Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño de una Estación Meteorológica WiFi basada en un microcontrolador ESP32

Design of a WiFi weather station based on a microcontroller ESP32

Autor/es

JORGE VENTURA GUILLAMÓN

Director/es

JESÚS LÁZARO PLAZA

Grado en Ingeniería Electrónica y Automática

2024

Resumen

El presente trabajo de fin de grado consiste en el diseño e implementación de un sistema domótico de monitorización ambiental mediante el uso de dos microcontroladores ESP32. El principal objetivo que enfrenta este proyecto es crear un sistema que permita la recolección y visualización en tiempo real de datos ambientales tales como temperatura, humedad, presión, luminosidad. Asimismo, también se desea monitorear la temperatura y humedad del interior de la vivienda, con el objetivo de diseñar un sistema de control que permita al usuario de fijar un valor de temperatura interior en la que el sistema por medio de dos relés active o desactive los dispositivos de climatización necesarios para estar siempre en el margen de temperatura seleccionada por el usuario. Tanto la visualización como el control del sistema se hará desde un servidor web, permitiendo al usuario acceder remotamente desde cualquier lugar con conexión a internet.

Abstract

This final degree project focuses on the design and implementation of a home automation system for environmental monitoring using two ESP32 microcontrollers. The primary objective of this project is to create a system that enables the real-time collection and visualization of environmental data such as temperature, humidity, pressure, and luminosity. Additionally, the project aims to monitor the indoor temperature and humidity, with the goal of designing a control system that allows the user to set a desired indoor temperature. The system will use two relays to activate or deactivate climate control devices, ensuring the indoor temperature remains within the user's selected range. Both visualization and control of the system will be accessible via a web server, allowing users to monitor and manage it remotely from anywhere with an internet connection.

Escuela Universitaria Politécnica - Teruel

Universidad Zaragoza

Escuela Universitaria Politécnica de Teruel

MEMORIA

Diseño de una Estación Meteorológica WiFi basada en un microcontrolador ESP32

Design of a WiFi weather station based on a microcontroller ESP32

Autor: Jorge Ventura Guillamón

Director: Jesús Lázaro Plaza

Fecha: Diciembre 2024

Índice de contenido

1. Introducción 1
1.1. Contexto y motivación 1
1.2. Importancia de la Monitorización Ambiental 1
1.3. Objetivos del proyecto 2
1.4. Metodología 3
1.5. Estructura de la memoria 4
2. Marco Teórico 5
2.1. ESP32 5
2.1.1. Familia ESP32 5
2.1.2. Módulos ESP32 6
2.1.3 Placas de desarrollo ESP32 7
2.1.3 Placa de desarrollo HW-394 7
2.1.4 Conclusión 10
2.2. Descripción de los dispositivos de entrada utilizados 10
2.2.1. Sensor BME280 10
2.2.2. Sensor FC-37 11
2.2.3. Sensor LDR 12
2.2.4. Sensor DHT22 13
2.2.5. Pulsadores de Botón 14
2.3. Descripción de los dispositivos de salida utilizados 14
2.3.1. Modulo Relé 3.3V 2CH 15
2.3.2 Diodos LED 15
2.4. Descripción de los elementos de control de energía utilizados 16
2.4.1. Fuente de alimentación MB102 16
2.4.2 Optoacoplador 4N35 17
2.5. Comunicación Bluetooth y Wi-Fi 18
2.5.1. Bluetooth 18
2.5.2. Wi-Fi 19
2.6. Servidores web y Adafruit IO 20
2.6.1. Características de Adafruit IO 20
3. Diseño del Sistema 21
3.1. Arquitectura del sistema 21
3.2. Diagrama de bloques 22

4. Implementación 24
4.1. Configuración del hardware 24
4.1.1. Conexión de sensores y relé al ESP32_Interior 24
4.1.2. Conexión de sensores al ESP32_Exterior 26
4.2 Implementación de Software 29
4.2.1. Visual Studio Code con PlatformIO 29
4.2.2. Soportes de programación 30
4.3 Bibliotecas 30
4.4. Programación del ESP32_Interior 31
4.4.1. Lectura de sensores 32
4.4.2. Recepción de datos por Bluetooth 32
4.4.3. Envío de datos al servidor web Adafruit IO 33
4.4.4. Bucle de control 35
4.4.5. Interrupciones 36
4.5. Programación del ESP32_Exterior 37
4.5.1. Lectura de sensores 37
4.5.2. Envío de datos por Bluetooth 38
4.5.3. Gestión de la alimentación de los sensores 39
4.6 Configuración del servidor web Adafruit IO 39
4.6.1 Creación de cuenta y obtención de credenciales 40
4.6.2 Integración de credenciales en ESP32_Interior 40
4.6.3 Recepción de datos en Adafruit IO 41
4.6.4 Creación de paneles interactivos 41
5. Resultados 42
5.1 Monitorización de datos 42
5.2 Interacción con el usuario 43
5.3. Pruebas y validación 44
5.3.1. Pruebas de comunicación entre ESP32 44
5.3.2. Pruebas de envío de datos a Adafruit IO 44
5.3.3. Pruebas del bucle de control 45
5.4 Conclusiones 45
6. Conclusiones 46
6.1 Líneas de mejoras 47
Bibliografía 1

Índice de tablas

Tabla 1 Características ESP32-WROOM-32 8
Tabla 2 Características BME280 11
Tabla 3 Características FC-37 12
Tabla 4 Características DHT22 13
Tabla 5 Características pulsadores de botón 14
Tabla 6 Características módulo Relé 3.3V 2CH 15
Tabla 7 Características Diodo LED Azul 16
Tabla 8 Características MB102 17
Tabla 9 Características 4N35 18
Tabla 10 Características Bluetooth ESP32-Wroom-32 19
Tabla 11 Características Wi-Fi ESP32-Wroom-32 19

Índice de figuras

Figura 1 Módulos ESP32 [4] 6
Figura 2 Placa de desarrollo HW-394 [5] 7
Figura 3 Pinout HW-394 [5] 9
Figura 4 BME280 11
Figura 5 FC-37 12
Figura 6 LDR 13
Figura 7 DHT22 13
Figura 8 Pulsador de Botón 14
Figura 9 Módulo Relé 3.3V 2CH 15
Figura 10 Diodo LED Azul 16
Figura 11 MB102 17
Figura 12 4N35 17
Figura 13 Diagrama de flujo 23
Figura 14 Conexionado MB102 24
Figura 15 Conexionado DHT22 25
Figura 16 Conexionado módulo relé 3.3V 2CH 25
Figura 17 Conexionado pulsadores de botón 26
Figura 18 Conexionado MB102 26
Figura 19 Conexionado 4N35 27
Figura 20 Conexionado BME280 27
Figura 21 Conexionado FC-37 28
Figura 22 Conexionada LDR 28
Figura 23 Creación de proyecto PlatfomrIO - Visual Studio Code 29
Figura 24 Bibliotecas ESP32_Interior 30
Figura 25 Bibliotecas ESP32_Exterior 31
Figura 26 Lectura de sensores ESP32_Interior 32
Figura 27 Recepción de datos por Bluetooth ESP32_Interior 33
Figura 28 Envío de datos al servidor web Adafruit IO ESP32_Interior 34
Figura 29 Bucle de control ESP32_Interior 35
Figura 30 Interrupciones ESP32_Interior 36
Figura 31 Lectura de los sensores ESP32_Exterior 37
Figura 32 Envío de datos por Bluetooth ESP32_Exterior 38
Figura 33 Gestión de la alimentación de los sensores ESP32_Exterior 39
Figura 34 Creación de cuenta y obtención de credenciales Adafruit IO 40
Figura 35 Integración de credenciales en ESP32_Interior 40
Figura 36 Pestaña Feeds Adafruit IO 41
Figura 37 Creación de paneles interactivos Adafruit IO 41
Figura 38 Prototipo ESP32_Exterior 42
Figura 39 Prototipo ESP32_Interior 42
Figura 40 Dashboard Adafruit IO 43
Figura 41 Suspensión profunda ESP32_Exterior 47

1. Introducción

1.1. Contexto y motivación

Actualmente los sistemas de monitoreo ambiental son realmente importantes en diversos aspectos de la vida, desde la agricultura, la gestión de calidad del aire en áreas urbanas, la gestión de edificios inteligentes hasta el control de desastres naturales. Basan su funcionamiento en la recopilación de datos ambientales, que posteriormente se procesan, analizan y se ejecutan las acciones necesarias, que pueden ir desde controlar con actuadores las condiciones perfectas en invernaderos, ajustar en tiempo real sistemas de calefacción/ventilación para haber un ambiente perfecto en un edificio, o enviar alertas tempranas cuando se detecten aumentos considerables en el cauce de un rio.

La finalidad de este proyecto es desarrollar un sistema domótico ambiental en el que el usuario pueda visualizar el estado ambiental tanto de su casa como de su entorno en tiempo real desde cualquier lugar con conexión a internet asimismo también será posible controlar varios actuadores que incidan sobre los datos recopilados por los sensores.

La principal motivación es crear un sistema eficiente, preciso y a precio económico, dado que los sistemas de monitoreo ambiental a la venta actualmente tienen un precio considerablemente mayor con prestaciones inferiores a las que se presta este proyecto.

1.2. Importancia de la Monitorización Ambiental

Hoy día, la monitorización ambiental cuenta con gran relevancia debido a los grandes avances tecnológicos que se han dado en la electrónica y en el mundo de las IoT, abriendo así un mundo entero de posibilidades. Debido a ello actualmente los sistemas domóticos de monitorización son mucho más accesibles tanto económicamente como en facilidad de uso. Por ello se ha hecho más popular la

implementación de estos sistemas en viviendas. Estos sistemas son perfectos para mantener el confort y la eficiencia en los hogares. Controlar parámetros básicos como la temperatura y la humedad del interior no solo mejora la calidad de vida y el bienestar, sino que también puede contribuir a la optimización del consumo energético. Un sistema de monitorización domótico que pueda actuar en función de los parámetros de entrada puede ayudar a reducir significativamente el consumo energético, haciendo así un hogar más cómodo y eficiente.

[1]

1.3. Objetivos del proyecto

Los objetivos de este proyecto son los siguientes:

El objetivo principal es crear un sistema domótico de monitoreo ambiental mediante el uso de dos microcontroladores ESP32 el cual tiene que ser efectivo, preciso y completo.

Para ello el siguiente objetivo será el control de las comunicaciones, punto crucial del proyecto ya que será necesaria la implementación de varios tipos de comunicaciones, como la comunicación serie I2C para la recopilación de datos de los sensores como también las comunicaciones inalámbricas bidireccionales de Bluetooth y Wi-Fi para transmitir datos, y así enviarlos a un servidor web para desde el que poder visualizarlos y realizar acciones en tiempo real desde cualquier punto con conexión a internet.

También se va a implementar un bucle de control que, basado en la temperatura interior y un valor de consigna seleccionado por el usuario en el servidor web, active o desactive unos relés que permiten el control de dispositivos externos como pueden ser sistemas de aire acondicionado o calefacción.

Se diseñará un circuito electrónico que optimice el consumo energético de los sensores, así como se controlarán los modos de suspensión del microcontrolador para mejorar la eficiencia del sistema.

1.4. Metodología

Este proyecto inicialmente se dividió en varias fases claramente definidas con el fin de asegurar un desarrollo ordenado y consistente. A continuación, se detallan las fases del proceso:

- Planificación: Se llevó a cabo un proceso de investigación para la selección más adecuada de los componentes, así como un estudio de un sistema que satisficiera los objetivos propuestos. Se llevaron a cabo comparaciones entre componentes, así como también se tuvieron en cuenta las compatibilidades de estos. Para de este modo crear un sistema compatible entre el de buen rendimiento y bajo coste.

- Diseño: Mediante diagramas y esquemas se esbozó la arquitectura que plantea la solución al proyecto, definiendo de manera precisa la interconexión entre componentes y la secuencia de procesos. En esta fase también se elaboraron los planos eléctricos.

- Implementación: Etapa importante, aquí se llevó a cabo el desarrollo tanto del software como del hardware. En esta etapa, se ensamblaron los componentes, se programaron ambos microcontroladores y se integraron todos los periféricos necesarios. Esta etapa requirió de un gran esfuerzo y una inversión significativa de tiempo, ya que surgieron pequeños problemas que fueron retrasando el progreso.

- Pruebas y validación: Se procedió a comprobar y verificar el correcto funcionamiento del sistema. Se realizaron pruebas exhaustivas, con el fin de asegurar que el sistema operará de forma precisa, sin fallos y cumpliendo los objetivos propuestos. En algunas áreas se perfeccionaron ciertos aspectos y se realizaron los cambios necesarios para conseguir un mejor funcionamiento del sistema.

- Documentación: Última fase, en ella se explica de manera detallada el desarrollo del proyecto. Proporciona un registro completo y organizado para facilitar la compresión del proyecto.

1.5. Estructura de la memoria

La memoria está estructurada de la siguiente manera:

Punto 1: Introducción: Se presenta el contexto, la motivación y los objetivos del proyecto, así como también la estructura del documento.

Punto 2: Marco Teórico: Se describe la teoría y los conceptos relevantes para el proyecto, incluyendo los microcontroladores ESP32, los sensores utilizados, y las comunicaciones.

Punto 3: Diseño del Sistema: Se detalla la arquitectura del sistema, y se presenta el diagrama de flujo que explica el funcionamiento del sistema.

Punto 4: Implementación: Se explica la configuración del hardware y la programación software, incluyendo la gestión de la alimentación de los sensores, bucle de control e interrupciones.

Punto 5: Resultados: Se presentan las pruebas realizadas, los resultados obtenidos y su análisis.

Punto 6: Conclusiones: Se resumen los logros del proyecto, las limitaciones encontradas y las posibles mejoras futuras.

Punto 7: Bibliografía: Se listan las fuentes de información utilizadas.

2. Marco Teórico

2.1. ESP32

Los microcontroladores ESP32, están desarrollados por la empresa china Espressif Systems, están fabricados por la multinacional taiwanesa TSMC, son dispositivos de bajo coste y alto rendimiento, son superiores al resto de microcontroladores que hay en el mercado en relación prestaciones/precio debido a su producción en gran escala lo que le permite a Espressif Systems ofrecerlos a un precio muy competitivo.

De hecho, Espressif con su amplia gama de microprocesadores ha conseguido afianzarse de una grandísima parte de los dispositivos IoT, debido a su versatilidad, capacidad de procesamiento, conectividad de bajo consumo energético, y más mejoras con las que hace que se desmarque de la competencia.

2.1.1. Familia ESP32

La multinacional china Espressif Systems ha desarrollado diferentes familias de microcontroladores, comenzaron presentando el ESP8089 en 2013, posteriormente durante el verano de 2014 lanzaron el ESP8266 y dos años después presentaron el ESP32, actualmente este último cuenta con muchas versiones y variantes.

- ESP8266: Primer microcontrolador desarrollado por Espressif enfocado en el IoT, tuvo un gran impacto en el mercado, se hizo prontamente conocido debido principalmente a su conectividad Wi-Fi integrada, lo que le otorgaba una ventaja competitiva respecto al resto de microcontroladores de su precio. El Wi-Fi integrado y el bajo costo hizo que fuera ampliamente utilizado en proyectos de IoT, actualmente sigue siendo una opción muy a tener en cuenta para aplicaciones específicas.

- ESP32: Familia ESP32, representa la evolución del ESP8266, éste cuenta con múltiples mejoras en términos de capacidad de procesamiento, conectividad, almacenamiento y así como otras muchas optimizaciones. Esta familia ESP32

cuenta con múltiples variantes como el ESP32-S2, ESP32-S3, ESP32-C3, entre otros. Cada una con características distintas, este es uno de los motivos por los que la familia ESP32 es tan popular, su versatilidad.

2.1.2. Módulos ESP32

Cada chip de la familia ESP32 puede integrarse en diferentes módulos, cada módulo ofrece distintas prestaciones en cuanto a almacenamiento, opciones adicionales de conectividad, mayor potencia de procesamiento entre otras. De este modo ofrecen la posibilidad de que haya un módulo que se adapte perfectamente a los requisitos del usuario. Algunos módulos populares son los siguientes:

Figura 1 Módulos ESP32 [2]

- ESP32-WROOM-32: Es uno de los más populares, incluye todas las características básicas necesarias para utilizarlo en aplicaciones de IoT ya que cuenta con Wi-Fi y Bluetooth integrados. Prestaciones muy equilibradas lo que lo hace versátil y una opción muy a tener en cuenta.
- ESP32-WROVER: Esté modulo cuenta con más memoria RAM y con opciones adicionales de conectividad, como soporte para ethernet o antenas y conectores U.FL.
- ESP32-SOLO: Es una versión mononúcleo del ESP32, está diseñado y enfocado para aplicaciones que requieran menor capacidad de procesamiento.

2.1.3 Placas de desarrollo ESP32

Las placas de desarrollo o “Devkits” son placas diseñadas para facilitar el uso de microcontroladores. Cuentan con todo lo necesario para facilitar la puesta en marcha rápida del microcontrolador, como fuente de alimentación, puertos de comunicación o pines de conexión rápida de otros periféricos. La familia ESP32 cuenta con multitud de placas de desarrollo unas de las más populares son las siguientes:

- ESP32 DevKitC: Esta placa de desarrollo puede montar diferentes microcontroladores de la familia ESP32. Es muy utilizada y cuenta con cantidad de información disponible.
- ESP32 DevKit V1: Este modelo a diferencia del anterior monta únicamente el módulo ESP32-WROOM-32. Es una versión más antigua del DevKitC con lo que cuenta con hardware más antiguo careciendo así de mejoras y optimizaciones presentes en modelos más nuevos.

2.1.3 Placa de desarrollo HW-394

La placa seleccionada para la realización de este proyecto ha sido la placa HW-394 (Figura 2). En este apartado se van a exponer las características y especificaciones con las que cuenta este modelo.

Figura 2 Placa de desarrollo HW-394 [3]

Esta placa de desarrollo monta el módulo ESP32-WROOM-32, uno de los más avanzados equilibrados y versátiles de la familia ESP. La HW-394 se distingue por la robustez que tiene en cuanto a la implementación de aplicaciones de IoT ya que tiene una excelente capacidad de manejar comunicaciones Wi-Fi y Bluetooth.

- Módulo ESP32-Wroom-32 sus principales características son las siguientes:
  - Conectividad Wi-Fi y Bluetooth: Cuenta con Wi-Fi 802.11 b/g/n y Bluetooth 4.2.
  - Procesador Dual-Core: Integra un procesador binucleo Xtensa® LX6 con capacidad de operar hasta los 240 MHz.
  - Almacenamiento: Incluye 520 KB de SRAM y 4 MB de Flash.
  - Periféricos: Cuenta con interfaces de comunicación (UART, SPI, I2C), así como también conversores ACDs DACs y salidas PWM.
  - Bajo Consumo Energético: En su versión más restrictiva cuenta con un consumo aproximado de 10 µA.

Especificaciones técnicas (Ficha técnica del fabricante):

Rangos de funcionamiento	
Voltaje	3.0 - 3.6 V DC
Corriente	80mA (media)
Temperatura	-40 - 85ºC
Caracteristicas	
Pines GPIO	25
Pines entradas ADC	12 pines (ADC de 12 bits)
Pines salida DAC	2 pines (DAC de 8 bits)

Tabla 1 Características ESP32-WROOM-32

Pinout placa de desarrollo HW-394:

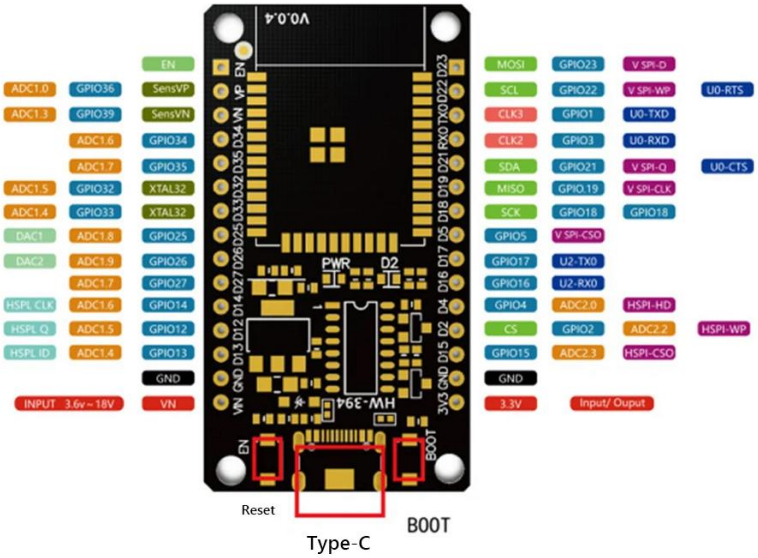


Figura 3 Pinout HW-394 [3]

La (Figura 3) detalla la distribución de patillas de la placa de desarrollo utilizada; en este caso es la suministrada por el vendedor, la imagen muestra una valiosa información, necesaria a la hora de realizar proyectos con él ya que muestra el funcionamiento que se le puede atribuir a cada pin de esta placa.

Algunas peculiaridades e informaciones para tener en cuenta son las siguientes:

Tiene 5 pines que intervienen durante la configuración de arranque del ESP32 los cuales son los siguientes GPIO (0,2,4,5,12 y el 15), estos es mejor evitarlos en medida de lo posible.

Unos de los pines característicos de esta placa de desarrollo es el GPIO0 que es el utilizado para poner el ESP32 en modo arranque. Otro es Pin GPIO2 que controla un led integrado en la placa, muy útil en diversas aplicaciones.

Esta placa también cuenta con la ventaja de incorporar resistencias de pull-up y pull-down internas en varios de sus pines lo que facilita enormemente su uso.

Cuenta con un puerto de comunicación tipo-C el cual sirve tanto para la comunicación y programación del microcontrolador como para la alimentación del mismo.

2.1.4 Conclusión

Para el proyecto se ha optado por seleccionar la placa de desarrollo HW-394 por su versatilidad su capacidad de conectividad y sus características avanzadas con las que cuenta. Esta placa de desarrollo se ajusta perfectamente a las exigencias que requiere este proyecto. La estación meteorológica requiere de un módulo que tenga la capacidad de conectarse tanto por Bluetooth como por Wi-Fi. Además, es necesario que cuente con comunicación de tipo I2C para el uso de determinados sensores, con suficientes puertos GPIO, así como que cuente con la capacidad de almacenamiento necesaria. La placa HW-394 con su microcontrolador ESP32-WROOM-32 cumple con todas las especificaciones mencionadas con el añadido de contar con un modo energético ultra eficiente que le permitirá gozar de gran autonomía. Todo ello a un coste muy competitivo, lo que lo convierte en la opción perfecta para esta aplicación.

[2], [3], [4], [5], [6]

2.2. Descripción de los dispositivos de entrada utilizados

En este apartado se van a exponer todos los sensores y dispositivos de entrada utilizados para este proyecto los cuales se utilizan para medir diferentes parámetros ambientales o realizar algunas acciones. A continuación, se describen los sensores empleados.

2.2.1. Sensor BME280

BME280, sensor que mide la temperatura, la humedad y la presión barométrica asimismo también devuelve el valor de la altura respecto del nivel del mar mediante la relación entre la presión del aire y la altitud. La diferencia con el BMP280 es que este último no cuenta con la capacidad de medir la humedad relativa.

Figura 4 BME280

Fabricados por Bosch, este sensor es muy conocido por su alta precisión y bajo consumo de energético. El BME280 utiliza un protocolo de comunicación I2C o SPI, lo que facilita enormemente su integración con el ESP32.

Especificaciones técnicas (Ficha técnica del fabricante):

Rangos de funcionamiento	
Voltaje	1.71-3.6 VDC
Temperatura	-40-85 ºC
Humedad relativa	0-100% HR
Presión	300- 1100 hPa
Precisión	
Temperatura	±1 ºC
Humedad relativa	±3% (hasta 80% RH)
Presión	±1 hPa
Resolución	
Temperatura	0.01 ºC
Humedad relativa	0.01 %
Presión	0.01 hPa (0.001 m)

Tabla 2 Características BME280

Este sensor es perfecto para para este proyecto ya que combina en un solo dispositivo compacto las funciones de termómetro, barómetro e higrómetro, todos ellos con una excelente precisión combinado también con un bajo consumo energético lo que lo hace idóneo para este tipo de proyectos.

[7]

2.2.2. Sensor FC-37

El detector de lluvia FC-37 es un dispositivo capaz de detectar de lluvia y/o nieve en su superficie de forma rápida y precisa.

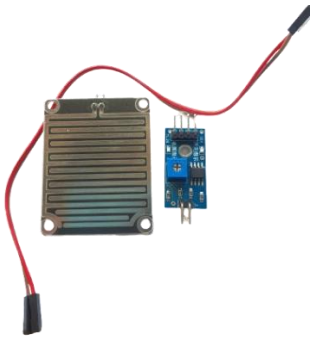


Figura 5 FC-37

Este sensor consta de una placa de detección y un módulo de control. Este sensor tiene dos modos de funcionamiento, permite tanto configurarse como detector de lluvia enviando una señal digital indicando así la presencia o ausencia de agua en su superficie de detección, como también la configuración analógica la cual proporciona una señal que varía en función de la cantidad de agua detectada. El módulo de control cuenta con un potenciómetro el cual permite ajustar la sensibilidad estableciendo así el punto a partir el cual activa la detección cuando opera en modo digital.

Especificaciones técnicas:

Modelos similares	FC-37/ YL-83/ MH-RD
Voltaje de funcionamiento:	3.3 - 5V DC
Voltaje de la señal de salida	0 ~ 5V DC

Tabla 3 Características FC-37

Operando en modo analógico y con su superficie de detección dispuesta en un ángulo de 30º, permite este sensor registrar la intensidad de la lluvia de manera efectiva.

[8], [9]

2.2.3. Sensor LDR

El sensor LDR (Light Dependent Resistor) es un dispositivo capaz de medir la intensidad de la luz. Las LDR están fabricadas de un semiconductor de alta resistencia como el sulfuro de cadmio, componente principal, el cual tiene la capacidad de variar su resistencia dependiendo de la cantidad de luz que haya incidida sobre él. De modo que cuanta mayor sea la cantidad de luz incendia sobre él más baja será su resistencia y viceversa cuanta menos luz incida sobre el sulfuro de cadmio mayor será su

resistencia. Los valores que presentan las resistencias LDR abarcan un amplio rango, pueden variar desde unos pocos cientos de ohmios cuando están bajo una iluminación intensa hasta más de un mega ohmio en condiciones de completa oscuridad.

Figura 6 LDR

Los LDRs son ampliamente utilizados en aplicaciones como sistemas de iluminación automática, dispositivos de ahorro de energía y monitoreo ambiental, debido a su simplicidad y bajo coste.

[10]

2.2.4. Sensor DHT22

El DHT22 es un sensor que mide la temperatura y la humedad relativa. Para medir la humedad lo hace por medio de un condensador de polímero y la temperatura mediante un termistor, esto hace de él un componente preciso y muy fiable.

Figura 7 DHT22

Especificaciones técnicas (Ficha técnica del fabricante):

Rangos de funcionamiento	
Voltaje	3.3 - 6V DC
Temperatura	-40 - 80 ºC
Humedad relativa	0 - 100% HR
Precisión	
Temperatura	±0.5 ºC
Humedad relativa	±2% HR (Máx. ±5% HR)
Resolución	
Temperatura	0.1 ºC
Humedad relativa	0.1 %

Tabla 4 Características DHT22

En general es un sensor de tamaño muy reducido, bajo consumo, preciso y con una larga distancia de transmisión hasta 20 metros, eso sí, algo en contra que tiene es que no puede tomar muestras cada menos de dos segundos, pero para este proyecto no es ningún problema.

[11]

2.2.5. Pulsadores de Botón

Pulsador de cuatro terminales de configuración DPST (Double Pole Single Throw) las siglas se refieren al número de circuitos que puede controlar. “Corte” se refiere a la posición del pulsador cuando está presionado. Sus terminales están dispuestos dos a dos y únicamente se conectan mientras se esté presionando el pulsador.

Figura 8 Pulsador de Botón

Especificaciones técnicas (Ficha técnica del fabricante):

Tipo	Normalmente Abierto (N.O.)
Voltaje de funcionamiento	0 - 5V DC
Resistencia de contacto	10K

Tabla 5 Características pulsadores de botón

Estos pulsadores son totalmente esenciales para proporcionar una interfaz de usuario directa y sencilla, permiten controlar diversas funciones. Es extremadamente común la utilización de estos en proyectos debido a su simplicidad y fiabilidad.

[12]

2.3. Descripción de los dispositivos de salida utilizados

En este apartado se van a exponer los preaccionadores y actuadores utilizados en el proyecto.

2.3.1. Modulo Relé 3.3V 2CH

Dispositivo esencial, permite controlar dispositivos externos de alto consumo utilizando señales de bajo voltaje como las provenientes de microcontroladores.

Figura 9 Módulo Relé 3.3V 2CH

Este módulo consta de dos relés completamente independientes, ya que cada uno de ellos es controlado por una línea de control. De este modo enviando una pequeña señal se pueden manejar cargas de hasta 10A, este módulo permite conmutar cargas tanto de corriente alterna como de continua lo que le otorga gran versatilidad. También incluye dos diodos LED que muestran el estado de cada relé, proporcionando así una experiencia más intuitiva y agradable al usuario.

Especificaciones técnicas (Ficha técnica del fabricante):

Rangos de funcionamiento	
Voltaje de funcionamiento	0 - 5V DC
Señal de control TTL	3.3 - 5V DC
Capacidad de conmutacion	10A a 250V AC o 10A a 30V DC

Tabla 6 Características módulo Relé 3.3V 2CH

Este módulo es perfecto para proyectos de domótica, ya que en combinación con un microcontrolador es capaz de actuar sobre cargas de alto consumo de forma automatizada o incluso remotamente.

[13]

2.3.2 Diodos LED

Los diodos LED (Light Emitting Diodes) son componentes electrónicos capaces de emitir luz cuando se les suministra una corriente eléctrica. Son conocidos por su alta

eficiencia energética, durabilidad y capacidad para emitir luz en una variedad de colores dentro del espectro visible, aunque también hay diodos que son capaces de emitir en otros rangos de longitudes de ondas las cuales son invisibles al ojo humano como pueden ser los infrarrojos o los ultravioletas.

Figura 10 Diodo LED Azul

Especificaciones técnicas:

Rangos de funcionamiento	
Color	Azul
Voltaje de funcionamiento	2.48- 3.7V DC
Corriente de funcionamiento	20 - 30mA

Tabla 7 Características Diodo LED Azul

[14]

2.4. Descripción de los elementos de control de energía utilizados

Esta sección describe los componentes encargados de suministrar y controlar la energía en este proyecto. Consta de dos elementos principales son:

2.4.1. Fuente de alimentación MB102

La placa MB102 en un módulo de fuente de alimentación, está diseñada para proporcionar una alimentación estable en proyectos montados sobre placas de pruebas. Este módulo es capaz de suministrar dos salidas de voltaje independientes 3.3V o de 5V, el usuario por medio de unos jumpers selecciona el voltaje deseado en cada una de las salidas. Se le puede conectar como entrada una pila de 9V y así tener el proyecto desconectado de la red eléctrica, cuenta con un pulsador para activar o desactivar las salidas, esto hace que la placa MB102 sea ideal para proyectos en los que están involucrados microcontroladores.

Figura 11 MB102

Especificaciones técnicas (Ficha técnica del fabricante):

Rangos de funcionamiento	
Voltaje de entrada	6.5 - 12V DC
Voltaje de salida	3.3V y 5V (Seleccionables)
Corriente máxima de salida	<700mA

Tabla 8 Características MB102

[15]

2.4.2 Optoacoplador 4N35

El 4N35 es un optoacoplador muy utilizado, el cual incluye un diodo LED infrarrojo y un fototransistor en su interior, todo ello encapsulado en un paquete de 6 pines (5 útiles). Este dispositivo permite la transferencia de señales eléctricas entre dos partes de un circuito mientras proporciona aislamiento eléctrico, normalmente el circuito de mando se conecta al diodo que al activarse se encarga de saturar el fototransistor conectado al circuito que quiere ser controlado, de este modo separa dos circuitos a la par que aumenta la protección y seguridad al sistema. Este dispositivo también se usa comúnmente como interruptor digital, muy útil en diversas aplicaciones.

Figura 12 4N35

Especificaciones técnicas (Ficha técnica del fabricante):

Rangos de funcionamiento	
Voltaje para saturacion (Vf)	1.2 - 1.4V DC
Corriente para saturacion (If)	10-20mA
Voltaje maximo colector-emisor (Vceo)	80V

Tabla 9 Características 4N35

[16]

2.5. Comunicación Bluetooth y Wi-Fi

Punto importante de este proyecto, en él se van a exponer las comunicaciones tanto Bluetooth como Wi-Fi, ya que son las utilizadas en este proyecto, gracias a ellas se permite tanto la comunicación entre ambos dispositivos como el envío de información al exterior.

2.5.1. Bluetooth

Bluetooth Low Energy (BLE), la versión que se ha utilizado en este proyecto es una versión optimizada del estándar Bluetooth y está enfocada para aplicaciones que requieren de baja energía y largas duraciones de batería, lo cual lo hace idóneo para este proyecto. El BLE es ideal para dispositivos IoT gracias a su eficiencia energética y capacidad de mantener la conexión constante sin grandes consumos energéticos. Una pequeña desventaja del BLE es que abarca un menor rango en comparación con el Bluetooth convencional, la diferencia es mínima. Tiene un menor consumo debido a que tiene una menor potencia de transmisión, más de 10 veces menor en comparación al Bluetooth clásico, también es más eficiente gracias a que es capaz de cambiar de modo de espera a modo activo mucho más rápido que el otro estándar aun así tiene una velocidad de transmisión de datos menor. En cómputo global las ventajas del BLE aplicadas a este proyecto mejoran significativamente a las del estándar Bluetooth convencional, por eso en este proyecto se ha decantado por utilizar esta versión, ya que está específicamente pensada para aplicaciones como las requiere este proyecto.

Especificaciones técnicas:

Frecuencia	2.4GHz
Ancho de banda	1Mbps
Distancia efectiva	10 - 30m
Distancia máxima	<50m

Tabla 10 Características Bluetooth ESP32-Wroom-32

En este proyecto se utiliza el BLE para la comunicación y transferencia de datos entre los dos ESP32, dado que solo se enviarán los datos recolectados cada 30 segundos, solo será necesario que estén en modo activo en esos intervalos de tiempo. La desventaja de la velocidad de transmisión no es ningún problema ni limitación debido a la pequeña cantidad de datos que transfieren, sumado el menor consumo por la menor potencia de transmisión hace que este estándar sea el más adecuado para este proyecto.

[17]

2.5.2. Wi-Fi

El ESP32 cuenta con capacidades Wi-Fi integradas. Soporta múltiples modos de operación, como el “Station”, esta configuración le permite conectarse a una red Wi-Fi ya existente, otro modo es el “SoftAp” en el que el propio ESP32 actúa como punto de acceso, creando así su propia red Wi-Fi. También ofrece funciones avanzadas de cifrado proporcionándole una seguridad robusta.

Especificaciones técnicas:

Frecuencia	2.4GHz
Conectividad	Wi-Fi 802.11 b/g/n
Velocidad de transmision	Hasta 150 Mbps
Seguridad	cifrado WPA/WPA2 y TLS/SSL
Distancia efectiva	
Interiores	<50m
Exteriores	<100m

Tabla 11 Características Wi-Fi ESP32-Wroom-32

Para este proyecto se utilizará la comunicación Wi-Fi en modo “Station” conectarse a una red Wi-Fi con conexión a internet de esta manera podrá enviar los datos que vaya recolectando en tiempo real a un servidor web.

[18]

2.6. Servidores web y Adafruit IO

Con el auge de las IoT, se han desarrollado cantidad de servidores web ya diseñados con el propósito de visualizar los datos enviados por microcontroladores. Ejemplos de estos son ThingSpeak, Adafruit IO o AWS IoT. Estas webs, prefabricadas y optimizadas para este cometido, permiten la recolección, almacenamiento y visualización de datos en tiempo real, facilitando mucho la creación de paneles interactivos y el análisis de datos en múltiples aplicaciones IoT. Su configuración es sencilla y de amplia compatibilidad con diferentes microcontroladores, de modo que el uso de estos servidores web los convierte en una opción perfecta.

Se han considerado y probado varios servidores web y al final se ha optado por utilizar la versión gratuita del servidor web Adafruit IO, ya que destaca por ofrecer las características más interesantes y adecuadas a las necesidades de este proyecto.

2.6.1. Características de Adafruit IO

La versión gratuita de Adafruit IO ofrece las siguientes características que facilitan la gestión y visualización de datos.

Paneles interactivos o “dashboards”: La parte principal de estos servidores web. Permiten crear interfaces muy visuales para monitorear y controlar datos y dispositivos de manera muy rápida y eficiente ya que le permite al usuario que de un simple vistazo pueda visualizar gran cantidad de datos de forma rápida e intuitiva.

Los feeds de datos: Son cada uno de los parámetros o variables que se quiere monitorizar, en la versión gratuita hay un límite de 10 feeds, es un punto en contra con respecto a otros servidores web, sin embargo, para el acometido de este proyecto es suficiente ya que el propósito es de monitorear 9 variables distintas.

Tasa de envío de datos y almacenamiento: Adafruit IO gratuito cuenta con una tasa máxima de envío de datos o “data rate” de 30 datos por minuto y permite almacenar los datos durante 30 días, lo cual es más que suficiente para las necesidades de este proyecto.

3. Diseño del Sistema

3.1. Arquitectura del sistema

La arquitectura de este sistema está compuesta por los dos módulos principales, los dos Esp32, cada uno de los dos funciona independientemente realizando sus funciones y tareas específicas. Se comunican en los intervalos necesarios garantizando así el funcionamiento correcto y eficaz del sistema. Se pasa a detallar su funcionamiento:

Esp32_Interior: Este es el módulo principal del proyecto. Cuenta con la capacidad de medir temperatura y humedad relativa del interior gracias a que cuenta un sensor DHT22. También es el responsable de enviarle la señal de “start” al ESP_Exterior para que empiece con la recolección de datos, una vez enviada la señal, le llegarán datos del ESP_Exterior cada 30 segundos. El Esp_Interior se encarga de recibirlos, tratarlos, almacenarlos y enviarlos con los suyos propios al servidor web mediante Wi-Fi. El usuario puede acceder al servidor web para visualizar los datos ambientales a tiempo real, tanto del exterior como del interior, así como manejar el bucle de control. Este bucle permite que desde cualquier punto con conexión a internet el usuario modifique una consigna que, en función de la temperatura interior se active o desactive el módulo de relés, permitiendo así control sobre cargas de alto consumo que serían los aires acondicionados y/o sistemas de calefacción. También se pueden controlar manualmente por vía de los dos pulsadores que manejan independientemente cada uno de los relés.

Esp32_Exterior: Este módulo está situado en el exterior y es el encargado de recolectar los datos ambientales por medio de los siguientes sensores:

- BME280 Sensor de temperatura, humedad relativa y presión
- FC-37 Sensor detector de lluvia
- LDR Sensor de luz

Estos sensores permiten monitorear de manera precisa las variables ambientales más relevantes proporcionando así una visión completa de las condiciones

exteriores. Este módulo no empieza a recolectar datos del entorno hasta que no ha establecido exitosamente comunicación Bluetooth con el Esp_Interior y recibe la orden de comenzar con la recolección de datos. Es así ya que no tiene la capacidad de almacenar datos ni enviarlos al servidor web por sí mismo, por lo que permanece en reposo hasta recibir la señal de “start”. Una vez recibida la señal del Esp_Interior comienza a recolectar y enviar los datos de los sensores cada 30 segundos, esto se puede observar ya que cada vez que envía datos por Bluetooth al interior se activa un diodo led azul durante un segundo.

El diseño de esta arquitectura garantiza que los datos ambientales sean monitorizados y controlados de manera eficiente asegurando tanto la precisión en la recolección de datos como la optimización de los recursos. Esto permite un monitorio detallado y eficaz sin realizar más acciones de las necesarias.

3.2. Diagrama de bloques

El siguiente flujograma proporciona una visión integral del funcionamiento del sistema, incluyendo la interconexión, la secuencia de procesos sus bucles y condiciones. Es una herramienta muy útil para comprender completamente como interactúa el sistema, así como también para identificar posibles debilidades o mejoras que pueda tener (Figura 13).

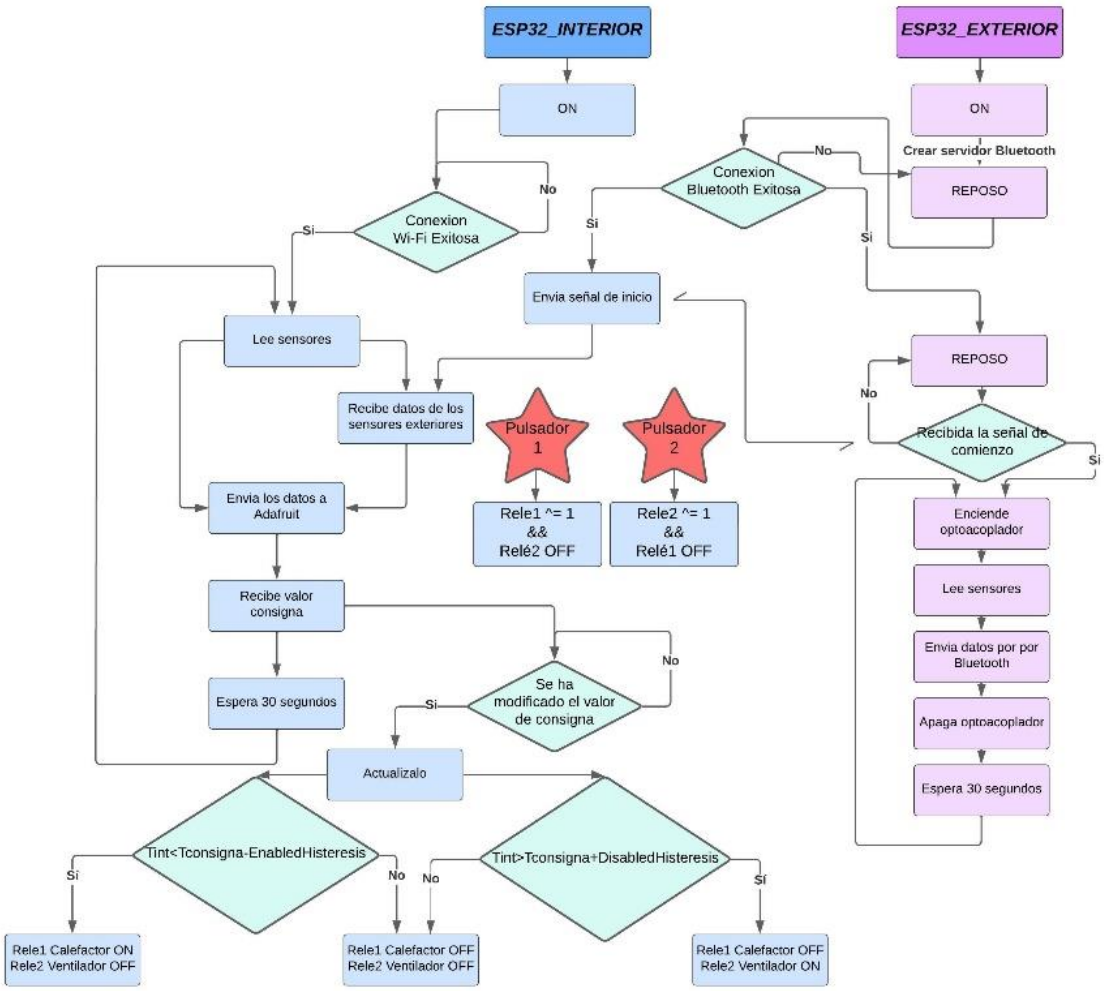


Figura 13 Diagrama de flujo

Una de las peculiaridades que se puede apreciar en el diagrama de flujo es que el ESP32_Exterior hasta que no ha establecido la conexión Bluetooth con éxito y ha recibido la señal de comienzo, este se mantiene a la espera. No comienza con la recolección de datos hasta que se cumplan dichas condiciones. Mientras que por el contrario el ESP32_Interior independientemente de si la conexión se ha establecido con éxito él empieza con monitorización de datos del interior.

Ambos pulsadores se han representado con una estrella debido a que son interrupciones que, al ser presionados desencadenan una acción inmediata.

4. Implementación

4.1. Configuración del hardware

En esta sección se va a detallar como se conectan y configuran físicamente los diferentes componentes involucrados en este proyecto, incluyendo todos los sensores y actuadores, como también los elementos encargados de la gestión de alimentación.

4.1.1. Conexión de sensores y relé al ESP32_Interior

El ESP32_Interior está equipado con el sensor DHT22 un módulo relés y dos pulsadores. A continuación, se detallan las conexiones:

- Fuente de alimentación MB102: Para este proyecto se utiliza el módulo MB102 para proporcionar voltaje estable de 3.3V a este circuito. A ella le suministra el voltaje una pila de 9V, que debido a la eficiencia y bajos consumos del circuito será capaz de suministrar energía durante largos periodos de tiempo sin necesidad de ser sustituida. Diagrama de conexiones (Figura 14):

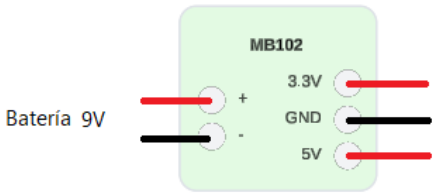


Figura 14 Conexionado MB102

- DHT22: Sensor de temperatura y humedad relativa. Tiene únicamente 3 pines de conexión, dos de alimentación y uno para el envío de datos de manera digital. Utiliza un protocolo de comunicación propio en el que únicamente necesita la conexión a una entrada cualquiera del ESP32 para su funcionamiento. Diagrama de conexiones (Figura 15):

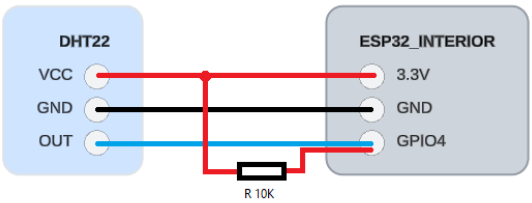


Figura 15 Conexionado DHT22

La resistencia de 10K entre Vcc y la línea del GPIO4 es una resistencia de pull-up es crucial para asegurar que la línea de datos se mantiene en estado alto “High” mientras el sensor no está enviando datos, de esto consigue estabilizar la señal y prevenir fluctuaciones. Evita que la línea de datos se quede en un estado flotante cuando no hay transmisión de datos.

- Modulo Relé 3.3V 2CH: Permite el control de dispositivos de alto consumo manteniendo en alto o en bajo el nivel de los pines de control. Diagrama de conexiones (Figura 16):

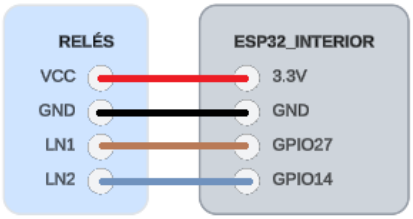


Figura 16 Conexionado módulo relé 3.3V 2CH

Como se puede observar el control de los relés únicamente ocupan una salida del ESP32 cada uno de ellos, poniéndola en alto o en bajo se activa o desactiva el relé, en este caso al poner en bajo la salida del ESP32 de la línea de control se activa el relé conmutando y luciendo su led, mientras que al poner en alto la salida vuelve a conmutar y se apaga el led.

- Pulsadores: Los pulsadores permiten el control manual de los relés. Para este proyecto se han incorporado dos uno para el control independiente de cada relé. Tienen 4 patillas conectadas internamente dos a dos. Se tratan de pulsadores

normalmente abiertos que al pulsar se conectan todos terminales. Diagrama de conexiones (Figura 17):

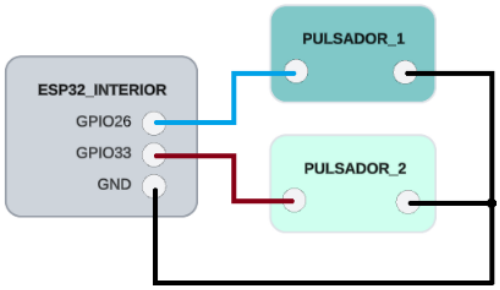


Figura 17 Conexionado pulsadores de botón

4.1.2. Conexión de sensores al ESP32_Exterior

El ESP32_Exterior el encargado de recolectar los datos ambientales del entorno exterior. La configuración de los siguientes componentes es crucial para el correcto funcionamiento del sistema.

- Fuente de alimentación MB102: La alimentación del dispositivo interior también se realiza mediante el módulo MB102 y una pila de 9V. Diagrama de conexiones (Figura 18):

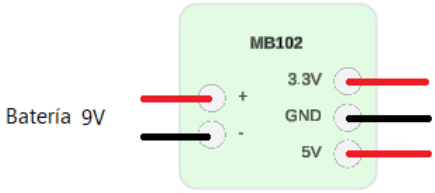


Figura 18 Conexionado MB102

- Optoacoplador 4N35: Utilizado como interruptor de modo que una salida del del ESP32 es la encargada de manejar el diodo infrarrojo, de modo que cuando la salida esté en alto el optoacoplador dejará pasar toda la corriente y voltaje proveniente de la fuente de alimentación para la alimentación de los sensores. Como protección es necesario poner entre la salida y el ánodo del optoacoplador

una resistencia externa de 180Ω para limitar la corriente. Diagrama de conexiones (Figura 19):

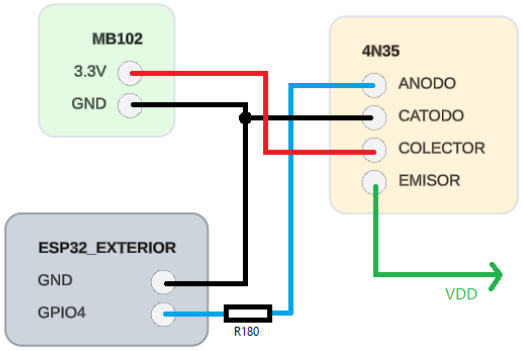


Figura 19 Conexionado 4N35

- BME280: Mide temperatura, humedad relativa y presión atmosférica con muy buena precisión. Este sensor envía los datos al ESP32 por medio del protocolo I2C, por lo que se conecta a las GPIO 21 y 22 del microcontrolador. Los pines CSB y SD0 del BME280 derivados a tierra configuran el sensor para funcionar en modo I2C y para asignarle la dirección especifica 0x76. Diagrama de conexiones (Figura 20):

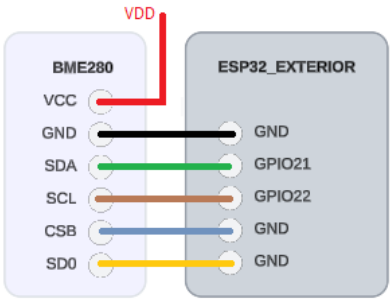


Figura 20 Conexionado BME280

- FC-37: Sensor detector de lluvia FC-37, como se mencionó en apartados anteriores este sensor se puede conectar para funcionar tanto en modo digital como en modo analógico. Para este proyecto se ha optado por configurarlo para operarlo en modo analógico y su diagrama de conexión es el siguiente (Figura 21):

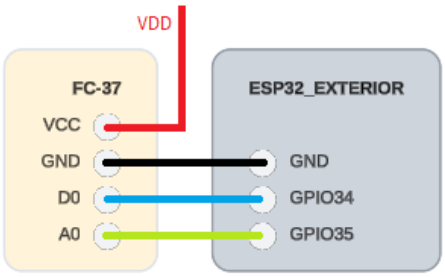


Figura 21 Conexionado FC-37

En este caso se va a utilizar únicamente en modo analógico bastaría con alimentarlo y conectarle el pin A0 al cualquier GPIO del ESP32, ya que solamente necesita una entrada para enviar la información, en este caso se ha utilizado la GPIO35. El pin D0 del FC-37 no se va a utilizar en este caso ya que se utiliza únicamente en modo digital, de todos modos, dado que sobran GPIOs del ESP32 se ha conectado a la GPIO34, de forma si en un futuro se quisiera utilizar en modo digital únicamente habría que modificar el código del ESP32.

- LDR: Sensor de luz. Como se mencionó en apartados anteriores este sensor varía su resistencia en función de la intensidad lumínica que incide sobre él. Para medir su resistencia se forma un divisor resistivo y el punto medio se coloca una entrada analógica del ESP32. Diagrama de conexiones (Figura 22):

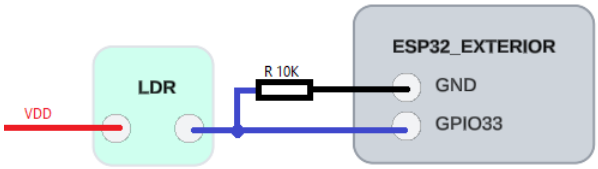


Figura 22 Conexionada LDR

Un extremo de la LDR se conecta a 3,3V, el punto central se conecta a una entrada analógica del ESP32 y de ahí parte una resistencia a GND, de modo que se forma un divisor de voltaje, lo cual conlleva mejoras sustanciales como que así se acotan los valores en un rango más manejable y especifico, lo cual facilita interpretación de mediciones. También la resistencia ayuda a estabilizar las variaciones reduciendo el

ruido y haciendo que los cambios sean más suaves y menos susceptibles a pequeñas a pequeñas perturbaciones.

4.2 Implementación de Software

Un punto importante de este proyecto es el software, los programas utilizados, las plataformas que han facilitado la programación, documentación y gestión del código. En este apartado se van a exponer las principales herramientas utilizadas.

4.2.1. Visual Studio Code con PlatformIO

Visual Studio Code es un programa de edición de código de programación, está desarrollado por Microsoft y es una herramienta poderosa y muy versátil ya que soporta multitud de lenguajes de programación. Proporciona una serie de funcionalidades que ayudan notablemente al usuario como el resaltado de errores, autocompletado de código o la depuración integrada. Para este proyecto se ha combinado Visual Studio Code con la extensión PlatformIO, una extensión que facilita notablemente la gestión de proyectos y de bibliotecas.

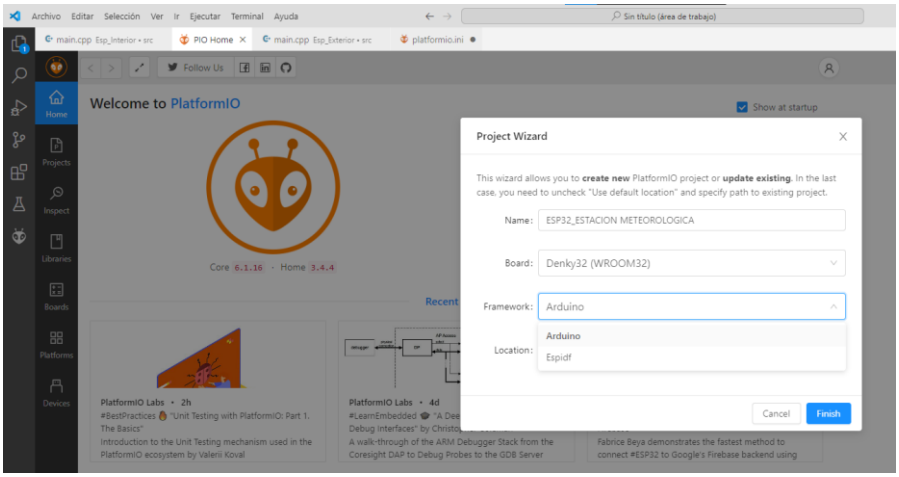


Figura 23 Creación de proyecto PlatfomrIO - Visual Studio Code

Desde PlatformIO se crean los proyectos, introduciendo el nombre del proyecto, modelo de la placa a programar [en este caso la Denky32(WROOM32)] y permite programarla tanto en Arduino como el lenguaje propio de Espressif Systems.

Este proyecto se ha programado completamente en Arduino, ya que cuenta con muchísimo más soporte y documentación debido a su gran popularidad.

4.2.2. Soportes de programación

Para la programación se ha hecho uso de distintas páginas webs como recursos de búsqueda de información, principalmente dos:

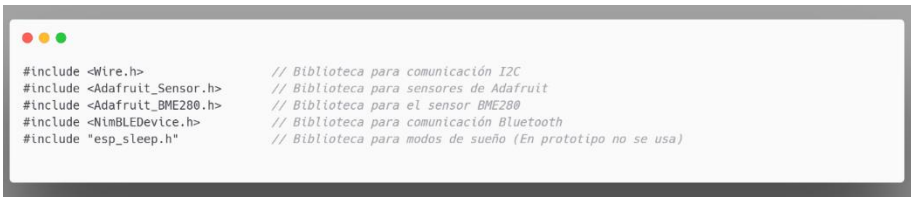
- Web oficial de Espressif Systems, proporciona una infinidad de recursos como guías perfectamente detalladas, referencias, tutoriales de programación, fichas técnicas de multitud de componentes, con el añadido de la seguridad que brinda el hecho datos los aporte la empresa desarrolladora del microcontrolador, hacen de ella la primera opción a consultar.
- Github es una web que alberga gran cantidad de información, de librerías, códigos fuente, documentación, ejemplos de programas. La propria empresa Espressif Systems cuenta con un repositorio en Github donde aporta multitud de recursos específicos.

4.3 Bibliotecas

Comenzamos los apartados de software y programación nombrando las bibliotecas utilizadas en este proyecto, especificando su uso, así como también nombrando su autor y bibliografía, ya que son elementos esenciales para el control y la implementación de funciones (Figura 24), (Figura 25).

```
#include <Arduino.h>              // Biblioteca principal de Arduino para funciones básicas
#include <NimBLEDevice.h>         // Biblioteca para manejar dispositivos Bluetooth de baja energía
#include <Adafruit_Sensor.h>      // Biblioteca de sensores de Adafruit para estandarizar la lectura de datos
#include <DHT.h>                  // Biblioteca para sensores de temperatura y humedad DHT
#include <Wire.h>                 // Biblioteca para comunicación I2C
#include <WiFi.h>                 // Biblioteca para conectar a redes WiFi
#include <WiFiClientSecure.h>     // Biblioteca para manejar conexiones WiFi seguras
#include <Adafruit_MQTT.h>        // Biblioteca para manejar la comunicación MQTT
#include <Adafruit_MQTT_Client.h> // Biblioteca para manejar clientes MQTT en conjunto con Adafruit IO
```

Figura 24 Bibliotecas ESP32_Interior



```
#include <Wire.h>                // Biblioteca para comunicación I2C
#include <Adafruit_Sensor.h>     // Biblioteca para sensores de Adafruit
#include <Adafruit_BME280.h>     // Biblioteca para el sensor BME280
#include <NimBLEDevice.h>        // Biblioteca para comunicación Bluetooth
#include "esp_sleep.h"           // Biblioteca para modos de sueño (En prototipo no se usa)
```

Figura 25 Bibliotecas ESP32_Exterior

- Arduino.h: Biblioteca principal de Arduino para funciones básicas.
- Wire.h: Biblioteca para comunicación I2C.
- esp_sleep.h: Biblioteca para modos de sueño.
- NimBLEDevice.h: Biblioteca para comunicación Bluetooth, autor: (by h2zero)
- Adafruit_Sensor.h: Biblioteca para sensores de Adafruit.
- Adafruit_BME280.h: Biblioteca para el sensor BME280 de Adafruit.
- DHT.h: Biblioteca para sensores de temperatura y humedad DHT de Adafruit
- WiFi.h: Biblioteca para conectar a redes Wi-Fi.
- Adafruit_MQTT.h: Biblioteca para manejar la comunicación MQTT de Adafruit.
- Adafruit_MQTT_Client.h: Biblioteca para manejar clientes MQTT en conjunto con Adafruit IO.

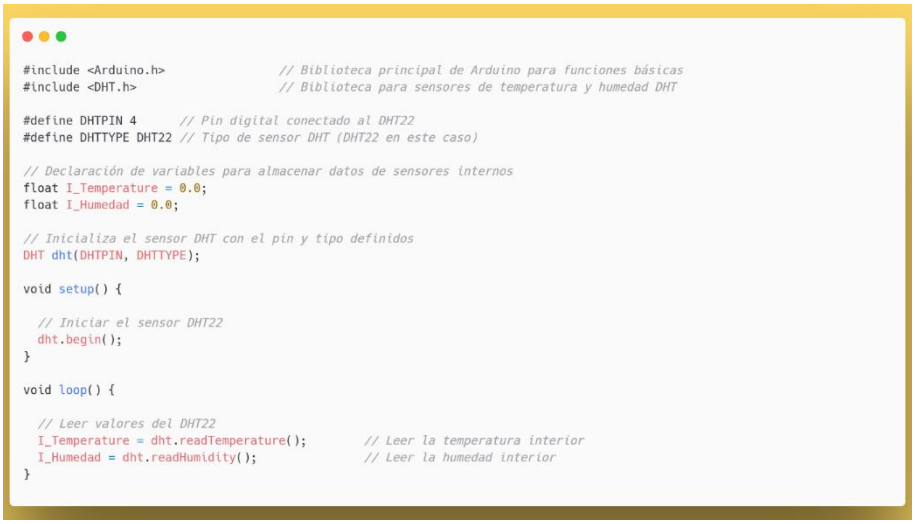
[19], [20], [21], [22]

4.4. Programación del ESP32_Interior

En este punto se va a describir y explicar detalladamente el código cargado en el ESP32_Interior. Se procederá a explicar por partes separadas el procedimiento que sigue para leer los sensores, recibir los datos del ESP32_Exterior por Bluetooth, el envío de datos al servidor web, así como también la implementación y funcionamiento de todo que engloba al bucle de control.

4.4.1. Lectura de sensores

El ESP32 lee los datos provenientes del sensor DTH22 periódicamente, es necesario incluir la biblioteca DHT.h, necesario también indicarle el tipo de sensor y el pin digital al que está conectado, en este caso el GPIO4. Declara las variables I_Temperatute e I_humedad en las que guardará los valores (Figura 26).



```
#include <Arduino.h>                // Biblioteca principal de Arduino para funciones básicas
#include <DHT.h>                    // Biblioteca para sensores de temperatura y humedad DHT

#define DHTPIN 4      // Pin digital conectado al DHT22
#define DHTTYPE DHT22 // Tipo de sensor DHT (DHT22 en este caso)

// Declaración de variables para almacenar datos de sensores internos
float I_Temperature = 0.0;
float I_Humedad = 0.0;

// Inicializa el sensor DHT con el pin y tipo definidos
DHT dht(DHTPIN, DHTTYPE);

void setup() {

  // Iniciar el sensor DHT22
  dht.begin();
}

void loop() {

  // Leer valores del DHT22
  I_Temperature = dht.readTemperature();        // Leer la temperatura interior
  I_Humedad = dht.readHumidity();               // Leer la humedad interior
}
```

Figura 26 Lectura de sensores ESP32_Interior

4.4.2. Recepción de datos por Bluetooth

El ESP32_Interior recibe los datos enviados por el ESP32_Exterior mediante BLE. Se hace uso de la biblioteca NimBLEDevice.h para establecer la comunicación y recibir los datos de los sensores del exterior. Previamente ha creado e inicializado las variables que va a recibir y las ha metido en una estructura para almacenarlas. Una vez recibida la estructura con los valores del exterior la procesa y asigna los datos recibidos a las variables correspondientes (Figura 27). (Omitida la parte de configuración de callbacks y manejo de conexiones por su extensión. Código completo en los anexos).

```
#include <NimBLEDevice.h>          // Biblioteca para manejar dispositivos Bluetooth de baja energía (Autor: H2zero)

// Definir UUIDs como constantes para la comunicación Bluetooth
const char* SERVICE_UUID = "12345678-1234-1234-1234-123456789abc";                // Definir UUIDs como constantes para la
comunicación Bluetooth
const char* CHARACTERISTIC_UUID = "87654321-4321-4321-4321-cba987654321";         // UUID de la característica principal
const char* CONTROL_CHARACTERISTIC_UUID = "abcdefab-cdef-cdef-cdef-abcdefabcdef"; // UUID para la característica de control

                                   /* Declaraciones de objetos Bluetooth */

// Declaración de variables para almacenar datos de sensores exteriores
float E_Temperature = 0.0;
float E_Humedad = 0.0;
float E_Pressure = 0.0;
float E_Altitude = 0.0;
float E_RainPercentage = 0;
float E_LdrPercentage = 0;

// Estructura para almacenar los datos del los sensores exteriores
struct DataPacket {
  float temperature;
  float humidity;
  float pressure;
  float altitude;
  float rainPercentage;
  float ldrPercentage;
};

  // Procesar los datos recibidos
  DataPacket* dataPacket = (DataPacket*)pData;

 // Asignar los datos recibidos a las variables correspondientes
  E_Temperature = dataPacket->temperature;
  E_Humedad = dataPacket->humidity;
  E_Pressure = dataPacket->pressure;
  E_Altitude = dataPacket->altitude;
  E_RainPercentage = dataPacket->rainPercentage;
  E_LdrPercentage = dataPacket->ldrPercentage;

               /*IMPLEMENTACION DE CALLBACKS Y GESTION DE CLIENTES BLUETOOTH*/

  // Enviar señal de inicio al Esp_Exterior
  pControlCharacteristic->writeValue("START");    // Escribir el valor "START" en la característica de control
  Serial.println("Señal de inicio enviada al ESP32 exterior");

  return true; ;                                  // Devolver "true" si la conexión y la configuración son exitosas
}

void setup() {

  NimBLEDevice::init("");                         // Inicializar el dispositivo Bluetooth Low Energy (BLE)

// Configuración del escaneo Bluetooth
  NimBLEScan* pScan = NimBLEDevice::getScan();    // Obtener el objeto de escaneo BLE
  pScan->setAdvertisedDeviceCallbacks(new AdvertisedDeviceCallbacks());
  pScan->setInterval(45);                         // Establecer el intervalo de escaneo a 45 ms
  pScan->setWindow(15);                           // Establecer la ventana de escaneo a 15 ms
  pScan->setActiveScan(true);                     // Habilitar el escaneo activo
  pScan->start(0, scanEndedCallback);             // Iniciar el escaneo
}

void loop() {

    // Intentar conectar al servidor si es necesario
  if (doConnect) {
    if (connectToServer()) {
      Serial.println("Conectado al servidor");  // Imprimir mensaje de conexión exitosa
    } else {
      Serial.println("Fallo en conectar al servidor");
    }
    doConnect = false;                          // Resetear la bandera de conexión
  }
}
```

Figura 27 Recepción de datos por Bluetooth ESP32_Interior

4.4.3. Envío de datos al servidor web Adafruit IO

Una vez que ha recibido procesado y tratados los datos de los sensores el ESP32_Interior envía la información al servidor web Adafruit IO mediante Wi-Fi. Para ello se utilizan las bibliotecas propias de Adafruit, así como la biblioteca nativa Wi-Fi.h. Hay que suministrarle las credenciales de una red wifi con conexión a internet para que sea capaz de transmitir los datos, así como también proporcionarle el nombre de usuario y la clave de la cuenta de Adafruit IO (Figura 28).

```
#include <WiFi.h>                   // Biblioteca para conectar a redes WiFi
#include <WiFiClientSecure.h>       // Biblioteca para manejar conexiones WiFi seguras
#include <Adafruit_MQTT.h>          // Biblioteca para manejar la comunicación MQTT
#include <Adafruit_MQTT_Client.h>   // Biblioteca para manejar clientes MQTT en conjunto con Adafruit IO

//Wifi
const char* ssid = "SSID*****";             // Nombre de la red WiFi
const char* password = "PASS*******";       // Contraseña de la red WiFi

//Adafruit IO
#define AIO_SERVER      "io.adafruit.com"                       // Servidor de Adafruit IO
#define AIO_SERVERPORT  1883                                    // Puerto del servidor (8883 para SSL)
#define AIO_USERNAME    "*****"                                 // Nombre de usuario de Adafruit IO
#define AIO_KEY         "**********************************"    // Nombre de usuario de Adafruit IO

// Crear una clase WiFiClient para conectar con el servidor MQTT
WiFiClient client;

// Configurar la clase MQTT
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

// Configurar feeds para Adafruit IO
Adafruit_MQTT_Publish E_Temperature_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/E_Temperature");
Adafruit_MQTT_Publish E_Humedad_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/E_Humedad");
Adafruit_MQTT_Publish E_Pressure_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/E_Pressure");
Adafruit_MQTT_Publish E_Altitude_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/E_Altitude");
Adafruit_MQTT_Publish E_RainValue_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/E_RainValue");
Adafruit_MQTT_Publish E_LdrValue_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/E_LdrValue");

Adafruit_MQTT_Publish I_Temperature_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/I_Temperature");
Adafruit_MQTT_Publish I_Humedadd_feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/I_Humedadd");

// Suscribirse al feed para recibir de Adafruit IO el valor de la variable Setpoint
Adafruit_MQTT_Subscribe Setpoint_feed = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/Setpoint");

// Función para establecer la conexión MQTT
void MQTT_connect() {
  int8_t ret;                                       // Variable para almacenar el estado de la conexión

  // Detener si ya está conectado.
  if (mqtt.connected()) {
    return;                                         // Salir de la función si ya está conectado
  }

  Serial.print("Conectando a MQTT... ");            // Imprimir mensaje de conexión

  uint8_t retries = 3;                              // Número de intentos de reconexión
  while ((ret = mqtt.connect()) != 0) {             // Devolverá 0 si está conectado
       Serial.println(mqtt.connectErrorString(ret));  // Imprimir mensaje de error de conexión
       Serial.println("Reintento de conexion a MQTT en 5 segundos...");  // Imprimir mensaje de conexión exitosa
       mqtt.disconnect();                           // Desconectar del servidor MQTT
       delay(5000);
       retries--;                                   // Decrementar el número de reintentos en uno
       if (retries == 0) {

         while (1);                                 // Entrar en bucle infinito
       }
  }
  Serial.println("MQTT Conectado!");                // Imprimir mensaje de conexión exitosa
}

void setup() {

  // Suscribirse al feed para establecer la consigna en Adafruit IO
  mqtt.subscribe(&Setpoint_feed);

  //Wifi

    WiFi.begin(ssid, password);                 // Iniciar la conexión a la red WiFi con el SSID y la contraseña

    // Esperar a que la conexión WiFi se establezca
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);                            // Esperar 1 segundo
        Serial.println("Conectando a WiFi...");  // Imprimir mensaje indicando que está intentando conectar
    }
    Serial.println("Conectado a WiFi");         // Imprimir mensaje indicando que la conexión se ha establecido
}

void loop() {

// Conectar a MQTT
  MQTT_connect();

  // Procesar mensajes MQTT
  Adafruit_MQTT_Subscribe* subscription;                // Variable para manejar las suscripciones MQTT

  // Leer suscripciones MQTT durante 5 segundos
  while ((subscription = mqtt.readSubscription(5000))) {
    if (subscription == &Setpoint_feed) {               // Verificar si el mensaje recibido es del feed de Setpoint
      Setpoint = atof((char*)Setpoint_feed.lastread);   // Convertir el valor recibido a float y asignarlo a Setpoint
      Serial.print("Nuevo valor de consigna: ");
      Serial.println(Setpoint);                         // Imprimir el nuevo valor de Setpoint en el monitor serial
    }
  }
  delay(30000); // Esperar 30 segundos antes de intentar leer de nuevo
}
```

Figura 28 Envío de datos al servidor web Adafruit IO ESP32_Interior

4.4.4. Bucle de control

En esta parte se controla todo lo relativo al bucle de control, esta parte se gestiona el estado de los relés basándose en la temperatura interior y en el valor consigna (Setpoint) definido por el usuario a través del servidor web.

El relé 1 (calefactor) se activará si la temperatura interior es inferior a la temperatura seleccionada por el usuario menos dos grados de histéresis de activación y se desactivará una vez llegué a la temperatura marcada por el usuario más un grado de histéresis. Mientras que el relé 2 (ventilador) se podrá en marcha cuando la temperatura seleccionada por el usuario sea mayor a la temperatura interior más dos grados de histéresis y se apagará cuando la temperatura interior baje a la seleccionada menos un grado de histéresis de desactivación (Figura 29).

```
// Variables volátiles para el control manual de los relés
volatile bool relayHeaterManual = false;    // Declara la variable volátil relayHeaterManual
volatile bool relayFanManual = false;       // Declara la variable volátil relayFanManual

float Setpoint = 25.0;                      // Valor de consigna en grados centígrados
const float activationHysteresis = 2.0;     // Histeresis para la activación
const float deactivationHysteresis = 1.0;   // Histeresis para la desactivación

// Suscribirse al feed para recibir de Adafruit IO el valor de la variable Setpoint
Adafruit_MQTT_Subscribe Setpoint_feed = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/Setpoint");

void setup() {
    // Configuración de pines para los relés y pulsadores
    pinMode(relayHeater, OUTPUT);    // Configurar el pin del relé del calefactor como salida
    pinMode(relayFan, OUTPUT);       // Configurar el pin del relé del ventilador como salida
    pinMode(button1, INPUT_PULLUP);  // Configurar el pin del pulsador 1 como entrada con resistencia pull-up
    pinMode(button2, INPUT_PULLUP);  // Configurar el pin del pulsador 2 como entrada con resistencia pull-up

    // Inicializar los relés en estado apagado (HIGH)
    digitalWrite(relayHeater, HIGH); // Establecer el relé del calefactor en estado apagado
    digitalWrite(relayFan, HIGH);    // Establecer el relé del ventilador en estado apagado

    // Suscribirse al feed para establecer la consigna en Adafruit IO
    mqtt.subscribe(&Setpoint_feed);
}

void loop() {
    // Procesar mensajes MQTT
    Adafruit_MQTT_Subscribe* subscription;  // Variable para manejar las suscripciones MQTT

    // Leer suscripciones MQTT durante 5 segundos
    while ((subscription = mqtt.readSubscription(5000))) {
        if (subscription == &Setpoint_feed) {               // Verificar si el mensaje recibido es del feed de Setpoint
            Setpoint = atof((char*)Setpoint_feed.lastread);  // Convertir el valor recibido a float y asignarlo a Setpoint
            Serial.print("Nuevo valor de consigna: ");
            Serial.println(Setpoint);                       // Imprimir el nuevo valor de Setpoint en el monitor serial
        }
    }
    // Control de relés basado en la temperatura interior y el valor de consigna Setpoint
    if (!relayHeaterManual) {
        if (I_Temperature < Setpoint - activationHysteresis) {  // Si la temperatura interior baja del valor de consigna
menos la histeresis de activación (2°C)
            digitalWrite(relayHeater, LOW);  // Activar rele 1 (calefactor)
        } else if (I_Temperature > Setpoint + deactivationHysteresis) {  // Si la temperatura interior supera el valor de
consigna más la histeresis de desactivación (1°C)
            digitalWrite(relayHeater, HIGH);  // Desactivar rele 1 (calefactor)
        }
    }
    if (!relayFanManual) {
        if (I_Temperature > Setpoint + activationHysteresis) {  // Si la temperatura interior supera el valor de consigna
más la histeresis de activación (2°C)
            digitalWrite(relayFan, LOW);  // Activar rele 2 (ventilador)
        } else if (I_Temperature < Setpoint - deactivationHysteresis) {  // Si la temperatura interior baja del valor de
consigna menos la histeresis de desactivación (1°C)
            digitalWrite(relayFan, HIGH);  // Desactivar rele 2 (ventilador)
        }
    }
    delay(30000);  // Esperar 30 segundos antes de intentar leer de nuevo
}
```

Figura 29 Bucle de control ESP32_Interior

4.4.5. Interrupciones

La (Figura 30) muestra el código perteneciente a las interrupciones de los pulsadores, para ello se han utilizado los GPIO (26 y 33) en este punto es necesario el acompañamiento de software y hardware dado que las interrupciones se han configurado de tipo “FALLING” lo cual quiere decir que los disparadores interrumpen cuando el pin cambia de “HIGH” a “LOW” por lo que el otro extremo de los pulsadores se ha tenido que conectar a GND. Los pulsadores tienen prioridad e inmediatez de modo que, si el usuario presiona uno, y automáticamente el relé correspondiente a ese pulsador cambiara de valor. Dado que la finalidad de relé 1 y 2 es totalmente la opuesta se ha implementado una lógica que impide la activación simultanea de ambos, de modo que al activarse uno de los relés desactiva automáticamente el contrario, evitando así que los sistemas de calefacción y ventilación funcionen al mismo tiempo.

```
// Declaración de las funciones antes de usarlas
void toggleRelay1();                    // Declaración de la función para alternar el estado del relé 1 (Heater)
void toggleRelay2();                    // Declaración de la función para alternar el estado del relé 2 (Fan)

// Definir como constantes los pines de los relés y pulsadores
const int relayHeater = 27;   // Pin del relé para el calefactor
const int relayFan = 14;      // Pin del relé para el ventilador
const int button1 = 26;       // Pin del pulsador 1
const int button2 = 33;       // Pin del pulsador 2

void setup() {
  // Configuración de pines para los relés y pulsadores
  pinMode(relayHeater, OUTPUT);          // Configurar el pin del relé del calefactor como salida
  pinMode(relayFan, OUTPUT);             // Configurar el pin del relé del ventilador como salida
  pinMode(button1, INPUT_PULLUP);        // Configurar el pin del pulsador 1 como entrada con resistencia pull-up
  pinMode(button2, INPUT_PULLUP);        // Configurar el pin del pulsador 2 como entrada con resistencia pull-up

  // Inicializar los relés en estado apagado (HIGH)
  digitalWrite(relayHeater, HIGH);       // Establecer el relé del calefactor en estado apagado
  digitalWrite(relayFan, HIGH);          // Establecer el relé del ventilador en estado apagado

  // Suscribirse al feed para establecer la consigna en Adafruit IO
  mqtt.subscribe(&Setpoint_feed);

  // Configurar las interrupciones para los pulsadores
  attachInterrupt(digitalPinToInterrupt(button1), toggleRelay1, FALLING);   // Configurar interrupción para el pulsador 1
  attachInterrupt(digitalPinToInterrupt(button2), toggleRelay2, FALLING);   // Configurar interrupción para el pulsador 2
}

// Función para alternar el estado del relé 1 Heater
void toggleRelay1() {
  relayHeaterManual = !relayHeaterManual; // Cambiar el estado de relayHeaterManual a su opuesto (true a false, false a
true)
  if (relayHeaterManual) {               // Verificar si relayHeaterManual es true
    relayFanManual = false;              // Desactivar ventilador manualmente
    digitalWrite(relayFan, HIGH);        // Asegurar que el ventilador esté apagado
  }
  digitalWrite(relayHeater, relayHeaterManual ? LOW : HIGH); // Activar/desactivar el calefactor basado en el nuevo estado
}

// Función para alternar el estado del relé 2 Fan
void toggleRelay2() {
  relayFanManual = !relayFanManual;         // Permutar el estado del reléFan
  if (relayFanManual) {                     // Verificar si relayFanManual es true
    relayHeaterManual = false;              // Desactivar calefactor manualmente
    digitalWrite(relayHeater, HIGH);        // Asegurar que el calefactor esté apagado
  }
  digitalWrite(relayFan, relayFanManual ? LOW : HIGH);  // Activar/desactivar el calefactor basado en el nuevo estado
}
```

Figura 30 Interrupciones ESP32_Interior

4.5. Programación del ESP32_Exterior

En los siguientes apartados se describe el comportamiento del ESP32_Exterior el cual le permite establecer conexión con el ESP32_Interior, recolectar los datos de los sensores y enviarlos al interior periódicamente de manera eficiente.

4.5.1. Lectura de sensores

Punto crucial, tiene que ser capaz de monitorizar todas variables del ambiente, para ello lee los valores que le llegan del BME280, FC-37 y la LDR. Para este apartado es necesaria la implementación de las bibliotecas específicas de cada sensor. Se normalizan los valores del FC-37 y de la LDR para que posteriormente muestre un porcentaje de 0 a 100 facilitando así su interpretación. Además, al valor de presión proveniente del BME280 se le aplica un factor de conversión para de este modo calcular la altura a la que se encuentra respecto del nivel del mar (Figura 31).

```
#include <Wire.h>                 // Biblioteca para comunicación I2C
#include <Adafruit_Sensor.h>      // Biblioteca para sensores de Adafruit
#include <Adafruit_BME280.h>      // Biblioteca para el sensor BME280

#define LDR_PIN 33                //Pin para LDR
#define RAIN_SENSOR_PIN 35        //Pin para sensor de lluvia
#define LED_PIN 17                //Pin para LED Azul, indica que acaba de enviar datos por bluetooth
#define OPTO_PIN 4                // Pin para controlar el optoacoplador

Adafruit_BME280 bme;                                    // Objeto para el sensor BME280
NimBLEServer* pServer = nullptr;                        // Puntero al servidor Bluetooth
NimBLECharacteristic* pCharacteristic = nullptr;        // Puntero a la característica principal
NimBLECharacteristic* pControlCharacteristic = nullptr; // Puntero a la característica de control
bool startDataCollection = false;                       // Flag para indicar si se inicia la recopilación de datos

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

   // BME280
  while (!Serial);                                  // Espera a que el puerto serial esté listo

  if (!bme.begin(0x76)) {                           // Direccion predeterminada si CSB Y SD0 conectados a GND -> 0x76
    Serial.println("No se encuentra sensor BME280"); // Mensaje de error si no se encuentra el sensor
    while (1);                                      // Entra en un bucle infinito si no se encuentra el sensor
  }
  pinMode(RAIN_SENSOR_PIN, INPUT);                  // Configura el pin del sensor de lluvia como entrada
  pinMode(LDR_PIN, INPUT);                          // Configura el pin del sensor LDR como entrada
  pinMode(LED_PIN, OUTPUT);                         // Configura el pin del LED como salida
  delay(10000);
}

void loop() {

  // Si la recopilación de datos ha comenzado, leer valores de los sensores
  if (startDataCollection) {
    // Leer valores del BME280
    float temperature = bme.readTemperature();          // Leer la temperatura del BME280
    float humidity = bme.readHumidity();                // Leer la humedad del BME280
    float pressure = bme.readPressure() / 100.0F;       // Leer la presión y convertir a hPa
    float altitude = bme.readAltitude(1013.25);         // Calcular la altitud asumiendo una presión a nivel del mar de
1013.25 hPa

    // Leer valor del sensor de lluvia
    int rainValue = analogRead(RAIN_SENSOR_PIN);
    // Normalizar el valor del detector de lluvia a un porcentaje
    float rainPercentage = ((4095.0 - rainValue) / 4095.0) * 100.0;

    // Leer valor de la LDR
    int ldrValue = analogRead(LDR_PIN);
    // Normalizar el valor de la LDR a un porcentaje
    float ldrPercentage = (ldrValue / 4035.0) * 100.0;  //Maximo 4035 enfocando con linterna, 3818 luz natural
  }
```

Figura 31 Lectura de los sensores ESP32_Exterior

4.5.2. Envío de datos por Bluetooth

El ESP32_Exterior se mantiene a la espera de establecer conexión Bluetooth con el microcontrolador interior. Una vez se han conectado con éxito el ESP32_Interior le envía la señal de comienzo, una vez recibida, el ESP32_Exterior comienza con su rutina cíclica de recolecta de datos y envío de estos al interior. Se puede apreciar visualmente cada vez que el dispositivo envía datos al interior ya que inmediatamente después de realizar el envío, se activa un LED azul durante un segundo (Figura 32). (Omitida la parte de configuración de callbacks y manejo de conexiones por su extensión. Código completo en los anexos).

```
#include <NimBLEDevice.h>          // Biblioteca para comunicación Bluetooth

#define LED_PIN 17                 //Pin para LED Azul, indica que acaba de enviar datos por bluetooth

// Definir UUIDs como constantes para la comunicación Bluetooth
const char* SERVICE_UUID = "12345678-1234-1234-1234-123456789abc";               // Definir UUIDs como constantes
const char* CHARACTERISTIC_UUID = "87654321-4321-4321-4321-cba987654321";        // UUID de la característica principal
const char* CONTROL_CHARACTERISTIC_UUID = "abcdefab-cdef-cdef-cdef-abcdefabcdef"; // UUID para la característica de control

    if (value == "START") {                                     // Si el valor es "START"
      startDataCollection = true;                               // Iniciar la recopilación de datos
      Serial.println("Iniciando recopilación de datos");        // Imprimir mensaje en el monitor serial
    }

void setup() {

  // Bluetooth
  NimBLEDevice::init("ESP32_Exterior");                         // Inicializa el dispositivo Bluetooth con nombre
"ESP32_Exterior"
  pServer = NimBLEDevice::createServer();                       // Crea un servidor Bluetooth
  NimBLEService* pService = pServer->createService(SERVICE_UUID);    // Crea un servicio Bluetooth con el UUID definido

  // Crea una característica para notificaciones
  // Crea una característica de control para escritura
  // Establece las devoluciones de llamada para la característica de control


  pService->start();
  NimBLEAdvertising* pAdvertising = NimBLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->start();
  Serial.println("El Bluetooth ha sido inicializado");
}

void loop() {

        /* Si la recopilación de datos ha comenzado es decir si ha recibido la señal "start" del esp_Interior,
           entonces leer valores de los sensores*/
  if (startDataCollection) {
    // Leer valores del BME280
  }
    // Estructura para almacenar los datos de los sensores
    struct DataPacket {
      float temperature;
      float humidity;
      float pressure;
      float altitude;
      float rainPercentage;
      float ldrPercentage;
    } dataPacket;

    // Asignar los valores leídos a las respectivas variables en el paquete de datos
    dataPacket.temperature = temperature;
    dataPacket.humidity = humidity;
    dataPacket.pressure = pressure;
    dataPacket.altitude = altitude;
    dataPacket.rainPercentage = rainPercentage;
    dataPacket.ldrPercentage = ldrPercentage;

    pCharacteristic->setValue((uint8_t*)&dataPacket, sizeof(dataPacket));   // Establece el valor de la característica con
los datos del paquete
    pCharacteristic->notify();                                              // Envía una notificación a los dispositivos
suscritos con el nuevo valor

    // Activar LED durante 1 segundo (Indicando que se ha hecho el envio)
    digitalWrite(LED_PIN, HIGH);
    delay(1000);
    digitalWrite(LED_PIN, LOW);

    delay(30000);
}
```

Figura 32 Envío de datos por Bluetooth ESP32_Exterior

4.5.3. Gestión de la alimentación de los sensores

Dado que la mayor parte del tiempo no se están tomando datos de los sensores el microcontrolador corta la alimentación de éstos por medio del optoacoplador. Este permite el paso de la alimentación cuando la GPIO4 está en “HIGH”. Para optimizar el consumo energético se activa la GPIO4 medio segundo antes de realizar las mediciones y se desactiva justo antes de empezar la espera de 30 segundos (Figura 33).

```
#define OPTO_PIN 4               // Pin para controlar el optoacoplador

void setup() {

  // Configurar el pin del optoacoplador como salida
  pinMode(OPTO_PIN, OUTPUT);

  // Encender el optoacoplador para alimentar los sensores
  digitalWrite(OPTO_PIN, HIGH);
  delay(500);

  // Inicializacion del bus I2C y el sensor BME280

void loop() {

  // Encender el optoacoplador para alimentar los sensores
  digitalWrite(OPTO_PIN, HIGH);
  delay(500);

 // Reinicializar el bus I2C y el sensor BME280 (Debido a que entre mediciones se le quita la alimentacion)

    // Leer valores del BME280
    // Leer valor del sensor de lluvia
    // Leer valor de la LDR
    // Estructura para almacenar los datos de los sensores
    // Asignar los valores leídos a las respectivas variables en el paquete de datos
    // Establece el valor de la característica con los datos del paquete
    // Envía una notificación a los dispositivos suscritos con el nuevo valor

    // Activar LED durante 1 segundo (Indicando que se ha hecho el envio)
    digitalWrite(LED_PIN, HIGH);
    delay(1000);
    digitalWrite(LED_PIN, LOW);

    // Apagar el optoacoplador para cortar la alimentación de los sensores
    digitalWrite(OPTO_PIN, LOW);

    delay(30000);

  }
}
```

Figura 33 Gestión de la alimentación de los sensores ESP32_Exterior

4.6 Configuración del servidor web Adafruit IO

En este apartado se va a detallar como se configura el entorno web de Adafruit IO para que sea capaz de recibir, enviar y mostrar todos los datos para el correcto funcionamiento de la estación meteorológica.

4.6.1 Creación de cuenta y obtención de credenciales

El primer paso una vez creada la cuenta es ir al logotipo de la llave, ahí se encuentran las credenciales necesarias para la comunicación del microcontrolador con Adafruit IO (Figura 34). Vienen ya preparadas para ponerlas en diferentes tipos de lenguajes de programación.

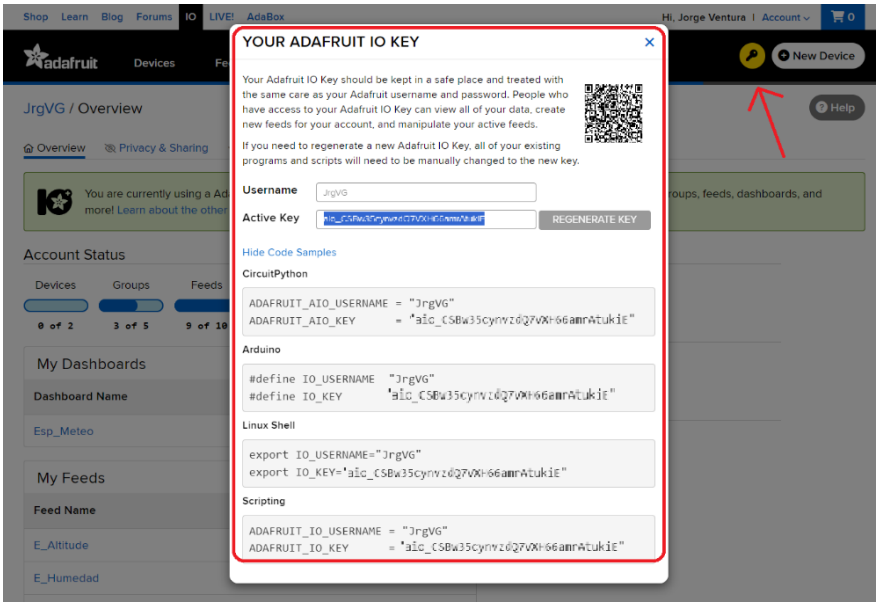


Figura 34 Creación de cuenta y obtención de credenciales Adafruit IO

4.6.2 Integración de credenciales en ESP32_Interior

Una vez obtenidas las credenciales de Adafruit IO es hora de integrarlas en el código fuente del microcontrolador. Simplemente se pegan en Visual Studio Code junto con el servidor de Adafruit y el puerto utilizado (Figura 35).

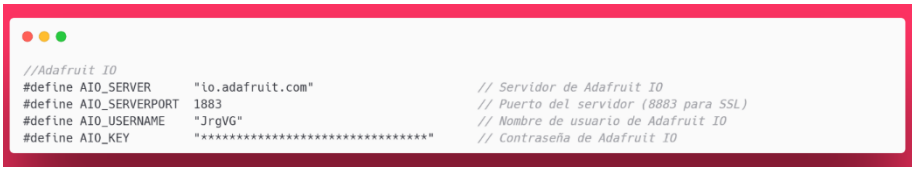


Figura 35 Integración de credenciales en ESP32_Interior

4.6.3 Recepción de datos en Adafruit IO

Una vez introducidas y cargadas en el ESP32_interior está todo preparado para encender el microcontrolador y que el servidor web reciba los datos de los sensores continuamente. Para visualizar y configurar los datos que recibe se hace desde la sección “Feeds”, aquí se pueden crear organizar, separar y renombrar las variables enviadas por el microcontrolador (Figura 36).

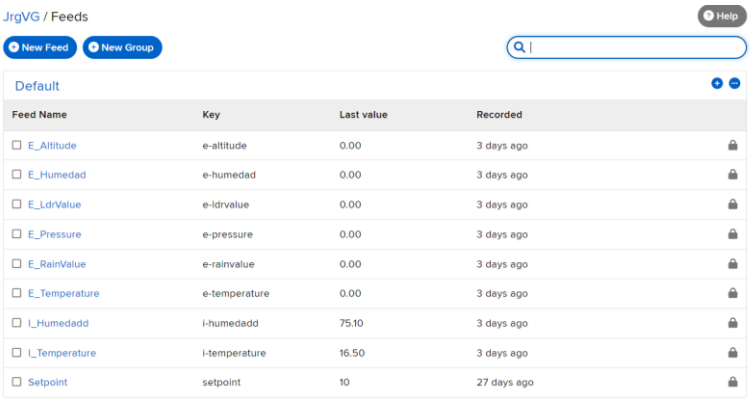


Figura 36 Pestaña Feeds Adafruit IO

4.6.4 Creación de paneles interactivos

Finalmente, en “Dashboards” se crean los paneles interactivos en los que se muestran todas las variables de forma muy visual. Para ello se van creando nuevos bloques con las variables de deseadas. Permite elegir entre gran cantidad de modelo de bloques para representar la información de manera muy gráfica, permitiendo así que el bloque se adapte perfectamente a la información que se desea representar.

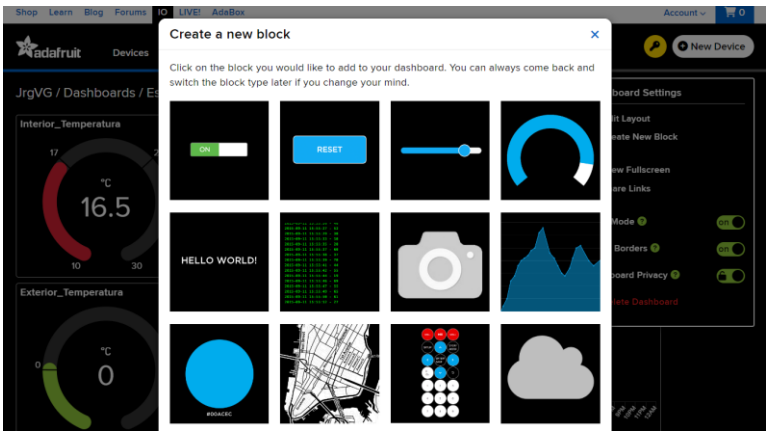


Figura 37 Creación de paneles interactivos Adafruit IO

5. Resultados

Finalmente, con ambos dispositivos con su conexionado montado, con la programación cargada en los controladores y con la web de Adafruit perfectamente configurada se procede a presentar los resultados tras un periodo de prueba.

A continuación, la (Figura 38) y la (Figura 39) muestran el resultado final del prototipo ensamblado:

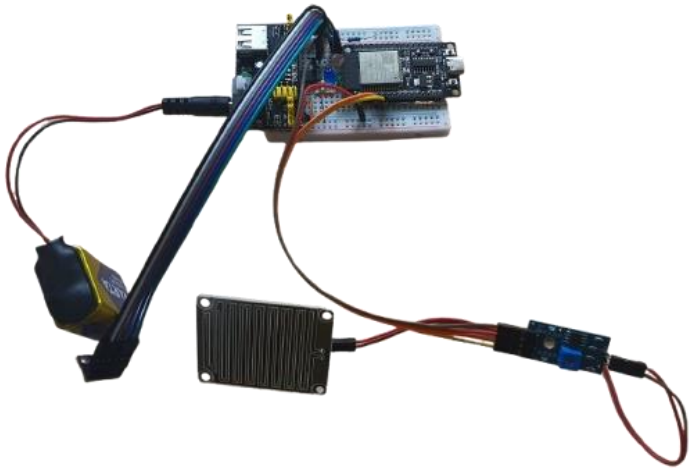


Figura 38 Prototipo ESP32_Exterior

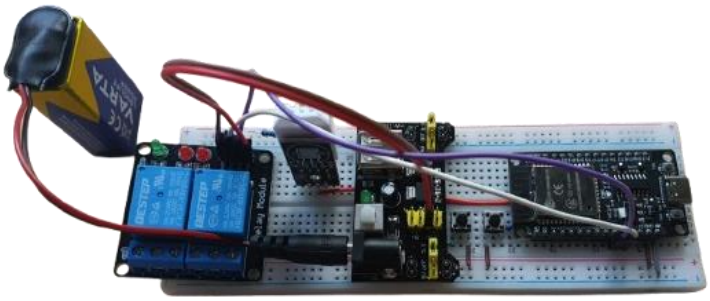


Figura 39 Prototipo ESP32_Interior

5.1 Monitorización de datos

El servidor web recibe correctamente cada 30 segundos todos los datos necesarios recolectados por los microcontroladores interior y exterior, permite visualizar a tiempo real de manera muy gráfica todos los datos monitoreados del ambiente, así como también se les han introducido a las variables del interior unos rangos los cuales si se

sale de ahí los resalta haciéndolo rápidamente detectable por el usuario. La (Figura 40) muestra el resultado final:

Figura 40 Dashboard Adafruit IO

Con este diseño visualmente atractivo el usuario puede visualizar rápidamente los datos actuales y los históricos. Las gráficas de la derecha permiten monitorear las mediciones pasadas de humedad y presión de los últimos 30 días, facilitando un análisis detallado de las condiciones ambientales presentes y pasadas.

5.2 Interacción con el usuario

El usuario puede modificar en cualquier momento desde cualquier lugar con conexión a internet el valor de la consigna accediendo al servidor web con sus credenciales. Siempre que se cumplan las condiciones específicas del valor de la consigna y de la temperatura interior se activarán o desactivarán los relés correspondientes dado que el bucle de control se ha comprobado de manera exhaustiva y funciona correctamente. Asimismo, el usuario puede cambiar el estado de los relés manualmente mediante los

pulsadores asociados a cada uno de ellos haciendo que permuten de estado de manera inmediata.

5.3. Pruebas y validación

En este apartado se describen las pruebas realizadas para validar el correcto funcionamiento del sistema. Se incluyen pruebas de comunicación entre los ESP32, pruebas de envío de datos a Adafruit IO y pruebas del bucle de control.

5.3.1. Pruebas de comunicación entre ESP32

Para validar el funcionamiento correcto de las comunicaciones entre dispositivos se hizo que el ESP32_Exterior imprimiera por pantalla los datos de los sensores conforme los iba recopilando, simultáneamente se hizo que el ESP32_Interior mostrará los datos que iba recibiendo del exterior. Tras comparar ambos conjuntos de datos se comprobó que la comunicación Bluetooth entre dispositivos funciona perfectamente ya que hubo una correlación exacta durante todo el periodo de análisis. Con esta prueba se deja constancia de que los datos se transmiten de manera precisa y confiable confirmando la eficacia de la comunicación implementada.

Para comprobar la distancia máxima que pueden estar separados ambos dispositivos se mantuvo conectado el ESP32_Interior al pc para monitorear los datos que van llegando del microcontrolador del exterior, a medida que iban llegando correctamente se fue aumentando la distancia entre ambos dispositivos en un entorno sin paredes ni obstáculos. De este modo se comprobó que la comunicación funciona perfectamente hasta los 11 metros, superada esta distancia el sistema comenzó a presentar fallos en la transmisión de datos. Con esta prueba se ha demostrado que la distancia máxima efectiva que pueden estar separados ambos dispositivos es de entorno a los 10 metros, en condiciones ideales sin obstáculos ni paredes.

5.3.2. Pruebas de envío de datos a Adafruit IO

Para validar el correcto envío de datos a Adafruit IO se le ha realizado una prueba similar a la del apartado anterior, consistiendo en que antes del envío de datos al servidor web, se ha procedido a mostrar por pantalla los valores de todas las variables a enviar, posteriormente se ha verificado que los valores que recibe Adafruit IO coinciden exactamente con los que muestra por pantalla el ESP32_Interior. De este modo que se confirma el correcto funcionamiento de la comunicación Wi-Fi y la integración con la plataforma Adafruit IO siendo capaz de enviar y recibir la información de manera confiable, sin errores ni alteraciones.

5.3.3. Pruebas del bucle de control

Se ha sometido al bucle de control a una observación exhaustiva para comprobar y testar su funcionamiento. Durante estas pruebas se ha verificado que el sistema responde de manera adecuada a los cambios de consigna y a la temperatura interior. Los relés se activan y desactivan de manera precisa cumpliendo así con las condiciones programadas.

Se ha observado que, esporádicamente al presionar alguno de los dos pulsadores el relé cambia de estado dos veces. Este comportamiento puede ser debido a la presencia de rebotes en el pulsador, pequeñas oscilaciones eléctricas que surgen cuando el pulsador es presionado. De este modo el microcontrolador recibe múltiples pulsos en lugar de solo uno. Aunque no se considera un problema significativo es un problema a solucionar en posteriores versiones.

5.4 Conclusiones

Las pruebas realizadas han demostrado que el sistema funciona de manera eficiente y confiable, se ha verificado que los datos recopilados tanto en el interior como en el exterior llegan correctamente hasta el servidor web donde el usuario los visualiza, garantizando que los datos ahí representados son fiables y exactos. También se ha comprobado el correcto funcionamiento del bucle de control, tanto la parte automática como la activación manual de él. De este modo se concluye con éxito el proceso de validación del sistema, superando con existo las expectativas establecidas.

6. Conclusiones

El presente proyecto ha conseguido cumplir con todos los objetivos propuestos inicialmente, se ha logrado monitorizar y gestionar las variables ambientales de manera precisa y fiable. Durante el desarrollo y su posterior implementación se han abordado y superado multitud de retos técnicos lo cual ha resultado con una solución completa y robusta.

Se propuso como objetivo que el sistema debía de ser eficiente. Ahora el sistema ha demostrado contar con una elevada eficiencia en la recolección de datos tanto en los tomados en el interior como en el exterior. Permite al usuario acceder remotamente al sistema para visualizar los datos en tiempo real, así como también le permite controlar la temperatura interior estableciendo el valor deseado de temperatura en el servidor web, así como también permite la activación manual de los sistemas de climatización mediante los pulsadores conectados el dispositivo interior.

Una de las principales ventajas que tiene el diseño de este sistema es la fácil escalabilidad con la que cuenta, ya que permite la integración de sensores adicionales, así como también de más microcontroladores conectados al ESP32_Interior, permitiendo así monitorear diferentes zonas sin la necesidad de realizar grandes cambios a la infraestructura existente.

Se ha demostrado que la elección de utilizar dos microcontroladores ESP32 como cerebro del proyecto ha sido una decisión acertada, ya que estos dos microcontroladores han ofrecido todas las capacidades necesarias para satisfacer todos los objetivos planteados. Cuentan con la integración de comunicaciones Bluetooth y Wi-Fi y han demostrado contar con el almacenamiento y la potencia de procesamiento necesarias para ejecutar el código requerido, de hecho, cuentan con margen de sobra para la implementación de futuras mejoras. Todo ello por un precio extremadamente competitivo hace que el ESP32 haya sido la mejor opción posible.

En conclusión, este resultado establece unas buenas bases para futuras aplicaciones y mejoras, asegurando así que el sistema continue ofreciendo monitoreo y control ambiental efectivo y confiable.

6.1 Líneas de mejoras

Superar la enorme limitación de únicamente poder trabajar con 10 variables o “Feeds” por el uso de la versión gratuita de Adafruit IO es algo crucial, ya que sin esta restricción la cantidad de variables que podría manejar el sistema es muchísimo mayor, pudiendo implementar más sensores, más microcontroladores conectados al ESP32_Interior, así como también más bucles de control, ofreciendo al usuario la capacidad de gestionar mayor número de variables. Superar esta limitación es algo vital, ambos microcontroladores cuentan con potencia y almacenamiento suficiente como para manejar programas más complejos con mayor cantidad de datos y parámetros.

Otra línea de mejora es la implementación de los modos de suspensión del microcontrolador, con el añadido de que uno de los puntos en los que destaca el ESP32 es en contar con uno de los modos de menor consumo de entre los microcontroladores, ya que en su modo más restrictivo tiene un consumo medio aproximado de 10 µA.

```
#include "esp_sleep.h"      // Biblioteca para modos de sueño

void setup() {
  // Inicializacion del Bluetooth
  // Inicializacion del bus I2C y el sensor BME280
 }
void loop() {
      // Reinicializacion del bus I2C y el sensor BME280

      // Leer valores del BME280
      // Leer valor del sensor de lluvia
      // Leer valor de la LDR

      // Enviar datos por Bluetooth
      // Activar LED durante 1 segundo

  // Configurar el temporizador para el modo de suspensión profunda
  esp_sleep_enable_timer_wakeup(30 * 1000000); // 30 segundos en microsegundos

  // Entrar en modo de suspensión profunda
  esp_deep_sleep_start();
}
```

Figura 41 Suspensión profunda ESP32_Exterior

Aunque esta funcionalidad de implementó y se testó, pero finalmente se optó por dejarla comentada ya que, a causa de los cortos intervalos entre mediciones propios de ser un prototipo, implementarla entorpecía el sistema (Figura 41).

Explotar este punto fuerte del ESP32 sería altamente beneficioso. En versiones posteriores con mayor tiempo entre mediciones, sería muy conveniente que el microcontrolador entrara en suspensión profunda aumentando así significativamente la eficiencia y mejorando su autonomía.

Bibliografía

[1] «IoT-Monitoreo Ambiental Basado: Tipos y casos de uso». [En línea]. Disponible en: https://es.digi.com/blog/post/iot-based-environmental-monitoring

[2] «Esp32 - **Electrodaddy**». [En línea]. Disponible en: https://electrodaddy.com/esp32/

[3] «Placa de desarrollo ESP32- AliExpress 44», aliexpress. [En línea]. Disponible en: //es.aliexpress.com/item/1005006336964908.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp=

[4] «ESP32», *Wikipedia, la enciclopedia libre*. 13 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=ESP32&oldid=162434469

[5] «esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf». [En línea]. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

[6] D. Hercog, T. Lerher, M. Truntič, y O. Težak, «Design and Implementation of ESP32-Based IoT Devices», *Sensors*, vol. 23, n.º 15, p. 6739, jul. 2023, doi: 10.3390/s23156739.

[7] «BST-BME280-DS002-1509607.pdf». [En línea]. Disponible en: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf?srsltid=AfmBOopG7Nm0eaYAtNhFuJ4zLGkh7zd8LGX2BoWR3qxfqR4yTWtyG9PN

[8] A. Singh, «Rain Sensor Module with Arduino Working and Applications», Hackatronic. [En línea]. Disponible en: https://www.hackatronic.com/rain-sensor-module-with-arduino-working-and-applications/

[9] «yl-83-rain-detector-datasheet_low.pdf». [En línea]. Disponible en: https://urolakostapk.wordpress.com/wp-content/uploads/2016/10/yl-83-rain-detector-datasheet_low.pdf

[10] «Fotorresistor», *Wikipedia, la enciclopedia libre*. 1 de septiembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Fotorresistor&oldid=162192185

[11] www.alldatasheet.com, «DHT22 datasheet(1/10 Pages) ETC2». [En línea]. Disponible en: http://www.alldatasheet.com/html-pdf/1132459/ETC2/DHT22/109/1/DHT22.html

[12] «Button.pdf». [En línea]. Disponible en: https://wiki-content.arduino.cc/documents/datasheets/Button.pdf

[13] «Módulo de relé de 3V, 2CH - AliExpress 13», aliexpress. [En línea]. Disponible en: //es.aliexpress.com/item/1005003750654499.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp=

[14] «Led», *Wikipedia, la enciclopedia libre*. 15 de noviembre de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Led&oldid=163591591

[15] «mb102-ps.pdf». [En línea]. Disponible en: https://www.handsontec.com/dataspecs/mb102-ps.pdf

[16] «4n35.pdf». [En línea]. Disponible en: https://www.vishay.com/docs/81181/4n35.pdf

[17] «Bluetooth de baja energía (BLE) en ESP32». [En línea]. Disponible en: https://www.cabotinoso.es/conceptos-basicos-de-esp32-bluetooth-de-baja-energia-ble/

[18] «ESP32 WIFI Modos – Prometec». [En línea]. Disponible en: https://www.prometec.net/wifi-modo-acces-point/

[19] *adafruit/Adafruit_MQTT_Library*. (19 de noviembre de 2024). C++. Adafruit Industries. [En línea]. Disponible en: https://github.com/adafruit/Adafruit_MQTT_Library

[20] *adafruit/Adafruit_Sensor*. (24 de noviembre de 2024). C++. Adafruit Industries. [En línea]. Disponible en: https://github.com/adafruit/Adafruit_Sensor

[21] *adafruit/DHT-sensor-library*. (24 de noviembre de 2024). C++. Adafruit Industries. [En línea]. Disponible en: https://github.com/adafruit/DHT-sensor-library

[22] h2zero, *h2zero/NimBLE-Arduino*. (26 de noviembre de 2024). C. [En línea]. Disponible en: https://github.com/h2zero/NimBLE-Arduino