UNIVERSIDAD
DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento

Lenguajes y Ciencias de la Computación

Área de Conocimiento

Lenguajes y Sistemas Informáticos

TRABAJO FIN DE GRADO

CONSTRUCCIÓN DE UNA HABITACIÓN INTELIGENTE PARA SU USO CON GEMELOS DIGITALES

Grado en

Ingeniería de Organización Industrial

Autor: SAÚL JURADO SÁNCHEZ

Tutor: LOLA BURGUEÑO CABALLERO

Cotutor: PAULA MUÑOZ ARIZA

MÁLAGA, Septiembre de 2.023

Agradecimientos

A mi madre, por ser una luchadora y
creer siempre en mí, por ti soy la persona
que soy, gracias por nunca dejarme caer.

A mis amigos, por escucharme siempre y
tenderme la mano siempre que lo he necesitado.

A mi familia, que siempre ha estado apoyándome
y haciéndolo todo un poco más ameno.

A mis dos ángeles que están en el cielo cuidándome y
guiándome. Papá, abuelo, lo he conseguido.

Abstract

The Internet of Things (IoT) has radically transformed our interaction with the current world, enabling the interconnection and data collection from a variety of devices. From home lighting to energy management, IoT has become essential in our daily lives, opening up new opportunities in sectors such as healthcare, transportation, and industry. Microcontrollers like ESP-8266, ESP-32, ESP-05, and ESP-01 play a crucial role in enabling data collection and device control through sensors and actuators, efficiently programmed in languages like C++.

This thesis focuses on the comprehensive enhancement of energy efficiency and home security. The main objective is to drive innovation and technological integration to transform and enhance the intelligence and efficiency of domestic environments. In this regard, the implementation of solutions harnessing the full potential of the Internet of Things (IoT) is contemplated.

As a result, a comprehensive system has been successfully implemented that integrates various technologies, enabling advanced home automation control.

Keywords: Internet of Things, ESP-8266, Home Automation, Sensors, Microcontrollers.

Resumen

El Internet de las Cosas (IoT) ha transformado radicalmente nuestra interacción con el mundo actual, permitiendo la interconexión y recopilación de datos de una variedad de dispositivos. Desde la iluminación del hogar hasta la gestión de la energía, el IoT se ha vuelto esencial en nuestra vida diaria, abriendo nuevas oportunidades en sectores como la salud, el transporte y la industria. Los microcontroladores, como ESP-8266, ESP-32, ESP-05 y ESP-01, desempeñan un papel crucial al permitir la recopilación de datos y el control de dispositivos mediante sensores y actuadores, programados eficientemente en lenguajes como C++.

Este Trabajo Fin de Grado se centra en la mejora integral de la eficiencia energética y la seguridad en el hogar. El objetivo principal es impulsar la innovación y la integración tecnológica para transformar y potenciar la inteligencia y eficiencia de los entornos domésticos. En este sentido, se considera la implementación de soluciones que aprovechan plenamente las capacidades del Internet de las Cosas (IoT).

Como resultado, se ha logrado implementar un sistema completo que integra diversas tecnologías, permitiendo un control avanzado de la domótica en el hogar.

Palabras clave: Internet de las cosas, ESP-8266, Domótica, Sensores, Microcontroladores.

Índice

Capítulo 1 INTRODUCCIÓN

En la actualidad el Internet de las Cosas (IoT, por sus siglas en inglés) está revolucionando la manera en que interactuamos con el mundo. Cada vez más dispositivos están conectados a la red, lo que permite una mayor interconexión entre ellos y una mejor recolección de datos. Desde la iluminación de nuestras casas hasta la gestión de la energía, el IoT se está convirtiendo en una parte cada vez más integral de nuestra vida cotidiana, abriendo nuevas posibilidades en sectores como la salud, el transporte y la industria, mejorando la eficiencia y ofreciendo propuestas innovadoras a los desafíos actuales.

Los microcontroladores son componentes fundamentales en el desarrollo de soluciones IoT, ejemplos de estos son ESP-8266, ESP-32, ESP-05 y ESP-01. Estos dispositivos permiten la recolección de datos y el control de dispositivos conectados a la red, a través de sensores, actuadores y otros componentes. Con lenguajes como C++ es posible programar este tipo de microcontroladores de manera eficiente, utilizando diferentes librerías y frameworks.

Un ejemplo del IoT en acción es un hogar inteligente, en el que los electrodomésticos, termostatos, cámaras de seguridad, cerraduras y otros dispositivos están conectados a una red y son controlados por un sistema centralizado, como un asistente virtual. Este sistema puede ajustar la temperatura de la casa según las preferencias del usuario, encender o apagar las luces según la hora del día o la presencia de personas en una habitación, y alertar al usuario en caso de que se detecte una fuga de agua o una intrusión.

Otro ejemplo podría ser el monitoreo del estado de un paciente en un hospital mediante sensores que miden su frecuencia cardíaca, presión arterial, temperatura y otros indicadores de salud. Los datos se pueden enviar a una plataforma de análisis en tiempo real que alerte al personal médico si se detecta algún problema o fluctuación en los signos vitales del paciente.

El potencial del IoT es enorme y está cambiando la forma en que vivimos y trabajamos, por lo que puede llegar a ser muy interesante explorar las diferentes opciones que puede aportarnos este campo en algo tan presente en nuestra vida como es nuestro hogar.

1.1. Objetivos

Este trabajo consiste en el diseño y desarrollo de una solución IoT que nos permita controlar y monitorear la domótica de nuestro hogar, utilizando placas Wi-Fi ESP-8266 como la que se muestra en la Figura 1 junto a una combinación de sensores unidos a un conjunto de actuadores como enchufes y luces inteligentes. Además de la parte hardware, en el presente trabajo, se desarrollará una aplicación software (e.g., una aplicación web), con la que se permitirá al usuario monitorizar la solución IoT. La propuesta se enfoca en mejorar la eficiencia energética y la seguridad de la vivienda.

Este Trabajo Fin de Grado (TFG) se enfoca en la automatización integral de una vivienda mediante la implementación de diversos sensores, incluyendo sensores de temperatura y humedad. También incorpora un sensor de ultrasonido que cumple la función de detector de movimiento, un sensor de presión y altitud que, en combinación con los sensores de temperatura, forma una estación meteorológica propia y un sensor de luz con el que podemos saber si se ha quedado algo encendido en la casa y poder apagarlo ya que se encuentra conectado a una regleta inteligente. Este conjunto de sensores y dispositivos se integra en un sistema centralizado basado en una Raspberry Pi que contiene Node-RED utilizado para diseñar flujos de datos y estos datos se almacenan de forma segura en una base de datos MongoDB, lo que facilita el acceso y análisis en tiempo real y a lo largo del tiempo

Este objetivo general puede ser descompuesto en los siguientes entregables:

O1. **Diseño, montaje y programación de la solución IoT:** Esta fase comprenderá el ensamblaje de la solución IoT con sus respectivos sensores, llevando a cabo la programación y comunicación necesaria entre dispositivos.

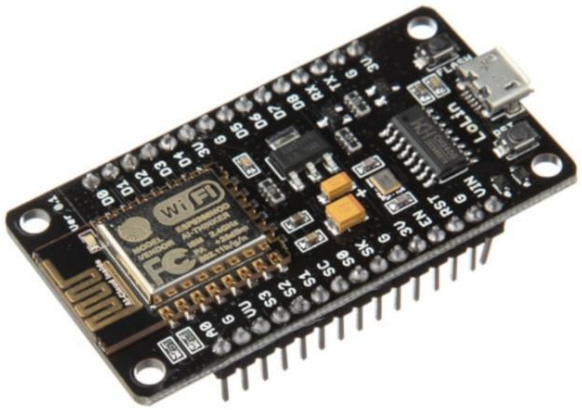


Figura 1. Placa NodeMCU Wi-Fi ESP-8266

O2. **Creación de un Servidor en Rapsberry Pi con Mosquitto, Node-Red y MongoDB:** Establecimiento de un servidor en una Rapsberry Pi con el broker MQTT Mosquitto, la base de datos MongoDB y la herramienta de desarrollo Node-Red. La motivación del presente objetivo es la de crear un centro de control tecnológico con el que gestionar y analizar datos de sensores en el entorno doméstico.

O3. **Creación de una aplicación software que permita a los usuarios monitorizar la solución IoT:** Que permita la supervisión a tiempo real de los datos recopilados por la solución IoT. Esta aplicación usará el protocolo MQTT garantizando así una transmisión de los datos fiable y eficiente.

O4. **Creación de una base de datos para la gestión de estos:** En la que se almacenarán los datos recogidos por la solución IoT facilitando así la posterior consulta y administración de la información necesaria para el desarrollo del plan energético.

O5. **Redacción de un manual de despliegue:** Que describa paso a paso las instrucciones necesarias para la instalación y despliegue de todos los componentes de la solución IoT. Este manual será una guía práctica para los futuros implementadores.

1.2. Estructura del documento

El documento está estructurado en siete capítulos que abarcan diversos aspectos del proyecto de IoT y domótica. El primer capítulo, "INTRODUCCIÓN", establece el contexto del proyecto y sus objetivos. El segundo capítulo, "INFRAESTRUCTURA DE LA APLICACIÓN ", describe la arquitectura del sistema montado. A continuación, el capítulo 3 "DESARROLLO DE LA APLICACIÓN" describe el proceso de desarrollo, destacando tanto el hardware como el software empleado. El capítulo 4 “INTEGRACIÓN DE DISPOSITIVOS CON NODE-RED" se adentra en la conexión de dispositivos IoT mediante Node-RED. A continuación, el capítulo 5 "INTERFACES GRÁFICAS" presenta el dashboard de Node-RED como interfaz de control para los sensores.

Finalmente, el capítulo 6 "CONCLUSIONES, PRESUPUESTO Y LÍNEAS FUTURAS" resume los logros del proyecto, aborda los desafíos encontrados, detalla el presupuesto utilizado y plantea áreas de mejora futuras, destacando especialmente la necesidad de fortalecer la seguridad y considerar expansiones adicionales. Esta estructura proporciona una visión completa y organizada de todo el proyecto en un formato que facilita la comprensión y la revisión.

1.3. Tecnologías usadas

1.3.1. Plataforma IoT para la conectividad inalámbrica

El dispositivo usado como eje principal de este proyecto es una placa WiFi de bajo coste ESP-8266 como la mostrada en la Figura 1, la cual se encuentra conectada a una protoboard para una mayor comodidad en su uso. La programación de estas placas Arduino se lleva a cabo mediante el lenguaje de programación C++, usando librerías proporcionadas por el propio entorno de programación IDE de Arduino y otras externas creadas por los usuarios. El IDE de Arduino es una herramienta esencial para programar y desarrollar proyectos de microcontroladores de Arduino, de esta manera tal y como describen en su sitio web, la misión de Arduino es permitir que cualquier persona mejore su vida a través de tecnologías digitales y electrónica accesibles (Arduino, 2023).

La placa ESP-8266 se comunica con el ordenador mediante un cable USB. Usando el IDE de Arduino, se selecciona la placa correspondiente y el puerto COM en el que se conecta en el menú desplegable “Herramientas”. Una vez desarrollado el programa, se verifica, compila y se transfiere la información entre la placa y el dispositivo.

Para este proyecto, se ha aprovechado la versatilidad de la placa ESP-8266, ampliamente reconocida por su popularidad en proyectos de IoT y una opción excelente para las personas que dan sus primeros pasos en la programación de sistemas integrados.

Gracias a que se encuentra dotada de conectividad Wi-Fi, la placa ESP-8266 se convierte en una elección excepcional para implementar el protocolo MQTT (Message Queuing Telemetry Transport) siendo este una columna vertebral en la comunicación entre dispositivos IoT. La ESP-8266 asume roles duales: actúa como publicador (emisor de datos) y suscriptor (receptor de datos) en entornos MQTT, lo que posibilita el intercambio fluido de información en tiempo real.

La polivalencia de la ESP-8266 permite la interacción con una diversidad de sensores. Mediante los pines GPIO (Entrada/Salida), la placa se sincroniza con sensores que van desde medidores de temperatura y humedad, detección de movimiento y luminosidad hasta cámaras de temperatura o de videovigilancia. Esta capacidad favorece la obtención precisa de datos y su posterior supervisión, siendo estos aspectos importantes en aplicaciones que conllevan control y automatización.

1.3.1. Sensores de arduino integrados en la solución IoT

La integración de sensores de Arduino presenta un papel clave en la implementación de soluciones IoT, posibilitando la captura y transmisión de datos. Estos sensores, cuidadosamente seleccionados por su relevancia y funcionalidad, son componentes esenciales que enriquecen la capacidad de la solución para adquirir información del entorno. A continuación, se describirán en detalle cada uno de los sensores de Arduino que, incorporados en el presente trabajo, examinando sus características, su funcionamiento y contribuciones específicas a la recopilación de datos en tiempo real.

1.3.1.1. Sensor DHT11

El sensor DHT11 es un dispositivo electrónico utilizado para la medición de la temperatura y humedad relativa del ambiente. Este se trata de un sensor simple pero efectivo que proporciona los datos necesarios para observar y controlar las condiciones ambientales a desarrollar.

Una de las ventajas que ofrece el DHT11, además de medir la temperatura y la humedad, es que es digital. A diferencia de sensores como el LM35, este sensor utiliza un pin digital para enviar la información y, por tanto, brinda una mayor protección frente al ruido (Hernández, 2020).

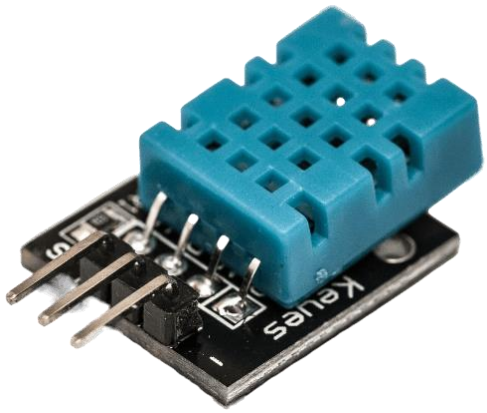


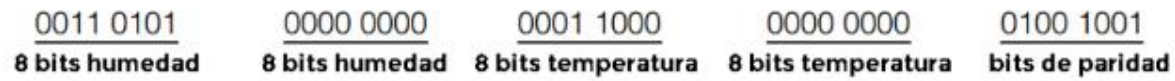
Figura 2. Sensor DHT11.

En la versión utilizada para la realización de este trabajo, el sensor se encuentra soldado a una placa PCB o circuito impreso que le permite tener una resistencia de 5 kΩ tal y como podemos ver en la Figura 2. Esta versión cuenta con 3 pines, siendo el colocado

más a la izquierda el pin de DATA o señal, el central VCC o alimentación y el colocado a la derecha GND o conexión a tierra.

Este sensor necesita de una alimentación de entre 3,3 V y 5 V y tiene un consumo de 2,5 mA. Su rango de medición de temperatura es de entre 0ºC y 50ºC con una precisión a partir de 25ºC de ±2ºC. En lo que respecta a la humedad su rango de medición es de entre 20% RH a 90% RH con una precisión entre 0ºC y 50ºC de ±5% RH.

La forma en la que transmite los datos es haciendo una conversión de analógico a digital dentro del mismo dispositivo. Tenemos 40 bits que se corresponden con la información sobre la temperatura y la humedad. A la vista de la **¡Error! No se encuentra el origen de la referencia.**Figura 3 los bits se dividen en grupos de 8, en los que se encuentra la parte entera y decimal tanto de la temperatura como de la humedad y un grupo final de



bits de paridad que aseguran que la información es correcta.

Figura 3. Grupo de bits temperatura y humedad.

1.3.1.2. Sensor BMP280

El sensor BMP280 es un tipo de sensor dedicado a la medición de la presión atmosférica y la temperatura, tal y como ya hacía su predecesor el sensor BMP180. Entre las ventajas a destacar frente a su predecesor se encuentra una mejora en la precisión tanto en la medición de la temperatura como en la presión atmosférica, un menor consumo y un mayor rango de medición, lo que lo hace mucho más versátil.

Entre las especificaciones generales de este sensor se encuentra un rango de medición de temperatura entre -40ºC y 85ºC y el de presión entre 300 hPa y 1100 hPa con una precisión de ±1ºC y ±1 hPa. En cuanto a la alimentación necesita de una tensión mínima de 3.3V y máxima de 5V, teniendo un consumo de 2.7 μA.

Una curiosidad de este sensor es su capacidad de ser leído mediante los protocolos de comunicación I2C (Inter-Integrated Circuit) o SPI (Serial Peripheral Interface). Con ello se consigue una integración más sencilla y adaptable a una amplia gama de sistemas y plataformas (Macho, 2020).

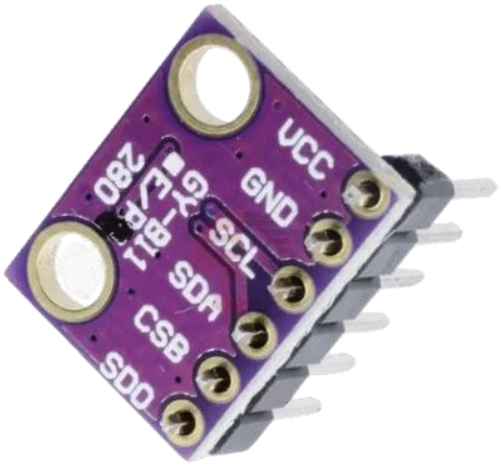


Figura 4. Sensor BMP280.

Observando la Figura 4, se destaca que este sensor esta compuesto de 6 pines que desempeñan roles específicos en su funcionamiento. El primer pin, denominado VCC o alimentación tal y como se ha comentado anteriormente necesita de una tensión de 3.3 V. A continuación se encuentra el pin GND o conexión a tierra.

El pin SCL o línea de reloj I2C, es el pin utilizado para sincronizar la transmisión de datos entre el sensor y otro dispositivo, funcionando análogamente a un reloj en su sincronización. El pin CSB se encarga de seleccionar el sensor cuando varios dispositivos comparten el mismo bus SPI. Y por último el pin SDO es el pin encargado de establecer la conexión I2C.

1.3.1.3. Sensor FC-51

El sensor FC-51 es un sensor de obstáculos reflectivo infrarrojo capaz de llevar a cabo mediciones de proximidad mediante la tecnología infrarroja (IR). Este se encuentra compuesto por un transmisor y un receptor, siendo estos dos de sus componentes esenciales (Figura 5). El transmisor se encarga de emitir una energía infrarroja que se propaga hacia la superficie circundante, mientras que el receptor capta esta energía,

permitiéndole detectar la presencia de obstáculos ubicados en la línea de visión del módulo.

Figura 5. Sensor FC-51.

Una de las particularidades de este sensor se encuentra en su habilidad para trabajar tanto en condiciones de luz ambiente como en entornos de baja luminosidad. Esta versatilidad le otorga un amplio espectro de aplicaciones prácticas en diversas situaciones, desde la detección de objetos en áreas iluminadas hasta la percepción de obstáculos en la oscuridad.

Este sensor cuenta con 3 pines, destacando en la imagen el ubicado en el extremo derecho como el pin VCC, el cual requiere una tensión de 3.3 V para su funcionamiento. En el centro se encuentra el pin GND, correspondiente a la conexión a tierra, y en la parte izquierda se localiza el pin de transmisión de datos. En cuanto a sus especificaciones, cabe destacar su chip de funcionamiento LM393 que contribuye a su funcionamiento óptimo y fiable. Además, es importante tener en cuenta que este sensor exhibe un ángulo de detección amplio y preciso, siendo este de 35º, abarcando así una zona significativa y convirtiéndolo en una herramienta sencilla y a la misma vez efectiva.

1.3.1.4. Sensor HC-SR04

El sensor HC-SR04 es un sensor ultrasónico diseñado para medir distancias con precisión gracias a la emisión y recepción de ondas ultrasónicas. Sus componentes principales tal y como podemos ver en la Figura 6 incluyen un transmisor ultrasónico, encargado de emitir pulsos ultrasónicos; un receptor ultrasónico, que captura ecos de los pulsos y genera una señal eléctrica como respuesta; y un microcontrolador que controla la generación y detección de estos pulsos y el cálculo de la distancia.

Figura 6. Sensor HC-SR04.

En el contexto de este proyecto, el fin de este sensor va a ser el de detectar movimiento. A pesar de esta variación, el principio fundamental sigue siendo el mismo: el sensor enviará pulsos ultrasónicos y medirá el tiempo que tarda en recibir el eco. La distinción radica en que, en lugar de calcular la distancia, tratará de identificar cambios en el tiempo de eco que indiquen la presencia de un objeto o persona en movimiento para proteger la privacidad y seguridad de la casa.

El sensor se encuentra configurado por 4 pines: el pin VCC o alimentación, el cuál necesita de una tensión de 5V; el pin Trig o pin de entrada, encargado de enviar el pulso ultrasónico; el pin Echo o pin de salida, trata de indicar el tiempo que se tarda en recibir el eco; finalmente, el pin GND o tierra, encargado de cerrar el circuito.

1.3.1.5. Sensor MQ-5 y MQ-135.

La familia de sensores MQ es una serie de dispositivos dedicados a la detección de gases muy conocidos y utilizados en aplicaciones de seguridad y domótica. Cada integrante de esta familia está diseñado para detectar un gas en particular. Su funcionalidad se basa en cambios en la resistencia eléctrica que se encuentra en su interior. Esta resistencia está fabricada con un material muy sensible a la presencia de determinados gases, siendo generalmente el material con el que están hechas óxidos metálicos.

En el caso del presente trabajo, que trata de aumentar la seguridad de la vivienda, actualmente se encuentra instalado gas natural. Por consiguiente, los sensores seleccionados son el sensor MQ-5, especializado en detectar fugas de gas natural, y el MQ-135, siendo este el más genérico y desempeñando el papel de sensor de humo.

Este tipo de sensores reaccionan químicamente con gases a temperatura elevada, alterando su resistencia provocando cambios en su conductividad eléctrica, detectan estos cambios y generan una señal analógica proporcional a la concentración del gas. La señal se procesa en un microcontrolador para interpretar la presencia y cantidad del gas detectado.

Figura 7. Sensor MQ-135 y MQ-5.

Ambos sensores están equipados con 4 pines, como se muestra en la Figura 7 y sus funciones son idénticas: el pin VCC, encargado de la alimentación a 5V asegurando estabilidad; el pin AOUT o salida analógica, que proporciona una señal analógica

reflejando la concentración de gas, detectada y medida por el microcontrolador; el pin DOUT o salida digital, en el caso del presente trabajo sin función particular; y el GND o conexión a tierra.

1.3.1.6. Sensor fotosensible LM393.

El sensor fotosensible LM393 es ampliamente utilizado en aplicaciones que requieren detección de luz. Este pertenece a la serie de comparadores de voltaje LMx93 fabricada por varias compañías.

Al ser un comparador de voltaje, su funcionamiento se basa en comparar dos señales de entrada y producir una salida digital. En este caso, una de las entradas se conecta al umbral y la otra entrada recibe la señal de luz. El sensor genera una señal de salida digital en función de si la luz detectada sobrepasa o no el umbral previamente establecido. Otro aspecto que destacar del sensor es que permite ajustar la sensibilidad, permitiendo calibrar el umbral según las necesidades del usuario.

Las posibles aplicaciones de este sensor van desde su uso en sistemas de iluminación tanto interiores como exteriores, interruptores de luz basados en la detección de luz, sistemas de seguridad para la noche y sistemas de ahorro de energía adaptativa a la luminosidad del entorno.

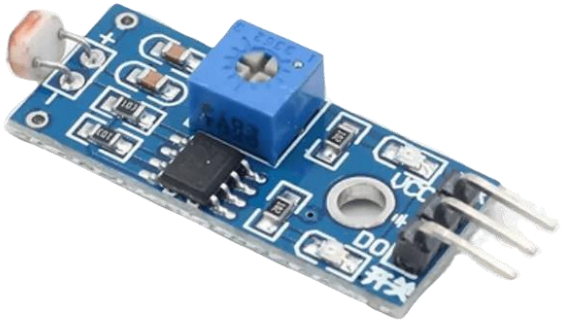


Figura 8. Sensor fotosensible LM393.

El sensor se encuentra equipado con 3 pines: Vcc (alimentación), GND (tierra) y OUT (salida). La salida OUT se conecta a un microcontrolador o a otros circuitos electrónicos para procesar la señal de detección.

1.3.1.7. Regleta inteligente.

En el caso del presente proyecto, se ha fabricado una regleta inteligente personalizada que permite controlar múltiples dispositivos eléctricos de forma remota y automatizada. Esta regleta inteligente se diferencia de las regletas convencionales en que está equipada con tecnología IoT (Internet de las cosas) que la hace completamente programable y controlable desde un dispositivo móvil u ordenador a través de la plataforma Node-RED y una ESP-8266.

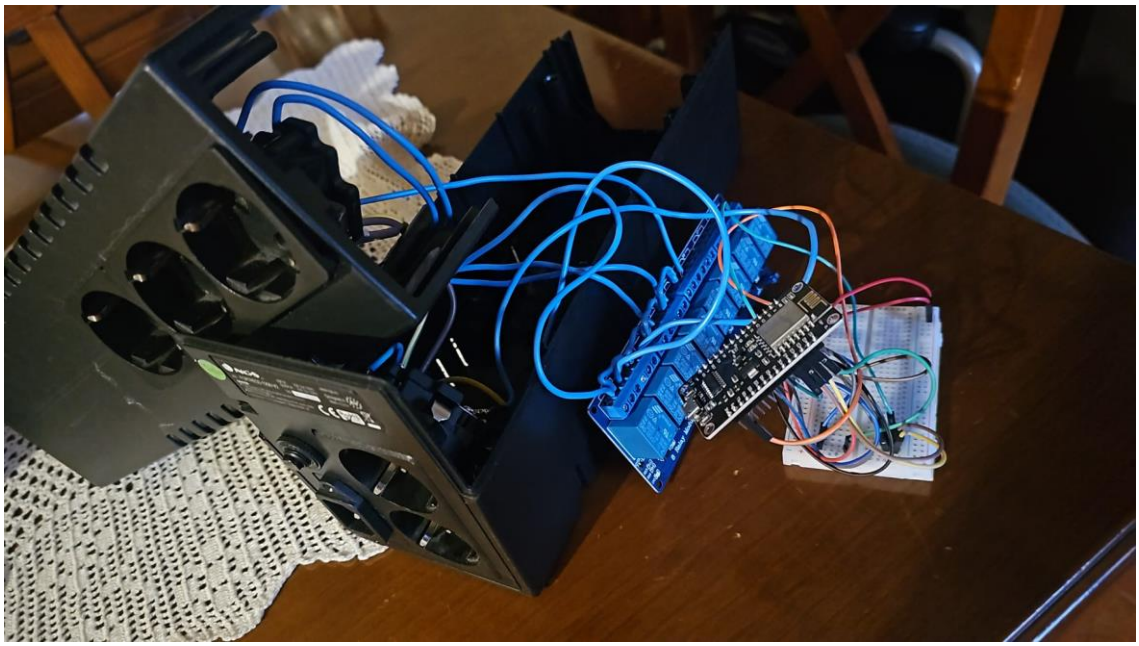


Figura 9. Regleta inteligente (interior).

Figura 10. Regleta inteligente (exterior).

1.3.2. Plataforma de control y servidor local basado en Raspberry Pi

La Raspberry Pi es un ordenador de bajo coste y con un tamaño compacto, parecido al de tarjeta de crédito, puede ser conectada a un monitor o una TV, y usarse con un ratón y teclado estándar. Es un pequeño ordenador que corre un sistema operativo Linux (Ubuntu Server 20.4) capaz de permitirle a personas de todas las edades explorar la computación y aprender a programar lenguajes como Scratch y Python. Es capaz de hacer la mayoría de las tareas típicas de un ordenador de escritorio, desde navegar en internet, reproducir videos en alta resolución, manipular documentos de ofimática, hasta reproducir juegos (MCIelectronics, 2019).

En el contexto del presente trabajo, la Raspberry Pi es una herramienta valiosa debido a su capacidad de actuar como un servidor local para una variedad de aplicaciones, todo ello para la gestión y control de dispositivos inteligentes en un entorno doméstico. Los servicios claves que se necesitan alojar en ella son los siguientes:

1.3.2.1. Integración de Node-RED

Tal y como se ha comentado en el apartado *<<1.1.Objetivos>>,* Node-RED es una herramienta de programación visual basada en nodos que facilita la automatización de tareas conectando los bloques de funciones predefinidas. Node-RED es un componente central para el presente trabajo, ya que se le dará el uso de plataforma de control,

proporcionando simplicidad y flexibilidad en la creación de la lógica fundamental de la automatización llevada a cabo.

Node-RED proporciona una interfaz gráfica intuitiva en la que se puede representar funciones específicas, como leer sensores, ejecutar acciones o enviar notificaciones. Estos nodos se pueden conectar y configurar para definir el comportamiento final deseado.

Al actuar la Raspberry Pi como servidor local, todos los flujos se ejecutan en esta, teniendo así un control centralizado sobre el sistema. Los flujos pueden ser modificados, actualizados y ajustados a futuros nuevos proyectos que se quieran realizar con ellos, lo que nos brinda una flexibilidad inigualable.

En lo que respecta a las ventajas más destacadas del uso de Node-RED, sobresale su pequeña barrera de entrada para aquellos no expertos en programación.

1.3.2.2. Almacenamiento en la base de datos MongoDB

La gestión eficiente de los datos obtenidos representa un aspecto crucial en este proyecto, ya que proporciona la base para el análisis, la toma de decisiones y posibles propuestas de mejoras futuras del entorno inteligente fabricado. Para el presente proyecto, se ha incorporado la base de datos MongoDB, una base de datos NoSQL, lo que significa que su almacenamiento de datos no es en tablas y filas, sino que se base en un almacenamiento en documentos. Esto proporciona una facilidad para proyectos de domótica en la que los dos datos generados o recibidos son variados y no poseen un formato en concreto.

La elección de una base de datos NoSQL como MongoDB en lugar de una base de datos SQL convencional para el presente proyecto, está respaldada por varias consideraciones. En primer lugar, las bases de datos NoSQL, son conocidas por su flexibilidad en términos de esquema de datos, lo que permite adaptarse a cambios en la estructura de los datos sin complicaciones. Esto es útil, ya que, en el presente proyecto, la variedad de datos puede evolucionar con el tiempo. Además, MongoDB ofrece escalabilidad horizontal, lo que facilita la gestión de grandes volúmenes de datos y tráfico en un entorno en constante crecimiento, como la domótica. Su alto rendimiento en operaciones de lectura y escritura es beneficioso para respuestas rápidas a eventos en tiempo real. Si los datos incluyen información no estructurada o semiestructurada, MongoDB puede manejarlos de manera eficiente. Además, su formato de almacenamiento de documentos BSON (similar a

JSON) es adecuado para representar datos de dispositivos IoT. MongoDB también se integra bien en el ecosistema IoT, simplificando la interacción con otras tecnologías utilizadas en proyectos de domótica.

Al igual que con Node-RED, la Raspberry Pi aloja una instancia de la base de datos MongoDB, estableciendo una infraestructura completa que abarca desde la creación de los flujos hasta el almacenamiento de los datos del entorno y agregando un nivel mayor de inteligencia al sistema.

Además de la capacidad de análisis, MongoDB ofrece respaldo y persistencia de datos. Esto asegura que los datos críticos no se pierdan en caso de fallos de hardware o interrupciones inesperadas. La Raspberry Pi actúa como un punto central de almacenamiento confiable, garantizando que los datos estén disponibles y accesibles en todo momento.

Para hacer un buen uso de esta, se dispone de una interfaz gráfica llamada MongoDB Compass, que facilita la administración y exploración de las bases de datos MongoDB.

Para hacer uso de esta herramienta, primero se debe instalar MongoDB Compass en el sistema, a través del sitio web oficial de MongoDB. Tras la instalación, se debe configurar el servidor MongoDB como se muestra en la Figura 11.

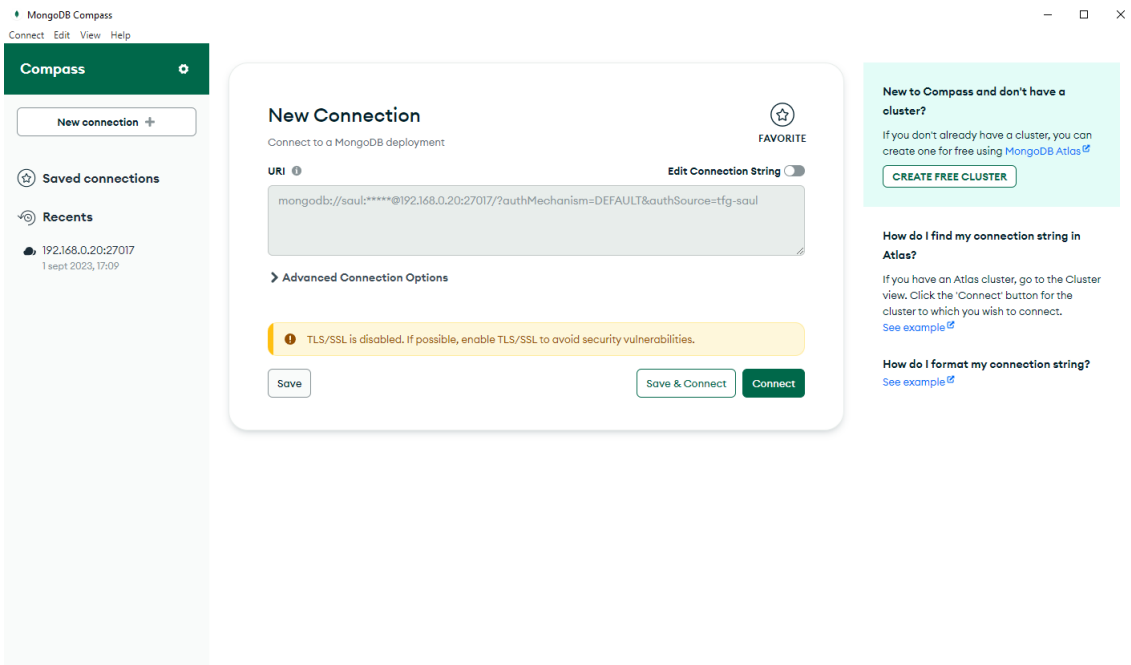


Figura 11. Inicio MongoDB Compass.

Para ello, se ingresa la dirección IP o el nombre de host del servidor MongoDB. Por defecto, MongoDB usa el puerto 27017. Si se ha habilitado la autenticación en el servidor MongoDB, se debe seleccionar la opción "Username/Password" e ingresar las credenciales.

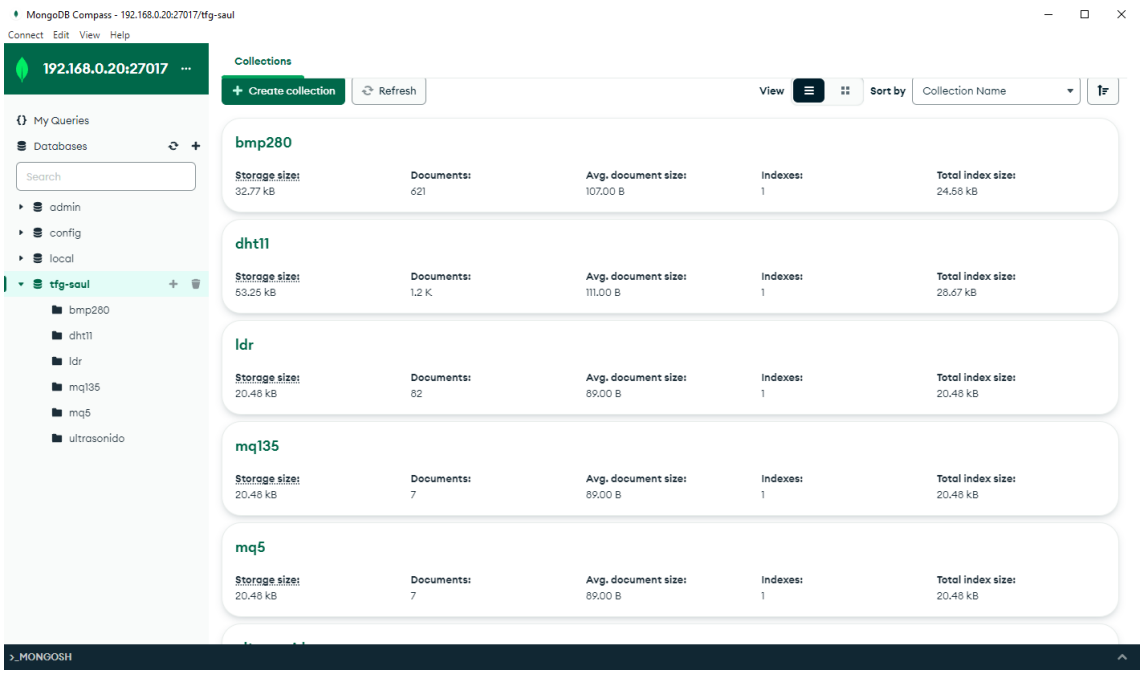


Figura 12. Base de datos.

Tras todo lo anterior, para integrar MongoDB con Node-RED y la ESP-8266, se configura la comunicación entre la ESP8266 y Node-RED, creando un flujo para procesar y validar los datos. Luego, se conecta Node-RED con MongoDB y se almacena los datos en las colecciones creadas. Finalmente, es aconsejable supervisar y mantener el sistema para garantizar un buen almacenamiento de los datos de los dispositivos IoT en MongoDB.

1.3.2.3. Funcionalidad del servidor MQTT Mosquitto

MQTT es un protocolo de comunicación altamente eficiente diseñado para aplicaciones en el Internet de las Cosas (IoT). Fue desarrollado por IBM en la década de 1990 y se ha convertido en uno de los protocolos más populares para la comunicación entre dispositivos IoT gracias a su simplicidad y eficiencia.

Para una comunicación efectiva entre los dispositivos inteligentes es fundamental lograr un entorno completamente coordinado y altamente funcional. Para ello, se ha implementado el servidor MQTT Mosquitto en la Raspberry Pi, conectado de manera bidireccional todos los componentes del sistema.

Mosquitto es un servidor de código abierto que permite la transmisión de mensajes mediante un sistema de publicación y suscripción a “topics”. Distintos dispositivos están suscritos a diversos temas y pueden estar a la espera de que se publiquen mensajes en ellos o ser quienes los manden.

Para respaldar este servidor, se ha utilizado una Raspberry Pi 3, similar a la que se muestra en la Figura 13.

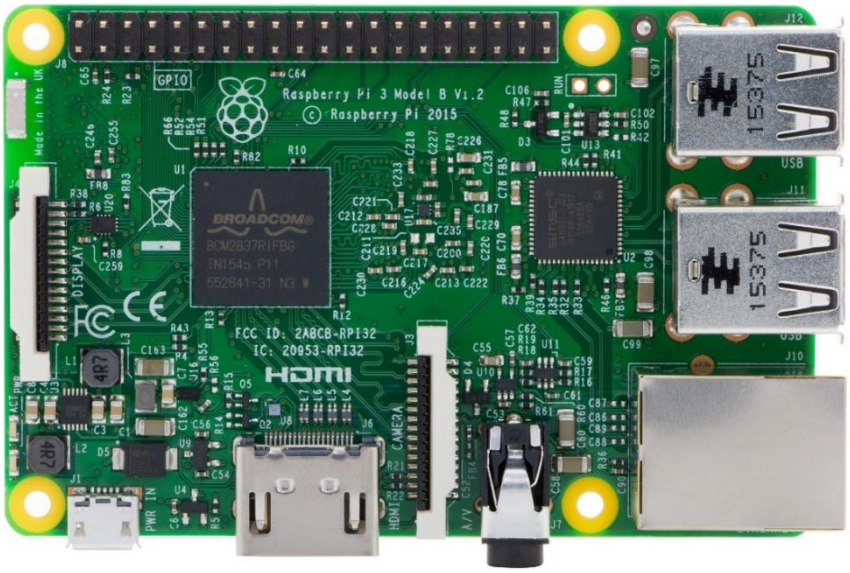


Figura 13. Raspberry Pi 3.

MQTT opera en una arquitectura de eventos, donde cada mensaje se distribuye a receptores previamente suscritos a un "tema" específico. En este contexto, el "Broker" trabaja de intermediario que dirige los mensajes a los destinatarios.

Lo más importante en MQTT es el "topic". Cuando un cliente se suscribe a un tema, está expresando su interés en recibir mensajes relacionados con ese tema, sin que el emisor conozca quiénes son los destinatarios finales. El emisor del mensaje no tiene conocimiento de los receptores específicos; en cambio, el Broker es quien administra esta información. (programarfacil, 2019)

Figura 14. Funcionamiento publicación/suscripción de MQTT.

El nivel de calidad de servicio (QoS) en MQTT determina la confiabilidad de la entrega de mensajes. MQTT ofrece tres niveles de QoS:

- QoS 0 (At most once): Entrega no garantizada, rápida pero con posibles pérdidas o duplicados.
- QoS 1 (At least once): Entrega garantizada al menos una vez, evita pérdidas, pero puede haber duplicados.
- QoS 2 (Exactly once): Entrega garantizada exactamente una vez, evita pérdidas y duplicados, pero es más lenta.

La interacción entre el servidor MQTT y Node-RED permite crear un escenario en el que un sensor detecta movimiento, el servidor MQTT notifica a Node-RED y este puede activar un flujo de trabajo para encender las luces en consecuencia.

1.4. Metodología de trabajo

En este apartado, se describe la metodología incremental e iterativa que se ha seguido para llevar a cabo el desarrollo del proyecto. El proceso se ha dividido en diferentes fases, cada una con sus objetivos y tareas específicas. A continuación, se detallan las etapas clave y los resultados obtenidos en cada una de ellas:

1. Estudio del estado del arte y tecnologías a utilizar: Se realizó una investigación exhaustiva para comprender la actualidad de la domótica y las tecnologías disponibles.

2. Diseño de la solución: Antes de proceder a la programación de las placas, se llevó a cabo un proceso de diseño detallado que incluyó la conceptualización de modelos y la selección de sensores adecuados para la implementación del sistema.

3. Programación de las placas ESP-8266: En esta fase, se abordó la programación de las placas ESP-8266, que funcionan como controladores de sensores y actúan como nodos en la red IoT. Se implementó la lógica para la lectura de los sensores y se estableció la comunicación bidireccional mediante el protocolo MQTT, garantizando la transmisión eficiente de datos entre los dispositivos y la aplicación central.

4. Desarrollo de la aplicación: Se creó un dashboard intuitivo y de fácil uso para controlar los actuadores, supervisar los dispositivos interconectados y almacenar los datos generados. La integración con Node-RED permitió la ejecución de flujos de automatización basados en eventos.

5. Instalación y montaje de la solución IoT: Esta etapa implicó la construcción física de la solución IoT, desde el prototipo inicial hasta el diseño final. Se montaron los dispositivos en ubicaciones estratégicas, estableciendo las conexiones necesarias y se configuraron los parámetros de red.

6. Verificación del sistema de monitorización y control de la domótica de la casa: Otra parte del proyecto realizado consistió en asegurarse de que todo el sistema implementado funcionara según lo esperado. Para lograrlo, después de desplegar la infraestructura en el hogar y confirmar su funcionamiento, se dejó en funcionamiento durante una semana para recopilar datos y asegurarse de que todo estuviera operando correctamente.

7. Elaboración de la documentación: Se preparó una documentación detallada que abarca todo el proceso desde el inicio hasta la finalización del proyecto.

1.5. Estado del arte

Con el propósito de contextualizar las técnicas y herramientas previamente discutidas, se ha considerado hacer referencia a investigaciones previas que se asemejan en enfoque y temática y compararlos con el presente proyecto.

En esta búsqueda, se ha identificado un trabajo titulado “Diseño y desarrollo de un sistema domótico basado en IoT para vivienda unifamiliar”. Al igual que en el presente trabajo, este se realizó con el objetivo de automatizar diversas actividades y posibilitar su control a través de internet. Para ello hace uso de microcontroladores equipados con módulos Wi-Fi ESP-8266. En comparación con otros proyectos, el empleo del módulo ESP-8266 es una característica que se comparte, ya que se considera una de las mejores opciones debido a su simplicidad, versatilidad y costo. La mayor diferencia respecto a este trabajo es la del uso de protocolos como Z-Wave, Zigbee, LoRaWAN y Sigfox. Estos protocolos son más adecuados para escenarios donde se necesita una cobertura más amplia o cuando se requiere una comunicación en malla para los dispositivos. Sin embargo, estos protocolos tienden a ser más complejos en su configuración y pueden requerir un hardware específico, lo que podría aumentar los costes y la complejidad en comparación con MQTT.

También existen otro tipo de propuestas más complejas, como el trabajo que lleva por título “Sistema de Gestión Domótica de una Vivienda”, en el cual se emprende el diseño e implementación de un sistema de gestión y control integral para una vivienda. A diferencia del presente proyecto, este trabajo se centra en una idea considerablemente más compleja, como la implementación y control de un sistema de calefacción radiante en el hogar, alejándose de la simplicidad de utilizar un microcontrolador para el encendido y apagado de un ventilador cuando la temperatura supera un umbral o para la detección de posibles fugas de gas en la cocina. También incluye como medida preventiva de seguridad un circuito de desconexión de electrodomésticos que ofrece la posibilidad de apagar los electrodomésticos y las luces de la vivienda cuando los habitantes salen de casa, lo que contribuye al ahorro de energía y a reducir el riesgo de posibles fallas eléctricas o cortocircuitos que podrían ocurrir cuando los dispositivos permanecen encendidos sin supervisión.

Este trabajo propone abordar los siguientes aspectos:

- Control de la Iluminación.
- Regulación de la Iluminación Artificial.
- Control de Temperatura.
- Control de Accesos.
- Gestión de Persianas.
- Detección de Presencia.
- Simulador de Presencia.
- Sistema de Alarmas.

Y finalmente realizar una simulación en Scada del producto final, que le permite al usuario conocer los estados de los distintos sensores.

Capítulo 2 INFRAESTRUCTURA DE LA APLICACIÓN

Este capítulo ofrecer una visión exhaustiva de la infraestructura desarrollada para este proyecto, que se basa en una Raspberry Pi como servidor local que aloja una serie de componentes esenciales. Entre ellos se destacan Node-RED, una herramienta de desarrollo versátil y flexible; MongoDB, una base de datos NoSQL que almacena datos de temperatura y humead, de detección de posibles fugas de gas natural y humo, de presión y altitud, de detección de movimiento y de detección de luz; y el servidor MQTT Mosquitto, que actúa como el canal de comunicación clave para los dispositivos IoT y sensores.

Además de esta base, se incorpora una solución IoT basada en la ESP-8266, junto con una variedad de sensores que recopilan datos vitales para la operación del sistema. Estos dispositivos se conectan al servidor a través del protocolo MQTT, permitiendo la transmisión eficiente y confiable de datos. La infraestructura también se enriquece con dos interfaces de usuario clave: un panel de control desarrollado en Node-RED que ofrece una visualización detallada y el control de dispositivos, y una interfaz de Telegram que simplifica la interacción con el sistema mediante comandos intuitivos.

La Figura 15 muestra un esquema de la infraestructura del sistema implementado.

2.1. Funciones y utilidad de cada componente del proyecto.

Para comprender completamente el funcionamiento de la infraestructura de este proyecto, es esencial analizar las funciones y la utilidad de cada componente y ver como se conectan entre ellos.

Comenzando con la Raspberry Pi que actúa como servidor central. Su función principal es la de ser el punto de convergencia donde se alojan y coordinan todos los componentes del sistema. Funciona como un cerebro que gestiona y controla la información y las acciones en el entorno. Es responsable de coordinar la comunicación entre los dispositivos IoT, procesar datos, almacenar información y ofrecer una interfaz para que los usuarios interactúen con el sistema.

Tal y como se ha comentado anteriormente, la Raspberry Pi alberga los servicios de Node-RED, la base de datos MongoDB y el servidor MQTT Mosquitto.

En el presente proyecto, Node-RED cumple una función fundamental como plataforma de desarrollo visual. Su principal función es proporcionar un entorno gráfico que facilite la creación de flujos de datos y la automatización del sistema. Esto significa que Node-RED actúa como un constructor, donde se pueden diseñar de manera eficiente cómo procesar los datos.

La comunicación entre Node-RED y la Raspberry Pi se basa en una arquitectura cliente-servidor, donde Node-RED actúa como el servidor web y la Raspberry Pi como el servidor central que aloja Node-RED.

En cuanto al servidor MQTT Mosquitto actúa como servidor de mensajería desempeñando la función principal de facilitar la comunicación entre los dispositivos IoT y el servidor central, que en este caso es la Raspberry Pi.

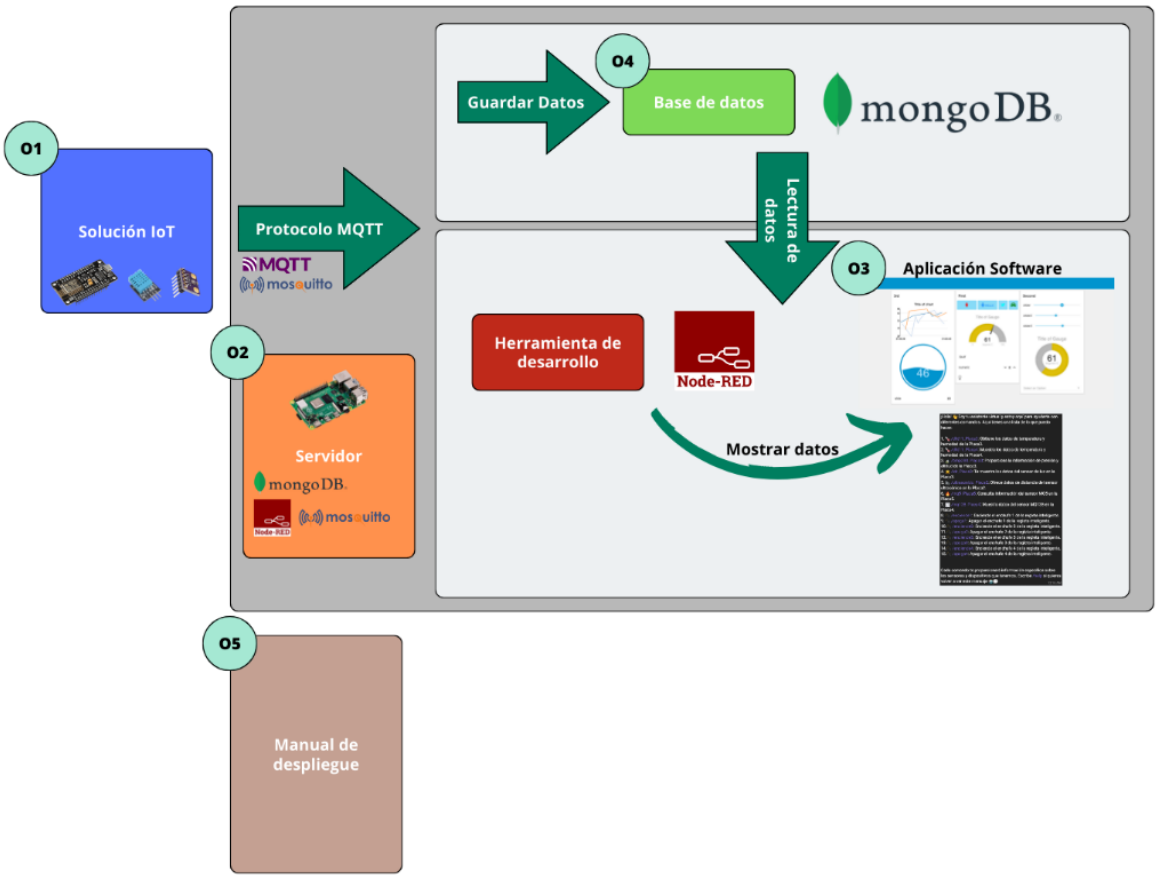


Figura 15. Esquema de la infraestructura del proeyecto.

Permite a las placas ESP-8266 publicar datos en temas específicos, que la Raspberry Pi recibe para su procesamiento y envío de comandos. Node-RED se integra como cliente MQTT para recibir datos y enviar comandos, actuando como una interfaz para el usuario. Esta comunicación bidireccional y eficiente garantiza la supervisión en tiempo real de datos de sensores y el control de dispositivos.

La función principal de MongoDB en este contexto es actuar como un sistema de almacenamiento para los datos recopilados de los sensores y dispositivos IoT. Al igual que Node-RED se encuentra instalada en la Raspberry Pi, y en el presente proyecto se usa para rescatar datos y poder mostrarlos en las interfaces gráficas creadas. Un claro ejemplo de su uso es la consulta del dashboard de los históricos generados tal y como podemos ver en Figura 16.

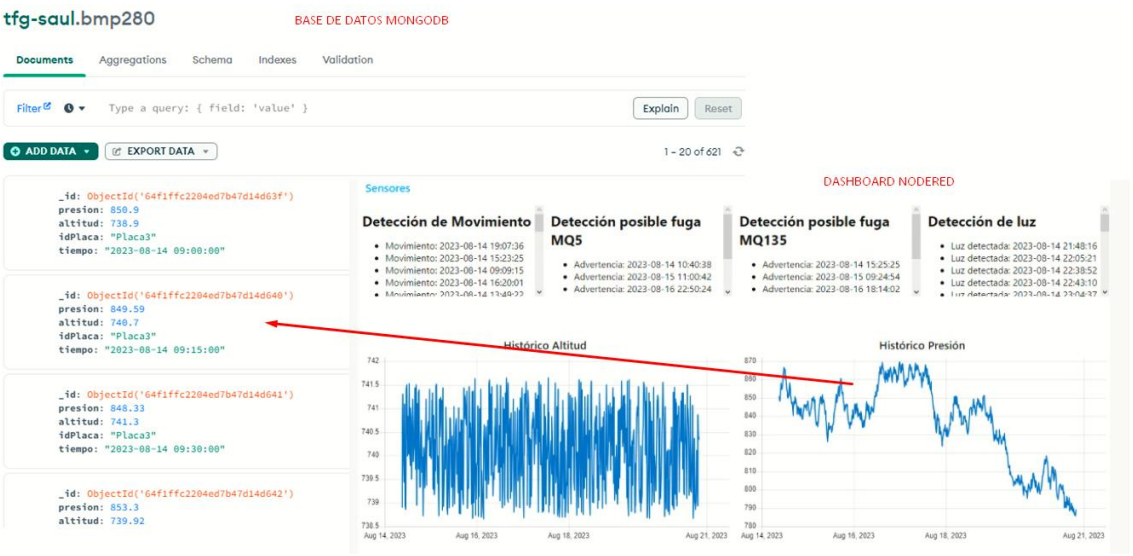


Figura 16. Interacción Node-RED y MongoDB.

Finalmente, las placas Wi-Fi ESP8266, equipadas con diversos sensores, están conectadas a la misma red que aloja el servidor central. Esta conectividad en la misma red simplifica enormemente la comunicación entre las placas y el servidor gracias al protocolo MQTT previamente mencionado, que se aprovecha de la capacidad de conexión inalámbrica Wi-Fi de las placas. Esto permite que las placas ESP8266 envíen y reciban datos al servidor central a través de la red Wi-Fi compartida, logrando así una comunicación fluida de toda la estructura del proyecto.

Capítulo 3 DESARROLLO DE LA APLICACIÓN

A lo largo de este apartado, se examinará en detalle el proceso de desarrollo de la aplicación, abordando desde la programación de los microcontroladores hasta la creación de flujos de automatización y la configuración de la infraestructura de comunicación.

Se verá cómo cada aspecto del desarrollo se alinea con los objetivos del proyecto y cómo estas elecciones impactan en términos de eficiencia, usabilidad y adaptabilidad de la aplicación. Este análisis nos permitirá no solo comprender el desarrollo en sí, sino también evaluar el éxito del proyecto en función de los resultados obtenidos. Se examinarán las métricas de rendimiento, la retroalimentación de los usuarios y la capacidad de adaptación a cambios futuros en el entorno tecnológico.

3.1. Desarrollo

3.1.1. Localización en el hogar

A continuación, se presenta un esquema de la disposición de las placas utilizadas en el proyecto. Estas placas están estratégicamente ubicadas en diferentes áreas de la casa para optimizar su funcionalidad y cobertura. La distribución busca abarcar zonas clave como la entrada principal, el dormitorio 1 y la cocina, garantizando un monitoreo efectivo de la temperatura, la seguridad y otros aspectos relevantes en todo el hogar. Cada ubicación se seleccionó considerando factores como la accesibilidad y la relevancia de los datos recopilados. Esta disposición busca proporcionar un control completo y una experiencia de usuario mejorada en todas las áreas de la vivienda sin interferir en la comodidad y el uso cotidiano de los espacios.

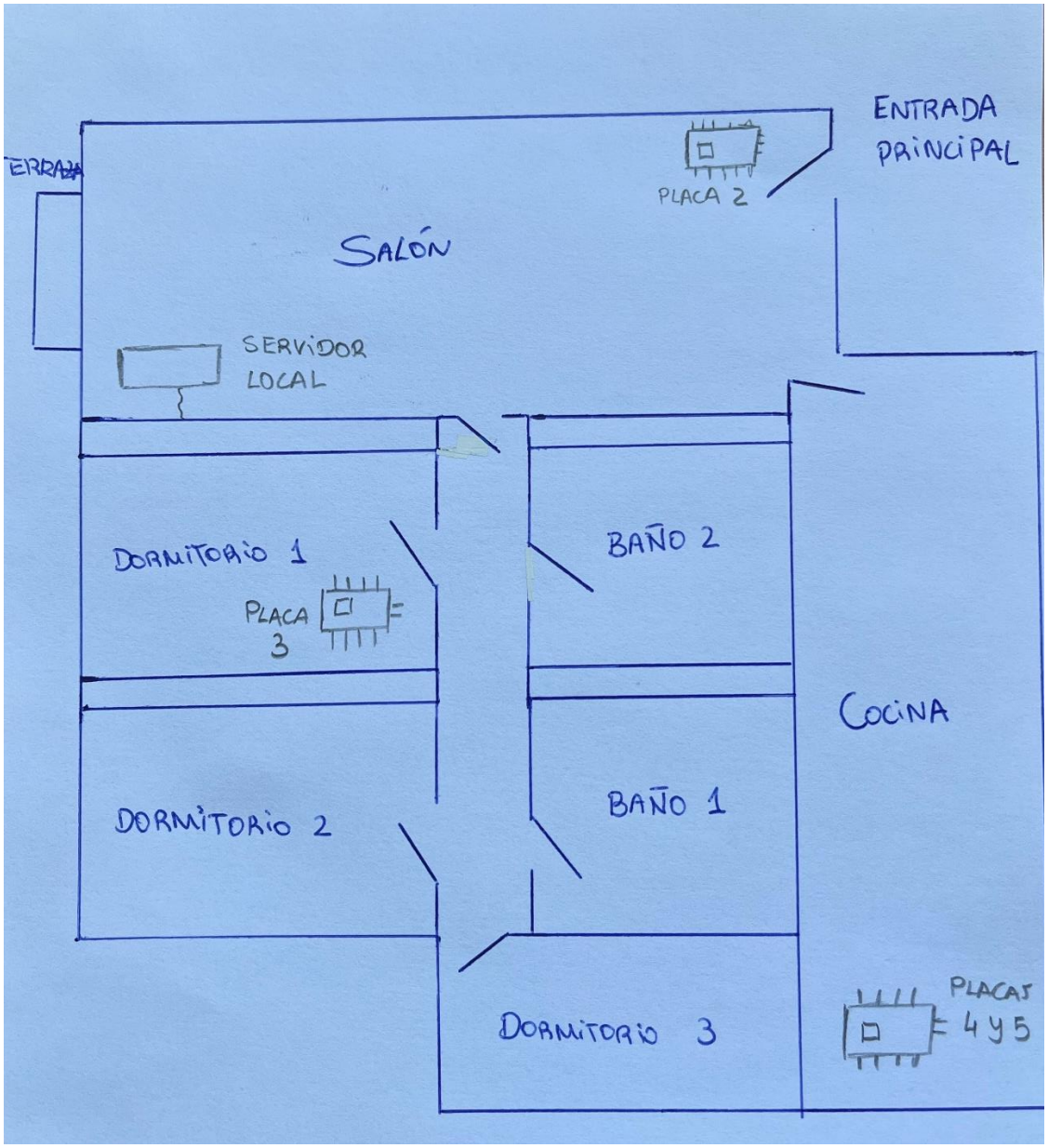


Figura 17. Esquema v*ivienda.*

3.1.2. Identificación de las placas ESP-8266

Para lograr una implementación eficiente y adaptativa en el montaje de las placas Wi-Fi ESP-8266, este se ha estructurado en cinco módulos. Cada módulo ha sido diseñado cuidadosamente para suplir todas las necesidades de la vivienda. Las placas han sido identificadas de la siguiente manera: la placa 2, se corresponde con la entrada principal; la placa 3 destinada a la habitación principal; y, finalmente, las placa 4 y 5 se encuentran en la cocina.

La placa 2 presenta una integración de un sensor de ultrasonido HC-SR04 para la detección de movimiento, un timbre y una alarma. Con esta configuración se consigue una mayor seguridad y comodidad en el entorno doméstico, gracias a la posibilidad de alertar al usuario ante situaciones inusuales (Figura 18).

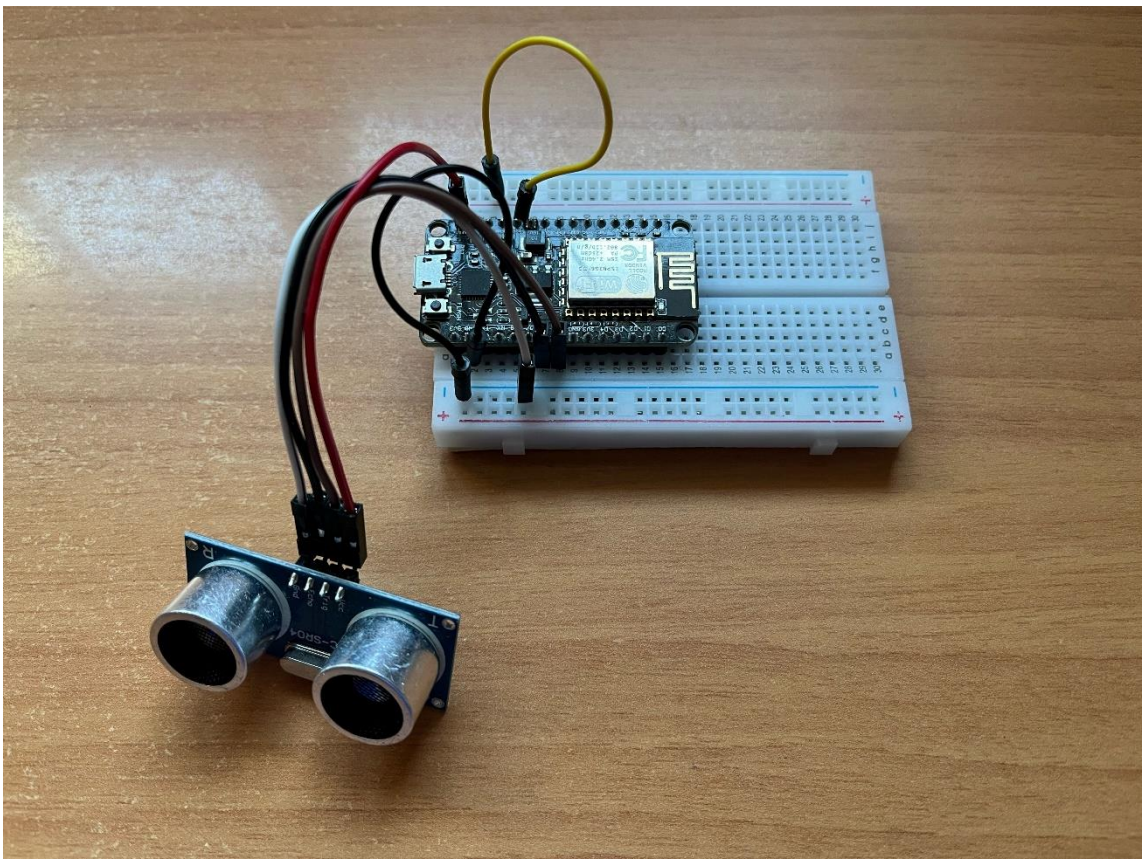


Figura 18. Placa 2 con sensor HC-SR04, alarma y timbre.

La placa 3, se encuentra configurada para optimizar el confort de la habitación. Se encuentra equipada con un sensor de temperatura DHT11, y un sensor de luz LM393 que facilita el ajuste automático de las condiciones ambientales de la habitación. Además, un relé controla el ventilador y la lámpara, permitiendo la interacción y el control remoto de los dispositivos (Figura 19).

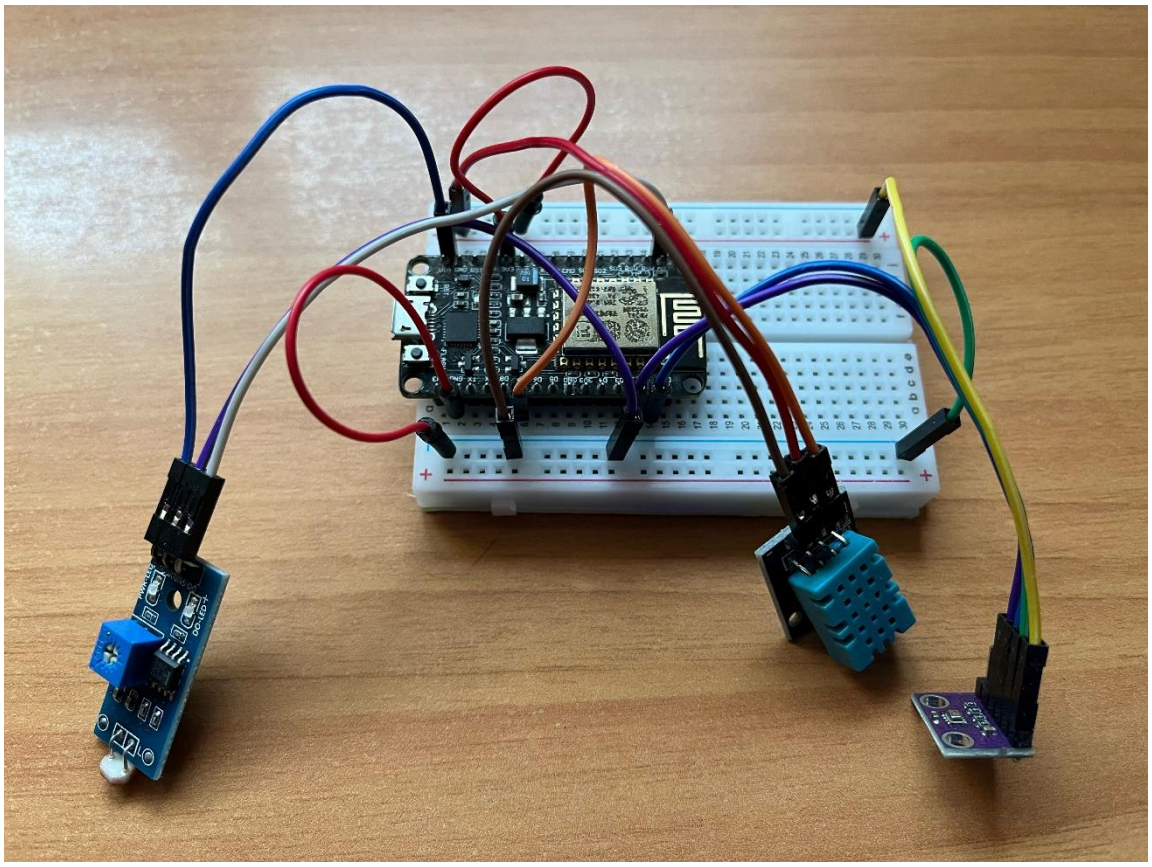


Figura 19. Placa 3 con sensor DHT11, LM393 y BMP280.

Las placas 4 y 5 situadas en la cocina destacan por brindar seguridad en este espacio. Se encuentran equipadas con un sensor de temperatura y humedad DHT11, el sensor de humos MQ-135, configurado para detectar posibles riesgos y el sensor MQ-5, configurado para alertar al usuario de una posible fuga de gas natural. La ubicación estratégica de estas placas refuerza la vigilancia y prevención del hogar (Figura 10 y Figura 21).

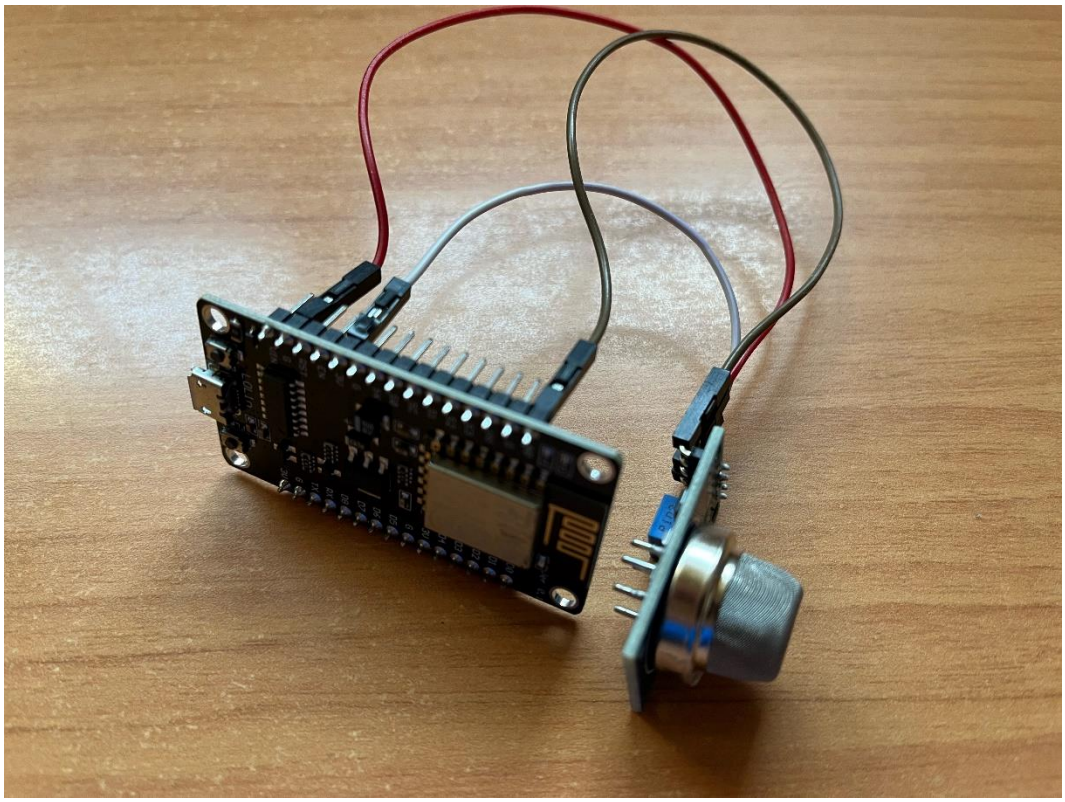


Figura 20. Placa 5 con sensor MQ-5.

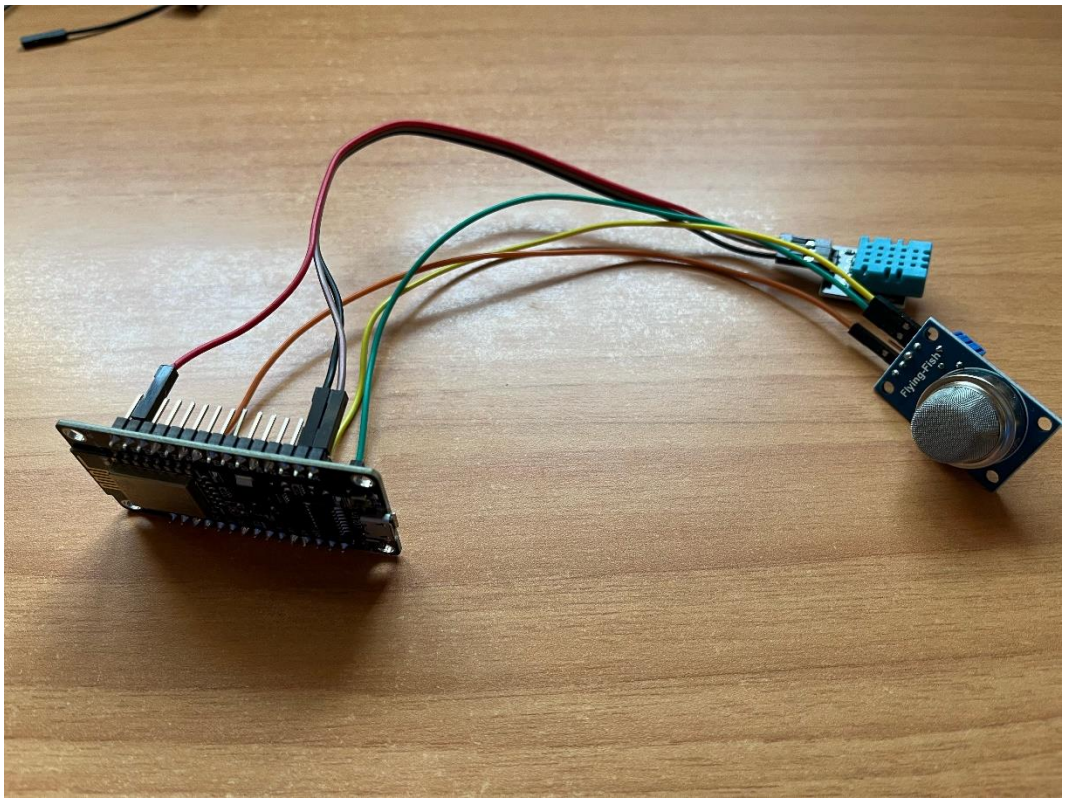


Figura 21. Placa 4 con sensor MQ-135 y DHT11.

3.1.3. Programación de las placas y sensores.

A continuación, se procede a detallar el funcionamiento de uno de los módulos montados, mostrando su configuración para desempeñar las funciones dadas. En el caso de los demás módulos, su funcionamiento es muy similar. Por lo tanto, en este documento actual no se detalla el funcionamiento de los otros módulos, pero la información completa siempre está disponible en los archivos adjuntos para su consulta.

En este caso, se procede a explicar el programa usado en la placa 3 que está formada por el sensor DHT11, el sensor LM393 y el BMP280.

```
1   #include <DHTesp.h>         // Sensor DHT11
2   #include <PubSubClient.h>   // MQTT
3   #include <ESP8266WiFi.h>    // PLACA
4   #include <DNSServer.h>
5   #include <Wire.h>
6   #include <SPI.h>
7   #include <Adafruit_Sensor.h>
8   #include <Adafruit_BMP280.h>
9   #include <ArduinoJson.h>
10
```

Figura 22. Código Placa 3 (sección "includes")

En la sección “includes” de un programa de C++ es donde se importan las bibliotecas necesarias en el programa. En este caso, se importan varias bibliotecas que proporcionan funciones y herramientas indispensables para el proyecto. Las bibliotecas incluidas son:

1. #include <DHTesp.h>: Esta biblioteca se utiliza para usar el sensor DHT11, encargado de medir la temperatura y la humedad.
2. #include <PubSubClient.h>: Esta biblioteca permite la conexión entre la placa ESP8266 y el servidor MQTT, pudiendo así comunicarse de una manera sencilla.
3. #include <ESP8266WiFi.h>: Esta biblioteca se usa específicamente para configurar la conexión Wi-Fi de la placa ESP8266.
4. #include <DNSServer.h>: Esta biblioteca ayuda en la resolución de nombres de dominio (DNS) en la red, lo que es importante cuando se trabaja con conexiones a través de Internet.

5. #include <Wire.h> y #include <SPI.h>: Estas bibliotecas son fundamentales para la comunicación entre dispositivos a través de los buses I2C y SPI, respectivamente.
6. #include <Adafruit_Sensor.h>: Esta biblioteca es parte del ecosistema de sensores de Adafruit y proporciona una interfaz común para trabajar con varios sensores.
7. #include <Adafruit_BMP280.h>: Esta biblioteca se utiliza para interactuar con el sensor de presión atmosférica BMP280, que mide la presión y la temperatura atmosférica.
8. #include <ArduinoJson.h>: Esta biblioteca es útil para manejar datos en formato JSON, lo que facilita la manipulación y el intercambio de datos estructurados.

Tras la importación de las bibliotecas necesarias para la realización del proyecto, se pasa a la asignación de pines que se utilizan para conectar los sensores a la placa ESP8266. La definición sería la siguiente:

```
12  // Definimos el pin al cual estan conectados los sensores
13  #define DHTPIN D7          //DHT11
14  #define sensorLDR A0       //LDR
15  #define BMP_SCK 13         //BMP280
16  #define BMP_MISO 12        //BMP280
17  #define BMP_MOSI 11        //BMP280
18  #define BMP_CS 10          //BMP280
```

Figura 23. Código Placa 3 (asignación de pines)

1. #define DHTPIN D7: Se define el pin D7 como el pin al que está conectado el sensor DHT11. Esto permitirá que el programa sepa dónde buscar los datos del sensor de temperatura y humedad.
2. #define sensorLDR A0: Se define el pin A0 como el pin al que está conectado el sensor de luz LDR. De esta manera, el programa sabrá desde qué pin leer los datos de luz ambiental.
3. #define BMP_SCK 13: Aquí, se define el pin 13 como el pin de reloj (SCK) para la comunicación con el sensor BMP280, que mide la presión atmosférica y la temperatura.
4. #define BMP_MISO 12: Se define el pin 12 como el pin de datos de entrada (MISO) para la comunicación con el sensor BMP280.
5. #define BMP_MOSI 11: El pin 11 se define como el pin de datos de salida (MOSI) para la comunicación con el sensor BMP280.

6. #define BMP_CS 10: Aquí, se define el pin 10 como el pin de selección de chip (CS) para la comunicación con el sensor BMP280.

Sin estas definiciones, el programa no podría comunicarse con los sensores en los pines correctos y no podría recopilar los datos necesarios para el proyecto.

Lo siguiente es poder interactuar con los sensores por lo que es necesario la creación de instancias, en este caso de dos clases diferentes:

```
21  // Creamos una instancia de la clase DHTesp
22  DHTesp dht;
23  Adafruit_BMP280 bme;
```

Figura 24. Código Placa 3 (creación instancias)

1. DHTesp dht;: Se crea una instancia de la clase DHTesp. se utiliza para interactuar con el sensor DHT11.
2. Adafruit_BMP280 bme;: Se crea una instancia de la clase Adafruit_BMP280. Se utiliza para interactuar con el sensor BMP280.

Se necesita configurar los parámetros de la red Wi-Fi y la conexión MQTT de la siguiente manera:

```
25  // Configuración de red WiFi
26  const char* SSID = "            ";
27  const char* PASSWORD = "            ";
28  const char* IDPLACA = "Placa3";
29
30  // Configuración de conexión MQTT
31  const char* BROKER = "192.168.0.20";
32  const char* TOPIC_DHT11 = "datos/dht11";
33  const char* TOPIC_LDR = "datos/ldr";
34  const char* TOPIC_BMP280 = "datos/bmp280";
35
36
37  WiFiClient wifiClient;
38  PubSubClient mqttClient(wifiClient);
```

Figura 175. Código Placa 3 (configuración WiFi y conexión MQTT)

La configuración de estos consiste en definir el nombre de la red Wi-Fi a la que se conectará la placa ESP8266 con su correspondiente contraseña y se define un identificador único para la placa, en este caso la placa 3 para poder identificarla en la red.

Para la conexión con el servidor MQTT, se define la dirección del servidor al que se conectará la placa. En el caso del presente proyecto, el servidor MQTT Mosquitto se encuentra alojado en una Raspberry Pi, por tanto, su dirección es la asignada por la misma.

Posteriormente se definen los nombres de los topics MQTT que se utilizarán para publicar datos del sensor DHT11, el sensor de luz LM393 (LDR) y el sensor BMP280.

Y finalmente, se crean las instancias necesarias para la conexión Wi-Fi (WiFiClient) y la que se utilizará para la comunicación MQTT, para conectarse el servidor y publicar o suscribirse a temas específicos (PubSubClient mqttClient).

En esta parte del código, se encuentra la función setup(), que es parte del ciclo de vida de un programa de Arduino y se ejecuta una sola vez al inicio del programa.

```
void setup()
{
  Serial.begin(9600);
  setupWiFi();
  setupMQTT();
  dht.setup(DHTPIN, DHTesp::DHT11);
  pinMode(sensorLDR, INPUT);
  Serial.println(F("BMP280 test"));
  if (!bme.begin(0x76))
  {
    Serial.println("Could not find a valid BMP280 sensor, check wiring!");
  }
  timeClient.begin();
}
```

Figura 186. Código Placa 3 (void setup)

Con el Serial.begin(9600) se inicia la comunicación serial a una velocidad de 9600 baudios, lo que permite enviar y recibir datos a través este puerto. Posteriormente, se llama al setupWiFi() para configurar la conexión Wi-Fi en la placa ESP8266 y configurar el sensor DHT11 anteriormente definido (Figura 18).

Con el pinMode(sensorLDR, INPUT), se establece el pin sensorLDR como entrada para poder leer datos.

```
53  void loop() {
54    mqttClient.loop();
55
56    float temperatura = dht.getTemperature();  //Obtengo lectura de temperatura y humedad del sensor DHT11
57    float humedad = dht.getHumidity();
58    int intensidadLDR = analogRead(sensorLDR);
59    float presion = bme.readPressure();
60    float altura = bme.readAltitude();
61
62
63    if (isnan(temperatura) || isnan(humedad)) {  //Si los valores no son números muestro error
64      Serial.println("Error al leer el sensor DHT11!");
65      return;
66    }
67
68
69    //TEMPERATURA
70
71    StaticJsonDocument<200> jsonDoc;
72    jsonDoc["temperatura"] = temperatura;
73    jsonDoc["humedad"] = humedad;
74    jsonDoc["idPlaca"] = IDPLACA;
75
76    char jsonBuffer[200];
77    serializeJson(jsonDoc, jsonBuffer);
78
79    Serial.println(jsonBuffer);
80
81    mqttClient.publish(TOPIC_DHT11, jsonBuffer);  //Y publico en el servidor MQTT a través de los topics definidos
82
```

Figura 197. Código Placa 3 (void loop parte 1)

```
83    //BMP280
84    StaticJsonDocument<200> bmpJsonDoc;
85    bmpJsonDoc["presion"] = presion;
86    bmpJsonDoc["altura"] = altura;
87    bmpJsonDoc["idPlaca"] = IDPLACA;
88
89    char bmpJsonBuffer[200];
90    serializeJson(bmpJsonDoc, bmpJsonBuffer);
91
92    Serial.println(bmpJsonBuffer);
93
94    mqttClient.publish(TOPIC_BMP280, bmpJsonBuffer);
95
96
97    //LDR
98    Serial.println(intensidadLDR);
99    if(intensidadLDR >= 400){
100     StaticJsonDocument<200> LDRJsonDoc;
101     LDRJsonDoc["encendido"] = 1;
102     LDRJsonDoc["idPlaca"] = IDPLACA;
103
104     char LDRJsonBuffer[200];
105     serializeJson(LDRJsonDoc, LDRJsonBuffer);
106
107
108     Serial.println(LDRJsonBuffer);
109
110     // Publicación en el servidor MQTT
111     mqttClient.publish(TOPIC_LDR, LDRJsonBuffer);
112   }
113
114   delay(2000);  //Espera de 2s para siguiente iteración
115 }
```

Figura 208. Código Placa 3 (void loop parte 2)

En la Figura 19 y Figura 20 se encuentra la función loop(), esta función se ejecuta de manera repetitiva en un ciclo infinito mientras el programa se encuentre en funcionamiento. Las acciones principales que realiza son las siguientes: con el mqttClient.loop() se permite que el cliente MQTT procese cualquier mensaje entrante o realice tareas internas necesarias para mantener la conexión MQTT activa; posteriormente se definen las variables en las que se van a guardar los datos obtenidos de los distintos sensores y se verifica si estas lecturas son números válidos.

Se crea un documento JSON para almacenar estas lecturas, añadiendo el identificador de la placa y se serializa este documento en un búfer JSON para mostrarlo en el monitor series. Este búfer se publica en el servidor MQTT definiendo el topic para cada uno de los sensores.

Finalmente se introduce un retraso de 2 segundos antes de repetir el ciclo completo.

```
117  void setupWiFi() {  //EStablezco conexión WiFi
118    Serial.println("Conectando a la red WiFi...");
119    WiFi.mode(WIFI_STA);
120    WiFi.begin(SSID, PASSWORD);
121
122    while (WiFi.status() != WL_CONNECTED) {
123      delay(1000);
124      Serial.println("Intentando conectarse...");
125    }
126
127    Serial.println("Conexión WiFi establecida!");
128    Serial.print("Dirección IP asignada: ");  //Muestro dirección IP asignada
129    Serial.println(WiFi.localIP());
130  }
```

Figura 219. Código Placa 3 (void setupWiFi)

La función setupWiFi() se encarga de establecer la conexión Wi-Fi de la placa a una red Wi-Fi específica. En ella se imprime un mensaje sobre la conexión Wi-Fi en el monitor series, se configura la placa ESP8266 para conectarse a una red Wi-Fi existente, se inicia el proceso de conexión con el nombre de la red (SSID) y contraseña (PASSWORD) proporcionados y se espera hasta que la conexión sea exitosa, con intervalos de un segundo entre intentos.

Finalmente se muestra un mensaje de "Conexión WiFi establecida" cuando la conexión se haya realizado con éxito y se muestra la dirección IP asignada a la placa ESP8266 en la red Wi-Fi.

```
132  void setupMQTT() {  //Establezco conexión con MQTT
133    mqttClient.setServer(BROKER, 1883);
134    mqttClient.setCallback(callback);
135
136    while (!mqttClient.connected()) {
137      Serial.println("Conectando al servidor MQTT...");
138      if (mqttClient.connect(IDPLACA, "saulpi", "sAuLpI1234")) {
139        Serial.println("Conexión MQTT establecida!");
140      } else {
141        Serial.print("Error al conectar al servidor MQTT, código de error: ");
142        Serial.println(mqttClient.state());
143        delay(2000);
144      }
145    }
146
147    mqttClient.subscribe(TOPIC_DHT11);  //configuro la suscripción a los topic
148    mqttClient.subscribe(TOPIC_LDR);  //configuro la suscripción a los topics
149    mqttClient.subscribe(TOPIC_BMP280);  //configuro la suscripción a los topi
150  }
```

Figura 3022. Código Placa 3 (void setupMQTT)

La función setupMQTT() se encarga de establecer la conexión con el servidor MQTT utilizando los parámetros necesarios. En ella se configura la dirección IP y puerto del servidor MQTT al que se conectará y se define una función de callback para procesar mensajes MQTT entrantes.

Se intenta conectar la placa ESP8266 al servidor MQTT usando credenciales y el identificador de placa. Si la conexión MQTT se establece correctamente, se muestra un mensaje de éxito, en el caso contrario se vuelve a intentar.

Se configura la suscripción a temas MQTT específicos para recibir datos, garantizando que la conexión MQTT esté activa antes de continuar y se habilita la comunicación bidireccional con el servidor MQTT para enviar y recibir datos.

```
153  void callback(char* topic, byte* payload, unsigned int length)
154    Serial.print("Mensaje recibido en el topic: ");
155    Serial.println(topic);
156
157    String message = "";
158
159    for (int i = 0; i < length; i++) {
160      message += (char)payload[i];
161    }
162
163    Serial.print("Contenido del mensaje: ");
164    Serial.println(message);
165  }
```

Figura 31. Código Placa 3 (void callback)

Esta función se activa cuando llega un mensaje MQTT a la placa ESP8266 en uno de los temas a los que está suscrita.

Se imprime el nombre del tema MQTT en el que se recibió el mensaje y se crea una cadena de caracteres llamada message para almacenar el contenido del mensaje. Finalmente, se recorre el array de bytes payload y se agrega cada byte como un carácter a la cadena message y se imprime en el monitor series.

Esta función es esencial para comprender y depurar los datos que se envían y reciben a través del protocolo MQTT, facilitando la visualización y registro de los mensajes MQTT entrantes para su análisis y seguimiento.

3.2. Presupuesto

A continuación, se presenta el presupuesto final del proyecto, el cual ha sido una parte esencial para su desarrollo y ejecución.

GASTOS

COMPONENTE	UNIDADES	PRECIO UNIDAD	PRECIO TOTAL
MEDIDOR DE DISTANCIAS POR ULTRASONIDOS HC-SR04 PARA ARDUINO	1	2,13€	2,13€
MODULO SENSOR DE TEMPERATURA Y HUMEDAD DHT11	3	2,66€	7,98€
SENSOR LDR FOTOSENSIBLE	1	0,86€	0,86€
SENSOR DE GASES MQ-5, SENSIBLE AL GAS NATURAL	1	2,77€	2,77€
CABLES DUPONT 10CM, M-H, 40 UDS	1	1,36€	1,36€
DETECTOR DE OBSTÁCULOS POR INFRARROJOS	1	0,82€	0,82€
MODULO BMP280 PARA MEDIR LA PRESIÓN BAROMÉTRICA	1	2,16€	2,16€
SENSOR DE GASES MQ-135	1	3,37€	3,37€
PLACA WI-FI ESP-8266	4	4,24€	16,99€
RASPBERRY PI 3 MODEL 2015	1	61,62€	61,62€
PLACAS PCB	2	3,49€	6,99€
TOTAL			107,05€

Figura 32. Gastos del proyecto.

Capítulo 4 INTEGRACIÓN DE DISPOSITIVOS CON NODE-RED

Esta capítulo presenta la fase de desarrollo de la aplicación en Node-RED, donde se explicará cómo los datos se integran y se convierten en acciones significativas para mejorar la comodidad, la eficiencia y la seguridad del hogar.

4.1. Introducción a Node-RED en la integración de dispositivos

A través de la interfaz gráfica del programa, se ha desarrollado un conjunto de cuatro flujos fundamentales para gestionar diferentes aspectos del sistema domótico. A continuación, se detallarán estos flujos uno por uno, destacando su funcionalidad y contribución al proyecto.

El primer flujo, denominado “Flujo de Históricos”, se encarga de la recopilación y el registro de datos históricos de los sensores y dispositivos del sistema. Este flujo usa nodos para almacenar en una base de datos MongoDB, permitiendo así la posterior consulta y análisis de información histórica.

El siguiente flujo, llamado “Flujo MQTT”, se encarga de gestionar la comunicación entre los dispositivos IoT y el servidor MQTT. Los nodos de este flujo permiten la recepción y el envío de datos entre los sensores y actuadores y el servidor MQTT.

El tercer flujo, denominado “Flujo de Telegram”, habilita la interacción remota con el sistema a través de la plataforma de mensajería Telegram. Este flujo permite recibir comandos y notificaciones desde la aplicación de Telegram.

El último flujo, llamado “Flujo de Regleta Inteligente”, se enfoca en la gestión de una regleta de enchufes inteligente. Este flujo se encarga de controlar individualmente cada toma de corriente de la regleta, permitiendo encender y apagar dispositivos conectados de manera remota.

4.2. Explicación de flujos de Node-RED

Antes de comenzar con la explicación de cada uno de los flujos empleados, el propósito general de la recopilación de los datos y el uso de históricos es el de poder detectar anomalías en el funcionamiento de la casa, como cambios abruptos de temperatura o alertas de los sensores MQ que podrían indicar posibles fugas de gas o incendios. Además, permite consultar la temperatura y humedad actuales, así como los registros históricos, actuando como una estación meteorológica interna. También detecta accesos no autorizados o sospechosos en el hogar, brindando mayor seguridad. Además, posibilita el control remoto de dispositivos conectados a la regleta inteligente, lo que facilita encender o apagar dispositivos desde cualquier lugar. Por último, ayuda a evitar el desperdicio de energía al notificar si alguna luz se ha quedado encendida en una habitación.

Comenzando con la explicación del Flujo de Históricos, este se encarga de recopilar y registrar datos históricos de los sensores y dispositivos del sistema. Se encuentra formado por los nodos “mongodb in” que se encargan de buscar datos en la base de datos. Cada uno de estos se conecta a una colección específica en la base de datos, como "ultrasonido," "mq5," "mq135," y "ldr" y recuperan los datos almacenados en esas colecciones para su posterior visualización.

El nodo “clock toolbar” se encarga de mostrar un reloj en la interfaz de usuario. Este reloj se actualiza cada segundo y muestra la hora actual.

Se implementa un nodo “inject” que inyecta un mensaje vacío cada 10 segundos en el flujo. Este mensaje se encarga de activar la búsqueda en las colecciones de MongoDB.

Los nodos de detecciones sensor ultrasonido, detecciones sensor mq5, detecciones sensor mq135 y detecciones sensor ldr se utilizan para mostrar los resultados de las búsquedas de datos, presentando los datos en forma de lista en la interfaz de usuario.

El uso de los nodos “function” son usados en el caso de la consulta placa 3 y placa 4 para enviar la consulta a la base de datos y obtener los datos de una placa específica. En el caso de los históricos se toman los datos de temperatura, humedad, presión o altitud de la base de datos, y los transforma en un formato adecuado para su visualización en gráficos.

Finalmente, los nodos “chart” se encargan de mostrar los gráficos correspondientes a los datos de temperatura y humedad de las placas 3 y 4. Gracias a la capacidad de los nodos "chart" para traducir los datos en gráficos visuales, se pueden identificar tendencias, patrones y variaciones con facilidad, lo que resulta muy útil en el monitoreo en tiempo real y en el análisis de los datos recopilados.

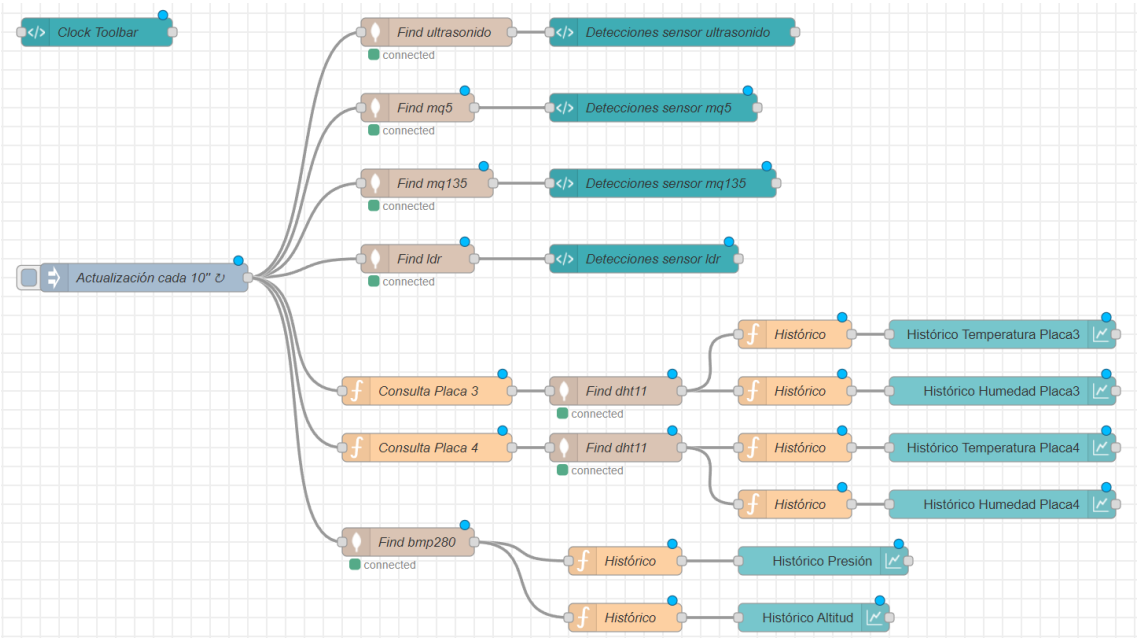


Figura 33. Flujo de Históricos

El flujo de MQTT está compuesto principalmente por los nodos “MQTT in”, este nodo se utiliza para recibir mensajes MQTT. Está configurado para escuchar los distintos topics, siendo estos: datos/ultrasonido, datos/mq5, datos/mq135, datos/ldr, datos/dht11 y datos/bmp280 con un nivel de calidad de servicio (QoS) de 2.

Los nodos “change” usados, como el de detección o encendido se utilizan para modificar el mensaje que pasa a través del flujo. En este caso, estan configurados para establecer el valor del campo "payload" en "true" cuando recibe un mensaje.

En el caso de este flujo, se necesita guardar los datos en la base de datos, por lo que se usa los nodos “mongodb out”. Estos datos se almacenan en la base de datos "tfg-saul" en el servidor MongoDB que se ejecuta en "localhost".

Los nodos “ui_led” muestran un indicador LED en el panel de control de Node-RED. El LED se utiliza para mostrar el estado de detección. Si el mensaje contiene un valor "true", el LED se iluminará en verde; si contiene "false", se iluminará en rojo.

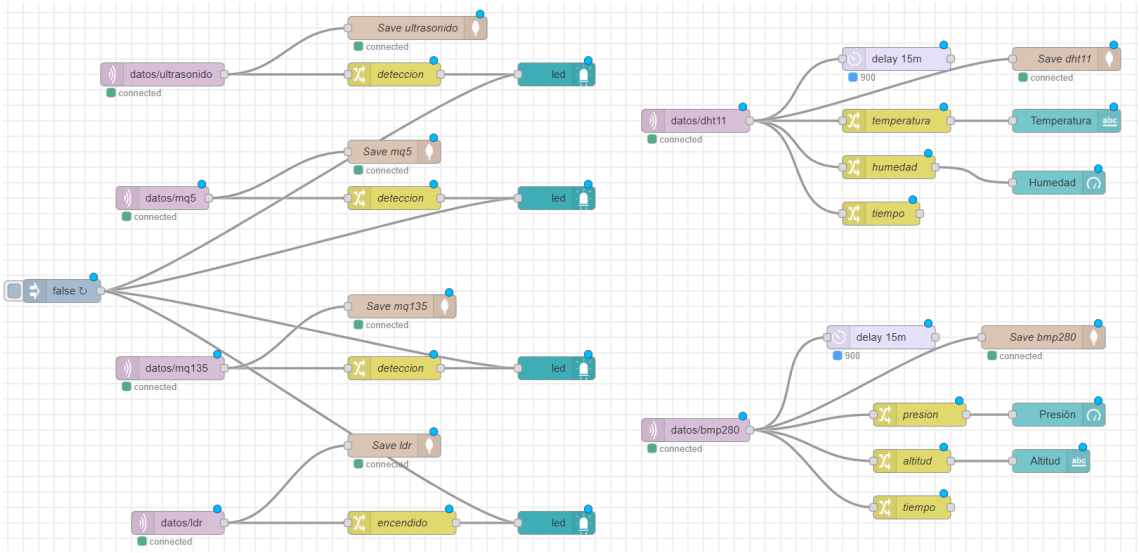


Figura 34. Flujo de MQTT

Para la realización del flujo de Telegram, se debe instalar previamente la librería “node-red-contrib-telegrambot” con la que se podrá hacer uso de los nodos dedicados a Telegram.

El propósito fundamental de recibir información a través de Telegram es el brindar al usuario un mayor control y comodidad en la gestión del sistema. Estas notificaciones permiten una interacción eficiente y oportuna con el sistema, mandando alertas sobre el estado de la casa, como cambios inesperados de temperatura o accesos no autorizados. Lo más importante es que Telegram capacita al usuario para tomar acciones desde fuera del hogar, ampliando de manera significativa las posibilidades de control más allá de la interfaz a veces compleja o poco flexible del dashboard de Node-RED. Esta integración con Telegram se traduce en un canal de comunicación ágil y accesible que simplifica la supervisión y gestión del sistema desde cualquier ubicación, enriqueciendo la experiencia del usuario de manera integral y versátil.

Los nodos de entrada son esenciales para interactuar con Telegram, para ello deben estar configurados de la siguiente manera:

1. Bot de Telegram: Primero, se debe crear un bot de Telegram siguiendo las instrucciones proporcionadas por el BotFather en Telegram. Este proporciona un token único para cada bot.
2. Configuración del Nodo: En Node-RED, se configura el nodo de entrada de Telegram utilizando el token del bot. Este nodo estará escuchando los mensajes y comandos entrantes en un chat específico o en un grupo de Telegram.
3. Comandos Personalizados: Se le asocia comandos personalizados a este nodo, como "/dht11_Placa3", "/dht11_Placa4", "/ldr_Placa3", etc., según los sensores y dispositivos se quieran controlar.

Después de recibir un comando en el nodo de entrada de Telegram, se usan nodos de procesamiento para analizar el comando y determinar qué acción tomar.

El nodo "function" llamado CONSULTA, se encarga de construir un objeto msg que se utilizará para realizar una consulta a la base de datos MongoDB. Establece el ID de la placa como "Placa3", limita la consulta a un solo resultado y ordena los resultados por tiempo en orden descendente (el más reciente primero).

Posteriormente, en el nodo "function" de Mensaje Respuesta, se extrae los datos del resultado de la consulta a la base de datos y se colocan en un formato legible. Luego, se coloca este mensaje en una propiedad llamada "content" dentro de msg.mensaje.

Por último, se utiliza un nodo de salida de Telegram que selecciona al bot "tfgsaul_bot" y envía el mensaje previamente formateado.

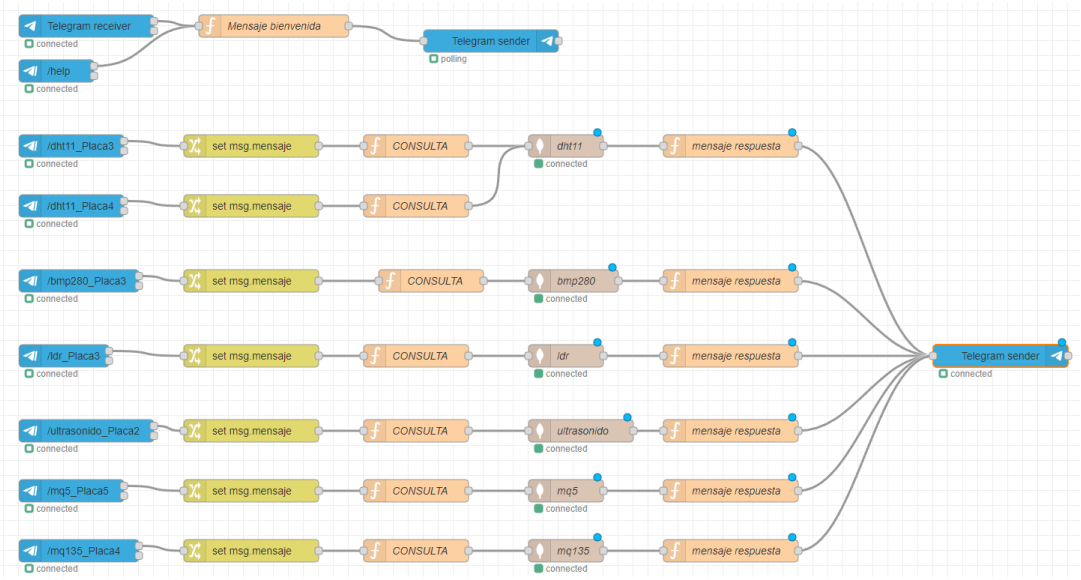


Figura 35. Flujo de Telegram

En el flujo de Regleta Inteligente se usan los nodos de Telegram previamente explicados unidos a unos nodos “function” programados para poder encender y apagar cambiando el contenido del msg.payload entre verdadero y falso.

Para mostrar un botón en la interfaz gráfica se hace uso del nodo “switch” y un led que cambia su color a verde si se encuentra encendido y rojo si está apagado.

Finalmente, con un nodo “MQTT out” se envía la señal a la regleta.

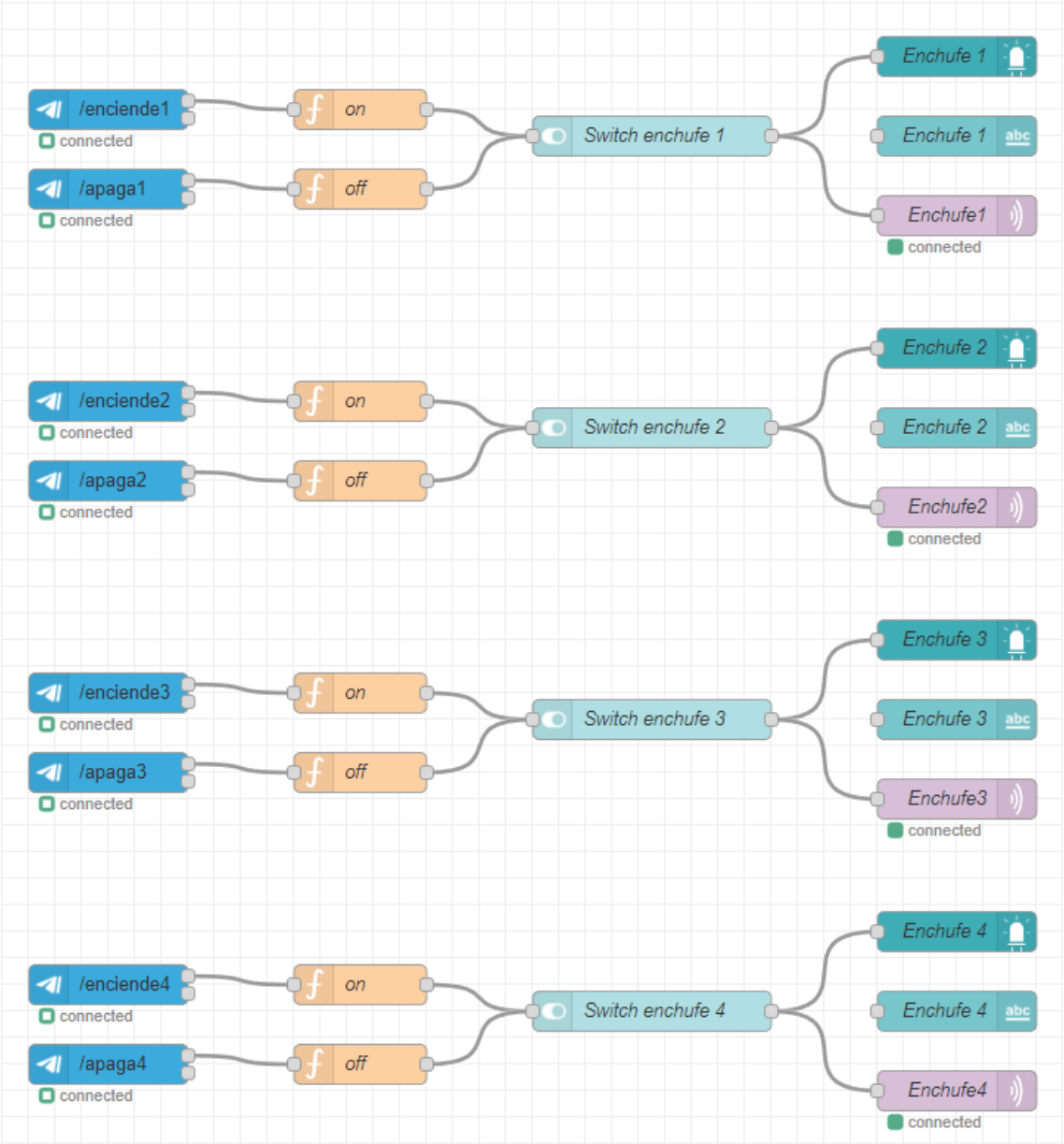


Figura 36. Flujo de Regleta Inteligente.

Capítulo 5 INTERFACES GRÁFICAS

En este apartado, se presenta la implementación de interfaces gráficas en el contexto de este proyecto. Se analizan dos herramientas fundamentales: el dashboard de Node-RED y la plataforma Telegram. A través de estas interfaces, el usuario puede supervisar y controlar de manera efectiva todos los sensores y dispositivos, lo que mejora la capacidad de gestión y la experiencia de usuario en la automatización del hogar.

5.1. Dashboard de Node-RED.

El dashboard de Node-RED está diseñado para interactuar y controlar dispositivos y flujos de datos en tiempo real.

Es una interfaz intuitiva en la que se permite realizar todo lo anterior sin conocimientos de programación avanzada. Es altamente personalizable, permitiendo a los desarrolladores diseñar paneles de control a medida, ofreciendo una variedad de widgets y nodos predefinidos para representar datos y controlar dispositivos, facilitando la comunicación en tiempo real y el soporte móvil para el acceso desde dispositivos móviles. Garantiza la seguridad con opciones de protección y es altamente versátil al ser compatible con diversos dispositivos y servicios, lo que simplifica la integración.

El dashboard implementado en el presente proyecto se divide en cinco pestañas: "Placa 4 y 5", "Placa 2", "Placa 3", "Regleta inteligente" e "Históricos". En cada una de estas pestañas se visualizan elementos distintos y necesarios para el control de los distintos sensores usados en el proyecto. Además de poder visualizar los valores de los sensores en tiempo real, se podrá consultar los datos guardados en la base de datos MongoDB.

A continuación, se muestran las distintas pestañas con su correspondiente explicación.

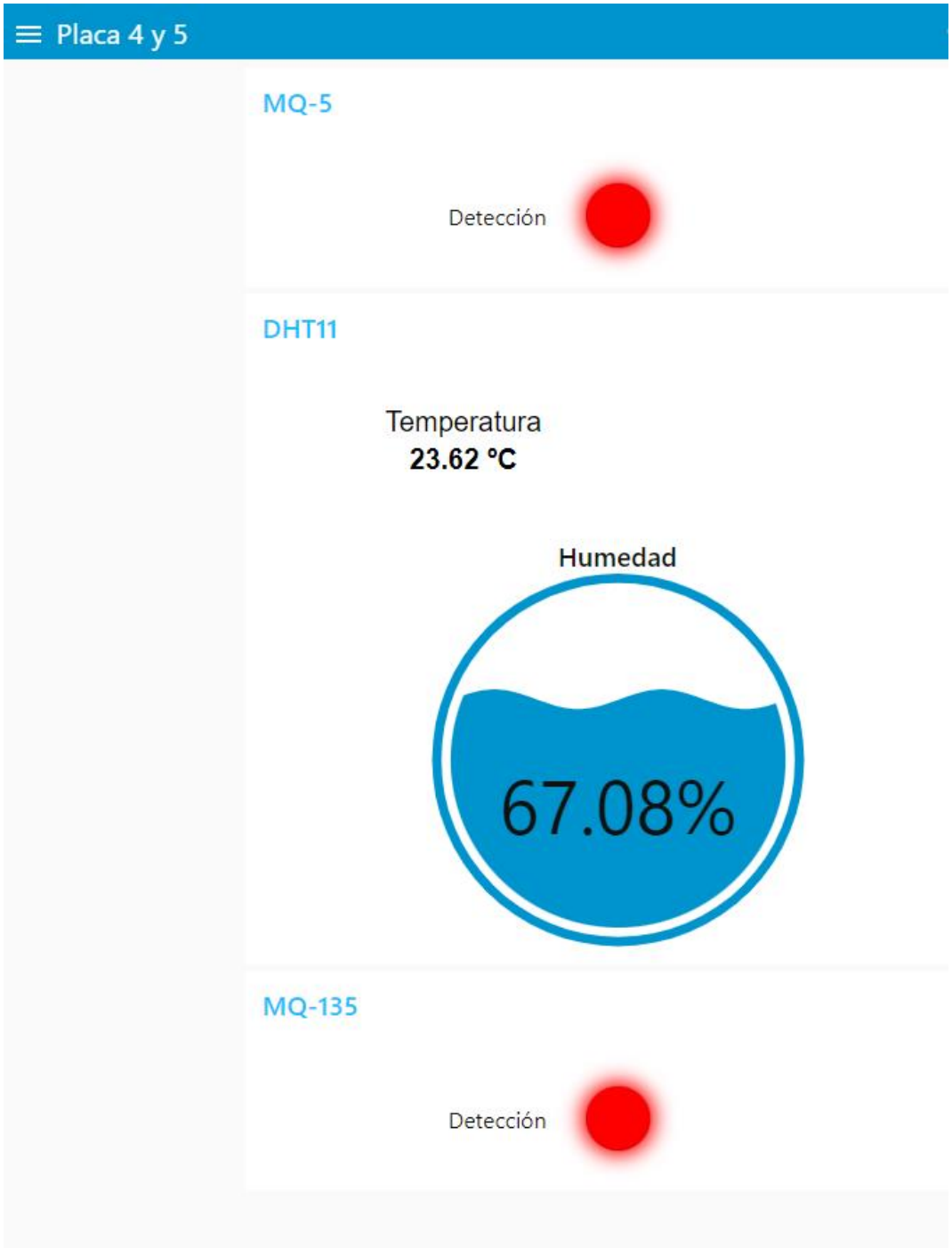


Figura 36. Dashboard Node-RED (Placa 3 y 4)

La primera pestaña de este dashboard se corresponde con la “Placa 3 y Placa 4”, en ella podemos encontrar los valores de temperatura y humedad del sensor DHT11 a tiempo real, junto con dos leds programados para que se cambien a color verde en el momento que los sensores MQ-5 y MQ-135 detecten una posible fuga de gas natural y humo.

Figura 37. Dashboard Node-RED (Placa 2)

La segunda pestaña está dedicada a "Placa 2" y está diseñada para mostrar la programación de un LED que cambia a color verde cuando el sensor de movimiento HC-SR04 detecta un cambio en la situación.

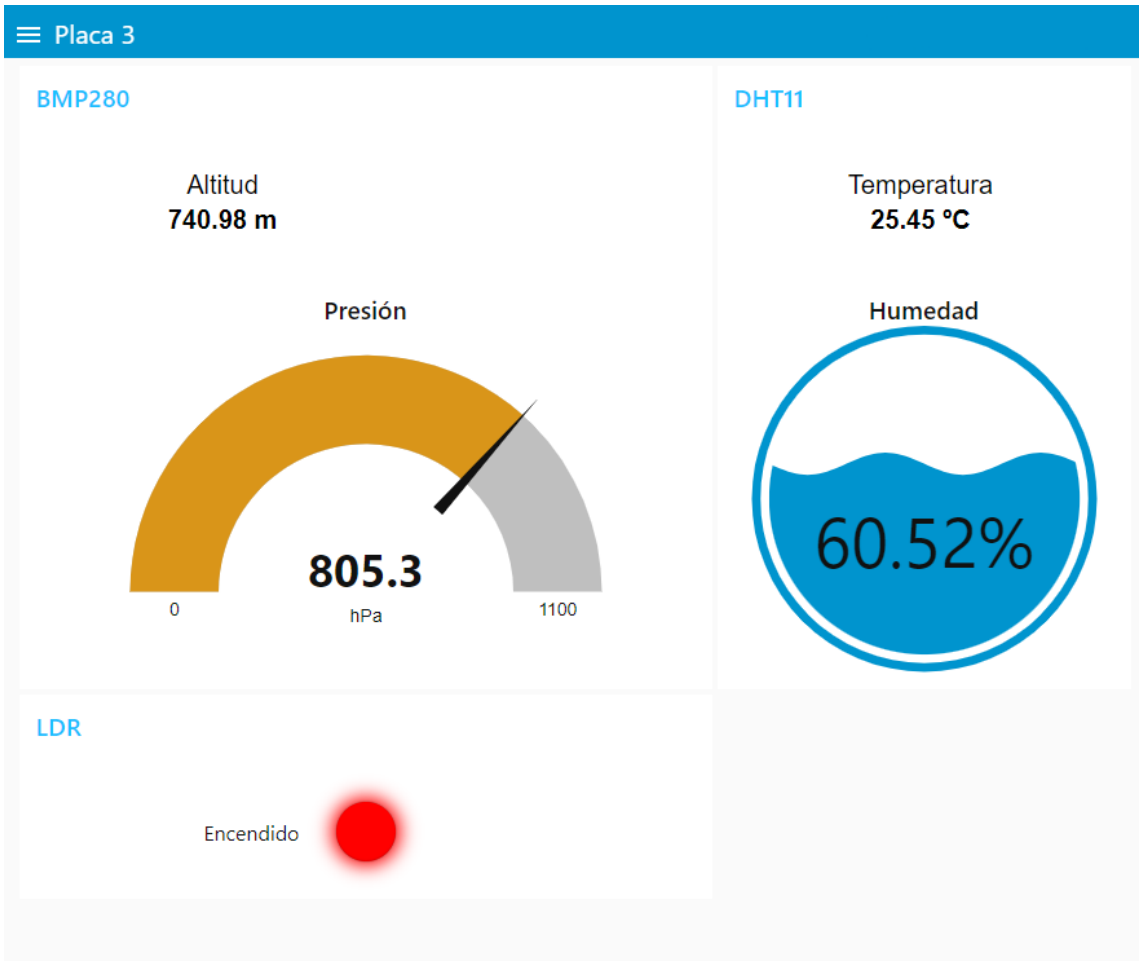


Figura 37. Dashboard Node-RED (Placa 3)

En la tercera pestaña, se encuentran los elementos correspondientes a "Placa 3". Dado que esta placa está equipada con un sensor BMP280, DHT11 y un sensor de luz LM-393, se pueden monitorear en tiempo real la temperatura y humedad, la altitud y la presión atmosférica. Además, proporciona información sobre la detección de luz o presencia en el entorno donde se encuentra ubicada la placa en ese momento.

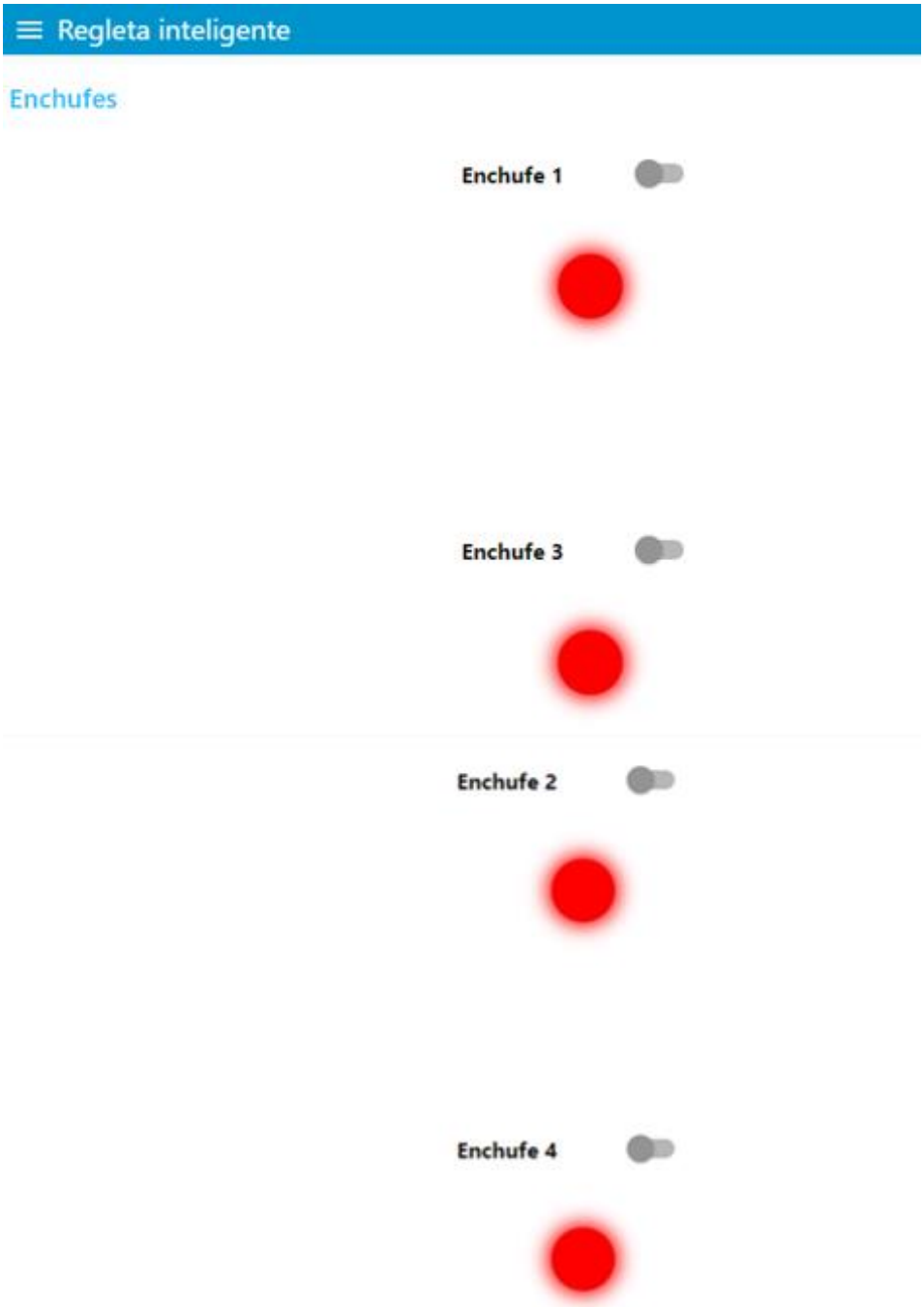


Figura 38. Dashboard Node-RED (Regleta inteligente)

En la cuarta pestaña, que está relacionada con la "Regleta inteligente", se presentan cuatro interruptores que permiten controlar individualmente los cuatro enchufes de la regleta. Además, se incluyen cuatro LEDs indicadores que muestran claramente si cada enchufe está en estado de encendido (verde) o apagado (rojo). Esto brinda un control eficaz sobre los dispositivos conectados a la regleta.

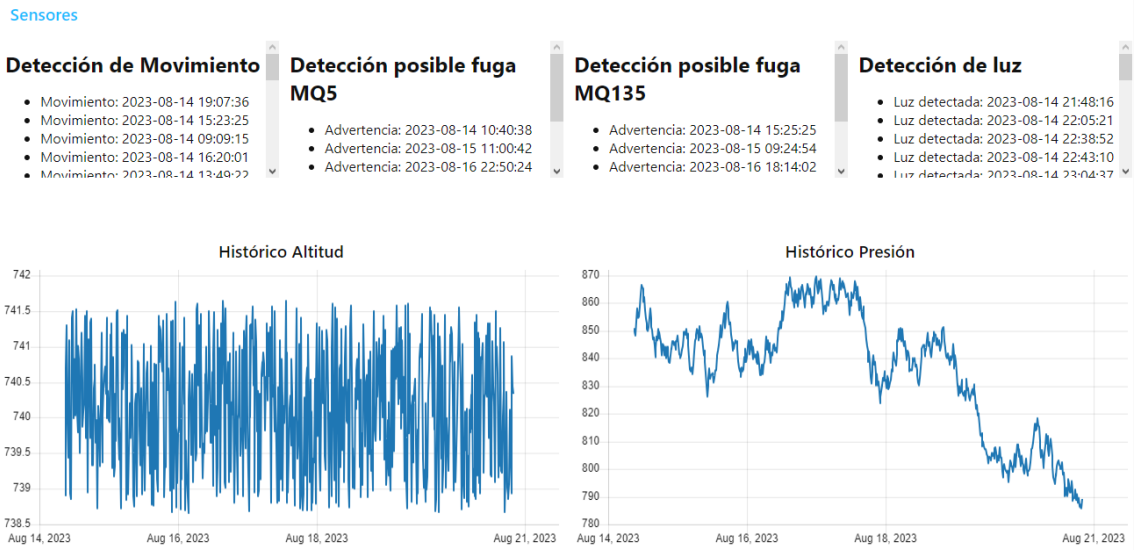


Figura 40. Dashboard Node-RED (Históricos Parte 1)

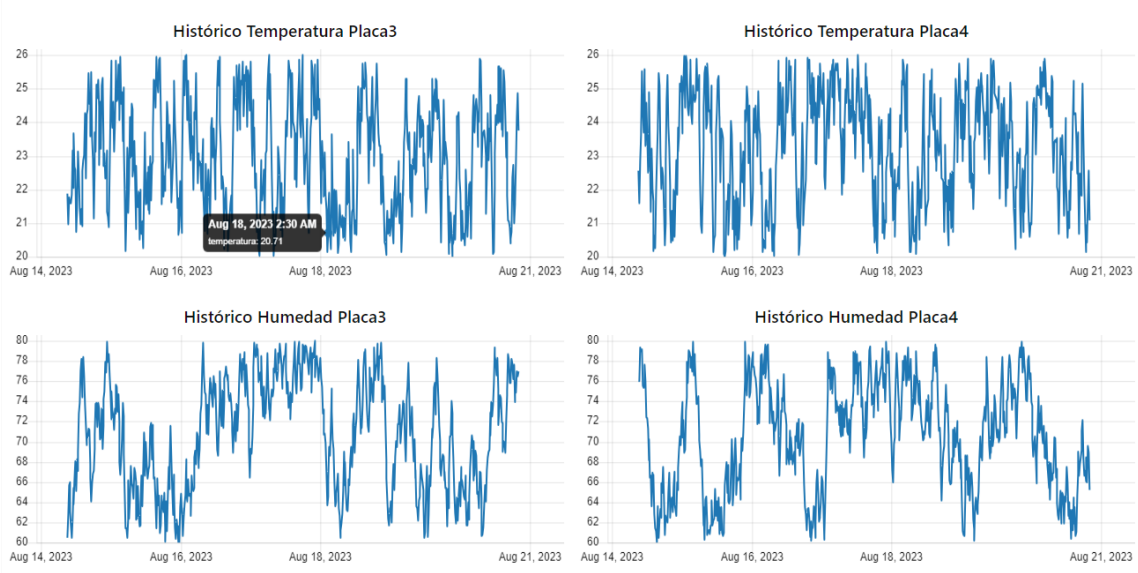


Figura 39. Dashboard Node-RED (Históricos Parte 2)

En la quinta y última pestaña, denominada "Históricos", se almacenan registros de todas las detecciones realizadas por los sensores. Estos registros incluyen la marca de tiempo precisa de cada detección, ya sea relacionada con movimiento, fugas, luz, altitud, presión, temperatura o humedad. Además, los datos históricos de altitud, presión, temperatura y humedad se presentan en gráficos interactivos que permiten analizar detenidamente la evolución en el tiempo, simplemente pasando el cursor sobre ellos para obtener detalles específicos en cualquier momento.

5.2. Telegram.

A través de Telegram, se ha creado una interfaz de control altamente efectiva que permite al usuario interactuar con sensores, dispositivos y sistemas desde cualquier lugar, todo ello con la comodidad de una aplicación de mensajería instantánea.

Después de haber creado el bot de Telegram, cuya configuración se ha detallado en la sección 4.2, y de haber configurado los nodos correspondientes en Node-RED, se ha establecido un componente central en el proyecto. Este bot de Telegram desempeña un papel fundamental al permitir no solo la interacción con los sensores, sino también la recepción de datos de los mismos y la capacidad de controlar su funcionamiento, incluyendo la capacidad de encenderlos o apagarlos.

A continuación, se presenta en detalle el funcionamiento de este bot de Telegram, incluyendo una lista completa de los comandos que ofrece y ejemplos ilustrativos de cómo se utilizan.

Cuando un usuario inicia una conversación con el bot, simplemente enviando un mensaje como "/start" para iniciar la interacción, el bot se activa y responde con un mensaje de bienvenida que ha sido predefinido. En este mensaje de bienvenida, el bot presenta de manera clara y concisa los diversos comandos disponibles, lo que permite a los usuarios solicitar y visualizar datos específicos, así como controlar el encendido y apagado de los enchufes de las regletas de manera conveniente.

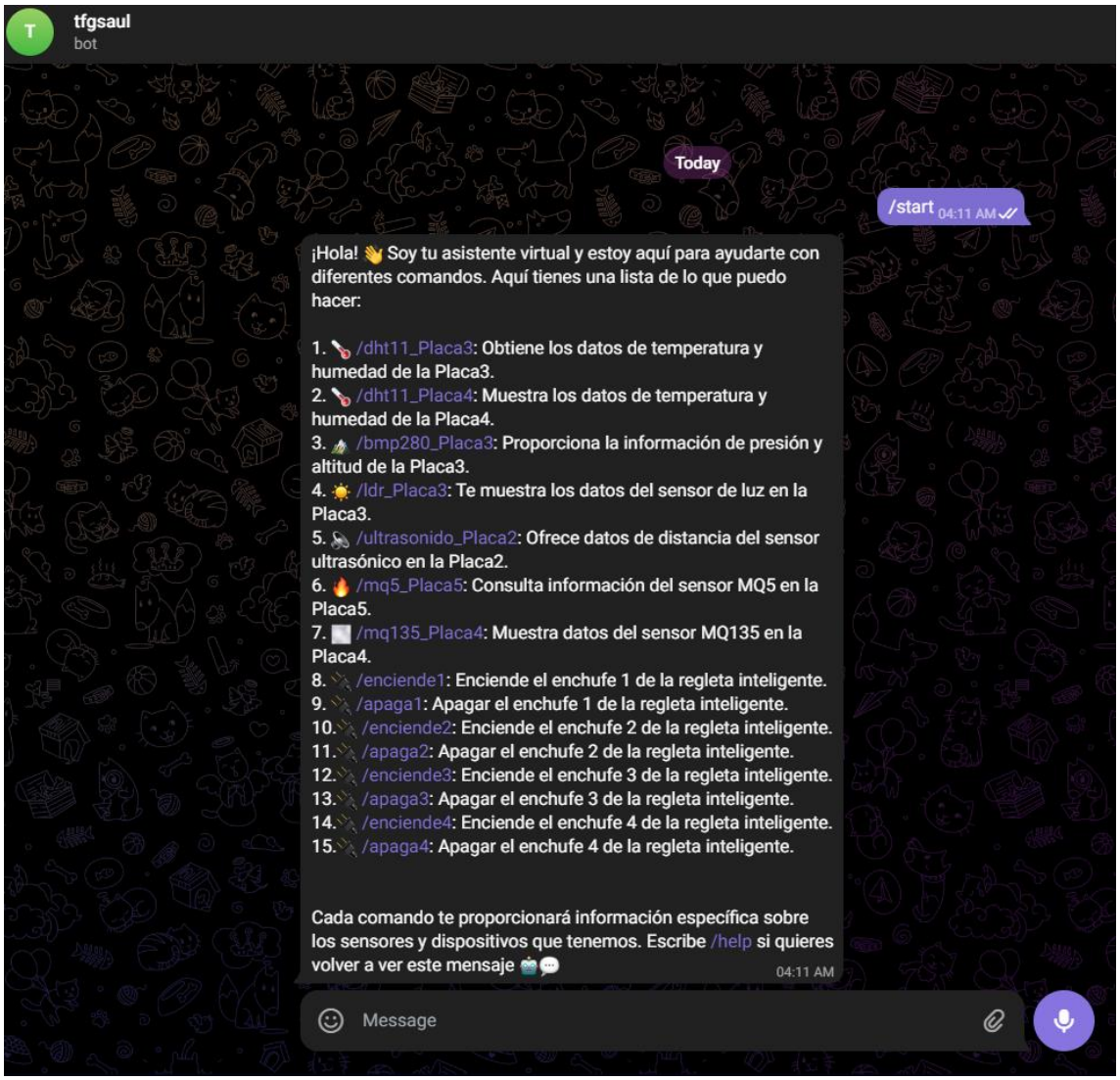


Figura 40. Comandos Bot de Telegram.

Los comandos pueden activarse de dos maneras: escribiéndolos directamente en el chat o haciendo clic en ellos para ejecutar la acción deseada. Cuando se realiza cualquiera de estas dos acciones, el bot proporciona las respuestas, buscando en la base de datos la última actualización de la consulta que se le esté haciendo.

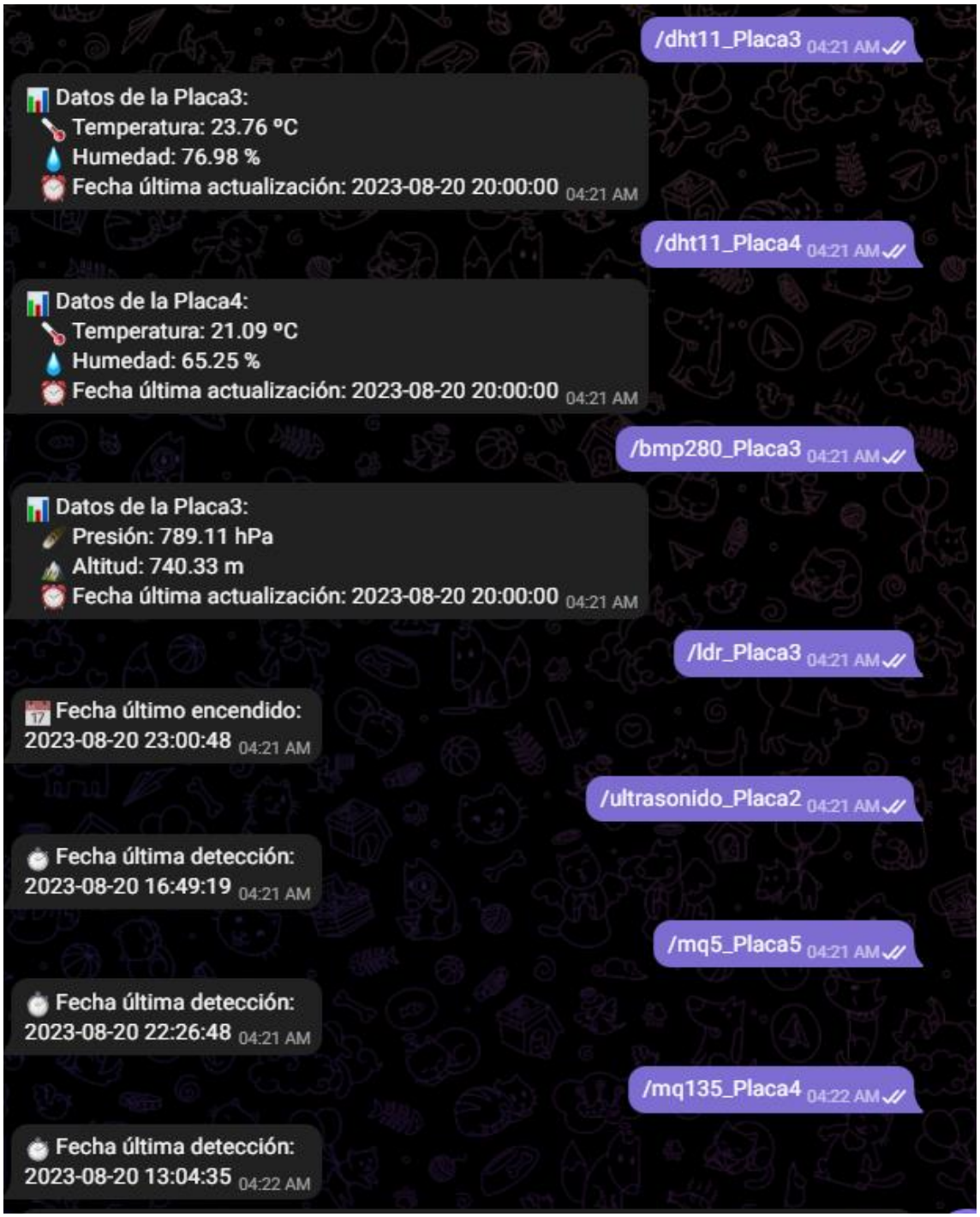


Figura 41. Respuestas bot de Telegram.

Capítulo 6 CONCLUSIONES Y LÍNEAS FUTURAS

En este apartado, se abordan las conclusiones fundamentales que se desprenden del proyecto. Además, se presentará el presupuesto utilizado en la implementación de la solución, destacando los recursos y costos involucrados. Por último, se esbozarán las posibles líneas futuras de investigación y desarrollo, identificando áreas de mejora y expansiones potenciales para el proyecto.

6.1. Conclusiones.

En el presente proyecto se ha llevado a cabo la creación de un sistema IoT de domótica que habilita el control y monitoreo remoto del hogar desde ubicaciones externas. Durante su desarrollo, se superaron diversos desafíos técnicos, como la implementación de una sólida comunicación de datos entre sensores y una plataforma web a través de bases de datos, la capacidad de enviar comandos al microcontrolador para su ejecución mediante una interfaz web, la calibración precisa de los sensores y la realización de cálculos específicos adaptados a cada aplicación. En todo momento, se mantuvo un equilibrio entre la calidad y el coste del sistema, de modo que este cumpliera con su propósito sin generar un aumento significativo en los costos finales.

En definitiva, el resultado obtenido se considera satisfactorio y se han alcanzado los objetivos establecidos inicialmente. Este proyecto ha sido una experiencia gratificante desde mi perspectiva personal, ya que me apasiona la exploración, el trabajo con dispositivos y la programación. Sin embargo, a lo largo del proceso, he enfrentado desafíos significativos debido a que mi formación académica no se centra específicamente en informática. Estos obstáculos, aunque retrasaron el avance del proyecto en algunas ocasiones, me brindaron valiosas oportunidades de aprendizaje y superación.

Desde una perspectiva profesional, considero que este proyecto tiene un gran potencial. El mundo está evolucionando hacia la automatización y la tecnología desempeña un papel cada vez más importante en nuestra vida diaria.

6.2. Líneas Futuras

A pesar de los logros significativos alcanzados en este proyecto de domótica, existen áreas clave que pueden ser objeto de mejoras y desarrollos futuros para enriquecer aún más la solución.

Se plantea la mejora del bot de Telegram con el objetivo de lograr una conversación más fluida y natural con los usuarios, evitando la necesidad de comandos predefinidos. Esto podría lograrse mediante la integración de inteligencia artificial (IA) o tecnologías similares que permitan al bot comprender de manera más intuitiva las solicitudes y preguntas de los usuarios, brindando respuestas contextualmente relevantes.La actualización y el monitoreo continuo también son aspectos cruciales que considerar en las futuras iteraciones del proyecto. Dado que la tecnología se encuentra en constante evolución, es esencial mantener tanto el hardware como el software actualizado para garantizar su funcionalidad y seguridad. El monitoreo constante del sistema para identificar posibles vulnerabilidades y aplicar parches de seguridad de manera oportuna es esencial para mantener el sistema resistente a las amenazas. En este sentido, la implementación de un sistema automatizado de actualización y monitoreo puede ser altamente beneficiosa para garantizar un rendimiento óptimo y una protección continua del proyecto.

Otra posible dirección futura para el proyecto es la integración de fuentes de energía renovable como paneles solares. Esto permitiría una gestión inteligente de la producción y el consumo de energía en el hogar, optimizando la eficiencia y reduciendo costos a largo plazo. Además, se podrían explorar opciones de almacenamiento de energía, como baterías, para garantizar un suministro confiable incluso en momentos de baja producción renovable. Esta expansión hacia la energía renovable contribuiría a la sostenibilidad y a la reducción de la huella de carbono en el hogar.

Por último, una perspectiva a largo plazo implica la expansión y escalabilidad del proyecto hacia la creación de un edificio inteligente completo. Esto implicaría la incorporación de una variedad de sensores, dispositivos y sistemas de automatización para optimizar la gestión de recursos, la seguridad y la eficiencia energética en un entorno más amplio.

Bibliografía

[1] Arduino. (2023). *IDE Arduino*. Obtenido de https://www.arduino.cc/en/about

[2] arduino. (s.f.). *arduino*. Obtenido de https://mosquitto.org

[3] Brotons, M. M. (2010). *Sistema de gestión domótica de una vivienda.* Barcelona.

[4] DataScientest. (7 de Abril de 2022). *MongoDB: todo sobre la base de datos NoSQL orientada a documentos*. Obtenido de https://datascientest.com/es/mongodb-todo-sobre-la-base-de-datos-nosql-orientada-a-documentos

[5] Hernández, L. d. (2020). *ProgramarFácil*. Obtenido de Sensor DHT11: https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/

[6] HiveMQ. (s.f.). *HiveMQ*. Obtenido de https://www.hivemq.com/mqtt-essentials/

[7] Macho, J. C. (2020). *Prometec*. Obtenido de Sensor de Presión y temperatura BMP280: https://www.prometec.net/sensor-de-presion-y-temperatura-bmp280/

[8] MCIelectronics. (2019). *Rapsberry Pi*. Obtenido de https://raspberrypi.cl/que-es-raspberry/

[9] mechatronics, n. (2020). *Sensor ultrasonido naylamp mechatronics*. Obtenido de https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html

[10] MongoDB. (s.f.). *MongoDB*. Obtenido de https://www.mongodb.com

[11] Mosquitto. (s.f.). *Mosquitto*. Obtenido de https://mosquitto.org

[12] Node-RED. (s.f.). *Node-RED*. Obtenido de https://nodered.org

[13] Porras, D. P. (2022). *Diseño y Desarrollo de un Gemelo Digital de un Brazo Robótico Arduino (TinkerKit Braccio).* Málaga.

[14] programarfacil. (2019). *Guía de introducción a MQTT con ESP8266 y Raspberry Pi*. Obtenido de https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/

[15] robótica, T. e. (2023). *Sensor de gases MQ-135*. Obtenido de https://www.turibot.es/sensor-de-gases-mq-135

[16] robótica, T. e. (s.f.). *Sensor de gases MQ-5* . Obtenido de https://www.turibot.es/sensor-de-gases-mq-5-sensible-al-gas-natural

[17] Ruiz, A. O. (2017). *Diseño y desarrollo de un sistema domótico.* Valencia.

[18] turiBOT. (2020). *Detector de obstáculos por infrarrojos*. Obtenido de https://www.turibot.es/emisor-y-receptor-de-infrarrojos-para-arduino

[19] turiBOT. (2020). *Medidor de distancias por ultrasonidos HC-SR04 para Arduino* . Obtenido de https://www.turibot.es/medidor-de-distancias-por-ultrasonidos-hc-sr04-para-arduino

[20] turiBOT. (2020). *Modulo BMP280 para medir la presión barométrica* . Obtenido de https://www.turibot.es/modulo-bmp280-para-medir-la-presion-barometrica

[21] turiBOT. (2020). *Sensor LDR fotosensible*. Obtenido de https://www.turibot.es/sensor-ldr-fotosensible

[22] turiBOT. (s.f.). *Modulo Sensor de temperatura y humedad DHT11* . Obtenido de https://www.turibot.es/modulo-sensor-de-temperatura-y-humedad-dht11

[23] UnitElectronics. (2020). *Tienda de componentes electrónicos UNITELECTRONICS*. Obtenido de https://uelectronics.com/producto/fc-51-sensor-de-obstaculos-reflectivo-infrarojo/

[24] Wikipedia. (s.f.). *Microcontrolador ESP-8266*. Obtenido de https://es.wikipedia.org/wiki/ESP8266

Apéndice A. Manual de despliegue

A.1. Requisitos previos.

A.1.1 Requisitos hardware

- Ordenador personal
- Raspberry Pi 3/4
- Tarjeta MicroSD 4GB o superior
- Lector tarjetas MicroSD
- Cables USB 2.0 tipo A/B.
- Placas Wifi ESP-8266.
- Sensores:
    - DHT11 (temperatura y humedad)
    - LM-382 (luz)
    - BM280 (presión y altitud)
    - MQ5 y MQ135 (gases)
    - Ultrasonidos (movimiento)
- Cables DuPont
- Placas PCB

A.1.2 Requisitos software

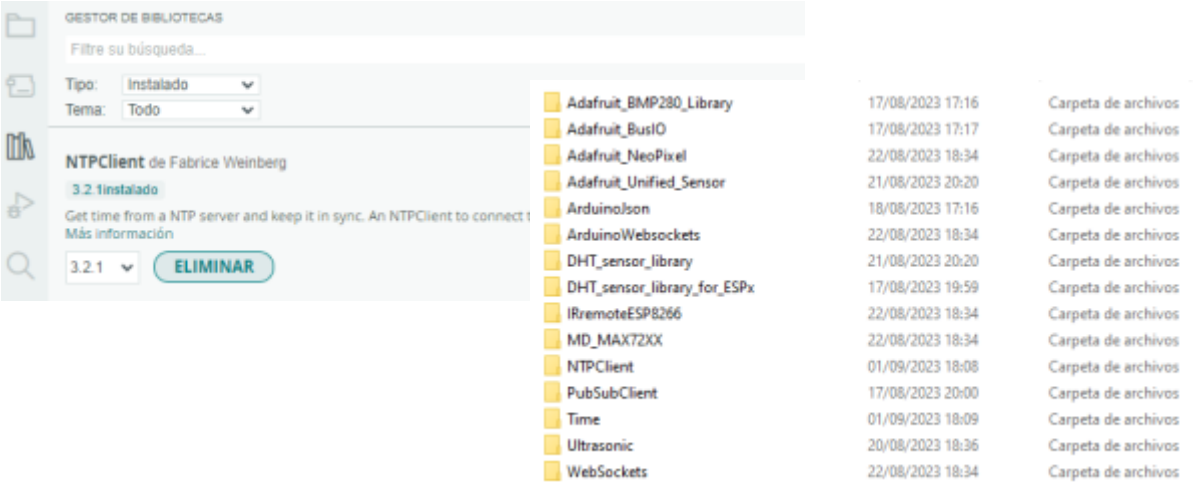
- Arduino IDE (entorno de desarrollo)
- MongoDB Compass (consulta BD visualmente)
- Mosquitto (bróker MQTT)
- Node-RED
- -Rasberry Pi Imager

A.2 Instalación

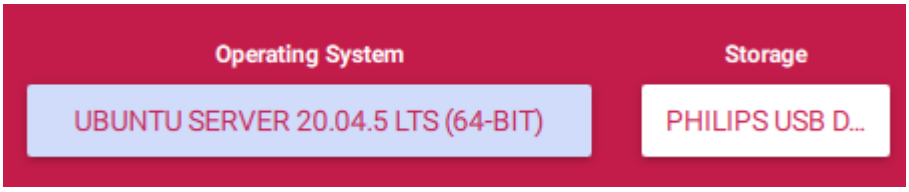
Para la instalación hemos de diferenciar los pasos a realizar sobre nuestro ordenador personal y sobre la Raspberry que actuará como servidor.

A.2.1. Ordenador personal

- *IDE de Arduino:*
    - Acceder a la siguiente página: https://www.arduino.cc/en/software, descargar e instalar la versión más reciente
    - Desde el IDE instalar las librerías que aparecen en la foto



- *Preparación de la Raspberry:*
    - Acceder a la siguiente página: https://www.raspberrypi.com/software/ para descargar el ejecutable.
    - Seleccionar como sistema operativo Ubuntu Server 20.04.5 LTS y conectar nuestra MicroSD con el lector y seleccionarla como almacenamiento
    - Configurar el usuario y contraseña, habilitar la conexión SSH y Wifi



A.2.2. Raspberry

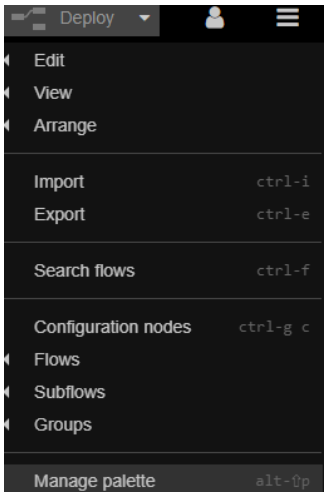
Una vez tenemos nuestra Raspberry funcionando hemos de realizar la instalación de los diversos servicios que nos permitirán el funcionamiento de nuestro sistema.

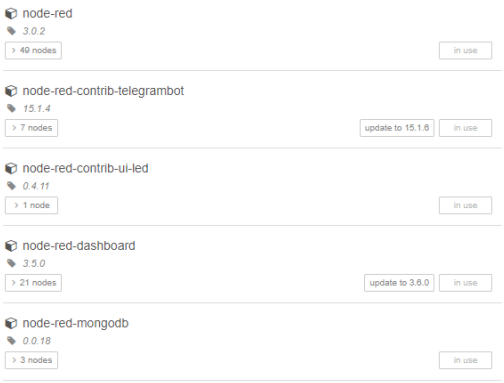
- *Mosquitto*:
  - Tutorial: https://www.vultr.com/docs/install-mosquitto-mqtt-broker-on-ubuntu-20-04-server/
    - Usuario: saulpi
    - Contraseña: sAuLpI1234

- *Instalación de MongoDB:*
  - Tutorial: https://www.mongodb.com/developer/products/mongodb/mongodb-on-raspberry-pi/
  - Crear un usuario con estos datos:
    - Usuario: saul
    - Contraseña: EgBOl03dItNCh6ym

- *Instalación y configuración de Node-RED:*
  - Tutorial: https://nodered.org/docs/getting-started/raspberrypi
    - Usuario: saulpi
    - Contraseña: s@ulPI1234
    - Passphrase: estasiquesi

- Una vez instalado Node-RED, hemos de acceder desde el menú a la opción “Manage Palette” e instalar las siguientes librerías.





  - Importar los flujos .json entregados.
  - Configurar los nodos de MQTT y MongoDB usando los usuarios y contraseñas anteriormente proporcionados.
  - Configurar el bot de telegram usando el token: *6417257682:AAG4ZKs6OmDagdYa8blLVTVD9FNnacrsG-A*

- Por último, hemos de abrir los puertos de la Raspberry para que dentro de la red local se tenga acceso a los servicios desde otros
- dispositivos conectados a la red. Seguir el siguiente tutorial y los puertos a abrir son los siguientes: 22, 1880, 1883 y 27017.
-
- Opcionalmente podemos acceder a nuestro router y redireccionar los mismos puertos para poder acceder a los servicios desde fuera usando nuestra dirección IP pública. Según el proveedor de Internet este proceso es diferente, por lo que se recomienda consultar según este.

Todos los programas y flujos usados se encuentran en el directorio de github: https://github.com/sajur30/tfgsaul.git