Universidad de Alcalá
Escuela Politécnica Superior

Grado Ingeniería Electrónica y Automática Industrial

**Trabajo Fin de Grado**

# Estación meteorológica IoT para instalaciones fotovoltaicas

**Autor:** Guillermo Quinteiro Díez

**Tutor:** Francisco Javier Rodríguez Sánchez
**Cotutor:** Miguel Tradacete Ágreda



2023

# UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

**Grado en Ingeniería en Electrónica y Automática Industrial**

**Trabajo Fin de Grado**

**Estación meteorológica IoT para instalaciones fotovoltaicas**

**Autor:** Guillermo Quinteiro Díez

**Tutor:** Francisco Javier Rodríguez Sánchez

**Cotutor:** Miguel Tradacete Ágreda

**Tribunal:**

**Presidente:** Juan Carlos García García

**Vocal 1:** Mª Rosario Fernández Ruiz

**Vocal 2:** Fco Javier Rodríguez Sánchez

Fecha de depósito: septiembre de 2023

*La grandeza no consiste en recibir honores, sino en merecerlos*

Aristóteles

# Agradecimientos

*A todos los que han dejado su huella en este capítulo de mi vida, mi más sincero agradecimiento.*

Quisiera dedicar esta instancia para expresar mi más sincero agradecimiento a todas aquellas personas que han formado parte de mi vida durante mi periodo como estudiante de Ingeniería Electrónica y Automática Industrial, culminando con la realización de este Trabajo Fin de Grado.

Agradezco enormemente a toda mi familia y pareja por su constante paciencia y comprensión, su apoyo incondicional me ha motivado a superar todos los obstáculos y a mantenerme enfocado en el objetivo.

No puedo olvidar a mis compañeros, en especial a Javier Aguilera y Hugo Aguirre, cuya amistad y compañía han sido claves en momentos de dudas y estrés, siendo un pilar fundamental durante este bonito viaje académico.

Por último, agradecer a mi tutor, Francisco Javier Rodríguez Sánchez, por confiar en mi para llevar a cabo este importante trabajo para sus investigaciones; y a mi cotutor, Miguel Tradacete Ágreda, fundamental para el desarrollo de este proyecto, por su valioso asesoramiento, y su disposición para ayudarme en todo momento.

# Resumen

El despliegue masivo de instalaciones fotovoltaicas ha creado la necesidad de disponer de sistemas que optimicen su funcionamiento para obtener así un mejor rendimiento. Para todas ellas es prioritario tener acceso a las posibles variables meteorológicas en tiempo real, con el fin de disponer de información que ayude a reducir las pérdidas energéticas, adaptando la gestión de las plantas a las condiciones externas.

Este TFG se enmarca en los proyectos de investigación llevados a cabo en el Grupo GEISER, relativos a la optimización del funcionamiento de instalaciones fotovoltaicas, con el objetivo de mejorar su estación meteorológica, la Estación Meteo, haciéndola más completa, moderna y autosuficiente. La nueva estación cuenta con la capacidad de registrar la temperatura ambiente, presión atmosférica, humedad relativa, velocidad y dirección del viento además de la precipitación y la irradiancia; un sistema de alimentación autosuficiente; la posibilidad de continuar trabajando en periodos de desconexión Wi-Fi; y la funcionalidad de actualización remota del firmware.

**Palabras clave:** Instalaciones fotovoltaicas, variables meteorológicas a tiempo real, estación meteorológica, autosuficiente, GEISER.

# Abstract

The massive deployment of photovoltaic systems has created the need to have systems that optimise their functioning to get better results. For all of them, it is a priority to have access in real time to all the possible meteorological variables, in order to reduce energy losses.

This TFG is part of the research projects carried out in the GEISER group, related to the optimisation of photovoltaic installations, improving its weather station, the "Estación Meteo" making it more complete, modern and self-sufficient with the ability to register environmental temperature, atmospheric pressure, relative air humidity, wind speed, wind direction, precipitation and irradiance measures; a self-sufficient power supply system; the possibility of continuing to work during periods of Wi-Fi disconnection; and remote firmware update functionality.

**Key words:** Photovoltaic systems, real time meteorological variables, self-sufficient weather station, GEISER group.

# Resumen extendido

Este Trabajo Fin de Grado se centra en la creación de una estación meteorológica basada en el Internet de las Cosas (IoT) en colaboración con el Grupo de Investigación GEISER de la Universidad de Alcalá.

El proyecto se origina a partir de una estación meteorológica ya existente, la Estación Meteo. El objetivo principal es convertirla en autosuficiente, además de enriquecer y ampliar sus funcionalidades para llevar a cabo una monitorización más exhaustiva y precisa de las condiciones climáticas locales.

Inicialmente, la Estación Meteo fue diseñada para medir muchas variables meteorológicas de interés y contribuir así a las investigaciones en curso en el grupo de investigación. Estas eran la temperatura, presión atmosférica, humedad relativa, velocidad y dirección del viento. Se tomó la decisión de ampliar las funcionalidades de la estación meteorológica incorporando la capacidad de medir irradiancia y precipitación. La medición de irradiancia es esencial para las investigaciones en las que se encuentra involucrado el Grupo GEISER, el proyecto COPILOT, relacionadas con el control, supervisión y optimización de plantas fotovoltaicas.

Un aspecto esencial del desarrollo de esta estación meteorológica IoT ha sido la elección de la base de datos para almacenar y gestionar las mediciones recopiladas. Inicialmente, se planteó la utilización de ThingSpeak, la nube de Matlab, como base de datos. Finalmente, se ha optado por emplear una base de datos desarrollada por el Grupo GEISER. Esta base de datos personalizada solicita y almacena los datos enviados por la estación meteorológica mediante una API REST integrada en el software de la estación.

Un objetivo fundamental de este trabajo fue garantizar la independencia de la estación meteorológica. Para lograr esto se ha implementado un sistema que prescinde de la red eléctrica de la universidad y en menor medida de las condiciones meteorológicas. La estación es alimentada por una batería en la cual se almacena toda la energía proveniente de un panel fotovoltaico anexo. Esta innovación no solo reduce la dependencia de fuentes de energía externas, sino que también promueve la sostenibilidad y la eficiencia energética.

Continuando con la idea de hacer la Estación Meteo robusta ante problemas externos, se ha desarrollado una solución para abordar fallos de conectividad Wi-Fi que puedan surgir. Cuando la estación no pueda comunicarse con la base de datos debido a problemas de conexión, las mediciones se almacenarán en la memoria interna del microcontrolador de la estación. Una vez restablecida la conexión, los datos almacenados se transfieren a la base de datos, evitando así la pérdida de información.

Entre las mejoras incorporadas a la estación se encuentra la incorporación de una funcionalidad de actualización remota del firmware. Esta característica nos brinda la

posibilidad de actualizar el software de la estación vía Wi-Fi. Esto nos permite implementar correcciones y nuevas características de manera sencilla y eficiente, dándonos la opción de emplazar la estación en lugares más idóneos para el estudio de las variables meteorológicas, aunque sean de peor acceso.

La combinación de capacidades de medición ampliadas, independencia energética, soluciones para problemas de conexión y actualización remota del firmware forman parte de una contribución valiosa a la investigación llevada acaba en el proyecto COPILOT.

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de ecuaciones

# Lista de acrónimos

| | |
|---|---|
| TFG | Trabajo Fin de Grado |
| GEISER | Grupo de Ingeniería Electrónica Aplicada a Sistemas de Energías Renovables |
| IoT | Internet of Things |
| HIL | Hardware In the Loop |
| USB | Universal Serial Bus |
| JST | Japan Solderless Terminals |
| OMM | Organización Meteorológica Mundial |
| WMO | World Meteorological Organization |
| OMI | Organización Meteorológica Internacional |
| AEMET | Agencia Estatal de Meteorología |
| OACI | Organización de Aviación Civil Internacional |
| EMA | Estación Meteorológica Automática |
| VMM | Vigilancia Meteorológica Mundial |
| SI | Sistema internacional |
| CGPM | Conferencia General de Pesas y Medidas |
| JCGM | Comité Conjunto para las Guías en Metrología |
| RSS | Root Sum of Squares |
| MPPT | Maximum Power Point Tracker |
| ICHG | Corriente de carga total de la batería |
| SoM | System on a Module |
| SoC | System on a Chip |
| CPU | Central Processing Unit |
| PCB | Printed Circuit Board |
| SPS | Muestras por segundo |
| DC | Direct Current |
| ADC | Analog-to-Digital Converter |
| VCC | Voltage Common Collector |
| VDD | Voltage Drain Drain |
| GND | Ground |
| I2C | Inter-Integrated Circuit |
| SDA | Serial Data Line |

| | |
|---|---|
| SCL | Serial Clock Line |
| SPI | Serial Peripheral Interface |
| MISO | Master In Slave Out |
| MOSI | Master Out Slave In |
| CS | Chip Select |
| UART | Universal Asynchronous Receiver-Transmitter |
| I2S | Inter-IC Sound |
| GPIO | General Purpose Input/Output |
| PWM | Pulse Width Modulation |
| RTD | Resistance Temperature Detector |
| MOX | Metal Oxide |
| STC | Standard Test Conditions |
| ISC | Short-Circuit Current |
| RTOS | Real-Time Operating System |
| API | Application Programming Interface |
| REST | Representational State Transfer |
| HTTP | Hypertext Transfer Protocol |
| JSON | JavaScript Object Notation |
| OTA | Over The Air |
| IP | Internet Protocol |

# Capítulo 1

# 1 Introducción

En la actualidad, debido al incremento masivo de instalaciones fotovoltaicas, se ha transformado radicalmente el panorama energético mundial, trayendo una alternativa sostenible y renovable de generación de energía eléctrica. La captación de la luz solar y su conversión en energía eléctrica gracias a paneles fotovoltaicos se ha asentado como una opción a años vista para reducir el impacto ambiental que suponía depender de otras fuentes de energía convencionales.

En consecuencia, del incremento significativo de sistemas fotovoltaicos, han surgido nuevos desafíos relacionados con la eficiencia y el rendimiento óptimo de estas instalaciones. Se conoce que la generación de energía fotovoltaica se encuentra directamente relacionada con las condiciones climáticas locales, siendo estas claves para la producción de energía. Por esta razón, es crucial contar con el apoyo de tecnologías que permitan la optimización y reducción de pérdidas energéticas para asegurarnos un funcionamiento eficiente y rentable.

El Trabajo Fin de Grado aquí presente tiene como objetivo abordar esta necesidad imperante en el campo de la energía solar fotovoltaica. Para satisfacer dichos objetivos, se ha propuesto el diseño y desarrollo de una estación meteorológica que permita obtener toda la información necesaria relacionada con las condiciones de la atmósfera, en tiempo real, para el estudio de la eficiencia de instalaciones fotovoltaicas con la intención de ayudar a su adaptación a las condiciones climatológicas cambiantes.

Entre las variables meteorológicas que se precisan para un estudio de estas características se encuentran la temperatura, la presión atmosférica, la humedad relativa, la velocidad y dirección del viento, la precipitación y la irradiancia. Estas variables son esenciales para poder comprender y prever el rendimiento de los paneles bajo distintas condiciones climáticas. La capacidad de anticipación que supondrá tener un registro a lo largo del tiempo de estas características climatológicas es crítica para maximizar la eficiencia de la instalación.

Para el registro y análisis en tiempo real de los datos meteorológicos que se van a medir con la estación desarrollada en este proyecto, se contará con la posibilidad de conectividad

inalámbrica, permitiendo transferir las observaciones a un servidor web en la nube. De esta forma, los datos estarán disponibles para su acceso y procesamiento de forma remota, lo que facilitará su uso para las actualizaciones y el mantenimiento de la instalación fotovoltaica. Además, se desarrollará un sistema para no perder las observaciones realizadas en periodos de desconexión Wi-Fi, haciéndola más resiliente.

Un enfoque innovador que se marca como objetivo este proyecto es la capacidad de generar la energía que requiere la estación meteorológica de manera autónoma, coordinando un pequeño panel fotovoltaico anexo a la propia estación y una batería de litio para su almacenamiento. Toda la energía generada por este pequeño panel será la encargada de alimentar todos los sensores y equipos de medición que disponga la estación. De esta forma se logra un ciclo energético auto sostenible que reducirá la dependencia de fuentes externas de electricidad.

Este TFG se enmarca dentro de los proyectos de investigación llevados a cabo por el grupo de investigación GEISER, de la Universidad de Alcalá, relativos a la optimización del funcionamiento de instalaciones fotovoltaicas. La participación de este prestigioso grupo de investigación proporcionará un respaldo en el desarrollo del proyecto, contando con el apoyo de expertos.

En conclusión, este trabajo busca contribuir en los proyectos relacionados con la generación de energía mediante paneles fotovoltaicos gracias a una estación meteorológica automática y autosuficiente, realizando las observaciones pertinentes, registrándolas y comunicándolas de manera inalámbrica a un servidor web en la nube para su posterior análisis.

## 1.1 Aplicaciones

Anteriormente, la estación Meteo fue creada con el propósito de contribuir al proyecto HELIOS Sharing, Este proyecto se centra en la predicción de la potencia fotovoltaica a través de la irradiancia mediante el uso de redes neuronales [1].

Desde entonces, la estación ha sido utilizada en multitud de ocasiones como herramienta fundamental en los Trabajos Fin de Grado de los estudiantes de Ingeniería de la Universidad de Alcalá.

Actualmente la estación está desempeñando un papel importante en el proyecto COPILOT, del cual forma parte el grupo de investigación GEISER en colaboración con el grupo CVAR-UPM de la Universidad Politécnica de Madrid. Además, este proyecto cuenta con el respaldo de números empresas lideres en el mercado, tales como Repsol, Nokia, Mytra control, entre otras. El objetivo del proyecto COPILOT es el control, supervisión y operación optimizada de plantas fotovoltaicas mediante la integración sinérgica de drones, IoT y tecnologías avanzadas de comunicaciones.

Dentro de los desarrollos necesarios para este proyecto, se está realizando un gemelo digital de la planta fotovoltaica (panel e inversor) del equipo, el cual requiere de mediciones precisas, en tiempo real, y con un periodo de muestreo inferior a 10 segundos, de las condiciones meteorológicas locales. Es aquí donde toma relevancia la Estación Meteo, con las respectivas mejoras, desarrolladas en este Trabajo Fin de Grado [2].

## 1.2 Objetivos

Los 5 objetivos marcados para la realización de este TFG son los siguientes:

1. Diseño e implementación de una estación meteorológica capaz de medir diferentes variables meteorológicas.

Esta estación contará con la posibilidad de medir la temperatura ambiente, presión atmosférica, humedad relativa, velocidad y dirección del viento, precipitación e irradiancia, siendo esta última clave para las aplicaciones a las que va destinada dicha estación.

2. Diseño e implementación del sistema de alimentación de la estación, a partir de una batería y un pequeño panel fotovoltaico.

De esta forma, se consigue una estación independiente de fuentes externas de electricidad, para evitar situaciones de fallos en la red ajenos a la instalación.

3. Recoger la información medida por los sensores y su almacenamiento en tiempo real en la nube, pudiendo acceder a ella desde cualquier lugar.

Debido a los proyectos en los que va a estar involucrada la Estación Meteo, es necesario contar con una base de datos sólida para poder guardar un historial de observaciones a lo largo del tiempo y tener la posibilidad de acceder a ella desde cualquier lugar.

4. Diseño e implementación de un sistema capaz de seguir registrando observaciones cuando se producen fallos en la conexión, para luego ser enviados a la base de datos correspondiente.

Debido a la alta precisión en los tiempos en los que se realiza la medición, es crítico conseguir una continuidad en las observaciones y así evitar "gaps" de información que perjudique el posterior estudio. Es por esto por lo que, tanto este punto, como el "2", tienen como objetivo crear una estación lo más independiente y resiliente posible.

5. Implementar la funcionalidad de actualización remota del firmware.

Actualmente, el emplazamiento de la estación utilizada por el Grupo GEISER para sus investigaciones es poco accesible, llegando a ser peligrosa la necesidad de realizar actualizaciones de manera presencial y mediante USB. La finalidad de implementar la posibilidad de actualización remota es por seguridad y comodidad, brindando también la posibilidad de ajustar el emplazamiento de la estación a un lugar más óptimo, pero menos accesible.

## 1.3 Estructura del documento

En el presente documento se describe en 5 capítulos, todo el procedimiento realizado para alcanzar los objetivos descritos.

El **Capítulo 1**, a modo de introducción, se realiza un breve resumen del proyecto, explicando el contexto y los objetivos de este.

En el **Capítulo 2** se encuentra un estado del arte en el que se recoge toda la información que se ha recopilado tras una exhaustiva investigación en relación con todas las temáticas

concernientes al campo de la meteorología.

En el **Capítulo 3** se diseña e implementa el sistema de alimentación de la estación, analizando en detalle todos los elementos que lo componen: panel solar, batería de litio, cargador solar, convertidor DC/DC y un monitor de carga de la batería.

A continuación, en el **Capítulo 4**, se exponen los instrumentos de medida utilizados para observar las diferentes variables meteorológicas y como se encuentran implementados. Se realizará una breve comparación para verificar si estos instrumentos cumplen con los requisitos de rendimiento establecidas por la OMM.

El **Capítulo 3** y **Capítulo 4** contienen un apartado final en el que se detallan los procedimientos seguidos de forma experimental para configurar los diferentes elementos utilizados.

En el **Capítulo 5** se detalla el firmware de la estación. Se explica cómo ha sido estructurado el programa encargado de la lectura de los sensores; cómo se comunica la estación con las bases de datos para almacenar las mediciones; cómo actúa en caso de perder la conectividad Wi-Fi; y cómo se ha implementado la funcionalidad de actualización remota del firmware.

Para concluir con este proyecto, en el **Capítulo 6** podemos encontrar tanto el procedimiento como los resultados obtenidos de las pruebas realizadas a la estación meteorológica para verificar su correcto funcionamiento.

En el **Capítulo 7** se mencionan a modo de conclusión todas las impresiones sobre la realización de este Trabajo Fin de Grado y sus posibles ampliaciones en el futuro.

Por último, en el **Capítulo 8** se realizará un desglose del coste de los diferentes elementos necesarios para este trabajo y se calculará su valor final.

Al final del documento se encuentra en el **Anexo 1** el programa "main.cpp" de la Estación Meteo; en el **Anexo 2**, el archivo "Credentials.h" que contiene el programa principal y donde se encuentran los datos de la red y una función necesaria para la actualización remota del firmware; y en el **Anexo 3**, el archivo "platformio.ini" donde podrá ver la configuración del programa, librerías, etc.

## 1.4 Medios y herramientas utilizadas

Los medios y herramientas utilizadas durante el desarrollo de este trabajo se mencionan a continuación, pero sus características y funcionamiento son estudiados en los diferentes capítulos del documento. Estas herramientas las clasificaremos en función del objetivo para el que se han requerido.

Para el hardware encargado de medir las diferentes variables meteorológicas:

- Microcontrolador ESP32.
- Sensor BME680 para la temperatura, presión atmosférica y humedad relativa.
- Anemómetro y veleta 6410 de Davis Instrument para la velocidad y dirección del viento.

- Pluviómetro 6466M de Davis Instrument para el cálculo de la precipitación.
- Pequeña célula solar P121 de Voltaic Systems con una resistencia Pt100 acompañada del convertidor de resistencia a digital MAX31865 y un sensor de baja corriente ACS723LLCTR-05AB-T[2] con amplificador para medir la irradiancia.
- Conversor ADC ADS1115 necesario para las mediciones de la dirección del viento y la irradiancia.

Para la implementación del sistema de alimentación de la estación:

- Panel fotovoltaico PRT-13782.
- Batería formada por 4 celdas de litio para almacenar la energía producida por el panel.
- Cargador solar SunnyBuddy de SparkFun para cargar la batería a partir del panel fotovoltaico.
- LC70920F para el monitoreo de carga y tensión de la batería

Para el desarrollo del software de la estación:

- Entorno de trabajo para implementar el software que gobierne la estación (Ordenador y VSCode).

# Capítulo 2

# 2 Estado del arte

En este segundo capítulo se recoge toda la información recopilada en una exhaustiva y profunda investigación sobre las estaciones meteorológicas, con el objetivo de darle un fundamento teórico y una base sólida a este proyecto.

## 2.1 La Organización Meteorológica Mundial (OMM)

La Organización Meteorológica Mundial (OMM) es la portavoz y organismo de las Naciones Unidas especializado en el campo relacionado con la atmósfera terrestre, englobando el clima que produce y su interacción con el medio ambiente. Su logo se muestra en la **Figura 1**.


**Figura 1. Logo de la Organización Meteorológica Mundial**

Fundada en 1947, actualmente, la OMM cuenta con 191 estados miembros entre los cuales pertenece España, estando su sede ubicada en Ginebra.

El objetivo de esta agencia internacional es impulsar la meteorología y las ciencias geofísicas afines, con la intención de promover y facilitar la cooperación mundial en beneficio de todos. Es la meteorología una de las actividades científicas con más historia de cooperación, siendo la predecesora de la OMM la antigua Organización Meteorológica Internacional (OMI), creada en Viena en 1873 [3].

En 1939 se propondría por primera vez un nuevo Convenio Meteorológico Mundial debido a que el desarrollo económico y tecnológico de la época estaba dejando evidencias de que, por aquel entonces, la OMI, organización no gubernamental, no se había adaptado a las necesidades que aquellos tiempos requerían. No fue hasta 1947 que se pospuso debido a la segunda guerra mundial cuando se llevó a cabo la Conferencia de directores en Washington para tratar todos los aspectos relacionados con la meteorología. En las más de 300 resoluciones que se trataron se encontraban temas como: unidades, diagramas, símbolos, instrumentos, métodos de observación, redes de estaciones, publicaciones, documentos, investigación en meteorología, etc., aunque el tema más importante a tratar en la posguerra era la condición y estructura de la OMI, haciéndose esfuerzos para transformar dicha organización en un organismo intergubernamental.

El nuevo Convenio de la OMI se redactó en julio de 1946 para considerar finalmente a la OMI como organismo intergubernamental y asociarse con las recién establecidas Naciones Unidas. Se aprobó por unanimidad, se firmó el 11 de octubre de 1947 y entró en vigor el 23 de marzo de 1950, convirtiéndose oficialmente en la Organización Meteorológica Mundial en 1951 [4, 3].

### 2.1.1 Agencia Estatal de Meteorología

La Agencia Estatal de Meteorología (AEMET) es el organismo nacional español que tiene como objetivo, según el artículo 1.3 del Real Decreto 186/2008, el "desarrollo, implantación, y prestación de los servicios meteorológicos de competencia del Estado y el apoyo al ejercicio de otras políticas públicas y actividades privadas, contribuyendo a la seguridad de personas y bienes, y al bienestar y desarrollo sostenible de la sociedad española.". Su logo se muestra en la **Figura 2**.


**Figura 2. Logo de la Agencia Estatal de Meteorología**

Este organismo estatal es el sucesor desde 2008 de la Dirección General del Instituto Nacional de Meteorología, el cual contaba con ya más de 150 años de actividad. Es la encargada de representar al país frente la OMM [5].

## 2.2 Qué es una estación meteorológica

Una estación meteorológica es una instalación conformada por un conjunto de instrumentos y equipos para medir y registrar diversos parámetros meteorológicos y climáticos con regularidad.

El principal objetivo de estas es estudiar y alimentar modelos numéricos para la predicción y estudio del comportamiento meteorológico. Antaño, estas instalaciones no eran independientes. Gracias al desarrollo tecnológico hemos sido capaces de implementar

automatismos capaces de tomar y transmitir los datos por multitud de vías de comunicación, eliminando la fuerte dependencia presencial que se requería. Sumado a esto, los avances en cuanto a las energías renovables también han dado la posibilidad de permitir su instalación en lugares remotos, donde es imposible acceder con cableado eléctrico, gracias a la energía solar obtenida por medio de paneles.

Las medidas más habituales que encontramos son: temperatura, humedad relativa, presión atmosférica, velocidad y dirección del viento, precipitación… aunque muchas permiten obtener medidas no tan habituales como la radiación solar, temperatura del suelo a diferentes profundidades, distancia de las nubes…

En la **Figura 3** podemos ver un ejemplo de una estación meteorológica común.

**Figura 3. Ejemplo de estación meteorológica típica**

## 2.3 Tipos de estaciones meteorológicas

La OMM no establece una lista de tipos de estaciones meteorológicas, sin embargo, existe una clasificación general en la que se basa la OMM para describir de manera general las diferentes estaciones.

### 2.3.1 Estaciones meteorológicas sinópticas

Engloban aquellas estaciones que forman parte de una red de estaciones meteorológicas con procedimientos estandarizados para medir y registrar datos con cierta periodicidad y precisión. Con los datos proporcionados por esta clase de estaciones se elaboran mapas meteorológicos y análisis sinópticos. Pueden estar situadas en tierra o en mar y pueden estar dotadas de personal o ser automáticas.

Las estaciones terrestres deben estar situadas en una zona donde los datos recogidos sean representativos del estado de la atmósfera del territorio. El área óptima es de una hectárea aproximadamente. Es necesario situar los instrumentos meteorológicos en un lugar

abierto, alejado de construcciones y bosques, para evitar falseamientos en la información capturada por la estación. La instalación debe estar a una distancia de más de 100 metros de cualquier masa de agua para considerarse terrestre, excepto cuando se necesiten medidas costeras.

Las estaciones marítimas cobran mucha importancia debido a que el 70% de la superficie de la Tierra está cubierta por los océanos y mares. Contar con información precisa de estas zonas es muy importante para realizar las predicciones oportunas y así facilitar en gran parte los servicios destinados a las actividades marítimas. Existen subgrupos dentro de las estaciones marítimas clasificadas por sus características principales: estaciones oceánicas, estaciones de buque faros, insulares y costeras, estaciones en plataformas fijas, estaciones marítimas móviles, estaciones sobre hielo flotante, entre otras.

Las estaciones meteorológicas automáticas (EMA), son aquellas estaciones que disponen de un sistema de recopilación y registro de datos de manera automática y continua, en tiempo real, diseñadas para funcionar sin la intervención directa de observadores. Los instrumentos de medida que se utilizan en estas estaciones para medir las distintas variables, su funcionamiento y calidad, se describen en la Guía de Instrumentos y Métodos de Observación Meteorológicos (OMM–N° 8), el cual estudiaremos con mayor profundidad en el apartado 2.4 de este documento. Las estaciones automáticas son útiles para medir las condiciones climatológicas de lugares hostiles, y, por consiguiente, se debe realizar un estudio del entorno para que el sistema aguante condiciones extremas y sea fiable, debiendo ser probada su fiabilidad para que el tiempo medio entre averías no sea superior a las 10 000 horas. Comúnmente, para mejorar esta fiabilidad se utilizan sistemas duplicados parciales o totales de la estación principal, siendo estos de bajo costo, para dar apoyo y observar las variables de mayor interés. Para mayor prevención de riesgos es conveniente contar con un suministro de energía eléctrica y unos canales de comunicación diferenciados. Una característica de esta filosofía de duplicación es que tanto el sistema primario como el secundario estarán en funcionamiento simultáneamente de manera continua excepto cuando uno de ellos se encuentre fuera de servicio. Por lo general, la duplicación parcial o total tiende a ser costosa y solamente es justificable si no existe un servicio de mantenimiento adecuado que garantice medidas correctivas dentro de un plazo razonable.

Los sensores básicos que componen una EMA son: temperatura, humedad relativa, presión atmosférica, velocidad y dirección del viento, radiación solar y precipitación.

### 2.3.2 Estaciones de observación en altitud

Estas estaciones, también conocidas como estaciones de radiosonda o de observación en altura, son estaciones meteorológicas que toman su lugar en la atmósfera libre, siendo diseñadas con la intención de registrar variables meteorológicas a diferentes altitudes.

Estos sistemas están formados por radiosondas transportados generalmente por un globo. Este instrumento incorpora los sensores necesarios para medir diferentes parámetros como la temperatura, presión atmosférica, la humedad, etc. Para medir la velocidad y dirección del viento se utilizan los globos pilotos. Estos globos, también llamados globos de seguimiento (**Figura 4**) llevan consigo un dispositivo de seguimiento para poder rastrear en todo momento su posición y movimiento. A diferencia de las radiosondas, estos instrumentos no recopilan datos directamente sobre las condiciones atmosféricas, son elementos valiosos para llevar a cabo experimentos y pruebas a diferentes altitudes.

**Figura 4. Fotografía de un globo de observación británico, 1908**

La observación con globos es uno de los métodos más antiguos de observación que se sigue utilizando en la actualidad.

Para la elección del emplazamiento, la OMM recomienda aplicar una serie de criterios en los que no vamos a entrar en mayor profundidad. Para más información consulte la Guía del Sistema Mundial de Observación (OMM-Nº 488).

En la **Figura 5** podemos observar un ejemplo de una instalación de observación en altitud, destacando los edificios principales: un cobertizo de inflado de globos y una oficina.

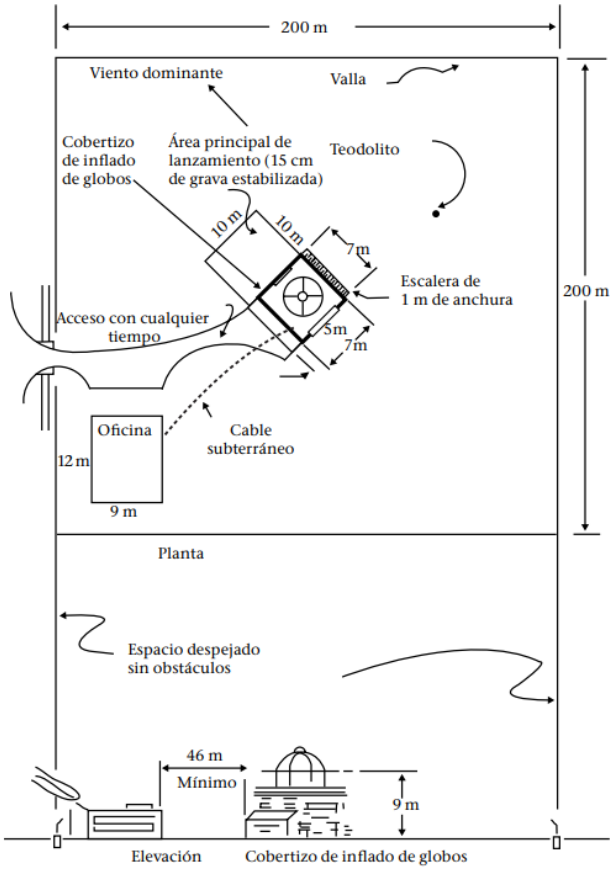


**Figura 5. Ejemplo del emplazamiento de una estación de observación en altitud**

### 2.3.3 Estaciones meteorológicas aeronáuticas

A pesar de los avances tecnológicos con respecto a la aviación, la seguridad de los vuelos sigue dependiendo de las condiciones atmosféricas y estas ejercen influencia sobre la economía y la regularidad de los vuelos comerciales. Por consiguiente, es crucial contar con información confiable sobre el estado atmosférico y de los aeródromos. Esta tarea recae sobre las estaciones meteorológicas aeronáuticas, que se encuentran además en puntos clave de la navegación aérea.

Los datos recopilados por estas estaciones son difundidos con carácter local y a otros aeródromos, según los acuerdos de conformidad entre las regiones. Los procedimientos de observación son establecidos por la OMM y la OACI (Organización de Aviación Civil Internacional). La OMM en este caso se hace responsable de proporcionar los medios para satisfacer las necesidades operativas especificadas por la OACI.

Estas estaciones utilizan de manera general los mismos instrumentos que las estaciones sinópticas (apartado 2.3.1). Sin embargo, existen algunos instrumentos como el nefobasímetro (para medir la altura de las nubes) y el transmisómetro (transmisividad atmosférica[1]) que sí utilizan normalmente las estaciones meteorológicas aeronáuticas. Estos

[1] Capacidad de la atmósfera para permitir el paso de la luz.

se muestran en la **Figura 6**.

**Figura 6. Nefobasímetro (izquierda) y transmisómetro (derecha)**

Al igual que en las estaciones automáticas, a las estaciones aeronáuticas se les exige un funcionamiento correcto en cualquier situación meteorológica, recurriendo a la instalación de instrumentos duplicados. Además, estas estaciones deben contar con fuentes auxiliares de energía eléctrica en caso de emergencia. Ha de hacerse una importante distinción entre las estaciones meteorológicas aeronáuticas y las sinópticas: estas últimas pretenden informar variables meteorológicas representativas de una zona amplia. Las observaciones meteorológicas para fines aeronáuticos se realizan en varios lugares para que representen diferentes zonas de interés y en horarios concretos, generalmente en los momentos de aterrizaje/despegue. Como cada aeródromo cuanta con sus diferencias, no se establece una norma para el emplazamiento de la estación.

### 2.3.4 Estaciones climatológicas

Estas estaciones deben proporcionar una representación de las características climáticas de todas las regiones del territorio. La red de estaciones climatológicas son competencia de cada miembro de la OMM interesado. Este deberá mantener y actualizar el catálogo de estaciones climatológicas existentes en su territorio.

La separación de en una misma región no debería ser mayor a 100 km siempre y cuando su funcionamiento sea practicable, inclusive para zonas despobladas se aplica esta norma. Sin embargo, siempre hay zonas donde la práctica de estas estaciones es inconcebible, en dicho caso deberán estar complementadas por estaciones automáticas que no requieran de personal. En ningún caso las estaciones de la red deben distar más de 500 km entre ellas. Es recomendable también para las estaciones pluviométricas (especializadas en medir la cantidad de precipitación) que se encuentren en mayor densidad.

Según el Manual del Sistema Mundial de Observación (OMM-Nº 544) las estaciones que deben componer la red de estaciones climatológicas son:

- Estaciones climatológicas de referencia: Cada miembro de la organización responsable de la red de estaciones climatológicas debe mantener una estación de referencia en cada una de las distintas regiones. Debe contar con un emplazamiento adecuado que asegure una exposición clara donde puedan efectuarse estudios representativos sin influencias locales durante un largo periodo de tiempo.
- Estaciones climatológicas principales: Estas estaciones deben ser revisadas dos veces al año (en verano e invierno) aunque de forma obligatoria deben hacerse inspecciones una vez al año. Su ubicación debe ser estratégica para determinar datos climatológicos significativos de una determinada zona geográfica para realizar análisis más detallados.
- Estaciones climatológicas ordinarias: El período de revisión de estas estaciones puede ser mucho más corto, pero en ningún caso menor a tres años. Se deben realizar inspecciones ocasionalmente con la intención de garantizar una gran calidad en las observaciones.
- Estaciones climatológicas para fines específicos: Estas estaciones tienen como objetivo realizar análisis más específicos. Vienen condicionadas por el número de variables que son capaces de observar. Las propias estaciones climatológicas para fines específicos son las que decidir su propia frecuencia, densidad y horario, sin necesidad de ser estos regulares.

### 2.3.5 Estaciones meteorológicas agrícolas

Estas estaciones tienen la finalidad de facilitar los datos necesarios para dar una representación meteorológica de las zonas agrícolas existentes.

Para cada estación de este tipo se deben especificar la biomasa natural; principal sistema agrícola y cultivos de la zona; y tipos de terreno, constantes físicas y perfiles del suelo. Las estaciones meteorológicas agrícolas tienen una clasificación similar a las climatológicas, siendo estas:

- Estación meteorológica agrícola principal.
- Estación meteorológica agrícola ordinaria.
- Estación meteorológica agrícola auxiliar.
- Estación meteorológica agrícola para fines específicos.

### 2.3.6 Estaciones especiales

Estas estaciones se utilizan para medir variables meteorológicas de especial interés con la idea de ayudar a las naciones en las que se encuentran. Además, pueden ser utilizadas para facilitar dicha información para fines generales de la Vigilancia Meteorológica Mundial.

Dentro de estas estaciones se encuentran numerosos subgrupos como son las Estaciones de radar meteorológico, estaciones de cohetes meteorológicos, estaciones de la Vigilancia de la Atmósfera Global, entre otras muchas.

## 2.4 Normas y requisitos del Sistema Mundial Integrado de Sistemas de Observación de la OMM

La OMM cuenta con un Sistema Mundial Integrado de Sistemas de Observación (WIGOS) cuyo propósito es la facilitación del intercambio y acceso a datos e información relacionados con las observaciones meteorológicas.

Debido al carácter que desempeña la estación con la que se realizará este trabajo, se hará especial hincapié en las normas de calidad y generalidades en las observaciones que deben cumplir las estaciones automáticas terrestres, puesto se ha considerado que la Estación Meteo, debido a sus capacidades, entra dentro de este grupo de estaciones. La parte que nos corresponde en WIGOS es la de la Vigilancia Meteorológica Mundial (VMM). También realizaremos un estudio y recopilaremos toda la información ligada a los instrumentos de medida de la estación.

### 2.4.1 Metadatos del WIGOS

Es importante proporcionar a los usuarios de las observaciones meteorológicas los metadatos de todas las cuestiones relacionadas con la calibración, mantenimiento, historial, exposición, etc., de la estación encargada de registrar los datos. El WIGOS establece 10 categorías para los metadatos [6].

1. Variable observada
2. Propósito de la observación
3. Estación/plataforma
4. Entorno
5. Instrumentos y métodos de observación
6. Muestreo
7. Proceso y notificación de datos
8. Calidad de los datos
9. Propiedad y política de datos
10. Datos de contacto

En la **Figura 7** se muestra un diagrama sobre el Lenguaje de Modelización Unificado de la Norma sobre metadatos del WIGOS.

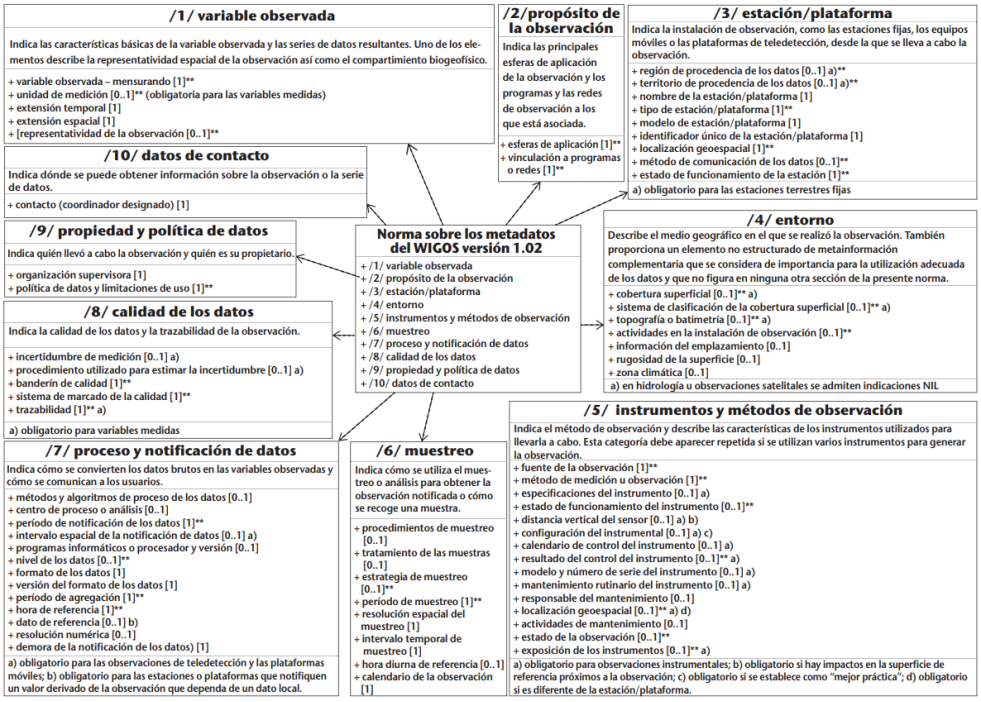


**Figura 7. Diagrama en Lenguaje de Modelización Unificado (UML) de la Norma sobre metadatos del WIGOS**

Los elementos obligatorios vienen marcados con un "[1]" a la derecha. Si estos adicionalmente llevan dos asteriscos significa que se espera una tabla de cifrado. Los opcionales o condicionales (estos últimos son obligatorios si se cumplen la condición indicada entre paréntesis) vienen indicados con un "[0..1]". Se puede indicar la falta de disponibilidad o de aplicación de algún valor con el texto "NIL".

A continuación, se harán una serie de comentarios que se han considerado oportunos destacar:

Categoría 1-02**;** Según el SI, 1.2: «Las magnitudes básicas empleadas en el SI son longitud, masa, tiempo, intensidad de corriente eléctrica, temperatura termodinámica, cantidad de sustancia e intensidad luminosa (**Tabla 1**). Las magnitudes básicas se consideran independientes, por convención. Las unidades básicas correspondientes del SI, elegidas por la CGPM, son el metro, el kilogramo, el segundo, el amperio, el kelvin, el mol y la candela». 2.2: «Las unidades derivadas se forman a partir de productos de potencias de unidades básicas. Las unidades derivadas coherentes son productos de potencias de unidades básicas en las que no interviene ningún factor numérico más que el 1».

| Magnitudes básicas | | Unidades SI básicas | |
|---|---|---|---|
| Nombre | Símbolo | Nombre | Símbolo |
| longitud | *l*, *x*, *r*, etc. | metro | m |
| masa | *m* | kilogramo | kg |
| tiempo, duración | *t* | segundo | s |
| corriente eléctrica | *I*, *i* | amperio | A |
| temperatura termodinámica | *T* | kelvin | K |
| cantidad de sustancia | *n* | mol | mol |
| intensidad luminosa | $I_v$ | candela | cd |

**Tabla 1. Unidades básicas del SI**

Categoría 3-01: La OMM divide a los miembros de la organización en regiones encargadas de coordinar todas las actividades llevadas a cabo en el territorio. Como se observa en la **Figura 8**, España se encuentra en la Región VI.

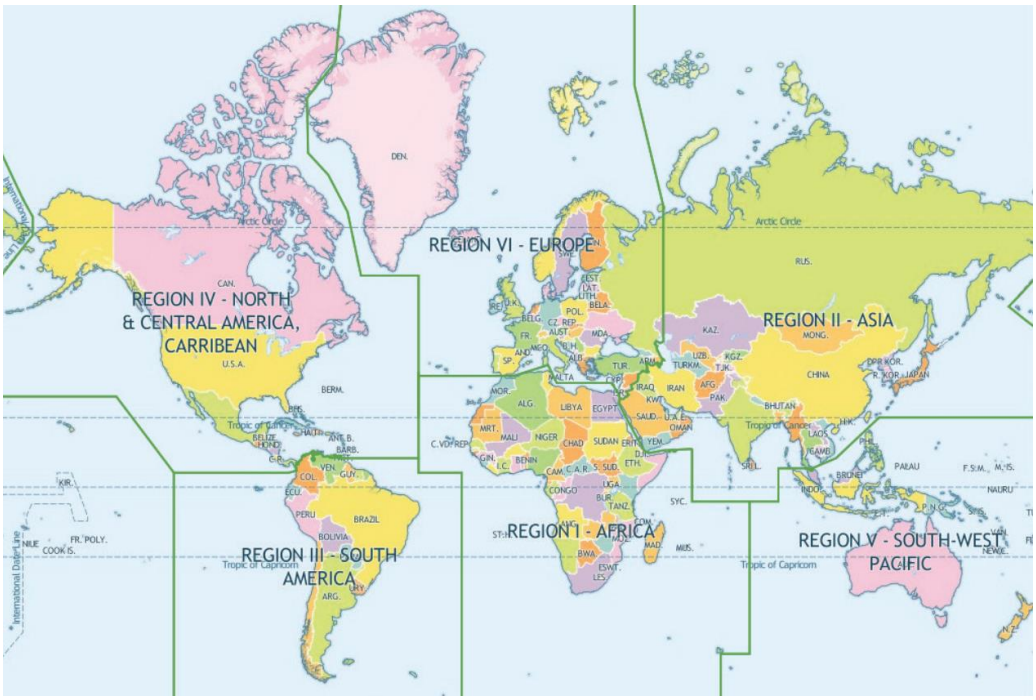


**Figura 8. División de los miembros de la OMM en regiones**

Categoría 3-06: El WIGOS asigna a cada estación un identificador único y permanente. De esta forma la instalación quedar registrada en la base de datos OSCAR/Surface para que los miembros de la OMM puedan acceder a sus metadatos.

Categoría 8-04: Hoy en día no existe un sistema de marcado para la calidad a escala mundial. Este utilizado deberá ser referenciado de algún modo; una dirección URL a un documento explicativo o un enlace a una tabla de cifrado.

## 2.4.2 Coordenadas de la estación

Es importante proporcionar toda la información acerca de la ubicación de la instalación. Por conformidad con el WIGOS, esta debe expresarse:

1) Latitud y longitud en grados, minutos y segundos enteros.
2) Altura respecto al nivel medio del mar, en metros hasta el segundo decimal.

Estas coordenadas expresan donde se encuentra la parcela desde la que ocurren las observaciones, no tiene por qué coincidir con las de la ciudad, pueblo… que nombran a la estación.

## 2.4.3 Inspección y mantenimiento

Todas las estaciones sinópticas terrestres (automáticas inclusive) deben ser inspeccionadas al menos una vez cada dos años, aunque para estaciones especiales que requieran una elevada calidad de las observaciones deben inspeccionarse en intervalos suficientemente cortos como para garantizarla.

Los objetivos de las inspecciones son asegurarse de que el emplazamiento y exposición de los instrumentos es adecuada y se encuentra bien documentada, comprobar que el estado de los instrumentos es correcto y que estos sean contrastados con instrumentos de referencia, ver si existe uniformidad en los cálculos de magnitudes derivadas, verificar si los observadores cumplen con las competencias mínimas para poder realizar sus tareas correctamente y si los metadatos se encuentren actualizados.

Los instrumentos deberán recibir un mantenimiento regular para lograr una calidad de las observaciones larga en el tiempo. Es preciso llevar un registro de las averías de los instrumentos, los cambios y correcciones en los metadatos.

## 2.4.4 Símbolos y unidades

Como se ha comentado con anterioridad en la sección de metadatos, se debe utilizar el SI como sistema de unidades en las mediciones. Hay mediciones comúnmente utilizadas en la meteorología que no han sido definidas en SI. Estas mediciones se pueden encontrar en la publicación *International Meteorological Tables* (WNO-Nº 188). Las unidades recomendadas son:

- Temperatura, $t$, en Celsius (ºC), o $T$, en grados kelvin (K)
- Presión atmosférica, $p$, en hectopascales (hPa)
- Humedad relativa, $U$, en porcentaje (%)
- Velocidad del viento en metros por segundo (m $s^{-1}$)
- Dirección del viento, en grados dextrorsum sentido horario (º) a partir del norte o en la escala 0-36, siendo 36 el viento del norte verdadero y 09 el viento del este verdadero.
- Precipitación, en milímetros (mm) o en kilogramos por metro cuadrado (kg $m^{-2}$ $s^{-1}$)
- Irradiancia, $E$, en vatios por metro cuadrado (W $m^{-2}$)

## 2.4.5 Incertidumbre y exactitud en las mediciones meteorológicas

En este apartado se recogen las definiciones más relevantes de las características que presentan los instrumentos de medición, basadas en JCGM (2012).

**Instrumento de medición.** Dispositivo utilizado para realizar mediciones, solo o en combinación con uno o varios dispositivos complementarios.

**Sensor.** Elemento de un sistema de medición que se ve directamente afectado por un fenómeno, cuerpo o sustancia que lleva una cantidad medible.

**Resultado de una medición.** Conjunto de valores de una magnitud atribuidos a un mesurando, acompañados de cualquier otra información relevante disponible.

**Exactitud (de una medición).** Término cualitativo referido a la proximidad entre un valor medido y un valor verdadero de un mesurando. La exactitud de medida se interpreta a veces como la proximidad entre los valores medidos atribuidos al mensurando. Aunque puede hablarse de la mayor o menor exactitud de un instrumento o de una medición, la medida cuantitativa de la exactitud se expresa en términos de incertidumbre.

**Incertidumbre.** Parámetro no negativo que caracteriza la dispersión de los valores atribuidos a un mesurando a partir de la información que se utiliza. (En este documento todos los valores de incertidumbre son para un 95% de confianza)

**Error (de medición).** Diferencia entre un valor medido de una magnitud y un valor de referencia

**Sensibilidad.** Cociente entre la variación de una indicación de un sistema de medida y la variación correspondiente del valor de la magnitud medida.

**Resolución.** Mínima variación de la magnitud medida que da lugar a una variación perceptible de la indicación correspondiente.

A continuación, pasaremos a especificar los requisitos mínimos establecidos para las diferentes variables meteorológicas que mediremos a lo largo del proyecto.

- Temperatura

Con el objetivo de obtener resultados representativos han de cumplirse una serie de requisitos generales de exposición a la hora de medir la temperatura del aire. El instrumento encargado de registrar dicha medida debe encontrarse en una zona lo más extensa posible, con exposición directa al sol, evitando sombras artificiales, y al viento, siendo una altura recomendada de entre 1,25 y 2 m respecto al nivel del suelo.

El rango de interés meteorológico para un sensor de temperatura es de -80 ºC a +60 ºC con una resolución de 0,1 K (siendo una diferencia de temperatura de un grado Celsius igual a un kelvin). Bien es cierto que en ocasiones y dependiendo del objetivo de la estación resulta más rentable el uso de sensores menos costosos, calibrados con arreglo a un patrón de laboratorio e introduciendo las correcciones necesarias a los valores medidos. Estas correcciones se encuentran limitadas para mantener así un margen de errores residuales y en cualquier caso deberá ir fechado el certificado de calibración. Para termómetros eléctricos, en valores de medición típicos, la diferencia entre el valor real y nominal no debe ser superior a 0,2 K.

La incertidumbre de medición requerida debe de ser 0,1 K para temperaturas de entre -40 ºC y 40 ºC y de 0,3 K para temperaturas fuera de dicho rango.

- Presión atmosférica

Además de medir el valor de la presión, se debe proporcionar también la tendencia de esta, definida como "el carácter y la cuantía de la variación de la presión atmosférica durante un período de 3 h, o de otro especificado que finalice en el momento de la observación".

La ubicación del instrumento debe escogerse cuidadosamente para que este cuente con una ausencia de corrientes de viento que puedan afectar a la medición. El montaje debe se sólido y estar bien protegido frente a vibraciones y movimientos bruscos.

La unidad básica para la presión atmosférica es el pascal (Pa), equivalente a un newton por metro cuadrado ($N\ m^{-2}$), aunque el hectopascal (hPa, equivalente a 100 Pa) se encuentra aprobado para indicar la presión con fines meteorológicos. Unidades como los milibares o milímetros de mercurio, son utilizadas también pero no pueden utilizarse para el intercambio internacional de datos.

A pesar de que la exactitud de esta variable meteorológica depende mucho de la calidad del sensor, la OMM ha establecido unos requisitos mínimos que se deben satisfacer. El intervalo de medición para la presión atmosférica debe ser de 500 hPa hasta 1 080 hPa para todos los casos, con una resolución de 0,1 hPa. La incertidumbre requerida para un buen barómetro es de 0,1hPa.

- Humedad relativa

La humedad relativa es el cociente entre la tensión de vapor observada ($e'$) — presión parcial de vapor de agua en el aire — y la tensión saturante del vapor ($e'_w$ y $e'_i$) — presiones de vapor en el aire en estado de equilibrio con la superficie de agua o de hielo, respectivamente — con respecto al agua, a la misma temperatura y presión, expresado en tanto por ciento (%).

La exposición de este instrumento debe ser similar a la de los instrumentos de temperatura, evitando la formación de microclimas cuando están protegidos por materiales como como la madera u otros sintéticos, los cuales absorben vapor de agua con la humedad atmosférica.

El rango de medición de la humedad relativa, al ser en tanto por ciento, es de 0 – 100 % con una resolución del 1%. La incertidumbre requeridos para la humedad relativa es de 1%.

- Velocidad y dirección del viento

La exposición de los instrumentos de viento es clave, y la parte más complicada de estos, para darle valor a dicha medición, puesto que la velocidad de este aumenta notoriamente con la altura. Se ha establecido una altura mínima recomendada para estos instrumentos por encima de los 10 m sobre el nivel del suelo. Se considera que, a esta altura, la velocidad y dirección del viento son representativas en la mayoría de las situaciones de terreno llano abierto. Si estas observaciones se realizan en zonas de edificios y árboles, entre otros obstáculos posibles, empiezan a carecer de valor, por lo que se recomienda siempre encontrar un emplazamiento lo más despejado posible y realizar las correcciones necesarias.

Las unidades utilizadas para la velocidad del viento son los metros por segundo ($m\ s^{-1}$). Se exige una resolución de 0,5 $m\ s^{-1}$ para un rango de 0 a 75 $m\ s^{-1}$ (270 $km\ h^{-1}$). En los informes se suele representar el promedio de un periodo de 10 minutos (este promedio puede variar según el uso objetivo de estas mediciones, utilizándose, por ejemplo, periodos más

cortos en aeródromos). La incertidumbre debe ser de 0,5 m $s^{-1}$ si nos encontramos por debajo de los 5 m $s^{-1}$ (18 km $h^{-1}$) e inferior al 10 % para velocidades por encima de los 5 m $s^{-1}$.

La velocidad del viento se debe acompañar con la escala Beaufort. Las equivalencias se encuentran en la **Tabla 2**.

| *Número de la escala Beaufort y descripción* | *Equivalencia de la velocidad del viento a una altura estándar de 10 m sobre terreno llano y despejado* | | | | *Especificaciones para estimar la velocidad sobre tierra* |
|---|---|---|---|---|---|
| | *(kt)* | *(m $s^{-1}$)* | *(km $h^{-1}$)* | *(mi $h^{-1}$)* | |
| 0 Calma | < 1 | 0 – 0,2 | < 1 | < 1 | Calma; el humo asciende verticalmente |
| 1 Ventolina | 1 – 3 | 0,3 – 1,5 | 1 – 5 | 1 – 3 | Se define la dirección del viento por la deriva del humo y no por las veletas |
| 2 Brisa ligera | 4 – 6 | 1,6 – 3,3 | 6 – 11 | 4 – 7 | El viento se siente en la cara; se mueven las hojas de los árboles; el viento mueve las veletas |
| 3 Brisa suave | 7 – 10 | 3,4 – 5,4 | 12 – 19 | 8 –12 | Las hojas y ramas pequeñas se hallan en constante movimiento; ondean las banderas livianas |
| 4 Brisa moderada | 11 – 16 | 5,5 – 7,9 | 20 – 28 | 13 – 18 | Se levantan polvo y papeles sueltos; se mueven las ramas pequeñas de los árboles |
| 5 Brisa fresca | 17 – 21 | 8,0 – 10,7 | 29 – 38 | 19 – 24 | Los árboles pequeños con hojas empiezan a moverse; se forman pequeñas olas en estanques y lagunas |
| 6 Brisa fuerte | 22 – 27 | 10,8 – 13,8 | 39 – 49 | 25 – 31 | Se mueven las ramas grandes de los árboles; silban los cables telegráficos; los paraguas se usan con dificultad |
| 7 Viento fuerte | 28 – 33 | 13,9 – 17,1 | 50 – 61 | 32 – 38 | Todos los árboles se mueven; es difícil caminar contra el viento |
| 8 Viento duro | 34 – 40 | 17,2 – 20,7 | 62 – 74 | 39 – 46 | Se rompen las ramas delgadas de los árboles; generalmente es difícil caminar contra el viento |
| 9 Viento muy duro | 41 – 47 | 20,8 – 24,4 | 75 – 88 | 47 – 54 | Se producen ligeros daños estructurales (caen chimeneas y tejas) |
| 10 Temporal | 48 – 55 | 24,5 – 28,4 | 89 – 102 | 55 – 63 | Se experimenta raramente tierra adentro; arranca árboles; hay daños estructurales considerables |
| 11 Borrasca | 56 – 63 | 28,5 – 32,6 | 103 – 117 | 64 – 72 | Muy poco frecuente; ocasiona daños generalizados |
| 12 Huracán | 64 y superior | 32,7 y superior | 118 y superior | 73 y superior | |

**Tabla 2. Equivalencias de la velocidad del viento**

Respecto a la dirección del viento, las unidades más comunes para expresarla son los grados (º), en sentido horario a partir del norte verdadero, siempre redondeados a la decena más próxima. Esta también se suele expresar como el promedio de un periodo de 10 minutos. El rango debe ser de 0º a 360º con una resolución de 1º y una incertidumbre de 5º.

- Precipitación

Para el caso de la precipitación es especialmente complicado conseguir una buena representatividad en las mediciones, siendo clave su exposición al viento y la topografía. Los metadatos para describir en qué circunstancias se han obtenido los valores de esta variable meteorológica cobran destacada importancia. Es por esto por lo que el emplazamiento idóneo son llanuras despejadas de obstáculos y al nivel del suelo para reducir así los efectos del viento.

La unidad utilizada es la profundidad lineal en milímetros (mm) — volumen/área — o kilogramos por metro cuadrado (kg $m^{-2}$) — masa/área —.

El rango de medición para esta variable es de 0 – 500 mm con una resolución mínima de 0,2 mm para valores diarios, y de 0,1 mm para valores semanales y/o mensuales. La incertidumbre no deberá exceder de 0,1 mm para valores de precipitación diaria de 5 mm y del 2% para mayores precipitaciones.

Respecto a la intensidad de la precipitación, el rango es de 0,02 – 2 000 mm $h^{-1}$, con una resolución de 0,1 mm $h^{-1}$ y una incertidumbre de medición del 5% para precipitaciones mayores a 2 mm $h^{-1}$.

- Irradiancia

Una de las variables meteorológicas que registrará la estación en la que se ha trabajado es la irradiancia global. Esta es entendida como el flujo radiante, de cualquier origen, que incide en un elemento de superficie. El instrumento utilizado para observar esta característica meteorológica suele ser el piranómetro.

La Estación Meteo no tiene integrado un piranómetro para medir la irradiancia solar, en su lugar se ha utilizado un sistema para medir la corriente de cortocircuito de una pequeña célula fotovoltaica, la cual es proporcional a la irradiancia, método comercializado bajo el nombre de sensores de irradiación solar de celda de referencia.

En la **Figura 9** se muestran los dos instrumentos más comunes para medir la irradiancia.

**Figura 9. Piranómetro convencional (izquierda) y sensor de irradiación solar de celda de referencia (derecha)**

Los sensores de irradiación solar de celda de referencia son una variante más precisa que los piranómetros convencionales ya que estos han sido calibrados en laboratorios meteorológicos con el fin de obtener mediciones más precisas y confiables. Sin duda, el sensor de irradiación solar de celda de referencia resulta más interesante para aplicaciones de monitoreo de plantas fotovoltaicas [7].

El emplazamiento recomendado para conseguir una mayor precisión con estas células, en relación con el rendimiento de una instalación fotovoltaica, debe coincidir la inclinación y orientación con los módulos de ensayo.

Estos instrumentos requieren de un cuidado especial, siendo necesario limpiar la óptica de los sensores periódicamente. Se recomienda contrarrestar la posibilidad de acumulación de rocío, escarcha o nieve, acompañándolos de ventiladores y/o calentadores de baja potencia.

Este sistema de observación, a pesar de ser común, no tiene establecidas por la OMM unas normas de calidad. Según las declaraciones de 15 laboratorios encuestados, la incertidumbre global de las células de referencia oscila entre el 1,8% y 5% [8].

Como complemento informativo sobre los piranómetros, según la OMM, para considerarse un piranómetro de alta calidad, la resolución mínima es de 1 W $m^{-2}$ con una incertidumbre máxima del 3% en una hora. Un piranómetro de calidad aceptable deberá tener como mínimo una resolución de 10 W $m^{-2}$ y una incertidumbre del 20% en una hora.

## 2.5 Estación Meteo V2

Este proyecto de fin de grado ha consistido en la optimización del sistema de adquisición de datos de la Estación Meteo, mediante la incorporación de nuevos instrumentos de medida y la implementación de funciones adicionales que hacen de esta una estación más automática. Dicha estación será empleada por el grupo de investigación GEISER (logo mostrado en la **Figura 10**), perteneciente a la Universidad de Alcalá con fines relativos a la optimización y estudio del funcionamiento de instalaciones fotovoltaicas.


**Figura 10. Logo del Grupo GEISER**

El grupo de investigación GEISER está compuesto por un equipo de 10 doctores de la Escuela Politécnica Superior además de varios investigadores, reconocido oficialmente por la Universidad de Alcalá. Este equipo de expertos desempeña un papel activo en la investigación sobre el uso de tecnologías electrónicas en sistemas de energía renovable. A raíz de sus trabajos de investigación han logrado notoriedad en el sector de las energías renovables, con publicaciones en revistas de relevancia en el sector, contribuciones a congresos, tesis doctorales, patentes, etc.

El carácter de esta estación tiene como objetivo principal dar soporte al Grupo GEISER en sus investigaciones por lo que los instrumentos de medida, y el tiempo de adquisición de datos ha sido cuidadosamente escogido para satisfacer sus necesidades.

En el Capítulo 3 de este documento se estudiarán con profundidad los instrumentos utilizados para medir las diferentes variables meteorológicas y se realizará una pequeña comparación con los requisitos que establece el WIGOS, ya estudiados en este capítulo.

# Capítulo 3

# 3 Sistema de alimentación

Uno de los objetivos planteados al comienzo del desarrollo de este proyecto consiste en convertir la Estación Meteo en una estación autosuficiente, desarrollando un sistema capaz de generar su propia energía evitando la dependencia de fuentes externas de electricidad.

El sistema que hace de esta una estación autosuficiente consiste en la implementación de un cargador solar, el cual convertirá la energía solar captada por un panel anexo a la estación en energía eléctrica. Se cuenta con una batería de litio formada por 4 celdas, donde se almacenará la energía eléctrica que alimentará la estación, pasando antes por un convertidor DC/DC para adaptar el voltaje a nuestras necesidades.

Con la idea de conseguir un mayor control, se implementará en el sistema de alimentación de la estación un chip para monitorear la tensión y carga de la batería.

En la **Figura 11** se muestra un esquema del sistema de alimentación que se utilizará para hacer energéticamente autosuficiente la Estación Meteo.

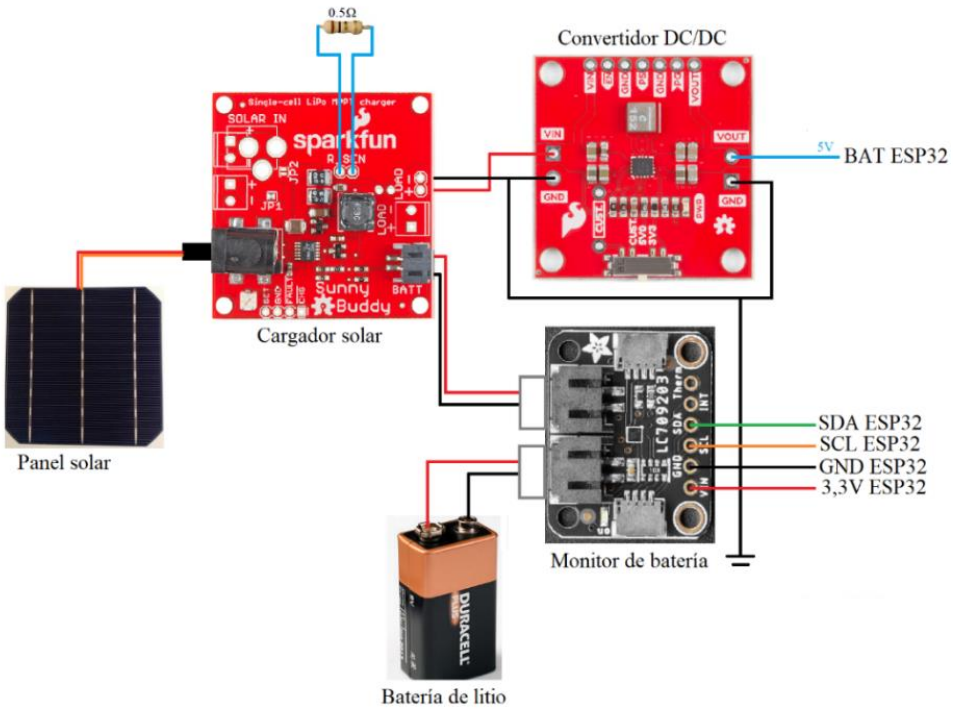


**Figura 11. Conexionado del sistema de alimentación de la estación**

A continuación, se analizará en profundidad los diferentes elementos que componen este sistema.

## 3.1 Cargador y panel fotovoltaico

Como encargado de realizar la conversión de la energía solar captada por el panel en energía eléctrica, almacenarla en la batería de litio y su posterior uso, se ha escogido el cargador solar Sunny Buddy (**Figura 12**).


**Figura 12. Cargador solar Sunny Buddy de SparkFun**

El Sunny Buddy es un cargador solar con seguimiento del punto de máxima potencia[2] (MPPT) desarrollado por SparkFun, equipado con un circuito de carga de batería LT3652 el cual funciona con tensiones de entrada desde 4,95 V hasta 32 V y es capaz de proporcionar una corriente de carga programable máxima de 2 A.

Este cargador solar está diseñado para cargar baterías de litio de una sola celda. De forma preconfigurada, el Sunny Buddy viene preparado para una corriente de carga máxima de 450 mA, brindándonos la posibilidad de, mediante un sistema de ajuste consistente en un divisor de resistencia externas, poder aumentarla para no desaprovechar la energía proporcionada por el panel [9].

El panel utilizado es el modelo PRT-13782 de SparkFun (**Figura 13**). Este panel de células monocristalinas fotovoltaicas es capaz de producir 3,5 W de potencia a pleno sol, con unos picos de tensión y corriente de 6 V a 615 mA [10].

**Figura 13. Panel fotovoltaico PRT-13782**

El PRT-13782, al tener una corriente de carga máxima mayor a la preconfigurada por el SunnyBuddy, en días muy soleados podríamos estar desaprovechando parte de la energía que nos estaría proporcionando. Ante esta situación, se decide optar por conectar en los terminales R_SENSE del Sunny Buddy una resistencia, quedando en paralelo con la resistencia interna que establece la corriente de carga (R1). De esta forma podemos configurar la corriente de carga máxima acorde a nuestro panel según la **ecuación ( 1 )** dada por el fabricante.

$$ICHGmax = {}^{0,1\ V}\!/_{R1||Rsense} > 0{,}615\ A \qquad (1)$$

Al tratarse R1 de una resistencia de 0,22 Ω se utilizará una R_SENSE de 0,62 Ω, lo que nos asegurará no desaprovechar la potencia del panel fotovoltaico, obteniendo una corriente de carga máxima de 0,616 A.

[2] Los paneles fotovoltaicos generan más energía cuando operan en un punto de voltaje y corriente específico, el cual varía debido a factores ambientales como la temperatura o la intensidad de la luz. MPPT, del inglés "Maximum Power Point Tracking", es una técnica utilizada para optimizar la eficiencia de conversión de energía ajustando constantemente la carga eléctrica del panel fotovoltaico para que opere cerca de su punto de máxima potencia.

Además, este panel de SparkFun viene ya equipado en la salida con un conector cilíndrico de 3,5 mm permitiéndonos conectarlo fácilmente al cargador solar.

El punto de máxima potencia es clave para aprovechar al máximo el rendimiento de nuestro panel fotovoltaico. Algunos fabricantes de paneles suelen indicar esta información, pero en el caso de SparkFun no es así. Para encontrar este punto conectamos nuestro panel fotovoltaico al SunnyBuddy con la batería, la carga y la R_SENSE desconectadas, y medimos la diferencia de tensión que aparece entre los pines SET y GND. A continuación, ajustamos el potenciómetro situado a la izquierda de estos pines hasta que la diferencia de tensión que aparezca sea de 3 V aproximadamente. En esta situación, el Sunny Buddy consumirá corriente hasta que lleguen 450 mA a la carga y a la batería o en caso de alcanzarse el 90% de la tensión nominal de circuito abierto a pleno sol del panel.

En el Sunny Buddy, la carga (nuestro convertidor DC/DC) debe conectarse en paralelo con la batería, la cual irá al conector JST ya instalado (**Figura 12**). Además, este cargador trae consigo unos pines de estado, no utilizados en este proyecto, configurados en binario; estos son los pines $\overline{\text{CHARGE}}$ y $\overline{\text{FAULT}}$. Los significados de estos vienen explicados en la **Tabla 3** [11].

| PINES DE ESTADO | | ESTADO DEL CARGADOR |
|---|---|---|
| $\overline{CHARGE}$ | $\overline{FAULT}$ | |
| OFF | OFF | No carga – Modo de espera o apagado |
| OFF | ON | Fallo de batería (Fallo del temporizador fin de ciclo[3]) |
| ON | OFF | Carga normal igual o superior a ICHGmax/10 |
| ON | ON | Fallo NTC (Temperatura de la batería fuera del rango configurado) |

**Tabla 3. Pines de estado del convertidor**

## 3.2 Monitorización de la batería externa

La batería utilizada para el almacenaje de la energía será un conjunto de 4 celdas de litio UR18650ZY en paralelo (**Figura 14**). A pesar de que el Sunny Buddy está diseñado para cargar baterías de litio de una sola celda, esta configuración es apta, puesto que al encontrarse en paralelo, la tensión es la misma para cada una de las celdas.

**Figura 14. Batería de litio UR18650ZY**

[3] El LT3652 otorga la posibilidad de terminar un ciclo de carga de la batería tras un determinado tiempo.

Estas baterías conforman una capacidad nominal de 2 450 mA h y 3,7 V de tensión nominal cada una, formando el conjunto de 4 de estas una batería de 9 800 mA h [12].

La corriente de carga de la batería (7 000 mA) es mucho mayor que la corriente máxima que puede proporcionar nuestro panel solar (615 mA) haciéndolos perfectamente compatibles.

Conociendo estos datos podemos estimar el tiempo de carga completo de la batería a pleno sol aplicando la **ecuación ( 2 )** siendo este de:

$$Tiempo\ de\ carga\ 0-100\% = \frac{9800\ mA\ h}{615\ mA} = 15\ horas \qquad (2)$$

Como ya se ha comentado en varias ocasiones, nuestro sistema de alimentación contará con la posibilidad de monitorear la carga y tensión de la batería. Para esta tarea se utilizará el monitor de carga LC709203F (**Figura 15**) .

Esta placa de desarrollo diseñada por Adafruit nos proporcionará información a través de I2C, acerca del voltaje y porcentaje de carga de la batería, además de, en el caso en el que nos interesase, la temperatura de la batería mediante el pin THERM.

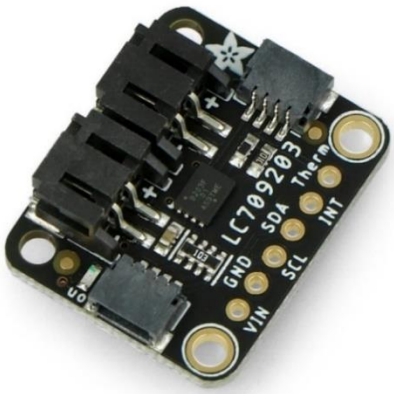


**Figura 15. Monitor de batería LC709203F de Adafruit**

El controlador LC79203F trabaja en un rango de alimentación entre 3 y 5 VDC. Para su uso cuenta con dos terminales JST ya instalados a los cuales se debe conectar la batería y el cargador solar.

Los pines SDA y SCL son los pines de datos y de reloj I2C, respectivamente, con los que podremos enviar la información para leer el voltaje y el porcentaje de la batería. Sin embargo, esta placa incluye a ambos lados unos conectores de tipo STEMMA QT compatibles con los I2C desarrollados por SparkFun (Qwiic) facilitando hacer las conexiones sin soldadura. En nuestro caso utilizaremos el conexionado convencional para la comunicación I2C.

Aparte, el LC709203F nos brinda la posibilidad de utilizar el pin THERM para poder conocer la temperatura de la batería y un pin INT para el supuesto caso en el que queramos generar una interrupción cuando el voltaje o el porcentaje de la batería caiga por debajo de un valor. En nuestro proyecto no utilizaremos ninguno de los dos pines [13].

## 3.3 Convertidor DC/DC

El encargado de gobernar la Estación Meteo es el microcontrolador ESP32 — el cual se estudiará en profundidad en el apartado 4.1 de este documento — cuya tensión de alimentación debe ser de 3,3 V. Para conseguir este voltaje nos apoyaremos en un convertidor de tensión DC/DC que controlará la tensión de salida de la batería y por tanto la entrada al sistema.

El convertidor DC/DC que utilizaremos será el Buck-Boost de SparkFun. Este se muestra en la **Figura 16**.

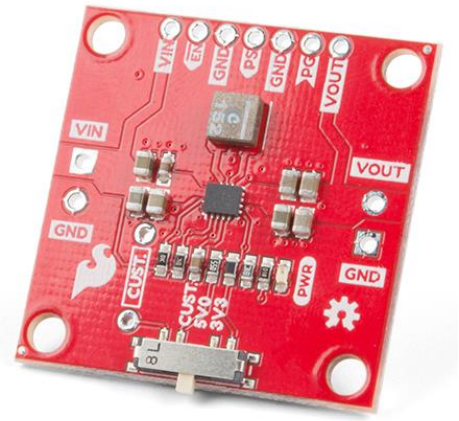


**Figura 16. Convertidor DC/DC Buck-Boost de SparkFun**

El Buck-Boost de SparkFun permite salidas configurables desde 2,5 V a 9 V. En nuestro caso, al necesitar 5 V, este convertidor tiene incorporado un switch que nos facilita seleccionar de manera muy sencilla la tensión deseada a la salida. En el caso de no ser así se le puede añadir una resistencia entre los terminales CUST. para obtener la salida deseada. La expresión para este caso viene mostrada en la **ecuación ( 3 )**:

$$R_{CUSTOM}(k\Omega) = 68{,}1 \cdot \frac{Vout}{0{,}8 - 1} \qquad (3)$$

Situaremos este interruptor para obtener una tensión a la salida de 5 V y conectaremos la salida LOAD del SunnyBuddy a la entrada del convertidor, el cual está capacitado para trabajar con tensiones de entrada de 3 a 16 V. Todas estas conexiones se encuentran representadas en la **Figura 12**.

Esta placa viene preconfigurada para funcionar en modo ahorro de energía en el momento en el que se le conecte una tensión de entrada y el consumo no sea mayor a 650 mA, sin embargo, se nos da la posibilidad de tener control sobre ella mediante los pines de

la parte superior. Con el pin ES situándolo a nivel bajo podemos deshabilitar el dispositivo y con el pin PS configurar si queremos trabajar en modo de ahorro de energía o no.

Por último, se cuenta con un pin PG (Power-good) el cual es capaz de indicar si la salida ha alcanzado su valor nominal o no dependiendo de si se encuentra a nivel alto o bajo respectivamente [14].

## 3.4 Trabajo experimental

Antes de soldar la R_SENSE y conectar el resto de los elementos, debemos ajustar nuestro cargador solar para funcionar en el punto de máxima potencia. El procedimiento seguido es el explicado ya en el apartado 3.1. Se ha observado que la tensión entre los pines SET y GND variaba según la hora del día por lo que se ha ajustado el potenciómetro hasta tener 3 V entre estos dos pines en el momento del día con mayor luz solar.

Después de añadir la resistencia R_SENSE, se conectaron el cargador solar y la batería al monitor de batería LC79203F y el convertidor DC/DC al Sunny Buddy, dejando su salida sin conectar hasta ser comprobada.

Para tener todo el sistema de alimentación listo faltaría comprobar el correcto funcionamiento del convertidor Buck-Boost. Ajustando el switch a los deseados 5 V, con un multímetro se verifican, siendo esta tensión la real.

El conexionado final se muestra en la **Figura 17**.

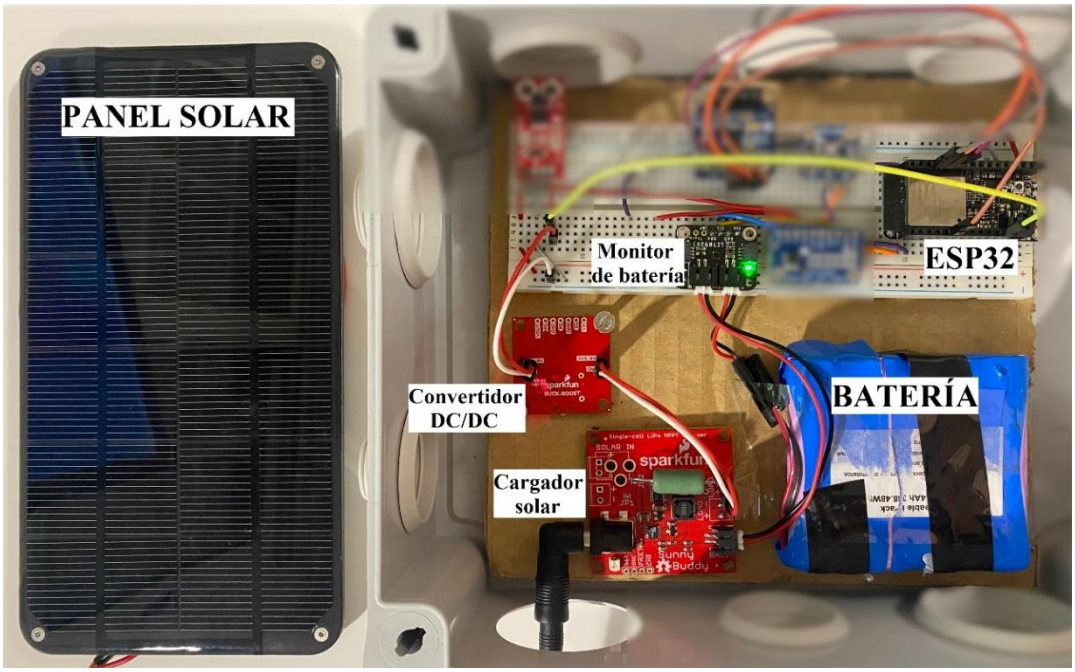


**Figura 17. Conexionado real del hardware de la estación**

# Capítulo 4

# 4 Sistema de adquisición de datos

La Estación Meteo tiene como objetivo medir de manera precisa y en tiempo real la temperatura ambiente, presión atmosférica, humedad relativa, velocidad y dirección del viento, precipitación e irradiancia.

Para ello se requiere de un microcontrolador encargado de gobernar la estación y coordinar todos los sensores e instrumentos necesarios para llevar a cabo dichas observaciones.

En este capítulo estudiaremos en profundidad el ESP32, el microcontrolador de la estación, y qué elementos y conexiones hemos utilizado para registrar las variables meteorológicas de interés.

## 4.1 El microcontrolador ESP32

Para controlar la estación meteorológica utilizaremos el módulo ESP32-WROOM-32 desarrollado por Espressif System.

El ESP32-WROOM-32 es un SoM (System on Module) el cual integra el SoC (System on a Chip) ESP32, una memoria flash, un cristal oscilador y antena Wi-Fi en PCB. Este microcontrolador de 32 bits y dos núcleos cuenta además con conectividad Wi-Fi y Bluetooth. La posibilidad de utilizar dos núcleos de manera independiente será clave para la coordinación entre la toma de datos de los diferentes instrumentos de medida y la comunicación Wi-Fi con la base de datos. Es por esto por lo que el ESP32 es ideal para nuestro proyecto.

En la práctica utilizaremos la placa de circuito impreso fabricada por Adafruit, el HUZZAH32. Este se muestra en la **Figura 18**.

Figura 18. Microcontrolador ESP32-WROOM-32 HUZZAH32

### 4.1.1 Especificaciones técnicas

La CPU que contiene este microcontrolador es una CPU Tensilica Xtensa LX6 de 32 bits de dos núcleos, controlables individualmente, con una frecuencia de operación programable hasta 240 MHz.

La memoria Flash es de 448 KB, ampliable hasta 16 MB, y su memoria SRAM interna es de 520 KB, ampliable hasta 8 MB.

En cuanto a la comunicación, los protocolos de comunicación serial con las que disponemos con el ESP32 son:

- Hasta 3 dispositivos UART.
- Clavija USB 2.0 micro-B, usada con anterioridad al tipo C para los móviles.
- 2 dispositivos I2C.
- 3 SPI.
- 2 I2S para dispositivos multimedia.

La característica más interesante en relación con nuestro proyecto es su conectividad Wi-Fi 802.11b/g/n hasta 150Mps. Cuenta también con Bluetooth v4 además de una interfaz Ethernet MAC, un bus CAN 2.0 y un driver de conexión por infrarrojo de 8 canales [15].

### 4.1.2 Pines de entrada/salida

En la **Figura 19** podemos observar un esquema de los pines de entrada y salida del microcontrolador ESP32

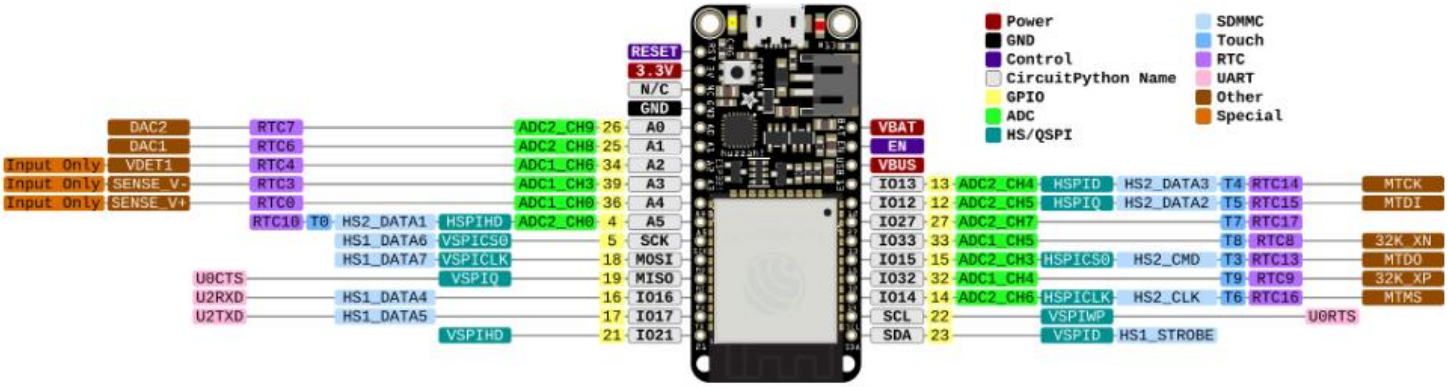


Figura 19. ESP32 Pinout

El ESP32 cuenta con 21 pines GPIO en total, de los cuales 14 son entradas analógicas que conforman los 2 ADC de 12 bits, aunque el pin A13 se encuentra oculto pues su uso está destinado para leer la tensión de la batería que alimenta al ESP32. El ADC2 solo se puede utilizar cuando la funcionalidad Wi-Fi no está activa.

De estos 21 GPIO, los pines 34, 39 y 36 solo se pueden utilizar como entradas. Los 18 restantes tienen la posibilidad de funcionar PWM y algunos GPIO tienen funcionalidades especiales para la comunicación UART, I2C, I2S y SPI.

Podemos alimentar el ESP32 mediante el puerto USB o mediante una batería de 5 V por el puerto JST ya integrado o por el pin BAT.

Con el pin EN podemos controlar los 3,3 V que nos proporciona nuestro microcontrolador con los cuales alimentaremos todos los instrumentos de medida, incluido el monitor de batería LC709203F [16].

## 4.2 Sensores

Es en este apartado donde analizaremos los instrumentos de medida utilizados en la estación para medir las diferentes variables meteorológicas de interés.

A modo de adelanto, se muestra a continuación un listado (**Tabla 4**) de los diferentes elementos que han sido utilizados para medir cada una de las variables:

| Variable meteorológica | Elementos utilizados |
|---|---|
| Temperatura ambiente | BME680 |
| Presión atmosférica | BME680 |
| Humedad relativa | BME680 |
| Velocidad del viento | Anemómetro |
| Dirección del viento | Veleta + Conversor A/D |
| Precipitación | Pluviómetro |
| Irradiancia | Célda fotovoltaico + Sensor de corriente + Conversor A/D |

**Tabla 4. Elementos utilizados para medir las diferentes variables meteorológicas**

### 4.2.1 Temperatura, presión y humedad relativa

Para las mediciones de temperatura ambiente, presión atmosférica y humedad relativa del aire se ha utilizado como instrumento de medida el sensor BME680.

Este sensor diseñado por BOSCH es un sensor digital 4 en 1 (gas, presión, temperatura y humedad) de alta precisión y bajo consumo. Su tensión de alimentación debe entrar en los rangos de 1,7 V y 3,6 V por lo que alimentarlo mediante los 3,3 V proporcionados por nuestro ESP32 es ideal. El consumo de este pequeño sensor es de 3,7 μA cuando se utilizan simultáneamente los sensores de temperatura, presión y humedad, y dispone tanto de comunicación I2C como SPI, de 3 o 4 hilos.

Respecto al sensor de temperatura, el rango de funcionamiento operativo es de -40 ºC a 85 ºC con una resolución de 0,01 ºC. En cuanto a su precisión, es máxima a los 25 ºC siendo de ±0,5 ºC; fuera de este rango la precisión es de ±1 ºC.

El BME680 es capaz de medir presiones desde 300 hPa hasta 1 100 hPa con una resolución de 0,18 Pa y una precisión de ±0,12 hPa, siendo esta menor (±0,6 hPa) cuando se trabaja entre 700-900 hPa y entre 25 ºC y 40 ºC.

La humedad relativa se mide en tanto por ciento (0 - 100%). La resolución es de 0,008% y tiene una incertidumbre de ± 3%.

Comparando las características principales de estos sensores con los estándares de calidad establecidos por la OMM para el intercambio internacional de datos podemos concluir que se cumplen excepto para la incertidumbre de la humedad.

El BME680 cuenta además con la capacidad de medir la calidad del aire (en interiores) a través del contenido en compuestos orgánicos volátiles. Esto se consigue gracias a un pequeño sensor MOX (sensor resistivo de óxidos metálicos) cuya resistencia varía al detectar gases y alcoholes como etanol, alcohol y monóxido de carbono. A pesar de contar con esta

función, al nosotros estar diseñando una estación meteorológica pensada para exterior, no la utilizaremos [17].

En la estación incorporaremos el BME680 integrado en una placa de circuito impreso desarrollada por Adafruit. Esta se puede ver en la **Figura 20**.


**Figura 20. Sensor BME680 desarrollado por Adafruit**

Este chip cuenta con 7 pines; 2 pines de alimentación (VIN y GND), un pin que nos otorga una salida de 3,3 V y 4 pines de comunicación (SCK, SDO, SDI Y CS) que nos permiten utilizar tanto I2C como SPI como protocolos de intercambio de datos. En nuestro caso, el protocolo de comunicación utilizado será el I2C [18].

En la **Figura 21** se muestran las conexiones utilizadas para incorporar el BME680 a la Estación Meteo.

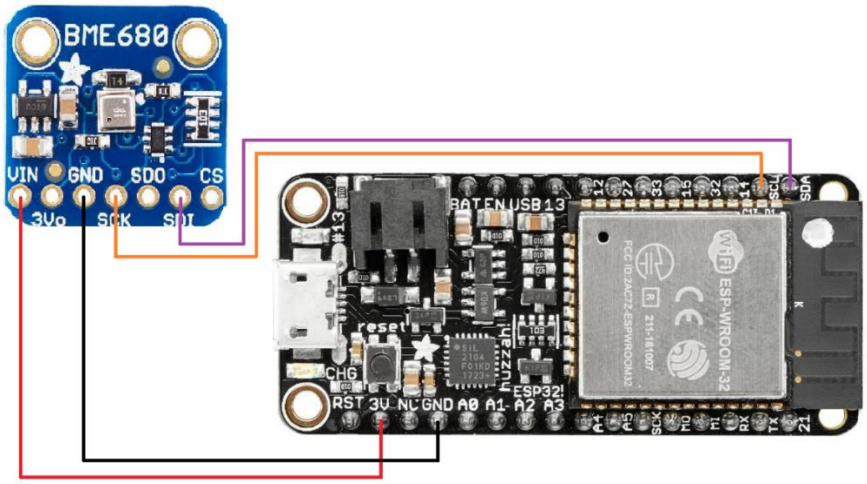


**Figura 21. Conexionado entre el BME680 y el ESP32**

### 4.2.2 Velocidad y dirección del viento

Para medir la velocidad y dirección del viento el instrumento utilizado es el anemómetro y veleta 6410 de Davis Instruments. Este instrumento cuenta con una conexión de 4 cables: 2 de alimentación, 1 salida analógica para la dirección del viento y 1 salida digital para la velocidad del viento [19, 20]. Estos elementos se encuentran representados en la **Figura 22**.


**Figura 22. Anemómetro y veleta 6410 de Davis Instrument (izquierda) y sus conexiones (derecha)**

Para el caso de la dirección del viento, la veleta tiene equipada una resistencia variable de 0 a 20 kΩ, siendo 180º 10 kΩ. Para leer y registrar en el sistema esta salida analógica utilizaremos un conversor analógico-digital externo de mayor resolución que los que se encuentran integrados en el ESP32. Estamos hablando del ADS1115 de Texas Instrument.

Este ADC de 16 bits de propósito general, con ganancia programable, cuenta con 4 canales de entrada que se pueden utilizar de manera individual o para convertir señales diferenciales. Debe ser alimentado en un rango de 2 a 5,5 V por lo que utilizaremos los 3,3 V suministrados por el ESP32. La comunicación se realiza mediante I2C con velocidad programable hasta 860 SPS.

Para la estación utilizaremos el circuito integrado desarrollado por Adafruit (**Figura 23**).

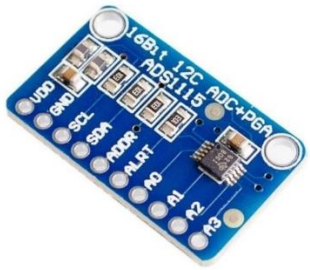


**Figura 23. ADC de 16 bits ADS1115 de Adafruit**

Contamos con 8 pines con las siguientes funcionalidades:

- Pines de alimentación: VDD y GND. Son los pines por lo que alimentaremos el ADC a través del ESP32.
- Pines de comunicación: SDA y SCL. Irán conectados a la línea de datos y a la línea de reloj de I2C de nuestro microcontrolador
- ADDR: Configura la dirección de comunicación para poder utilizar más ADS1115 en un mismo bus de comunicación. Por defecto la dirección será 0x48 (como si

estuviese conectado a GND). En nuestro caso, al no utilizar más ADS1115 no utilizaremos este pin

- ALRT: Genera interrupciones cuando la entrada alcanza un determinado valor. En nuestro caso no lo utilizaremos
- Entradas analógicas: A0, A1, A2 y A3. Se pueden utilizar como entradas individuales (referenciadas a la masa analógica) o como entradas diferencias (A0-A1 o A2-A3).

La incertidumbre de la veleta es de ±7º trabajando en un rango 360º con una resolución de 1º.

Con respecto al anemómetro, este es capaz de medir velocidades de hasta 332 km/h (89 m $s^{-1}$) con una resolución de 1 km/h (0,1 m $s^{-1}$) y una precisión de ±3 km/h (1 m $s^{-1}$) o de un ±5% (la mayor).

Comparando estas características con los requisitos de incertidumbre establecidos por la OMM encontramos que abarcamos un rango con una resolución correcta, pero con una mayor incertidumbre para las medidas de velocidad y dirección del viento.

En la **Figura 24** podemos ver como se han conectado tanto el ADS1115 como el anemómetro y la veleta al ESP32 para realizar correctamente la lectura de la velocidad y dirección del viento.

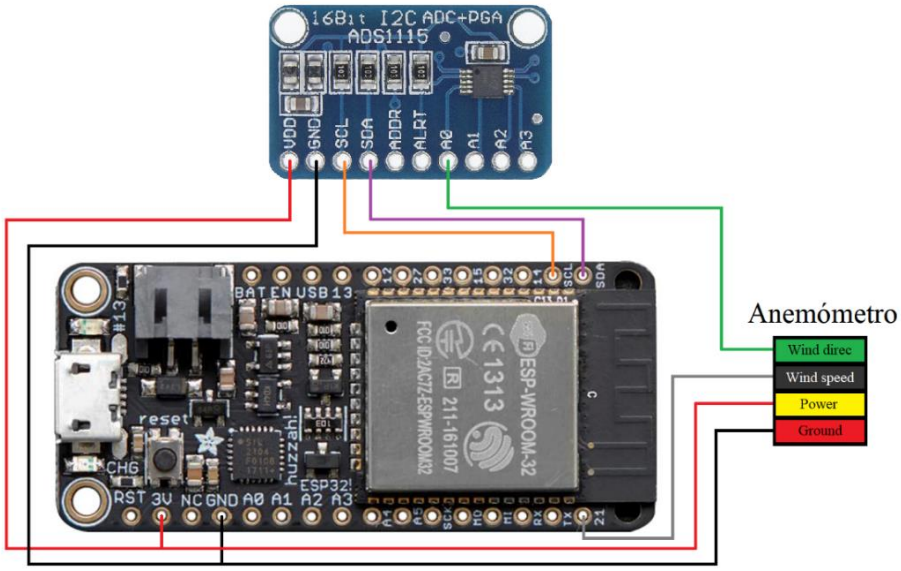


**Figura 24. Conexionado de los elementos para la medición de la velocidad y dirección del viento**

### 4.2.3 Precipitación

Para medir la precipitación líquida se utiliza el pluviómetro 6466M fabricado también por Davis Instrument (**Figura 25**).

**Figura 25. Pluviómetro 6466M de Davis Instrument**

Este pluviómetro mide la cantidad de lluvia mediante un balancín de vaciado automático. Cuando el peso del agua hace bascular el balancín se produce un pulso para poder cuantificar la precipitación. Este sistema se muestra en la **Figura 26**.

**Figura 26. Balancín y sistema de vaciado automático del pluviómetro 6466M de Davis Instrument**

Este instrumento cuenta con una conexión de 3 cables: 2 para la alimentación y uno para la salida del interruptor Reed, encargado de enviar los pulsos (**Figura 27**).

En cuanto a las especificaciones del sensor, los pulsos son enviados cuando es recolectado un área de 0,2 mm. La incertidumbre del sensor es de ±3% o de 0,2 mm (la mayor) para lluvias hasta 250 mm/h. Este pluviómetro está especialmente diseñado para cumplir con los requisitos establecidos por la OMM [21].

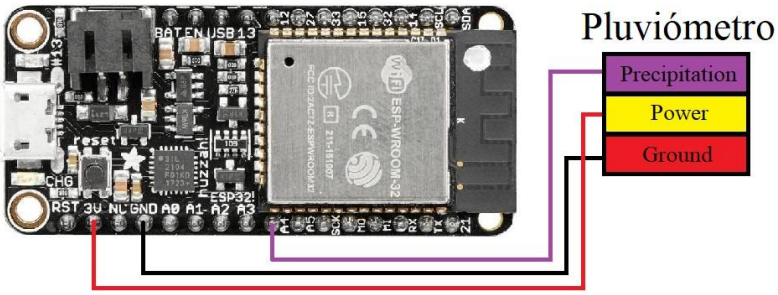


Figura 27. Conexionado entre el pluviómetro y el ESP32

### 4.2.4 Irradiancia

La irradiancia solar es la energía emitida por el sol en forma de radiación electromagnética por unidad de superficie.

Como ya se ha adelantado, para medir esta característica climatológica comúnmente se utiliza un piranómetro térmico (**Figura 9**), aunque también se pueden utilizar otros instrumentos para estimar la radiación solar.

El funcionamiento de un piranómetro es sencillo. Cuenta con una cúpula transparente por la cual pasa la radiación solar. En su interior existe un detector sensible formado por termopares estratégicamente colocados, junto a unas barras de cobre y una placa de latón, que provoca una tensión eléctrica a la salida proporcional a la irradiancia.

Para la estación meteorológica utilizaremos un pequeño panel fotovoltaico junto a un sensor de corriente, creando un sensor de irradiación solar de celda de referencia. Este sensor es ideal para aplicaciones fotovoltaicas ya que es una miniatura de los paneles fotovoltaicos. Su funcionamiento consiste en medir la corriente generada por la cantidad y distribución espectral de los fotones incidentes, resultando ser esta corriente proporcional a la irradiancia [22].

La célula fotovoltaica que utilizaremos para captar la radiación solar es la mostrada en la **Figura 28**, el modelo de Voltaic P121.


Figura 28. Célula fotovoltaica Voltaic P121

Las especificaciones técnicas dadas por el fabricante en condiciones estándar (STC)[4] de este panel son:

- Potencia máxima: 0,33 W
- Voltaje a máxima potencia: 2,43 V
- Corriente a máxima potencia: 0,13 A
- Tensión en circuito abierto: 2,84 V
- Corriente de cortocircuito ($I_{STC}$): 0,15 A
- Tolerancia ± 10%

La corriente de cortocircuito Isc es directamente proporcional a la irradiancia, siguiendo la **ecuación ( 4 )**:

$$Isc = E \cdot \frac{Isc_{STC}}{E_{STC}} \qquad (4)$$

Donde E es la irradiancia en W $m^{-2}$.

Además, la temperatura de la célula afecta directamente a la corriente de cortocircuito y en consecuencia a la irradiancia. El coeficiente de temperatura (α) para la corriente Isc en células de silicio se encuentra en torno a un 0,05% por grado centígrado. La expresión final de la irradiancia en función de la corriente de cortocircuito y la temperatura viene representada en la **ecuación ( 5 )**:

$$E = \frac{Isc \cdot 1000}{Isc_{STC} \cdot (1 + \alpha \cdot (Tc - T_{STC}))} \qquad (5)$$

Donde $E$ es la irradiancia en W $m^{-2}$ y $Tc$ la temperatura de la célula en ºC [23].

El sensor de corriente que utilizaremos para medir la corriente de cortocircuito entre los terminales $V^-$ y $V^+$ del panel es el ACS723LLCTR-05AB-T[2] incorporado junto un amplificador de ganancia ajustable (desde 2,2 a 22 V/V) en una placa desarrollada por SparkFun. Esta se muestra en la **0**.

[4] Los parámetros característicos de los paneles fotovoltaicos vienen medidos por los fabricantes en unas condiciones estándar ya establecidas. Estas condiciones estándar (STC) son: irradiancia $E_{STC}$ 1000 W/$m^2$, temperatura de la célula $T_{STC}$ 25 ºC y una distribución espectral de AM 1,5G [31].

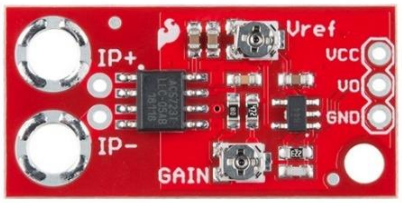


**Figura 29. Sensor de baja corriente ACS723 de SparkFun**

Este sensor de efecto Hall está diseñado para medir bajas corrientes. Su rango de medición es de -5 a 5 A, capaz de detectar corrientes bajas del rango de 10 mA, con una sensibilidad de 400 mV/A con un error de ±2%.

Este chip cuenta con: 2 pines de alimentación, VCC y GND; un pin Vo de salida analógica; los dos terminales $IP^+$ e $IP^-$ para medir la corriente de cortocircuito Isc; y dos potenciómetros, Vref para ajustar la tensión de salida cuando la corriente es nula y Gain para ajustar la sensibilidad del dispositivo. El ajuste de estos potenciómetros se detalla en el Capítulo 5 de este documento [24, 25].

Al ser una salida analógica debemos utilizar un canal del conversor analógico-digital ADS1115 para registrar estas mediciones. Este conversor ya se ha estudiado en profundidad en el apartado 4.2.2 de este capítulo.

Se conoce que la temperatura de la célula juega un papel fundamental para el cálculo de la irradiancia. Es por esto por lo que se ha incorporado en la estación el MAX31865. Este módulo es un amplificador de temperatura y convertidor analógico-digital que junto a un RTD pt100 permite registrar la temperatura del panel. Estos se muestran en la **Figura 30**.

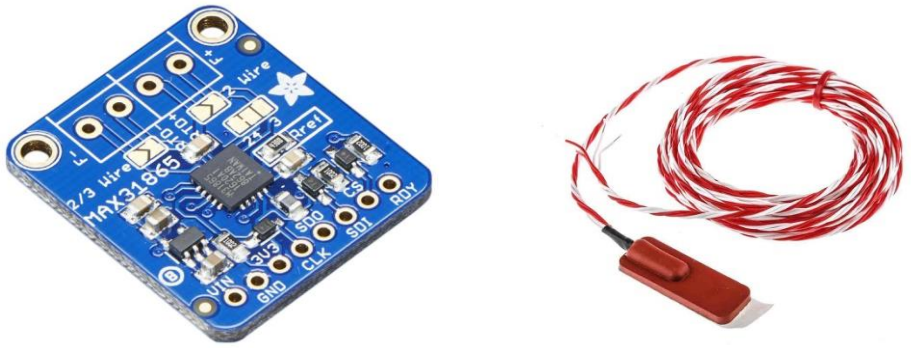


**Figura 30. MAX31865 de Adafruit (izquierda) y un RTD pt100 (derecha)**

Su tensión de alimentación es de 3,3 V. La resolución del ADC que lleva incorporado es de 15 bits, siendo la resolución de la temperatura de 0,03125 ºC con una precisión del 0,05% del fondo de escala. El protocolo de comunicación utilizado en este caso es SPI, a diferencia del utilizado con el resto de los sensores [26].

La conexión del pt100 utilizada es de 4 hilos debido a que la medida de su resistencia es mucho más precisa y eliminamos los posibles ruidos de offset e interferencias. A diferencia de las conexiones de 2 y 3 hilos, con 4 hilos no es necesario realizar ningún cambio en el MAX31865.

La incertidumbre que obtenemos en la medición viene marcada por los instrumentos utilizados a la hora de registrar la corriente de cortocircuito ya que las aportaciones del módulo MAX31865 y del RTD son despreciables en comparación con las incertidumbres de los datos del fabricante del P121 y del sensor de corriente. Como estas aportaciones son independientes se utilizará el método RSS para su cálculo, siendo la incertidumbre total calculada como en la **ecuación ( 6 )**:

$$Incertidumbre\ irradiancia = \sqrt{0{,}1^2 + 0{,}02^2} = 10{,}1\% \qquad (6)$$

El conexionado de todos los elementos utilizados para medir la irradiancia se muestra en la **Figura 31**.

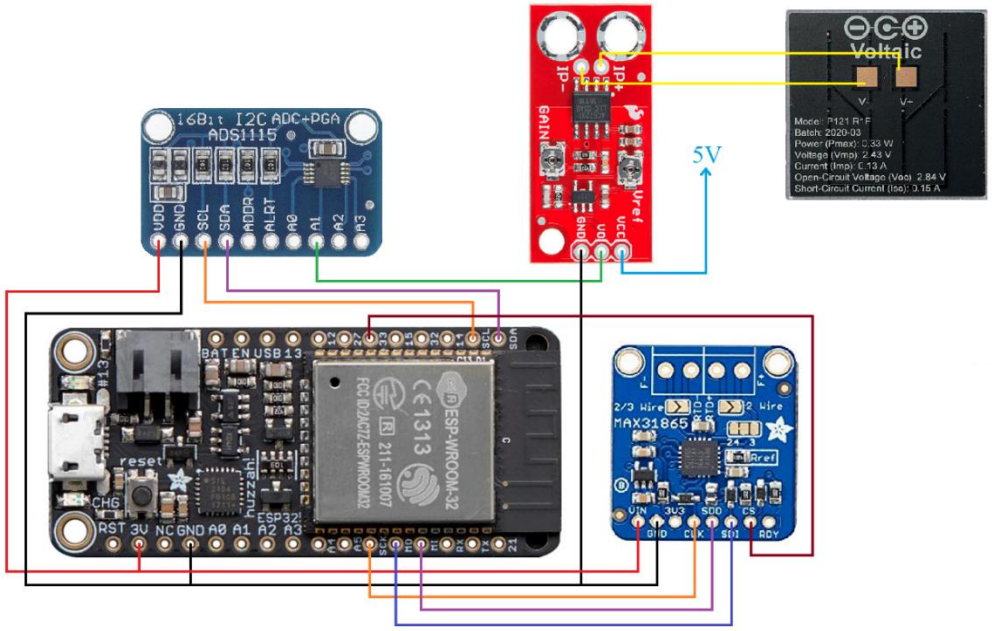


**Figura 31. Conexionado entre los diferentes elementos utilizados para la medición de la irradiancia**

## 4.3 Trabajo experimental

En la práctica el orden de los instrumentos de medida seguido a la hora de conectar y comprobar su funcionamiento fue:

1. BME680
2. MAX31865 y pt100
3. ADS1115
4. Anemómetro y veleta 6410
5. ACS723 y Voltaic P121
6. Pluviómetro 6466M

Las conexiones realizadas se pueden observar en la **Figura 32**.

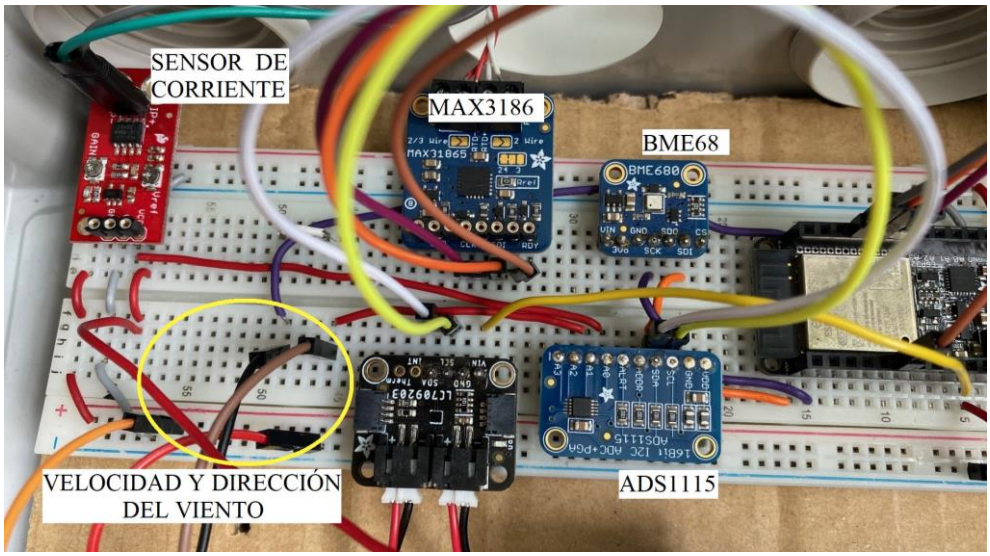


**Figura 32. Conexionado de los diferentes instrumentos de la Estación Meteo.**

El único sensor que requiere de unas comprobaciones previas para su correcto funcionamiento es el sensor de corriente ACS723, teniendo que ajustar los potenciómetros Vref y Gain.

Para el ajuste de Vref, con el sensor de corriente alimentado a 5 V y con corriente nula (con la célula fotovoltaica desconectada), ajustamos el potenciómetro hasta obtener 2,5 V a la salida (2,48 V en la práctica).

Para el ajuste de la ganancia debemos inyectarle al ACS723 una corriente ya conocida y ajustar el potenciómetro hasta observar la salida con la sensibilidad deseada. Esta ganancia debemos escogerla teniendo en cuenta que, a mayor ganancia, el ruido a la salida aumentará, pudiendo afectar en la precisión de la medida. Otro factor a tener en cuenta para este ajuste es que la salida no nos sature en ningún momento, siendo su valor máximo 5 V.

Se montó un circuito aparte (**Figura 33**) con resistencias en paralelo para modificar la corriente que atravesará el sensor, midiendo con un multímetro esta corriente para conocerla con exactitud a la vez que observamos como varía la salida con el osciloscopio al quitar o añadir resistencias

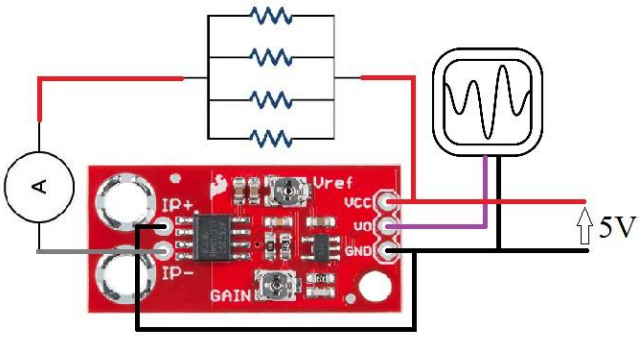


**Figura 33. Esquema del circuito utilizado para calibrar el sensor ACS723**

En la **Figura 34** se muestra el montaje real del circuito representado en la **Figura 33** y en la **Tabla 5** los resultados obtenidos.

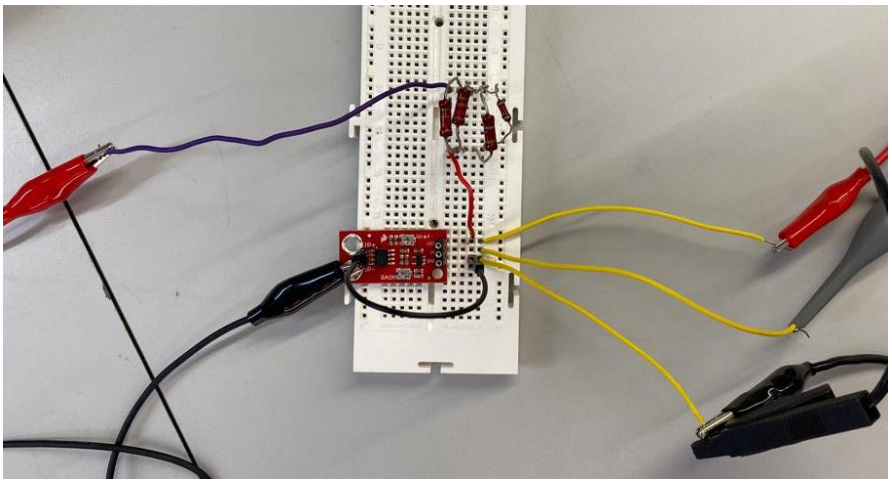

**Figura 34. Montaje utilizado para la calibración del sensor ACS723**

| Calibración irradiancia | | |
|---|---|---|
| Corriente [A] | Salida [V] | Sensibilidad [mV/A] |
| 0,161 | 2,65 | 1055,9 |
| 0,108 | 2,6 | 1111,1 |
| 0,068 | 2,55 | 1029,4 |
| 0,029 | 2,51 | 1034,5 |
| Sensibilidad media | | 1057,73 |

**Tabla 5. Sensibilidad obtenida tras medir la salida del sensor ACS723 para cuatro corrientes diferentes**

Debemos tener en cuenta que tras el ajuste de la ganancia puede haber variado ligeramente la configuración de Vref, lo que convendría reajustarlo [25].

# Capítulo 5

# 5 Software de la estación

En este capítulo se describe todo lo relacionado con el software de la estación con la intención de explicar cómo se han registrado todas las variables meteorológicas de los diferentes instrumentos y sensores descritos en el Capítulo 4; la comunicación REST API utilizada para registrar las mediciones en la base de datos del grupo GEISER; el diseño de un sistema para evitar la pérdida de información de las variables en caso de perder conectividad Wi-Fi; y la implementación de la funcionalidad de actualización remota del firmware.

## 5.1 Descripción general

El software de la estación se desarrolla en el entorno VisualStudio Code, haciendo uso de la extensión PlatformIO.

PlatformIO es una herramienta de desarrollo que admite múltiples plataformas y lenguajes de programación, incluyendo C/C++ para el ESP32. El *framework* utilizado en el proyecto es el de Arduino.

El ESP32 es un microcontrolador de alto rendimiento compatible con sistemas operativos en tiempo real. Estos sistemas operativos permiten planificar tareas garantizando su tiempo de ejecución máximo, la frecuencia y prioridad de estas, el manejo de interrupciones y la sincronización de tareas mediante semáforos, entre otras funcionalidades. El RTOS utilizado para este proyecto es FreeRTOS, el cual es distribuido de forma gratuita por el MIT e incluye librerías en constante crecimiento útiles para nuestro caso.

El aprovechamiento de los dos núcleos del ESP32 y el uso de FreeRTOS ha requerido seguir una estructura clara y ordenada, respetando siempre el tiempo de ejecución que necesita cada tarea para el correcto funcionamiento del software. La estructura general del programa de la Estación Meteo consta de dos tareas asignadas a cada uno de los núcleos.

El orden de inicialización de los diferentes elementos del programa ha sido elegido en función del posible tiempo en espera, siendo las inicializaciones inmediatas las primeras y dejando para el final todas aquellas que dependan del Wi-Fi. A continuación, se definen los pines 21 y 36 como entradas digitales que servirán para obtener la velocidad del viento y la precipitación, respectivamente; se crean las dos tareas principales, TaskLectura y TaskNube; y se crean las rutas para la comunicación con la base de datos.

La tarea asignada al núcleo 0, TaskLectura, del ESP32 se encargará de recoger todas las variables meteorológicas medidas con un periodo de 3 segundos, mientras el otro núcleo se encargará de todo lo relacionado con la conexión Wi-Fi, TaskNube. Se escoge el Core 1 para realizar esta tarea debido a que es el núcleo que utiliza por defecto el ESP32, facilitándonos comprobar con la función "loop" de manera sencilla y continuada si seguimos conectados a la red. En la **Figura 35** se muestra un esquema del funcionamiento del programa de la estación meteorológica.

La tarea "loop" es utiliza también para ejecutar las funciones "ArduinoOTA.handle" y "server.handleClient", las cuales es necesario que se ejecuten constantemente para mantener la actualización remota del firmware y para la comunicación con la base de datos.

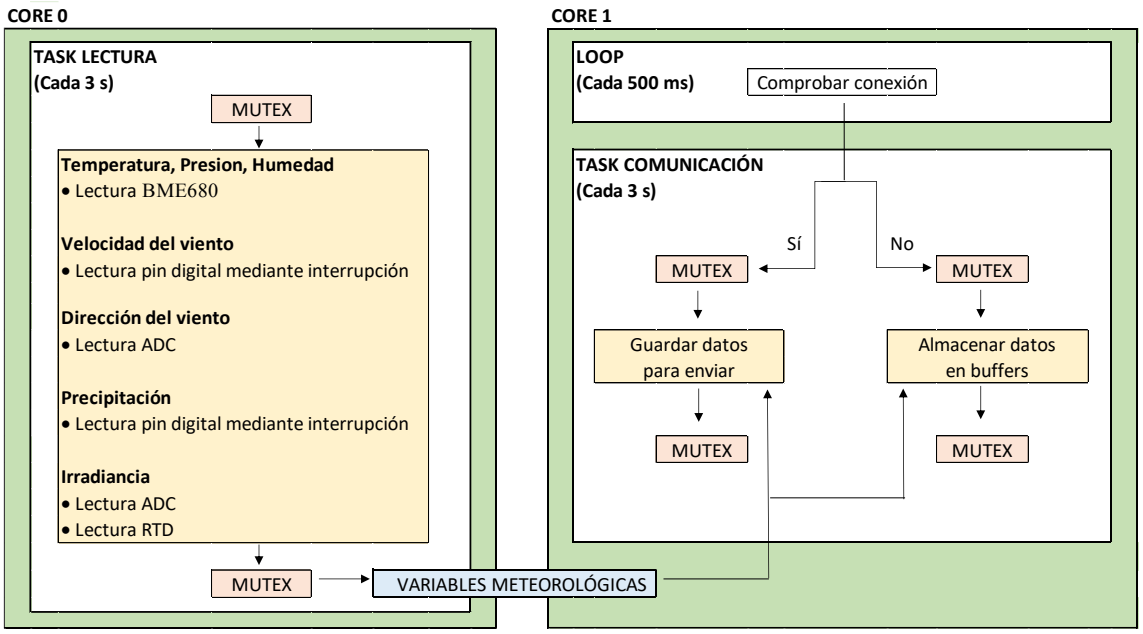


**Figura 35. Esquema del funcionamiento del programa de la Estación Meteo**

## 5.2 Adquisición de datos

El núcleo 0 es el encargado de realizar la lectura de los sensores y de realizar los cálculos necesarios para obtener la medida final en las unidades que nos interesan. Las variables meteorológicas observadas se actualizarán cada 3 segundos. Durante la adquisición de datos protegeremos las variables mediante un *mutex* para eliminar la posibilidad de un acceso simultáneo a estas por ambas tareas.

A continuación, se explica el procedimiento que se utiliza para obtener todas las variables meteorológicas:

Temperatura ambiente, presión atmosférica y humedad relativa

El sensor encargado de realizar las medidas de estas variables es el BME680, ya estudiado en el Capítulo 3. Su lectura es directa usando la librería < BME680.h >, dándonos la temperatura ambiente en ºC, la humedad relativa en % y la presión atmosférica en Pa. Esta última la dividiremos entre 100 para obtener la presión en hPa.

Velocidad y dirección del viento

Para la velocidad del viento, gracias FreeRTOS, generaremos una interrupción asociada a flancos de bajada recibidos en el pin 21. Con esta interrupción mediremos los pulsos que produce el anemómetro con cada giro de las cazoletas. Para obtener el valor final de la velocidad del viento en km/h se aplicará la **ecuación ( 7 )**:

$$WindSpeed\ [\frac{km}{h}] = \frac{Rotaciones \cdot 2\pi \cdot r \cdot 3600 \cdot 1000}{t \cdot 10^5} \qquad (7)$$

Siendo $r$ el radio del anemómetro, en cm; $t$ el tiempo durante el que se han contado los pulsos del anemómetro, en ms; y Rotaciones el número de pulsos contados por la interrupción durante el tiempo $t$.

La dirección del viento medida por la veleta es una señal analógica convertida por el canal 1 del ADS1115 en una señal digital. Los valores de esta señal digital se encuentran en los rangos de 0 a 20595, el cual mapearemos para trabajar con valores de 0º a 360º.

Después de calcular el offset y obtener la dirección final en grados, para obtener los puntos cardinales clasificamos los grados según la **Tabla 6**:

| Grados | Puntos del cardinales |
|---|---|
| 0º - 66º | Noreste "NE" |
| 67º - 111º | Este "E" |
| 112º - 156º | Sureste "SE" |
| 157º - 211º | Sur "S" |
| 212º - 246º | Suroeste "SW" |
| 247º - 291º | Oeste "W" |
| 292º - 336º | Noroeste "NW" |
| 337º - 360º | Norte "N" |

**Tabla 6. Puntos cardinales**

Precipitación

El método utilizado para leer la precipitación con el pluviómetro es similar al usado para medir la velocidad del viento. Se genera una interrupción por flancos de bajada en el pin digital 36 para contar pulsos que indicarán que la cazoleta del pluviómetro ha sido llenada, lo que supone 0,2 mm de agua recolectada. Al cabo de 3 segundos, los pulsos contados multiplicados por 0,2 mm nos indicarán la cantidad de agua precipitada en los últimos 3 segundos.

Irradiancia

Para la irradiancia debemos leer el canal 0 del ADS1115 que corresponde con la salida del sensor de corriente obteniendo así la corriente de cortocircuito del panel fotovoltaico, la cual es proporcional a la irradiancia. Para obtener la corriente de cortocircuito a partir de la medida del sensor de corriente debemos tener en cuenta los potenciómetros, Vref para ajustar el offset y Gain para la sensibilidad, ya ajustados como se ha descrito en el apartado 4.3.

Una vez ajustados correctamente los potenciómetros, calculamos la Isc aplicando la **ecuación ( 8 )**:

$$Isc = \frac{A0 - Vref}{Gain} \qquad (8)$$

Siendo *A0* la salida del sensor de corriente digitalizada, en mV; y *Gain* y *Vref* el ajuste de los potenciómetros de ganancia y offset.

Una vez obtenida la corriente de cortocircuito, para poder aplicar la **ecuación ( 5 )** para calcular la irradiancia, debemos conocer la temperatura del panel gracias al MAX31865 siendo su lectura directa si conocemos el valor de la resistencia nominal (100 Ω en nuestro caso al ser un pt100) y la resistencia de referencia (430 Ω en nuestro caso).

Ya conocidos la Isc y la temperatura del panel solo faltaría aplicar la **ecuación ( 5 )** (ya vista en el capítulo 4) para obtener la irradiancia.

## 5.3 Comunicación con la base de datos

Para la comunicación con la base de datos se ha implementado una API REST.

Las API REST, también conocidas como API RESTful, son una interfaz que utiliza los principios de REST (Representational State Transfer) para permitir la comunicación entre programas a través de internet.

El principio más importante de estas APIs es el uso de métodos HTTP para crear, obtener y manipular recursos de los servidores. La información entregada suele ser en formato JSON (JavaScript Object Notation), aunque también puede darse en formatos HTML, XLT, Python, PHP o texto sin formato. Es una arquitectura conocida como cliente-servidor, en la que el servidor almacena la información y la pone a disposición del cliente para que este, mediante peticiones, pueda acceder a ella. Al ser estas interacciones muy básicas y sencillas, este estilo de arquitectura REST es ideal para implementarlo en aplicaciones que no requieran una gran complejidad en la comunicación, como es nuestro caso [27].

Para que una API se considere de REST debe cumplir los siguientes requisitos [28]:

- **Cliente-servidor:** El cliente y el servidor deben ser independientes entre sí.
- **Sin estado:** La API de REST carece de estado. Deben poder realizarse peticiones independientes entre sí.

- **Caché:** Las respuestas deben ser capaces de ser almacenadas por el cliente para reducir la necesidad de reenviar la misma solicitud al servidor.
- **Interfaz uniforme:** Debe contar con una interfaz uniforme para una transferencia de recursos estandarizada. Los recursos deben ser identificado de manera única y deben contar con mensajes autodescriptivos para que el cliente sepa procesarlos.
- **Sistema de capas:** La arquitectura puede estar compuesta por varias capas jerárquicas. Cada capa realiza una función específica y no necesita conocer los detalles de las demás capas.

En nuestro proyecto, existen dos posibles documentos a transferir a la base de datos, ambos en formato JSON.

Por un lado, cuando contamos con conexión Wi-Fi, los valores enviados son de tipo float y el documento sigue la siguiente estructura:

```
obj["type"] = tag;
obj["value"] = value;
obj["unit"] = unit;
```

En este caso el cliente contará con 4 peticiones GET para acceder a la información:

- "/env", getEnv: Para obtener la temperatura ambiente, presión y humedad relativa.
- "/wnd", getWind: Para obtener la velocidad y dirección del viento.
- "/cell", getCell: Para obtener la irradiancia y la temperatura de la célula fotovoltaica.
- "/bat", getBattery: Para obtener la tensión y carga de la batería.
- "/pluv", getPluvi: Para obtener la precipitación.

Por otro lado, después de un periodo de desconexión, la base de datos deberá obtener todas las mediciones transcurridas en ese tiempo siendo enviada una cadena de valores del tipo char. La estructura del documento contará con la fecha de inicio de desconexión "Date" y el tiempo que se ha permanecido desconectado "Time":

```
objDISconnected["Date"] = date;
objDISconnected["Time"] = t;
objDISconnected["type"] = tag;
objDISconnected["value"] = value;
objDISconnected["unit"] = unit;
```

Para no saturar al cliente con demasiada información se ha decidido que en este caso el cliente deberá solicitar las medidas de cada variable meteorológica de forma independiente:

- "/tempdisconnected", getTempdisconnected: Para el buffer de temperaturas ambiente.
- "/pressdisconnected", getPressdisconnected: Para el buffer de presiones.
- "/humdisconnected", getHumdisconnected: Para el buffer de la humedad relativa.
- "/pcntdisconnected", getPcntdisconnected: Para el buffer de la carga de la batería.
- "/vdisconnected", getVdisconnected: Para el buffer de la tensión de la batería.
- "/irrdisconnected", getIrrdisconnected: Para el buffer de la irradiancia.
- "/rtddisconnected", getRTDdisconnected: Para el buffer de la temperatura del panel.
- "/direcdisconnected", getDirecdisconnected: Para el buffer de la dirección del viento.

- "/veldisconnected", getVeldisconnected: Para el buffer de la velocidad del viento.
- "/pluvidisconnected", getPluvidisconnected: Para el buffer de la precipitación.

## 5.4 Caso offline

Una de las mejoras incorporadas en la Estación Meteo es la incorporación de un sistema que evite la pérdida de información de las diferentes variables meteorológicas que estamos midiendo cuando perdemos la conexión Wi-Fi, ya que en este caso, la comunicación con la base de datos fallaría. De esta tarea se ocupará el núcleo 1.

Con la tarea "loop" del programa se irá comprobando periódicamente si estamos conectados a la red Wi-Fi. De forma visual, se ha utilizado el LED rojo incorporado ya en el ESP32 para indicarnos si estamos conectados a Wi-Fi cuando se ilumina. En este caso, el programa no almacenará las mediciones ya que se supondrá que la comunicación con la base de datos está siendo exitosa.

En el caso de perder la conexión, se indicará de manera externa apagándose el LED rojo del ESP32. Para este otro caso se han creado diferentes buffers donde se almacenarán cada una de las variables para ser enviadas en cuanto se recupere la conexión. Es importante para el estudio meteorológico conocer la hora precisa en la que se realizaron las mediciones por lo que se guardan la fecha y la hora del momento en el que se perdió la conexión.

Para configurar el RTC interno del ESP32 y así conocer con precisión la fecha y hora se ha utilizado la librería < ESP32Time.h >. Con esta librería podemos definir la hora y fecha inicial y a partir de esta ir trabajando, actualizándose automáticamente. Para mayor precisión, en el proyecto se ha optado por fijar la fecha y hora inicial sincronizando el reloj interno con la hora del servidor "pool.ntp.org"[5] [29].

Debido a la cantidad de información que se quiere almacenar cuando ocurre un periodo de desconexión, ha sido necesario limitar los buffers a una capacidad de 1220 bytes cada uno para el correcto funcionamiento del programa, pudiendo almacenarse hasta 101 mediciones. Esta cantidad de mediciones indican que podemos permanecer sin conexión hasta 5 minutos para no perder información.

Los buffers serán vaciados una vez se hayan producido todas las solicitudes de envío de los datos medidos durante el periodo de desconexión.

## 5.5 Actualización remota del firmware

Otro de los objetivos de este proyecto es implementar la funcionalidad de actualización remota del firmware de la estación, conocidas como actualizaciones OTA (Over The Air).

Las actualizaciones OTA tienen la capacidad de actualizar y enviar datos a dispositivos electrónicos inalámbricamente, vía Wi-Fi (u otros medios como Bluetooth) sin necesidad de realizar una conexión física por cable. Un ejemplo de actualizaciones OTA son aquellas que

[5] El programa está diseñado para poder trabajar sin Wi-Fi. Sin embargo, para el arranque inicial de este es necesario estar conectados por lo que siempre podremos sincronizarnos con la fecha y hora de la red al inicio.

se realizan de vez en cuando en nuestros dispositivos móviles cuando para corregir errores o introducir pequeñas mejoras.

Para cumplir con este objetivo se ha utilizado la librería Arduino OTA que dispone de todas las funciones necesarias para este proceso. Para conseguir su correcto funcionamiento debemos cargar el programa una primera vez, con la biblioteca Arduino OTA y la función "OTASetup", mediante el puerto serie. Una vez cargado el programa debemos identificar la IP de la estación en la red y, conocida esta, actualizamos el upload_protocol y el upload_port de la siguiente forma.

```
upload_protocol = espota
upload_port = XXX.XX.XX.X ;; Con la IP correspondiente
```

Una vez añadidas estas dos líneas de código a nuestro platformio.ini y con conexión Wi-Fi, podremos realizar la siguiente actualización del firmware inalámbricamente, compilando y cargando el programa de la misma forma que cuando se trata de una actualización vía USB.

# Capítulo 6

# 6 Resultados

Los capítulos anteriores se han destinado al estudio de los diferentes elementos que componen la estación y como se han ido implementando.

En este capítulo se mostrarán todas las pruebas prácticas que se le han realizado a la estación, en su conjunto (**Figura 36**), para validar su correcto funcionamiento.

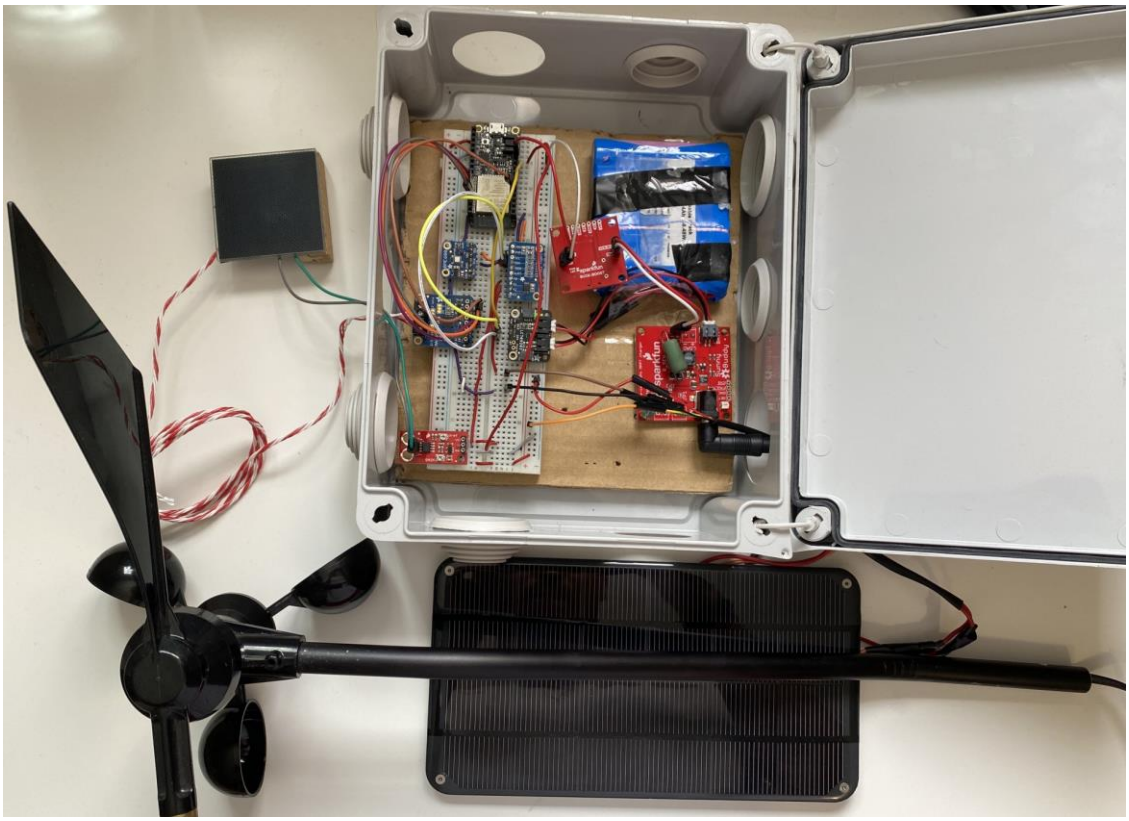

**Figura 36. Montaje completo de la Estación Meteo**

## 6.1 Hardware de alimentación

El sistema de alimentación autónomo del que dispone la estación ha sido probado para comprobar los tiempos de carga y de descarga reales de la batería.

Con una carga inicial del 70,40 % se ha dejado el sistema de alimentación, sin la carga conectada, durante 2 horas, habiendo aumentado la batería al 71,90 % (**Figura 37**). Con esta información podemos deducir que el tiempo de carga real de la batería del 0 al 100% es menor a las 200 horas, ya que durante este periodo de prueba el panel fotovoltaico no trabajó a pleno sol.

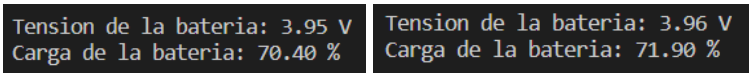


**Figura 37. Carga de la batería del día 12/09 a las 17:40 (izquierda) a las 19:40 (derecha)**

Para comprobar el consumo total de la estación se ha dejado funcionando la Estación Meteo por completo durante 2 horas con el panel solar desconectado, de tal forma que la batería no se cargue durante esta prueba. Durante este periodo la batería pasó de una carga inicial del 40,5 % a 34,5 %. Podemos deducir que el tiempo máximo que puede permanecer la estación funcionando sin energía solar es de 34 horas.

## 6.2 Adquisición de datos

Para comprobar el correcto funcionamiento del BME680 y del MAX31865 se ejecutó el programa (ver **Anexo 1**), obteniendo unos valores iniciales que, con la intención de aumentar la temperatura y comprobar si las mediciones eran iguales, tanto del BME680 como del pt100, se trataron de calentar mediante la aplicación de calor para observar si ambos variaban y medían lo mismo. Este cambio de temperatura se vio reflejado en los valores registrados (**Figura 38**) dando a entender un correcto funcionamiento de ambos sensores, ya que se obtuvo una diferencia en las mediciones dentro de los rangos de incertidumbre, de 0,34 ºC.

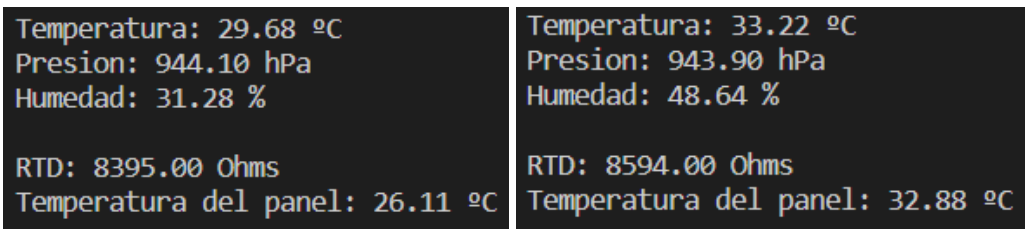


**Figura 38. Medidas iniciales (izquierda) y medidas después de aplicar calor (derecha)**

Respecto a la veleta y dirección del viento, la primera comprobación que se realizó fue variando manualmente la veleta y haciendo girar las cazoletas del anemómetro y observando que los valores registrados variaban siguiendo una lógica (**Figura 39**). A continuación, se calculó el offset de la veleta comparando el valor registrado de la dirección del viento con el proporcionado por la brújula del teléfono móvil. Este offset resultó ser de 75º aunque este ajuste debe comprobarse siempre que se cambie el emplazamiento de la veleta. Para la

velocidad del viento se comprobó gracias a una grabación a cámara lenta que las rotaciones contadas por el programa se correspondían con las que realizaba el anemómetro en la realidad al aplicarle una ráfaga de viento.

```
Dirección del viento: 18.00º N

Velocidad del viento: 0.00 km/h
Rotaciones: 0
```

```
Dirección del viento: 285.00º W

Velocidad del viento: 11.62 km/h
Rotaciones: 29
```

**Figura 39. Velocidad y dirección del viento iniciales (izquierda) y despúes de aplicarles variaciones (derecha)**

Para comprobar que la irradiancia medida por la estación era correcta se hicieron unas pruebas en la azotea de la universidad comparando el valor medido con un sensor de celda de referencia de la universidad (**Figura 40**).

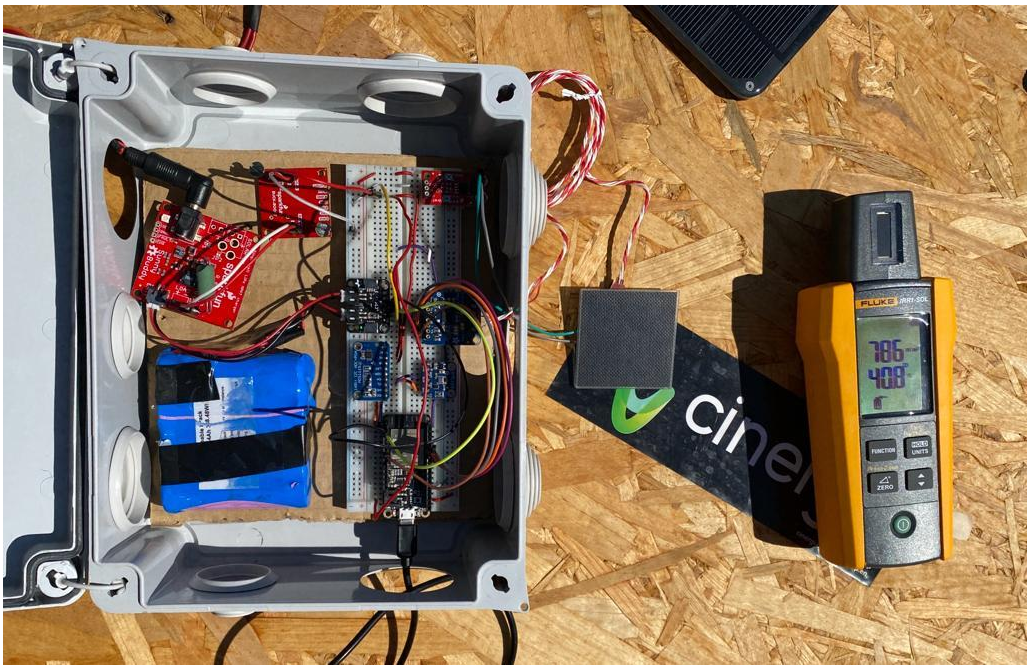


**Figura 40. Calibración del sensor de irradiancia de la Estación Meteo**

Para obtener la correcta medida de irradiancia resultó tener el sensor de corriente un offset de 2,55 V y una sensibilidad de 6 000 mV/A.

El Pluviómetro 6466M de Davis Instrument envía pulsos por su salida digital cuando son recolectados 2 mm de agua. La forma más sencilla de verificar el correcto funcionamiento de este instrumento es comprobar que nuestro programa registra un pulso cuando el balancín del pluviómetro cede al recolectar dicha cantidad de agua. Esta variable meteorológica no ha sido comprobada de forma experimental puesto que durante este proceso no se contaba con el instrumento.

Como la mayoría de los objetivos de este proyecto se encuentran relacionados con la conexión Wi-Fi, es primordial comprobar que nuestro ESP32 se conecta y desconecta de manera correcta y somos capaces de reflejarlo en el programa. Por este motivo se incorporó el detalle de iluminar el LED rojo del ESP32 cuando nos encontramos conectados y de

apagarse en caso de desconexión. Se comprueba en repetidas ocasiones que esto funciona correctamente (**Figura 41**).

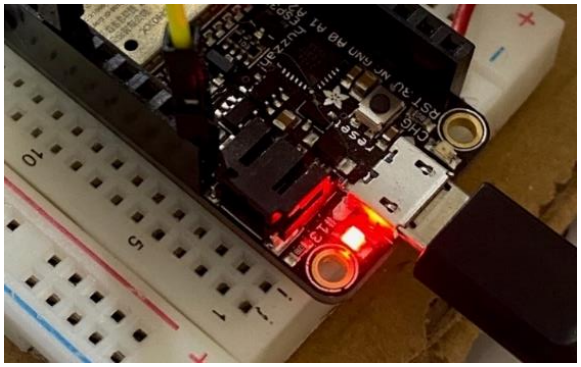

**Figura 41. LED rojo del ESP32 encendido al estar conectados a Wi-Fi**

Para comprobar la correcta sincronización de las dos tareas se decide mostrar por el monitor serial "-------" cada vez que se actualizan las mediciones de los diferentes sensores (TaskLectura) y mostrar sus valores (TaskNube). Como se muestra en la **Figura 42**, en ningún momento una acción interrumpe a otra.

```
BME680 ----------------
WIND DIRECC ----------
MAX31865 -------------
IRRADIACIA------------
LC709203F ------------
PLUVIOMETRO ----------
WINDSPEED ------------
3000.00
Temperatura: 28.08 ºC
Presion: 944.48 hPa
Humedad: 34.19 %

Dirección del viento: 338.00º N

Irradiancia: 14.44 V
Irradiancia: 0.04 A
Irradiancia: -99.51 W/m²

RTD: 8347.00 Ohms
Temperatura del panel: 24.52 ºC

Tension de la bateria: 3.62 V
Carga de la bateria: 4.40 %

Precipitacion en los ultimos 3 segundos: 0.00 mm

Velocidad del viento: 0.00 km/h
Rotaciones: 0
```

**Figura 42. Lecturas y valores de los diferentes sensores mostrados en monitor serial**[6]

[6] En el momento de la captura tanto el pluviómetro como la célula Voltaic P121 se encontraban desconectadas, por lo que sus mediciones mostradas en la **0** no son reales.

## 6.3 Comunicación con la base de datos

Para comprobar la comunicación de la API de REST se ha utilizado la herramienta Postman (logo mostrado en la **Figura 43**). Esta aplicación nos permite realizar peticiones de una manera sencilla para testear APIs de tipo REST [30].


**Figura 43. Logo de Postman**

En la **Figura 44** se puede observar cómo se han solicitado los datos de temperatura ambiente, humedad relativa y presión atmosférica mediante la petición GET "/env" y se han obtenido correctamente siguiendo la estructura descrita en el apartado 5.3 para este caso.

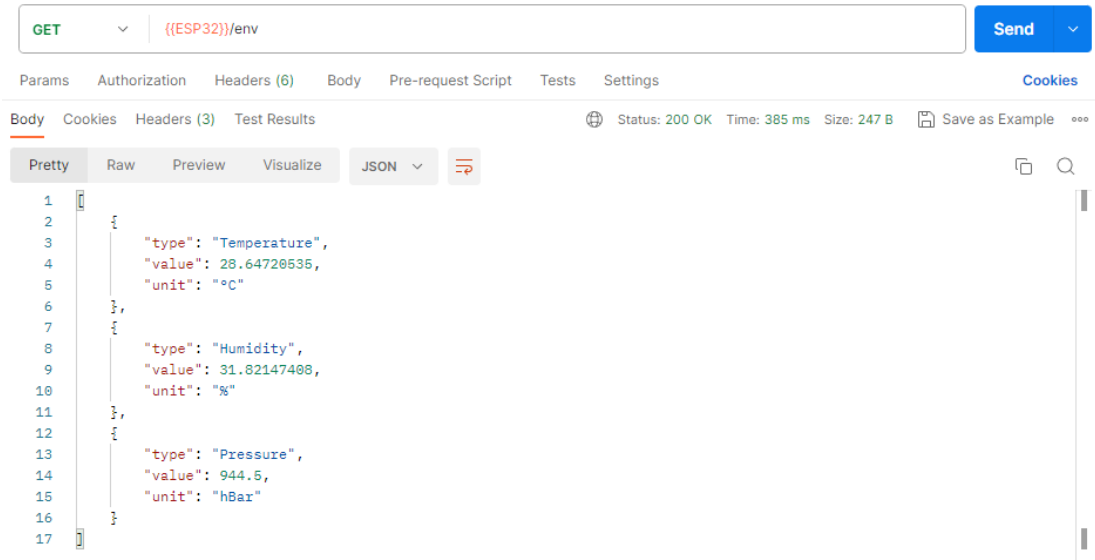


**Figura 44. Solicitud de datos mediante Postman de temperatura, humedad relativa y presión atmosférica en Postman**

## 6.4 Periodos de desconexión

Para comprobar que las variables se almacenan correctamente en sus respectivos buffers basta con desconectar de manera intencionada la estación de su conexión Wi-Fi. En la **Figura 45** se muestra como aparen estos buffers en el monitor serial.

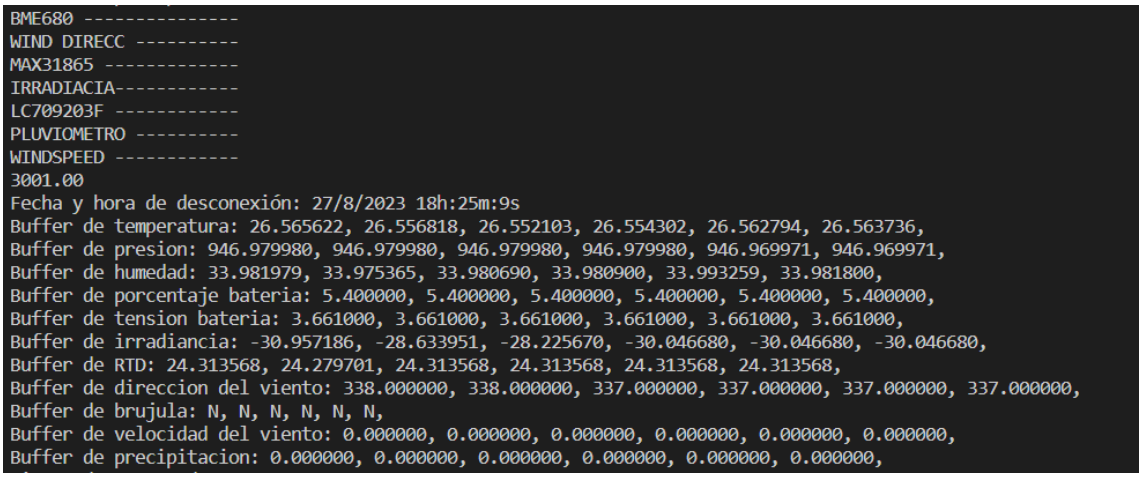


Figura 45. Almacenaje de las diferentes variables meteorológicas cuando perdemos la conexión

Podemos observar (Figura 46) como al reconectarnos podemos solicitar el envío de todas aquellas medidas que se han realizado mientras la estación se encontraba sin conexión

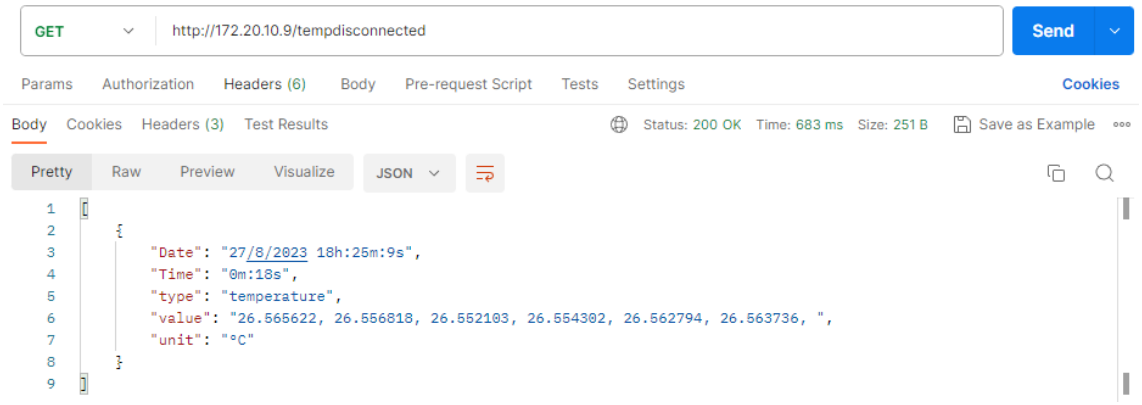


Figura 46. Solicitud de datos mediante Postman de temperatura durante el periodo de desconexión en Postman

Al repetir este procedimiento para todas las variables meteorológicas se puede observar que los buffers se actualizan correctamente.

## 6.5 Actualización remota del firmware

Para comprobar el último objetivo, la actualización remota del firmware, de forma experimental se ha visto la necesidad de desactivar el firewall del ordenador desde donde se va a modificar el firmware para su correcto funcionamiento[7]. Después de cargar el programa en el ESP32 mediante el puerto USB, se realizan los cambios en el platformio.ini para activar

[7] El firewall de Windows es una herramienta de seguridad la cual proporciona un filtrado de trafico de red bidireccional. Posiblemente para nuestro caso en concreto sería más recomendable configurar dicho firewall para que nos permitiese comunicarnos con la estación en vez de desactivarlo por completo.

la comunicación remota (cambios descritos en el apartado 5.5). Para realizar dichos cambios es necesario conocer la IP que se le ha asignado al ESP32 en la red, para ello el programa te la indica al inicializarse (**Figura 47**).

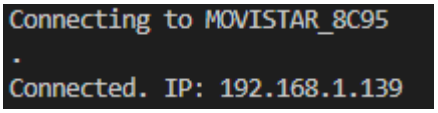


**Figura 47. IP asignada al ESP32 en la red**

A continuación, desconectando la estación del ordenador, ya podremos realizar todas las actualizaciones que deseemos de manera remota (**Figura 48**) siempre y cuando nos encontremos conectados a la misma red Wi-Fi que la estación.

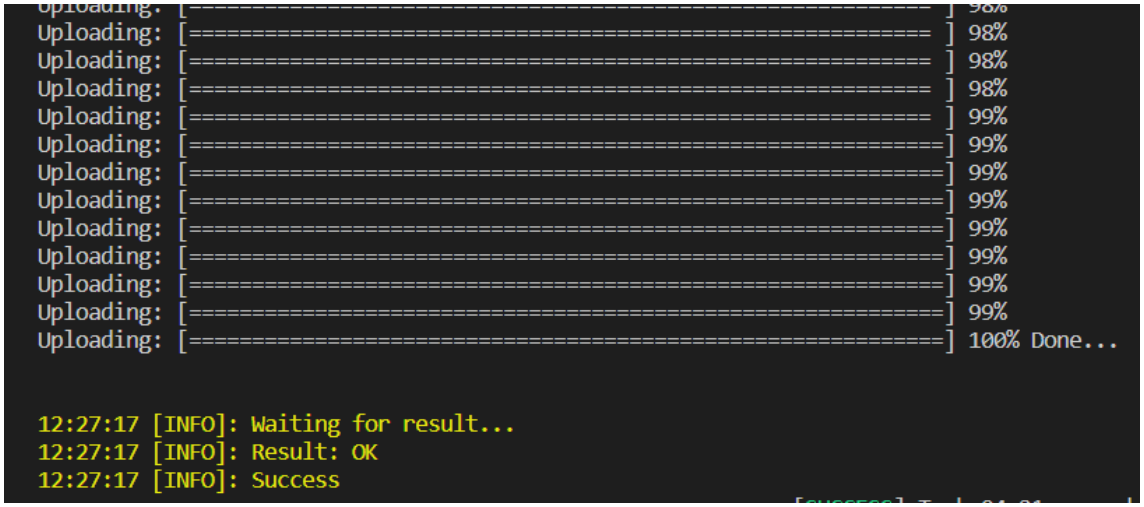


**Figura 48. Actualización remota del firmware realizada con éxito (monitor serial)**

La comprobación realizada inicialmente se basó en un programa Blink clásico que hiciera parpadear el led rojo cada segundo y se trató de cambiar el parpadeo a un periodo de 0,5 segundos de forma remota. Una vez logrado esto, se añadió dicha funcionalidad al programa principal, y se comprobó cambiando el valor de la temperatura ambiente a unidades Kelvin. En la **Figura 49** podemos observar como el cambio se ha realizado correctamente.

Se debe comprobar mediante Postman ya que al utilizar la funcionalidad de la actualización remota no podemos utilizar el monitor serial por lo que no hay forma de verificar que la actualización ha sido aplicada con éxito utilizando funciones como "Serial.Print".

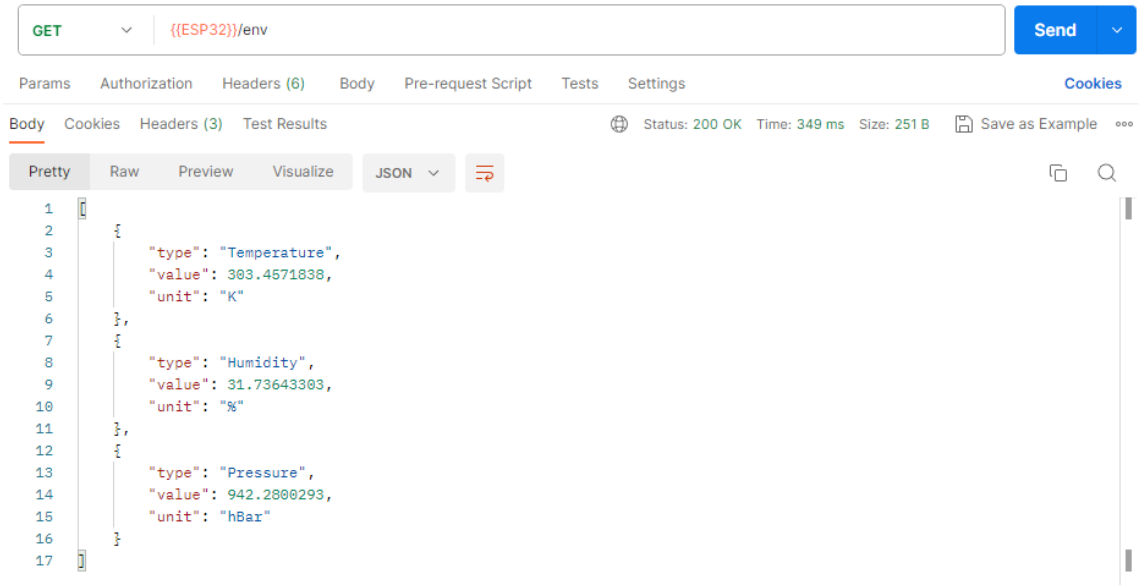


Figura 49. Comprobación mediante Postman de la actualización remota del firmware.

# Capítulo 7

# 7 Conclusiones y trabajos futuros

A modo de conclusión, en este capítulo se detallarán todas las impresiones que se han ido obteniendo durante la realización de este trabajo además de desarrollar ideas para la ampliación del proyecto.

## 7.1 Conclusiones

El objetivo principal de este proyecto ha sido hacer de la estación ya existente en la Universidad de Alcalá, la Estación Meteo, una estación más completa, profesional, e independiente.

Por un lado, se ha logrado de manera exitosa ampliar el número de variables meteorológicas medidas, siendo actualmente registradas hasta 7 diferentes características climáticas: Temperatura ambiente, presión atmosférica, humedad relativa, velocidad y dirección del viento, precipitación líquida e irradiancia.

Por otro lado, se ha conseguido una estación independiente añadiendo un sistema de alimentación autónomo mediante un panel fotovoltaico — permitiendo a la estación trabajar en momentos en los que la Universidad pueda sufrir cualquier tipo de problema eléctrico — y una batería, pudiendo trabajar hasta 34 horas sin luz solar.

Con la intención de proteger a la Estación Meteo y al estudio de las diferentes variables meteorológicos, también se ha añadido la posibilidad de seguir registrando las mediciones de los distintos instrumentos de medida en periodos de desconexión, durante al menos 5 minutos.

Otros aspectos significativos de la nueva estación meteorológica, haciéndola más moderna y completa que la estación con la que contaba el Grupo GEISER es la nueva forma de comunicación con cualquier base de datos, utilizando una API de REST, y la funcionalidad de actualización remota del firmware.

## 7.2 Trabajos futuros

La intención de este proyecto es asistir al Grupo GEISER con sus investigaciones relacionadas con la optimización de plantas fotovoltaicas. Por ello, la variable que desempeña un papel más importante para estos estudios es la irradiancia solar, siendo interesante sustituir los sensores actuales encargados de medir dicha variable por unos más precisos, como pueden ser los sensores de irradiación solar de celda de referencia ya probados en laboratorios profesionales.

Con la idea de hacer la Estación Meteo aún más independiente, otro trabajo futuro ha de ser la ampliación del tiempo que puede permanecer la estación sin conexión para no perder información de las variables meteorológicas medidas, añadiendo una memoria externa al microcontrolador ESP32 para así poder aumentar la capacidad de los buffers encargados de esta tarea y la ampliación del tiempo que puede permanecer la estación sin luz solar, añadiendo una batería de mayor capacidad o aumentando la velocidad de carga de esta.

Por último, un trabajo más enfocado en hacer la Estación Meteo más profesional sería aplicar todo lo estudiado en el Capítulo 2 acerca del intercambio de datos. De esta forma se podría aportar información relevante a las instituciones más importantes del sector como la AEMET o la OMM, además de hacer más sencillo el uso de mediciones registradas por otras estaciones meteorológicas.

# Capítulo 8

# 8 Presupuesto

En este apartado se detalla el presupuesto necesario para llevar a cabo este Trabajo Fin de Grado.

En la **Tabla 7** se puede observar de forma desglosada el precio de todos los materiales utilizados para la realizar este proyecto.

| | Concepto | Cantidad | Precio und. (€) | Total (€) |
|---|---|---|---|---|
| Instrumentos de medida | Adafruit Feather ESP32 | 1 | 20,43 € | 20,43 € |
| | ADS1115 16-bit ADC - 4 Canales | 1 | 24,14 € | 24,14 € |
| | Adafruit BME680 | 1 | 17,63 € | 17,63 € |
| | Sensor de baja corriente ACS723LLCTR-05AB-T | 1 | 16,28 € | 16,28 € |
| | Célula fotovoltaica Voltaic P121 | 1 | 5,07 € | 5,07 € |
| | RTD pt100 | 1 | 11,95 € | 11,95 € |
| | MAX31865 | 1 | 13,91 € | 13,91 € |
| | Anemómetro y veleta 6410 Davis Instrument | 1 | 229,00 € | 229,00 € |
| | Pluviómetro 6466M Davis Instrument | 1 | 150,00 € | 150,00 € |
| | Protoboard | 1 | 4,90 € | 4,90 € |
| | Kit de soldadura de estaño | 1 | 29,95 € | 29,95 € |
| | Caja derivación estanca 170 x 220 x 105 mm | 1 | 6,25 € | 6,25 € |
| Alimentación | Cargador solar Sunny Buddy | 1 | 26,94 € | 26,94 € |
| | Convertidor DC/DC Buck-Boost | 1 | 10,19 € | 10,19 € |
| | Adafruit LC709203F | 1 | 6,47 € | 6,47 € |
| | Panel fotovoltaíco PRT-13782 | 1 | 37,13 € | 37,13 € |
| | Resistencia 0,62 Ω | 1 | 0,10 € | 0,10 € |
| | Batería de litio UR18650ZY | 4 | 5,20 € | 20,80 € |

| | | | | |
|---|---|---|---|---|
| Software | Ordenador personal | 1 | 1.150,00 € | 1.150,00 € |
| | VisualStudio Code | 1 | - € | - € |
| | Windows 10 | 1 | - € | - € |
| | FreeRTOS | 1 | - € | - € |
| | Postman | 1 | - € | - € |
| | Cable USB 2.0 micro - B | 1 | 2,99 € | 2,99 € |

| | |
|---|---|
| TOTAL | 1.784,13 € |

**Tabla 7. Coste de los materiales para el desarrollo de la estación meteorológica**

El precio por hora trabajada de un Ingeniero Industrial para la realización de este trabajo es de aproximadamente 11,40 €/hora. Se consideran que se han trabajo 5 horas diarias en los 22 días laborables que tiene un mes de media, siendo la duración total de 7 meses. Resultando un costo de personal de 8.778,00 €.

Los gastos generales y de beneficio industrial se estiman un 20% del coste de materiales y personal, siendo el total 2.112,43 €.

Según el Colegio Oficial de Ingenieros Técnicos Industriales de Madrid, al no sobrepasar los 30.500 € de presupuesto, los honorarios facultativos se han establecido en un 0,255 % sobre el presupuesto total (costes de material y personal), conformando un total de 26,93 €.

El presupuesto total del proyecto a fecha de septiembre del año 2023 es de 12.701,49 €. La suma total de este valor se muestra en la **Tabla 8**.

| Concepto | Total (€) |
|---|---|
| Coste de material | 1.784,13 € |
| Coste de personal | 8.778,00 € |
| Gastos generales y beneficio industrial | 2.112,43 € |
| Coste de honorarios facultativos | 26,93 € |
| TOTAL | 12.701,49 € |

**Tabla 8. Coste total del trabajo**

# Bibliografía

[1] G. Moreno Baeza, «Biblioteca Digital Universidad de Alcalá,» 2022. [En línea]. Available: https://ebuah.uah.es/dspace/handle/10017/53830. [Último acceso: 2 agosoto 2023].

[2] Grupo GEISER, «Proyecto COPILOT,» [En línea]. Available: http://geiser.depeca.uah.es/copilot/index.php/es/. [Último acceso: 2 agosoto 2023].

[3] D. J. S. Quintana, Ministerio de asuntos exteriores, Unión Europea y cooperación, [En línea]. Available: https://www.exteriores.gob.es/RepresentacionesPermanentes/oficinadelasnacionesunidas/es/Organismo/Paginas/Organismos-especializados/OMM.aspx. [Último acceso: 24 julio 2023].

[4] Organización Meteorológica Mundial, «Historia de la OMM,» [En línea]. Available: https://public.wmo.int/es/acerca-de-la-omm/qui%C3%A9nes-somos/historia-de-la-omm. [Último acceso: 26 julio 2023].

[5] Agencia Estatal de Meteorología, «Quiénes somos,» [En línea]. Available: https://www.aemet.es/es/conocenos/quienes_somos. [Último acceso: 26 julio 2023].

[6] Organización Meteorológica Mundial, Norma sobre metadatos del Sistema Mundial Integrado de Sistemas de Observación de la OMM, 2019.

[7] «Seven Sensor,» 28 septiembre 2022. [En línea]. Available: https://www.sevensensor.com/es/comparacion-de-piranometros-vs-sensores-de-irradiacion-solar-de-celda-de-referencia. [Último acceso: 3 agosto 2023].

[8] IEA International Energy Agency, «Photovoltaic Module Energy Yield Measurements: Existing Approaches and Best Practice,» noviembre 2018. [En línea]. Available: https://iea-pvps.org/wp-content/uploads/2020/01/Photovoltaic_Module_Energy_Yield_Measurements_Existing_Approaches_and_Best_Practice_by_Task_13.pdf. [Último acceso: 4 agosto 2023].

[9] SparkFun, «SparkFun Sunny Buddy - MPPT Solar Charger,» [En línea]. Available: https://www.sparkfun.com/products/12885. [Último acceso: 4 agosto 2023].

[10] SparkFun, «SparkFun Solar Panel - 3.5W,» [En línea]. Available:

https://www.sparkfun.com/products/retired/13782. [Último acceso: 4 agosto 2023].

[11] SparkFun, «Sunny Buddy Solar Charger V13 Hookup Guide,» [En línea]. Available: https://learn.sparkfun.com/tutorials/sunny-buddy-solar-charger-v13-hookup-guide-?_ga=2.99768153.617346870.1682526379-1185931703.1680128409. [Último acceso: 4 agosto 2023].

[12] Panasonic, «Lithium Ion UR18650ZY,» [En línea]. Available: https://www.rowa.co.jp/data/img/UR18650ZY.pdf.

[13] Adafruit, «Adafruit LC709203F LiPoly / LiIon Fuel Gauge and Battery Monitor,» [En línea]. Available: https://www.adafruit.com/product/4712. [Último acceso: 24 agosto 2023].

[14] SparkFun, «SparkFun Buck - Boost Converter,» [En línea]. Available: https://www.sparkfun.com/products/15208 . [Último acceso: 6 agosto 2023].

[15] Naylamp mechatronics, «Naylamp mechatronics - MÓDULO ESP-WROOM-32 ESP32 WIFI,» [En línea]. Available: https://naylampmechatronics.com/espressif-esp/382-modulo-esp-wroom-32-esp32-wifi.html. [Último acceso: 7 agosto 2023].

[16] Makeability Lab; Professor Jon E. Froehlich, «Makeabilitylab Github ESP32,» [En línea]. Available: https://makeabilitylab.github.io/physcomp/esp32/esp32.html. [Último acceso: 8 agosto 2023].

[17] Bosch, BME680 - Datasheet, July 2017.

[18] Adafruit, «Adafruit BME680 - Temperature, Humidity, Pressure and Gas Sensor,» [En línea]. Available: https://www.adafruit.com/product/3660. [Último acceso: 2023 agosto 6].

[19] Adafruit, «Adafruit ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier,» [En línea]. [Último acceso: 6 agosto 2023].

[20] Meteo Shopping, «Meteo shopping - Anemómetro de paletas Vantage Pro,» [En línea]. Available: https://www.meteo-shopping.com/es/sensores/109-anemometro-de-paletas-vantage-pro.html. [Último acceso: 6 agosto 2023].

[21] Davis Instrument, «Pluviómetro de Cucharilla Basculante con AeroCone™ y Soporte de Montaje para Mástil,» [En línea]. Available: https://www.davisinstruments.com/products/aerocone-rain-collector-with-vantage-pro2-mounting-base?_pos=1&_sid=1b8e077ac&_ss=r&variant=39619863871649. [Último acceso: 12 agosto 2023].

[22] Seven, «Comparación de piranómetros vs. Sensores de irradiación solar de celda de referencia,» 28 septiembre 2022. [En línea]. Available: https://www.sevensensor.com/es/comparacion-de-piranometros-vs-sensores-de-irradiacion-solar-de-celda-de-referencia. [Último acceso: 13 agosto 2023].

[23] Ingelibre, «Ingelibre - Influencia de la irradiación y temperatura sobre una placa fotovoltaica,» 9 noviembre 2014. [En línea]. Available: https://ingelibreblog.wordpress.com/2014/11/09/influencia-de-la-irradiacion-y-temperatura-sobre-una-placa-fotovoltaica/. [Último acceso: 11 agosto 2023].

[24] SparkFun, «SparkFun Current Sensor Breakout - ACS723 (Low Current),» [En línea]. Available: https://www.sparkfun.com/products/14544. [Último acceso: 16 agosto 2023].

[25] SparkFun, «Current Sensor Breakout (ACS723) Hookup Guide,» [En línea]. Available: https://learn.sparkfun.com/tutorials/current-sensor-breakout-acs723-hookup-guide?_ga=2.145365169.445953029.1692876972-1185931703.1680128409&_gl=1*1671fui*_ga*MTE4NTkzMTcwMy4xNjgwMTI4NDA5*_ga_T369JS7J9N*MTY5Mjg5NDEzOC4xNy4wLjE2OTI4OTQxMzguNjAuMC4w. [Último acceso: 13 agosto 2023].

[26] Adafruit, «Adafruit PT100 RTD Temperature Sensor Amplifier,» [En línea]. Available: https://www.adafruit.com/product/3328. [Último acceso: 13 agosto 2023].

[27] M. Chojrin, «Platzi - Qué es una API REST (API RESTful),» [En línea]. Available: https://platzi.com/clases/1638-api-rest/21611-que-significa-rest-y-que-es-una-api-restful/. [Último acceso: 20 agosto 2023].

[28] Red Hat, «¿Qué es una API de REST?,» 31 julio 2023. [En línea]. Available: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Último acceso: 20 agosto 2023].

[29] «NTP Pool Project,» [En línea]. Available: https://www.ntppool.org/es/. [Último acceso: 19 agosto 2023].

[30] OpenWebinars, «Qué es Postman,» [En línea]. Available: https://openwebinars.net/blog/que-es-postman/. [Último acceso: 24 agosto 2023].

[31] Tecnoslab, «Tecnoslab - Características eléctricas de los paneles solares,» [En línea]. Available: https://tecnosolab.com/noticias/caracteristicas-electricas-de-los-paneles-solares/#:~:text=Los%20par%C3%A1metros%20caracter%C3%ADsticos%20de%20un,espectral%20de%20AM%201%2C5G.. [Último acceso: 11 agosto 2023].

# Anexo 1: Main.cpp

```
#define GMT 2 // Zona horaria GMT +2
#define HORARIO_VERANO 0 // 0 = no ; 1 = si

#include <Arduino.h>
#include <Wire.h>
#include <WiFi.h>
#include <math.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADS1X15.h>
#include <Adafruit_BME680.h>
#include <Adafruit_LC709203F.h>
#include <Adafruit_MAX31865.h>
#include <WebServer.h>
#include <ArduinoJson.h>
#include <ESP32Time.h>

bool connected; // Flag para saber si estamos conectados
bool FirstTime;  // Flag para saber si borrar buffers o no
bool InicioDesconexion; // Flag para calcular las fechas y tiempo de
desconexión


char Tiempo_desconectado[7];
float t_ini_desc;
void tiempoDesconectado(){
  float t_desc = millis() - t_ini_desc;
  int segundos_desconectado = t_desc / 1000;
  int minutos_desconectado = segundos_desconectado / 60;
  segundos_desconectado = segundos_desconectado % 60;
  InicioDesconexion = true;
  sprintf(Tiempo_desconectado, "%dm:%ds", minutos_desconectado,
segundos_desconectado);
  Serial.print("Tiempo desconectado:
");Serial.println(Tiempo_desconectado);
}

ESP32Time rtc;
char Fecha_desconexion[22];
```

```
char Fecha[22];
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = GMT*3600;
const int daylightOffset_sec = HORARIO_VERANO;
float tiempo_inicio_desconexion, tiempodesconectado;
int anioinicial, mesinicial, diainicial, horainicial, minutoinicial,
segundoinicial, bisiesto;
int anioActual, mesActual, diaActual, horaActual, minutoActual,
segundoActual;

void setupFecha(){ // Registro la fecha actual al iniciar del programa
  if(connected == true){
    if(InicioDesconexion == false) tiempoDesconectado(); // Calculo del
tiempo desconectados
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    struct tm timeinfo;
    if(getLocalTime(&timeinfo)) rtc.setTimeStruct(timeinfo);
      anioinicial = rtc.getYear();
      mesinicial = rtc.getMonth() + 1;
      diainicial = rtc.getDay();
      horainicial = rtc.getHour(true);
      minutoinicial = rtc.getMinute();
      segundoinicial = rtc.getSecond();
    sprintf(Fecha, "%d/%d/%d %dh:%dm:%ds", diainicial, mesinicial,
anioinicial, horainicial, minutoinicial, segundoinicial);
    tiempo_inicio_desconexion = millis(); //Para calcular el tiempo desde
que se actualizó la fecha con Wi-Fi
  }

  if(connected == false && InicioDesconexion == true){
    t_ini_desc = millis(); // Inicia el periodo sin conexión
    // Guardamos fecha inicio de desconexión
    mesActual = mesinicial;
    anioActual = anioinicial;
      if ( anioActual % 4 == 0 && anioActual % 100 != 0) bisiesto = 1;
        else if ( anioActual % 100 == 0 && anioActual % 400 == 0)
bisiesto = 1;
        else bisiesto = 0;
      if(mesActual == (1 || 3 || 5 || 7 || 8 || 10) && diaActual == 31 &&
horaActual == 23 && minutoActual == 59 && segundoActual >= 59)
mesinicial++;
      if(mesActual == (4 || 6 || 9 || 11) && diaActual == 30 &&
horaActual == 23 && minutoActual == 59 && segundoActual >= 59)
mesinicial++;
      if(mesActual == 2 && bisiesto == 0 && diaActual == 28 && horaActual
== 23 && minutoActual == 59 && segundoActual >= 59) mesinicial++;
      if(mesActual == 2 && bisiesto == 1 && diaActual == 29 && horaActual
== 23 && minutoActual == 59 && segundoActual >= 59) mesinicial++;
```

```
      if(mesActual == 12 && diaActual == 31 && horaActual == 23 &&
minutoActual == 59 && segundoActual >= 59) anioinicial++;
    tiempodesconectado = (millis() - tiempo_inicio_desconexion)/1000;
    horaActual = diainicial * 24 + horainicial;
    minutoActual = horaActual * 60 + minutoinicial;
    segundoActual = minutoActual * 60 + segundoinicial + (int)
tiempodesconectado;
    diaActual = segundoActual / (24 * 60 * 60); segundoActual %= (24 * 60
* 60);
    horaActual = segundoActual / (60 * 60); segundoActual %= (60 * 60);
    minutoActual = segundoActual / 60; segundoActual = segundoActual %
60;
    sprintf(Fecha_desconexion, "%d/%d/%d %dh:%dm:%ds", diaActual,
mesActual, anioActual, horaActual, minutoActual, segundoActual);
    Serial.print("Fecha y hora de desconexión:
");Serial.println(Fecha_desconexion);
    InicioDesconexion = false;
  }
}

// TASKS
#define CORE_0 0
#define CORE_1 1
#define T_TASK_LECTURA 3000  //ms
#define T_TASK_NUBE 3000  //ms
void *pvParameters;
TaskHandle_t handle_task_lectura, handle_task_nube;
void TaskLectura(void *pvParameters);
void TaskNube(void *pvParameters);

SemaphoreHandle_t mutex_datos;

// WIFI
#include "Credentials.h"
WebServer server(80);

void connectToWiFi() { // Nos conectamos InicioDesconexion la wifi
  Serial.print("Connecting to "); Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  }
  Serial.println("."); Serial.print("Connected. IP: ");
Serial.println(WiFi.localIP());
  connected = true;
}

// BME680
Adafruit_BME680 bme;
```

```
float Presion, Temperatura, Humedad, Gas;

// IRRADIANCIA
float adc0_V, Isc, Irradiancia;
int Vref = 2.55;
int Gain = 6000; //mV/A

// RTD
#define MAX_CS 33
#define MAX_CLK 5
#define MAX_MISO 19
#define MAX_MOSI 18
#define RNOMINAL 100
#define RREF 430
Adafruit_MAX31865 RTD(MAX_CS,MAX_MOSI,MAX_MISO,MAX_CLK);
float rtd, Temperatura_RTD;

// BATERIA
Adafruit_LC709203F LC709203F;
float porcentaje_bateria, tension_bateria;

// VELETA
Adafruit_ADS1115 ads;
#define Offset_direction -75;
int VaneValue;    // Valor RAW Analógico de la veleta
int Direction;    // Traduccion InicioDesconexion grados (0º
InicioDesconexion 360º)
int CalDirection; // Valor calibrado (con el offset añadido)
int LastValue;
float direc_viento;

// WINDSPEED
#define WindSensorPin (21)
#define wind_wait 1000
float radio = 5.5; // cm
volatile unsigned long Rotations;         // Contador de las rotaciones
(rutina con interrupcion)
volatile unsigned long ContactBounceTime; // TimeR para evitar contact
bounce en la interrupcion
float WindSpeed, t_ini_windspeed, t_windspeed; // km/h

//OBTENCION DEL RUMBO
char* brujula;
void getHeading(int direction){
  if (direction < 22) Serial.println("N");
  else if (direction < 67) {
    brujula = "NE";
    Serial.println("NE");
  }
```

```
  else if (direction < 112){
    brujula = "E";
    Serial.println("E");
   }
  else if (direction < 157){
    brujula = "SE";
    Serial.println("SE");
   }
  else if (direction < 212) {
    brujula = "S";
    Serial.println("S");
  }
  else if (direction < 247) {
    brujula = "SW";
    Serial.println("SW");
  }
  else if (direction < 292) {
    brujula = "W";
    Serial.println("W");
  }
  else if (direction < 337) {
    brujula = "NW";
    Serial.println("NW");
  }
  else{
    brujula = "N";
    Serial.println("N");
  }
}

//LECTURA DE DATOS DE LA VELETA
float leer_veleta_direccion(int16_t VaneValue){
  Direction = map(VaneValue, 0, 20595, 0, 360); //Mapeamos el valor de la
veleta (0-20595) InicioDesconexion un nuevo valor entre 0º y 360º
  CalDirection = Direction + Offset_direction; //Direccion calibrada
    if (CalDirection > 360) CalDirection = CalDirection - 360;
    if (CalDirection < 0) CalDirection = CalDirection + 360; // Solo
actualizamos el display si el valor cambia mas de 2 grados
  return CalDirection;
}

void isr_rotation(){
  if ((millis() - ContactBounceTime) > 15){ // El tiempo entre rotaciones
es menor InicioDesconexion 15ms
    Rotations++;
    ContactBounceTime = millis();
  }
}
```

```
// PLUVIOMETRO
#define PluviometerSensorPin (36)
#define T_MUESTREO_PRECIPITACION 3600000 // 1 hora
int pulsos = 0, q = 0;
float t_pluvi, t_ini_pluvi, lluvia_mm;

void isr_pluviometer(){
  pulsos++;
}

// BUFFERS
int x = 0;
const int sizeBuffer = 1220;
char *bufferFecha[sizeBuffer];
char bufferdate[sizeBuffer];
float bufferTemperatura[sizeBuffer], bufferPresion[sizeBuffer],
bufferHumedad[sizeBuffer];
char buffertemp[sizeBuffer], bufferpres[sizeBuffer],
bufferhum[sizeBuffer];
float bufferPorcentaje[sizeBuffer], bufferTension[sizeBuffer];
char bufferpcnt[sizeBuffer], bufferv[sizeBuffer];
float bufferIrradiancia[sizeBuffer];
char bufferirr[sizeBuffer];
float bufferRTD[sizeBuffer];
char bufferrtd[sizeBuffer];
char *bufferBrujula[sizeBuffer];
float bufferDirec[sizeBuffer], bufferVelocidad[sizeBuffer];
char bufferdirec[sizeBuffer], bufferbrujula[sizeBuffer],
buffervel[sizeBuffer];
float bufferPluviometro[sizeBuffer];
char bufferpluvi[sizeBuffer];

void generadorBuffers(){ // Cargamos las mediciones en los buffers
  int offsettemp = 0, offsetpres = 0, offsethum = 0, offsetdates = 0,
offsetpcnt = 0, offsetv = 0, offsetirr = 0, offsetrtd = 0;
  int offsetdirec = 0, offsetbruju = 0, offsetvel = 0, offsetpluvi = 0;

  //bufferFecha[x] = Fecha_desconexion;
  bufferTemperatura[x] = (Temperatura);
  bufferPresion[x] = (Presion);
  bufferHumedad[x] = (Humedad);
  bufferPorcentaje[x] = (porcentaje_bateria);
  bufferTension[x] = (tension_bateria);
  bufferIrradiancia[x] = (Irradiancia);
  bufferRTD[x] = (Temperatura_RTD);
  bufferBrujula[x] = brujula;
  bufferDirec[x] = (direc_viento);
  bufferVelocidad[x] = (WindSpeed);
  bufferPluviometro[x] = (lluvia_mm);
```

```
  x++;
  for (int i = 0; i < (x); i++) {
    int temperaturas = snprintf(buffertemp + offsettemp, sizeBuffer -
offsettemp, "%f, ", bufferTemperatura[i]);
    if (temperaturas >= 0 && temperaturas < sizeBuffer - offsettemp)
offsettemp += temperaturas;
    else break;
    // buffer presion (char)
    int presiones = snprintf(bufferpres + offsetpres, sizeBuffer -
offsetpres, "%f, ", bufferPresion[i]);
    if (presiones >= 0 && presiones < sizeBuffer - offsetpres) offsetpres
+= presiones;
    else break;
    // buffer humedad (char)
    int humedades = snprintf(bufferhum + offsethum, sizeBuffer -
offsethum, "%f, ", bufferHumedad[i]);
    if (humedades >= 0 && humedades < sizeBuffer - offsettemp) offsethum
+= humedades;
    else break;
    // buffer porcentaje de carga (char)
    int porcentajes = snprintf(bufferpcnt + offsetpcnt, sizeBuffer -
offsetpcnt, "%f, ", bufferPorcentaje[i]);
    if (porcentajes >= 0 && porcentajes < sizeBuffer - offsetpcnt)
offsetpcnt += porcentajes;
    else break;
    // buffer tension de la bateria (char)
    int tensiones = snprintf(bufferv + offsetv, sizeBuffer - offsetv,
"%f, ", bufferTension[i]);
    if (tensiones >= 0 && tensiones < sizeBuffer - offsetv) offsetv +=
tensiones;
    else break;
    // buffer irradiancia (char)
    int irradiancias = snprintf(bufferirr + offsetirr, sizeBuffer -
offsetirr, "%f, ", bufferIrradiancia[i]);
    if (irradiancias >= 0 && irradiancias < sizeBuffer - offsetirr)
offsetirr += irradiancias;
    else break;
    // buffer temperatura del panel(char)
    int rtds = snprintf(bufferrtd + offsetrtd, sizeBuffer - offsetrtd,
"%f, ", bufferRTD[i]);
    if (rtds >= 0 && rtds < sizeBuffer - offsetrtd) offsetrtd += rtds;
    else break;
    // buffer direccion del viento brujula (char)
    int brujulas = snprintf(bufferbrujula + offsetbruju, sizeBuffer -
offsetbruju, "%s, ", bufferBrujula[i]);
    if (brujulas >= 0 && brujulas < sizeBuffer - offsetbruju) offsetbruju
+= brujulas;
    else break;
    // buffer direccion del viento grados (char)
```

```
    int direcciones = snprintf(bufferdirec + offsetdirec, sizeBuffer -
offsetdirec, "%f, ", bufferDirec[i]);
    if (direcciones >= 0 && direcciones < sizeBuffer - offsetdirec)
offsetdirec += direcciones;
    else break;
    // buffer WindSpeed (char)
    int velocidades = snprintf(buffervel + offsetvel, sizeBuffer -
offsetvel, "%f, ", bufferVelocidad[i]);
    if (velocidades >= 0 && velocidades < sizeBuffer - offsetvel)
offsetvel += velocidades;
    else break;
    // buffer Pluviometer (char)
    int lluvias = snprintf(bufferpluvi + offsetpluvi, sizeBuffer -
offsetpluvi, "%f, ", bufferPluviometro[i]);
    if(lluvias >= 0 && lluvias < sizeBuffer - offsetpluvi) offsetpluvi +=
lluvias;
    else break;
  }
}

int TempSend = 0, PressSend = 0, HumSend = 0;
int  PcntSend = 0, VdSend = 0;
int IrrSend = 0, RTDSend = 0;
int  DirSend = 0, VelSend = 0;
int  PluviSend = 0;

void vaciarBuffers(){
  x = 0;
  //memset(bufferdate, '\0', sizeBuffer);
  memset(buffertemp, '\0', sizeBuffer);
  memset(bufferpres, '\0', sizeBuffer);
  memset(bufferhum, '\0', sizeBuffer);
  memset(bufferpcnt, '\0', sizeBuffer);
  memset(bufferv, '\0', sizeBuffer);
  memset(bufferirr, '\0', sizeBuffer);
  memset(bufferrtd, '\0', sizeBuffer);
  memset(bufferbrujula, '\0', sizeBuffer);
  memset(bufferdirec, '\0', sizeBuffer);
  memset(buffervel, '\0', sizeBuffer);
  memset(bufferpluvi, '\0', sizeBuffer);
  TempSend = 0; PressSend = 0; HumSend = 0;
  PcntSend = 0; VdSend = 0;
  IrrSend = 0; RTDSend = 0;
  DirSend = 0; VelSend = 0;
  PluviSend = 0;
}

// JSON BUFFER
```

```
StaticJsonDocument<500> jsonDocument; // Almacenaremos los datos en un
documento JSON
char buffer_WiFi[500]; // Creamos un buffer para guardar todo
void add_json_object(char *tag, float value, char *unit){ // Para anindar
mediciones en el JSON
  JsonObject obj = jsonDocument.createNestedObject();
  obj["type"] = tag;
  obj["value"] = value;
  obj["unit"] = unit;
}
StaticJsonDocument<2000> jsonDocumentDISconnected; // Almacenaremos los
datos en un documento JSON
char bufferDISconnected[2000]; // Creamos un buffer para guardar todo
void add_json_object_DISconnected(char *date, char * t, char *tag, char
*value, char *unit){ // Para anindar mediciones en el JSON
  JsonObject objDISconnected =
jsonDocumentDISconnected.createNestedObject();
  objDISconnected["Date"] = date;
  objDISconnected["Time"] = t;
  objDISconnected["type"] = tag;
  objDISconnected["value"] = value;
  objDISconnected["unit"] = unit;
}
void getEnv(){
  Serial.println("Get env");
  xSemaphoreTake(mutex_datos, portMAX_DELAY);
    jsonDocument.clear();
    add_json_object("Temperature", Temperatura, "K");
    add_json_object("Humidity", Humedad, "%");
    add_json_object("Pressure", Presion, "hBar");
    serializeJson(jsonDocument, buffer_WiFi);
    server.send(200, "application/json", buffer_WiFi);
  xSemaphoreGive(mutex_datos);
}
void getWind(){
  Serial.println("Get Wind Data");
  xSemaphoreTake(mutex_datos, portMAX_DELAY);
    jsonDocument.clear();
    add_json_object("WindDirection", direc_viento, brujula);
    add_json_object("WindSpeed", WindSpeed, "km/h");
    add_json_object("Rotationes", Rotations, " ");
    add_json_object("Time", t_windspeed, "ms");
    serializeJson(jsonDocument, buffer_WiFi);
    server.send(200, "application/json", buffer_WiFi);
  xSemaphoreGive(mutex_datos);
}
void getCell(){
  Serial.println("Get Cell Data");
  xSemaphoreTake(mutex_datos, portMAX_DELAY);
```

```
    jsonDocument.clear();
    add_json_object("Irradiance", Irradiancia, "W/m²");
    add_json_object("Cell temperature", Temperatura_RTD, "ºC");
    serializeJson(jsonDocument, buffer_WiFi);
    server.send(200, "application/json", buffer_WiFi);
  xSemaphoreGive(mutex_datos);
}
void getBattery(){
  Serial.println("Get Battery Data");
  xSemaphoreTake(mutex_datos, portMAX_DELAY);
    jsonDocument.clear();
    add_json_object("Battery charge", porcentaje_bateria, "%");
    add_json_object("Battery tension", tension_bateria, "V");
    serializeJson(jsonDocument, buffer_WiFi);
    server.send(200, "application/json", buffer_WiFi);
  xSemaphoreGive(mutex_datos);
}
void getPluvi(){
  Serial.println("Get Pluviometer Data");
  xSemaphoreTake(mutex_datos, portMAX_DELAY);
    jsonDocument.clear();
    add_json_object("Precipitation", lluvia_mm, "mm");
    add_json_object("Time", t_pluvi, "s");
    serializeJson(jsonDocument, buffer_WiFi);
    server.send(200, "application/json", buffer_WiFi);
  xSemaphoreGive(mutex_datos);
}
void getTempdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"temperature", buffertemp, "ºC");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    TempSend = 1;
}
void getPressdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"pressure", bufferpres, "hPa");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    PressSend = 1;
}
void getHumdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"humidity", bufferhum, "%");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
```

```
    HumSend = 1;
}
void getPcntdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Battery charge", bufferpcnt, "%");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    PcntSend = 1;
}
void getVdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Battery tension", bufferv, "V");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    VdSend = 1;
}
void getIrrdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Irradiance", bufferirr, "W");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    IrrSend = 1;
}
void getRTDdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Cell temperature", bufferrtd, "ºC");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    RTDSend = 1;
}
void getDirecdisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Wind direction", bufferdirec, bufferbrujula);
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    DirSend = 1;
}
void getVeldisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Wind speed", buffervel, "km/h");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    VelSend = 1;
```

```
}
void getPluvidisconnected(){
  jsonDocumentDISconnected.clear();
    add_json_object_DISconnected(Fecha_desconexion, Tiempo_desconectado,
"Water fall in 3 seconds", bufferpluvi, "mm");
    serializeJson(jsonDocumentDISconnected, bufferDISconnected);
    server.send(200, "application/json", bufferDISconnected);
    PluviSend = 1;
}
void setup_routing(){  // Configuramos las rutas
  server.on("/env", getEnv);
  server.on("/wnd", getWind);
  server.on("/cell", getCell);
  server.on("/bat", getBattery);
  server.on("/pluv", getPluvi);
  server.on("/tempdisconnected", getTempdisconnected);
  server.on("/pressdisconnected", getPressdisconnected);
  server.on("/humdisconnected", getHumdisconnected);
  server.on("/pcntdisconnected", getPcntdisconnected);
  server.on("/vdisconnected", getVdisconnected);
  server.on("/irrdisconnected", getIrrdisconnected);
  server.on("/rtddisconnected", getRTDdisconnected);
  server.on("/direcdisconnected", getDirecdisconnected);
  server.on("/veldisconnected", getVeldisconnected);
  server.on("/pluvidisconnected", getPluvidisconnected);
  server.begin(); // Iniciamos el servidor
}

void setup(){
  Serial.begin(115200);
  ads.begin();
  RTD.begin(MAX31865_4WIRE);
  mutex_datos = xSemaphoreCreateMutex();
  connectToWiFi();
  connected = true;
  FirstTime = true;
  InicioDesconexion = true;
  OTAsetup();
  setupFecha();
  Serial.print("Fecha y hora Inicial: ");Serial.println(Fecha);

  if(!bme.begin())Serial.println("No se pudo encontrar el sensor
BME680");
  if(!LC709203F.begin()) Serial.println("No se pudo medir correctamente
los datos de la bateria");

  pinMode(WindSensorPin, INPUT_PULLUP);
  pinMode(PluviometerSensorPin, INPUT_PULLUP);
  pinMode(13, OUTPUT);
```

```
  xTaskCreatePinnedToCore(TaskLectura, "Lectura", 4000, NULL, 1,
&handle_task_lectura, CORE_0);
  xTaskCreatePinnedToCore(TaskNube, "Prints", 4000, NULL, 1,
&handle_task_nube, CORE_1);

  setup_routing();
}

void loop(){
  ArduinoOTA.handle();
  if(!bme.performReading()){
    Serial.println("Error al realizar la lectura");
    return;
  }
  if(WiFi.status() != WL_CONNECTED){
    connected = false;
    digitalWrite(13, LOW);
  }
  else{
    connected = true;
    digitalWrite(13, HIGH);
  }
  server.handleClient();
  delay(500);
}

void TaskLectura(void *pvParameters){
  (void)pvParameters;
  TickType_t xLastTakeTime_Lectura;
  xLastTakeTime_Lectura = xTaskGetTickCount();
  for(;;){
      attachInterrupt(digitalPinToInterrupt(WindSensorPin), isr_rotation,
FALLING);
      attachInterrupt(digitalPinToInterrupt(PluviometerSensorPin),
isr_pluviometer, FALLING);
    xSemaphoreTake(mutex_datos, portMAX_DELAY);
      setupFecha();
    Serial.println("BME680 ---------------"); // BME680
      Temperatura = bme.temperature;
      Presion = bme.pressure/100.00;
      Humedad = bme.humidity;
    Serial.println("WIND DIRECC ----------"); // DIRECCION DEL VIENTO
      int adc1 = ads.readADC_SingleEnded(1);
      direc_viento = leer_veleta_direccion(adc1);
    Serial.println("MAX31865 -------------"); // RTD
      rtd = RTD.readRTD();
      Temperatura_RTD = RTD.temperature(RNOMINAL,RREF);
    Serial.println("IRRADIACIA------------"); // IRRADIANCIA
```

```
      int adc0 = ads.readADC_SingleEnded(0);
      adc0_V = adc0 * 0.0001875;   // 188uV/bit = 0.188mV/bit
      Isc = (((adc0_V - Vref)*1000) / Gain)*1000; //mA
      Irradiancia = Isc / (0.15 + 0.15 * 0.0005 * (Temperatura_RTD -
25));
    Serial.println("LC709203F ------------"); // BATERIA
      tension_bateria = LC709203F.cellVoltage();
      porcentaje_bateria = LC709203F.cellPercent();
    Serial.println("PLUVIOMETRO ----------"); // PLUVIOMETRO
      t_pluvi = millis() - t_ini_pluvi;
      lluvia_mm = pulsos * 0.2;
      pulsos = 0;
      t_ini_pluvi = millis();
    Serial.println("WINDSPEED ------------"); // WINDSPEED
      t_windspeed = millis() - t_ini_windspeed;
      WindSpeed = (Rotations * 2 * 3.1416 * radio * 3600 * 1000) /
((int)t_windspeed * 100000);
      Serial.println(t_windspeed);
      t_ini_windspeed = millis();
    xSemaphoreGive(mutex_datos);
    xTaskDelayUntil(&xLastTakeTime_Lectura, T_TASK_LECTURA);
  }
}

void TaskNube(void *pvParameters){
  (void)pvParameters;
  TickType_t xLastTakeTime_Nube;
  xLastTakeTime_Nube = xTaskGetTickCount();
  for(;;){
    xSemaphoreTake(mutex_datos, portMAX_DELAY);
    if(connected == true){
        FirstTime = true;
        Serial.print("Fecha y hora: ");Serial.println(Fecha);
        Serial.print("Temperatura: "); Serial.print(Temperatura);
Serial.println(" ºC");
        Serial.print("Presion: "); Serial.print(Presion);
Serial.println(" hPa");
        Serial.print("Humedad: "); Serial.print(Humedad);
Serial.println(" %");
      Serial.println(" ");
        Serial.print("Dirección del viento:
");Serial.print(direc_viento); Serial.print("º ");
getHeading(CalDirection);
      Serial.println(" ");
        Serial.print("Irradiancia:
");Serial.print(adc0_V);Serial.println(" V");
        Serial.print("Irradiancia: ");Serial.print(Isc);Serial.println("
mA");
```

```
        Serial.print("Irradiancia:
");Serial.print(Irradiancia);Serial.println(" W/m²");
      Serial.println(" ");
        Serial.print("RTD: "); Serial.print(rtd); Serial.println("
Ohms");
        Serial.print("Temperatura del panel: ");
Serial.print(Temperatura_RTD); Serial.println(" ºC");
      Serial.println(" ");
        Serial.print("Tension de la bateria: ");
Serial.print(tension_bateria); Serial.println(" V");
        Serial.print("Carga de la bateria: ");
Serial.print(porcentaje_bateria); Serial.println(" %");
      Serial.println(" ");
        Serial.print("Precipitacion en los ultimos ");
Serial.print((int)t_pluvi / 1000); Serial.print(" segundos: ");
        Serial.print(lluvia_mm); Serial.println(" mm");
      Serial.println(" ");
        Serial.print("Velocidad del viento: ");
Serial.print(WindSpeed);  Serial.println(" km/h");
        Serial.print("Rotaciones: ");Serial.println(Rotations);
      Serial.println(" ");
        Rotations = 0; // Reseteamos rotaciones
    }
    if(connected == false && FirstTime == true){
      if(TempSend, HumSend, PressSend, VelSend, DirSend, VdSend,
PcntSend, RTDSend, IrrSend, PluviSend == 1) vaciarBuffers();
      Serial.println("++++++++++++++++++++++++++++++++++++");
      FirstTime = false;
    }
    if(connected == false && FirstTime == false) {
        generadorBuffers();
        Serial.print("Fecha y hora de desconexión:
");Serial.println(Fecha_desconexion);
        Serial.print("Buffer de temperatura: ");
Serial.println(buffertemp);
        Serial.print("Buffer de presion: "); Serial.println(bufferpres);
        Serial.print("Buffer de humedad: "); Serial.println(bufferhum);
        Serial.print("Buffer de porcentaje bateria: ");
Serial.println(bufferpcnt);
        Serial.print("Buffer de tension bateria: ");
Serial.println(bufferv);
        Serial.print("Buffer de irradiancia: ");
Serial.println(bufferirr);
        Serial.print("Buffer de RTD: "); Serial.println(bufferrtd);
        Serial.print("Buffer de direccion del viento: ");
Serial.println(bufferdirec);
        Serial.print("Buffer de brujula: ");
Serial.println(bufferbrujula);
```

```
        Serial.print("Buffer de velocidad del viento: ");
Serial.println(buffervel);
        Serial.print("Buffer de precipitacion: ");
Serial.println(bufferpluvi);
    }
  xSemaphoreGive(mutex_datos);
  xTaskDelayUntil(&xLastTakeTime_Nube, T_TASK_NUBE);
  }
}
```

# Anexo 2: Credentials.h

```
#include <ArduinoOTA.h>

// IP: XXX.XXX.X.XXX
const char* ssid = "***********"; // Escribe el nombre de la red
const char* password = "***********";// Escribe la contraseña de la red

void OTAsetup(){
ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else
        type = "filesystem";
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
    .onError([](ota_error_t error) {
      Serial.printf("Error[%u]: ", error);
      if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
      else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
      else if (error == OTA_CONNECT_ERROR) Serial.println("Connect
Failed");
      else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive
Failed");
      else if (error == OTA_END_ERROR) Serial.println("End Failed");
    });

  ArduinoOTA.begin();
}
```

# Anexo 3: Platformio.ini

```
[platformio]
description = Estacion Meteo
default_envs = featheresp32
src_dir = .
include_dir = include
libdeps_dir = dependencies

[env:featheresp32]
platform = espressif32
board = featheresp32
framework = arduino
monitor_speed = 115200
;upload_protocol = espota ;; Comentar si estamos conectados por USB
;upload_port = XXX.XXX.X.XXX ;; Comentar si estamos conectados por USB
lib_deps =
    mathworks/ThingSpeak @ ^2.0.0
    adafruit/Adafruit ADS1X15 @ ^2.4.0
    adafruit/Adafruit Unified Sensor @ ^1.1.7
    adafruit/Adafruit BME680 Library @ ^2.0.2
    adafruit/Adafruit MAX31865 library @ ^1.6.0
    knolleary/PubSubClient @ ^2.8
    adafruit/Adafruit LC709203F@^1.3.2
    bblanchon/ArduinoJson@^6.21.2
    fbiego/ESP32Time@^2.0.3
```

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR