ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE LENGUAJES Y SISTEMAS INFORMÁTICOS

DESARROLLO DE UN DISPOSITIVO BASADO EN MICROCONTROLADOR PARA LA CAPTACIÓN DE DATOS DE DIVERSA NATURALEZA EN GLOBOS DE ALTA ALTITUD

D. Alonso González, David
TUTOR: D. Muñiz Sánchez, Rubén
COTUTOR: Díez Alonso, Enrique

FECHA: Julio de 2023

Contenidos

Figuras

Tablas

1. Introducción

El proyecto de fin de grado 'Desarrollo de un dispositivo basado en microcontrolador para la captación de datos de diversa naturaleza en globos de alta altitud' tendrá como tutor al docente e investigador de la Universidad de Oviedo, Rubén Muñiz Sánchez, cuya área de conocimiento es Lenguajes y Sistemas informáticos, como apoyo, se contará ejerciendo de cotutor del proyecto al docente de la Universidad de Oviedo Enrique Díez Alonso, docente en astrofísica miembro del ICTEA y del Departamento de matemáticas.

El principal objetivo de este proyecto de fin de grado es el desarrollo de una solución, que abarque los ámbitos hardware y software, logrando la captación de datos meteorológicos de relevancia para el estudio, tales como la temperatura, luminiscencia, presión… en experimentos comúnmente realizados a la estratosfera utilizando globos de alta altitud como medio.

Repositorio del proyecto:

https://drive.google.com/file/d/1gNlcNrBXYBkjMq_OVlWzwbC7Ef6p9dIg/view?usp=sharing

Contiene:

- Código fuente para microprocesador y sensores.
- Código fuente aplicación web.
- Código fuente aplicación móvil.
- .APK de la aplicación móvil.
- Instalador aplicación web.

2. Objetivos y alcance

Es de vital importancia para ser capaces de figurar el objetivo del proyecto, comprender que los datos de diversas naturalezas tales como temperatura, humedad, presión… captados en la estratosfera son y llevan siendo de gran importancia numerosas décadas para el estudio de la misma. La gran problemática de este tipo de estudio, es la elección de los dispositivos captadores de datos y el software que apoyará tanto la toma de datos, como en paralelo el análisis de estos si es requerido.

A sabiendas de lo anteriormente citado, este proyecto se dividió en varios objetivos, intentando enriquecerlo lo más posible en el campo software como en el campo hardware:

- Diseño hardware del dispositivo microcontrolador (procesador) con los respectivos sensores que captarán datos de diversos intereses.
- Desarrollo de software para microprocesador, que permita manejar y almacenar los distintos datos captados por los periféricos.
- Desarrollo de aplicación móvil que permita monitorear en directo los datos captados por el microcontrolador y los sensores.
- Desarrollo de aplicación web que posibilite un análisis más cuidado de los datos recogidos por microcontrolador y sensores.

2.1.-Diseño Hardware

La elección del hardware debe ser la adecuada, se necesita tener en consideración las posibles dificultades del trayecto a la estratosfera realizado por un globo de alta altitud. Se estudiarán alternativas y se seleccionarán correctamente los dispositivos finales entre los que destacan:

- Un microcontrolador, se encargará del procesamiento de los datos recogidos y el control sobre los sensores y otros periféricos presentes.
- Un móvil, de sistema operativo Android por ahorro de costes, proporcionando una fuente de alimentación, conexión a internet y localización GPS constante del globo de alta altitud.
- Sensores de captación de diversa naturaleza y otros periféricos, tales como:
    - Sensor de temperatura
    - Sensor de humedad
    - Sensor de presión
    - Sensor de incidencia ultravioleta
    - SD reader para guardar el fichero de datos.

2.2.-Desarrollo de software para microcontrolador

Deberemos emplear tiempo de trabajo en el desarrollo de un robusto código para microprocesador, que permita almacenar y gestionar los datos sensoriales posibilitando que el hardware mencionado en el apartado anterior funcione en perfecta sincronía y no induzca en errores en el tratamiento de los datos.

2.3.-Desarrollo de una aplicación móvil

Se desarrollará una aplicación móvil que permita al usuario cliente el monitoreo en tiempo real de los datos captados por nuestros sensores, recibiendo el usuario un feedback continuo del correcto funcionamiento del dispositivo captador de datos. Los datos captados, serán presentados en esta aplicación de manera cruda, pues la aplicación que servirá para el análisis de los mismos será la aplicación web.

2.4.- Desarrollo de una aplicación web

Finalmente y como colofón del proyecto se diseñará una aplicación que permita al usuario cliente, con su ordenador personal y de manera más cuidadosa y exhaustiva que con la aplicación móvil, un análisis completo de los datos recogidos a lo largo del trayecto del globo de alta altitud, para la ayuda en el proceso, la aplicación contará con distintas gráficas que darán un fiel reflejo de los datos captados y permitirán la comparación de distintos ficheros de datos, además de numerosas maneras de exportación de datos.

3. Estudios y análisis previos

Como se ha mencionado anteriormente, es muy extendida la práctica de la captación de datos de diversa naturaleza en distintos niveles de altitud para el estudio de la estratosfera, la gran problemática de esta práctica es el coste de los productos destinados para ello.

He realizado un pequeña investigación previa al desarrollo y es rotundamente cierto que el coste de los instrumentos para realizar estos estudios es bastante elevado, un buen ejemplo para demostrarlo es la página 'https://www.stratoflights.com' muy conocida en el mundo de la astronomía debido a sus productos para el estudio del espacio aéreo, el rango de los kits de globos que captan datos similares a los que se obtendrán en este proyecto van del rango de los 500 a los 2000 euros, en contraposición, nuestro proyecto tiene la ambición de no superar el límite de los 120 euros.

Además, solamente unos pocos kits de los anteriormente citados disponen de aplicaciones para el posterior análisis de datos y en caso de existir, son aplicaciones de poca adaptación a los requerimientos del usuario, por esto mismo, fue de gran ayuda el hecho de contar con la participación de Enrique para transmitirme el feedback de lo que buscaba realmente en una aplicación de estas características y poder desarrollar aplicaciones a partir de sus requerimientos.

3.1.-Alternativas Tecnológicas

Después de tener definido el objetivo del proyecto, tocaba valorar las posibles alternativas para poder alcanzarlo sin superar coste y evitando perder funcionalidades en el proceso. Las alternativas tecnológicas se desglosaron en los siguientes apartados.

3.1.1.- Desarrollo del código del microcontrolador

En este apartado se busca un lenguaje que esté bastante adaptado a dispositivos tipo Arduino, enriquecido con bibliotecas que nos permitan manejar los distintos sensores y cuya complejidad de programación no sea muy elevada, las opciones a barajar fueron las siguientes:

- **MicroPython**: Lenguaje de desarrollo que supone una adaptación de python3 para la programación orientada a microcontroladores tales como Arduino, ESP32… El bajo nivel de abstracción que proporciona este lenguaje permite al desarrollador sacar provecho de las funcionalidades disponibles de la mayoría de los dispositivos electrónicos.
- **C**: Fue el primer lenguaje que se adaptó ligeramente para el desarrollo de programas orientados a microcontroladores, proporciona al desarrollador instrucciones maquina optimizadas para su finalidad, optimizando el rendimiento.
- **C++**: Supone el lenguaje de uso más predominante hoy en día en cuanto a la programación orientada a microcontroladores, aporta un diseño de programación orientada a objetos a diferencia del lenguaje C, flexibilizando la solución.

Finalmente, se utilizará C++, constituyendo una alternativa entre MicroPhython y C, facilitando el desarrollo y no comprometiendo el desarrollo.

3.1.2.- Desarrollo de la aplicación móvil

La aplicación que se desarrollará para móvil será sencilla y por lo tanto no demandará tantos requisitos iniciales como la web, a pesar de ello, se valoraron de nuevo tres posibles alternativas, buscando la opción que mejor se amoldase a nuestro objetivo:

- **Java**: Es un lenguaje de programación ampliamente utilizado en el desarrollo de aplicaciones móviles destinadas a la plataforma Android. Además, ofrece una amplia gama de bibliotecas y herramientas para construir aplicaciones robustas. Java permite

implementar el patrón de diseño MVC para desarrollar aplicaciones móviles estructuradas y escalables.

- **C#:** Es un lenguaje de programación versátil y potente para el desarrollo de aplicaciones móviles en la plataforma de Microsoft, especialmente para Windows y Windows Phone. C# ofrece una sintaxis similar a Java y es compatible con el paradigma de programación orientada a objetos.
- **Kotlin**: Se trata de un lenguaje de programación moderno y conciso, diseñado para el desarrollo de aplicaciones móviles en la plataforma Android. Su gran punto a favor es su compatibilidad pudiendo ser utilizado junto con el código existente de Java. Kotlin ofrece una sintaxis más legible y reduce la cantidad de código necesario en comparación con Java. También incluye características como la inferencia de tipos, nulabilidad segura y funciones de extensión, lo que facilita el desarrollo de aplicaciones más seguras y eficientes.

Finalmente nos decantaremos por Kotlin, se trata de un lenguaje de programación en auge, más moderno y la tendencia se inclina a ser el sucesor de java en el desarrollo de aplicaciones móviles. Google lo posiciona como la opción para el desarrollo de aplicaciones Android.

3.1.3.- Desarrollo de la aplicación de web

La aplicación web tendrá una combinación de BackEnd + FrontEnd, en esta sección entraremos a valorar las posibles alternativas de FrontEnd exclusivamente, pues el BackEnd inicialmente seleccionado sin apenas discusión fue Node.Js, debido a su versatilidad y adaptación, no tuvo alternativas que lograsen ser equivalentes. Para el FrontEnd, sin embargo, se realizó un estudio de posibles alternativas:

- **Node + Angular**: Angular es un framework de desarrollo web basado en TypeScript que permite crear interfaces de usuario ricas y dinámicas. Al utilizar Node.js como backend y Angular como Frontend, es posible construir aplicaciones web con una arquitectura cliente-servidor robusta y altamente interactiva.

- **Node + Vue**: Vue constituye un framework de desarrollo Frontend que permite construir interfaces de usuario modernas, junto a Node.js como backend se obtiene una arquitectura cliente-servidor eficiente y flexible.
- **Node + React**: React supone un FrontEnd muy potente, ofreciendo una gran cantidad de bibliotecas y herramientas disponibles, una comunidad de desarrollo activa y una curva de aprendizaje suave.

Utilizaremos React + Node.js, en gran parte, por su gran funcionamiento totalmente reactivo ante el relativamente bajo volumen de datos esperado en el proyecto y por el gran surtido de bibliotecas que proporcionarán implementaciones de graficas interesantes para nuestro estudio de los datos.

3.1.4.- Sistema de almacenamiento de mensajes Mqtt en AWS

En primer lugar, reseñar que se usará la plataforma AWS en concreto, usaremos la plataforma IOT Core. Esta plataforma constituye un servicio en la nube que proporciona una plataforma robusta y escalable para implementar y administrar soluciones IOT, lo que permite desarrollar aplicaciones y casos de uso innovadores en el ámbito del Internet de las cosas.

El túnel para transferir datos en tiempo real entre el dispositivo microcontrolador y la aplicación móvil será el protocolo MQTT usando como base el formato JSON para codificar la información. MQTT en AWS permite que un dispositivo se comunique de manera transparente con la nube, con total independencia de la infraestructura de red utilizada para las comunicaciones.

Cabe reseñar que el protocolo MQTT es una solución muy común para conexiones bajo CG-NAT, en la que múltiples usuarios comparten una dirección IP pública mediante la traducción de direcciones IP privadas y puertos. Esta técnica se utiliza para mitigar la escasez de direcciones IPv4.

Antes de establecer el protocolo anteriormente citado, deberemos valorar un sistema de almacenamiento modo 'pila' de los mismos. Se valorarán servicios de almacenamiento de la plataforma AWS:

- **Amazon S3**: Se trata de un servicio de almacenamiento en la nube altamente escalable y duradero que también se puede utilizar para guardar mensajes MQTT. Proporciona una forma segura y confiable de almacenar y recuperar grandes volúmenes de datos, incluyendo mensajes MQTT. Para más inri, ofrece una estructura de almacenamiento basada en objetos y es compatible con múltiples formatos de datos.

- **Amazon Kinesis Data**: Servicio de streaming de datos de alta velocidad, puede ser utilizado para almacenar y procesar mensajes MQTT. Proporciona una plataforma escalable y duradera para el almacenamiento de datos en tiempo real. Kinesis Data Streams permite el procesamiento en tiempo real de los mensajes MQTT, lo que permite acciones inmediatas y análisis en tiempo real.
- **Amazon DynamoDB**: Constituye una base de datos NoSQL escalable y administrada que es ideal para almacenar mensajes MQTT. Ofrece un rendimiento rápido y predecible, así como una alta disponibilidad y durabilidad. Su modelo de datos flexible permite un almacenamiento eficiente de los mensajes y su escalabilidad automática garantiza un manejo eficiente de grandes volúmenes de mensajes MQTT.

La elección será Amazon DynamoDB, es una opción segura, muchos desarrolladores, como es el caso de mi tutor Rubén Muñiz Sánchez, se encuentran a menudo en esta misma valoración de alternativas al utilizar el protocolo Mqtt para múltiples casos como la domótica, para estos sectores de desarrollo, DynamoDB se adapta de manera muy natural debido a su escalado automático.

3.1.5.- Base de datos aplicación web

La carga de datos esperada por la aplicación no es alta, este hecho nos permite valorar las alternativas de base de datos para la aplicación web de manera más superficial y elegir

una alternativa, en función de lo familiarizado que nos encontremos con ese gestor de base de datos.

- **MongoDB**: Es una base de datos NoSQL ampliamente utilizada y escalable que se integra bien con Node.js. Es especialmente útil si resulta necesario trabajar con datos no estructurados o semi estructurados.
- **SQLite**: SQLite es una base de datos relacional ligera y sin servidor que se puede utilizar fácilmente en aplicaciones web en Node.js. Al ser una base de datos de archivos, no requiere una configuración de servidor separada y se puede integrar fácilmente en

  una aplicación Node.js sin necesidad de dependencias adicionales. Esta será la más adecuada para su uso.

SQLite parece la más adecuada para nuestros requerimientos, resulta interesante que sea relacional para explotar el almacenamiento y extracción de datos al máximo posible optimizando la gestión.

3.2.- Metodología

La metodología usada en este proyecto ha sido la popularmente conocida metodología Scrum, supone una práctica ágil fácilmente adaptable al desarrollo de un proyecto software como el que estamos abordando, debido a la colaboración, adaptabilidad y la entrega incremental del mismo. Permite proporcionar un marco de trabajo flexible que se centra en la entrega de valor de manera constante a lo largo del proyecto (Sprints).

Se realizarán de manera consensuada entre tutores y alumno un sprint de revisión de manera casi semanal, en el que alumno mostrará a los distintos roles presentes en la reunión el desarrollo realizado y posibles inconvenientes encontrados desde el ultimo sprint realizado.

De manera paralela y cuando sea necesario, habrá ciertos momentos en los sprints destinados a añadir requisitos de usuarios a la pila de producción, otorgándoles una prioridad en la pila a cada uno de ellos.

Las principales figuras presentes a lo largo del desarrollo del proyecto serán las siguientes:

- **Product Owner (Dueño del Producto)**: El responsable de representar a los interesados y clientes del proyecto. Su principal función es definir y priorizar los requisitos del producto, en este caso representado por Enrique Díez Alonso.
- **Scrum Master**: Es el facilitador y líder del equipo Scrum. Su rol principal es asegurarse de que se sigan las prácticas y principios de Scrum, eliminar cualquier obstáculo que afecte al equipo y promover un entorno de trabajo colaborativo. Personificado por Rubén Muñiz Sánchez.
- **Equipo de Desarrollo**: Está formado por los profesionales encargados de llevar a cabo el trabajo necesario para entregar el producto, la cual supone la parte del proyectante.

3.3.- Repositorio

La plataforma de GitHub se ha usado como repositorio oficial a lo largo del proyecto, esta plataforma ofrece una serie de características que pueden ayudar a facilitar la implementación de la metodología Scrum, desde la gestión de tareas y control de versiones, hasta la colaboración y comunicación entre los miembros del equipo.

Además, con el correo de la universidad contaremos con la versión pro en esta plataforma, solo aporta herramientas y estadísticas que serán irrelevantes para este proyecto, pero no deja de ser un aliciente más para su uso.

Para la entrega final, debido a los 300Mb que suponían la entrega total de código fuente, instaladores y demás, se ha realizado en Google drive, ya que GitHub no admitía debido al almacenamiento juntar los distintos entregables.

4. Planificación

El proyecto se ha dividido en ocho sprints de mínimo una semana y máximo tres semanas, se ha intentado optimizar los plazos para poder presentar el proyecto en Julio de 2023, empezando con el mismo la primera semana de febrero. Los objetivos de cada sprint se desglosarán a continuación.

	Task Name	Duration	Start	Finish
1	Sprint 1: Presentación de los requirimientos básicos y objetivos	7 days	Wed 01/02/23	Thu 09/02/23
2	Sprint 2: Definición de arquitectura hardware necesaria	14 days	Fri 10/02/23	Wed 01/03/23
3	Sprint 3: Software principal ejecutado en el ESP32	21 days	Thu 02/03/23	Thu 30/03/23
4	Sprint 4: Requisitos y objetivos de las aplicaciones (móvil y escritorio)	7 days	Thu 23/03/23	Fri 31/03/23
5	Sprint 5: Desarrollo de aplicación móvil y evaluación de la misma	14 days	Mon 03/04/23	Thu 20/04/23
6	Sprint 6: Desarrollo de la aplicación de escritorio y evaluación de la misma	14 days	Fri 21/04/23	Wed 10/05/23
7	Sprint 7: Protocolos MQTT aws y DynamoBd	21 days	Thu 11/05/23	Thu 08/06/23
8	Sprint 8: Recogida de datos y analidis de los mismos	7 days	Fri 09/06/23	Mon 19/06/23

Figura 4.1.- Diagrama de Gantt

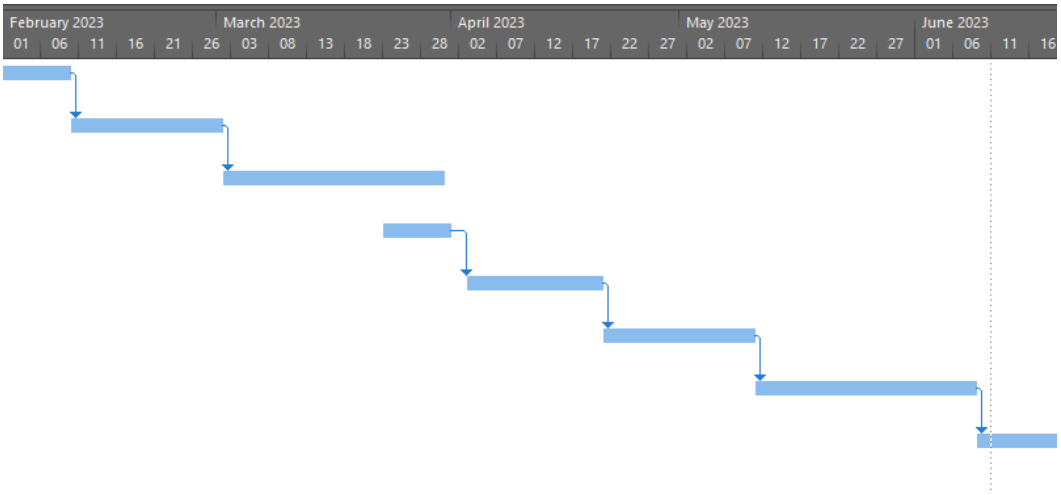


Figura 4.2.- Diagrama de Gantt

4.1.- Sprint 1: Presentación de los requerimientos básicos y objetivos

- Fecha de inicio: 10/02/2023
- Fecha de finalización: 1/03/2023
- Duración: 14 días.

Sprint inicial, en el cual tutor y cotutor me han trasladado las premisas principales del proyecto y las pretensiones del mismo. Se trató de un sprint corto de siete días, pero esencial como base para entender de mejor forma los sprints consecuentes, por eso se ha destinado una semana entera a la asimilación y coordinación de las bases del proyecto.

4.2.- Sprint 2: Definición de la arquitectura hardware necesaria

- Fecha de inicio: 1/02/2023
- Fecha de finalización: 9/02/2023
- Duración: 7 días.

Un sprint muy desapercibido en un proyecto informático, cuyo énfasis principal es el software, pero realmente importante en este proyecto en concreto. La captación de datos muy específicos implica un diseño hardware adaptativo a la recopilación exacta y inequívoca de los mismos. Se dedicaron dos semanas a la valoración y al consensuado a cerca de los mismos.

4.3.- Sprint 3: Definición del software ejecutado en el ESP32

- Fecha de inicio: 2/03/2023
- Fecha de finalización: 30/03/2023
- Duración: 21 días.

Tercer sprint global y el pionero en abordar en profundidad sobre el apartado software del proyecto. El software que implementará el microprocesador supone la base del proyecto, tres semanas dedicadas a su desglosamiento, no se quería avanzar al siguiente sprint con líneas de código erráticas que detuviesen el avance del proyecto.

4.4.- Sprint 4: Requisitos y objetivos de las aplicaciones (móvil y web)

- Fecha de inicio: 23/03/2023
- Fecha de finalización: 31/03/2023
- Duración: 7 días.

Realizado en paralelo respecto a la última semana del sprint previo, se presentó en este sprint los objetivos mínimos esperados de las aplicaciones, tanto de la aplicación móvil, como de la aplicación web, argumentando los posibles lenguajes de desarrollo y entornos de programación.

4.5.- Sprint 5: Desarrollo de aplicación móvil y evaluación de la misma

- Fecha de inicio: 03/04/2023
- Fecha de finalización: 20/04/2023
- Duración: 14 días.

Dos semanas destinadas para este quinto sprint destinado al desarrollo de la aplicación móvil y su posterior prueba, no se deseaba encontrar imprevistos en las aplicaciones el día del test, en especial en esta aplicación que se encargará del track de datos captados en directo.

4.6.- Sprint 6: Desarrollo de la aplicación web y evaluación de la misma

- Fecha de inicio: 21/04/2023
- Fecha de finalización: 10/05/2023
- Duración: 14 días.

Similar al previo sprint tanto en tiempo como en planificación, pero orientado a el desarrollo y evaluación de la aplicación web.

4.7.- Sprint 7: Protocolos Mqtt AWS y DynamoBd

- Fecha de inicio: 11/05/2023
- Fecha de finalización: 08/06/2023
- Duración: 21 días.

En este séptimo sprint se trabajará sobre el protocolo de mensajería Mqtt de la plataforma AWS y el almacenamiento de los mismos en la base de datos DynamoBd, una vez realizado, se estudiará el envío de mensajes Mqtt que contendrán los datos captados desde el microprocesador y finalmente, creando una estructura circular, la publicación de una API que contendrá los datos almacenados en DynamoBd y será la consultada por la aplicación móvil.

4.8.- Sprint 8: Recogida de datos y análisis de los mismos

- Fecha de inicio: 09/06/2023
- Fecha de finalización: 19/06/2023
- Duración: 7 días.

Sprint final, culmen de todos los sprints previos, se realizará el test que consistirá en la captación de datos del nivel del mar a los 2000 metros de altitud, se evaluará si los datos recogidos son válidos, en primera instancia, mediante el track en el dispositivo móvil y posteriormente, se analizarán detenidamente con la aplicación web.

5. Coste

Desglosaremos el coste del proyecto en tres apartados, coste software y coste hardware y recursos humanos.

5.1.- Hardware

En el coste hardware hay que considerar tanto los requisitos hardware del proyecto, como los distintos datos de diversa naturaleza a los que se quiere tener acceso mediante sensores periféricos.

Concepto	**Cantidad**	**Precio Por Unidad**	**Precio Total**
Batería externa / Fuente de alimentación	1 unidad	12 euros	12 euros
Microprocesador ESP32	1 unidad	11 euros	11 euros
Sensor de temperatura	2 unidades	3,29 euros	6,58 euros
Sensor de incidencia de rayos UV	1 unidad	16,99 euros	16,99 euros
Sensor de presión barométrica	1 unidad	11,49 euros	11,49 euros
Sensor de humedad	1 unidad	5,99 euros	5,99 euros
SD Reader	1 unidad	3,78 euros	3,78 euros
Concepto	**Cantidad**	**Precio Por Unidad**	**Precio Total**

Cables / Resistencias / Otros	No cuantitativo	15 euros	15 euros
PC Lenovo para el desarrollo	1 unidad	1500 euros	1500 euros
Periféricos PC	No cuantitativo	300 euros	300 euros

Tabla 5.1.- Desglose coste hardware

Coste hardware sin depreciación: 1882.83 euros

Ahora se deberá calcular el coste real del hardware teniendo en cuenta la depreciación del mismo durante la duración del proyecto. Estimando una vida del hardware total de 4 años, o 48 meses en su defecto, por otro lado, la duración del proyecto ha sido de 5 meses, el coste real en términos de depreciación es:

Coste hardware con depreciación = 1882.83 euros x (5 meses / 48 meses) = 196.12 euros.

Coste hardware con depreciación: 196.12 euros

5.2.-Software

Los entornos desarrollo software que usaremos, son open source y por lo tanto su coste es nulo, aun así, los he reflejado para hacer inciso en lo accesible que es para un estudiante este tipo de plataformas de desarrollo software

Licencias de desarrollo	**Precio / Mensual**
Platform.Io	0 euros

Android Studio	0 euros
VsCode	0 euros
Licencia SO Windows Home	145 euros
Licencia Office 365 2019	149 euros
Licencia Microsoft Project 2019	929 euros

Tabla 5.2.- Desglose coste Software

Coste Software sin depreciación: 1223 euros

Si aplicamos la depreciación del software con exactamente los mismos parámetros que en el apartado anterior, nos quedaría:

Coste software con depreciación = 1223 euros x (5 meses / 48 meses) = 127.39 euros.

Coste software con depreciación: 127.39 euros

5.3.-Recursos humanos

En esta sección se tendrán en cuenta las horas tanto del estudiante como las horas de los tutores pertenecientes a la universidad.

Concepto	**Cantidad**	**Coste por hora**	**Coste total**
Horas de estudiante	300	25.00 euros	7500 euros
Concepto	**Cantidad**	**Coste por hora**	**Coste total**

Horas tutores de la universidad	30	30 euros	900 euros

Tabla 5.3.- Desglose coste Software

Coste total recursos humanos: 8400 euros

5.4.-Total

Ahora sumaremos el total de los tres costes para obtener el coste total del proyecto:

Concepto	**Coste**
Hardware	196.12 euros
Software	127.39 euros
Recursos humanos	8400 euros

Tabla 5.4.- Desglose coste Total

Coste Total: 8723.51 euros

6. Requisitos de usuario

Los requisitos de usuario también conocidos como requisitos de alto nivel o requisitos del cliente, son declaraciones generales de lo que el sistema debe lograr desde la perspectiva del usuario o del cliente. Estos requisitos se centran en las necesidades, expectativas y objetivos de los usuarios finales del sistema.

La lista de requisitos de usuario ha sido definida a lo largo de los ocho sprints realizados, en los cuales los tutores, sobre todo Enrique Díez Alonso ha intervenido como cliente.

Listado de requisitos de usuario:

ID Requisito de Usuario	Descripción
R.U.1	El peso del microcontrolador principal más la suma de los periféricos no debe superar los tres kilos. No debe superar ese taraje para no dificultar el vuelo y probar impedimentos en llegar a la altitud o provocar una desviación de la trayectoria.
R.U.2	Se quiere captar temperatura, humedad, presión barométrica e incidencia de rayos UV de manera precisa y sin que los factores externos generados por la altitud interfieran en el proceso.
R.U.3	Se quiere comprobar de manera remota desde un dispositivo móvil los datos captados pues el cliente quiere detectar posibles incidencias “on time”

ID Requisito de Usuario	Descripción
R.U.4	Se quieren representar los datos en algún tipo de gráfico general, para corroborar que los datos se captaron de manera adecuada y poder compararlos con otros datos de la misma naturaleza.

Tabla 6.1.- Requisitos de usuario

7. Requisitos de Sistema

Los requisitos de sistema se desglosarán en requisitos funciones y requisitos no funcionales.

7.1.-Lista de requisitos funcionales

Estos requisitos referencian las funciones y las acciones específicas que debe realizar el sistema para cumplir con los requisitos de usuario. Describen las interacciones del sistema con los usuarios y otros sistemas, definiendo las entradas, las salidas y el comportamiento esperado del sistema en diferentes situaciones.

Listado de requisitos funcionales:

ID Requisito Funcional	**Descripción**	**ID requisitos de usuario afectados**
R.F.1	Se utilizará un dispositivo microcontrolador ESP32 debido a la gran capacidad de procesamiento que provee con sus dos núcleos de procesador.	R.U.1 R.U.2
R.F.2	Se utilizarán periféricos de Arduino/ESP 32 como sensores de captación integrados en los pines del microprocesador para no exceder el peso máximo.	R.U.1 R.U.2

ID Requisito Funcional	Descripción	ID requisitos de usuario afectados
R.F.3	Se utilizarán servicios AWS, mensajería MQTT y una API en conjunto para que la aplicación de móvil tenga un acceso casi inmediato a los datos captados por el microprocesador.	R.U.3
R.F.4	La aplicación web importará datos, generará graficas junto a estadísticas básicas como media, desviación típica y permitirá la exportación de los datos en diversos formatos.	R.U.4

Tabla 7.1.- Requisitos funcionales.

7.2.-Lista de requisitos no funcionales

Estos requisitos hacen referencia a características y restricciones del sistema que no están directamente relacionadas con las funciones específicas que realiza, sino con sus atributos de calidad, rendimiento, seguridad, usabilidad y otros aspectos.

ID Requisito No Funcional	Descripción
R.N.F.1	El único punto de acceso a internet del microprocesador y sus periféricos la proporcionará un móvil y una sim incorporada al mismo.
R.N.F.2	El código implementado a lo largo del proyecto debe ser legible, estructurado y documentado.
R.N.F.3	Las interfaces de las aplicaciones deben ser sencillas pero atractivas al mismo tiempo, pudiendo ser usadas por un usuario con poca experiencia tecnológica, pero resultar agradables a la vista.
R.N.F.4	El track de los datos en directo en la aplicación móvil debe tener menos de un minuto de delay entre su captación por los sensores periféricos y su correspondiente representación en el móvil.

Tabla 7.2.- Requisitos no funcionales

8. Diseño

Como se ha citado en las anteriores secciones, el diseño del proyecto se desglosará en tres módulos que funcionarán de forma conjunta o al menos funcionarán de forma complementaria, aportando distintas funcionalidades, estos tres módulos son los siguientes:

- Microprocesador y sensores periféricos
- Aplicación de móvil
- Aplicación web

8.1.- Microprocesador y sensores periféricos

8.1.1.-Configuración hardware

A pesar de ser un proyecto cuyo núcleo fundamental es el desarrollo software, el diseño hardware de este módulo es de vital importancia, a continuación, analizaremos en profundidad el hardware utilizado y sus especificaciones:

Modulo ESP32 AZ DELIVERY

Microprocesador elegido, entre sus especificaciones y puntos fuertes destaca:

- Integra funciones Wi-Fi y Bluetooth, esencial para mantener una conexión Wi-Fi con nuestro móvil Android que actuará de enlace para proveer internet al dispositivo.
- Consumo ultra bajo de energía, aspecto muy favorable para realizar el test de campo, no queremos que el microprocesador agote la batería de la fuente de energía.
- Dispondremos de 38 pines, para ser utilizados como conexión a los diferentes periféricos a los que le daremos uso.
- Proporciona pines de fuente a 5v y 3.3v según las necesidades de los periféricos.

- El microcontrolador ESP32 dispone de dos núcleos, que pueden ser utilizados haciendo uso de las primitivas proporcionadas por el sistema operativo FreeRTOS, el cual ha ganado popularidad debido a su facilidad de uso, su enfoque en la eficiencia de recursos y su amplia comunidad de usuarios y desarrolladores que brindan soporte y contribuyen con mejoras continuas. Esto permite la obtención de un código capaz de ejecutar tareas en paralelo, tales como adquisición de datos y comunicaciones.

Figura 8.1.1.- Modulo ESP32

Placa de expansión de almacenamiento microSD

Proporciona fácil inserción y extracción de la tarjeta microSD. Configuración actual de pines con el ESP32:

Pin SD Reader	**GND**	**VCC**	**MISO**	**MOSI**	**SCK**	**CS**
Pin ESP32	GND	5V	G19	G18	G23	G5

Tabla 8.1.1.- Disposición pines placa de almacenamiento microSD

Se utilizará una tarjeta MicroSD de 8GB formateada a FAT32 para la correcta gestión del almacenamiento, MicroSD proporciona una tecnología de inserción de datos lo suficientemente rápida para nuestros requerimientos.

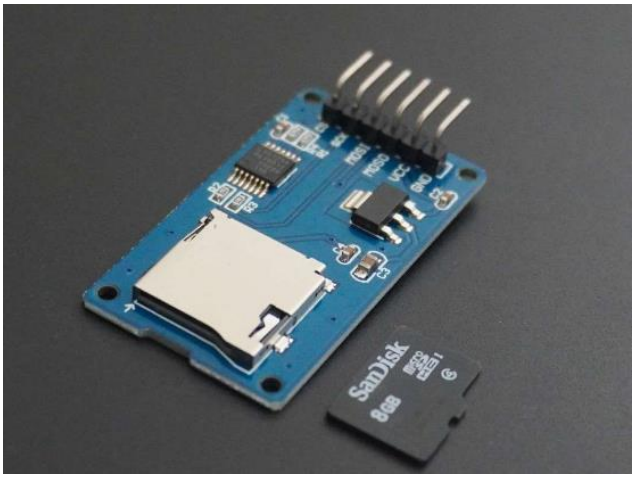


Figura 8.1.1.- Modulo Placa de expansión y tarjeta SD

Sensor de temperatura DS18B20

Sensor protegido por un tubo térmico, previniendo cortocircuitos y aislando el chip de otras incidencias similares como el agua.

Rango de medición de temperatura es de -55 ° C a +125 ° C con una precisión de +- 0.5° C, por lo tanto, la instalación de este sensor nos da garantías de que la temperatura obtenida es fiable.

Configuración actual de pines con el ESP32:

DS18B20	**VCC**	**GND**	**DATA**
ESP32	5V	GND	G32

Tabla 8.1.1.- Disposición pines sensor de temperatura

Figura 8.1.1.- Sensor de temperatura DS18B20

Uv-sensor-ml8511

Tiene la capacidad de convertir la fotocorriente en tensión para detectar la intensidad UV, detectando un rango de longitudes de onda de 280-390nm. El módulo tiene un modo de consumo ultra bajo, permitiendo ahorrar batería a un dispositivo portátil como nuestro modulo Esp32, además el peso del dispositivo es de apenas 2 gramos.

Configuración actual de pines con el ESP32:

ml8511	**VIN**	**VCC**	**GND**	**OUT**	**EN**
ESP32	NaN	3V3	GND	G34	Nan

Tabla 8.1.1.- Disposición pines Uv-sensor-ml8511

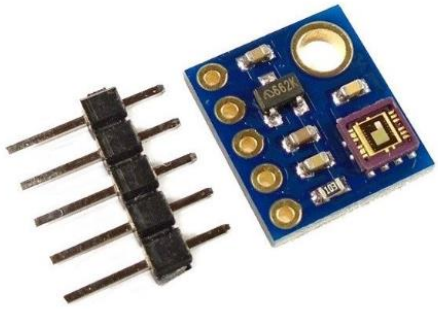


Figura 8.1.1.- Uv-sensor-ml8511

DHT 22 Sensor de humedad

Este sensor puede funcionar tanto con 3,3V como con 5V, aunque en nuestro caso es indistinto debido a que el ESP32 como ya hemos dicho dispone de los dos tipos de alimentación, puede llegar a medir la presencia de humedad relativa en porcentaje, en un rango de 0 a 100 con precisión de +-2%.

Configuración actual de pines con el ESP32:

Dht22	**VCC**	**Data**	**Nan**	**GND**
ESP32	3v3	G26	Nan	GND

Tabla 8.1.1.- Disposición pines DHT 22 Sensor de humedad

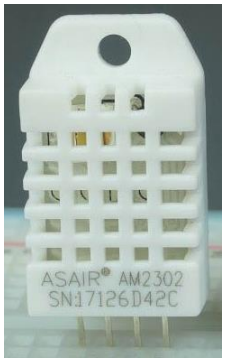


Figura 8.1.1.- DHT 22 Sensor de humedad

Bmp180 Sensor de presión barométrica

Sensor capaz de medir la presión barométrica en un rango de in 300-1100 hPa, con una precisión de +-10hpa, económico respecto al resto de dispositivos de la misma familia, además proporciona un bajo consumo.

Configuración actual de pines con el ESP32:

Dht22	VCC	GND	SCL	SDA
ESP32	3v3	GND	G22	G21

Tabla 8.1.1.- Disposición pines Bmp180

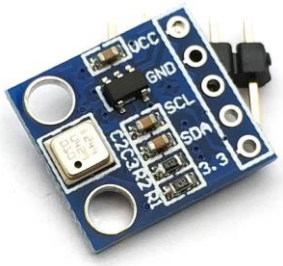


Figura 8.1.1.- Bmp180

Hardware restante necesario para la conexión entre modulo y periféricos

- Placa protoboard arduino

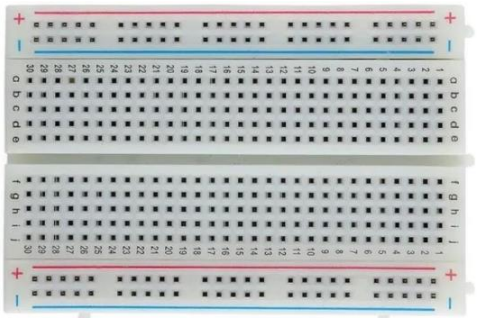


Figura 8.1.1.- Placa protoboard arduino

- Cables macho-macho y cables macho-hembra

Figura 8.1.1.- Cables macho-macho y cables macho-hembra

Disposición hardware del conjunto

Finalmente se presentan fotos reales del hardware previamente visto y siguiendo el esquema de conexiones presente en las respectivas tablas.

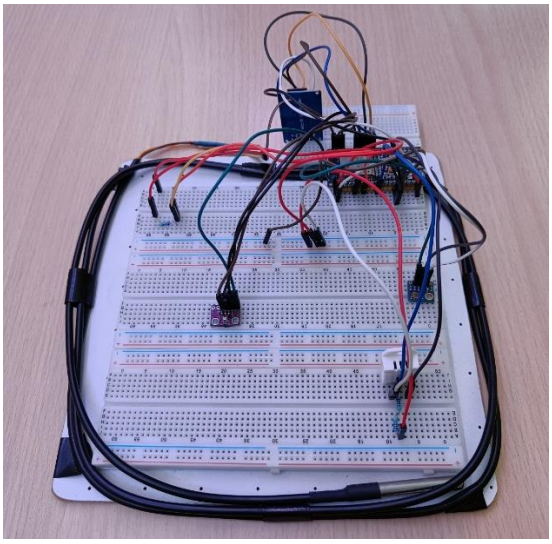


Figura 8.1.1.- Disposición general hardware

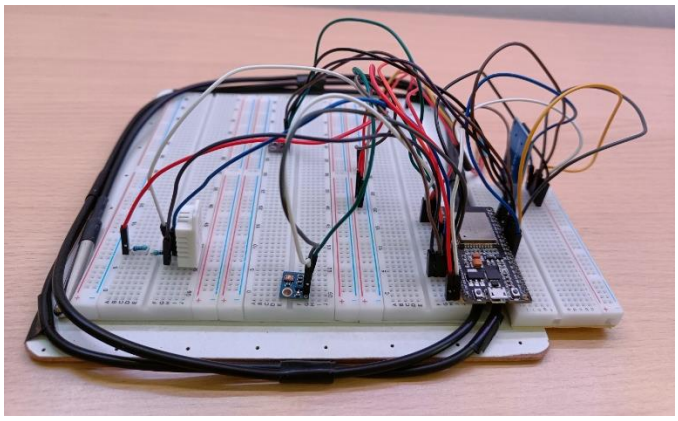


Figura 8.1.1.- Disposición general hardware

8.1.2.-Configuración software

El código destinado al control del microprocesador y la posterior toma de datos está desarrollado en C++, utilizando el entorno de desarrollo Patform.Io, ideal para la programación destinada a microprocesadores.

La estructura del proyecto contenedora del código tiene dos pilares fundamentales, por un lado, un archivo *platform.ini* que contiene las blibliotecas del proyecto y un *main.cpp* que contendrá el código fuente del proyecto.

El código fuente presente en el *main.cpp* cumple la siguiente estructura:

En primera instancia, como es de costumbre en el código C++ la incorporación de las bibliotecas de las cuales se va a hacer uso, con la sintaxis típica de C++ *#include<biblioteca>*, gran parte de las incorporadas son destinadas al manejo de los sensores periféricos de captación de datos.

En segundo lugar, se definirán las constantes, con la consecuente sintaxis *#define<constante>*. A continuación, las variables no constantes, utilizadas en su mayoría para asignarles datos provenientes de sensores.

Posteriormente se definirán los métodos, cuya principal función se explica de manera resumida a continuación:

- *void messageHandler(String &topic, String &payload)*: Encargado de formatear adecuadamente el mensaje enviado mediante protocolo Mqtt.
- *void connectAWS()*: Permite conectarnos a nuestra base de datos DynamoBd de AWS
- *void publishMessage()*: Envía el mensaje con los datos ya integrados mediante el protocolo Mqtt.
- *void appendFile(fs::FS &fs, const char * path, const char * message)*: Añade los datos captados por los sensores periféricos al archivo creado en la tarjeta SD.

- *void logSDCard()*: Prepara el mensaje que va a ser añadido al archivo de texto presente en la tarjeta SD.
- *void writeFile(fs::FS &fs, const char * path, const char * message)*: Escribe en el fichero dentro del almacenamiento de la tarjeta SD insertada, si no existe se crea.
- *void Start_SD()*: Inicializa la tarjeta SD, para poder gestionar su almacenamiento.
- *int averageAnalogRead(int pinToRead)*: Devuelve una media de los valores devueltos por el PinOut Analógico.
- *float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)*: Tranforma los datos de analógicos a digitales.

En última instancia, tendremos el método *setup()*, frecuente en la programación orientada a microprocesadores, como su nombre indica permite la inicialización de objetos y realizar llamadas a métodos que se quieren ejecutar antes de correr el bucle de ejecución principal.

A continuación, en vez de tener un método común *loop()* encargado de ejecutar cíclicamente un proceso, tenemos dos "loops". Esta estructura permite explotar la oportunidad que nos brinda el módulo Esp32 al tener incorporados dos núcleos, cada loop se ejecutará en un núcleo diferente. Un bucle es destinado a la captación de datos provenientes de los diferentes sensores y el bucle restante será destinado a insertar los datos captados por los periféricos en la base de datos de AWS y a la tarjeta SD del módulo SD reader.

8.1.3.-Conexión del microprocesador con AWS

El microprocesador mantendrá una conexión con los servicios AWS gracias al Wi-Fi que provee el dispositivo móvil. Se ha creado previamente una base de datos no relacional DynamoBD cuya clave de partición (clave principal) es un Id generado con el timestamp del mensaje en segundos, por lo tanto, es único en la tabla. Siguiendo este sencillo proceso y sin falta de realizar configuraciones adicionales, en esta base de datos se pueden empezar a insertar mensajes mediante protocolo Mqtt.

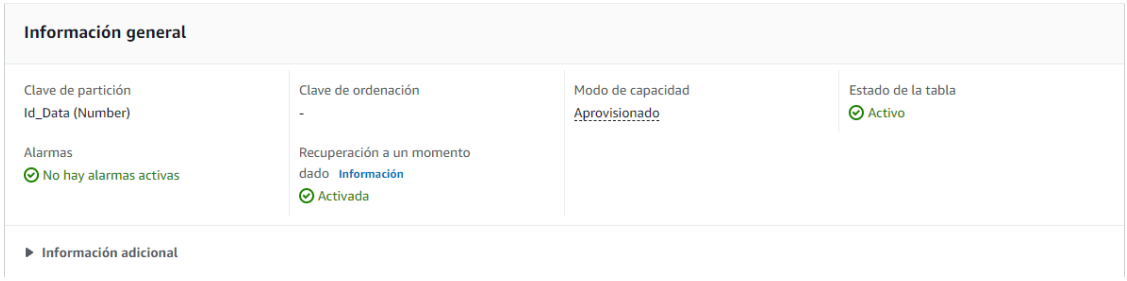


Figura 8.1.3.- Base de datos DynamoBd de AWS

Se ha elegido para nuestra base de datos no relacional un modo de capacidad de espacio aprovisionado, en el que el coste de un uso bajo, como es el de nuestra prueba de vuelo previsto, sale a coste nulo.

Desde el punto de vista del código presente en el microprocesador, los datos que contienen los valores ambientales recogidos por los sensores serán encapsulados en un mensaje ‘parseado’ a JSON y posteriormente serán enviados mediante el protocolo MQTT a la base de datos DynamoBd citada previamente, de modo que si visualizamos los elementos de la Base de datos tendremos el id de partición y un conjunto denominado ‘payload’ que contendrá nuestro mensaje con los datos captador por los sensores.

Figura 8.1.3.- Base de datos DynamoBd de AWS

8.2.-Diseño Aplicación Móvil

8.2.1.-Diseño Software

Como se ha comentado en secciones previas, el lenguaje elegido para su desarrollo ha sido Kotlin y la plataforma para lograrlo será Android Studio, debido a su interoperabilidad la implementación para dispositivos Android y la función de debug, útil para el desarrollo.

8.2.1.1.-Front End o UI

El diseño de la aplicación móvil buscaba la sencillez, facilitando al usuario la comprensión de los datos que tenía a su disposición, por ello, se decidió utilizar la estructura *recyclerview*, idónea para estos casos, ya que se adapta a grandes conjuntos de datos de manera eficiente en una interfaz de usuario.

El *recyclerview* necesitará de los siguientes subcomponentes para funcionar correctamente:

- Adaptador (Adapter): El RecyclerView requiere un adaptador para proporcionar los datos y crear las vistas para cada elemento. El adaptador es responsable de crear nuevas vistas (inflar) cuando sea necesario y también de vincular los datos a cada vista.
- ViewHolder: Un ViewHolder es una clase que contiene las referencias a las vistas individuales dentro del diseño de un elemento de la lista.
- LayoutManager: El LayoutManager es responsable de la disposición y posicionamiento de las vistas dentro del RecyclerView.

Figura 8.2.1.1.- FrontEnd Aplicación móvil

8.2.1.2.-Back End

La principal función de esta aplicación era mostrar casi en tiempo real los datos almacenados en la BBDD no relacional de AWS DynamoBD, por ello, después de una rigurosa investigación me he decantado por la siguiente estructura.

Se utilizará el servicio de AWS para crear una API, con x dirección de acceso, dicha API se actualizará con lo que se conoce clásicamente como ‘disparador o *trigger*’. Este disparador se accionará con cada actualización en la base de datos, bien una inserción o una eliminación de los mismos. La aplicación consultará cíclicamente dicha API en busca de cambios y traerá cuando existan cambios los datos actualizados. Dentro de ella desglosará el texto en formato JSON proveniente de la API para mostrarlo al usuario plasmado en el *Recyclerview* citado en la sección previa.

8.3.-Diseño Aplicación web

La aplicación web se desarrollará en React + Node.JS, y será válida exclusivamente para PCs de sistema operativo Windows. En caso de querer una mayor globalización, se podría cambiar la estructura y hostear en una dirección web desde la que fuese accesible a todos los PCs con distinto tipo de sistema, con la única condición de tener acceso a internet. A pesar de ser valorada esta opción, se descartó debido a la globalización de los sistemas operativos Windows como ordenadores personales.

8.3.1-Front End y flujo de navegación

El Front End de la aplicación diseñado con React.Js tendrá tres pantallas principales, complementadas con "*pop ups*" añadiendo mensajes de carácter informativo para el usuario. Se trata de una aplicación sencilla y agradable, pero sobre todo fácil de usar pudiendo ser manejada sin ningún tipo de problema por un usuario no experto.

En primera instancia tenemos la pantalla de bienvenida.

Figura 8.3.1.- Pantalla de bienvenida aplicación web

Esta pantalla nos dará por un lado la opción de cargar nuevos datos o, por el contrario, si ya se tienen añadidos a la base de datos, se puede pulsar en el botón siguiente y pasar a la representación de los datos.

En caso de añadir un archivo erróneo que no cumpla el formato de cabecera 'Nombre,Id,Temperatura,Humedad,Presión,Luminiscencia' nos saldrá el siguiente mensaje de error informativo.

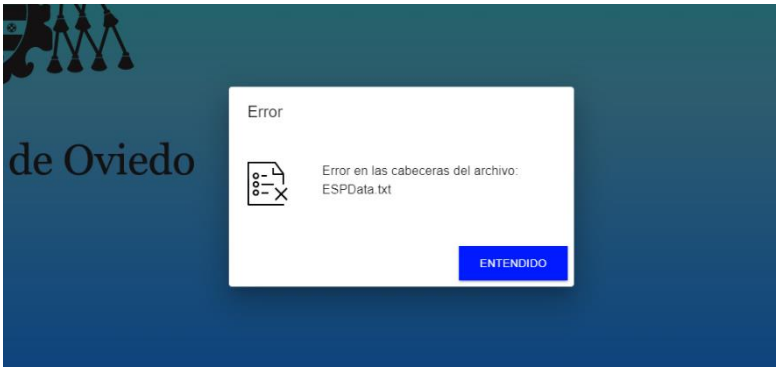


Figura 8.3.1.- Mensaje de error en formato aplicación web

En caso de subir un archivo de formato valido, pasaremos a la siguiente pantalla.

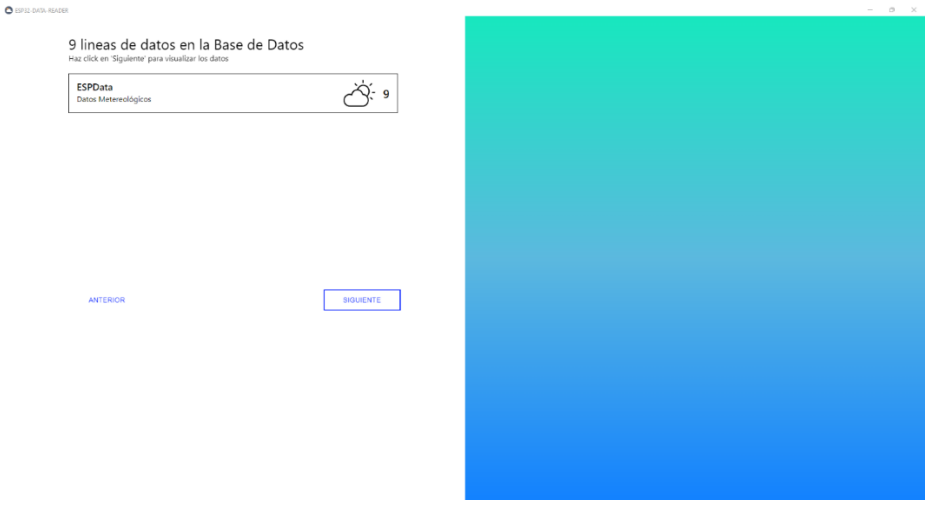


Figura 8.3.1.- Pantalla de visualización líneas de fichero aplicación de web

Esta pantalla tiene la principal función de ser informativa. Nos mostrará un recuadro con cada nombre de archivo existente en la base de datos (no solo los recién importados, sino todos) y las líneas de datos presentes en cada fichero. Si pulsamos en siguiente accederemos a la pantalla final, en caso contrario volveremos a la de bienvenida.

En la pantalla final se tendrán múltiples opciones, las cuales se irán detallando a continuación:

1 Figura 8.3.1.- Pantalla final aplicación web

En primer lugar, empezando de izquierda a derecha, se tiene un menú destinado a diferentes modos de exportación de los datos importados, tres para ser más exactos: El formato texto y Excel creará un archivo .txt y. xslx respectivamente. Se exportará el fichero en un directorio elegido por el usuario mediante una ventana emergente. Finalmente, el modo gráfico actual guardará una imagen del gráfico actual.

En el menú desplegable de ficheros, situado en la parte superior, se podrá elegir entre los distintos ficheros almacenados la base de datos. A la derecha de este componente tendremos el

botón “Cargar Datos” para volver a importar distintos ficheros de datos a los ya almacenados en la base de datos.

En el centro de la pantalla tenemos el gráfico que representa los datos seleccionados. Este gráfico tiene la opción de ser intercambiable, permitiendo al usuario representar los datos en dos modos diferentes, grafico de área sin zoom, y gráfico con media y zoom. Para realizar el cambio entre ambos solo tendremos que hacer clic en el botón “hamburguer” para ello, como muestra la imagen:

Figura 8.3.1.- Elección de gráfico

Figura 8.3.1.- Gráfico de área sin zoom

Figura 8.3.1.- Gráfico de barras con zoom

En la parte inferior de la pantalla tendremos un desplegable con diferentes estadísticas de los datos representados en la gráfica, aportando estadísticas interesantes para el estudio de los datos como pueden ser la media, la varianza y la deviación típica.

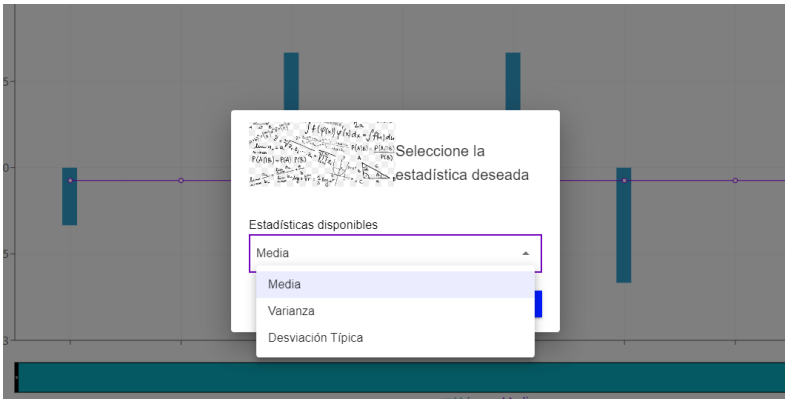


Figura 8.3.1.- Estadísticas para aplicar a los datos representados

Finalmente, en la esquina inferior derecha de la pantalla tenemos el botón para reiniciar la base de datos, eliminando completamente todos los archivos importados, una vez realizada la acción nos devuelve a la pantalla principal para realizar una nueva importación.

8.3.2.- Base de datos aplicación web

La base de datos de la aplicación web consistirá en una base de datos relacional de tipo SQLite, realmente sencilla pues no es necesaria ninguna complejidad.

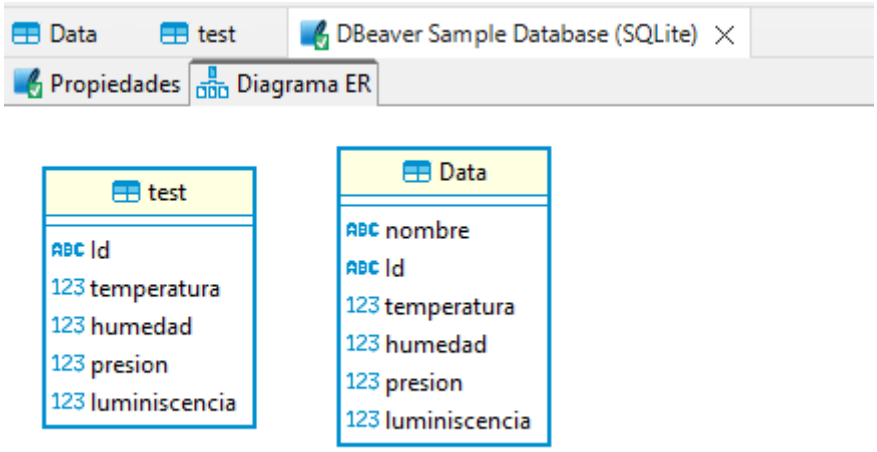


Figura 8.3.2.- Diagrama base de datos Sqlite

En esta sencilla base de datos existirán dos tablas, una con el nombre “Data” contendrá todas las líneas de los ficheros importados, acompañados del respectivo nombre del fichero para las consultas de búsqueda en la base de datos y una tabla temporal llamada “test” que contendrá los datos seleccionados para representar en la gráfica, optimizando su gestión y realizando en menor tiempo las estadísticas correspondientes a estos datos.

El lugar de creación de la base de datos Sqlite en el Pc de instalación de la aplicación será por defecto en la siguiente ruta:

- C:\Users\XXX\ESP32-DATA_Database\ESP32-DATA-READER
  donde XXX= usuario correspondiente.

8.3.3.-Backend

El backend realizado en Node.Js, utilizará código en typescript, más robusto que javascript debido a su obligatoriedad de tipado. El proyecto constará de la siguiente estructura de carpetas:

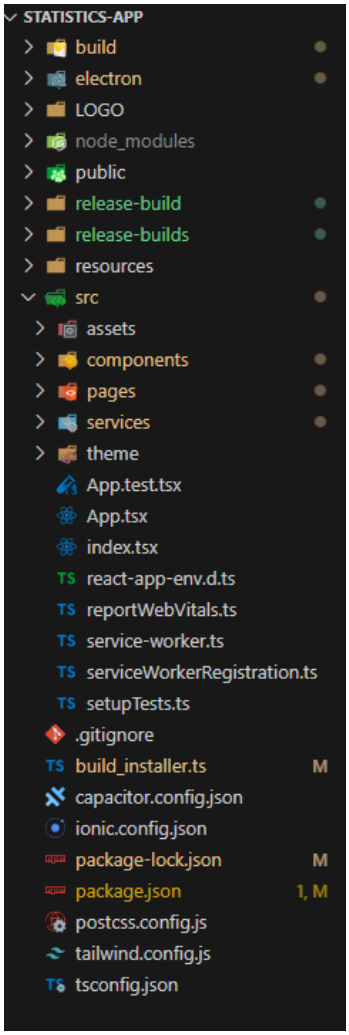


Figura 8.3.3.- Estructura proyecto de aplicación web

Inicialmente se ha generado un proyecto con la herramienta electron, herramienta necesaria para poder generar la aplicación web. Dentro de dicha carpeta tendremos archivos como el setup.ts que permitirán distintas configuraciones de la aplicación como el tamaño de ventana, icono, nombre la aplicación, barra de herramientas incluida… es una herramienta verdaderamente útil y usada en todo tipo de aplicaciones.

En "src" tendremos el proyecto principal. Electron compilará los archivos que tengamos en su interior para generar la aplicación web. En la carpeta "pages" tendremos las tres pantallas principales vistas previamente, desarrolladas en typescript. En la carpeta components, tendremos diversos componentes como cierto estilo de botones usados recurrentemente en diversas pantallas, por lo tanto, es buena práctica unificarlos en la carpeta y poder importarlos a cada pantalla en la que se hacen uso, en vez de repetir código por cada pantalla que hace uso de estos componentes.

La carpeta "services" contendrá los métodos de consulta, inserción, eliminación y modificación de la base de datos. Sus archivos también programados en typescript, contienen consultas que gestionan la base de datos Sqlite.

El resto de los archivos son archivos generados por defecto al iniciar el proyecto, para configuración de la aplicación o control de versiones de las dependencias instaladas en las que no es necesario detallar.

9. Test

9.1.-Definición del test y posterior ejecución

El test consta de un pilar fundamental de cualquier proyecto informático y este no iba a ser menos, en un principio como se ha comentado en los apartados iniciales de la memoria, en el test se iba realizar la recogida de datos mediante la realización de una ruta que nos permitiese ir subiendo de altitud progresivamente y recoger datos meteorológicos x veces a lo largo de la misma para poder evaluarlos.

Finalmente se diseñó una ruta que nos permitiese tomar muestras cada 100 metros de los 0 a 2000 metros de altitud aproximadamente, para poder ver una variación en los datos captados por los diferentes sensores y comprobar su correcto funcionamiento.

La ruta trazada para satisfacer nuestras necesidades previas ha sido la siguiente:

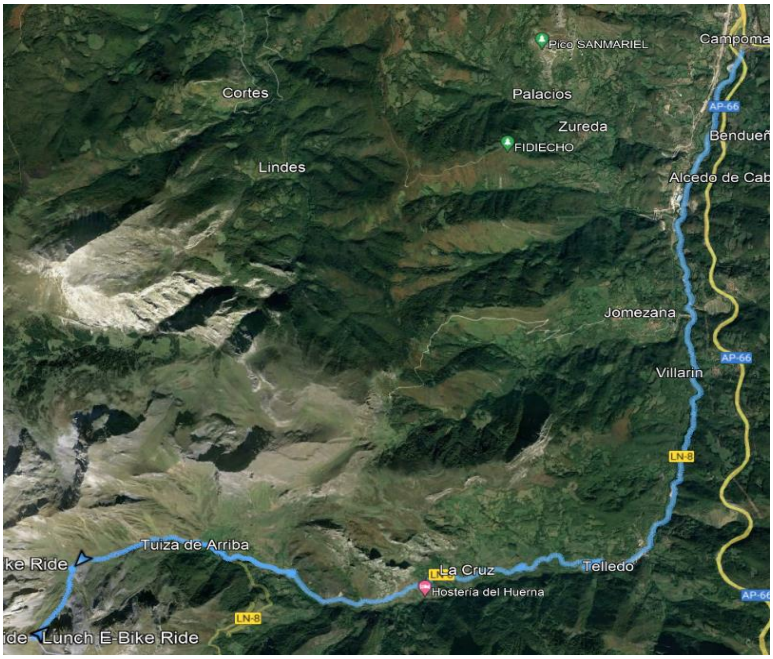


Figura 9.1.- Vista satélite de la ruta

Esta ruta que podemos ver en la parte posterior de la figura 9.1 cubre un rango de altitudes de 400 m a 2000m. Las tomas realizadas de 100 m a 300m fueron tomadas en el viaje de enlace a esta (mismo día), pero debido a la disparidad de las ubicaciones no se recogen en la vista satélite.

Por lo tanto, una vez con la ruta ya planificada, llegó el día de realizarla. El plan era como se ha comentado anteriormente realizar una toma cada 100 metros de altitud aproximadamente hasta alcanzar los 2000 metros, por lo tanto 18 tomas de datos han sido captadas por nuestros sensores.

Para saber la certeza de los datos captados era importante valorar la climatología que reinaba ese día, por ello haremos un breve resumen de la misma:

- De los 100 metros de altitud a los 900 metros la climatología era soleada y casi despejada pero la humedad estaba presente al tratarse una montaña sombreada desde los 500 metros de altitud.

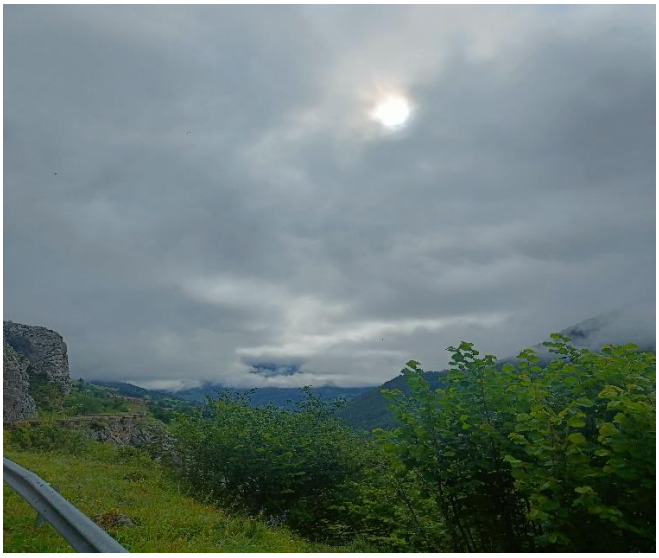


Figura 9.1.- Imagen real a 600 metros de altitud

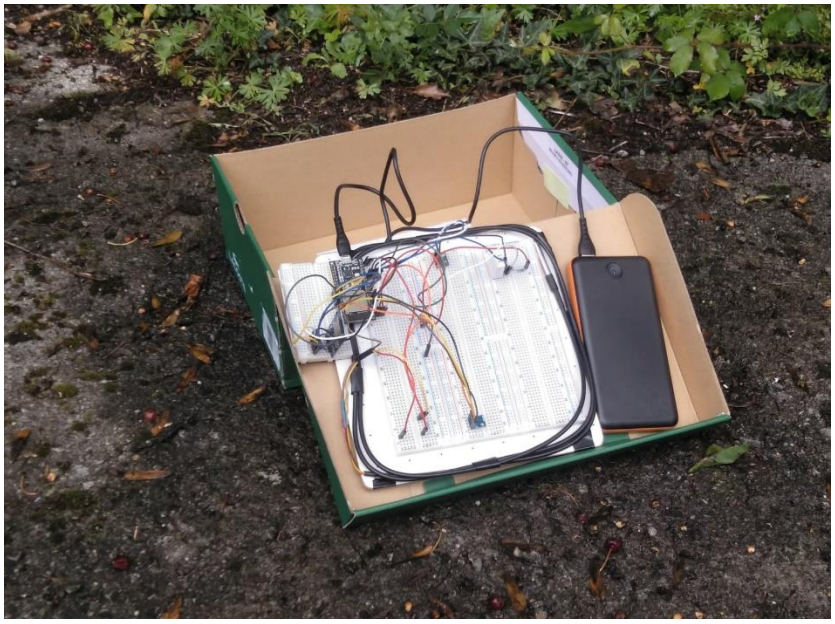
- De los 900m a los 1700 metros de altitud, había una densa niebla incremental según subíamos de altitud, por lo tanto, la temperatura era menor.

2 Figura 9.1.- Imagen real a 1400 metros de altitud

- A partir de los 1700 metros la niebla se superó. El sol y la temperatura elevada volvió a hacerse presente.

Figura 9.1.- Imagen real a 1900 metros de altitud

La protoboard con los sensores fue portada en una caja para no ocasionar daños en las conexiones, finalmente la fuente de alimentación la dio una batería externa y la conexión a internet la proveímos con un dispositivo móvil.



3 Figura 9.1.- Imagen del hardware

La conexión a internet que gozábamos desde los 100 a 900 metros ha permitido tener un track en directo de los datos recogidos en dichas alturas desde la aplicación móvil mediante servicios AWS.

Figura 9.1.- Aplicación móvil

En cada toma realizada en el intervalo de 100 metros obteníamos un minuto justo de datos captados, al tener un delay de 3 segundos en el bucle de almacenamiento de datos, obtuvimos 20 líneas de datos diferentes por cada 100 metros, de esos 20 datos captados por sensor realizamos la media aritmética para quedarnos con una parte representativa de los mismos.

Después de este procesamiento, obtuvimos los siguientes datos:

Altitud	Temperatura	Humedad	Presión	Luminiscencia
100m	21.02ºC	66.50%	1008hpa	0.02mw/m^2
200m	21.74ºC	66.65%	1004hpa	0.12mw/m^2
300m	21.82ºC	66.72%	1001hpa	0.25mw/m^2
400m,	23.02ºC	69.90%	980hpa	0.41mw/m^2
500m	23.22ºC	67.50%	970hpa	0.18mw/m^2
600m	24.40ºC	66.20%	952hpa	0.58mw/m^2
700m	25.44ºC	63.00%	940hpa	0.22mw/m^2
800m	24.25ºC	62.30%	931hpa	0.50mw/m^2
900m	25.06ºC	66%	918hpa	1.14mw/m^2
1000m	24.92ºC	63.50%	910hPa	0.84mw/m^2
1100m	23.50ºC	64.70%	905hPa	0.36mw/m^2
1200m	21.60ºC	65.80%	900hPa	0.15mw/m^2
1300m	20.10ºC	66.90%	895hPa	0.09mw/m^2
1400m	18.80ºC	68.10%	890hPa	0.06mw/m^2
1500m	17.60ºC	69.20%	885hPa	0.04mw/m^2
1600m	16.50ºC	70.40%	880hPa	0.03mw/m^2
1700m	25.06ºC	66.00%	875hPa	0.50mw/m^2

Altitud	Temperatura	Humedad	Presión	Luminiscencia
1800m	26.20ºC	65.10%	870hPa	0.68mw/m^2
1900m	27.30ºC	64.20%	865hPa	0.87mw/m^2

1 Tabla 9.1.- Datos recogidos en el test

Figura 9.1.- Imagen visualizando los datos en tiempo real

9.2.-Análisis de los datos obtenidos

Para realizar el análisis de los datos obtenidos y poder corroborar su fiabilidad usaremos nuestra aplicación web, observando la tendencia que toman los distintos datos según la variable de la altitud se iba incrementando e intentaremos explicar el sentido de los distintos aspectos que tomarán las gráficas representativas de cada uno. Los archivos con los datos estarán disponibles en el drive, en la carpeta del proyecto test.

Ahora como colofón, importaremos las 19 líneas de datos recogidas en un fichero que corresponderán a los 19 puntos de altitud diferentes, ordenadas en orden creciente de metros de altitud. Las gráficas dispondrán una barra de error en color rojo, calculada con la desviación de dichos datos para cada altitud.

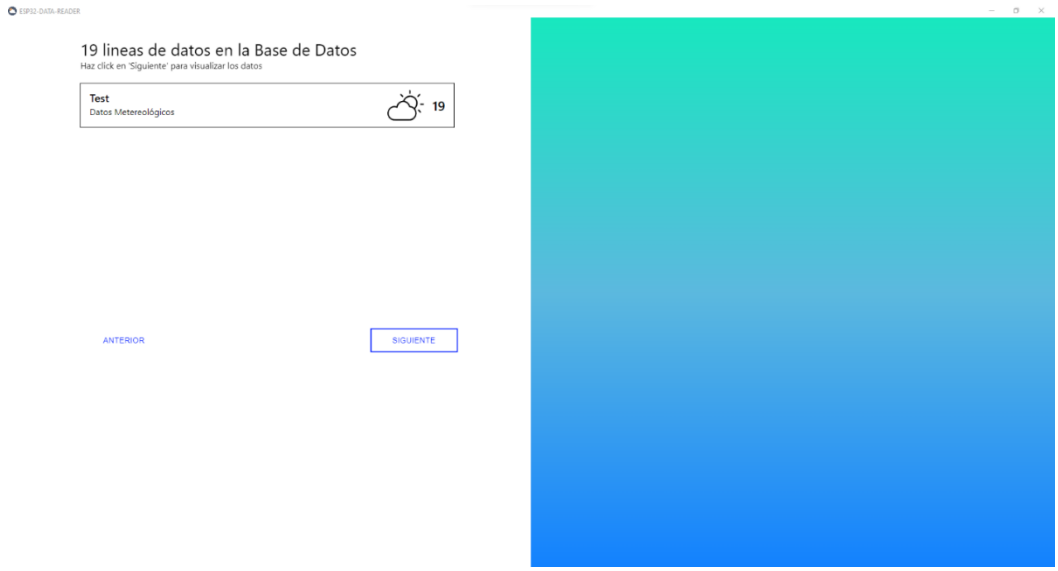


Figura 9.2.- Importación datos test

- **Temperatura**

  Los datos de temperatura a pesar de ser bastante constantes reflejan con fidelidad los efectos de una espesa niebla de los 900 a 1700 metros, siendo los puntos más bajos de temperatura. También influye entre los 100 metros y 400 metros la hora del día próxima a las 9 de la mañana, además de la localización en el fondo del valle, por lo tanto, la temperatura es menor. El error es muy pequeño, aspecto positivo, provocando que la barra de error se convierta en un punto.

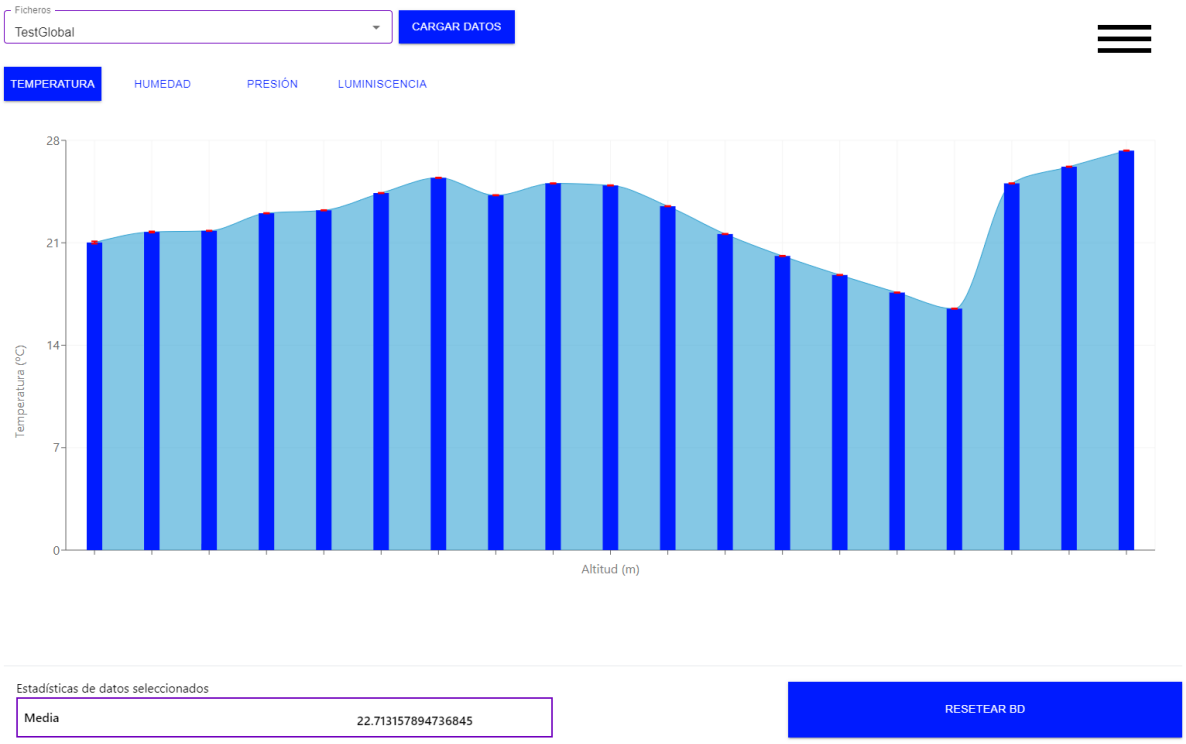


Figura 9.2.- Datos temperatura test global

- **Humedad**

La humedad no contiene gran variación, siendo bastante constante en todos los puntos a la media sin apenas variación, no es un dato que nos pueda proveer información acerca de la altitud en la que nos encontramos. El error vuelve a ser muy pequeño, provocando que la barra de error se convierta en un punto.

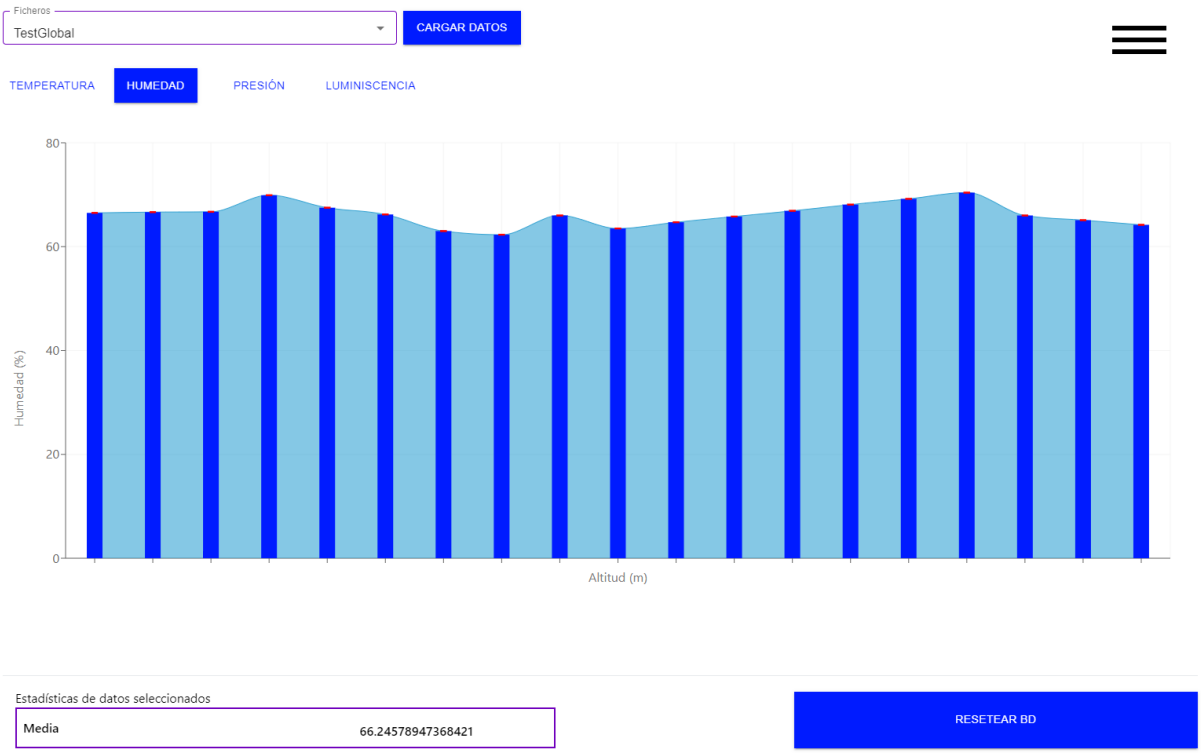


Figura 9.2.- Datos humedad test global

- **Presión**

  La presión como era de esperar resulta ser un índice muy relevante para determinar la altitud. Según vamos ascendiendo tendremos menos aire ejerciendo presión sobre nosotros. Podemos observar como la gráfica va disminuyendo casi de forma constante según nos acercamos al pico máximo de altura de nuestra ruta. El error es muy pequeño de nuevo.

Figura 9.2.- Datos presión test global

- **Luminiscencia**

  Los datos de luminiscencia explican de forma muy clara la niebla presente en el tercer cuarto de la ruta, donde la incidencia es menor debido a la densidad de la niebla y por tanto vemos ese "valle" en la gráfica. El error esta vez es visible mediante una barra, el sensor lumínico parece ser el sensor de los disponibles que más induce al error en la toma de datos.

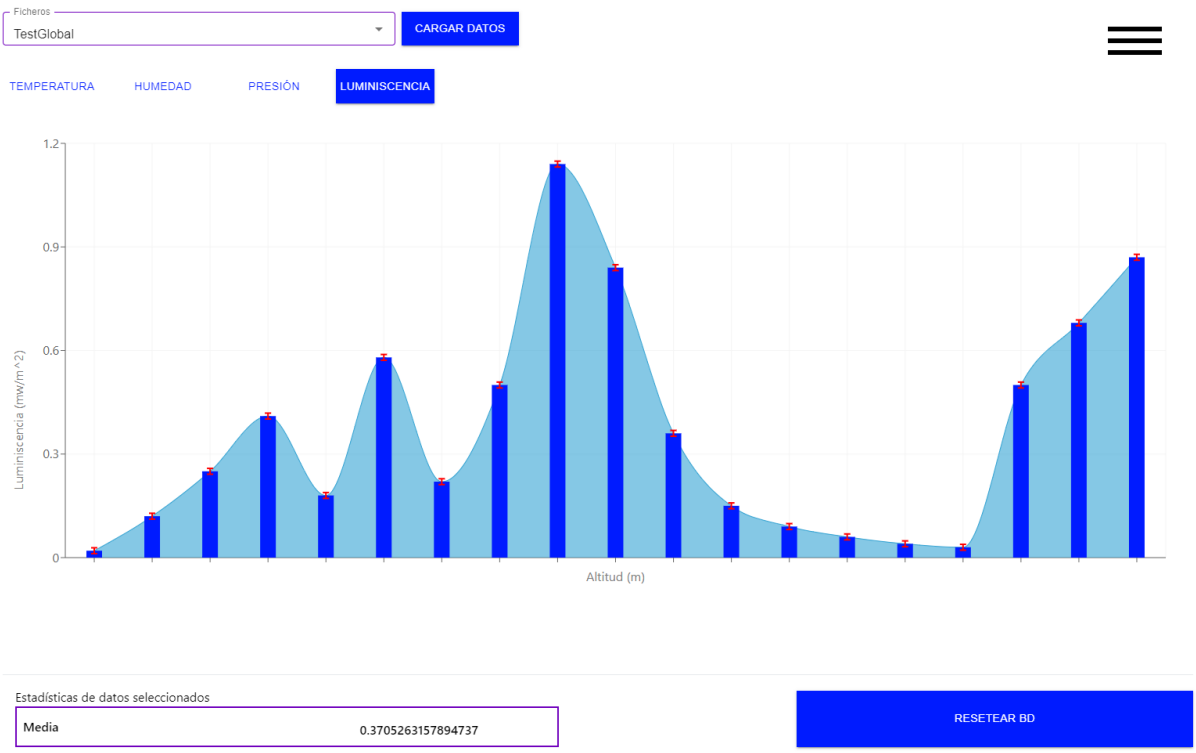


Figura 9.2.- Datos luminiscencia test global

10. Manual de usuario y del instalador

10.1.- Manual de Usuario

Se ha decidido en base a la sencillez tanto de la aplicación web como de la aplicación móvil que estas aplicaciones no se acompañarán de manual de usuario. Para más inri, la aplicación web, pudiendo resultar un poco más complejo su funcionamiento, tiene el flujo de navegación explicado en el apartado 8.3.

10.2.- Instalador de la aplicación móvil

- En primer lugar, se debe descargar el archivo .APK disponible en el repositorio en el móvil deseado.

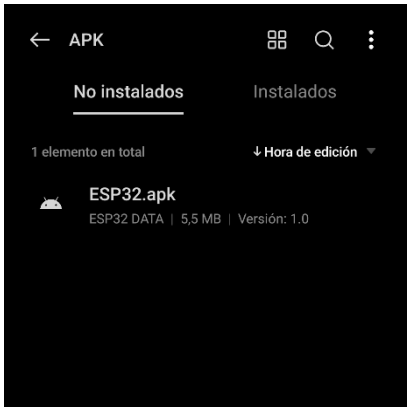


Figura 10.2.- Instalador app móvil pantalla 1

- Abrir dicho archivo desde el dispositivo móvil:

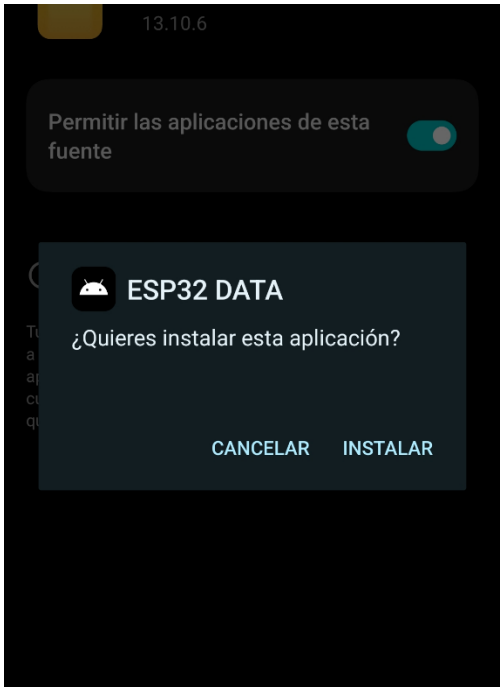


Figura 10.2.- Instalador app móvil pantalla 2

- Se debe confirmar que se conoce el origen de la aplicación y que se tiene confianza en la misma.

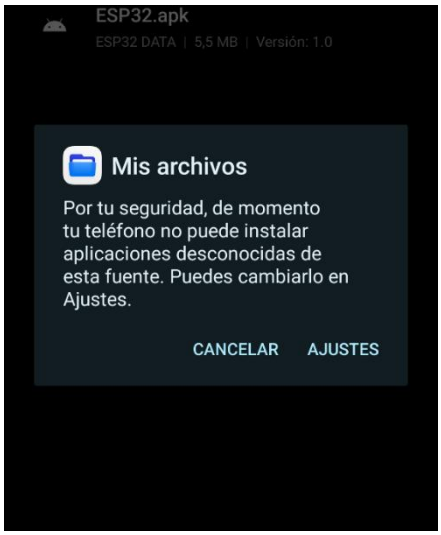


Figura 10.2.- Instalador app móvil pantalla 3

- Finalmente, la aplicación se encuentra instalada en el dispositivo móvil y lista para su uso.

Figura 10.2.- Instalador app móvil pantalla 4

10.3.- Manual de instalador de aplicación web.

- Descargar el instalador disponible en el repositorio, se corresponde al archivo .msi y una vez descargado, ejecutarlo.

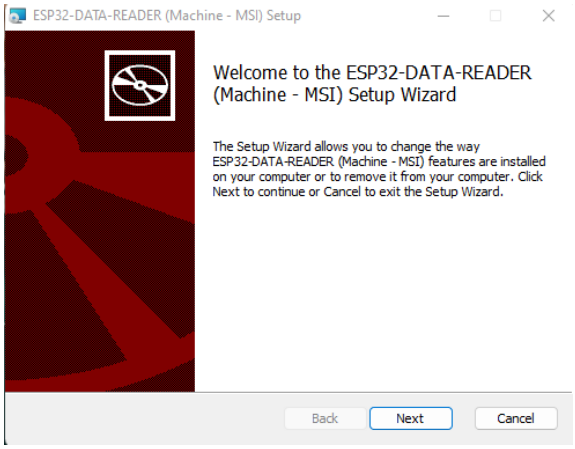


Figura 10.3.- Instalador app web pantalla 1

- Posteriormente deberemos seleccionar la localización de instalación de la aplicación en nuestro PC, donde podemos observar que el tamaño de la aplicación no supera los 350mb.

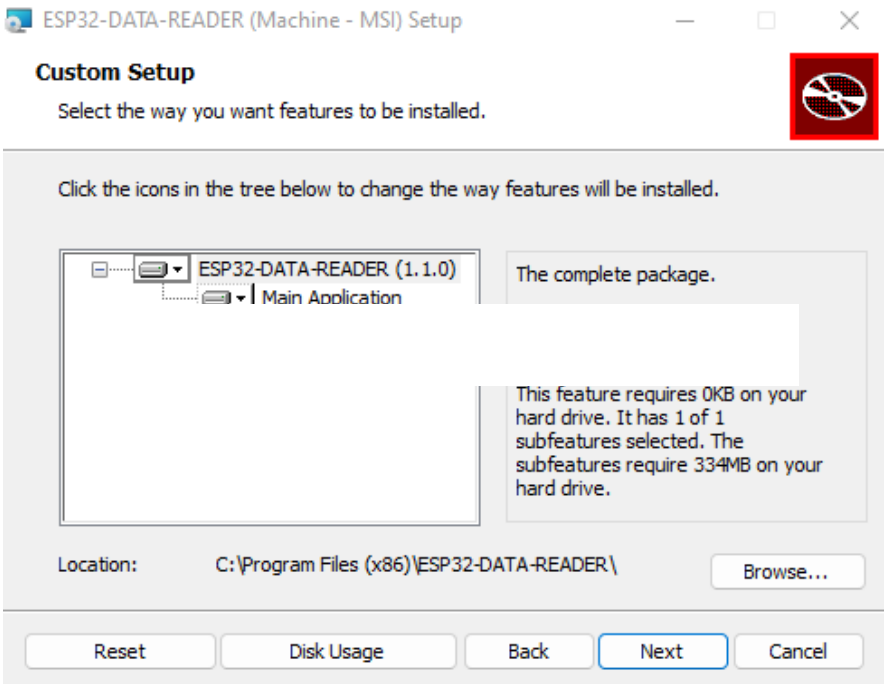


Figura 10.3.- Instalador app web pantalla 2

- Finalmente, instalar con permisos de usuario administrador para escribir en disco y la aplicación deberá estar lista para su ejecución.

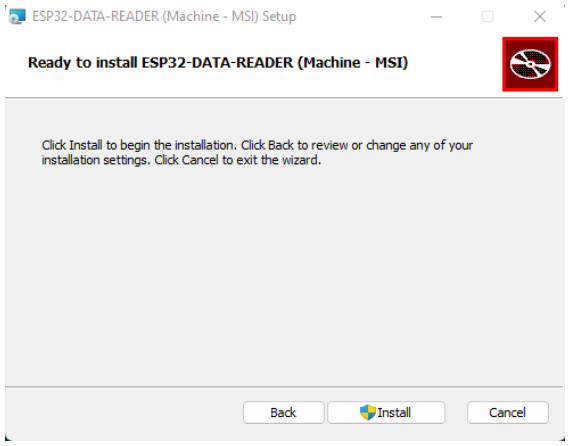


Figura 10.3.- Instalador app web pantalla 3

Figura 10.3.- Instalador app web pantalla 4

11. Conclusión y posibles mejoras

Hemos sido capaces de desarrollar un dispositivo capaz de captar datos meteorológicos con aceptable grado de fiabilidad y pudiendo adaptarse adecuadamente a propuestas con grandes variaciones de altura, ajustándonos al presupuesto inicial estimado sin renunciar a requerimientos de funcionalidad.

Por otro lado, la aplicación móvil fue realmente útil para dar constancia no solo de que el programa contenido en el microcontrolador funcionaba correctamente, sino la certeza de los datos que se estaban captando en el momento, pudiendo ser comparados con otro dispositivo o artilugio que sirva de punto de referencia de los datos recogidos con nuestro dispositivo.

Como guinda del proyecto, la aplicación web me ha parecido indispensable para corroborar el buen funcionamiento del dispositivo de captación de datos, debido a la visión que nos otorgan sus diferentes gráficas y métricas.

He de reseñar la importancia de las asignaturas “Desarrollo de aplicaciones móviles” y la asignatura “Tecnologías Web para el desarrollo de las aplicaciones”, especialmente la primera ya que supuso mi iniciación en el lenguaje de Kotlin.

Ha sido muy gratificante aplicar conocimientos adquiridos en la carrera como los citados en el párrafo previo para resolver soluciones reales en este proyecto, lo que me ha dado confianza y me ha reforzado en la toma de decisiones durante el proyecto.

La metodología Scrum ha permitido no solo un rápido y fluido desarrollo del producto, también ha conseguido mejorar la comunicación entre estudiante y tutores. Sin la aplicación de esta metodología, estoy seguro de que no se hubiese cumplido el plazo de entrega.

He adquirido una gran cantidad de conocimientos muy interesantes en este proyecto, conocimientos relacionados con microcontroladores, servicios AWS, sistemas de comunicación

para servicios IOT… Aspectos que no he visto en ningún momento en mi carrera de estudiante y resultan realmente interesantes.

Una posible mejora que se ha valorado y por falta de tiempo se ha descartado, consistía en realizar un PCB que facilitase el transporte de los dispositivos, debido a la formación requerida en el programa de diseño de PCBs 'Proteus' y la correspondiente elaboración del diseño, no ha dado tiempo físico a realizarla.

La aplicación móvil desarrollada ha sido muy básica para cubrir solamente el track de los datos en directo, pero con el potencial de los servicios IOT que proporciona la plataforma AWS, se podía haber desarrollado una aplicación móvil bastante más compleja. Se puede valorar implementar en el futuro, una gestión de la base de datos desde la aplicación del móvil pudiendo borrar o editar datos desde el dispositivo móvil.

Del mismo modo, quizás resultaría interesante añadir nuevas métricas o gráficas a la aplicación web, que permitan la comparación entre datos de distintos ficheros. De este modo, se podría ver la evolución de datos tomados desde distintos puntos sin la necesidad de realizar dos gráficas distintas.

Para complementar la aplicación móvil con la visualización de los datos en directo, se tendrá que añadir a los dispositivos hardware una pantalla LCD de unas 4 pulgadas para no romper la portabilidad del dispositivo, que nos permitiese ver los datos recogidos en directo, el día que se realizó el test se echó en falta algo similar.

Quizás se podría adquirir nuevos dispositivos al hardware que permitiesen captar otros datos, que a pesar de ser menos relevantes fuesen de alguna utilidad para un posible cliente. Enriqueciendo no solo los datos recogidos, sino aportando nuevas funcionalidades a ambas aplicaciones complementarias.

Finalmente, para completar el proyecto, se enviará el dispositivo diseñado en un globo de alta altitud, para poder captar datos desde el nivel del mar hasta la estratosfera. El test que hemos realizado previamente nos certifica la fiabilidad y resistencia del dispositivo.

Se espera que esta captación de datos en la estratosfera nos provea datos interesantes para el estudio de la misma, pudiendo ayudar a investigadores en su labor.

12. Glosario

Listado de palabras específicas del campo de la informática, cuyo significado para un usuario sin experiencia en la materia pueda despistar:

- SD: Tarjeta de memoria extraíble utilizada para almacenar datos digitales.
- Feedback: Respuesta o retroalimentación proporcionada por un sistema o usuario.
- Nulabilidad: Estado de un objeto o variable que puede tener un valor nulo.
- Backend: Parte de un sistema informático que se encarga del procesamiento y almacenamiento de datos.
- Frontend: Parte de un sistema informático que se encarga de la interfaz de usuario y la interacción con el usuario.
- Framework: Conjunto de herramientas y bibliotecas que facilitan el desarrollo de software.
- Mqtt: Protocolo de mensajería de telemetría de máquina a máquina.
- Streaming: Transmisión de datos en tiempo real a través de una red.
- Track: Pista o registro de información que se sigue o registra en un sistema informático.
- API: Interfaz de programación de aplicaciones que permite la comunicación entre diferentes componentes de software.
- On time: Realización o ejecución puntual de una tarea o evento.
- Delay: Retraso o pausa en la ejecución de un proceso o evento.
- Setup: Configuración o preparación inicial de un sistema o dispositivo.
- Loop: Estructura de control que permite repetir un conjunto de instrucciones x veces.
- Timestamp: Marca de tiempo que indica cuándo ocurrió un evento o se generó un dato.
- Debug / Depuración: Proceso de identificar y corregir errores o fallos en un programa.
- Hostear: Alojar en un servidor un sitio web o una aplicación.
- PCB: Placa de circuito impreso que conecta componentes electrónicos en un dispositivo.

13.-Bibliografía

1. Documentación del módulo ESP32 adquirido
   Guía rápida AZ-Delivery ESP32, disponible con compra del dispositivo.

2. Documentación del sensor DS18B20
   https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

3. Documentación del sensor ML8511
   https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/ML8511_3-8-13.pdf

4. Documentación del sensor DHT22
   https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

5. Documentación del sensor BMP180
   http://www.datasheet.es/PDF/770150/BMP180-pdf.html

6. Documentación modulo lector SD para ESP32
   https://www.electronicwings.com/esp32/microsd-card-interfacing-with-esp32

7. Consola de gestión de Amazon
   https://eu-west-3.console.aws.amazon.com/

8. Documentación de servicios AWS de Amazon
   https://docs.aws.amazon.com/