UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA MECÁNICA

"DISPOSITIVO DE MONITORIZACIÓN AMBIENTAL BASADO EN SENSORES PARA LA MEDICIÓN DE TEMPERATURA Y HUMEDAD CON GEOLOCALIZACIÓN"

TRABAJO FIN DE GRADO

Junio -2023

AUTOR: Marco León Cardona

DIRECTOR: Oscar Martínez Bonastre

ÍNDICE

1. INTRODUCCIÓN

Este proyecto pretende la creación de un dispositivo capaz de abarcar el estudio, la monitorización y posibles opciones de control a un abanico de problemas actuales muy ligados a los fenómenos ambientales, en concreto, aquellos campos donde la medición de valores como temperatura y humedad puedan ser útiles para detectar y analizar cambios en el entorno.

Para ello, será necesario unificar diferentes ideas, conceptos y tecnologías que nos ayuden a darle sentido al proyecto.

Uno de los elementos clave para el éxito de este proyecto será la integración de los sensores y la capacidad de transmitir datos geolocalizados para que, de esta forma, pueda ser cargada a una plataforma centralizada donde puedan ser visualizados, estudiados y finalmente, aportar respuestas a los problemas detectados.

1.1 ANTECEDENTES. ESTADO DEL ARTE.

En el ámbito de sensores, más concretamente los especializados en la medición de temperatura y humedad, se han experimentado grandes avances en los últimos tiempos. Los avances tecnológicos han permitido el desarrollo de sensores más precisos y fiables, así como de dispositivos más compactos y fáciles de usar.

A continuación, se presentan algunos de los avances destacados:

- Sensores cada vez más precisos: Los sensores en ámbito general han mejorado en términos de precisión y estabilidad. El desarrollo de sensores de alta calidad que ofrecen lecturas más exactas y consistentes da un valor crucial en aplicaciones donde se requiere un control preciso del ambiente.

- Tecnología inalámbrica y conectividad: La incorporación de tecnología inalámbrica, ha facilitado la transmisión de datos en tiempo real desde los dispositivos de medición hacia sistemas de gestión o dispositivos móviles. Esto permite la monitorización remota y una supervisión continua de la temperatura y humedad en tiempo real.

- Integración con sistemas de gestión de datos: Los dispositivos de medición de temperatura y humedad pueden integrarse con sistemas de gestión de datos, como bases de datos en la nube o plataformas de IoT (Internet de las cosas). Esto facilita el almacenamiento, análisis y visualización de los datos recopilados, lo que ayuda a identificar patrones, tendencias y realizar ajustes oportunos en función de los resultados obtenidos.

- Miniaturización y portabilidad: Los dispositivos de medición de temperatura y humedad se han vuelto cada vez más compactos y portátiles, lo que permite su uso en una amplia gama de aplicaciones. Estos dispositivos son más fáciles de transportar, instalar y utilizar, lo que ha ampliado su disponibilidad y accesibilidad.

- Uso de tecnologías avanzadas: Algunos dispositivos de medición de temperatura y humedad están adoptando tecnologías avanzadas como la inteligencia artificial y el aprendizaje automático. Estas tecnologías permiten el análisis y la interpretación automatizada de los datos recopilados, lo que facilita la detección temprana de anomalías, la optimización de procesos y la toma de decisiones más precisas.

A continuación, es necesario clarificar el concepto mencionado “Internet de las cosas” (IoT, por sus siglas en inglés, Internet Of Things). El concepto de IoT [1] tuvo su origen en 1990 cuando John Romkey creó el primer objeto conectado a internet, una tostadora que se encendía y apagaba en remoto. Actualmente, IoT es una de las nueve tecnologías que engloban la Industria 4.0, la cuarta revolución industrial.

Figura 1. Tecnologías 4.0

La tecnología IoT en concreto busca la conexión de cualquier objeto a internet para permitir su control en tiempo real y desde cualquier lugar. Básicamente, se trata de la conexión de dispositivos físicos a la red, permitiendo que estos intercambien información y realicen tareas de manera autónoma o colaborativa. La idea detrás del Internet de las cosas es la de convertir objetos comunes en "inteligentes" al dotarlos de sensores, actuadores y capacidad de comunicación. Estos dispositivos pueden ser desde electrodomésticos, accesorios como relojes, vehículos y sistemas de control ambiental, hasta dispositivos industriales, sensores ambientales y sistemas de monitorización. La interconexión de estos dispositivos habilita una amplia gama de aplicaciones y beneficios. Por ejemplo, en el ámbito doméstico, podemos tener hogares inteligentes donde los electrodomésticos se comunican entre sí para ofrecer mayor comodidad y eficiencia energética.

En el sector de la salud [2], se pueden crear dispositivos portátiles que monitorean constantemente la salud de los pacientes y envían alertas a los médicos en caso de cualquier anomalía. En la industria [3], los sensores conectados pueden proporcionar información en tiempo real sobre el rendimiento y el mantenimiento de las máquinas, lo que permite una gestión más eficiente de los recursos y una reducción de costos.

En conclusión, el Internet de las cosas representa una nueva frontera tecnológica que promete transformar diversos sectores y mejorar la vida cotidiana. La conexión de dispositivos físicos a través de Internet abre un mundo de posibilidades, permitiendo una mayor automatización, eficiencia y toma de decisiones basada en datos. Sin embargo, es esencial abordar los desafíos asociados con la seguridad y la privacidad para garantizar el uso responsable y beneficioso de esta tecnología.

Por último, enfocándonos en nuestro proyecto en concreto, el Internet de las cosas (IoT) en relación con sensores ambientales ha revolucionado la forma en la que monitoreamos y comprendemos nuestro entorno. Mediante la integración de sensores ambientales en dispositivos conectados a la red, se puede lograr recopilar datos precisos y en tiempo real.

Estos sensores ambientales conectados permiten una supervisión continua de las condiciones ambientales en diferentes entornos.

El IoT ambiental ofrece una serie de ventajas significativas [4]. En primer lugar, permite un monitoreo más eficiente y detallado del medio ambiente, proporcionando datos en tiempo real que ayudan a identificar patrones, tendencias y anomalías. Esto permite una respuesta más rápida y precisa ante problemas ambientales.

Además, el IoT ambiental facilita la recopilación de datos a gran escala. Esto proporciona una visión más completa y detallada de las condiciones ambientales, lo que resulta especialmente útil para la toma de decisiones.

1.2 OBJETIVOS DEL PROYECTO.

El propósito del proyecto es desarrollar un dispositivo que pueda recopilar y monitorear datos ambientales, específicamente temperatura y humedad, utilizando los sensores seleccionados que se describirán más adelante. El dispositivo también tendrá la capacidad de geolocalizarse a sí mismo y registrar la ubicación geográfica de las mediciones.

Por ende, el enfoque primordial de este proyecto es proporcionar un medio eficiente y suficientemente preciso para recopilar información ambiental relevante en aquellas ubicaciones por las que se desplace. Al monitorear la temperatura y la humedad en tiempo real, el dispositivo permitirá obtener datos valiosos sobre las condiciones ambientales en un área específica.

Dicho esto, los objetivos asociados con este proyecto incluirán:

1. Control y seguimiento ambiental: Proporcionar a los usuarios una herramienta para monitorear y registrar los niveles de temperatura y humedad en el entorno en tiempo real.

2. Alertas y notificaciones: Configurar el dispositivo para enviar alertas o notificaciones automáticas cuando los valores de temperatura o humedad excedan umbrales predefinidos.

3. Geolocalización: Registrar la ubicación geográfica de cada medición ambiental realizada por el dispositivo.

4. Análisis de datos: Recopilar y almacenar los datos de temperatura y humedad en una base de datos para su posterior análisis.

5. Optimización del consumo de energía: Diseñar el dispositivo de manera eficiente en términos de consumo de energía para garantizar una duración óptima de la batería.

1.3. ALCANCE Y PLANIFICACIÓN DEL PROYECTO.

Teniendo claros los objetivos del proyecto, se mencionará a continuación el alcance esperado del mismo:

- Diseño y desarrollo de un dispositivo compacto y portátil de monitorización ambiental.
- Integración de sensores de temperatura y humedad en el dispositivo.
- Implementación de tecnología de geolocalización para rastrear y registrar la ubicación de las mediciones.
- Desarrollo de una interfaz de usuario intuitiva para visualizar los datos recolectados.

El cual tendrá que cumplir los siguientes requisitos:

- Sensibilidad y precisión adecuadas de los sensores de temperatura y humedad.
- Capacidad de comunicación inalámbrica para enviar los datos recolectados.
- Duración de la batería suficiente para un uso óptimo sin recarga.
- Interfaz de usuario intuitiva y fácil de usar.

Respecto a la planificación del proyecto, a continuación, se mencionan los pasos seguidos de manera cronológica, separando la totalidad del trabajo por etapas:

1. Definición y análisis de requisitos:

- Documentación de los requisitos técnicos y funcionales del dispositivo.
- Análisis de las tecnologías de sensores y geolocalización disponibles en el mercado.

2. Diseño y desarrollo del hardware:

- Selección de los sensores adecuados y montaje del dispositivo.
- Pruebas de funcionalidad y precisión de los sensores.

3. Desarrollo del software:

- Diseño de la interfaz de usuario y la estructura de la base de datos.
- Desarrollo del software de adquisición y procesamiento de datos.
- Pruebas de integración y verificación de la funcionalidad del software.

4. Integración hardware-software:

- Integración del hardware y el software del dispositivo.
- Pruebas de comunicación y sincronización entre los componentes.
- Verificación de la precisión y confiabilidad de las mediciones.

5. Pruebas y validación:

- Realización de pruebas del dispositivo en diferentes entornos y condiciones.
- Validación de los resultados obtenidos mediante comparación con mediciones de referencia.
- Identificación y corrección de posibles errores o fallos.

6. Documentación y preparación:

- Elaboración de la documentación (redacción del proyecto)

7. Evaluación y revisión final:

- Evaluación general del proyecto para identificar lecciones aprendidas y áreas de mejora.

Etapas	Tiempo de duración de las etapas en semanas											
	Mayo				Junio				Julio			
	1	2	3	4	1	2	3	4	1	2	3	4
1												
2												
3												
4												
5												
6												
7												

Figura 2. Diagrama de Gantt. Planificación del proyecto

2. MATERIAL Y MÉTODOS

La elección de los materiales necesarios para el proyecto tiene una importancia significativa, dado que de estas elecciones dependerá la calidad del dispositivo creado, así como el presupuesto final para la creación de este.

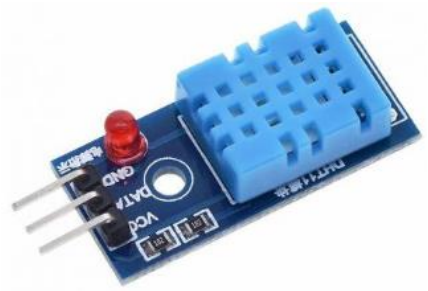
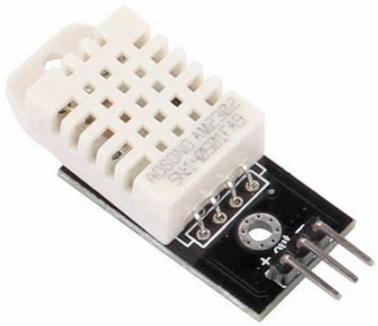
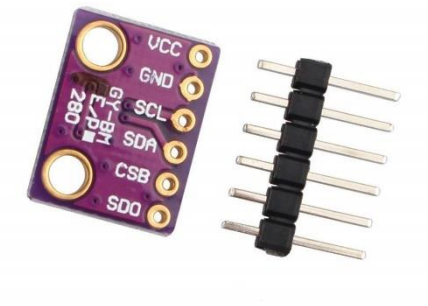
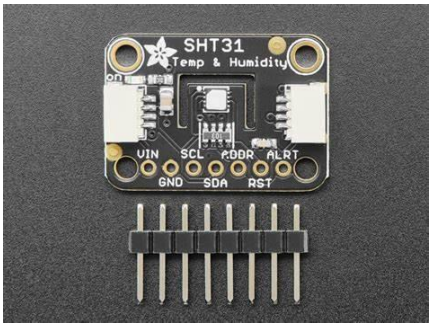
A continuación, se enumerará todo el conjunto de posibles materiales o componentes necesarios para el ensamblaje del dispositivo, para de esta forma, exponer las diferentes opciones con las que se podría haber abordado el proyecto teniendo en cuenta las especificaciones y precio, finalmente, se justificará la elección tomada.

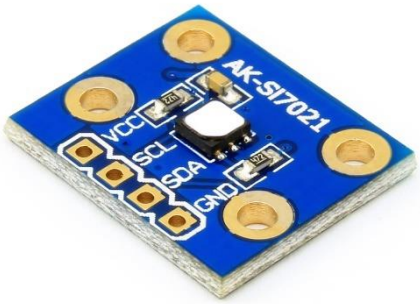
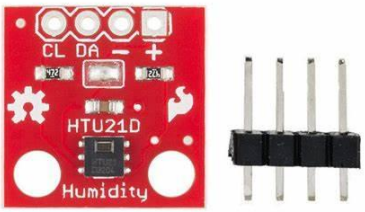
2.1 MATERIALES EMPLEADOS

Los materiales o componentes necesarios para el desarrollo del dispositivo son listados a continuación:

- Sensores de temperatura y humedad.
- Módulo de geolocalización.
- Placa de desarrollo.
- Batería o fuente de alimentación.
- Componentes adicionales: Cableado, destornilladores, ordenador, etc.

2.1.1 SENSORES DE TEMPERATURA Y HUMEDAD

Sensor	Especificaciones técnicas	Imagen
DHT11	• Rango de temperatura: 0°C a 50°C • Rango de humedad: 20% a 90%< • Precisión: ±2°C, ±5% de humedad relativa	
DHT22	• Rango de temperatura: -40°C a 80°C • Rango de humedad: 0% a 100% • Precisión: ±0.5°C, ±2% de humedad relativa	
BME280	• Rango de temperatura: -40°C a 85°C • Rango de humedad: 0% a 100% • Precisión: ±1°C, ±3% de humedad relativa	
SHT31	• Rango de temperatura: -40°C a 125°C • Rango de humedad: 0% a 100% • Precisión: ±0.3°C, ±2% de humedad relativa	

AM2302	• Rango de temperatura: -40°C a 80°C • Rango de humedad: 0% a 99.9% • Precisión: ±0.5°C, ±2% de humedad relativa	
Si7021	• Rango de temperatura: -10°C a 85°C • Rango de humedad: 0% a 100% • Precisión: ±0.4°C, ±3% de humedad relativa	
HDC1080	• Rango de temperatura: -40°C a 125°C • Rango de humedad: 0% a 100% • Precisión: ±0.2°C, ±2% de humedad relativa	
HTU21D	• Rango de temperatura: -40°C a 125°C • Rango de humedad: 0% a 100% • Precisión: ±0.3°C, ±2% de humedad relativa	

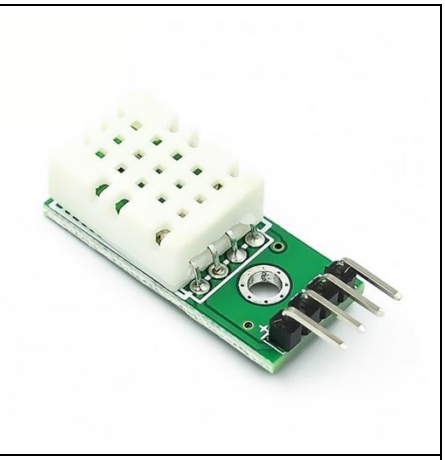
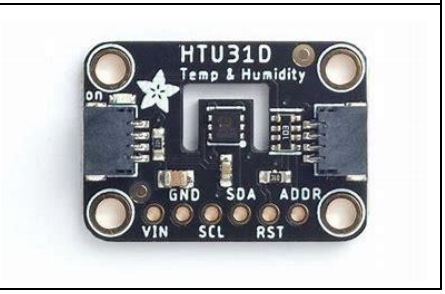
SHTC3	• Rango de temperatura: -40°C a 125°C • Rango de humedad: 0% a 100% • Precisión: ±0.2°C, ±2% de humedad relativa	
HTU31D	• Rango de temperatura: -40°C a 125°C • Rango de humedad: 0% a 100% • Precisión: ±0.3°C, ±2% de humedad relativa	

Tabla 1. Sensores de temperatura y humedad

Teniendo en cuenta este abanico de opciones en relación con el sensor de temperatura y humedad, finalmente se opta por el sensor DHT11, ver imagen adjunta.

Figura 3. Sensor DHT11

Los argumentos que justifican su elección son:

1. Coste: El sensor DHT11 es generalmente más económico en comparación con otros sensores de temperatura y humedad disponibles en el mercado. Debido a que este proyecto busca ser asequible y accesible, el factor económico se deberá priorizar.

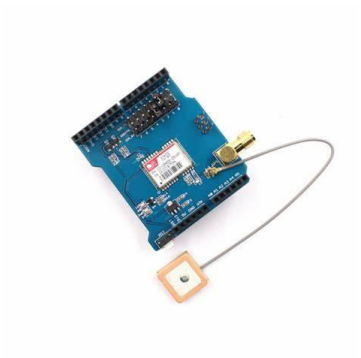
2. Facilidad de uso: El sensor DHT11 es bastante sencillo de utilizar, ya que solo requiere de un par de conexiones y un código relativamente simple para obtener mediciones de temperatura y humedad.

3. Disponibilidad: Debido a su popularidad, el sensor DHT11 es fácil de encontrar en tiendas en línea y establecimientos especializados en electrónica.

4. Rango de medición adecuado: Las necesidades de medición para este proyecto se encuentran dentro de los rangos de operación del sensor DHT11 (0°C a 50°C para temperatura y 20% a 90% para humedad relativa).

2.1.2 SENSOR DE GEOLOCALIZACIÓN

Sensor	Especificaciones técnicas	Imagen
GPS6MV2	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	
NEO-6M	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	
Ublox MAX-M8Q	• Módulo GPS/GLONASS/BeiDou compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	

SIM28	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	
VK16E	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	
GY-NEO6MV2	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	

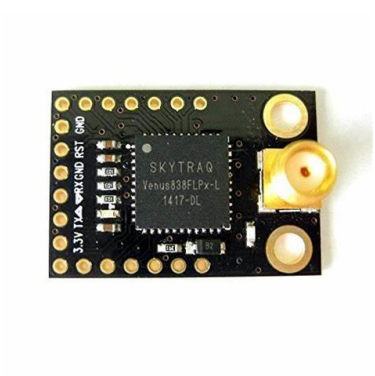
Beitian BN-880	• Módulo GPS/GLONASS/BeiDou compatible con NMEA 0183 y UART • Alimentación: 5V • Precisión: hasta 2.5 metros	
Quectel L76	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 2.8V a 4.3V • Precisión: hasta 2.5 metros	
MTK3339	• Módulo GPS compatible con NMEA 0183 y UART • Alimentación: 2.7V a 3.3V • Precisión: hasta 3 metros	
Skytraq Venus	• Módulo GPS/GLONASS compatible con NMEA 0183 y UART • Alimentación: 3.3V • Precisión: hasta 2.5 metros	

Tabla 2. Sensores de geolocalización

Respecto al módulo de geolocalización, todas las opciones ofrecen especificaciones similares:

- La precisión, se considera suficiente en todos los modelos mencionados.
- La compatibilidad es amplia en todos los casos y tampoco supone un factor decisivo.

Finalmente, se ha optado por la elección del sensor GPS6MV2 (ver imagen adjunta) debido a los siguientes argumentos:

- Fiabilidad: El GPS6MV2 es ampliamente utilizado y conocido por su fiabilidad. Ha sido probado y utilizado en numerosas aplicaciones y proyectos.

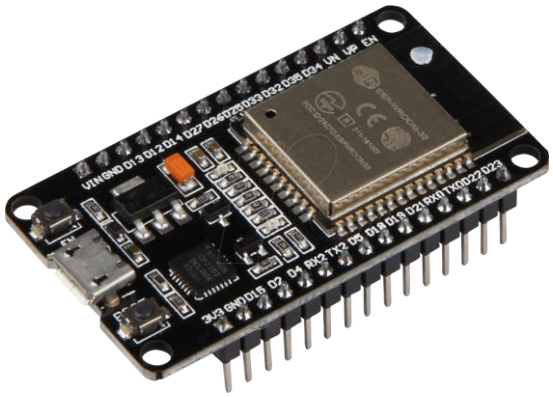
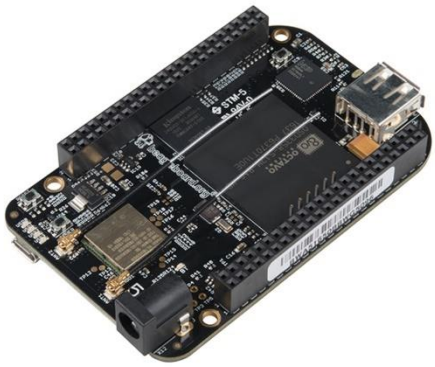
- Disponibilidad: Dado que el GPS6MV2 es un modelo estándar en el ámbito de aplicaciones IoT, es fácilmente accesible y esté disponible en muchos proveedores y plataformas de compra.

- Coste: Precio competitivo en comparación con otros sensores de características similares.

Figura 4. Sensor GPS6MV2

2.1.3 PLACA DE DESARROLLO

Sensor	Especificaciones técnicas	Imagen
Arduino Uno	• Microcontrolador: ATmega328P • Memoria Flash: 32 KB (0.5 KB ocupado por el bootloader) • SRAM: 2 KB • Velocidad de reloj: 16 MHz	
Raspberry Pi 4 Molel B	• Procesador: Broadcom BCM2711, Quad-Core Cortex-A72 (ARMv8) a 1.5GHz • Memoria RAM: 2GB, 4GB o 8GB LPDDR4-3200 SDRAM • Conectividad: Ethernet, Wi-Fi, Bluetooth, USB, HDMI	
NodeMCU v3 ESP8266	• Microcontrolador: ESP8266 • Memoria Flash: 4MB • Wi-Fi: 802.11 b/g/n • Conexión USB: Micro USB • GPIO: 17 pines digitales, 1 ADC, I2C, SPI, UART • Velocidad de reloj: 80 MHz	

Arduino Mega 2560	• Microcontrolador: ATmega2560 • Memoria Flash: 256 KB • SRAM: 8 KB • Velocidad de reloj: 16 MHz	
Rasberry Pi Zero W	• Procesador: Broadcom BCM2835, 1 GHz single-core ARM11 • Memoria RAM: 512 MB LPDDR2 SDRAM • Conectividad: Wi-Fi, Bluetooth, USB	
ESP32	• Microcontrolador: ESP32-D0WDQ • Memoria Flash: 4MB • Wi-Fi: 802.11 b/g/n • Bluetooth: v4.2 • GPIO: 36 pines digitales, I2C, SPI, UART, ADC • Velocidad de reloj: 240 MHz	
BeagleBone Black	• Procesador: TI Sitara AM335x, 1 GHz ARM Cortex-A8 • Memoria RAM: 512 MB DDR3 • Conectividad: Ethernet, USB, HDMI, UART, SPI, I2C, ADC	

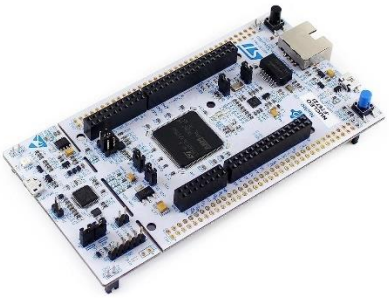
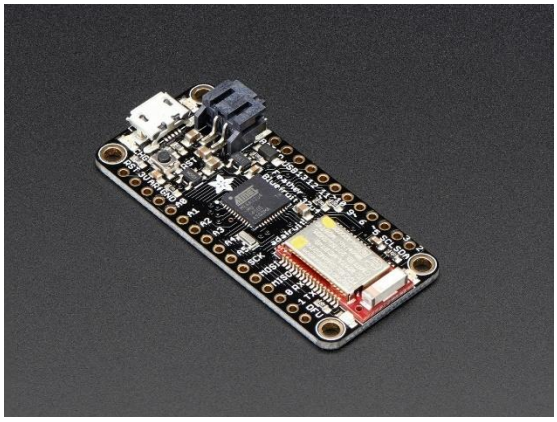
Teensy 4.0	• Microcontrolador: NXP i.MX RT1062, 600 MHz ARM Cortex-M7 • Memoria Flash: 2 MB • SRAM: 512 KB • GPIO: 36 pines digitales, I2C, SPI, UART, ADC	
STM32 Nucleo-F767ZI	• Microcontrolador: STM32F767ZIT6, 216 MHz ARM Cortex-M7 • Memoria Flash: 2 MB • SRAM: 512 KB • GPIO: 144 pines digitales, I2C, SPI, UART, ADC	
Adafruit Feather M0	• Microcontrolador: ATSAMD21G18, 48 MHz ARM Cortex-M0 • Memoria Flash: 256 KB • SRAM: 32 KB • GPIO: 20 pines digitales, I2C, SPI, UART, ADC	

Tabla 3. Placas de desarrollo

Como se puede observar, la oferta de placas de desarrollo es amplia, no obstante, se termina eligiendo el modelo NodeMCU v3 ESP8266 (ver imagen adjunta) debido a los siguientes motivos:

- Conectividad Wi-Fi: El NodeMCU v3 ESP8266 cuenta con conectividad Wi-Fi integrada, lo que lo hace ideal para proyectos de IoT y aplicaciones que requieren conectividad inalámbrica.

- Potencia de procesamiento: A pesar de su tamaño compacto, el ESP8266 ofrece un rendimiento sólido con su microcontrolador de alta velocidad y memoria flash de 4 MB. Esto permite ejecutar aplicaciones y manejar tareas complejas.

- Amplia comunidad y documentación: El NodeMCU v3 ESP8266 cuenta con una gran comunidad de desarrolladores y usuarios que proporcionan soporte y recursos en línea. Facilidad para encontrar tutoriales, ejemplos de código,etc.

- Gran cantidad de pines GPIO: El NodeMCU v3 ESP8266 ofrece 17 pines GPIO digitales, lo que brinda flexibilidad para conectar y controlar una amplia variedad de dispositivos y sensores externos.

- Compatibilidad con Arduino: El NodeMCU v3 ESP8266 se puede programar utilizando el entorno de desarrollo de Arduino, lo que facilita la migración de proyectos existentes o el acceso a la amplia gama de bibliotecas y recursos disponibles para Arduino.

Figura 5. NodeMCU v3 ESP8266

2.1.4 BATERÍA DE ALIMENTACIÓN

La elección de la batería se regirá por las siguientes condiciones:

- Ligereza: Dado que el dispositivo está pensado para un abanico de opciones de montaje y de condiciones de uso muy amplio, la ligereza de la batería es fundamental.
  De esta manera, aplicaciones de montaje en dispositivos aéreos, como, por ejemplo, drones, se verá menos limitada.

- Voltaje: Se tiene que verificar que el voltaje proporcionado a los componentes electrónicos del dispositivo está dentro del rango de funcionamiento.

    - La placa NodeMCU v3 ESP8266 funciona con un voltaje de 3.3V

    - El módulo de geolocalización GPS6MV2 y el sensor DHT11 generalmente operan en un rango de voltaje de 3.3V a 5V.

- Duración de la batería

- Tamaño

Finalmente, teniendo en cuenta las condiciones mencionadas, se ha optado por una batería externa universal, modelo GNG-109, Input DC 5V - 1000MAH - Otput DC 5V - 1000 MAH (Max) – Capacidad 2600MA, según imagen adjunta.

Figura 6. Batería GNG-109

2.1.5 MONTAJE Y CABLEADO

Dado que la elección de componentes ha sido justificada, en este apartado se detallará paso a paso el montaje del dispositivo.

Empezando por el componente principal, la placa de desarrollo NodeMCU V3 ESP8266 consta de diferentes tipos de conexiones:

- Conexión micro USB: Se utiliza para alimentar la placa y establecer comunicación con un ordenador o dispositivo externo.

- Pines GPIO (*General Purpose Input/Output*): La placa cuenta con varios pines GPIO digitales y analógicos que se pueden utilizar para conectar sensores, actuadores y otros dispositivos electrónicos.

- Conexión UART: La placa tiene un puerto UART (Universal Asynchronous Receiver/Transmitter) que permite la comunicación serie con otros dispositivos.

- Conexión I2C: La placa incluye un bus I2C (Inter-Integrated Circuit) que permite la conexión de periféricos compatibles con este protocolo, como sensores y pantallas.

- Conexión SPI: La placa dispone de un bus SPI (Serial Peripheral Interface) que permite la comunicación con dispositivos que utilizan este protocolo, como pantallas TFT y tarjetas SD.

- Conexión ADC (*Analog-to-Digital Converter*): La placa tiene un pin ADC que permite la lectura de señales analógicas.

- Conexión de alimentación: La placa tiene pines de alimentación para proporcionar voltaje a los dispositivos conectados, incluyendo 3.3V y 5V.

A continuación, se muestra el “pinout” de la placa de desarrollo elegida:

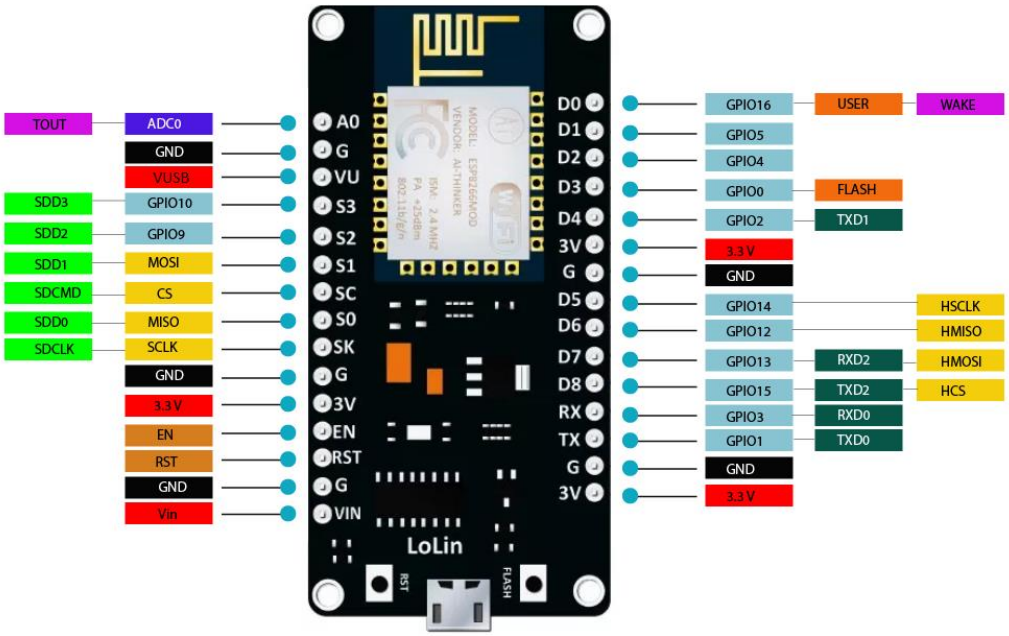


Figura 7. Diagrama de conexiones, NodeMCU v3 ESP8266

Teniendo claro el tipo de conexiones que ofrece la placa de desarrollo y el diagrama de conexiones de esta, empezaremos a conectar los diferentes sensores que equipará nuestro dispositivo:

- Conexión del sensor DHT11:

  El sensor de temperatura y humedad DHT11 deberá estar conectado a un pin de alimentación (VCC), a un pin de tierra y a un pin de datos (DATA), este último funcionando como transmisor (TX)

En este caso, la conexión se efectúa en los 3 pines mostrados a continuación:

Figura 8. Pines de conexión para el sensor DHT11

- Conexión del sensor GPS6MV2:

El sensor de geolocalización GPS6MV2, de la misma manera que el sensor DHT11, deberá estar conectado a la alimentación y a tierra. Por otra parte, este tipo de sensor necesita enviar y recibir datos, por lo que se tendrán que usar dos pines GPIO, uno a modo de transmisor (TX) y otro a modo de receptor (RX).

Por lo tanto, los pines utilizados para la conexión del sensor GPS6MV2 serán los siguientes:

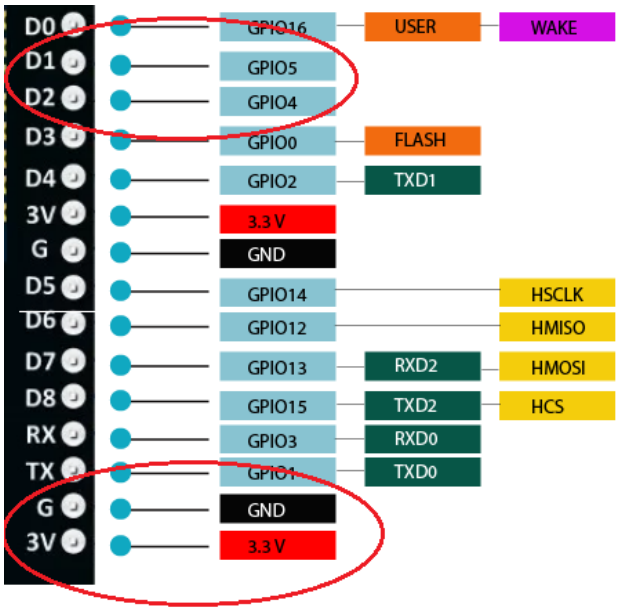


Figura 9. Pines de conexión para el sensor GPS6MV2

Como se puede observar, para la transferencia de datos, se usarán los pines GPIO4 y GPIO5.

- Conexión de la batería:

La batería se encargará de dar alimentación al dispositivo.

Esta será conectada a la placa de desarrollo mediante la conexión micro USB.

El resultado final con todos los componentes se muestra en la imagen adjunta.

Figura 10. Vista general del dispositivo

2.1.6 PRESUPUESTO

A continuación, se muestra el presupuesto necesario para el montaje del dispositivo. Como se ha comentado ya previamente, la optimización de costes ha sido una de las principales prioridades de este proyecto, ya que la creación de este pretende ser posible y accesible para un gran número de personas.

Componente	**Coste**
Sensor de Temperatura y Humedad	2,45€
Módulo de geolocalización	2,52€
Microcontrolador	3,95€
Batería	11€
Cableado	1,60€
Alojamiento Web (*)	0€
Tecnología de desarrollo (*)	0€
Coste Total	**21,52€**

Tabla 4. Presupuesto

(*) Para el desarrollo de la aplicación, tal como se mostrará en las siguientes secciones, se utilizaron programas de uso libre. El alojamiento de la aplicación Web fue gratuito en https://es.000webhost.com

2.2 METODOLOGÍA

Uno de los pilares de este proyecto, es que la medición que realizan los sensores en tiempo real sea cargada en una base de datos, la cual pueda ser representada en una página web, ofreciendo esta última, herramientas para procesar y visualizar los datos recopilados.

Llegados a este punto, el hardware del dispositivo está finalizado y, por ende, siguiendo la planificación del proyecto propuesta en la sección 1.3, se abordará el desarrollo del software.

El software deberá estructurarse o dividirse en diferentes ramas, a continuación, se especificará cada una de las etapas del proceso, detallando su desarrollo.

2.2.1 ENTORNO DE PROGRAMACIÓN DE LOS SENSORES

El sensor de temperatura y humedad DHT11 y el sensor de geolocalización GPS6MV2 deberán proporcionar datos de lectura a la placa de desarrollo NodeMCU v3 ESP8266.

Para ello será necesario utilizar una plataforma de unifique y conecte hardware y software. En el siguiente apartado se describirá la plataforma seleccionada con la que seremos capaces de usar nuestros sensores para la función deseada.

2.2.1.1 ARDUINO

Como hemos mencionado, necesitamos una herramienta que sea capaz de enlazar hardware y software. Arduino [5] será la herramienta elegida para programar el dispositivo. Dicha herramienta fue seleccionada por ser una plataforma de código abierto que permite programar una gran cantidad de microcontroladores en el ámbito de IoT.

Mediante el entorno de desarrollo integrado (IDE) de Arduino podremos escribir, compilar y cargar programas a la placa de desarrollo, de esta manera conectaremos los sensores necesarios.

Figura 11. Logo Arduino

2.2.1.2 CONFIGURACIÓN DEL SENSOR DE TEMPERATURA Y HUMEDAD

Para la configuración del sensor DHT11 será necesario realizar los siguientes pasos. Es necesario aclara que el código escrito a continuación se carga a la placa de desarrollo mediante la herramienta Arduino.

En primer lugar, será necesario llamar a las librerías de cabecera para el sensor DHT11:

```
#include "DHT.h"
#define DHTPIN 14
#define DHTTYPE DHT11
```

A continuación, definiremos las variables para su lectura y la creación del objeto DHT.

```
float t;
float h;
float f;
float hif;
float hic;
DHT dht(DHTPIN, DHTTYPE)
```

Siendo:

- `float t;` - Declaramos la variable `t` de tipo float. Se utilizará para almacenar el valor de temperatura en grados Celsius.

- `float h;` - Declaramos la variable `h` de tipo float. Se utilizará para almacenar el valor de humedad.

- `float f;` Se utilizará para leer y almacenar el valor de temperatura en grados Fahrenheit.

- `float hif;` - La variable “hif” será el índice de calor en grados Fahrenheit.

- `float hic;` - La variable “hic” será el índice de calor en grados Celsius.

- `DHT dht(DHTPIN, DHTTYPE);` - Creamos un objeto de la clase DHT, utilizándolo para interactuar con el sensor de temperatura y humedad. Será necesario tomar los parámetros: `DHTPIN` y `DHTTYPE`.

    - El valor de `DHTPIN` representa el pin en el que está conectado el sensor DHT, en este caso nuestro sensor DHT11 estará conectado al pin número 14 de la placa de desarrollo.
    - `DHTTYPE` especifica el modelo o tipo de sensor DHT que se está utilizando. Por lo que, como se ha descrito, nuestro sensor será un DHT11.

A continuación tendremos que definir la función “setup()” en Arduino. Ejecutándola al principio del programa nos permitirá realizar las configuraciones iniciales:

`Serial.begin(115200);` De esta manera iniciaremos la comunicación por línea serie a una velocidad de 115200 baudios, permitiendo la comunicación entre Arduino y el ordenador mediante el puerto serie.

`dht.begin();` Iniciamos la comunicación con el sensor DHT11, permitiendo a Arduino leer los datos del sensor.

A continuación, mediante el siguiente código realizaremos el proceso de intento de conexión a una red Wi-Fi.

```
Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());
```

Seguidamente definiremos un bucle infinito que se ejecutará continuamente, leyendo los valores de temperatura y humedad; verificando si son valores válidos utilizando el comando “isnan()”.

En el caso de que alguno de los valores no sea válido, se imprimirá un mensaje de error que saldrá del bucle principal.

Habrá que definir también el índice de calor utilizando la función `dht.computeHeatIndex()`, que se almacenarán en “hif” y “hic” respectivamente.

```
void loop()
{
        h = dht.readHumidity();
        t = dht.readTemperature();
        f = dht.readTemperature(true);
        if (isnan(h) || isnan(t) || isnan(f)) {
          Serial.println(F("Failed to read from DHT sensor!"));
          return;
        }
        hif = dht.computeHeatIndex(f, h);
        hic = dht.computeHeatIndex(t, h, false);

}
```

2.2.1.3 CONFIGURACIÓN DEL SENSOR DE GEOLOCALIZACIÓN

De la misma manera que ha sido configurado el sensor de temperatura y humedad, hará falta escribir el código necesario para hacer el funcionar el sensor de geolocalización GPS6MV2.

Definiremos la librería para la comunicación con el sensor:

```
#include <TinyGPS++.h>
```

Crearemos los objetos y variables necesarios para el sensor:

`TinyGPSPlus gps;` Objeto de la clase TinyGPS++ oara obtener los datos del GPS.

`SoftwareSerial ss(4, 5);` Objeto de la clase SoftwareSerial para comunicarse con el dispositivo GPS a través de los pines 4 (RX) y 5 (TX) de la placa de desarrollo ESP8266.

```
const double TORRETAMARIT_LAT = 38.278605371975175;
const double TORRETAMARIT_LNG = -0.6920385047340057;
double distanceKm;
float latitude , longitude;
int year , month , date, hour , minute , second;
String date_str , time_str , lat_str , lng_str, zona;
int pm;
```

Se ha definido un punto base mediante latitud y longitud, incluyendo una variable para almacenar la distancia a dicha estación en kilómetros. Además, es necesario almacenar las variables de latitud y longitud. Por último, necesitaremos las variables para almacenar la fecha y la hora del GPS.

De igual forma que se ha realizado con el sensor de temperatura y humedad, mediante el siguiente código realizaremos el proceso de intento de conexión a una red Wi-Fi.

```
Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());
```

A continuación, mediante una función “loop()” se realizan las siguientes acciones:

```
void loop()
{
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
    {
      if (gps.location.isValid())
```

- `while (ss.available() > 0)`: Este bucle se ejecuta mientras haya datos disponibles para leer en el objeto `ss` (que es una instancia de `SoftwareSerial`). `ss.available()` devuelve el número de bytes disponibles para leer.

- `if (gps.encode(ss.read()))`: Esta condición comprueba si se pueden leer valores del sensor GPS. `ss.read()` lee un byte de datos del objeto `ss`, que representa la comunicación serial con el dispositivo GPS. El resultado se pasa a la función `gps.encode()`, que intenta interpretar los datos GPS y actualizar los valores en el objeto `gps`.

- `if (gps.location.isValid())`: Esta condición verifica si la ubicación del GPS es válida. La función `isValid()` del objeto `gps.location` devuelve `true` si se ha recibido una ubicación válida del GPS.

```
latitude = gps.location.lat();
lat_str = String(latitude , 6);
          // Almacenamiento de la Latitud en un string
longitude = gps.location.lng();
lng_str = String(longitude , 6);
```

- `latitude = gps.location.lat();`: Se obtiene la latitud actual del objeto `gps.location` y se almacena en la variable `latitude`.

- `lat_str = String(latitude, 6);`: La latitud se convierte a una cadena de caracteres y se almacena en la variable `lat_str`. El número `6` especifica el número de decimales que se mostrarán en la cadena.

- `longitude = gps.location.lng();`: Se obtiene la longitud actual del objeto `gps.location` y se almacena en la variable `longitud`.

- `lng_str = String(longitude, 6);`: La longitud se convierte a una cadena de caracteres y se almacena en la variable `lng_str`. El número `6` especifica el número de decimales que se mostrarán en la cadena.

```
distanceKm = gps.distanceBetween(latitude, longitude,
TORRETAMARIT_LAT, TORRETAMARIT_LNG) / 1000.0;
Serial.print("Distancia (km) a ESTACION BASE TORRETAMARIT: ");
Serial.println(distanceKm);
Serial.print("Velocidad (m/s): ");
Serial.println(gps.speed.mps()); // Velocidad en m/s (double)
Serial.print("Velocidad (Km/s): ");
Serial.println(gps.speed.kmph()); // Velocidad en Km/h (double)
```

- `distanceKm = gps.distanceBetween(latitude, longitude, TORRETAMARIT_LAT, TORRETAMARIT_LNG) / 1000.0;`: Se calcula la distancia entre la ubicación actual (definida por `latitude` y `longitude`) y las coordenadas de la estación base (TORRETAMARIT_LAT y TORRETAMARIT_LNG) utilizando la función `distanceBetween()` del objeto `gps`. El resultado se divide por 1000.0 para obtener la distancia en kilómetros y se almacena en la variable `distanceKm`.

- `Serial.print("Distancia (km) a ESTACION BASE TORRETAMARIT: ");`: Se muestra un mensaje en el monitor serie (puerto serial) indicando que se imprimirá la distancia a la estación base.

- `Serial.println(distanceKm);`: Se imprime la distancia en kilómetros a la estación base en el monitor serie.

- `Serial.print("Velocidad (m/s): ");`: Se muestra un mensaje indicando que se imprimirá la velocidad en metros por segundo.

- `Serial.println(gps.speed.mps());`: Se imprime la velocidad en metros por segundo utilizando la función `mps()` del objeto `gps.speed`.

- `Serial.print("Velocidad (Km/s): ");`: Se muestra un mensaje indicando que se imprimirá la velocidad en kilómetros por hora.

- `Serial.println(gps.speed.kmph());`: Se imprime la velocidad en kilómetros por hora utilizando la función `kmph()` del objeto `gps.speed`.

En resumen, este código lee los datos del sensor GPS y realiza cálculos como la latitud, longitud, distancia a una estación base y velocidad. Luego, muestra estos valores en el monitor serie para su visualización y análisis.

Por último, el siguiente bloque de código recopila los datos de hora del sensor GPS, los ajusta a un formato legible y los muestra en el monitor serie. A continuación, se explica línea a línea la función del código:

```
{
  date_str = "";
  date = gps.date.day();
  month = gps.date.month();
  year = gps.date.year();
```

2.2.1.4. ENVÍO DE ALARMAS

En este apartado describiremos el código utilizado para la función de envío de alarmas en formato de correo electrónico en función de ciertos umbrales predefinidos.

- Las primeras líneas establecen la configuración del servidor SMTP al que se enviarán los correos electrónicos. `SMTP_HOST` representa el nombre del host del servidor SMTP, `SMTP_PORT` indica el número de puerto del servidor SMTP, `AUTHOR_EMAIL` es la dirección de correo electrónico del remitente y `AUTHOR_PASSWORD` es la contraseña del remitente.

```
session.server.port = SMTP_PORT;
session.login.email = AUTHOR_EMAIL;
session.login.password = AUTHOR_PASSWORD;
session.login.user_domain = "";
```

- Luego se crea una instancia de la clase `SMTP_Message`, que representa el mensaje de correo electrónico que se enviará.

```
session.server.host_name = SMTP_HOST;
SMTP_Message message;
```

- A continuación, se configuran los campos del mensaje de correo electrónico. Se establece el nombre y la dirección de correo electrónico del remitente en `message.sender.name` y `message.sender.email`, respectivamente.

```
message.sender.name = "TFG_IOT_2023";
message.sender.email = AUTHOR_EMAIL;
```

Anotación: El resto del Código podrá consultarse en el Anexo I

2.2.1.5. ENVÍO DE DATOS AL SERVIDOR WEB

Este apartado muestra la explicación de las principales partes y funcionalidades del código encargado de enviar los datos a un servidor web utilizando el método POST.

- En primer lugar, se crea una cadena llamada `datos_a_enviar` que contiene los datos a enviar al servidor web. Esta cadena se construye concatenando varias cadenas y variables utilizando el operador de concatenación `+`. Los datos incluyen la temperatura (`t`), la humedad (`h`), el calor (`hic`), la latitud (`lat_str`) y la longitud (`lng_str`).

```
String datos_a_enviar = "temperatura=" + String(t) + "&humedad=" + String(h)+ "&calor=" + String(hic)+ "&latitud=" + String(lat_str)+ "&longitud=" + String(lng_str);
```

- Se agrega un retardo de 70000 milisegundos (70 segundos) utilizando la función `delay()`. Esto significa que habrá un retraso de 70 segundos antes de que se ejecute el siguiente bloque de código.

```
delay(70000);
```

- Se establece la conexión HTTP con el servidor web utilizando la función `http.begin()`. Se proporciona la URL del servidor web como argumento.

- Se agrega un encabezado a la solicitud HTTP utilizando la función `http.addHeader()`. En este caso, se establece el tipo de contenido como "application/x-www-form-urlencoded", que es un tipo de contenido comúnmente utilizado para enviar datos de formulario.

```
http.begin(client,"http://tfgmarcoleon.000webhostapp.com/esppost.php");
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
```

- Se realiza la solicitud POST al servidor web utilizando la función `http.POST()`. Se proporcionan los datos a enviar (`datos_a_enviar`) como argumento. El resultado de esta función se almacena en la variable `codigo_respuesta`. Se verifica si `codigo_respuesta` es mayor que 0. Esto indica que la solicitud se realizó correctamente y no se produjo un error. Si `codigo_respuesta` es igual a 200, significa que la solicitud fue exitosa. Se muestra en el monitor serie el código de respuesta HTTP y se lee la respuesta del servidor utilizando la función `http.getString()`. La respuesta del servidor se almacena en la variable `cuerpo_respuesta` y se muestra en el monitor serie. Si `codigo_respuesta` es diferente de 200, indica que ocurrió un error en la solicitud. Se muestra en el monitor serie el código de respuesta HTTP correspondiente al error. Se muestra en el monitor serie los datos que se van a enviar al servidor web. Se llama a la función `http.end()` para liberar los recursos utilizados por la conexión HTTP. En resumen, este código establece una conexión con un servidor web y envía datos utilizando el método POST. Luego muestra la respuesta del servidor en el monitor serie.

```
int codigo_respuesta = http.POST(datos_a_enviar);
// Enviar datos a Servidor Web por el metodo POST

If(codigo_respuesta>0)
  // Comprobar si ha ocurrido error
{
  Serial.println("Código HTTP: "+ String(codigo_respuesta));
  if (codigo_respuesta == 200)    // todo OK, NO ha ocurrido error
  {
    String cuerpo_respuesta = http.getString();
    Serial.println("El servidor respondió: ");
    Serial.println(cuerpo_respuesta);
  }
} else
  {
  Serial.print("Error enviado POST, código: ");
  // Mostrar codigo de error al enviar datos a servidor Web
  Serial.println(codigo_respuesta);
```

```
  }
Serial.println(datos_a_enviar);
  // Mostrar datos a enviar
http.end();
  // Liberar recursos
}
```

2.2.2 ENTORNO DE PROGRAMACIÓN DE LA PÁGINA WEB

Mediante la herramienta Arduino, hemos sido capaces de cargar un código de programación a nuestra placa de desarrollo ESP8266. Ahora nuestros sensores son capaces de medir los valores necesarios y de enviarlos a una base de datos.

Estas funciones proporcionadas por los sensores y la placa de desarrollo deberán ser expuestas en una aplicación web que nos permita visualizar de manera cómoda los valores recogidos.
Deberá ser una herramienta que permita estudiar y sacar conclusiones del trabajo realizado por el dispositivo de medición.

Para este proyecto hemos optado por el servicio de alojamiento web 000webhost.com, ya que como usuario nos permite crear y publicar sitios web en línea de manera gratuita.

Algunas de las características y servicios más relevantes ofrecidos por este servicio de alojamiento web son:

1. Alojamiento gratuito.

2. Constructor de sitios web: 000webhost.com proporciona un constructor de sitios web fácil de usar que permite a los usuarios crear y diseñar su sitio web sin necesidad de conocimientos de programación.

3. Panel de control: Los usuarios tienen acceso a un panel de control intuitivo desde donde pueden administrar su sitio web, realizar configuraciones, administrar archivos, bases de datos y correos electrónicos, entre otras funciones.

4. Soporte de PHP y MySQL.

2.2.2.1 CONFIGURACIÓN DE LA BASE DE DATOS

La base de datos ha sido utilizada por la página web del proyecto ubicada en 000webhostapp.com. Ha sido creada con MySQL, considerada como la base de datos de código abierto más popular [6]. Para este proyecto, se ha creado una tabla llamada “mediciones”. La tabla está definida mediante siete columnas, donde cada una de ellas será una variable: identidad, temperatura, humedad, calor, latitud, longitud y tiempo.

Campo	Tipo	Descripción
Id	Int	Este valor, autonumérico, es la clave principal que identifica unívocamente cada fila de valores.
Temperatura	Float	Temperatura obtenida
Humedad	int(11)	Humedad obtenida
Calor	int(11)	Índice de calor obtenido
Latitud	Float	Latitud obtenida según el módulo GPS
Longitud	Float	Longitud obtenida según el módulo GPS
Tiempo	datetime	Fecha y hora de la medida obtenida

Tabla 5. Mediciones

Una vez creada, la tabla tendrá el siguiente aspecto:

	id	temperatura	humedad	calor	latitud	longitud	tiempo
Editar Copiar Borrar	2226	22.6	71	22.77	38.2788	-0.692044	2023-06-07 15:29:13
Editar Copiar Borrar	2227	22.3	71	22.44	38.2786	-0.692006	2023-06-07 15:30:38
Editar Copiar Borrar	2228	22.2	71	22.33	38.2786	-0.692006	2023-06-07 15:32:02
Editar Copiar Borrar	2229	22.2	71	22.33	38.2786	-0.692014	2023-06-07 15:33:27
Editar Copiar Borrar	2230	22.6	71	22.77	38.2786	-0.692014	2023-06-07 15:34:51
Editar Copiar Borrar	2231	22.6	71	22.77	38.2786	-0.692002	2023-06-07 15:36:16
Editar Copiar Borrar	2232	22.6	71	22.77	38.2786	-0.691997	2023-06-07 15:37:40
Editar Copiar Borrar	2233	22.6	71	22.77	38.2786	-0.691997	2023-06-07 15:39:05
Editar Copiar Borrar	2234	22.6	71	22.77	38.2786	-0.692	2023-06-07 15:40:30
Editar Copiar Borrar	2235	22.6	71	22.77	38.2786	-0.692026	2023-06-07 15:41:55
Editar Copiar Borrar	2236	22.6	71	22.77	38.2786	-0.692026	2023-06-07 15:43:19
Editar Copiar Borrar	2237	22.6	71	22.77	38.2787	-0.691993	2023-06-07 15:44:45
Editar Copiar Borrar	2238	22.6	71	22.77	38.2787	-0.691993	2023-06-07 15:46:09

Figura 12. Fragmento de la tabla Mediciones

2.2.2.2 PÁGINA WEB

La página web será la herramienta de la que dispondremos para visualizar de manera filtrada aquellos datos que nos interesen, previamente recopilados en la base de datos. La programación de la página web se realiza con ficheros PHP, HTML y CSS, todos de código abierto. PHP se utiliza comúnmente en aplicaciones web por su integración con HTML y su compatibilidad con numerosas bases de datos, como la utilizada para este proyecto. HTML es un lenguaje basado en etiquetas que define los elementos que se muestran en una página web con elementos formateados con CSS (Cascading style sheet) con el fin de tener mejor personalización.

A continuación, se mostrará la apariencia de la página y los diferentes menús por los que se podrá navegar. De acuerdo a las funcionalidades que nos ofrece la página, procedemos a explicar las posibles maneras de analizar los datos en función de la herramienta que seleccionemos, es decir, se explicará con mayor profundidad cada uno de los apartados que la página web nos aporta.

- Acceso a la página:

La dirección de la página web será la siguiente:

https://tfgmarcoleon.000webhostapp.com/

  - Para acceder a la página será necesario iniciar sesión, de manera que se solicitará un usuario y contraseña:

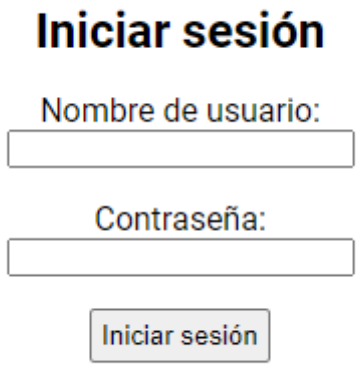


Figura 13. Acceso a la aplicación

- Interfaz de la página:

  - Inicio:

    Una vez iniciamos sesión de manera exitosa, se nos redirigirá a la página de inicio de la web. En esta, como podemos ver a continuación, se expondrá el título del proyecto y una breve introducción al mismo.

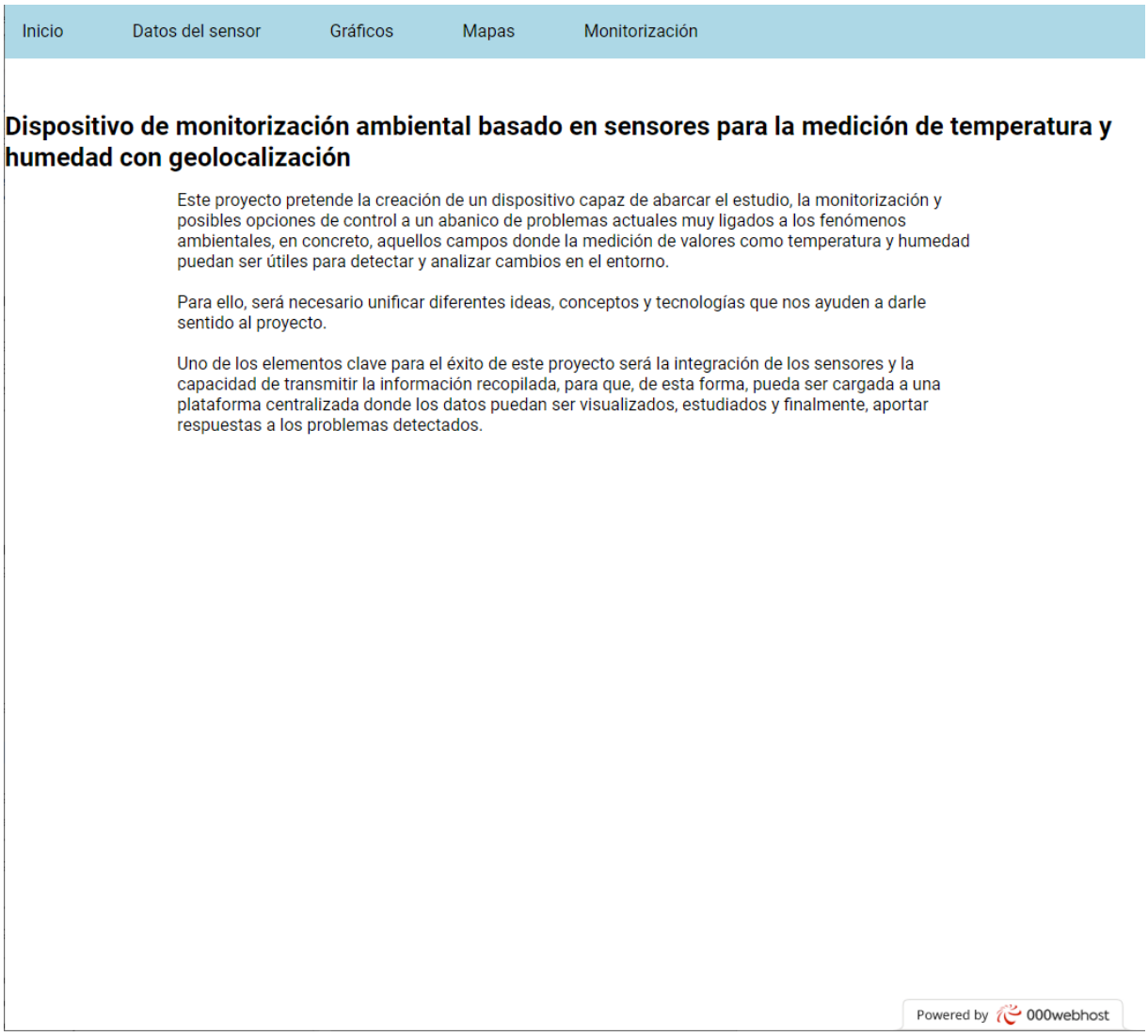


Figura 14. Página de Inicio

- Datos del sensor:

  En esta pestaña recibiremos las mediciones de los sensores del dispositivo. Se irán cargando de forma automática siempre que el sensor envíe datos nuevos.

La tabla muestra los valores de la misma manera que son almacenados en la base de datos. Imagen adjunta a continuación:

ID	Temperatura	Humedad	Calor	Latitud	Longitud	Tiempo
2258	22.6	70	22.74	38.2787	-0.691968	2023-06-07 16:31:17
2257	22.6	69	22.72	38.2787	-0.691968	2023-06-07 16:29:53

Figura 15. Datos del sensor

En esta pequeña muestra que se ha cogido como ejemplo podemos observar:

- ID: Número de identificación correspondiente a cada una de las medidas realizadas. La tabla los ordenará automáticamente en orden ascendente.
- Temperatura: Medida en Celsius e incluyendo décimas.
- Humedad: Medida en porcentaje.
- Calor: Índice de calor obtenido.
- Latitud
- Longitud
- Tiempo

Además, tendremos la opción de filtrar valores, siendo válidas las siguientes opciones:

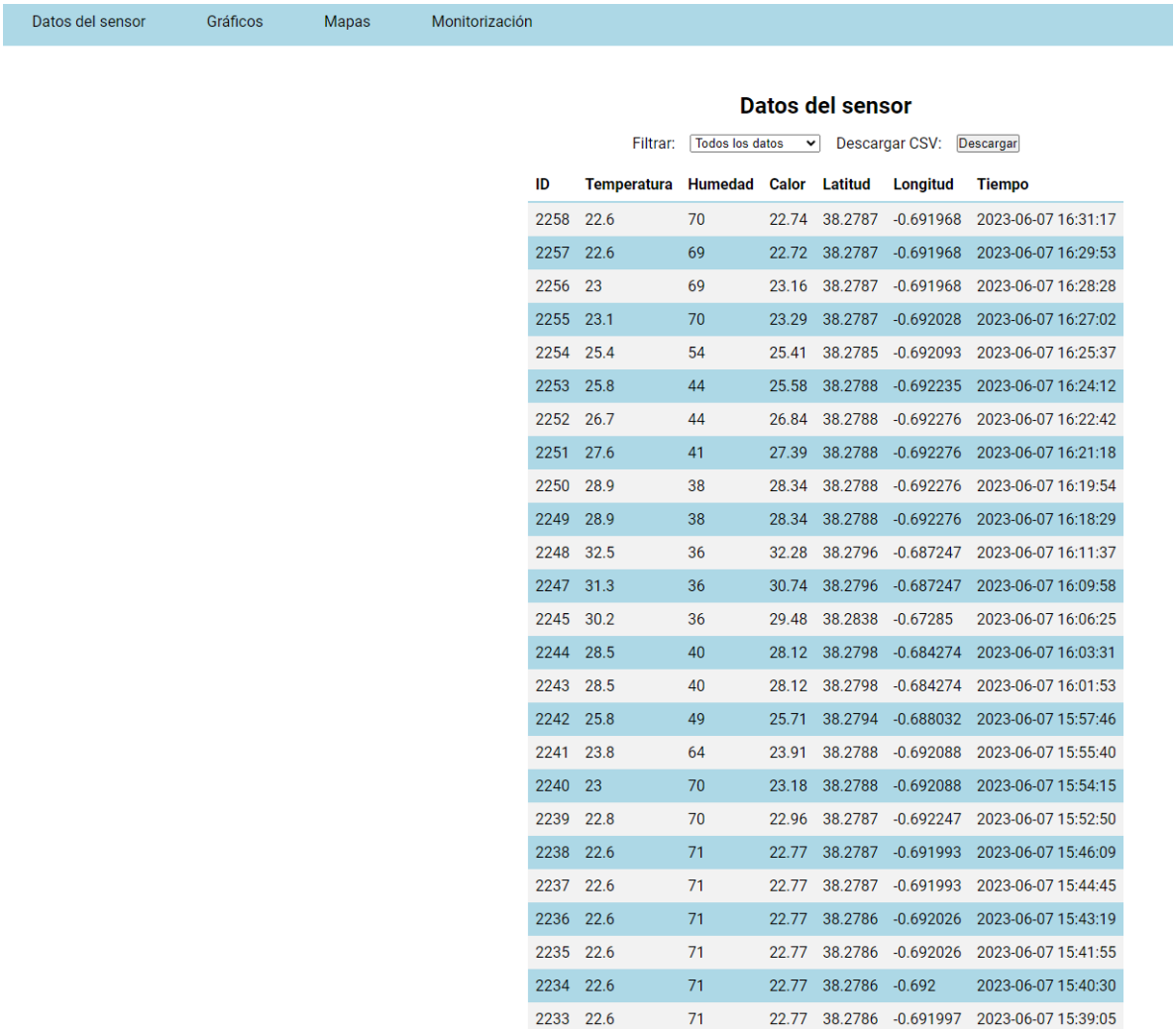
- Filtrar por:

    - Todos los datos.
    - Últimas 24h
    - Intervalo de fechas, donde se nos abrirá un nuevo desplegable en el cual tendremos que insertar la fecha de inicio y la fecha final deseada.

Figura 16. Intervalo de fechas

Adicionalmente, tendremos la opción de descargar ficheros CSV; de esta manera, la selección de datos mediante filtros podrá ser trabajada con cualquier programa de tratamiento de datos, por ejemplo, Excel, para la creación de gráficas de todo tipo.

A continuación, se adjunta vista global del menú descrito:



ID	Temperatura	Humedad	Calor	Latitud	Longitud	Tiempo
2258	22.6	70	22.74	38.2787	-0.691968	2023-06-07 16:31:17
2257	22.6	69	22.72	38.2787	-0.691968	2023-06-07 16:29:53
2256	23	69	23.16	38.2787	-0.691968	2023-06-07 16:28:28
2255	23.1	70	23.29	38.2787	-0.692028	2023-06-07 16:27:02
2254	25.4	54	25.41	38.2785	-0.692093	2023-06-07 16:25:37
2253	25.8	44	25.58	38.2788	-0.692235	2023-06-07 16:24:12
2252	26.7	44	26.84	38.2788	-0.692276	2023-06-07 16:22:42
2251	27.6	41	27.39	38.2788	-0.692276	2023-06-07 16:21:18
2250	28.9	38	28.34	38.2788	-0.692276	2023-06-07 16:19:54
2249	28.9	38	28.34	38.2788	-0.692276	2023-06-07 16:18:29
2248	32.5	36	32.28	38.2796	-0.687247	2023-06-07 16:11:37
2247	31.3	36	30.74	38.2796	-0.687247	2023-06-07 16:09:58
2245	30.2	36	29.48	38.2838	-0.67285	2023-06-07 16:06:25
2244	28.5	40	28.12	38.2798	-0.684274	2023-06-07 16:03:31
2243	28.5	40	28.12	38.2798	-0.684274	2023-06-07 16:01:53
2242	25.8	49	25.71	38.2794	-0.688032	2023-06-07 15:57:46
2241	23.8	64	23.91	38.2788	-0.692088	2023-06-07 15:55:40
2240	23	70	23.18	38.2788	-0.692088	2023-06-07 15:54:15
2239	22.8	70	22.96	38.2787	-0.692247	2023-06-07 15:52:50
2238	22.6	71	22.77	38.2787	-0.691993	2023-06-07 15:46:09
2237	22.6	71	22.77	38.2787	-0.691993	2023-06-07 15:44:45
2236	22.6	71	22.77	38.2786	-0.692026	2023-06-07 15:43:19
2235	22.6	71	22.77	38.2786	-0.692026	2023-06-07 15:41:55
2234	22.6	71	22.77	38.2786	-0.692	2023-06-07 15:40:30
2233	22.6	71	22.77	38.2786	-0.691997	2023-06-07 15:39:05

Figura 17. Vista global de datos del sensor

- Gráficos:

En este apartado dispondremos de la visualización de dos gráficos, uno de ellos enfocado a la temperatura, y el otro, enfocado a la humedad.

Ambos con las variables mencionadas en el eje de coordenadas y dependientes del tiempo en el eje de abscisas.

También dispondremos de la capacidad de filtrar con las mismas opciones explicadas para el menú de datos del sensor.

A continuación, se muestra un ejemplo:

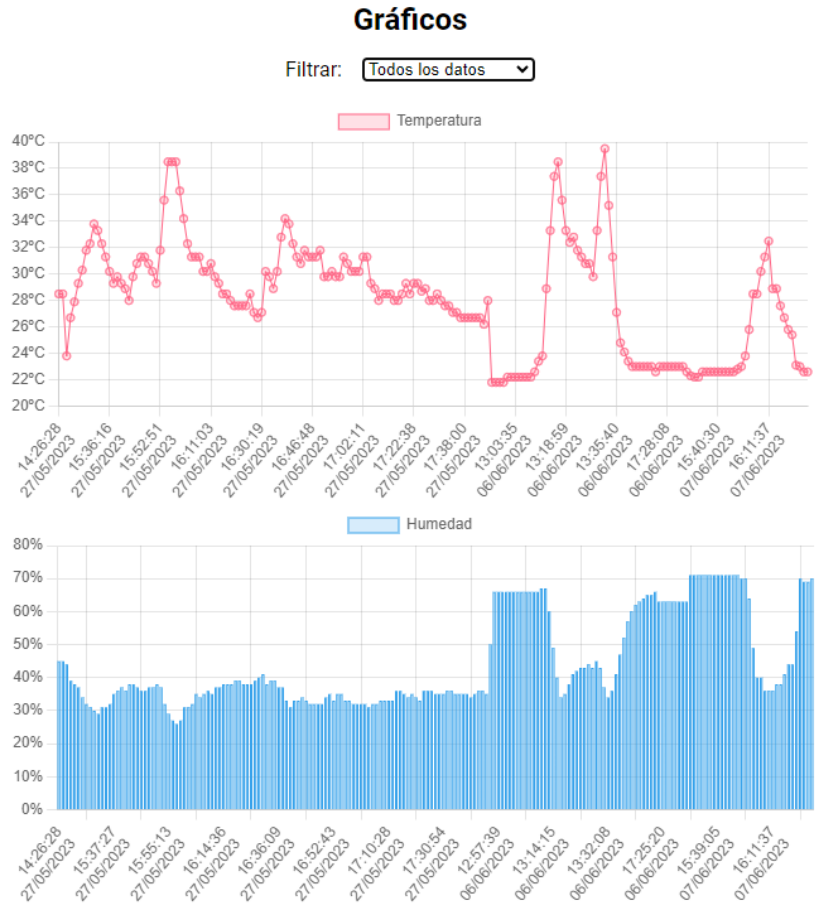


Figura 18. Gráficas de temperatura y humedad

- Mapas:

La herramienta mapas nos va a permitir crear mapeados tanto de temperatura como de humedad. De modo independiente para cada medición, podremos navegar, acercar o alejar la vista, etc. para identificar el conjunto de puntos medidos.

Las mediciones irán asociadas a un gradiente de color, de manera que una representación de muchas mediciones nos aportará un mapeado de la zona analizada. Los valores mostrados también se podrán filtrar con las mismas opciones ya mencionadas.

Figura 19. Mapas de temperatura y humedad

- Monitorización:

Por último, la opción de monitorización nos permitirá filtrar mediante coordenadas, es decir, mediante 2 coordenadas dadas, la herramienta creará un recinto cuadrilateral y todas aquellas mediciones tomadas en la zona seleccionada serán el objetivo a estudiar.

Una vez se insertan las coordenadas deseadas y después de clicar el botón “agregar” nos aparecerá un recuadro de color naranja en el mapa, indicando la zona seleccionada. Además, aparecerá un nuevo apartado donde podremos elegir el tipo de medición deseada, una cifra de referencia y si queremos que los datos mostrados sean:

- Igual que
- Mayor que
- Menor que
- Mayor o igual que
- Menor o igual que

Una vez se han seleccionado los requisitos deseados, y tras pulsar el botón “mostrar”, se mostrarán aquellas mediciones que cumplan con las condiciones. A continuación, se muestra un ejemplo de la función de esta herramienta:

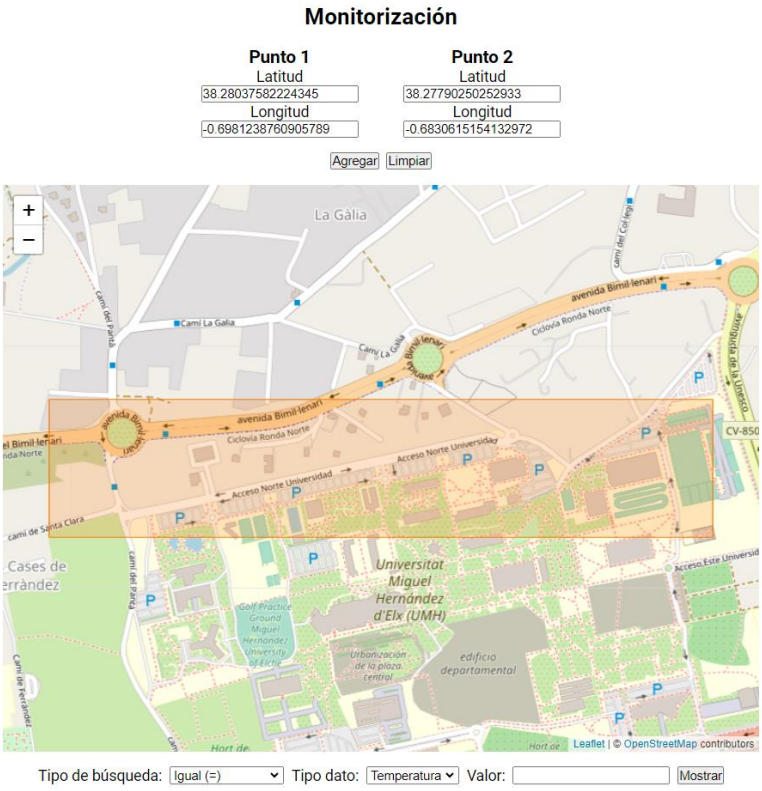


Figura 20. Monitorización según coordenadas

2.2.2.3 LECTURA, ENVÍO Y REGISTRO DE DATOS

Como hemos podido observar, la base de datos recibe la información de los sensores y la clasifica en la columna correspondiente. Esto se consigue mediante un script en PHP que establecerá una conexión con nuestra base de datos MySQL y realizará la inserción de registros en la tabla llamada “mediciones”. A continuación, se explica la configuración de dicho script para hacer posible la conexión:

1. Establecemos los datos de conexión:

- La variable `$servername` almacena el nombre del servidor de la base de datos, en este caso, "localhost".
- La variable `$database` almacena el nombre de la base de datos a la que se desea conectar.

- La variable `$username` almacena el nombre de usuario utilizado para acceder a la base de datos.
- La variable `$password` almacena la contraseña asociada al usuario

```
$servername = "localhost";
$database = "id20727547_sensor";
$username = "id20727547_marco";
$password = "********”
```

.

2. Creamos la conexión:

- Se utiliza la función `mysqli_connect()` para establecer una conexión con la base de datos, enviando como parámetros el nombre del servidor, el nombre de usuario, la contraseña y el nombre de la base de datos. El resultado de la conexión se guarda en la variable `$con`.

```
// Create connection
$con = mysqli_connect($servername, $username, $password, $database);

//mysqli_close($conn);
```

3. Comprobación de la conexión:

- Se verifica si la conexión se ha establecido correctamente utilizando una estructura condicional `if`. Si la variable `$con` tiene un valor verdadero (es decir, si la conexión se estableció correctamente), ¡¡¡se muestra el mensaje de “Conexión con base de datos exitosa!!!".

```
if ($con) {
echo "Conexion con base de datos exitosa!!!\n";
```

4. Obtención de los datos del formulario:

- El código verifica si se han enviado datos mediante el método POST desde un formulario HTML. Se utilizan varias estructuras condicionales `if` para comprobar si se han enviado variables específicas ('temperatura', 'calor', 'humedad', 'latitud' y 'longitud') y, en caso afirmativo, se asignan a las variables correspondientes.

```
if(isset($_POST['temperatura'])) {
    $temperatura = $_POST['temperatura'];
    echo "Estación meteorológica.";
    echo " Temperatura : ".$temperatura;
//echo "\n<br>";
    }
if(isset($_POST['calor'])) {
    $calor = $_POST['calor'];
    echo " Calor : ".$calor;
    }
if(isset($_POST['humedad'])) {
    $humedad = $_POST['humedad'];
    echo " Humedad : ".$humedad;
    }
if(isset($_POST['latitud'])) {
    $latitud = $_POST['latitud'];
    echo "Latitud : ".$latitud;
    }
if(isset($_POST['longitud'])) {
    $longitud = $_POST['longitud'];
```

7. Inserción de los datos en la base de datos:

- Se construye una consulta SQL de inserción utilizando los valores de las variables obtenidas anteriormente. La consulta se guarda en la variable `$consulta`.

- Se utiliza la función `mysqli_query()` para ejecutar la consulta en la base de datos, pasando como parámetros la conexión `$con` y la consulta `$consulta`.

- Se utiliza una estructura condicional `if` para verificar si la consulta se ejecutó correctamente y se muestra un mensaje correspondiente.

```
        $consulta      =      "INSERT      INTO      mediciones
(temperatura,humedad,calor,latitud,longitud,tiempo)       VALUES
('$temperatura','$humedad','$calor','$latitud','$longitud','$fec
ha_actual')";
        $resultado = mysqli_query($con, $consulta);
        if ($resultado){
            echo " Registro en base de datos OK! ";
          echo "\n<br>\n"; //* Esto es un salto de linea
            } else {
            echo " Falla! Registro BD\n";
            echo " Fecha de error en Registro BD: ".$fecha_actual;
                    }
    }
} else {
    echo "Falla! conexion con Base de datos ";
}
?>
```

2.2.3 ENTORNO DE PRUEBAS

Una vez la parte de Hardware y Software han sido terminadas, el dispositivo está listo para empezar a probarse.

El entorno de pruebas va a estar condicionado por la época del año en la cual este proyecto ha sido desarrollado, ya que no se ha dispuesto de un largo periodo de tiempo y los cambios ambientales no han sido muy notables.

El entorno se ha reducido a las ciudades de Alicante y Elche, donde se han efectuado diversas pruebas.

Durante las pruebas o ensayos con el dispositivo principalmente se ha buscado:

- Test de duración de la batería de alimentación.

- Fiabilidad de los datos medidos mediante el uso de instrumentos externos que verifiquen la precisión de los sensores utilizados.

- Búsqueda de cambios bruscos de temperatura y humedad que pongan a prueba la capacidad de reacción del dispositivo.

- Control de la continuidad y alcance de la señal Wi-Fi proporcionada para que el dispositivo pueda actualizar en tiempo real las mediciones que están siendo tomadas.

3. RESULTADOS

Habiendo definido el entorno de pruebas, se realizan varios experimentos que intentan cumplir con las condiciones propuestas.

A continuación, se describen los dos experimentos realizados con su respectivo análisis de resultados.

3.1 EXPERIMENTOS REALIZADOS

Experimento número 1:

El día 27 de mayo de 2023 se pone el dispositivo en marcha, conectando la batería y suministrándole internet mediante un punto Wi-Fi externo. El dispositivo es equipado en un vehículo que seguirá una ruta de varios kilómetros por la ciudad de Alicante.

A continuación, se muestra una imagen de la ruta seguida:

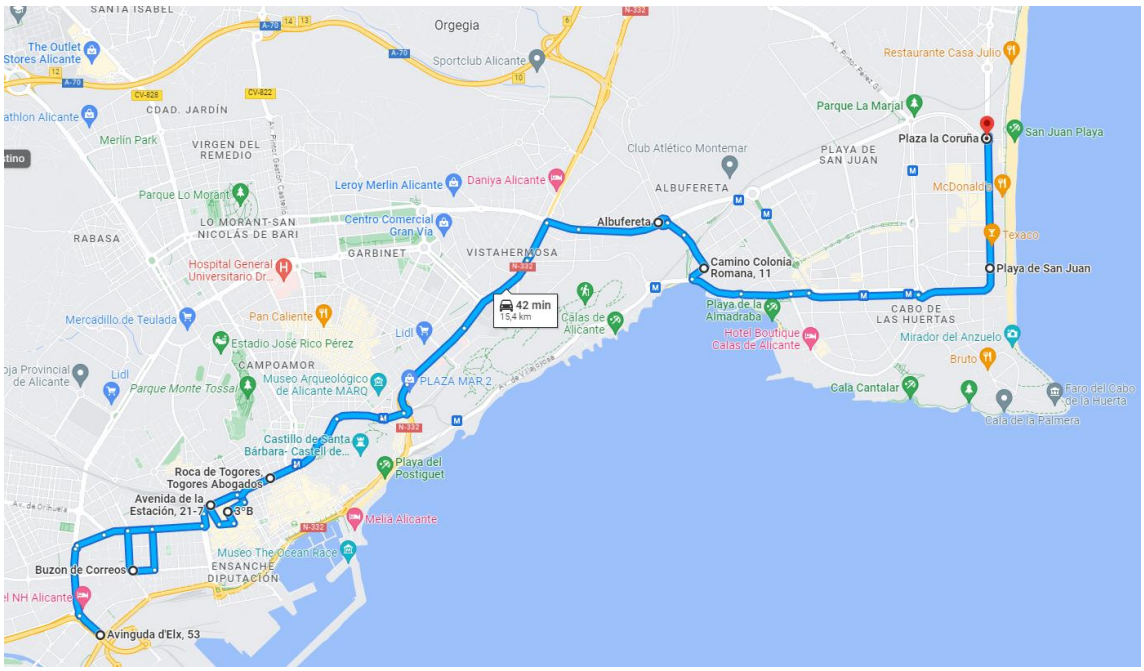


Figura 21. Ruta Experimento 1

Experimento número 2:

Durante los días 6 y 7 de junio de 2023 el dispositivo se estuvo probando por el campus de la Universidad Miguel Hernández de Elche, realizando pruebas sobre un entorno relativamente pequeño y realizando pasadas frecuentemente.

A continuación, se muestra una imagen de la zona de pruebas:

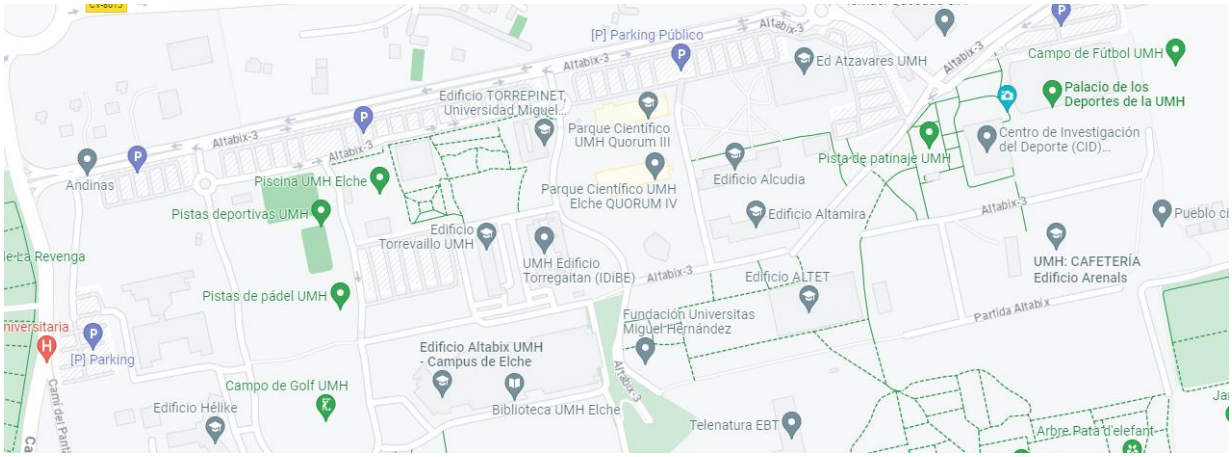


Figura 22. Ruta Experimento 2

Los objetivos de los experimentos son:

- Determinar si en la práctica, la batería será capaz de suministrar alimentación durante el tiempo necesario.

- Confirmar que las mediciones GPS son estables y precisas al comparar los puntos enviados por el dispositivo con la ruta realmente seguida por el vehículo.

- Tomar los puntos de medición suficientes para poder usar las herramientas de las que dispone la página web creada y poder de esta forma analizar los resultados.

- Mediante las herramientas disponibles, analizar los resultados conseguidos para la búsqueda de relaciones y aplicaciones donde este dispositivo pueda ser de gran utilidad.

3.2 ANÁLISIS DE RESULTADOS

Tras realizar los dos experimentos descritos anteriormente, se procede al análisis de los datos recopilados utilizando las herramientas disponibles.

Como se ha mencionado, el primer experimento se realiza el día 27 de junio; por lo tanto, desde el menú “gráficos” filtraremos por fecha para obtener los gráficos de temperatura respecto al tiempo y humedad de acuerdo con el tiempo del día de la prueba:

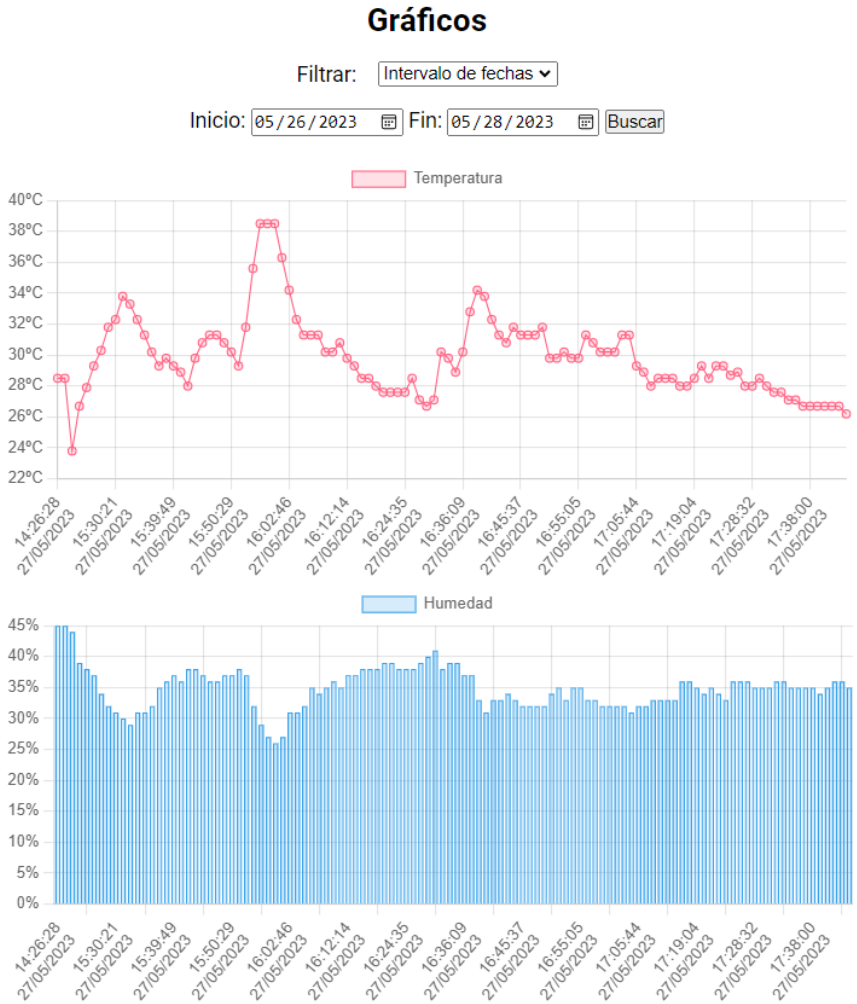
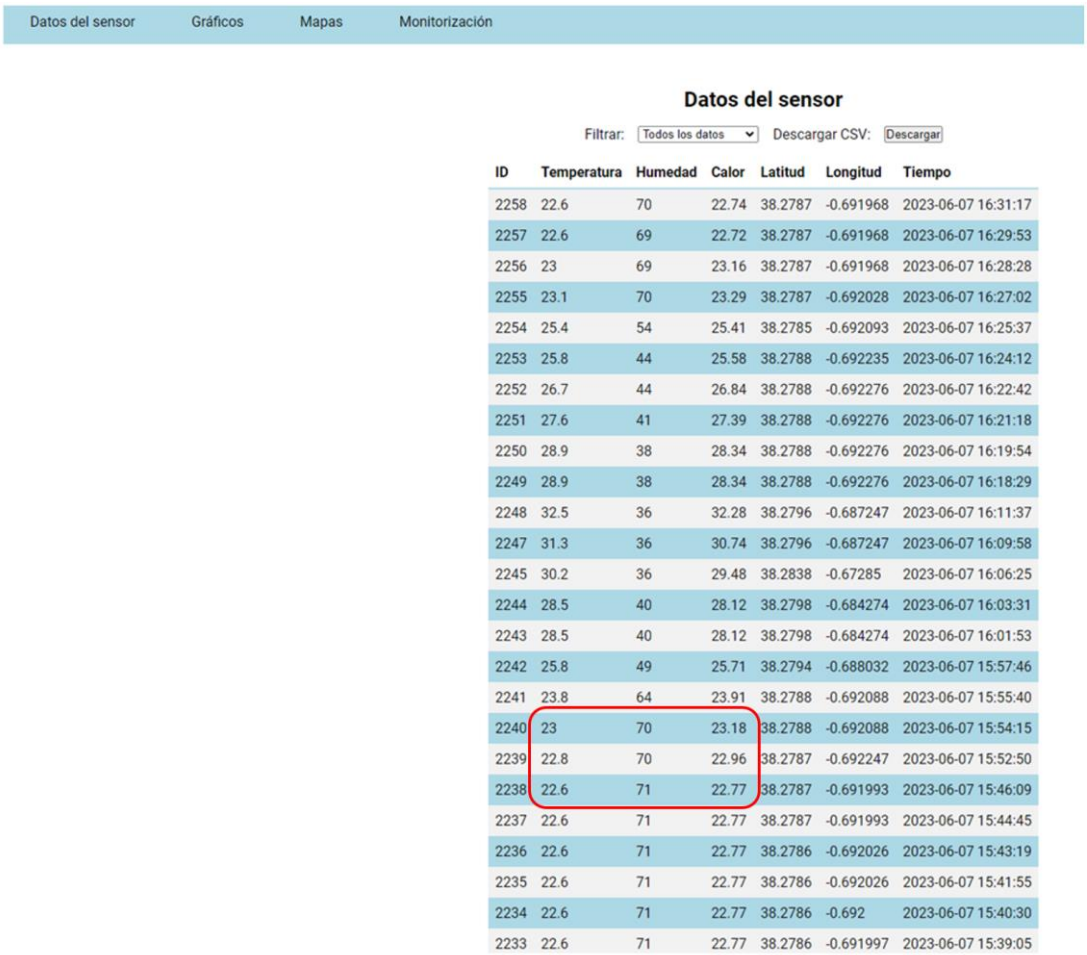


Figura 23. Temperatura y humedad, Experimento 1

Respecto a la temperatura, se puede observar claramente como ciertas lecturas destacan de manera notoria por encima de la media, con valores más elevados. Después de analizar la ruta realizada por el dispositivo y relacionado la hora de la lectura de las mediciones de temperatura con el punto de geolocalización captado para ese mismo instante, se puede concluir que el incremento de temperatura se debe a mediciones estáticas en la zona centro de la ciudad.

También es interesante observar el índice de calor como la combinación de la temperatura del aire y la humedad. Este índice, interpretado como temperatura percibida, muestra un valor mayor cuando la humedad relativa se suma a la temperatura real. En la figura adjunta se puede apreciar cómo el valor de calor aumenta, por ejemplo, con valores de humedad superiores al 70 por ciento.



ID	Temperatura	Humedad	Calor	Latitud	Longitud	Tiempo
2258	22.6	70	22.74	38.2787	-0.691968	2023-06-07 16:31:17
2257	22.6	69	22.72	38.2787	-0.691968	2023-06-07 16:29:53
2256	23	69	23.16	38.2787	-0.691968	2023-06-07 16:28:28
2255	23.1	70	23.29	38.2787	-0.692028	2023-06-07 16:27:02
2254	25.4	54	25.41	38.2785	-0.692093	2023-06-07 16:25:37
2253	25.8	44	25.58	38.2788	-0.692235	2023-06-07 16:24:12
2252	26.7	44	26.84	38.2788	-0.692276	2023-06-07 16:22:42
2251	27.6	41	27.39	38.2788	-0.692276	2023-06-07 16:21:18
2250	28.9	38	28.34	38.2788	-0.692276	2023-06-07 16:19:54
2249	28.9	38	28.34	38.2788	-0.692276	2023-06-07 16:18:29
2248	32.5	36	32.28	38.2796	-0.687247	2023-06-07 16:11:37
2247	31.3	36	30.74	38.2796	-0.687247	2023-06-07 16:09:58
2245	30.2	36	29.48	38.2838	-0.67285	2023-06-07 16:06:25
2244	28.5	40	28.12	38.2798	-0.684274	2023-06-07 16:03:31
2243	28.5	40	28.12	38.2798	-0.684274	2023-06-07 16:01:53
2242	25.8	49	25.71	38.2794	-0.688032	2023-06-07 15:57:46
2241	23.8	64	23.91	38.2788	-0.692088	2023-06-07 15:55:40
2240	23	70	23.18	38.2788	-0.692088	2023-06-07 15:54:15
2239	22.8	70	22.96	38.2787	-0.692247	2023-06-07 15:52:50
2238	22.6	71	22.77	38.2787	-0.691993	2023-06-07 15:46:09
2237	22.6	71	22.77	38.2787	-0.691993	2023-06-07 15:44:45
2236	22.6	71	22.77	38.2786	-0.692026	2023-06-07 15:43:19
2235	22.6	71	22.77	38.2786	-0.692026	2023-06-07 15:41:55
2234	22.6	71	22.77	38.2786	-0.692	2023-06-07 15:40:30
2233	22.6	71	22.77	38.2786	-0.691997	2023-06-07 15:39:05

Figura 24. Índice de calor

Mediante la herramienta “mapas” podremos verificar la observación que acabamos de realizar. Una vez filtramos las mediciones para obtener las del día del experimento 1 y tras buscar en el mapa la zona del centro de la ciudad descrita anteriormente, observamos lo siguiente:

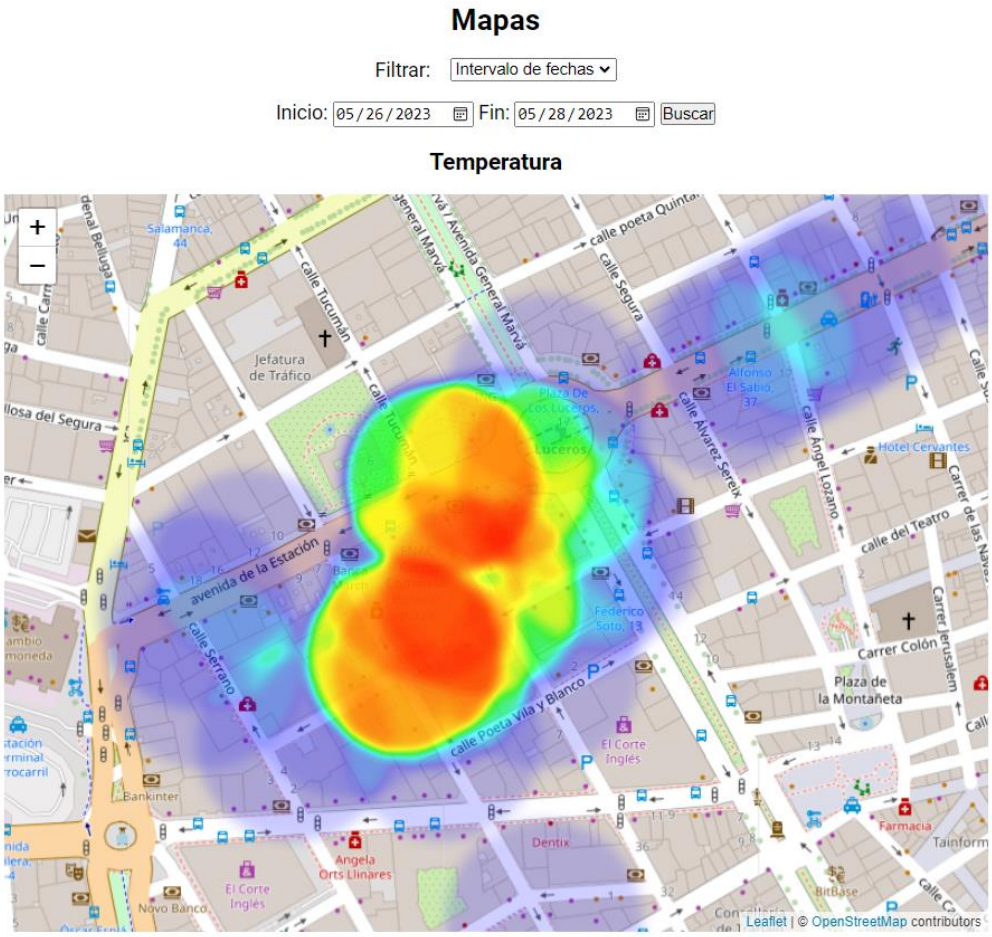


Figura 25. Mapa de temperatura del experimento 1

Teniendo una confirmación clara de que ha habido un incremento notable de temperatura en la ruta cuando se ha pasado por el centro de la ciudad.

Al tener localizada la zona donde ha ocurrido este incremento de temperatura, nos puede ser de gran utilidad la última herramienta de la que disponemos; el apartado de monitorización.

La primera acción que realizamos es acotar la zona a estudiar, en este caso se seleccionan 2 puntos del mapa que nos permitan crear un cuadrilátero.

Figura 26. Monitorización según coordenadas

Una vez agregamos los puntos, la herramienta nos creará la siguiente área:

Figura 27. Zona de Monitorización según coordenadas

En este caso concreto, nos interesará usar la siguiente configuración de filtros:

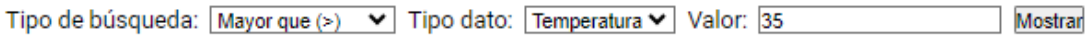


Figura 28. Filtro de Monitorización según temperatura

Ya que, de esta manera, podremos contabilizar todos los puntos en los cuales se ha superado el umbral de 35 grados dentro de la zona.

4. CONCLUSIONES

La finalidad del uso del dispositivo durante los experimentos realizados ha sido facilitar el desarrollo de conclusiones, que nos permitan dar utilidades específicas a proyecto en un ámbito de ingeniería.

4.1 ANÁLISIS DE CONCLUSIONES OBTENIDAS

Las mediciones que nos aporta el dispositivo y las herramientas configuradas para el análisis de estas nos permiten establecer relaciones entre las variables de temperatura y la humedad, que están estrechamente relacionada con varios conceptos asociados con la termodinámica que nos van a permitir aprovechar el dispositivo para un gran abanico de aplicaciones.

A continuación, se muestran algunas de las interacciones clave entre la temperatura y la humedad:

- Punto de rocío:

  El punto de rocío es la temperatura a la cual el aire se satura y alcanza su máxima capacidad para contener vapor de agua. Cuando el aire se enfría hasta alcanzar su punto de rocío, la humedad relativa llega al 100% y se produce la condensación. Esto significa que el aire ya no puede retener toda su humedad y el exceso de vapor de agua se convierte en pequeñas gotas líquidas, formando rocío o niebla.

  Para calcular este fenómeno se utilizará la siguiente formula, que se basa en las variables termodinámicas de temperatura, humedad relativa y punto de rocío.

$$T_d = T - \frac{100 - H_R}{5}$$

Donde:

- T_d es el punto de rocío en grados Celsius.
- T es la temperatura del aire en grados Celsius.
- H_R es la humedad relativa del aire en porcentaje.

- Capacidad de carga de humedad:

Mientras que el punto de rocío es una propiedad que depende solo de la temperatura y refleja el momento en el que se alcanza la saturación, la capacidad de carga de humedad o capacidad de saturación de vapor es una propiedad termodinámica que representa la máxima cantidad de vapor de agua que el aire puede contener a una temperatura y presión dadas.

La cantidad máxima de vapor de agua que el aire puede contener está directamente relacionada con su temperatura. A temperaturas más altas, el aire puede retener más humedad, mientras que, a temperaturas más bajas, su capacidad para mantener el vapor de agua disminuye. Por lo tanto, cuanto más cálida es el aire, mayor es la cantidad de humedad que puede contener.

Con el uso de las mediciones proporcionadas por el dispositivo, podremos desarrollar las siguientes fórmulas que nos permitirán hallar valores para el concepto explicado:

$$PV_S = 610.78 \times 10^{(\frac{7.5 \times T}{T+237.3})}$$

Donde:

- PV_S es la presión de vapor de saturación en Pascales.
- T es la temperatura del aire en grados Celsius

Hay que tener en cuenta que la fórmula es una aproximación y será válida para rangos de temperatura y presión atmosféricas típicas.

También es necesario añadir que esta fórmula proporciona la capacidad de carga de humedad en términos de presión de vapor de saturación. Para obtener la cantidad de vapor de agua real presente en el aire, es necesario considerar la humedad relativa, que se calcula dividiendo la presión parcial de vapor real (PVr) entre la presión de vapor de saturación (PVs) y multiplicando por 100:

$$HR = \frac{PV_r}{PV_s} \times 100$$

Donde:

- HR es la humedad relativa en porcentaje.
- PV_r es la presión parcial de vapor real en Pascales.

4.2 POSIBLES APLICACIONES DEL DISPOSITIVO

Se ha concluido que la medición de temperatura y humedad en puntos geolocalizados no solo será útil para la recopilación de datos en un entorno concreto, sino también, nos permitirá hallar conceptos como los mencionados anteriormente; herramientas que ayudan en gran medida a optimizar procesos o prevenir incidentes.

Algunas posibles aplicaciones donde el dispositivo desarrollado puede ser muy útil son las siguientes:

1. Control de clima en edificios o instalaciones:

Se va a exponer mediante un ejemplo práctico, como nuestro dispositivo puede tener una gran utilidad para la aplicación mencionada.

Tomamos como ejemplo el experimento número 2 realizado en el campus de la Universidad Miguel Hernandez de Elche. Suponemos que, durante 3 meses, cada uno de los días el dispositivo ha sido conectado, efectuando con él, una ruta previamente diseñada que haga medir temperatura y humedad relativa en puntos de interés del recinto. Tras este periodo, dispondremos de una base de datos repleta de mediciones de temperatura y humedad que abarcará todo el terreno.

Véase figuras adjuntas:

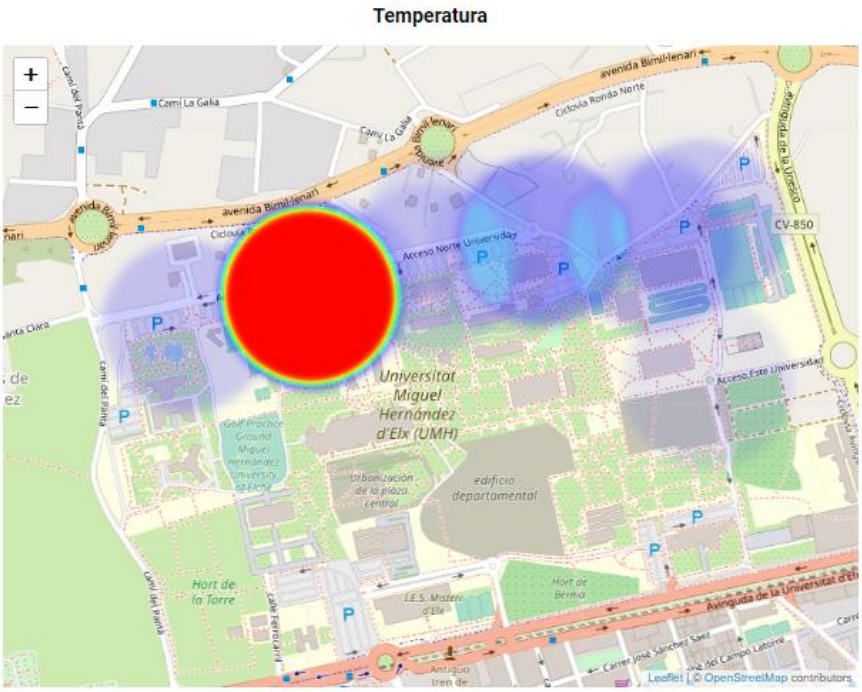


Figura 29. Control de clima, temperatura

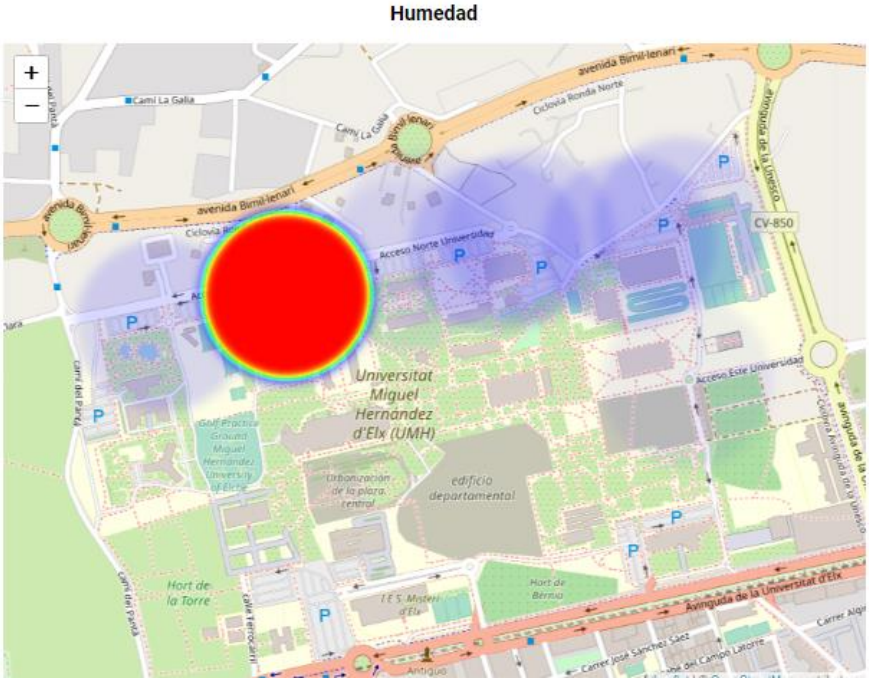


Figura 30. Control de clima, humedad

Hay que tener en cuenta que las imágenes mostradas corresponden al experimento número 2, donde el dispositivo estuvo recopilando información.

En el ejemplo que exponemos, suponemos que el tiempo de funcionamiento del sensor ha sido durante tres meses.

Dicho esto, podemos observar unas lecturas de temperatura y humedad más elevadas en la zona del edificio Torretamarit.

Cogiendo como ejemplo algunas de las mediciones asociadas a esta localización:

Dentro del apartado datos del sensor, descargaremos un fichero .csv y localizamos los datos que nos interesan:

2250	28.9	38	28.34	38.2788	-0.692276	2023-06-07 16:19:54
2249	28.9	38	28.34	38.2788	-0.692276	2023-06-07 16:18:29
2248	32.5	36	32.28	38.2796	-0.687247	2023-06-07 16:11:37
2247	31.3	36	30.74	38.2796	-0.687247	2023-06-07 16:09:58
2245	30.2	36	29.48	38.2838	-0.67285	2023-06-07 16:06:25

Figura 31. Control de clima, datos sensor

Tomando los identificadores de mediciones 2245 a 2250 tendremos los siguientes valores medios:

- Temperatura media: 30.36
- Humedad relativa media: 36.8

Y aplicando la fórmula del punto de rocío explicada anteriormente, tendremos un resultado de:

$$T_d = T - \frac{100 - H_R}{5}$$

$$T_d = 17.72$$

Esto nos indica, que si la temperatura en esa zona bajase hasta los 17.72 grados Celsius, el aire no sería capaz de contender toda su humedad y aparecería condensación.

De esta manera, sabiendo la temperatura media de punto de rocío en esa zona y teniendo la capacidad de seguir monitorizando temperatura y humedad, hemos obtenido un sistema de prevención que podría ser útil para prevenir desarrollo de moho, debilitamiento rápido desgaste de estructuras, etc.

Por otro lado, otros campos donde el ejemplo anterior también puede aportar mejoras serían:

- En sistemas de climatización y acondicionamiento de aire en edificios, conocer el punto de rocío ayuda a evitar la condensación en superficies frías y prevenir problemas de moho y humedad excesiva en el ambiente interior. Además, el control del punto de rocío permite mantener un ambiente cómodo y saludable para los ocupantes.

- Agricultura y almacenamiento de productos agrícolas: En la agricultura, el conocimiento del punto de rocío es relevante para prevenir la condensación y el desarrollo de hongos en cultivos y productos agrícolas almacenados.

- Procesos industriales: En diversos procesos industriales, como pintura, galvanización, revestimientos y otros, la detección del punto de rocío es relevante para asegurar una aplicación y curado adecuados de los materiales y prevenir problemas asociados con la humedad.

4.3 ACCIONES DE MEJORA

Tras la realización de los experimentos explicados anteriormente y después de haber analizado los resultados, hemos encontrado funcionalidades prácticas para el dispositivo que permiten el control y prevención de factores en un amplio campo de la ingeniería. Las posibles mejoras que se podrían implementar en el dispositivo son notorias, ya que aumentaría significativamente las aplicaciones en las cuales se podría ver implicado el dispositivo. A continuación, se citan las acciones de mejora más relevantes, así como algún ejemplo de aplicación a que estas mejoras darían pie.

- Integración de un módulo de comunicación móvil:

Actualmente el dispositivo se provee de internet mediante la conexión a un punto Wi-Fi externo. Este factor limitante no nos permite distanciarnos mucho del punto Wi-Fi, y de querer mover el dispositivo a grandes distancias, será necesario que el punto Wi-Fi externo también realice el mismo trayecto, condicionando el envío de mediciones. Una acción de mejora sería el añadir un módulo de comunicación móvil al dispositivo; de esta manera se autoabastecería de internet convirtiéndolo en una herramienta mucho más independiente.

Otra mejora seria añadir una unidad de almacenamiento de datos para, en caso de una pérdida temporal de la señal de comunicaciones, disponer de una copia de los datos registrados.

- Integración del dispositivo en un vehículo no tripulado (dron)

El estudio de la integración del dispositivo en un dron de tamaño reducido aumentaría de manera notoria las posibles funciones a desempeñar. Además de medir valores de temperatura, humedad y geolocalización, el dispositivo podría ampliarse con valores de altitud, velocidad, carga de batería, calidad del aire según concentración de CO_2, entre otros.

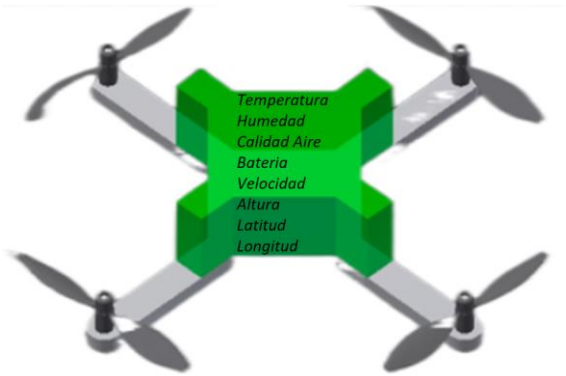


Figura 32. Instalación del dispositivo en un dron.

Para llevar a cabo esta mejora, es importante matizar que se abordaría un análisis de mecánica de fluidos para determinar la posición óptima de los sensores de modo que los flujos en las hélices del dron no produzcan interferencias en las medidas.

Con todo lo anterior se lograría ampliar la explotación del dispositivo en escenarios de interés como control del clima y monitorización de los cultivos [7], [8]. Por ejemplo, el dron podría seguir diferentes rutas programadas por campos de cultivo para, al cabo de varios meses, reunir una valiosa base de datos sobre la cual poder realizar estudios analíticos sobre cambios climáticos o qué hectáreas han experimentado cambios debido a factores externos previamente desconocidos.

5. BIBLIOGRAFÍA

[1] What is IoT (Internet of Things)?, https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iot/.Consultado en Julio 2023.

[2] El Internet de las Cosas en la asistencia sanitaria europea, https://digital-strategy.ec.europa.eu/es/policies/internet-things-european-healthcare .Consultado en Julio 2023.

[3] ¿Qué es IIOT? Internet Industrial de las Cosas, https://www.iberdrola.com/innovacion/que-es-iiot .Consultado en Julio 2023.

[4] IoT-Monitoreo Ambiental Basado: Tipos y casos de uso, https://es.digi.com/blog/post/iot-based-environmental-monitoring .Consultado en Julio 2023.

[5] Arduino. https://www.arduino.cc/en/software. Consultado en Julio 2023.

[6] MySQL. https://es.wikipedia.org/wiki/MySQL . Consultado en Julio 2023.

[7] Medición de los índices de vegetación con drones, https://www.hobbytuxtla.com/medicion-indices-vegetacion-con-drones/#:~:text=Existen%20varios%20modelos%20de%20drones,plantas%20y%20determinar%20su%20salud .Consultado en Julio 2023.

[8] Los mejores drones para la investigación meteorológica, https://ts2.space/es/los-mejores-drones-para-la-investigacion-meteorologica/ .Consultado en Julio 2023.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ANEXOS

ANEXO I: CÓDIGO ARDUINO

```
#include <TinyGPS++.h>           // librería para el sensor GPS
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ESP_Mail_Client.h>
// Librerias de cabecera para lectura del sensor DHT 11
#include "DHT.h"
#define DHTPIN 14
#define DHTTYPE DHT11
// Variables para lectura del sensor GPS
TinyGPSPlus gps;                  // objeto TinyGPS++
SoftwareSerial ss(4, 5);          // Conexion serieal al dispositivo GPS
const char* ssid = "******";      //ssid de la red wifi
const char* password = "*******"; //password de la red wifi
// Coordenadas Latitud a la estación base
const double TORRETAMARIT_LAT = 38.278605371975175;
// Coordenadas Longitud a la estación base
const double TORRETAMARIT_LNG = -0.6920385047340057;
double distanceKm;                // Distancia a la estacion base
float latitude , longitude;       // Latitud y Longitud del sensor GPS
int year , month , date, hour , minute , second;
String date_str , time_str , lat_str , lng_str, zona;
int pm;
WiFiServer server(80);
// Variables para lectura del DHT 11
float t;
float h;
float f;
float hif;
float hic;
DHT dht(DHTPIN, DHTTYPE);
HTTPClient http;  // creación del objeto http
WiFiClient client;
// Variables de Correo Electronico
/*Configuración del servidor SMTP de Gmail*/
#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465
/* Datos de acceso a cuenta. */
#define AUTHOR_EMAIL "tfgiot2023@gmail.com"
#define AUTHOR_PASSWORD "********"
/* Correo electrónico del recipiente*/
#define RECIPIENT_EMAIL "mmarcoleoncardona@gmail.com"
```

```
/* Objeto SMTP para enviar el correo electrónico */
SMTPSession smtp;
String textMsg;
float t_actual = 0;
float t_umbral30 = 22.0; // Umbral Bajo de temperatura
float t_umbral35 = 24.0; // Umbral Medio de temperatura
float t_umbral40 = 27.0; // Umbral Alto de temperatura
float d_umbral1 = 0.1; // Umbral Bajo de distancia a la Estación Base
float d_umbral5 = 0.5; // Umbral Medio de distancia a la Estación Base
float d_umbral9 = 0.9; // Umbral Alto de distancia a la Estación Base
/* Declarar la configuración de la sesión de correo electrónico */
ESP_Mail_Session session;

void setup()
{
  Serial.begin(115200);
  ss.begin(9600);                         //Sensor GPS
  dht.begin();                            //Sensor DHT 11
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);             //Conectando a red wifi
  while (WiFi.status() != WL_CONNECTED) //Mientras no hay conexion a wifi
  {
    delay(500);
    Serial.print(".");              //Intentos de conexión. print "...."
  }
  Serial.println("");
  Serial.println("WiFi connected");
  server.begin();
  Serial.println("Server started");
  Serial.println(WiFi.localIP());   // Imprimir la dirección IP

//Configuracion de Correo Electrónico
  smtp.debug(1);
  /* Configurar datos de sesión */
  ESP_Mail_Session session;
  /* Configurar la sesión */
  session.server.host_name = SMTP_HOST;
  session.server.port = SMTP_PORT;
  session.login.email = AUTHOR_EMAIL;
  session.login.password = AUTHOR_PASSWORD;
  session.login.user_domain = "";
  /* Declara la clase del mensaje */
  SMTP_Message message;
  /* Configura cabecera del mensaje */
  message.sender.name = "Tfg_IoT";
```

```
  message.sender.email = AUTHOR_EMAIL;
  message.subject = "Alerta Inicio de Funcionamiento Sensor";
  message.addRecipient("Alerta Inicio de Funcionamiento Sensor",
  RECIPIENT_EMAIL);
  //Manda texto
  textMsg = "Se ha iniciado el sistema de sensores correctamente";
  message.text.content = textMsg.c_str();
  message.text.charSet = "us-ascii";
  message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
  message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_low;
  message.response.notify = esp_mail_smtp_notify_success |
  esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
  /* Conecta al servidor */
  if (!smtp.connect(&session))
    return;
  /* Manda correo y cierra sesión */
  if (!MailClient.sendMail(&smtp, &message))
    Serial.println("Error en el envio de Email, " + smtp.errorReason());
//END Set Up de Correo Electronico
}

void loop()
{
  while (ss.available() > 0)   //Mientras los datos GPS están disponibles
    if (gps.encode(ss.read())) //Leer valores del sensor GPS
    {
      if (gps.location.isValid())
                  //Comprobar si la localización del GPS es valida
      {
        latitude = gps.location.lat();
        lat_str = String(latitude , 6);
                  // Almacenamiento de la Latitud en un string
        longitude = gps.location.lng();
        lng_str = String(longitude , 6);
                  // Almacenamiento de la Longitud en un string
        distanceKm = gps.distanceBetween(latitude, longitude,
        TORRETAMARIT_LAT, TORRETAMARIT_LNG) / 1000.0;
        Serial.print("Distancia (km) a ESTACION BASE TORRETAMARIT: ");
        Serial.println(distanceKm);
        Serial.print("Velocidad (m/s): ");
        Serial.println(gps.speed.mps()); // Velocidad en m/s (double)
        Serial.print("Velocidad (Km/s): ");
        Serial.println(gps.speed.kmph()); // Velocidad en Km/h (double)

        h = dht.readHumidity();      // Lectura del valor de humedad
        t = dht.readTemperature();   // Lectura del valor de temperatura
        f = dht.readTemperature(true);
```

```
        // Compronar si lectura ha fallado y salir (y volver a intentar)
        if (isnan(h) || isnan(t) || isnan(f)) {
          Serial.println(F("Failed to read from DHT sensor!"));
          return;
        }
        // Calculo del índice de calor en grados Fahrenheit (por defecto)
        hif = dht.computeHeatIndex(f, h);
  // Calculo del índice de calor en grados Celsius (isFahreheit = false)
        hic = dht.computeHeatIndex(t, h, false);
      //LOOP Correo Electronico
      t_actual = t;                     // Temperatura actual
      /* Configura la sesión de correo electrónico*/
      session.server.host_name = SMTP_HOST;
      session.server.port = SMTP_PORT;
      session.login.email = AUTHOR_EMAIL;
      session.login.password = AUTHOR_PASSWORD;
      session.login.user_domain = "";
      /* Declara la clase del mensaje */
      SMTP_Message message;
      /* Configura cabecera del mensaje */
      message.sender.name = "TFG_IOT_2023";
      message.sender.email = AUTHOR_EMAIL;
      //LOOP Correo Electrónico Nivel Bajo Temperatura
      if(t_actual > t_umbral30 && t_actual < t_umbral35){
        /** Habilitar la depuración, debug
        * none debug or 0
        * basic debug or 1 */
        smtp.debug(1);  //Se enviará 1 mensaje
      /* Declara la clase del mensaje */
        message.subject = "Alerta Nivel Bajo por Temperatura";
        message.addRecipient("TFG IoT 2023", RECIPIENT_EMAIL);
        //Manda texto por correo electronico
        String textMsg = "Se ha superado el umbral de temperatura para el
dispositivo con coordenadas actuales en Latitud: "+ String(lat_str) + " y
Longitud: "+ String(lng_str) + ". La temperatura actual es "+
String(t_actual) + " grados y se encuentra por encima del umbral con
recomendación inferior a "+ String(t_umbral30) + " grados.";
        message.text.content = textMsg.c_str();
        message.text.charSet = "us-ascii";
        message.text.transfer_encoding =
Content_Transfer_Encoding::enc_7bit;
        message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;
        message.response.notify = esp_mail_smtp_notify_success |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
        if (!smtp.connect(&session)) /* Conecta al servidor */
          return;
```

```
        if (!MailClient.sendMail(&smtp, &message))
      /* Manda correo y cierra sesion */
        Serial.println("Error en el envio de Email, " +
smtp.errorReason());
      }
      //LOOP Correo Electronico Nivel Medio Temperatura
      if(t_actual > t_umbral35 && t_actual < t_umbral40){
        /** Habilitar la depuración, debug
        * none debug or 0
        * basic debug or 1 */
        smtp.debug(1);  //Se enviará 1 mensaje
      /* Declara la clase del mensaje */
        message.subject = "Alerta Nivel Medio por Temperatura";
        message.addRecipient("TFG IoT 2023", RECIPIENT_EMAIL);
        //Manda texto por correo electrónico
        String textMsg = "Se ha superado el umbral de temperatura para el
dispositivo con coordenadas actuales en Latitud: "+ String(lat_str) + " y
Longitud: "+ String(lng_str) + ". La temperatura actual es "+
String(t_actual) + " grados y se encuentra por encima del umbral con
recomendación inferior a "+ String(t_umbral35) + " grados.";
        message.text.content = textMsg.c_str();
        message.text.charSet = "us-ascii";
        message.text.transfer_encoding =
Content_Transfer_Encoding::enc_7bit;
        message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;
        message.response.notify = esp_mail_smtp_notify_success |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
        if (!smtp.connect(&session)) /* Conecta al servidor */
          return;
        if (!MailClient.sendMail(&smtp, &message))
      /* Manda correo y cierra sesión */
        Serial.println("Error en el envio de Email, " +
smtp.errorReason());
      }
      //LOOP Correo Electronico Nivel Alto Temperatura
      if(t_actual > t_umbral40){
        /** Habilitar la depuración, debug
        * none debug or 0
        * basic debug or 1 */
        smtp.debug(1);  //Se enviará 1 mensaje
      /* Declara la clase del mensaje */
        message.subject = "Alerta Nivel Alto por Temperatura";
        message.addRecipient("TFG IoT 2023", RECIPIENT_EMAIL);
        //Manda texto por correo electrónico
        String textMsg = "Se ha superado el umbral de temperatura para el
dispositivo con coordenadas actuales en Latitud: "+ String(lat_str) + " y
```

```
Longitud: "+ String(lng_str) + ". SE RECOMIENDA TRAYECTORIA DE REGRESO A
ESTACION BASE. La temperatura actual es "+ String(t_actual) + " grados y
se encuentra por encima del umbral con recomendación inferior a "+
String(t_umbral40) + " grados.";
        message.text.content = textMsg.c_str();
        message.text.charSet = "us-ascii";
        message.text.transfer_encoding =
Content_Transfer_Encoding::enc_7bit;
        message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;
        message.response.notify = esp_mail_smtp_notify_success |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
        if (!smtp.connect(&session)) /* Conecta al servidor */
          return;
        if (!MailClient.sendMail(&smtp, &message))
        /* Manda correo y cierra sesión */
        Serial.println("Error en el envio de Email, " +
smtp.errorReason());
      }
      //LOOP Correo Electrónico Distancia Umbral 100m
      if(distanceKm > d_umbral1 && distanceKm < d_umbral5){
        /** Habilitar la depuración, debug
        * none debug or 0
        * basic debug or 1 */
        smtp.debug(1);  //Se enviará 1 mensaje
        /* Declarar la configuración de datos en la sesión */
        message.subject = "Alerta Nivel Bajo por Distancia";
        message.addRecipient("TFG IoT 2023", RECIPIENT_EMAIL);
        //Manda texto por correo electrónico
        String textMsg = "Alerta Nivel Bajo por Distancia Umbral. SE
RECOMIENDA OPTIMIZAR TRAYECTORIA para el dispositivo con coordenadas
actuales en Latitud: "+ String(lat_str) + " y Longitud: "+
String(lng_str) + ". La distancia actual a Estación Base "+
String(distanceKm) + " Km y se encuentra por encima del umbral por
distancia superior a "+ String(d_umbral1) + " Km.";
        message.text.content = textMsg.c_str();
        message.text.charSet = "us-ascii";
        message.text.transfer_encoding =
Content_Transfer_Encoding::enc_7bit;
        message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;
        message.response.notify = esp_mail_smtp_notify_success |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
        if (!smtp.connect(&session)) /* Conecta al servidor */
          return;
        if (!MailClient.sendMail(&smtp, &message))
        /* Manda correo y cierra sesión */
```

```
        Serial.println("Error en el envio de Email, " +
smtp.errorReason());
      }
      //LOOP Correo Electrónico Distancia Umbral 500m
      if(distanceKm > d_umbral5 && distanceKm < d_umbral9){
        /** Habilitar la depuración, debug
        * none debug or 0
        * basic debug or 1 */
        smtp.debug(1);  //Se enviará 1 mensaje
        /* Declarar la configuración de datos en la sesión */
        message.subject = "Alerta Nivel Medio por Distancia";
        message.addRecipient("TFG IoT 2023", RECIPIENT_EMAIL);
        //Manda texto por correo electrónico
        String textMsg = "Alerta Nivel Medio por Distancia Umbral. El
dispositivo con coordenadas actuales en Latitud: "+ String(lat_str) + " y
Longitud: "+ String(lng_str) + ". La distancia actual a Estación Base "+
String(distanceKm) + " Km y se encuentra por encima del umbral por
distancia superior a "+ String(d_umbral5) + " Km.";
        message.text.content = textMsg.c_str();
        message.text.charSet = "us-ascii";
        message.text.transfer_encoding =
Content_Transfer_Encoding::enc_7bit;
        message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;
        message.response.notify = esp_mail_smtp_notify_success |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
        if (!smtp.connect(&session)) /* Conecta al servidor */
          return;
        if (!MailClient.sendMail(&smtp, &message))
        /* Manda correo y cierra sesión */
        Serial.println("Error en el envio de Email, " +
smtp.errorReason());
      }
     //LOOP Correo Electronico Distancia Umbral 900m
      if(distanceKm > d_umbral9 ){
        /** Habilitar la depuración, debug
        * none debug or 0
        * basic debug or 1 */
        smtp.debug(1);  //Se enviará 1 mensaje
        /* Declarar la configuración de datos en la sesión */
        message.subject = "Alerta Nivel Alto por Distancia";
        message.addRecipient("TFG IoT 2023", RECIPIENT_EMAIL);
        //Manda texto por correo electrónico
        String textMsg = "Alerta Nivel Alto por Distancia. SE RECOMIENDA
TRAYECTORIA DE REGRESO A ESTACION BASE para el dispositivo con
coordenadas actuales en Latitud: "+ String(lat_str) + " y Longitud: "+
String(lng_str) + ". La distancia actual a Estación Base "+
```

```
String(distanceKm) + " Km y se encuentra por encima del umbral por
distancia superior a "+ String(d_umbral9) + " Km.";
        message.text.content = textMsg.c_str();
        message.text.charSet = "us-ascii";
        message.text.transfer_encoding =
Content_Transfer_Encoding::enc_7bit;
        message.priority =
esp_mail_smtp_priority::esp_mail_smtp_priority_low;
        message.response.notify = esp_mail_smtp_notify_success |
esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
        if (!smtp.connect(&session)) /* Conecta al servidor */
          return;
        if (!MailClient.sendMail(&smtp, &message))
        /* Manda correo y cierra sesion */
        Serial.println("Error en el envio de Email, " +
smtp.errorReason());
      }
      z=-1;
      String datos_a_enviar = "temperatura=" + String(t) + "&humedad=" +
String(h)+ "&calor=" + String(hic)+ "&latitud=" + String(lat_str)+
"&longitud=" + String(lng_str);
      delay(70000);
      http.begin(client,"http://tfgmarcoleon.000webhostapp.com/esppost.ph
p");     // Servidor Web
      http.addHeader("Content-Type", "application/x-www-form-
urlencoded");        // defino texto plano
      int codigo_respuesta = http.POST(datos_a_enviar);
      // Enviar datos a Servidor Web por el método POST
      if
(codigo_respuesta>0)
// Comprobar si ha ocurrido error
      {
        Serial.println("Código HTTP: "+ String(codigo_respuesta));
        if (codigo_respuesta == 200)    // todo OK, NO ha ocurrido error
        {
          String cuerpo_respuesta = http.getString();
          Serial.println("El servidor respondió: ");
          Serial.println(cuerpo_respuesta);
        }
      } else
        {
        Serial.print("Error enviado POST, código: ");
        // Mostrar código de error al enviar datos a servidor Web
        Serial.println(codigo_respuesta);
        }
      Serial.println(datos_a_enviar);
        // Mostrar datos a enviar
```

```
      http.end();
        // Liberar recursos
      }
      if
(gps.date.isValid())
//Comprobar si los datos del sensor gps date son validos
      {
        date_str = "";
        date = gps.date.day();
        month = gps.date.month();
        year = gps.date.year();
        if (date < 10)
          date_str = '0';
        date_str += String(date);
        // Valores de fecha, mes y año son almacenados en un string
        date_str += " / ";

        if (month < 10)
          date_str += '0';
        date_str += String(month);
        // Valores de fecha, mes y año son almacenados en un string
        date_str += " / ";
        if (year < 10)
          date_str += '0';
        date_str += String(year);
        // Valores de fecha, mes y año son almacenados en un string
      }
      if (gps.time.isValid())
        //Comprobar si los datos del sensor gps date son validos
      {
        time_str = "";
        hour = gps.time.hour();
             // Calculo de la hora del sensor GPS
        minute = gps.time.minute();
             // Calculo de los minutos del sensor GPS
        second = gps.time.second();
             // Calculo de los segundos del sensor GPS
        minute = (minute + 30);    // Conversion a IST
        if (minute > 59)
        {
          minute = minute - 60;
          hour = hour + 1;
        }
        hour = (hour + 5) ;
        if (hour > 23)
          hour = hour - 24;          // Conversion a IST
        if (hour >= 12)              // Comprobar si es AM o PM
```

```
          pm = 1;
        else
          pm = 0;
        hour = hour % 12;
        if (hour < 10)
          time_str = '0';
        time_str += String(hour);
      // Valores de fecha, mes y año son almacenados en un string
        time_str += " : ";
        if (minute < 10)
          time_str += '0';
        time_str += String(minute);

      // Valores de fecha, mes y año son almacenados en un string
        time_str += " : ";
        if (second < 10)
          time_str += '0';
        time_str += String(second);

      // Valores de fecha, mes y año son almacenados en un string
        if (pm == 1)
          time_str += " PM ";
        else
          time_str += " AM ";

        Serial.println();
        Serial.print("TIEMPO: ");
        Serial.println(time_str);
      }
    }

 WiFiClient client = server.available();
      // Comprobar si el cliente se ha conectado
  if (!client)
  {
    return;
  }

  delay(9000);
}
```

ANEXO II: CÓDIGO PHP PARA INTRODUCIR MEDIDAS EN BASE DE DATOS

```
<?php
$servername = "localhost";
$database = "id20727547_sensor";
$username = "id20727547_marco";
$password = "********";
// Create connection
$con = mysqli_connect($servername, $username, $password, $database);
    if ($con) {
    echo "Conexion con base de datos exitosa!!!\n";

    if(isset($_POST['temperatura'])) {
        $temperatura = $_POST['temperatura'];
        echo " Temperatura : ".$temperatura;
          }
    if(isset($_POST['calor'])) {
        $calor = $_POST['calor'];
        echo " Calor : ".$calor;
        }
    if(isset($_POST['humedad'])) {
        $humedad = $_POST['humedad'];
        echo " Humedad : ".$humedad;
        }
    if(isset($_POST['latitud'])) {
        $latitud = $_POST['latitud'];
        echo "Latitud : ".$latitud;
        }
    if(isset($_POST['longitud'])) {
        $longitud = $_POST['longitud'];
        echo "Longitud : ".$longitud;
        date_default_timezone_set('Europe/Paris');
        $fecha_actual = date("Y-m-d H:i:s");
        echo " Fecha : ".$fecha_actual;
          echo "\n<br>";
        echo "-----------------------------------------\n";
        $consulta = "INSERT INTO mediciones
(temperatura,humedad,calor,latitud,longitud,tiempo) VALUES
('$temperatura','$humedad','$calor','$latitud','$longitud','$fecha_actual
')";
        $resultado = mysqli_query($con, $consulta);
        if ($resultado){
            echo " Registro en base de datos OK! ";
          echo "\n<br>\n"; //* Esto es un salto de linea
            } else {
            echo " Falla! Registro BD\n";
            echo " Fecha de error en Registro BD: ".$fecha_actual;
                    }
    }
} else {
    echo "Falla! conexion con Base de datos ";
}
?>
```

ANEXO III: CÓDIGO PHP DATOS GRÁFICAS

```
<?php
$servername = "localhost";
$username = "id20727547_marco";
$password = "********";
$dbname = "id20727547_sensor";
// Crea la conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verifica si la conexión fue exitosa
if ($conn->connect_error) {
  die("Conexión fallida: " . $conn->connect_error);
}

// Selecciona los datos de la tabla mediciones
$sql = "SELECT temperatura, humedad, tiempo FROM mediciones ORDER BY
tiempo ASC";
$resultado = $conn->query($sql);

// Verifica si hay resultados y los convierte en un array de objetos
if ($resultado->num_rows > 0) {
  $datos = array();
  while ($fila = $resultado->fetch_assoc()) {
    $objeto = array(
      "temperatura" => $fila["temperatura"],
      "humedad" => $fila["humedad"],
      "tiempo" => $fila["tiempo"]
    );
    $datos[] = $objeto;
  }
} else {
  $datos = array(
    "error" => "No hay resultados"
  );
}

// Cierra la conexión a la base de datos
$conn->close();

// Devuelve el array de objetos como respuesta en formato JSON
header('Content-Type: application/json');
echo json_encode($datos);
?>
```

ANEXO IV: CÓDIGO PHP DATOS MAPA

```
<?php
$servername = "localhost";
$username = "id20727547_marco";
$password = "********";
$dbname = "id20727547_sensor";
// Parámetro para determinar qué datos obtener: temperatura o humedad
$tipoDato = $_GET['dato'];

// Create the database connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check if the connection was successful
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Select data from the "mediciones" table based on the parameter
if ($tipoDato == "temperatura") {
  $sql = "SELECT temperatura, latitud, longitud, tiempo FROM mediciones
ORDER BY tiempo ASC";
} elseif ($tipoDato == "humedad") {
  $sql = "SELECT humedad, latitud, longitud, tiempo FROM mediciones ORDER
BY tiempo ASC";
} else {
  $datos = array(
    "error" => "Invalid parameter"
  );
}

if (!isset($datos)) {
  // Execute the SQL query
  $resultado = $conn->query($sql);

  // Check if there are results and convert them to an array of
normalized coordinates with values
  if ($resultado->num_rows > 0) {
    $datos = array();
    $maxValor = 0;
    while ($fila = $resultado->fetch_assoc()) {
      if ($tipoDato == "temperatura") {
        $valor = floatval($fila[$tipoDato]);
      } else {
        $valor = intval($fila[$tipoDato]);
      }
      $maxValor = max($maxValor, $valor);
      $coordenadas = array(
        floatval($fila["latitud"]),
        floatval($fila["longitud"]),
        $valor
      );
      $datos[] = $coordenadas;
    }
```

```
    // Normalize the values between 0 and 1
    foreach ($datos as &$coordenadas) {
      $coordenadas[2] = $coordenadas[2] / $maxValor;
    }
    unset($coordenadas);
  } else {
    $datos = array(
      "error" => "No results found"
    );
  }
}

// Close the database connection
$conn->close();

// Return the array of normalized coordinates with values as a JSON
response
header('Content-Type: application/json');
echo json_encode($datos);
?>
```

ANEXO V: CÓDIGO PHP DATOS MAPA MONITORIZACIÓN

```
<?php
$servername = "localhost";
$username = "id20727547_marco";
$password = "********";
$dbname = "id20727547_sensor";

// Crear la conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar si la conexión fue exitosa
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Consultar la tabla para obtener los valores de latitud y longitud
$sql = "SELECT latitud, longitud FROM mediciones";
$result = $conn->query($sql);

$datos = array();

if ($result->num_rows > 0) {
    // Obtener los valores de latitud y longitud y agregarlos al array
    while ($row = $result->fetch_assoc()) {
        $coordenadas = array(
            floatval($row['latitud']),
            floatval($row['longitud'])
        );
        $datos[] = $coordenadas;
    }
}
```

```
// Cerrar la conexión a la base de datos
$conn->close();

// Devolver el array de coordenadas normalizadas como respuesta JSON
header('Content-Type: application/json');
echo json_encode($datos);
?>
```

ANEXO VI: CÓDIGO PHP TABLA MEDICIONES

```
<?php
$servername = "localhost";
$username = "id20727547_marco";
$password = "********";
$dbname = "id20727547_sensor";

// Crea la conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verifica si la conexión fue exitosa
if ($conn->connect_error) {
  die("Conexión fallida: " . $conn->connect_error);
}

// Obtiene los parámetros de filtrado si se proporcionaron
$ultimas24 = isset($_GET['ultimas24']) && $_GET['ultimas24'] === 'true';
$fecha_inicio = isset($_GET['fecha_inicio']) ? $_GET['fecha_inicio'] :
null;
$fecha_fin = isset($_GET['fecha_fin']) ? $_GET['fecha_fin'] : null;

// Construye la consulta SQL con los parámetros de filtrado
$sql = "SELECT id, temperatura, humedad, calor, latitud, longitud, tiempo
FROM mediciones";

if ($ultimas24) {
  $sql .= " WHERE tiempo >= NOW() - INTERVAL 24 HOUR";
} elseif ($fecha_inicio && $fecha_fin) {
  $sql .= " WHERE DATE(tiempo) BETWEEN '$fecha_inicio' AND '$fecha_fin'";
}

// Ejecuta la consulta SQL
$resultado = $conn->query($sql);

// Crea una tabla HTML con los datos
if ($resultado->num_rows > 0) {
  $tabla =
"<table><tr><th>ID</th><th>Temperatura</th><th>Humedad</th><th>Calor</th>
<th>Latitud</th><th>Longitud</th><th>Tiempo</th></tr>";
  while($fila = $resultado->fetch_assoc()) {
    $tabla .= "<tr><td>" . $fila["id"] . "</td><td>" .
$fila["temperatura"] . "</td><td>" . $fila["humedad"] . "</td><td>" .
$fila["calor"] . "</td><td>" . $fila["latitud"] . "</td><td>" .
$fila["longitud"] . "</td><td>" . $fila["tiempo"] . "</td></tr>";
```

```
  }
  $tabla .= "</table>";
} else {
  $tabla = "No hay resultados";
}
// Cierra la conexión a la base de datos
$conn->close();
// Devuelve la tabla HTML como respuesta
echo $tabla;
?>
```

ANEXO VII: CÓDIGO PHP TABLA MONITORIZACIÓN

```
<?php
$servername = "localhost";
$username = "id20727547_marco";
$password = "********";
$dbname = "id20727547_sensor";

// Crea la conexión a la base de datos
$conn = new mysqli($servername, $username, $password, $dbname);

// Verifica si la conexión fue exitosa
if ($conn->connect_error) {
  die("Conexión fallida: " . $conn->connect_error);
}

// Obtiene los parámetros de filtrado si se proporcionaron
$latitud1 = isset($_GET['latitud1']) ? $_GET['latitud1'] : null;
$longitud1 = isset($_GET['longitud1']) ? $_GET['longitud1'] : null;
$latitud2 = isset($_GET['latitud2']) ? $_GET['latitud2'] : null;
$longitud2 = isset($_GET['longitud2']) ? $_GET['longitud2'] : null;
$temperatura = isset($_GET['temperatura']) ? $_GET['temperatura'] : null;
$tipo_filtrado = isset($_GET['tipo_filtrado']) ? $_GET['tipo_filtrado'] :
null;
// Construye la consulta SQL con los parámetros de filtrado
$sql = "SELECT id, temperatura, humedad, calor, latitud, longitud, tiempo
FROM mediciones WHERE";

if ($latitud1 && $longitud1 && $latitud2 && $longitud2 && $temperatura &&
$tipo_filtrado) {
  $sql .= " latitud >= $latitud1 AND latitud <= $latitud2 AND longitud >=
$longitud1 AND longitud <= $longitud2 AND";
  switch ($tipo_filtrado) {
    case '>':
      $sql .= " temperatura > $temperatura";
      break;
    case '>=':
      $sql .= " temperatura >= $temperatura";
      break;
    case '<':
      $sql .= " temperatura < $temperatura";
      break;
    case '<=':
```

```
      $sql .= " temperatura <= $temperatura";
      break;
    case '=':
      $sql .= " temperatura = $temperatura";
      break;
    default:
      $sql = "No se proporcionaron correctamente los parámetros de
filtrado";
      break;
  }
} else {
  $sql = "No se proporcionaron todos los parámetros necesarios";
}
// Ejecuta la consulta SQL
$resultado = $conn->query($sql);

// Crea una tabla HTML con los datos
if ($resultado->num_rows > 0) {
  $tabla =
"<table><tr><th>ID</th><th>Temperatura</th><th>Humedad</th><th>Calor</th>
<th>Latitud</th><th>Longitud</th><th>Tiempo</th></tr>";
  while($fila = $resultado->fetch_assoc()) {
    $tabla .= "<tr><td>" . $fila["id"] . "</td><td>" .
$fila["temperatura"] . "</td><td>" . $fila["humedad"] . "</td><td>" .
$fila["calor"] . "</td><td>" . $fila["latitud"] . "</td><td>" .
$fila["longitud"] . "</td><td>" . $fila["tiempo"] . "</td></tr>";
  }
  $tabla .= "</table>";
} else {
  $tabla = "No hay resultados que cumplan los criterios de filtrado";
}

// Cierra la conexión a la base de datos
$conn->close();

// Devuelve la tabla HTML como respuesta
echo $tabla;
?>
```