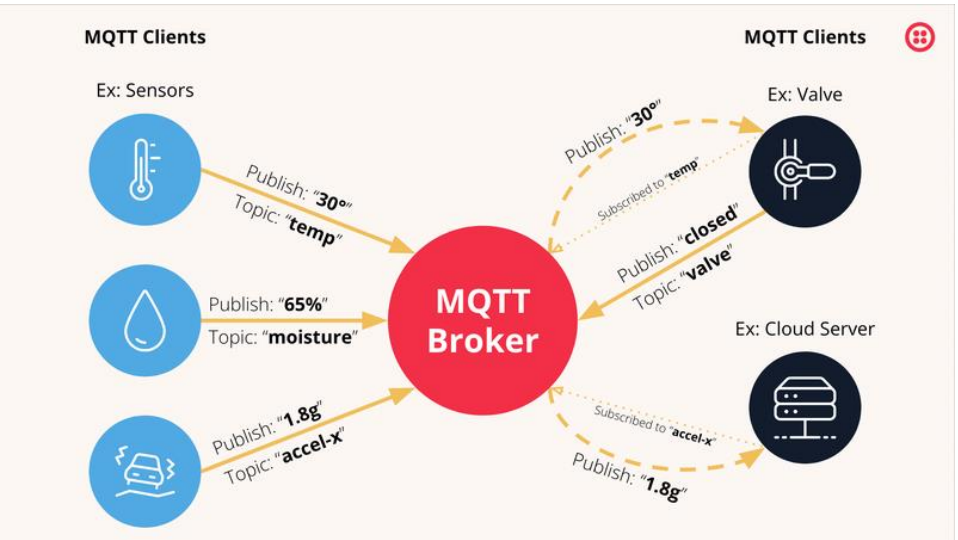


Figura 11: MQTT (HUAWEI, 2023)

A continuación, se observa en la imagen Dashboard MQTT:

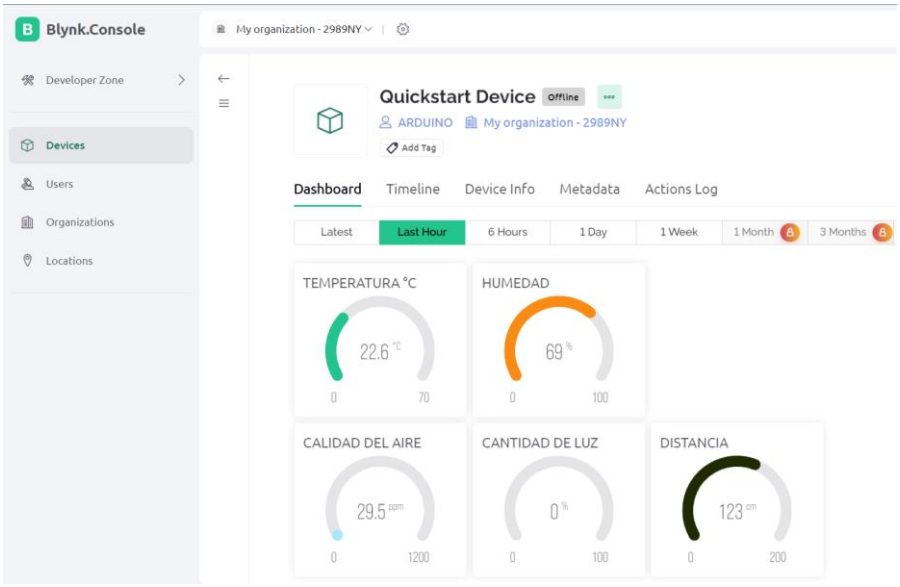


Figura 12: Dashboard MQTT

2.9.3 Protocolo HTTP (Hypertext Transfer Protocol)

HTTP es un protocolo de aplicación que se utiliza para la transferencia de información en la World Wide Web. Define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores y navegadores) para comunicarse (HUAWEI, 2023).

- Ventajas de HTTP:

Universalidad: Es ampliamente utilizado y es el protocolo estándar para la comunicación web.

Simplicidad: Su estructura es simple y fácil de entender.

Compatibilidad con Navegadores: Es compatible con navegadores web, lo que facilita la implementación en aplicaciones web.

- Desventajas de HTTP:

Estadoless: HTTP es un protocolo sin estado, lo que significa que cada solicitud se procesa de forma independiente sin conocimiento del estado anterior.

Seguridad Limitada: Puede carecer de características de seguridad avanzadas, lo que requiere el uso de HTTPS para la transmisión segura de datos.

A continuación, se observa en la imagen Protocolo HTTP:

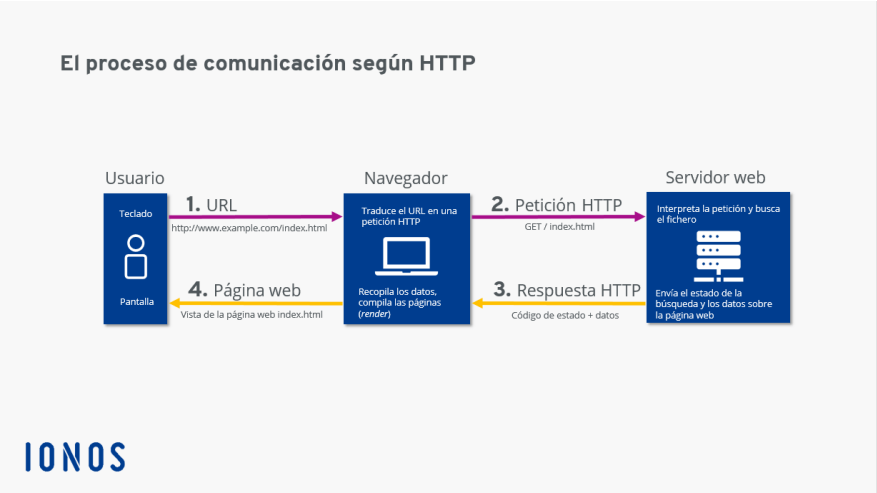


Figura 13: Protocolo HTTP (IONOS, 2020)

Comparativa:

Característica	TCP	MQTT	HTTP
Fiabilidad	Alta	Moderada a Alta	Moderada
Uso de Ancho de Banda	Moderado	Bajo	Moderado
Modelos de Comunicación	Orientado a Conexión	Publicación/Suscripción	Orientado a Mensajes
Complejidad	Alta	Moderada	Baja
Aplicaciones	Transferencia de Archivos, Navegación Web	IoT, Mensajería Instantánea	Navegación Web, Transferencia de Datos

2.10 Tablero de sensores de bajo costo

El objetivo del proyecto diseñado por Israel Brito es la interacción de sensores con el internet IoT utilizando placa de desarrollo ESP32, el módulo fue equipado por sensores de bajo costo para su uso didáctico, contiene conectores de tipo plug para realizar prácticas de laboratorio y pruebas (Brito, 2023).

A continuación, se observa en la imagen del módulo didáctico de sensores:

Figura 14: Módulo Didáctico de sensores

2.11 Sensor de Gas MQ-135

El sensor de gas MQ-135 es un dispositivo semiconductor diseñado para detectar varios gases en el aire. Se utiliza comúnmente para medir la concentración de gases como amoníaco, dióxido de carbono, metano y vapores de alcohol.

Principio de Funcionamiento: El MQ-135 opera según el principio de la resistencia eléctrica del material semiconductor en presencia de gases. La resistencia cambia en función de la concentración de gases, lo que permite medir y detectar su presencia.

Características Principales:

Sensibilidad a Gases Específicos:

El MQ-135 tiene sensibilidad a gases específicos, con respuestas particulares a diferentes sustancias químicas. Puede ser calibrado para detectar un gas específico según las necesidades de la aplicación.

Rango de Detección:

Tiene un rango de detección que abarca concentraciones bajas y moderadas de gases. Esto lo hace útil para aplicaciones de seguridad y control de calidad del aire.

Sensor de Semiconductores:

Utiliza un elemento semiconductor para medir la concentración de gases. El material semiconductor reacciona a la presencia de gases, alterando su resistencia eléctrica.

Calibración:

Puede requerir calibración periódica para mantener su precisión. La calibración ajusta el sensor para que proporcione lecturas precisas en el ambiente específicos.

Interfaz de Salida Analógica:

Proporciona una señal analógica proporcional a la concentración de gas detectado. Esta señal puede ser interpretada por microcontroladores o sistemas de monitoreo.

Sensibilidad a la Temperatura y Humedad:

La precisión del MQ-135 puede verse afectada por cambios en la temperatura y la humedad. Es importante tener en cuenta estas variables al interpretar las lecturas del sensor.

Aplicaciones:

- Calidad del Aire en Interiores:

El MQ-135 se utiliza para medir la calidad del aire en interiores, detectando niveles de dióxido de carbono y otros gases que pueden afectar la salud.

- Sistemas de Control de Ventilación:

Se integra en sistemas de control de ventilación para activar la ventilación cuando se detectan concentraciones elevadas de gases peligrosos.

- Dispositivos de Seguridad:

Se implementa en dispositivos de seguridad para alertar sobre posibles fugas de gases peligrosos, como en sistemas de detección de gas en el hogar.

- Proyectos de IoT y Monitoreo Ambiental:

Se utiliza en proyectos de Internet de las Cosas (IoT) para monitorear la calidad del aire en tiempo real en ambientes específicos.

A continuación, se observa en la imagen sobre el sensor calidad del aire MQ135:

Figura 15: MQ135 (Geekfactory, 2017)

2.12 Sensor Ultrasónico HC-SR04

El sensor ultrasónico HCSR04 es un dispositivo utilizado para medir distancias utilizando ondas ultrasónicas. Este sensor es conocido por su simplicidad y precisión en la medición de distancias sin contacto.

Principio de Funcionamiento: El HCSR04 utiliza la tecnología de ultrasonido para calcular la distancia entre el sensor y un objeto. El principio de funcionamiento se basa en la emisión de pulsos ultrasónicos y la medición del tiempo que tarda en recibir el eco de esos pulsos.

Características Principales:

- Transductor Ultrasónico:

El sensor contiene un transductor ultrasónico que emite pulsos de sonido ultrasónico, generalmente a una frecuencia de 40 kHz.

- Emisión y Recepción de Ultrasonido:

Emite una señal ultrasónica y mide el tiempo que tarda en recibir el eco de esa señal después de rebotar en un objeto.

- Cálculo de Distancia:

La distancia se calcula utilizando la velocidad del sonido en el aire y el tiempo transcurrido entre la emisión y la recepción del eco.

- Precisión y Resolución:

El HCSR04 ofrece una buena precisión en la medición de distancias, y su resolución está determinada por la duración de los pulsos ultrasónicos.

- Ángulo de Operación:

Tiene un ángulo de operación que define la zona en la que puede detectar objetos. Este ángulo varía según el modelo específico del sensor.

- Interfaz de Control:

Se controla fácilmente a través de pulsos de entrada y salida, y la medición de la distancia se obtiene leyendo el tiempo de eco.

A continuación, se observa en la imagen el sensor Ultrasónico HC-SR04:

Figura 16: Sensor Ultrasónico HC-SR04 (Gines, 2019)**.**

2.13 Sensor DHT22

El sensor DHT22, también conocido como AM2302, es un sensor de temperatura y humedad relativa de alta precisión. Diseñado para aplicaciones de monitoreo ambiental, el DHT22 proporciona lecturas digitales de temperatura y humedad en tiempo real.

Principio de Funcionamiento: El DHT22 utiliza un sensor capacitivo para medir la humedad relativa y un termistor para medir la temperatura. Ambos elementos están integrados en el mismo paquete y se comunican a través de un único cable.

Características Principales:

- Sensor de Humedad Capacitivo:

Emplea un sensor capacitivo para medir la humedad relativa del aire. Este tipo de sensor detecta la capacitancia en función de la humedad.

- Termistor para Temperatura:

Utiliza un termistor, un tipo de resistor sensible a la temperatura, para medir la temperatura del ambiente.

- Rango de Medición:

El DHT22 tiene un amplio rango de medición, típicamente desde -40°C hasta 80°C para la temperatura y del 0% al 100% para la humedad relativa.

- Precisión:

Ofrece una precisión aceptable, siendo comúnmente utilizado para aplicaciones de monitoreo general y sistemas de control ambiental.

- Interfaz de Salida Digital:

Proporciona lecturas digitales de temperatura y humedad a través de un solo cable de datos, facilitando la integración con microcontroladores.

- Calibración Integrada:

Viene calibrado de fábrica, eliminando la necesidad de ajustes manuales y mejorando la precisión de las lecturas.

A continuación, se observa en la imagen sensor rango de humedad DHT22:

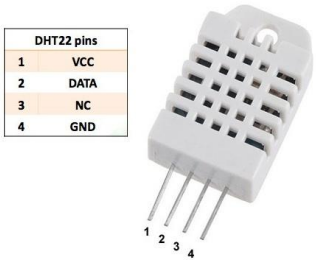


Figura 17: Sensor DHT22 (Bricogeek, 2022)**.**

2.14 Sensor LDR LM393

La LDR es un resistor cuya resistencia varía en función de la intensidad de la luz incidente sobre ella. Cuanto mayor es la intensidad de la luz, menor es la resistencia de la LDR, y viceversa.

Principio de Funcionamiento: La LDR se basa en el efecto fotoconductivo, donde la luz incidente sobre el semiconductor provoca un cambio en la resistencia eléctrica del dispositivo.

Aplicaciones:

- Se utiliza comúnmente en circuitos de control de luz, como en sistemas de iluminación automática.
- En aplicaciones de seguridad para detectar la presencia o ausencia de luz.

A continuación, se observa en la imagen sensor de intensidad de luz:

Figura 18: Fotorresistencia (Portilla, 2015)**.**

2.15 Blynk IoT

Blynk es una plataforma de desarrollo de IoT (Internet de las Cosas) que facilita la creación de proyectos conectados a Internet y controlados a través de dispositivos móviles. Blynk proporciona una interfaz amigable para la creación de aplicaciones personalizadas que interactúan con hardware en tiempo real.

Principios Clave:

- Interfaz Gráfica Intuitiva:

Blynk ofrece una interfaz gráfica intuitiva para diseñar aplicaciones IoT sin necesidad de programación extensiva. Los usuarios pueden arrastrar y soltar widgets para controlar hardware y visualizar datos.

- Widgets Personalizables:

Ofrece una variedad de widgets personalizables, como botones, deslizadores, gráficos y más, que permiten controlar y monitorear dispositivos conectados.

- Conectividad a Diversos Hardware:

Compatible con una amplia gama de placas de desarrollo y microcontroladores populares, como Arduino, Raspberry Pi y ESP8266, permitiendo la conexión de diferentes dispositivos al ecosistema Blynk.

- Comunicación en Tiempo Real:

Facilita la comunicación en tiempo real entre la aplicación Blynk y el hardware, lo que permite actualizaciones instantáneas y control remoto.

- API para Desarrolladores:

Proporciona una API para desarrolladores que permite la integración de Blynk con otras plataformas y servicios, ampliando las posibilidades de desarrollo.

- Blynk Cloud:

La plataforma Blynk opera en la nube (Blynk Cloud), eliminando la necesidad de configuraciones complejas de red y permitiendo un acceso fácil y seguro desde cualquier lugar.

Aplicaciones Prácticas:

- Monitoreo de Sensores:

Utilizado para la recolección y visualización de datos de sensores en tiempo real, como temperatura, humedad, o calidad del aire.

- Sistemas de Riego Automático:

Desarrollo de sistemas de riego que pueden ser controlados y monitoreados a distancia.

Consideraciones de Uso:

Seguridad:

Al implementar proyectos IoT, es crucial considerar medidas de seguridad para proteger la integridad de los datos y la privacidad.

- Conectividad a Internet:

Blynk requiere una conexión a Internet para operar, por lo que es necesario considerar la disponibilidad de la conexión en el entorno de implementación.

Blynk proporciona una solución accesible y poderosa para aquellos interesados en explorar proyectos de IoT sin requerir una profunda experiencia en programación. Su interfaz intuitiva y versatilidad hacen de Blynk una opción popular para el desarrollo de proyectos IoT interactivos y conectados.

A continuación, se observa en la imagen Blynk Dashboard:

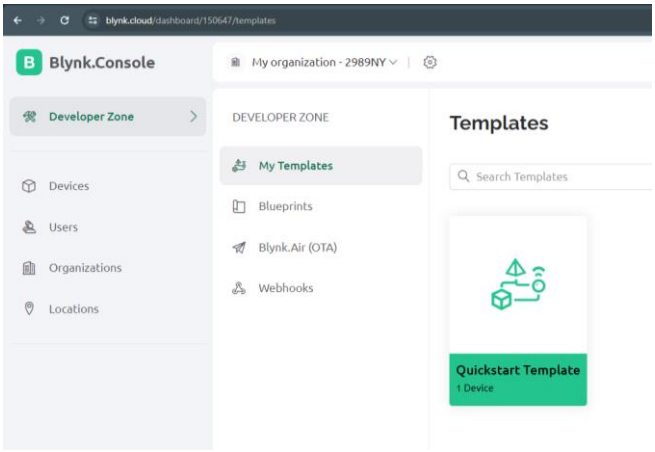


Figura 19: Blynk Dashboard.

3. MARCO METODOLÓGICO

3.1 Descripción del módulo didáctico

El proyecto de potenciación del módulo didáctico se centra en mejorar su funcionalidad y accesibilidad mediante la integración de tecnologías avanzadas y la implementación de un entorno de desarrollo completo. El módulo base cuenta con un controlador ESP32 y una variedad de sensores para la medición de variables físicas como distancia, humedad, temperatura, cantidad de luz y calidad del aire, además de entradas y salidas digitales configurables.

Componentes Agregados:

1. Raspberry Pi como Servidor de Página Web en Django: Se incorporará una Raspberry Pi para actuar como servidor web, utilizando el framework Django para el desarrollo de una interfaz de usuario dinámica y amigable.

2. Node-RED para Servidor de Grabación de Datos en Base de Datos: La implementación de Node-RED permitirá gestionar y grabar los datos obtenidos por los sensores en una base de datos de manera eficiente y escalable.

3. Base de Datos PHPMyAdmin: Se utilizará PHPMyAdmin como herramienta de gestión de base de datos para almacenar y organizar los datos recopilados por el sistema.

4. Blynk IoT como API de Integración: Se empleará Blynk IoT como interfaz de programación de aplicaciones (API) para facilitar la integración entre la página web desarrollada en Django y los datos obtenidos por el controlador ESP32.

5. Digital Ocean para Alojamiento de la Página Web con Dominio Propio: Se utilizará la plataforma de alojamiento en la nube Digital Ocean para hospedar la página web desarrollada, proporcionando un dominio propio para facilitar el acceso y la difusión del proyecto.

A continuación, se observa en la imagen sobre el esquema del proyecto:

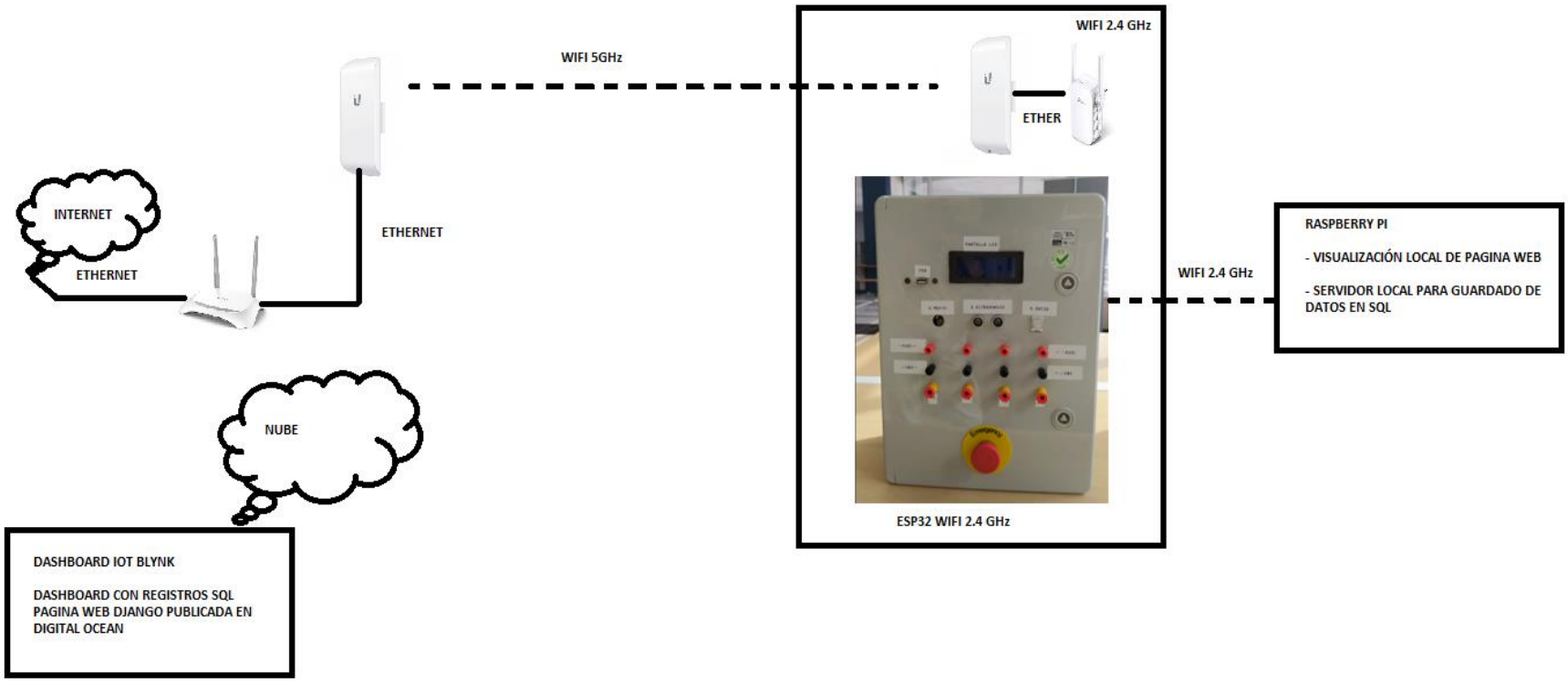


Figura 20: Esquema de proyecto

3.2 Programación en ARDUINO IDE

Se realiza la programación en ARDUINO IDE usando y configurando el entorno para ESP32, se utilizan varios sensores para recopilar datos de temperatura, humedad, distancia, cantidad de luz y calidad de aire usando MQ135, se envían los datos a plataforma BLYNK a través de wifi para su visualización y control remoto.

Explicación del código:

1. Inclusión de bibliotecas y definición de constantes:

El código comienza incluyendo las bibliotecas necesarias para el funcionamiento del programa, como MQ135 para el sensor de calidad del aire MQ135, Wire para la comunicación I2C, LiquidCrystal_I2C para el control de la pantalla LCD, y varias otras para el funcionamiento de los sensores DHT22 y el módulo ESP32. Además, se definen constantes como el pin utilizado para el sensor MQ135 (PIN_MQ135) y la configuración de red Wi-Fi (ssid y pass).

2. Inicialización y configuración:

En la función setup(), se inicializan los objetos de los sensores, la pantalla LCD, la conexión serial y Blynk. También se configuran y se imprimen detalles sobre los sensores DHT22 y MQ135.

Función de bucle principal (loop()):

Esta función se ejecuta repetidamente en un bucle infinito. Aquí se realizan las siguientes tareas:

Se leen los datos de temperatura y humedad del sensor DHT22.

Se leen los datos de distancia del sensor ultrasónico.

Se leen los datos de calidad del aire (resistencia y PPM) del sensor MQ135.

Se Leen los datos de la cantidad de luz receptada por un LDR.

Se envían los datos recopilados a la plataforma Blynk para su visualización remota.

Se actualiza la pantalla LCD con los datos recopilados.

3. Manejo de eventos de Blynk:

Se definen funciones de devolución de llamada de Blynk, como BLYNK_WRITE y BLYNK_CONNECTED, que se utilizan para manejar eventos como la escritura en pines virtuales y la conexión exitosa a Blynk Cloud. Por ejemplo, en BLYNK_CONNECTED(), se cambia el mensaje de un botón web en la interfaz de usuario de Blynk y se establece un enlace URL.

A continuación, se observa en la imagen sobre la codificación Arduino IDE ESP32 :

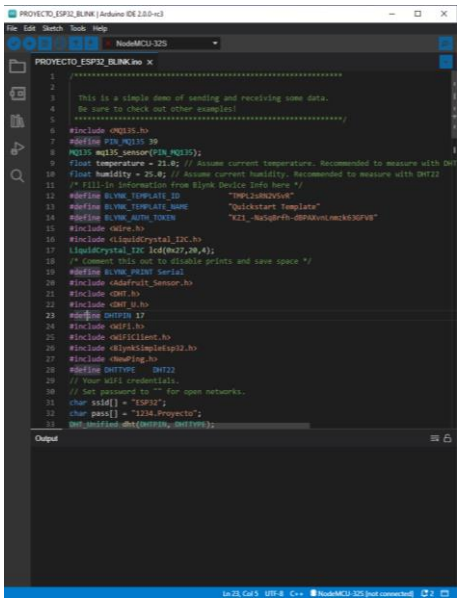


Figura 21: Arduino IDE ESP32

3.3 Configuración de Blynk IoT

Se crea un proyecto en Blynk IoT usando una cuenta de Gmail de Google y se realizan los siguientes pasos:

1. Creación del Proyecto Blynk:

Se inicia sesión en la aplicación Blynk.

Se crea un nuevo proyecto, seleccionando el tipo de hardware adecuado (en este caso, ESP32) y el tipo de conexión (Wi-Fi).

Al crear el proyecto, se asigna un token de autenticación (AUTH TOKEN), necesario para conectar el dispositivo ESP32 al proyecto.

2. Agregar Widgets al Dashboard:

Se utiliza la interfaz de usuario de la aplicación Blynk para agregar widgets que representen las variables que se desean monitorear. Estos pueden incluir:

Widgets de valor numérico (Value Display) para temperatura, humedad y calidad del aire (PPM).

Un widget de gráfico en tiempo real (Real-time Graph) para mostrar la distancia medida por el sensor ultrasónico.

Otros widgets adecuados para representar el valor de la LDR, como un widget de valor numérico o un gráfico en tiempo real.

3. Configuración de Widgets:

Se personaliza la configuración de cada widget para reflejar las preferencias de visualización, incluyendo unidades de medida, rangos de valores, colores y etiquetas.

Se asignan los pines virtuales correspondientes a cada widget. Por ejemplo, el pin virtual V0 puede asignarse para la temperatura, V1 para la humedad, V2 para la distancia, V3 para la calidad del aire y V4 para el valor de la LDR.

4. Integración con el Código del Dispositivo ESP32:

En el código de Arduino para el ESP32, se asegura haber incluido la biblioteca Blynk (#include <BlynkSimpleEsp32.h>).

Después de leer el valor de la LDR en la función loop(), se utiliza la función Blynk.virtualWrite() para enviar este valor al pin virtual correspondiente (por ejemplo, Blynk.virtualWrite(V4, valor_LDR)).

5. Subir el Código al Dispositivo:

Se conecta el dispositivo ESP32 a la computadora.

Selecciona la tarjeta y el puerto correctos en el entorno de desarrollo Arduino.

Se carga el código en el dispositivo.

6. Monitoreo en Tiempo Real:

Una vez que el dispositivo ESP32 esté conectado a Blynk, se puede monitorear en tiempo real las variables ambientales y de calidad del aire, así como el valor de la LDR, a través del dashboard de la aplicación Blynk en el dispositivo móvil.

Se realizan ajustes adicionales en la configuración del widget o en el código del dispositivo según sea necesario para una experiencia óptima de monitoreo y visualización de datos.

7. Obtención de la API desde Blynk:

Una vez que el proyecto esté creado en Blynk, el usuario puede obtener la API del proyecto desde la configuración del proyecto en la aplicación Blynk. La API proporciona acceso a los datos y la capacidad de realizar acciones en el proyecto a través de solicitudes HTTP.

Valores para leer con API REST desde Blynk:

Temperatura en el pin virtual V0.

Humedad en el pin virtual V1.

Distancia en el pin virtual V2.

Calidad del aire en el pin virtual V3.

Valor de la LDR en el pin virtual V4.

Ejemplo de API:

https://blynk.cloud/external/api/get?token=Rps15JICmtRVbFyS_95houlLbm6xIQ2L&v1

A continuación, se observa en la imagen Blynk Dashboard:

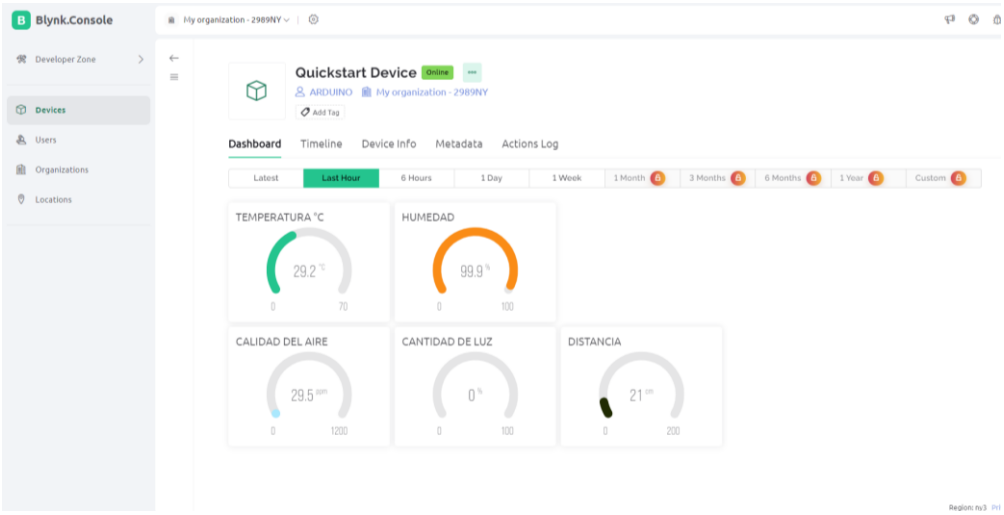


Figura 22: Blynk Dashboard

3.4 Configuración de Raspberry pi 4 con servidor Node-Red.

Para la configuración se puede usar una pantalla HDMI o se ingresa a VNC Viewer con la dirección IP de Raspberry Pi

Requisitos Previos:

- Una Raspberry Pi 4 con una instalación funcional de Raspberry Pi OS.
- Conexión a internet estable.

Pasos para Configurar el Servidor Node-RED:

Actualizar el Sistema Operativo:

- Se abre una terminal en la Raspberry Pi 4.
- Se ejecutan los siguientes comandos para asegurarse de que el sistema operativo y todos los paquetes estén actualizados:

sudo apt update sudo apt upgrade

- Instalar Node.js:

Node-RED requiere Node.js. Se instala ejecutando los siguientes comandos:

sudo apt install -y nodejs npm

- Instalar Node-RED:

Se utiliza npm para instalar Node-RED de forma global:

sudo npm install -g --unsafe-perm node-red

- Iniciar Node-RED:

Una vez instalado, se puede iniciar Node-RED ejecutando el siguiente comando:

node-red

- Acceder a Node-RED:

Se abre un navegador web en el dispositivo y se navega a la dirección IP de la Raspberry Pi seguida por el puerto 1880. Por ejemplo: http://<IP_Raspberry>:1880.

- Configuración Automática de Arranque:

Para que Node-RED se inicie automáticamente cada vez que se inicie la Raspberry Pi, se utiliza el administrador de servicios systemd.

- Se crea un archivo de servicio para Node-RED:

sudo nano /etc/systemd/system/nodered.service

Se pega el siguiente contenido en el archivo:

Description=Node-RED graphical event wiring tool After=syslog.target network.target [Service] ExecStart=/usr/bin/node-red-pi --max-old-space-size=128 -v Restart=on-failure KillSignal=SIGINT SyslogIdentifier=Node-RED User=pi Environment="NODE_OPTIONS=--max-old-space-size=128" [Install] WantedBy=multi-user.target

Se guarda y cierra el archivo (Ctrl + X, Y, Enter).

- Se habilita y arranca el servicio Node-RED:

sudo systemctl enable nodered.service sudo systemctl start nodered.service

A continuación, se observa en la imagen Node Red en Raspberry Pi:

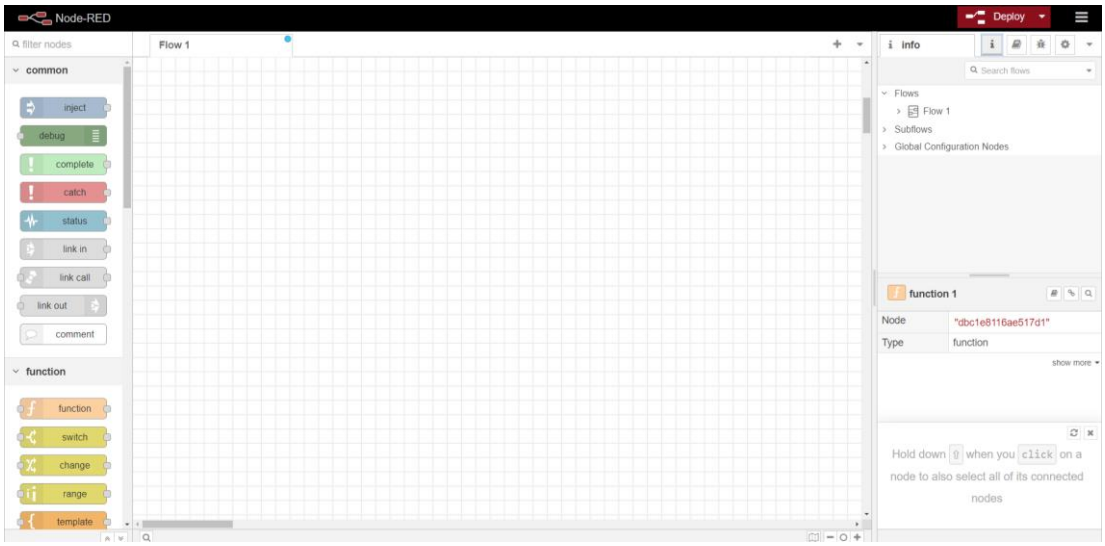


Figura 23 Node Red en Raspberry Pi.

3.5 Configuración de base de datos y Programación de servidor para recepción de datos vía API ´REST desde Blynk y envío de datos a base de datos SQL con Node-red

Se configura el servidor de PHPMyAdmin que se obtuvo en línea con la compra de un dominio web de la siguiente manera:

- Iniciar Sesión en PHPMyAdmin:

El usuario abre un navegador web y accede a la interfaz de PHPMyAdmin.

Se inicia sesión con las credenciales de administrador proporcionadas.

- Crear la Tabla:

En la interfaz de PHPMyAdmin, se selecciona la base de datos en la que se desea crear la tabla esp.

Se accede a la pestaña "SQL" en la parte superior para abrir el editor de consultas SQL.

En el editor de consultas SQL, se pega el siguiente código para crear la tabla esp:

CREATE TABLE `esp` (`id` int(255) NOT NULL, `temperatura` varchar(255) NOT NULL, `humedad` varchar(255) NOT NULL, `distancia` varchar(255) NOT NULL, `mq` varchar(255) NOT NULL, `luz` varchar(255) NOT NULL, `hora` time(6) NOT

NULL, `fecha` date NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

- Definir la Clave Primaria:

Para definir la clave primaria en el campo id, se pega el siguiente código en el editor de consultas SQL y se ejecuta:

ALTER TABLE `esp` ADD PRIMARY KEY (`id`);

- Hacer que el Campo id sea Autoincremental:

Para hacer que el campo id sea autoincremental, se pega el siguiente código en el editor de consultas SQL y se ejecuta:

ALTER TABLE `esp` MODIFY `id` int(255) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21;

A continuación, se observa en la imagen sobre las tablas esp para guardado de datos de sensores:

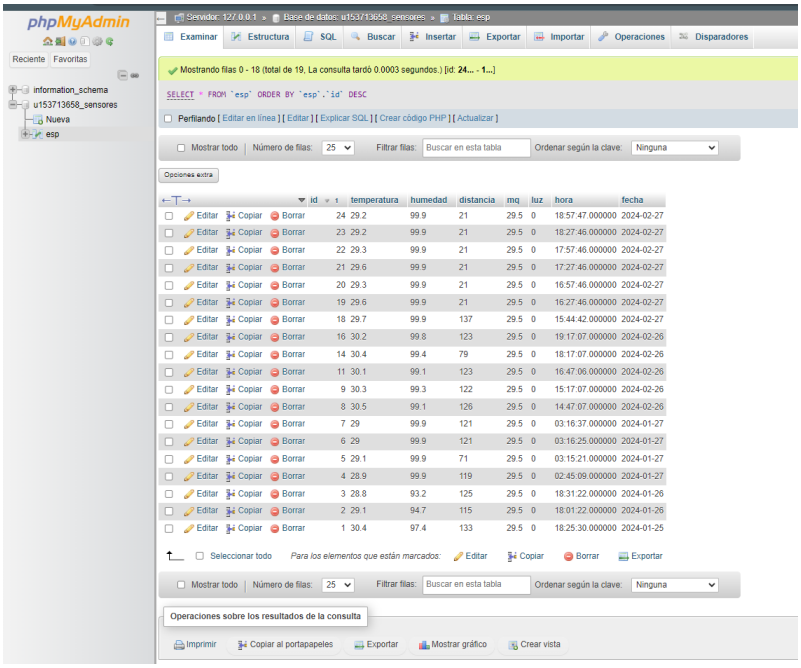


Figura 24: Tabla esp para guardado de datos de sensores

Se programa el servidor Node-Red para recepción de datos desde Blynk IoT con API REST para el almacenamiento de datos usando los siguientes bloques:

- Bloque de Inyección (inject):

Este bloque se configura para inyectar un mensaje cada cierto intervalo de tiempo (en este caso, cada 1800 segundos o 30 minutos). Este mensaje contiene un valor booleano true, que activa el flujo y desencadena la siguiente acción.

- Bloque de Solicitud HTTP (http request):

Este bloque realiza una solicitud GET a una URL específica que representa una API externa. La URL contiene un token de autenticación y una lista de pines virtuales (v0, v1, v2, v3, v4) de los cuales se obtienen los datos.

- Bloque de Función (function):

Este bloque ejecuta una función personalizada en JavaScript que procesa los datos obtenidos de la API. Extrae los valores de temperatura, humedad, calidad del aire, cantidad de luz y distancia de la respuesta de la API. Luego, formatea estos valores junto con la fecha y la hora actuales en una consulta SQL para insertar los datos en una tabla específica de la base de datos.

- Bloque Debug (debug):

Este bloque se utiliza para visualizar los mensajes de depuración. Muestra el contenido del mensaje en la consola de depuración de Node-RED.

- Bloque MySQL (mysql):

Este bloque se encarga de realizar la inserción de los datos en la base de datos MySQL. Utiliza los parámetros de conexión proporcionados para acceder a la base de datos y ejecutar la consulta SQL generada por el bloque de función.

- Configuración de la Base de Datos MySQL (MySQLdatabase):

Este bloque define los parámetros de conexión a la base de datos MySQL, como la dirección del servidor, el puerto, el nombre de la base de datos, la zona horaria y el juego de caracteres.

A continuación, se observa en la imagen sobre la programación Node-Red:

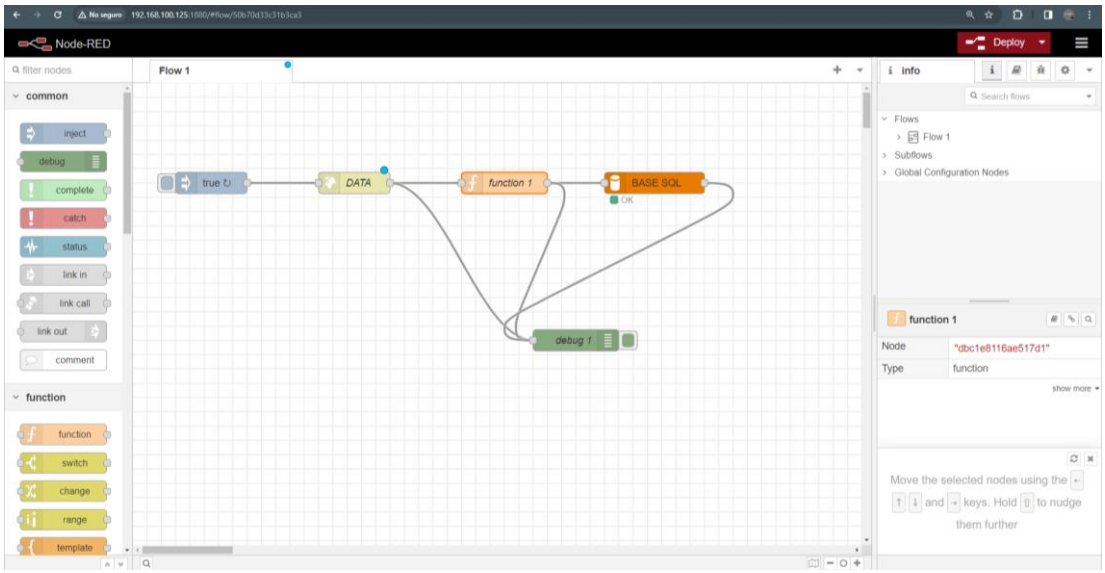


Figura 25: Programación Node-Red

A continuación, se observa en la imagen sobre los resultados de Servidor Node-Red:

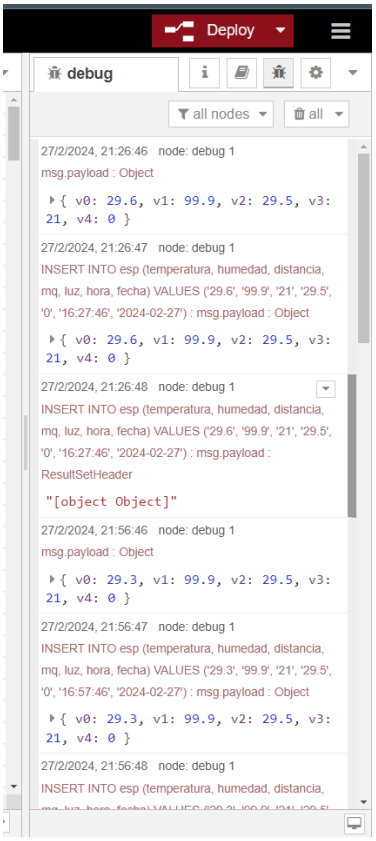


Figura 26: Resultados de Servidor Node-Red

3.6 Configuración de Visual Code en Raspberry

Para realizar el servidor de página web se requiere instalar Visual Code en Raspberry Pi, al ser un sistema basado en Linux se facilita el proceso siguiendo los siguientes pasos:

1. Instalación de Visual Studio Code:

Se instala Visual Studio Code en Raspberry Pi siguiendo las instrucciones proporcionadas por el sitio web oficial de Visual Studio Code o mediante la línea de comandos.

2. Configuración de Git en Raspberry Pi:

Se instala Git en Raspberry Pi utilizando el administrador de paquetes de su sistema operativo.

Luego, se configura Git con nombre de usuario y dirección de correo electrónico (Correo electrónico usado para el proyecto), utilizando los siguientes comandos en la terminal:

git config --global user.name "Usuario"

git config --global user.email "correo@email.com"

3. Creación de un Repositorio en GitHub:

Se accede a GitHub desde un navegador web y se crea un nuevo repositorio, siguiendo las instrucciones proporcionadas por GitHub.

Se obtiene la URL del repositorio recién creado, que se utilizará para vincular el repositorio local con el remoto en GitHub.

4. Clonación del Repositorio en Visual Studio Code:

Se abre Visual Studio Code en Raspberry Pi.

Utilizando el comando "Ctrl + Shift + P" (o "Cmd + Shift + P" en macOS) para abrir la paleta de comandos.

Se selecciona la opción "Git: Clone" e ingresa la URL del repositorio de GitHub.

Se elige una ubicación local para clonar el repositorio.

5. Configuración de Control de Versiones en Visual Studio Code:

El usuario se vincula con Visual Studio Code con Git seleccionando la opción "Source Control" en la barra lateral.

Se selecciona "Git" como el sistema de control de versiones.

Visual Studio Code detecta automáticamente los cambios en los archivos del proyecto y los muestra en el área de control de versiones.

6. Commit y Push de Cambios:

Se realiza los cambios necesarios en el proyecto utilizando Visual Studio Code.

Luego, se utiliza la función "Commit" en Visual Studio Code para confirmar los cambios localmente, proporcionando un mensaje descriptivo para cada commit.

Finalmente, se utiliza la función "Push" para enviar los cambios confirmados al repositorio remoto en GitHub.

A continuación, se observa en la imagen Visual Code:

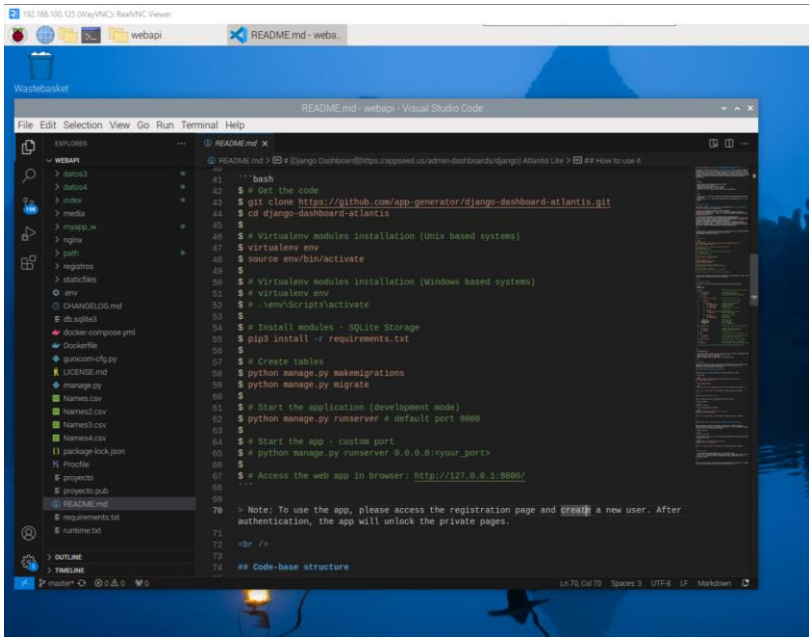


Figura 27: Visual Code

3.7 Instalación y puesta en marcha de página web Django en Raspberry Pi

Para inicializar el proyecto Django en Visual Code, se abre un terminal y se siguen los siguientes pasos:

Descarga de la plantilla: Primero, se descarga una plantilla de Internet y se guarda en un directorio local en tu sistema. Esta plantilla se encuentra en repositorios descargables de Google, puede ser cualquier plantilla, en este caso se descargo una plantilla de Django Atlantis Master que contiene elementos básicos en vistas HTML y CSS, también posee una autenticación y registro básico a la página web.

Instalación de Django: Se instala Django utilizando pip, el gestor de paquetes de Python. Para ello, se ejecuta el siguiente comando en la terminal:

pip install django

Instalación de dependencias: Una vez descargada la plantilla, se verifica si contiene un archivo llamado requirements.txt. Este archivo generalmente incluye una lista de dependencias de Python necesarias para ejecutar la plantilla. Para instalar estas dependencias, se ejecuta el siguiente comando en la terminal:

pip3 install -r requirements.txt

Esto instalará todas las dependencias necesarias para ejecutar la plantilla en el entorno local del usuario.

Aplicación de migraciones: Después de instalar las dependencias, es posible que sea necesario aplicar migraciones de base de datos para configurar la base de datos de acuerdo con los requisitos de la plantilla. Para ello, se ejecuta el siguiente comando en la terminal:

python manage.py migrate

Este comando aplicará todas las migraciones pendientes y configurará la base de datos de acuerdo con las especificaciones de la plantilla.

Inicio del servidor de desarrollo: Una vez que se han instalado las dependencias y se han aplicado las migraciones, el usuario está listo para iniciar el servidor de

desarrollo y ejecutar la plantilla. Esto se logra ejecutando el siguiente comando en la terminal:

python manage.py runserver

Este comando iniciará el servidor de desarrollo y hará que la plantilla esté disponible en el navegador web local del usuario mediante la URL http://localhost:8000/.

Creación de una aplicación: Si se desea agregar funcionalidades específicas al proyecto, se puede crear una aplicación Django adicional. Para ello, se utiliza el siguiente comando en la terminal:

python manage.py startapp nombre_de_la_aplicacion

Esto generará un nuevo directorio con el nombre de la aplicación especificada, junto con la estructura inicial de archivos necesarios para una aplicación Django.

Para este proyecto se crearon las siguientes aplicaciones:

- Index: Contiene la lógica de la pagina principal index.html, en esta vista se procesan los datos SQL que se visualizaran en las tablas de registros históricos de los sensores.
- Registros: crea la lógica para realizar la consulta SQL de todos los datos de registros históricos en para mostrarlos en la página tables-simple.html
- Datos: contiene la clase para la función de creación de reporte en Excel, donde selecciona los datos de los registros históricos de los sensores y crea un Excel descargable para los usuarios.
- Api: es la carpeta donde se realiza una función para leer la API blynk en tiempo real, en la vista HTML se realiza una solicitud AJAX que permite actualizar y consultar URL sin tener que refrescar la página

Vistas utilizadas:

- Index.html: Contiene la vista del dashboard en tiempo real en HTML, también permite desarrollar lógica en Java Script que permite la creación de solicitud AJAX para actualizar valores de lectores analógicos en tiempo real, también contiene la lógica para rellenar gráficas de los datos al momento de cargar la página.

- Tables-simple.html: recarga la página web con los registros históricos que se obtuvo desde el la vista de Python que permite rellenar una tabla con los datos SQL del proyecto usando HTML y Java Script.

A continuación, se observa en la imagen sobre la página de Dashboard en tiempo real:

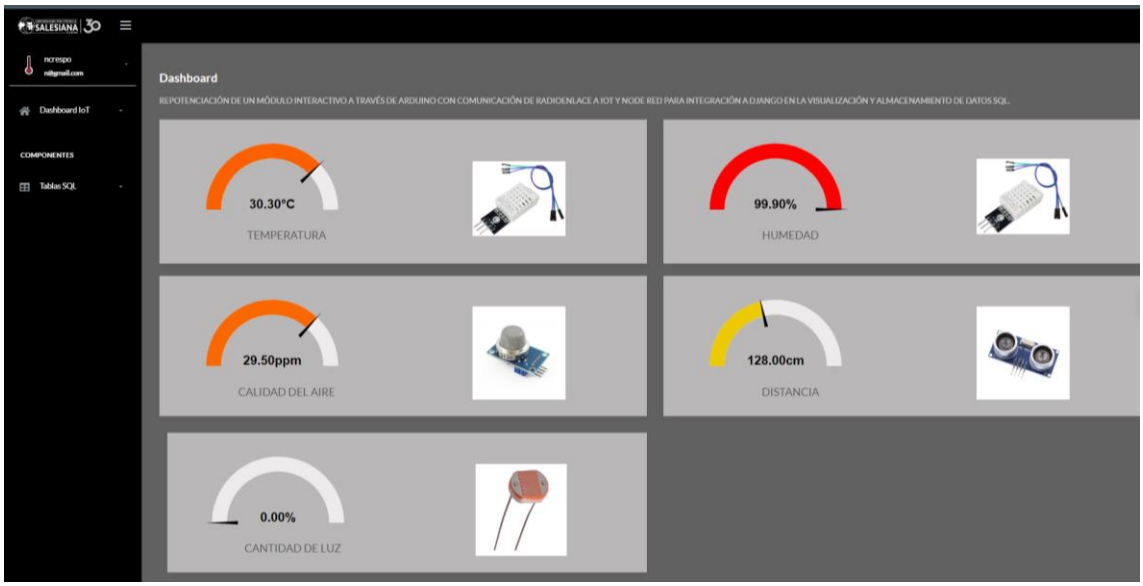


Figura 28: Página de Dashboard en tiempo real

A continuación, se observa en la imagen sobre la gráfica de Temperatura:

Figura 29: Grafica de Temperatura

A continuación, se observa en la imagen sobre las tablas de registros SQL:

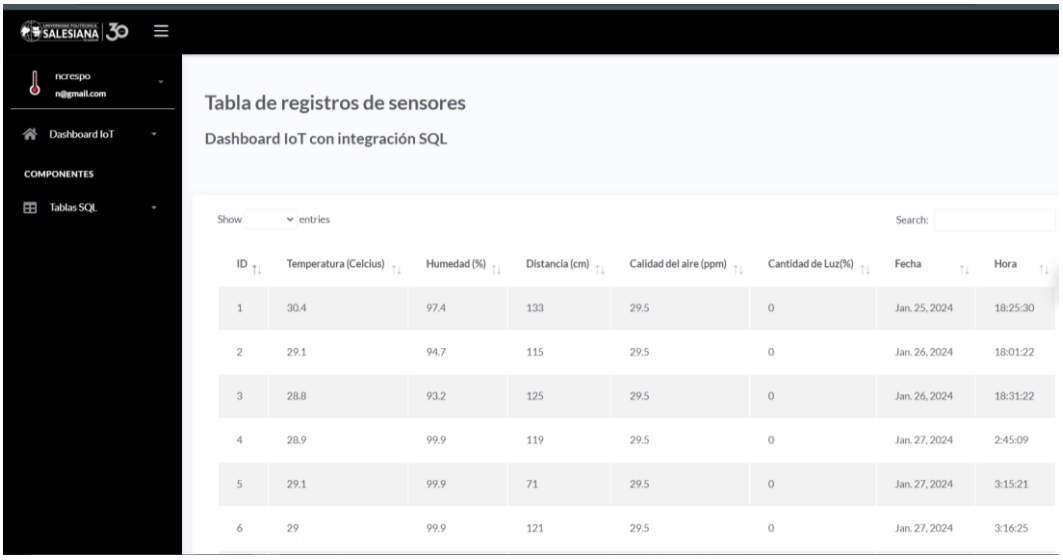


Figura 30: Tablas de registros SQL

3.8 Red de antenas para enlace a Internet.

Para el diseño de la red que provee internet al dispositivo se usan dos antenas Ubiquiti NanoStation y un repetidor extensor de rango de wifi siguiendo los siguientes pasos:

Configuración de la antena principal (Access Point): La antena principal Ubiquiti está configurada en modo Access Point y se conecta al router que proporciona Internet. Esta antena emite una red Wi-Fi llamada "acceso". Los dispositivos se conectan a esta red para acceder a Internet.

A continuación, se observa en la imagen Antena Ubiquiti Loco M5:

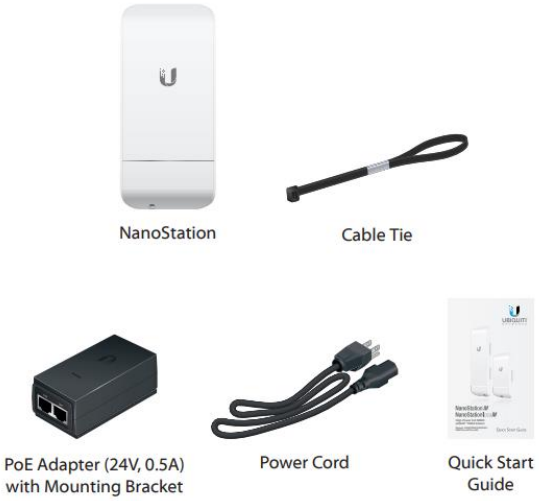


Figura 31: Antena Ubiquiti Loco M5 (Free Electron, 2020)

Configuración de la antena secundaria (Estación): La segunda antena Ubiquiti está configurada en modo Estación y se conecta a la red "acceso" emitida por la antena principal. Esta antena secundaria se configura como un router, utilizando NAT (Network Address Translation), y establece una red local con un rango de direcciones IP que va desde 192.168.100.2 hasta 192.168.100.254. Además, se activa el servidor DHCP para asignar automáticamente direcciones IP a los dispositivos que se conecten a esta red local.

Conexión al router extensor de rango TP-Link: La antena Ubiquiti secundaria se conecta a un router extensor de rango TP-Link. Este router extensor proporciona conexiones de 2.4GHz, lo que permite que dispositivos como la ESP32 y la Raspberry

Pi se conecten a la red Wi-Fi. El router extensor se conecta al dispositivo Ubiquiti a través de un cable de red.

Conexión de dispositivos a la red Wi-Fi: Los dispositivos, como la ESP32 y la Raspberry Pi, se conectan a la red Wi-Fi emitida por el router extensor de rango TP-Link. Este router extensor, a su vez, está conectado a la antena Ubiquiti secundaria, que proporciona acceso a Internet a través de la conexión principal.

A continuación, se observa en la imagen del diagrama de Red de proyecto:

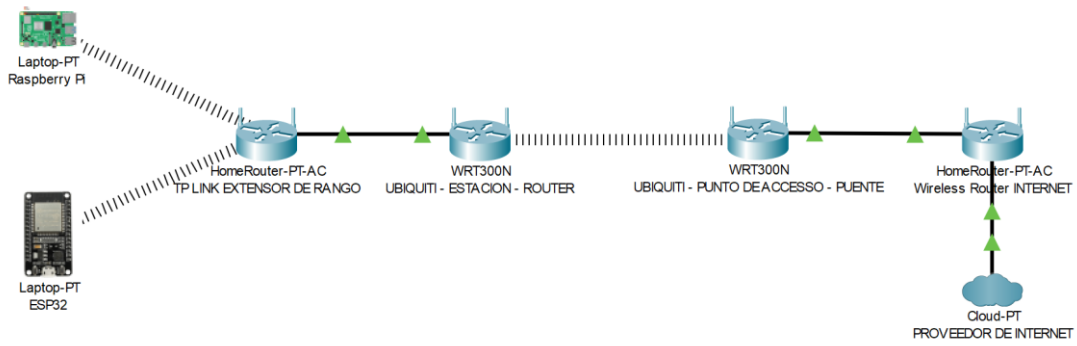


Figura 32: Diagrama de Red de proyecto

3.9 Publicación de pagina web en Digital Ocean

Teniendo la programación en la carpeta PROYECTO que se ejecuta en Visual Code en Raspberry, se realizó los siguientes pasos para subir un proyecto preparado a un repositorio de Git para que digitalOcean pueda acceder al repositorio.

3.9.1 Preparación de proyecto y Git:

- Se crea un correo Gmail para poder crear una cuenta git. Una vez creada la cuenta Git con autenticación de Google, se crea un repositorio llamado PROYECTO.

A continuación, se observa en la imagen de la creación de repositorio GIT:

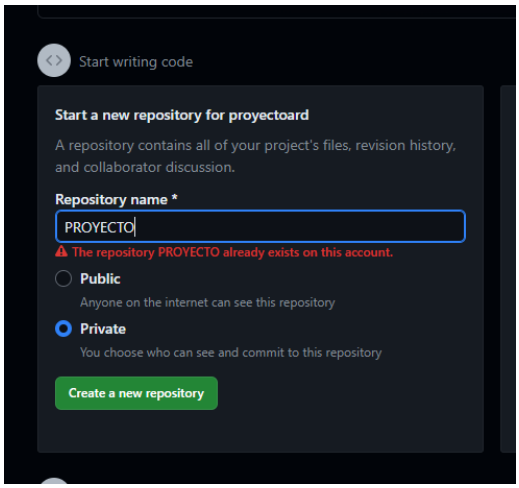


Figura 33: Creación de repositorio GIT

- El repositorio se encuentra vacío, para poder acceder al repositorio desde Visual Code en Raspberry se crea un token de acceso y se selecciona todos los permisos.

A continuación, se observa en la imagen de la creación de token de acceso en Git:

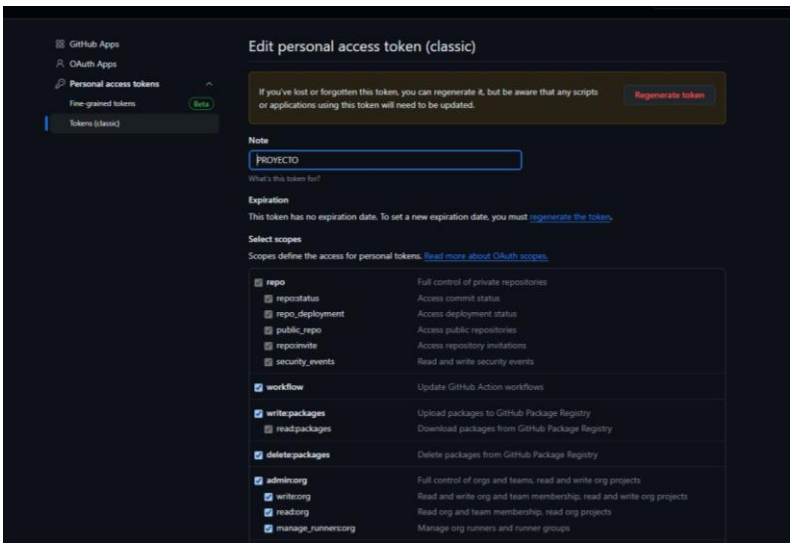


Figura 34: Creación de token de acceso en Git

- Una vez creado el Token de acceso este servirá para iniciar sesión desde visual Code siguiendo los siguientes pasos:

1. Para inicializar un repositorio Git, el usuario debe abrir una terminal y navegar hasta la carpeta del proyecto. Luego, se inicializa un repositorio Git ejecutando el siguiente comando:

   git init

   Este comando crea un nuevo repositorio Git en la carpeta del proyecto.
2. Para agregar los archivos al área de preparación, se usa el comando **git add** para agregar los archivos del proyecto al área de preparación. Por ejemplo, para agregar todos los archivos, se ejecuta:

   git add .

   Esto agregará todos los archivos y cambios nuevos al área de preparación.

3. Después de agregar los archivos al área de preparación, se realiza un commit para confirmar los cambios. Se ejecuta el siguiente comando y reemplazando "Mensaje de commit" con un mensaje descriptivo de los cambios que está realizando:

   git commit -m "Mensaje de commit"

4. Luego con el usuario GIT llamado proyectoard se agrega el repositorio remoto al proyecto local. Utilizando el siguiente comando y se reemplaza <URL_del_repositorio> con la URL del repositorio que se creó en este caso la URL es: https://github.com/proyectoard/PROYECTO.git

   git remote add origin <URL_del_repositorio>

5. Finalmente, se debe empujar los cambios locales al repositorio remoto ejecutando el siguiente comando:

   git push -u origin master

   Esto enviará todos los commits locales al repositorio remoto que se ha configurado como "origin", Visual Code pedirá el usuario y el token Personal creado para este proyecto. Con estos pasos, se habrá subido la carpeta de proyecto a un repositorio Git en un servicio de alojamiento remoto, lo que

permitirá a otros colaboradores clonar el repositorio, acceder desde DigitalOcean, trabajar en el proyecto y contribuir con cambios.

A continuación, se observa en la imagen el proyecto subido a Git.

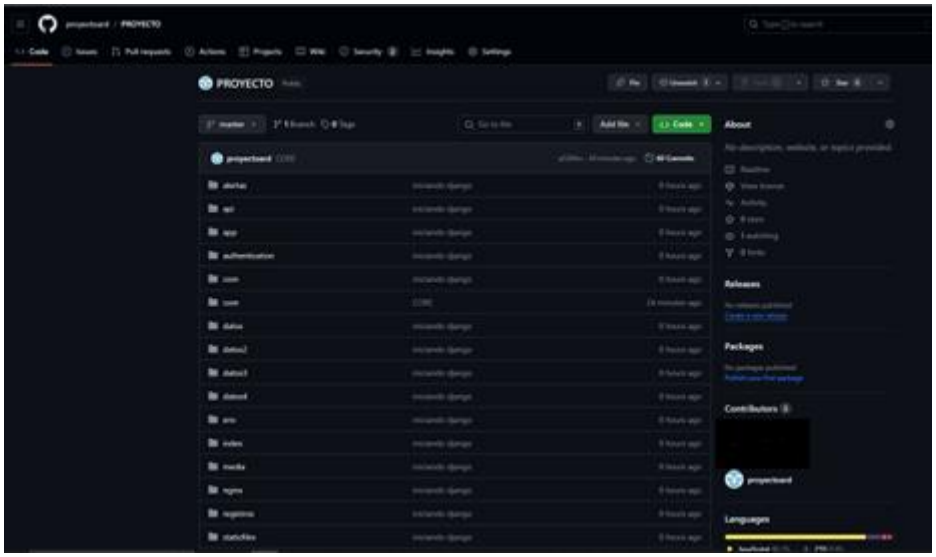


Figura 35: Proyecto subido a Git.

3.9.2 Preparación de DigitalOcean

DigitalOcean permite cargar aplicaciones Web en su servicio de alojamiento, para ello se debe preparar la plantilla Django y seguir los siguientes pasos:

1. Con la misma cuenta de correo electrónico de Git se crea una cuenta para DigitalOcean, al ingresar se debe agregar una tarjeta de banco, ya que el servicio de alojamiento cobra por el uso y las características de maquina virtual, en este caso al ingresar por primera vez se acreditan $200 para uso de prueba.
2. Se crea una app Platform que permite desplegar el código en un servicio web del alojamiento, iniciando sesión en Git se selecciona el repositorio remoto del proyecto creado.

A continuación, se observa en la imagen sobre la configuración Wireless de Antenas:

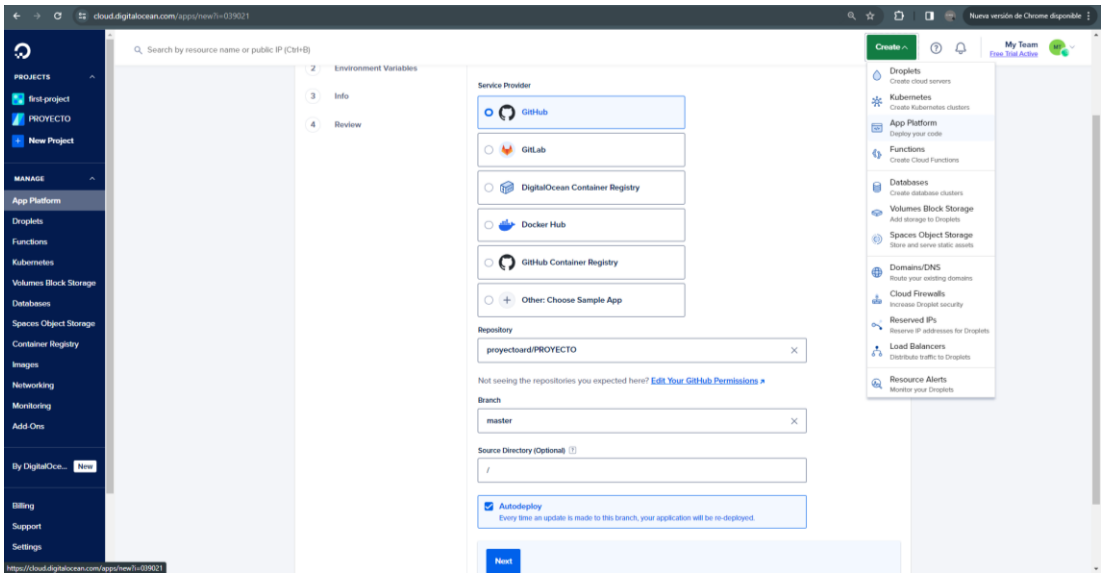


Figura 36: Configuración Wireless de Antenas

3. Se configuran el comando de despliegue (gunicorn --worker-tmp-dir /dev/shm core.wsgi este comando inicia el servidor Gunicorn y le dice que cargue la aplicación web definida en core.wsgi, utilizando /dev/shm como directorio temporal para los trabajadores del servidor), el puerto del servicio local (8000) y la ruta de la página principal donde se escucharan las solicitudes HTTP, en este caso solo se coloca (/)

A continuación, se observa en la imagen sobre la configuración de servicios Django:

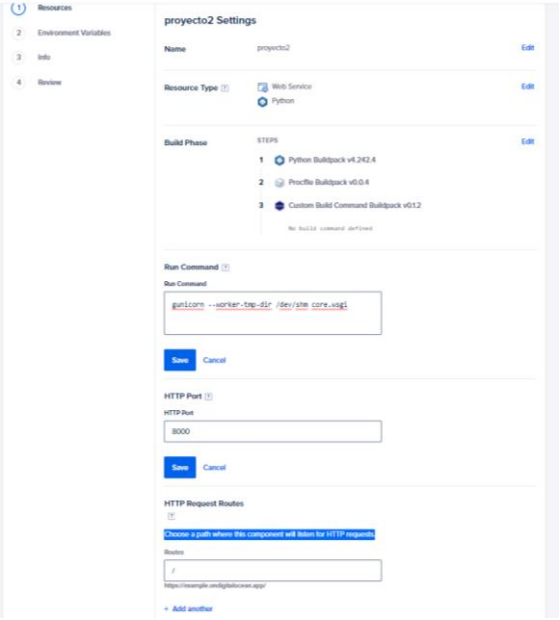


Figura 37: Configuración de servicios Django.

4. Se configura el precio de la Instancia que está definida por la RAM y la cantidad de vCPU's de los recursos de los servicios web, para este caso por ser una página web que no ocupa demasiados recursos, se define la instancia en básica y se usa los recursos de 512 Mb con 1 vCPU. A medida que crece la página web, sea en dispositivos o procesamiento de datos, se puede editar los precios y los recursos a usar.

A continuación, se observa en la imagen Selección de recursos de servicios Web:

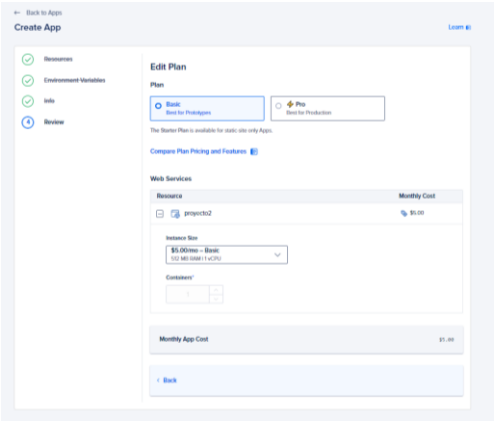


Figura 38: Selección de recursos de servicios Web

5. Una vez creado el recurso, DigitalOcean despliega la página web en una URL propio del alojamiento, para ello se agrega en el archivo core/settings.py la URL de DigitalOcean junto con la URL obtenida para el proyecto (esp32io.tech), se envían los cambios a repositorio de Git y DigitalOcean detecta los cambios pudiendo visualizar la página web en la URL de DigitalOcean.

A continuación, se observa en la imagen Configuración Settings.py:

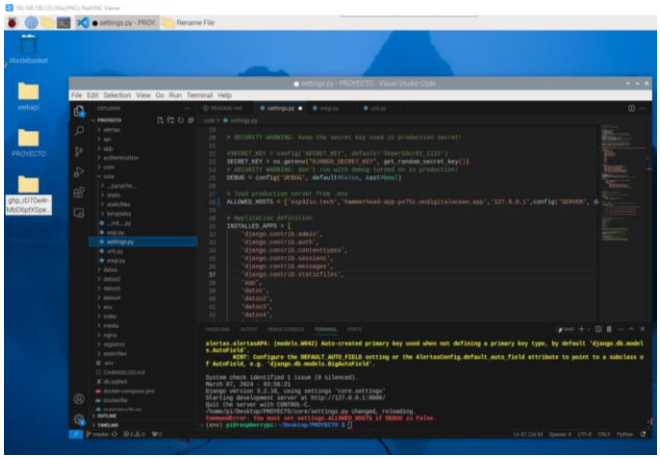


Figura 39: Configuración Settings.py

3.9.3 Configuración de Dominio para visualizar página web en dominio propio

Se debe apuntar el dominio esp32io.tech a los servidores de DigitalOcean, para ello se deben enviar los nameservers de DigitalOcean al proveedor del dominio y colocarlos en los apartados correspondientes.

A continuación, se observa en la imagen Nameservers digitalocean:

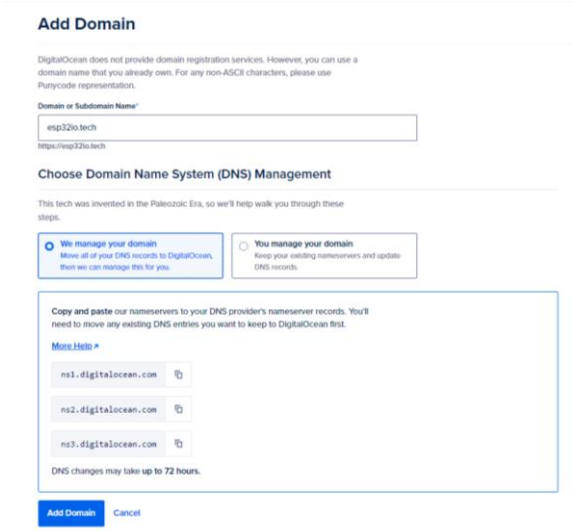


Figura 40: Nameservers DigitalOcean

Una vez realizada esta acción los servidores tardan alrededor de 72 Horas para apuntar el dominio a la página web ya configurada, una vez pasado ese periodo de tiempo se puede redirigir la página web a https:// esp32io.tech con todas las redirecciones HTTP del proyecto.

A continuación, se observa en la imagen sobre la visualización de página web en dominio:

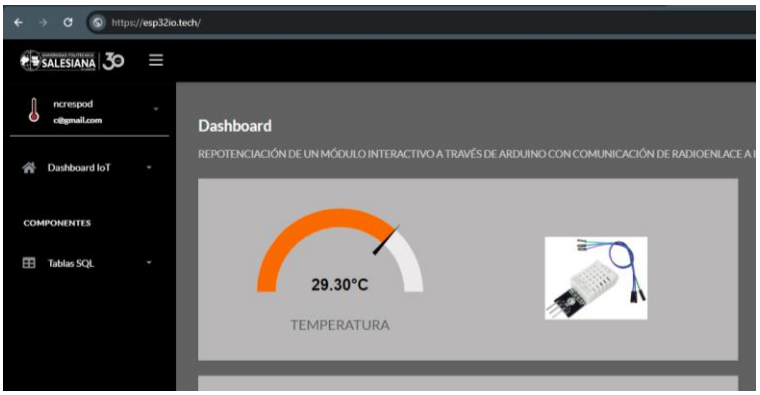


Figura 41: Visualización de página web en dominio *esp32io.tech*.

4. Análisis de resultados obtenidos

Los análisis y resultados para este proyecto sirven para evaluar la potenciación que se realizó al módulo didáctico obtenido para validación y evaluación del proyecto, optimizaciones continuas, y comunicación del proyecto.

1. Datos Recopilados y Almacenados:

- Se tienen datos recopilados de los sensores de temperatura, humedad, calidad de aire, cantidad de luz, y posiblemente otros parámetros ambientales relevantes con fecha y hora de ingreso a base.
- Estos datos se han almacenado de manera efectiva en una base de datos en la nube utilizando PHPMyAdmin, lo que permite un acceso seguro y centralizado a los mismos desde cualquier lugar con conexión a internet.

A continuación, se observa en la imagen sobre la visualización de registros de datos en página web:

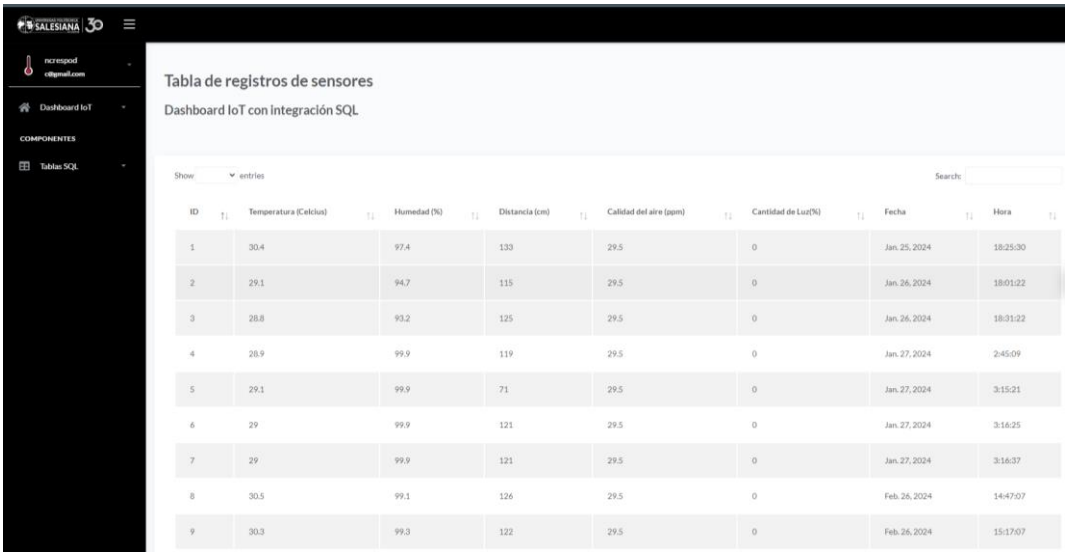


Figura 42: Visualización de registros de datos en página web

2. Dashboard Interactivo en Tiempo Real:

- Se ha desarrollado un dashboard interactivo en tiempo real utilizando Blynk IoT, que permite a los usuarios visualizar y monitorear los datos de los sensores

en tiempo real desde sus dispositivos móviles o computadoras además de aprovechar la comunicación mediante API REST que la plataforma proporciona para integraciones a otras aplicaciones.

A continuación, se observa en la imagen Visualización en Blynk IoT:

Figura 43: Visualización en Blynk IoT

3. Procesamiento y Análisis de Datos:

- Los datos recopilados se han procesado y analizado utilizando Node-RED en una Raspberry Pi, lo que permite realizar acciones como el almacenamiento periódico de datos que en el programa se dio cada 30 minutos, y la integración con otros sistemas y servicios como la página web en Django.

A continuación, se observa en la imagen Node Red Server:

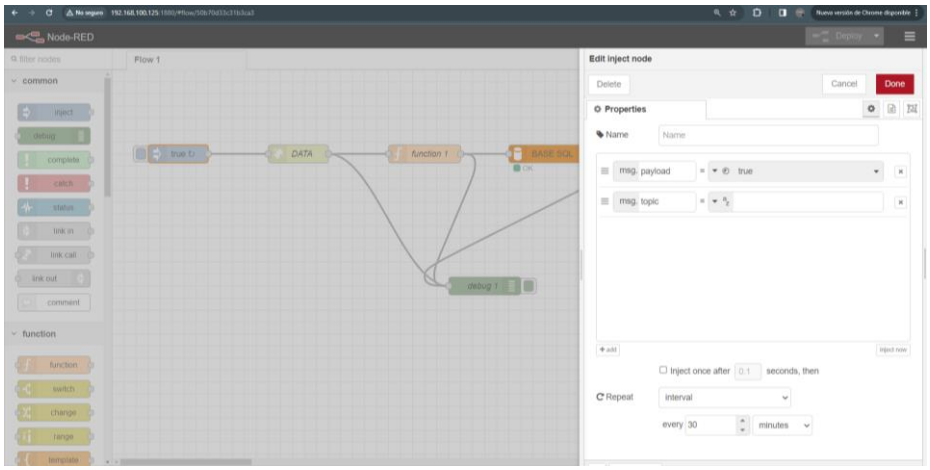


Figura 44: Node Red Server

4. Visualización Avanzada en una Aplicación Web:

- Se ha desarrollado una aplicación web utilizando Django, que proporciona una visualización avanzada de los datos recopilados, incluyendo gráficos interactivos, tablas de datos, y un dashboard programado en tiempo real.
- Esta aplicación web se ha desplegado en un servidor de DigitalOcean y se ha enlazado a un dominio web propio, lo que permite un acceso global a la misma.

A continuación, se observa en la imagen Dominio enlazado en Página Web:

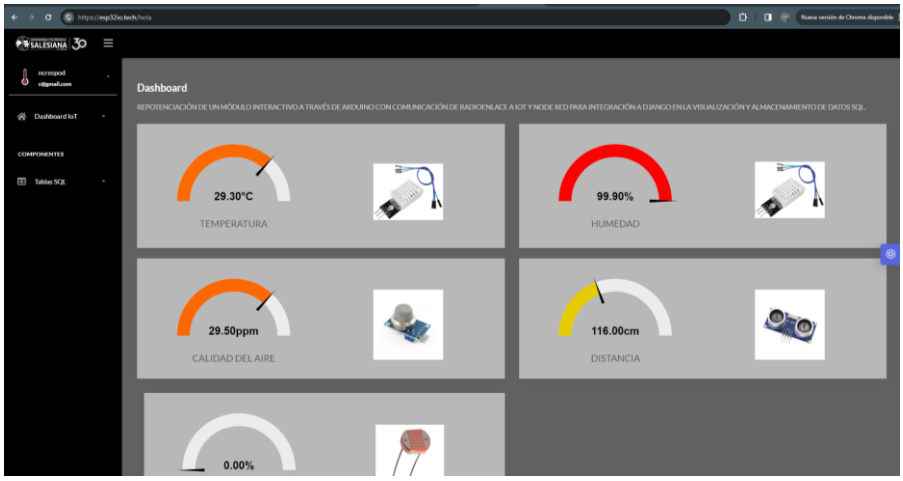


Figura 45: Dominio enlazado en Página Web

A continuación, se observa en la imagen las gráficas obtenidas de los datos almacenados:

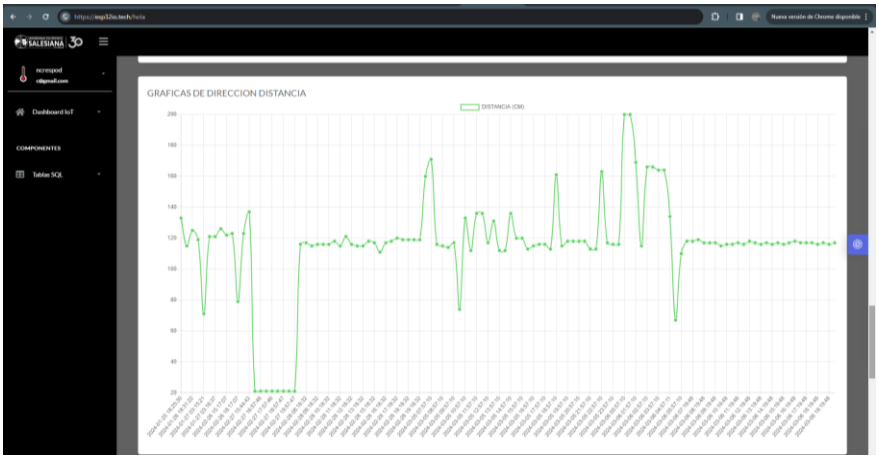
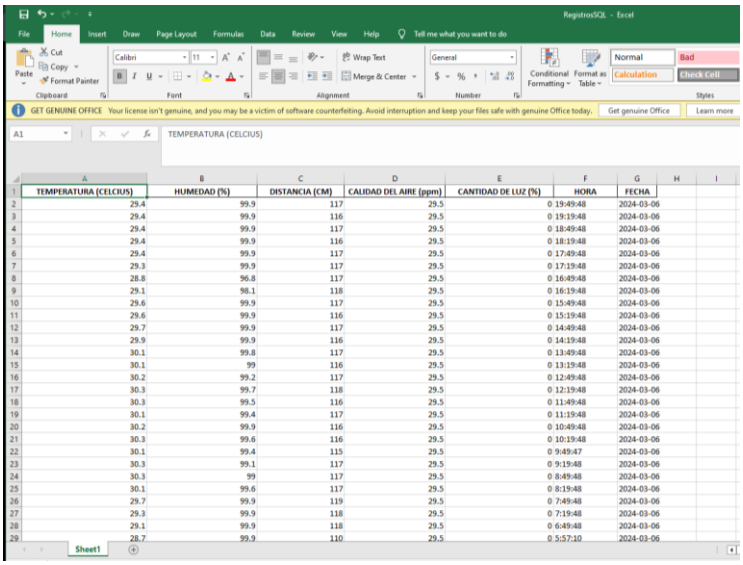


Figura 46: Gráficas obtenidas de los datos almacenados (Sensor vs Tiempo).

5. Descarga de Datos y Generación de Informes:

- Los usuarios tienen la capacidad de descargar los datos recopilados en formato Excel desde la aplicación web, lo que facilita el análisis posterior y la generación de informes personalizados.

A continuación, se observa en la imagen el reporte en Excel con registros Históricos:



TEMPERATURA (CELCIUS)	HUMEDAD (%)	DISTANCIA (CM)	CALIDAD DEL AIRE (ppm)	CANTIDAD DE LUZ (%)	HORA	FECHA
29.4	99.9	117	29.5	0	19:49:48	2024-03-06
29.4	99.9	116	29.5	0	19:19:48	2024-03-06
29.4	99.9	117	29.5	0	18:49:48	2024-03-06
29.4	99.9	116	29.5	0	18:19:48	2024-03-06
29.4	99.9	117	29.5	0	17:49:48	2024-03-06
29.3	99.9	117	29.5	0	17:19:48	2024-03-06
28.8	96.8	117	29.5	0	16:49:48	2024-03-06
29.1	98.1	118	29.5	0	16:19:48	2024-03-06
29.6	99.9	117	29.5	0	15:49:48	2024-03-06
29.6	99.9	116	29.5	0	15:19:48	2024-03-06
29.7	99.9	117	29.5	0	14:49:48	2024-03-06
29.9	99.9	116	29.5	0	14:19:48	2024-03-06
30.1	99.8	117	29.5	0	13:49:48	2024-03-06
30.1	99	116	29.5	0	13:19:48	2024-03-06
30.2	99.2	117	29.5	0	12:49:48	2024-03-06
30.3	99.7	118	29.5	0	12:19:48	2024-03-06
30.3	99.5	116	29.5	0	11:49:48	2024-03-06
30.1	99.4	117	29.5	0	11:19:48	2024-03-06
30.2	99.9	116	29.5	0	10:49:48	2024-03-06
30.3	99.6	116	29.5	0	10:19:48	2024-03-06
30.1	99.4	115	29.5	0	9:49:47	2024-03-06
30.3	99.1	117	29.5	0	9:19:48	2024-03-06
30.3	99	117	29.5	0	8:49:48	2024-03-06
30.1	99.6	117	29.5	0	8:19:48	2024-03-06
29.7	99.9	119	29.5	0	7:49:48	2024-03-06
29.3	99.9	118	29.5	0	7:19:48	2024-03-06
29.1	99.9	118	29.5	0	6:49:48	2024-03-06
28.7	99.9	110	29.5	0	5:57:10	2024-03-06

Figura 47: Reporte en Excel con registros Históricos

6. Diseño de red de radioenlace:

- La red de radioenlace ha mostrado proporcionar una conectividad confiable a Internet, lo que permite a los dispositivos del proyecto acceder a recursos en línea y servicios basados en la nube de manera efectiva.
- Se ha observado una estabilidad en la conexión a Internet a través del radioenlace, con un mínimo de interrupciones o caídas de conexión. Esto es fundamental para garantizar un acceso continuo a los recursos en línea y mantener la productividad del proyecto.
- El router integrado ha facilitado la creación de una red privada para los equipos del proyecto, lo que proporciona una capa adicional de seguridad y control sobre el acceso a los recursos y datos compartidos en la red.
- El router ha permitido una gestión eficiente de los dispositivos conectados a la red privada, facilitando la asignación de direcciones IP, la configuración de reglas de firewall y la supervisión del tráfico de red.

A continuación, se observa en la imagen **l**as pruebas que se realizó con los equipos conectados a la red:

Figura 48: Prueba de equipos conectados a la red

A continuación, se observa en la imagen las prueba que se realizó en la conexión de Antenas Ubiquiti:

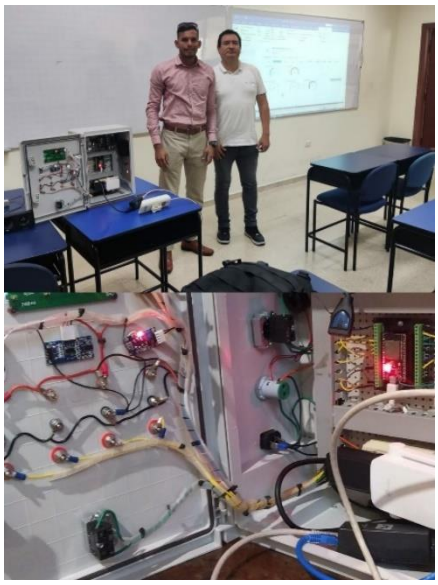


Figura 49: Prueba de conexión de Antenas Ubiquiti

4.1 Descripción de los resultados

El proyecto que utiliza un ESP32 para leer datos de varios sensores y luego procesa y almacena esos datos para su visualización y análisis presenta varios aspectos importantes que pueden ser analizados y documentados.

El uso del ESP32 como microcontrolador para la adquisición de datos de sensores de temperatura, humedad, calidad de aire y cantidad de luz demuestra una elección eficiente y económica para la recolección de datos en tiempo real.

La comunicación y diseño de la red de radioenlace permite una comunicación de red privada con uno o varios dispositivos, además de centralizar un único servidor como la Raspberry Pi en una única ubicación, los equipos pueden estar ubicados en varios puntos estratégicos aumentando el alcance gracias a las antenas, esto permite que el nivel del proyecto sea elevado para ambientes industriales o de campo donde las distancias pueden ser largas y la comunicación de equipos sea complicada.

La implementación de un dashboard en tiempo real a través de Blynk IoT ofrece una interfaz de usuario dinámica y de fácil uso para visualizar los datos recopilados por los sensores.

La incorporación de Node-RED en una Raspberry Pi para utilizar la API REST de Blynk proporciona una solución escalable para recopilar y procesar datos a intervalos regulares (cada 30 minutos), ampliando las capacidades de adquisición y almacenamiento de datos del proyecto que pueden ser configurables.

La elección de utilizar una base de datos en la nube gestionada a través de PHPMyAdmin permite un almacenamiento seguro y centralizado de los datos recopilados, facilitando su acceso y gestión desde cualquier lugar con conexión a internet.

La creación de una aplicación web con Django para visualizar los datos recopilados agrega un nivel adicional de análisis y presentación de datos. El dashboard programado en tiempo real y la capacidad de visualizar gráficos y descargar datos en formato Excel proporcionan una experiencia de usuario completa y enriquecida además de aprovechar las bondades del lenguaje de programación Python para desarrollo de la página web.

El proyecto permite a los usuarios analizar y tomar decisiones basadas en datos en tiempo real, lo que puede ser especialmente útil en ambientes como agricultura, monitorización ambiental, control de la calidad del aire, entre otros.

El proyecto demuestra una implementación efectiva de IoT y análisis de datos, combinando hardware (ESP32), servicios en la nube (Blynk, PHPMyAdmin) y desarrollo de software (Node-RED, Django) para proporcionar una solución integral y escalable para la recopilación, procesamiento y visualización de datos en tiempo real aprovechando las bondades de una red privada de internet proporcionada por el diseño de una red de radioenlace.

Además, se destaca el despliegue exitoso de la aplicación web en un servidor de DigitalOcean ya que es una alternativa de bajo costo para la aplicación de visualización de datos y gráficas además de ser compatible con Linux, Django y repositorios Git para mantenimiento y ejecución de la página web, lo que amplía aún más la accesibilidad y utilidad del proyecto.

A continuación, se observa en la imagen el módulo de sensores repotenciado:

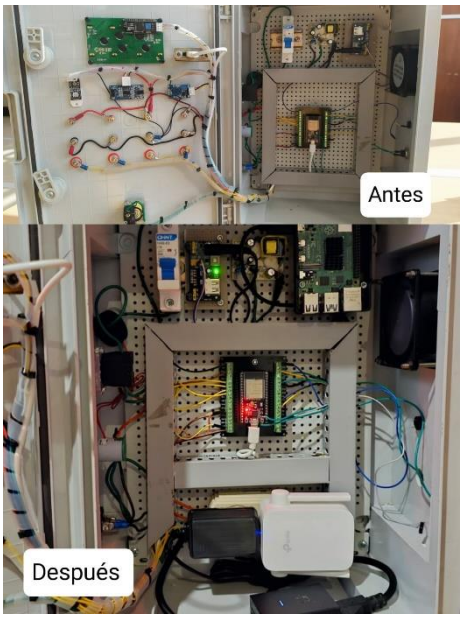


Figura 50: Módulo de sensores repotenciado

CONCLUSIONES

Se destaca la importancia y eficacia de la integración de tecnologías IoT para análisis, monitorización y gestión de variables de sensores a través de dispositivos y herramientas de software de bajo costo para el uso de la escalabilidad y desarrollo de un proyecto que pueda tener varias estaciones de monitoreo.

Se mostró la viabilidad de usar las herramientas tanto de software como hardware para recopilar datos ya que solo el monitoreo no permite un análisis a futuro de los datos obtenidos.

Los datos se proporcionan directamente en la nube para su acceso remoto, también con la gestión de los datos desde página web desarrollada en Django permitió personalizar el proyecto.

El despliegue en DigitalOcean permitió tener un acercamiento mas profesional en ambientes de producción para software a nivel industrial, esta combinación de

tecnologías y prácticas garantiza la fiabilidad, escalabilidad y seguridad necesaria para implementaciones en ambientes industriales exigentes.

La implementación de una red de radioenlace para compartir internet, permitió tener seguridad en la red mediante contraseñas seguras y una red privada, lo que permite la escalabilidad y comunicación con varios equipos de ser necesario.

RECOMENDACIONES

- Seguridad de Datos: Es importante implementar prácticas de seguridad robustas en todas las etapas del proyecto, desde la adquisición de datos hasta el almacenamiento y la visualización. Utilizar métodos de cifrado y autenticación en Node-Red, proteger la red con Firewall, contraseñas y mantenimientos a bases de datos, puede proteger los datos sensibles y prevenir accesos no autorizados.

- Optimización del Código: Se recomienda revisar y optimizar el código en todas las partes del proyecto para garantizar un rendimiento óptimo y eficiencia en el uso de recursos. Esto incluye la optimización de consultas SQL, la reducción de la complejidad del código y la eliminación de redundancias.

- Monitoreo y Mantenimiento: Implementar herramientas de monitoreo y alerta puede ser útil para supervisar el estado del sistema en tiempo real y detectar posibles problemas o fallos de manera proactiva. Realizar mantenimiento regular del hardware y software es fundamental para asegurar un funcionamiento continuo y sin problemas.

- Escalabilidad: Diseñar el sistema con la capacidad de escalar verticalmente - horizontalmente según sea necesario para manejar aumentos en la carga de trabajo o la cantidad de datos es esencial. Esto puede implicar la adición de recursos de hardware o la optimización de la arquitectura del sistema.

- Documentación Completa: Es importante documentar detalladamente cada aspecto del proyecto, incluyendo la arquitectura del sistema, los procedimientos de configuración, las decisiones de diseño y cualquier otra información relevante. Esto facilitará la comprensión y el mantenimiento del proyecto para futuras actualizaciones o modificaciones.

- Pruebas Exhaustivas: Realizar pruebas exhaustivas en todas las etapas del proyecto es clave para garantizar su funcionamiento correcto y la precisión de los datos recopilados y procesados. Esto incluye pruebas de integración, pruebas de carga y pruebas de estrés para validar la estabilidad y rendimiento del sistema.

- Optimización de la red dada por radioenlace: Colocar las antenas en ubicaciones estratégicas que minimicen la interferencia y maximicen la señal. Esto puede implicar la instalación de antenas en lugares elevados y libres de obstáculos para mejorar la línea de visión y reducir la atenuación de la señal.

BIBLIOGRAFIA

Air Max. (2020). Carresel. Obtenido de https://www.carrecel.com/tienda/antenas-ubiquiti/antena-ubiquiti-nanostation-loco-m5-13dbi-internacional/

Boreal. (2024). Boreal. Obtenido de https://borealtech.com/que-es-el-internet-de-las-cosas-iot/

Bricogeek. (2022). Obtenido de https://tienda.bricogeek.com/sensores-temperatura/510-sensor-ds18b20-estanco.html#:~:text=Caracter%C3%ADsticas%20del%20sensor%20DS18B20%3A,9%20a%2012%20bits%20(configurable)&text=Multiples%20sensores%20puede%20compartir%20el,C%20a%20%2B85%C2%B0C)

Brito, I. (2023). DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO INTERACTIVO IOT USANDO. Guayaquil: Universidad Politécnica Salesiana.

Django. (2024). https://www.djangoproject.com/. Obtenido de https://www.djangoproject.com/

Espressif Systems (Shanghai) Co., L. (2016). ESP32-DevKitC V4 with ESP32-WROOM-32 module soldered. Obtenido de Espressif: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html

Free Electron. (2020). Obtenido de https://www.free-electron.com.ar/modems-routers-ap-antenas/2300-ubiquiti-nanostation-loco-m5-5ghz-13dbi-ap-router-cliente.html

Geekfactory. (29 de Mayo de 2017). Geekfactory. Obtenido de https://www.geekfactory.mx/tutoriales/tutoriales-arduino/lcd-16x2-por-i2c-con-arduino/

Gines, E. (19 de Julio de 2019). Obtenido de https://aprendiendoarduino.wordpress.com/2018/04/14/sensores-arduino-3/

https://docs.digitalocean.com/. (2024). DigitalOcean. Obtenido de https://docs.digitalocean.com/

HUAWEI. (2023). Protocolo MQTT. Obtenido de https://forum.huawei.com/

IONOS. (2020). IONOS. Obtenido de https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/

phpMyAdmin. (2024). Obtenido de https://www.phpmyadmin.net/

Pi, R. (2022). Raspberry Pi. Obtenido de https://www.raspberrypi.com/

Portilla, L. (2015). Obtenido de https://www.unipamplona.edu.co/unipamplona/portalIG/home_74/recursos/visual-basic-para-excel/17052017/u5_fotoresistencia.jsp#:~:text=Una%20fotorresistencia%20es%20un%20componente,en%20ingl%C3%A9s%20light%2Ddependent%20resistor.