



UNIVERSITAT POLITÈCNICA DE CATALUNYA

BARCELONATECH

Facultat d'Informàtica de Barcelona



EXPLORACIÓN DE REDES LORAMESH PARA APLICACIONES EXPERIMENTALES

JUAN CARLOS RUBIO DIAZ

Director/a

FELIX FREITAG (Departamento de Arquitectura de Computadores)

Codirector/a

ROGER BAIG VIÑAS (Departamento de Arquitectura de Computadores)

Titulación

Grado en Ingeniería Informática (Tecnologías de la información)

Memoria del trabajo de fin de grado

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

21/01/2025

Índice de contenidos

| | |
|----------------------------------------------------------------------|-----------|
| 1. Contextualización y Alcance del proyecto..... | 9 |
| 1.1 Contextualización..... | 9 |
| 1.1.1 Problema a resolver..... | 10 |
| 1.1.2 Definición de conceptos..... | 10 |
| 1.1.2.1 LoRa..... | 10 |
| 1.1.2.2 LoRa Mesher & LoRaChat..... | 11 |
| 1.2 Justificación..... | 12 |
| 1.2.1 Soluciones existentes..... | 13 |
| 1.2.2 Soluciones posibles..... | 13 |
| 1.3 Alcance..... | 14 |
| 1.3.1 Objetivo..... | 14 |
| 1.3.2 Requisitos..... | 14 |
| 1.3.3 Riesgos y obstáculos..... | 15 |
| 1.3.4 Análisis de mercado..... | 15 |
| 1.3.4.1 Redes LoRa comerciales y de código abierto..... | 16 |
| 1.3.4.2 Soluciones de sensores y redes IoT basadas en LoRa..... | 16 |
| 1.3.4.3 Librerías y herramientas de código abierto relacionadas..... | 16 |
| 1.3.5 Identificación de leyes y regulaciones..... | 17 |
| 1.4 Metodología y rigor..... | 18 |
| 1.4.1 Metodología de trabajo..... | 18 |
| 1.4.2 Seguimiento..... | 18 |
| 1.4.3 Metodología de trabajo actualizada..... | 19 |
| 2. Planificación temporal..... | 19 |
| 2.1 Descripción de las tareas..... | 20 |
| 2.1.1 Gestión del proyecto (GP)..... | 20 |
| 2.1.2 Análisis de tecnologías (AT)..... | 21 |
| 2.1.3 Propuestas del trabajo (PT)..... | 22 |
| 2.1.4 Elaboración del proyecto (EP)..... | 23 |
| 2.2 Recursos..... | 24 |
| 2.2.1 Recursos humanos..... | 24 |
| 2.2.2 Recursos materiales..... | 24 |
| 2.3 Estimaciones y Gantt..... | 25 |
| 2.3.1 Estimaciones..... | 25 |
| 2.3.2 Diagrama de Gantt inicial..... | 27 |
| 2.4 Gestión del riesgo: Planes alternativos y obstáculos..... | 28 |
| 2.5 Planificación definitiva..... | 28 |
| 2.5.1 Diagrama de Gantt final..... | 29 |
| 2.5.2 Problemas..... | 30 |
| 2.5.3 Estado actual del proyecto (9/12/2024)..... | 30 |
| 3. Gestión económica y sostenibilidad..... | 31 |
| 3.1 Presupuesto..... | 31 |

| | |
|-------------------------------------------------------|-----------|
| 3.1.1 Recursos humanos..... | 31 |
| 3.1.2 Recursos hardware..... | 33 |
| 3.1.3 Recursos software..... | 34 |
| 3.1.4 Costos genéricos..... | 34 |
| 3.1.5 Contingencia..... | 35 |
| 3.1.6 Imprevistos..... | 35 |
| 3.1.7 Presupuesto final..... | 37 |
| 3.2 Control de gestión..... | 37 |
| 3.3 Informe de sostenibilidad..... | 38 |
| 3.3.1 Autoevaluación..... | 38 |
| 3.3.2 Análisis de la sostenibilidad del proyecto..... | 39 |
| 3.3.2.1 Ambiental..... | 40 |
| 3.3.2.2 Económico..... | 40 |
| 3.3.2.3 Social..... | 41 |
| 4. Conjunto tecnológico..... | 42 |
| 4.1 Placas T-Beam..... | 42 |
| 4.2 LoRa..... | 43 |
| 4.2.1 Redes LoRa Mesh..... | 43 |
| 4.2.2 Librería LoRaMesher..... | 43 |
| 4.2.2.1 Introducción y diseño..... | 43 |
| 4.2.2.2 Tipos de mensajes..... | 44 |
| 4.2.2.3 Herramientas de depuración..... | 44 |
| 4.3 Visual Studio & PlatformIO..... | 45 |
| 4.4 MQTT..... | 45 |
| 4.4.1 Mosquitto..... | 46 |
| 4.4.2 EMQX..... | 46 |
| 4.5 Servicios para monitorizar..... | 47 |
| 4.5.1 InfluxDB & Telegraf..... | 47 |
| 4.5.2 Grafana..... | 47 |
| 4.5.3 React..... | 47 |
| 5. Implementación del estado inicial..... | 48 |
| 6. Análisis de la base..... | 57 |
| 6.1 Investigar la base..... | 57 |
| 6.2 Objetivo del trabajo..... | 58 |
| 6.2.1 Propuesta 1: Deployment..... | 58 |
| 6.2.1.1 Beneficios..... | 58 |
| 6.2.1.2 Desafíos..... | 59 |
| 6.2.2 Propuesta 2: Sensores..... | 59 |
| 6.2.2.1 Beneficios..... | 59 |
| 6.2.2.2 Desafíos..... | 60 |
| 6.2.3 Propuesta 3: Actualización..... | 60 |
| 6.2.3.1 Beneficios..... | 60 |
| 6.2.3.2 Desafíos..... | 61 |
| 6.2.4 Propuesta 4: ESP32..... | 61 |

| | |
|-------------------------------------------|-----------|
| 6.2.4.1 Beneficios..... | 61 |
| 6.2.4.2 Desafíos..... | 62 |
| 6.3 Evaluación del desarrollo..... | 62 |
| 7. Integración de las mejoras..... | 63 |
| 7.1 Cambios en el JSON & React..... | 63 |
| 7.2 Cambios en Grafana..... | 67 |
| 7.3 Conexiones en las placas..... | 69 |
| 7.4 Deployment..... | 71 |
| 8. Resultados..... | 73 |
| 9. Conclusiones..... | 78 |
| 10. Referencias..... | 79 |

Lista de Figuras

| | |
|-----------------------------------------------------------------------------------|----|
| Figura 1: LoRaChat, captura del proyecto en Visual Studio..... | 12 |
| Tabla 1: Tabla resumen de las tareas..... | 25 |
| Figura 2: Diagrama de Gantt inicial del proyecto..... | 27 |
| Figura 3: Diagrama de Gantt final del proyecto..... | 29 |
| Tabla 2: Tabla de sueldos elaborada manualmente..... | 31 |
| Tabla 3: Tabla con los costes estimados por tarea..... | 32 |
| Tabla 4: Tabla de precios en el momento de adquisición..... | 33 |
| Tabla 5: Tabla de costes estimados por contingencias..... | 35 |
| Tabla 6: Tabla de costes estimados por imprevistos..... | 36 |
| Tabla 7: Tabla del presupuesto final estimado..... | 37 |
| Figura 4: Stack tecnológico del proyecto de Jack..... | 42 |
| Figura 5: T-Beam con el programa cargado y en funcionamiento..... | 43 |
| Figura 6: Conexión de las placas mediante cables USB..... | 48 |
| Figura 7: Captura del terminal del Visual Studio al listar dispositivos..... | 49 |
| Figura 8: Captura del terminal del Visual Studio al cambiar los permisos..... | 49 |
| Figura 9: Captura del código con los cambios realizados..... | 50 |
| Figura 10: Captura con el código a cargar en la placa ACM1..... | 51 |
| Figura 11: Captura con el código a cargar en la placa USB0..... | 51 |
| Figura 12: T-Beams con el código cargado y funcionando..... | 52 |
| Figura 13: Captura del terminal iniciando los servicios de monitoreo..... | 53 |
| Figura 14: Captura de las pestañas abiertas al empezar a monitorear..... | 53 |
| Figura 15: Captura del servicio EMQX inicialmente..... | 53 |
| Figura 16: WebApp de React sin estar conectado..... | 54 |
| Figura 17: Captura de React cuando el servicio está conectado..... | 54 |
| Figura 18: Captura de EMQX con React activo..... | 55 |
| Figura 19: Captura de React al hacer la query “Services”..... | 56 |
| Figura 20: Captura de Grafana después de generar tráfico de red..... | 56 |
| Figura 21: Captura de Grafana con el gráfico que muestra el valor del sensor..... | 68 |
| Figura 22: Conexión de los sensores a las T-Beam..... | 69 |
| Figura 23: Pines de las placas T-Beam..... | 70 |
| Figura 24: Captura de los mensajes MQTT que llegan a la MV..... | 72 |
| Figura 25: Captura que muestra mensajes MQTT con los valores del sensor..... | 74 |
| Figura 26: Captura de React al hacer la query “Sensor Value”..... | 74 |
| Figura 27: Captura de Grafana con los valores recogidos durante 48 horas..... | 75 |
| Figura 28: Captura que muestra mensajes MQTT en la máquina virtual..... | 76 |
| Figura 29: T-Beam con el código cargado conectado a la MV..... | 77 |

Resumen

El proyecto "Exploración de redes LoRaMesh para aplicaciones experimentales" busca optimizar la tecnología LoRa en redes de malla, superando las limitaciones inherentes a las redes en estrella tradicionales, como su dependencia de gateways centralizados. Este trabajo propone mejorar la librería de código abierto LoRaMesher, y ampliar sus usos mediante el desarrollo de aplicaciones prácticas para el Internet de las Cosas (IoT).

El proyecto aborda la integración de sensores en nodos LoRa, y la extensión de capacidades existentes, como el despliegue en entornos de Internet. Esto permite avanzar hacia redes más resilientes y escalables, útiles para aplicaciones como monitorización ambiental o ciudades inteligentes. Los resultados obtenidos confirman la viabilidad de los avances propuestos, destacando el potencial de las redes LoRaMesh para su uso práctico.

Resum

El projecte "Exploració de xarxes LoRaMesh per a aplicacions experimentals" busca optimitzar la tecnologia LoRa en xarxes de malla, superant les limitacions inherents a les xarxes en estrella tradicionals, com la seva dependència de gateways centralitzats. Aquest treball proposa millorar la llibreria de codi obert LoRaMesher i ampliar-ne els usos mitjançant el desenvolupament d'aplicacions pràctiques per a l'Internet de les Coses (IoT).

El projecte aborda la integració de sensors en nodes LoRa i l'extensió de capacitats existents, com el desplegament en entorns d'Internet. Això permet avançar cap a xarxes més resilients i escalables, útils per a aplicacions com la monitorització ambiental o les ciutats intel·ligents. Els resultats obtinguts confirmen la viabilitat dels avenços proposats, destacant el potencial de les xarxes LoRaMesh per al seu ús pràctic.

Abstract

The project "Exploration of LoRaMesh Networks for Experimental Applications" aims to optimize LoRa technology in mesh networks, overcoming the limitations inherent to traditional star-topology networks, such as their dependence on centralized gateways. This work proposes improving the open-source library LoRaMesher and expanding its uses through the development of practical applications for the Internet of Things (IoT).

The project addresses the integration of sensors into LoRa nodes and the extension of existing capabilities, such as deployment in Internet environments. This enables progress toward more resilient and scalable networks, useful for applications such as environmental monitoring or smart cities. The results obtained confirm the feasibility of the proposed advancements, highlighting the potential of LoRaMesh networks for practical use.

1. Contextualización y Alcance del proyecto

Este trabajo corresponde al proyecto final del Grado en Ingeniería Informática con especialización en Tecnologías de la Información. El grado es cursado en la Facultad de Informática de Barcelona, dentro de la Universitat Politècnica de Catalunya.

El proyecto se realiza en modalidad A, publicado en la web del Racó de la Fib, y tiene como director a Felix Freitag y como codirector a Roger Baig Viñas.

Por lo que respecta a las competencias técnicas asociadas a este proyecto tenemos [1]:

- CTI3.1: Concebir sistemas, aplicaciones y servicios basados en tecnologías de red, teniendo en cuenta Internet, web, comercio electrónico, multimedia, servicios interactivos y computación ubicua. [En profundidad].
- CTI3.3: Diseñar, implantar y configurar redes y servicios. [En profundidad].
- CTI3.4: Diseñar *software* de comunicaciones. [Bastante].

1.1 Contextualización

En los últimos años, las tecnologías de comunicación inalámbrica han tenido un notable crecimiento debido a la expansión del Internet de las Cosas (o abreviando, *IoT*, por sus siglas en inglés) y la necesidad de conectar dispositivos de forma eficiente en todo tipo de entornos. Una de las tecnologías protagonistas en este contexto es *LoRa* (*Long Range*), un protocolo de comunicación que ofrece un largo alcance, usando baja potencia, por lo que es ideal para aplicarlo en este campo.

Sin embargo, no todo son ventajas. *LoRa* presenta limitaciones tales como la dependencia de una infraestructura de red centralizada basada en estaciones base (*gateways*) que no tiene por qué ser siempre adecuada. Con el fin de remediar esto, surgen las redes en malla (*mesh networks*) como alternativa para ampliar las capacidades de las redes *LoRa*, ya que, proporcionan una topología descentralizada donde cada nodo puede hacer de repetidor. Esto permite extender el alcance de la propia red y mejorar la resiliencia ante fallos de los nodos.

La descripción del proyecto era: “We have been building at UPC a library called *LoRaMesher* that enables creating *LoRa* mesh network with embedded devices. The project should make a contribution to this open source library and help to make it more usable or showing new usages [2].”

1.1.1 Problema a resolver

El objetivo del trabajo es mejorar un trabajo ya existente. El punto de partida es el TFG de Jack Griffiths, también estudiante de la FIB, que conectó las placas que usaré en este proyecto. Por lo que respecta a nivel de madurez tecnológica o TRL, podríamos clasificar lo de Jack como TRL4 (nivel más bajo de simulación). En este trabajo se busca incrementar este TRL, teniendo como objetivo llegar a TRL6, el nivel más alto en la fase de simulación [3].

Hoy en día se han incrementado exponencialmente las aplicaciones que se le dan los dispositivos *IoT* siendo necesarias en implementaciones de sensores o aplicación inteligentes o *Smart*. Con el fin de llevar a cabo este sistema, es necesario un método adecuado y eficiente para la comunicación. Actualmente, este método es *LoRa*, pero tiene problemas en cuanto a diseño. Para el correcto funcionamiento de esta tecnología, se usan redes en estrella, lo que significa que es poco resiliente a fallos, entre otras cosas. Este trabajo pretende investigar otro diseño de la red: redes en malla o, aplicados a *LoRa*, *LoRamesher*, que soluciona los problemas provocados por las ya mencionadas, redes en estrella.

1.1.2 Definición de conceptos

1.1.2.1 *LoRa*

La tecnología *LoRa*, desarrollada por *Semtech*, se implementa dentro de una red *LoRaWAN*, en la cual dispositivos de *IoT*, como medidores inteligentes o sistemas de alarmas, se conectan a través de *gateways* que, a su vez, están conectados a Internet. Esto permite que los nodos de *IoT* interactúen con un servidor de red central. *LoRaWAN* es una red de área amplia de largo alcance (*Wide Area Network*), que usa los dispositivos *IoT* de *LoRa*, por lo que se ha convertido en la tecnología *LPWAN* más destacada y extendida. *LoRa* es preferida sobre otras soluciones debido a su bajo consumo de energía, mayor autonomía de batería, amplio rango de cobertura, coste reducido y facilidad de integración.

Actualmente, podemos encontrar *LoRa* en aplicaciones *Smart*, es decir, ciudades, hogares, agricultura y sistemas de medición inteligentes, ya que están impulsados por los propios dispositivos *IoT*.

Encontramos tres tipos de dispositivos *LoRa* dependiendo de su comportamiento y consumo [4]:

- Clase A: Estos dispositivos son los más eficientes en cuanto a consumo energético, ya que permanecen inactivos la mayor parte del tiempo. Estos se

activan para enviar un mensaje y entran en modo escucha hasta recibir una respuesta.

- Clase B: Consumen más energía que los de Clase A al activarse de manera periódica para enviar datos.
- Clase C: Los instrumentos de clase C están activos siempre en espera de recibir un mensaje, y cuando es necesario, transmiten. Esto los hace los dispositivos con mayor coste energético.

La decisión de utilizar *LoRa* en este proyecto se debe a que pretende ampliar una librería existente llamada *LoRaMesher*, que se detalla más adelante.

1.1.2.2 *LoRa Mesher & LoRaChat*

LoRaMesher es una librería desarrollada por Joan Miquel Solé, exalumno de la UPC, que implementa un protocolo de enrutamiento de vector de distancia en redes *LoRa*. Esta librería será utilizada en el proyecto para explorar y crear aplicaciones “cross-networks” en redes de malla *LoRa*.

En la página siguiente, se presenta una captura del código de *LoRaChat*, que utiliza *LoRaMesher* como servicio. *LoRaChat* facilita la creación de aplicaciones para dispositivos *LoRa* y será una herramienta clave en este proyecto para desarrollar las aplicaciones *IoT* deseadas. El proyecto está organizado de la siguiente forma [5]:

- */src*: Contiene las aplicaciones creadas para los dispositivos *LoRa*. Actualmente, ya existen varias *apps* con diferentes funciones, y el objetivo del proyecto es expandir esta carpeta con nuevas aplicaciones. Para ello, se utilizarán *apps* existentes como referencia.
- */src/message/dataMessage.h*: Las aplicaciones deben estar referenciadas en este archivo para ser utilizadas (cada una está asignada a un número).
- *config.h*: Especifica la configuración de los nodos *LoRa* y las aplicaciones correspondientes, que pueden ser activadas o desactivadas.
- *main.cpp*: Aquí se crean las instancias de las aplicaciones para que puedan ser ejecutadas.
- *platformio.ini*: Archivo de configuración de *PlatformIO*. Define dispositivos *IoT* y librerías utilizadas.
- *echo monitoring and cdp min.py*: Archivo *Python* que actúa como publicador y suscriptor de los mensajes enviados a través de *MQTT*.

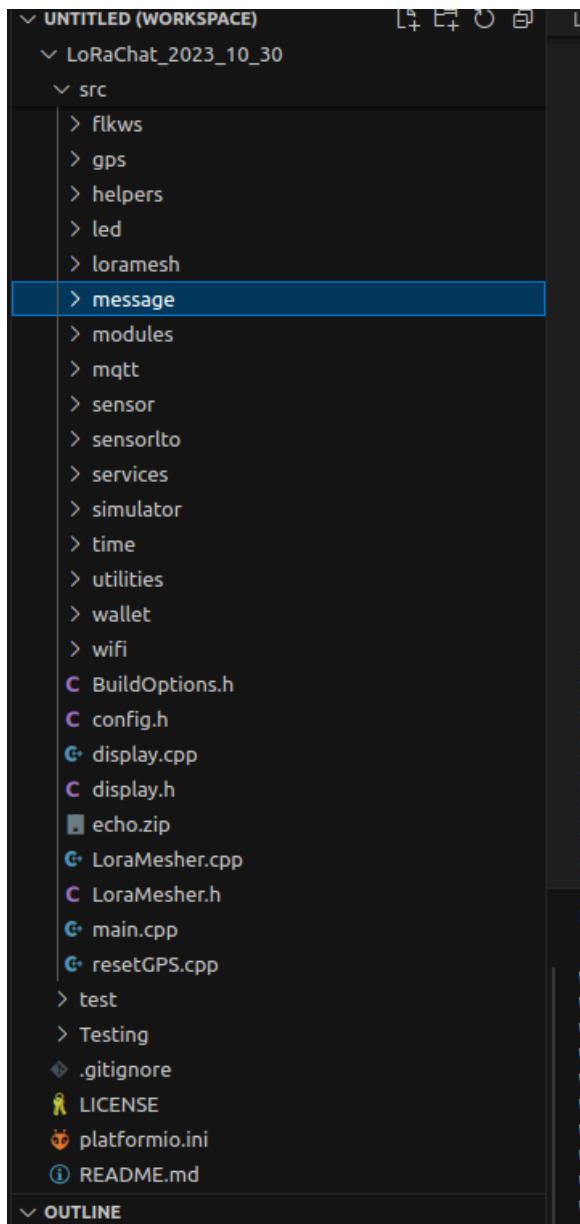


Figura 1: LoRaChat, captura del proyecto en Visual Studio.

1.2 Justificación

Este proyecto será útil si, como ya se ha mencionado en el apartado 1.1.1 Problema a resolver, se consigue un incremento en la escala de madurez, que es sinónimo de: un proyecto en fase de investigación a punto de ser utilizable en el mundo real.

1.2.1 Soluciones existentes

Actualmente, hay soluciones que ya implementan lo que se espera como final de este proyecto. Dado que hay sistemas puestos en marcha utilizando redes *LoRa*, podríamos decir que “no estamos haciendo nada nuevo”, sino que, lo que se hará es buscar otro método o forma más eficiente, para hacer lo que ya se hace. En este momento, ya podemos encontrar sistemas que monitorizan la agricultura, gestionan ciudades y casas “inteligentes” o miden el consumo de energía o el agua.

1.2.2 Soluciones posibles

En cuanto a la propuesta en sí, de momento, estamos evaluando qué opción es más factible e interesante. Se comentaron las siguientes opciones:

1. *Deployment*: del entorno local a Internet. Esta opción consiste en usar una VM y MQTT broker en Internet, teniendo las placas en casa. Para ampliar, en Campus Nord, se dejarían las placas conectadas a miniPCs que se controlarán remotamente.
2. Añadir sensores a las placas y extender que el *monitoring* envía valores de sensores (actualmente, el *monitoring* envía valores del *LoRaMesher*, o sea, de su propio software). Podríamos conectar sensores a las placas, leer los valores y enviarlos.
3. "Portar" los componentes de Jack al *LoRaMesher/LoRaChat* actual. Jack usó un *LoRaMesher/LoRaChat* de hace 1 año. Se podría integrar lo que ha hecho en una versión actualizada. Habría que adaptar ciertos *calls* del trabajo e incluir la carpeta al nuevo *LoRaMesher/LoRaChat* actual.
4. Validar con otras placas, no solo los *T-Beam*. Se dispone de otro tipo de placas con *ESP32*. Si funcionan, se podría validar y usar en el *deployment* para tener uno con 2 tipos de nodos.

En todos los casos, el trabajo se basa en mejorar el TFG de Jack (ya mencionado anteriormente). Sabiendo que el proyecto que estoy llevando a cabo parte de otro proyecto existente, no hay demasiadas alternativas a la hora de implementar las propuestas. En este caso, lo que sí se ha hecho, es analizar las 4 propuestas iniciales, para evaluar cual es la más adecuada (ver apartado 6.2 Objetivo del trabajo).

1.3 Alcance

El proyecto se va a desarrollar siguiendo unos objetivos propuestos. Se hará el trabajo de investigación para ver si las ideas pensadas son viables en el mundo real y se justificará por qué sí, o, porque no valen para lo que se había imaginado inicialmente.

1.3.1 Objetivo

El objetivo principal de este proyecto es contribuir al mundo del *IoT* mediante la creación de aplicaciones útiles para redes de malla *LoRa*. Otra forma de decirlo sería hacer un uso más común y extendido de este tipo de redes, haciendo que sean tanto eficientes como útiles.

Para conseguir este objetivo principal, hemos de alcanzar, también, los siguientes subobjetivos:

- Comprensión adecuada de las redes *LoRa*.
- Familiarización con la programación de aplicaciones para nodos *IoT*.
- Identificación del tipo de aplicaciones creadas y por crear.
- Creación de aplicaciones para los nodos *IoT*.
- Pruebas de las aplicaciones para los nodos *IoT*.
- Revisión y optimización de las aplicaciones.
- Configuración correcta de una red de malla *LoRa*.

1.3.2 Requisitos

A continuación, hay una lista de requisitos necesarios para el correcto desarrollo del proyecto, que se podrían clasificar según sean funcionales o no funcionales:

Funcionales:

- Adaptabilidad y compatibilidad con los demás nodos dentro de la red *LoRa*.
- Garantizar el uso correcto de las aplicaciones, lo que significa que no debe haber errores.
- Comentar y documentar el código para su fácil comprensión.

No funcionales:

- Los resultados deben ser de fácil acceso y servir de referencia a los que busquen mejorar este trabajo.
- Informarse del funcionamiento de redes *LoRa* y nodos *IoT*.
- Revisión continua del proyecto para asegurar el correcto avance.

1.3.3 Riesgos y obstáculos

Para el proyecto se han identificado los siguientes obstáculos y/o riesgos que pueden afectar al desarrollo de este:

- Fecha límite: hay un plazo establecido para la entrega, por lo que se ha de gestionar bien el tiempo. Puede influir al tiempo que se dedique en cada tarea e impactar negativamente a la calidad del proyecto en caso de retraso.
- Falta de experiencia y disponibilidad de la información: no se ha trabajado antes con este tipo de aplicaciones, además de no haber una gran cantidad de información disponible. Es necesaria una fase de adaptación y comprensión de las tecnologías. Gran parte del código está desarrollado por otra persona, por lo que, hay que entender la base.
- Errores o retrasos: como ya se ha comentado, en caso de retraso, fallo de las aplicaciones o errores de programación afectan considerablemente la organización y planificación de tareas además de la calidad del trabajo.
- Fallos inesperados o incontrolables: en todo caso, aunque no sea muy probable, hay que tener en cuenta la posibilidad de una falla de *hardware* (tanto placas como el ordenador en cuestión), además de situaciones médicas imprevistas.
- Desconocimiento del final: en los proyectos de investigación, nos podemos encontrar que el camino a seguir no lleva a una solución exitosa. Lo que me recuerda a la siguiente frase “Caminante, no hay camino, se hace camino al andar”[6], que sugiere que no hay un destino predefinido, sino que, se construye a medida que avanzamos.

1.3.4 Análisi de mercado

A continuación se listan posibles productos relacionados clasificándolos en las 3 partes del proyecto: redes, *IoT* y librerías.

1.3.4.1 Redes *LoRa* comerciales y de código abierto

The Things Network (TTN)[7]:

TTN es una infraestructura global de *LoRaWAN* diseñada para conectar dispositivos *IoT* utilizando *gateways*. Aunque utiliza *LoRa*, no implementa la red en malla. Es ideal para áreas con cobertura de *gateways*, pero menos práctico en zonas remotas donde una red en malla tiene ventaja.

Helium Network[8]:

Una red descentralizada que utiliza *LoRaWAN* y *blockchain* para aplicaciones *IoT*. Su modelo es más comercial y requiere *gateways* específicos para integrarse. Las redes en malla podrían complementar este modelo al ofrecer conectividad en lugares donde no haya cobertura.

Mesh Extenders (Serval Project[9]):

Aunque no está basado en *LoRa*, este proyecto experimental utiliza radios de baja potencia para crear redes en malla. Es una idea similar, pero con tecnologías diferentes. El enfoque en *LoRa* podría ofrecer un alcance mayor y un costo más bajo.

1.3.4.2 Soluciones de sensores y redes *IoT* basadas en *LoRa*

Sensores *LoRaWAN* comerciales (como *Dragino*[10], *Seeed Studio*):

Empresas como *Dragino* ofrecen sensores de temperatura, humedad, y calidad del aire que usan *LoRaWAN* para la comunicación. La infraestructura que utilizan es muy similar a la que utilice yo, el autor, en el proyecto. Sin embargo, *Dragino* depende de un *gateway LoRaWAN* para conectarse a la red y no implementa directamente una topología en malla.

1.3.4.3 Librerías y herramientas de código abierto relacionadas

La librería “*LoRaMesher*” que estamos utilizando es un ejemplo único de implementación de redes en malla en *LoRa*. Por lo que su uso en proyectos reales no está muy extendido. Debido a esto, este proyecto podría destacar al mostrar una aplicación práctica y funcional.

A continuación, presento similares a *LoRaMesher*:

Mesh Networking en *IoT*: Aunque no basado en *LoRa*, existen tecnologías, como es el caso de *Zigbee*[11], que implementan redes en malla para *IoT*. *LoRa* ofrece un mayor alcance y mejor penetración en áreas rurales.

Comparativa y posición del proyecto:

- Ventajas frente a redes *LoRaWAN* comerciales: Las redes en malla eliminan la necesidad de *gateways* centralizados, haciéndolo más útil para zonas rurales o con infraestructura limitada.
- Innovación frente a sensores comerciales: La integración de sensores en una red en malla *LoRa* ofrece una solución escalable para *IoT*, combinando monitoreo ambiental con conectividad flexible.
- Contribución tecnológica: Al mejorar y demostrar la viabilidad de redes en malla *LoRa*, se podrían abrir nuevas aplicaciones en *IoT*, destacándose especialmente en contextos donde la cobertura o los costos de infraestructura son limitantes.

1.3.5 Identificación de leyes y regulaciones

En el desarrollo de este proyecto he encontrado un total de 3 leyes que se debería tener en cuenta:

“1. Regulaciones sobre el uso del espectro de frecuencias

Reglamento de la UIT (Unión Internacional de Telecomunicaciones): Define las bandas *ISM (Industrial, Scientific, and Medical)* para uso sin licencia, donde opera *LoRa*.

Normativa europea (*ETSI EN 300 220*): Regula el uso de dispositivos de corto alcance en la banda de 868 MHz en Europa. Especifica límites de potencia de emisión (25 mW) y un *Duty Cycle* máximo del 1% para transmisiones en esta banda.

2. Normativa sobre equipos electrónicos

Marcado CE: Los dispositivos electrónicos en la Unión Europea deben cumplir las directivas de seguridad eléctrica, compatibilidad electromagnética (*EMC*) y restricciones de sustancias peligrosas (*RoHS*).

Directiva RED (2014/53/EU): Regula equipos de radio y sus características técnicas en la UE.

3. Regulaciones de Internet y comunicaciones

Ley de Servicios Digitales (*DSA*): En la UE regula la prestación de servicios a través de Internet.

Ley de Telecomunicaciones (España): Establece las condiciones para la interconexión de redes y servicios de telecomunicaciones.” - *ChatGPT*

1. Si las placas *T-Beam* están configuradas para operar en la banda de 868 MHz (en Europa) y cumplen los límites de potencia y *Duty Cycle*, el proyecto debería ajustarse a estas normativas.
2. Las placas *T-Beam* deben contar con certificación CE y cumplir la Directiva RED si se utilizan comercialmente. En entornos de prueba académica, esta exigencia puede ser menos estricta, pero es recomendable asegurarse de que los módulos *LoRa* utilizados están certificados.
3. En caso de hacer el *deployment* a Internet, se debe garantizar la seguridad en la comunicación *MQTT* (por ejemplo, mediante *TLS*) y evaluar la necesidad de registro o permisos si la red interactúa con servicios regulados.

1.4 Metodología y rigor

1.4.1 Metodología de trabajo

Este proyecto adoptará una metodología *Agile*. Esta forma es la que más nos interesa debido a la transparencia y la comunicación, el desarrollo iterativo, la planificación adaptativa y la mejora continua. Se realizará una planificación con el fin de identificar cuándo se llevará a cabo cada cosa. Durante el desarrollo del proyecto, se realizarán reuniones con el director (usando *Meet*) para discutir los avances. En estas reuniones se podrán formar nuevas tareas y añadirlas al plan existente. De forma añadida, también habrá comunicación por correo electrónico con el director con el fin de resolver dudas.

1.4.2 Seguimiento

Ya se ha comentado que a medida que se vaya avanzando en el proyecto, irán surgiendo nuevas mejoras y ampliaciones a llevar a cabo. Para esto, sería necesario llevar un correcto seguimiento del trabajo con el fin de tener siempre en cuenta los objetivos que se han alcanzado, los que se irán alcanzando y el tiempo que llevará cada uno de estos.

Primeramente se planificará a través de un diagrama de *Gantt*, posteriormente mostrado, con el que tendremos una vista aproximada del tiempo del proyecto. Este diagrama especifica las tareas a realizar y el tiempo que van a durar estas, por lo que es una herramienta de control bastante fiable.

A la misma vez, se harán reuniones a través de *Google Meet* con el fin de mantener un contacto más directo y rápido con el director del trabajo que usando correo

electrónico, que será más frecuente para dudas puntuales y con respuesta breve o sencilla.

1.4.3 Metodología de trabajo actualizada

La metodología actual del trabajo funciona de la siguiente manera:

1. Hablar con el tutor sobre el objetivo. (ya sea implementar la base, los cambios, etc.)
2. Realizar este objetivo
 - a. Si hay algún problema/impedimento, volver a contactar con el tutor con el fin de solucionarlo o buscar alguna alternativa.
 - b. En caso de que en el plazo de aproximadamente 1 semana, no se haya conseguido, se reporta al tutor el estado.
3. Probar que funcione correctamente.
4. Compartir el nuevo proyecto con el cambio hecho.
5. Volver al paso 1.

Dado que la metodología actual de trabajo funciona correctamente, no vale la pena hacer cambios en ella. En cuanto al seguimiento original, tampoco cambiaría.

2. Planificación temporal

En esta sección del proyecto especificaremos una planificación temporal de las tareas que deben llevarse a cabo para cumplir con uno de los requisitos principales del trabajo de fin de grado, completarlo dentro del plazo establecido.

La fecha de inicio es el miércoles 18 de septiembre de 2024 y se prevé finalizar el 24 de enero de 2025 (último día entre las fechas de lectura). Por lo tanto, se calcula que el desarrollo del trabajo se llevará a cabo en 19 semanas, incluyendo la preparación de la presentación para la defensa oral del proyecto con la documentación ya entregada.

Se estima una dedicación diaria de 4 horas al proyecto. La estimación es un promedio, ya que pueden surgir imprevistos durante la semana. En tal caso, se recuperará el tiempo, principalmente, durante el fin de semana, cuando se dispone de más tiempo para dedicar al trabajo.

Con esto, se pueden calcular aproximadamente 28 horas semanales y un total aproximado de 480 horas necesarias para el desarrollo del trabajo.

2.1 Descripción de las tareas

2.1.1 Gestión del proyecto (GP)

Nuestro primer bloque de tareas son las que se relacionan de alguna manera con la gestión del proyecto. Para elaborar un trabajo exitoso, es necesario este bloque, que incluye la planificación de tiempo y trabajo además de la documentación de este.

GP1 - Contextualización y alcance

Como tarea inicial del proyecto nos encontramos con esta contextualización. Aquí hemos de hacer un análisis de la situación actual, definiendo objetivos y límites concretamente. Adicionalmente, hemos de identificar los recursos necesarios para cumplir con las metas establecidas.

Empezando esta parte en la primera entrega de GEP y habiendo de corregir y completar, se estima una dedicación de unas 32 horas.

GP2 - Planificación

La segunda tarea del proyecto es la parte de la planificación temporal de este. Incluye especificaciones sobre las tareas a realizar y la estimación del tiempo que durarán estas. Se ha de tener en cuenta todos los tiempos supuestos y los posibles obstáculos que pueden surgir para cumplir con los plazos y completar el trabajo exitosamente. La planificación tiene como fin controlar si se avanza de manera satisfactoria el cumplir los objetivos y ajustar el tiempo en caso de inconvenientes.

Esta tarea corresponde con la segunda entrega de GEP y, corrigiendo y completando, se estima una dedicación de unas 27 horas.

GP3 - Presupuesto y sostenibilidad

Tarea número 3 y, entregable 3 de GEP. Estamos en la parte de evaluar el presupuesto y la sostenibilidad al realizar el proyecto. En otras palabras, se mide el impacto económico, social y medioambiental del trabajo.

Igual que en las 2 tareas anteriores, contando corrección y perfección, se estima una duración de unas 25 horas.

GP4 - Entrega final GEP

Última parte de GEP. La entrega final consiste en unificar en un solo documento, con las correcciones ya hechas, las 3 entregas anteriores de GEP, por lo tanto, es dependiente de GP1, GP2 y GP3.

Teniendo en cuenta que las correcciones de las entregas están incluidas en las tareas anteriores, podemos suponer una dedicación total de 1 hora.

GP5 - Memoria

La escritura de la memoria es progresiva, es decir, se irá escribiendo mientras se elaboran las tareas del proyecto. Esta consiste en toda la documentación que se irá generando mientras dure el trabajo. Se prevé una dedicación total de unas 50 horas a esta tarea.

GP6 - Defensa del proyecto

Una vez finalizado el proyecto y entregada la memoria, tal como indica la Facultad de Informática de Barcelona, se realizará la defensa del proyecto. Se dedicarán alrededor de 20 horas para preparar el material necesario, incluyendo la presentación.

GP7 - Reuniones con el tutor

Fuera de plazos establecidos y de dependencias, se realizará, a ser posible, una reunión semanal con el tutor del TFG, Felix Freitag, con tal de informar sobre el desarrollo del trabajo y decidir una ruta a seguir. Se incluye aquí el contacto por correo electrónico y las reuniones por *Meet*, que irá en aumento cuando avance el trabajo. Se prevé una dedicación total de 30 horas (1 hora semanal * 19 semanas + correos + reuniones de últimas semanas).

2.1.2 Análisis de tecnologías (AT)

En el conjunto de tareas llamado análisis de tecnologías, dedicaré el tiempo a reproducir el TFG de Jack Griffiths, antiguo alumno de la FIB. La idea aquí es obtener como resultado el proyecto realizado con propósito de utilizarlo de base para este.

AT1 - Investigación y preparación del entorno

La tarea se basa en investigar y preparar el entorno, tal como dice su propio nombre. Con esto vengo a referirme al hecho de informarse y comprender sobre las

tecnologías utilizadas anteriormente para la elaboración del sistema ya mencionado. También incluyo aquí la preparación de los sistemas a usar, al haber de preparar un pc con el sistema operativo de *linux* y la instalación del *Visual Studio* y *plugins* necesarios para su funcionamiento. Sin contar tiempos de espera de instalación puede suponer una duración total de 15 horas para ponerlo todo en marcha.

[AT2 - Aprender a usar las herramientas](#)

Una vez esté disponible el entorno a usar, el siguiente paso será aprender a usarlo. Para esto el propio Jack en su TFG explica los pasos para poner el sistema en marcha. Este explica el uso que le da a *Git* o como trabaja con *MQTT* (o *Mosquitto*), entre otras cosas. La tarea comprende también el tiempo necesario para el aprendizaje de *Visual Studio* y respectivos *plugins* para poner en marcha la configuración. En definitiva, se estima una dedicación total de unas 10 horas.

[AT3 - Entender la base del trabajo](#)

Como ya he comentado varias veces, este trabajo tiene como punto de partida, el punto final del trabajo de Jack. De manera obvia, al ser su trabajo apto para un TFG, podemos ver que tendrá una cierta complejidad tanto tecnológica como de comprensión, sumado a que está presentado en inglés. Él, por su parte, hizo un gran trabajo, por lo que dedicaré alrededor de 20 horas a entender qué, cómo y por qué lo hizo.

[AT4 - Implementación y prueba de funcionamiento de la base](#)

Una vez esté todo preparado para comenzar a programar como tal, se probará la implementación ya hecha que se tomará como base. No espero que todo salga correctamente la primera vez, por lo que aquí se aplica un bucle de prueba-error hasta encontrar la manera exitosa de poner a funcionar el sistema. Esta tarea se completará exitosamente al ver que nuestro sistema funciona exactamente como le funcionaba al estudiante anterior en sus días.

Se aproxima una cantidad de unas 30 horas de dedicación a este apartado.

[2.1.3 Propuestas del trabajo \(PT\)](#)

Este conjunto de tareas se hará en gran parte con la ayuda del tutor del TFG y tiene como fin evaluar qué objetivo propuesto es más necesario, factible o interesante para la sociedad actual.

PT1 - Comprensión de los problemas actuales

Para comenzar, se estudian las necesidades de los usuarios, a quién está dirigido el proyecto y que problemas les soluciona. Es una tarea de investigación totalmente independiente, al ser una mera búsqueda sobre los métodos utilizados actualmente y que diferencia este proyecto de investigación de estos.

Se aproxima una cantidad de unas 10 horas de dedicación a este apartado.

PT2 - Análisis sobre los distintos objetivos a conseguir

Como ya he comentado se han hecho varias propuestas para sacar adelante este trabajo (ver en el apartado 2.1 Objetivo y propuestas). Se han de investigar la viabilidad de cada una de estas ideas para empezar a trabajar con un objetivo específico en mente. Para esto haremos uso de la comunicación con el tutor con el fin de obtener la información de cada una. Se supone una duración de 15 horas.

PT3 - Evaluar y elegir el objetivo

De nuevo, comunicándome con el tutor, y teniendo las opciones sobre la mesa, se evaluarán las propuestas y se elegirá una en función de los siguientes puntos: necesidad, motivación, factibilidad y relevancia. Podemos aproximar 10 horas de dedicación.

2.1.4 Elaboración del proyecto (EP)

El conjunto de tareas elaboración del proyecto se basará en las mejoras aportadas por este proyecto. Es la fase de innovación y mejora que propondré para este trabajo.

EP1 - Investigar la tecnología y marcar subobjetivos

La tarea principal del bloque, una vez sepamos hacia dónde queremos ir y esté el objetivo decidido, se investigará sobre cómo sería posible implementarlo, como se hace actualmente. Diariamente se marcarán subobjetivos de manera que se tendrán que cumplir en plazos muy cortos. Estos podrían ser por ejemplo, en caso de que se decida aplicar la mejora de añadir sensores a la implementación (propuesta 2): informarse sobre el tipo de sensores, compra de estos, e integración en el sistema. Este es un tipo de tarea dinámica que dependerá de muchos factores, como el objetivo principal o los subobjetivos marcados, además de los posibles imprevistos que traigan estos. Se hará una aproximación muy abrupta de 50 horas del tiempo dedicado.

EP2 - Diseño e implementación de la nueva funcionalidad

Esta tarea podría haberse llamado, también, código. Posiblemente la tarea más extensa del trabajo, ya que se basa en “hacer la mejora” al trabajo. En esto entraría todo el tema de programar usando las mismas herramientas que ya se usaron una vez, ya sabiendo que queremos modificar y teniendo una base funcional. Se incluye en la tarea siguiente EP3 lo relacionado a corrección de errores además de las pruebas y mejoras que se vayan realizando. Esta tarea se podría dividir en 2, diseño e implementación, dependiendo la idea a seguir, pero las evaluaré cómo conjuntas. Se estiman un total aproximado de 100 horas.

EP3 - Pruebas de funcionamiento y depuración

La tarea de corrección de errores de nuestro código y la puesta a prueba con tal de que el sistema sea funcional. En definitiva, se intuye una dedicación total de unas 35 horas.

2.2 Recursos

Este proyecto se está desarrollando en pequeña escala, por lo que se limitarán, en general, los recursos que se usarán. Seré yo el encargado de todos los roles y el realizador de todas las tareas.

2.2.1 Recursos humanos

Aunque ya se haya especificado que el propio autor es el que llevará a cabo todos los roles, para el trabajo se separarán en 3: director de proyecto, analista y programador. El director de proyecto se encargará de las tareas de gestión, el analista de las tareas de investigación y propuestas y el programador del diseño e implementación.

2.2.2 Recursos materiales

Para llevar a cabo un trabajo de estas características, es necesario disponer de:

- Un espacio físico para el desarrollo (una oficina o habitación).
- Conexión a Internet.
- Ordenador con *Linux* como sistema operativo (se podría hacer en *Windows*, pero se complicaría un poco).

- Una cuenta de *Google* para usar las herramientas *Google Meet* y *Google Drive*.

2.3 Estimaciones y Gantt

2.3.1 Estimaciones

| ID | Tarea | Tiempo | Dependencias |
|-----------|------------------------------------------------------|-------------|---------------|
| GP | GESTIÓN DEL PROYECTO | 185h | - |
| GP1 | Contextualización y alcance | 32h | - |
| GP2 | Planificación | 27h | GP1 |
| GP3 | Presupuesto y sostenibilidad | 25h | GP2 |
| GP4 | Entrega final GEP | 1h | GP3 |
| GP5 | Memoria | 50h | - |
| GP6 | Defensa del proyecto | 20h | GP4, GP5 |
| GP7 | Reuniones con el tutor | 30h | - |
| AT | ANÁLISIS DE LAS TECNOLOGÍAS | 75h | - |
| AT1 | Investigación y preparación del entorno | 15h | - |
| AT2 | Aprender a usar las herramientas | 10h | AT1 |
| AT3 | Entender la base del trabajo | 20h | - |
| AT4 | Implementación y prueba de funcionamiento de la base | 30h | AT2, AT3 |
| PT | PROPUESTAS DEL TRABAJO | 35h | - |
| PT1 | Comprensión de los problemas actuales | 10h | - |
| PT2 | Análisis sobre los distintos objetivos a conseguir | 15h | PT1 |
| PT3 | Evaluar y elegir el objetivo | 10h | PT2 |
| EP | ELABORACIÓN DEL PROYECTO | 185h | AT, PT |
| EP1 | Investigar la tecnología y marcar subobjetivos | 50h | PT3 |
| EP2 | Diseño e implementación de la nueva funcionalidad | 100h | AT4, EP1 |
| EP3 | Pruebas de funcionamiento y depuración | 35h | EP2 |

| | | | |
|---|--------------|-------------|---|
| - | TOTAL | 480H | - |
|---|--------------|-------------|---|

Tabla 1: Tabla resumen de las tareas. Elaboración propia.

2.3.2 Diagrama de Gantt inicial

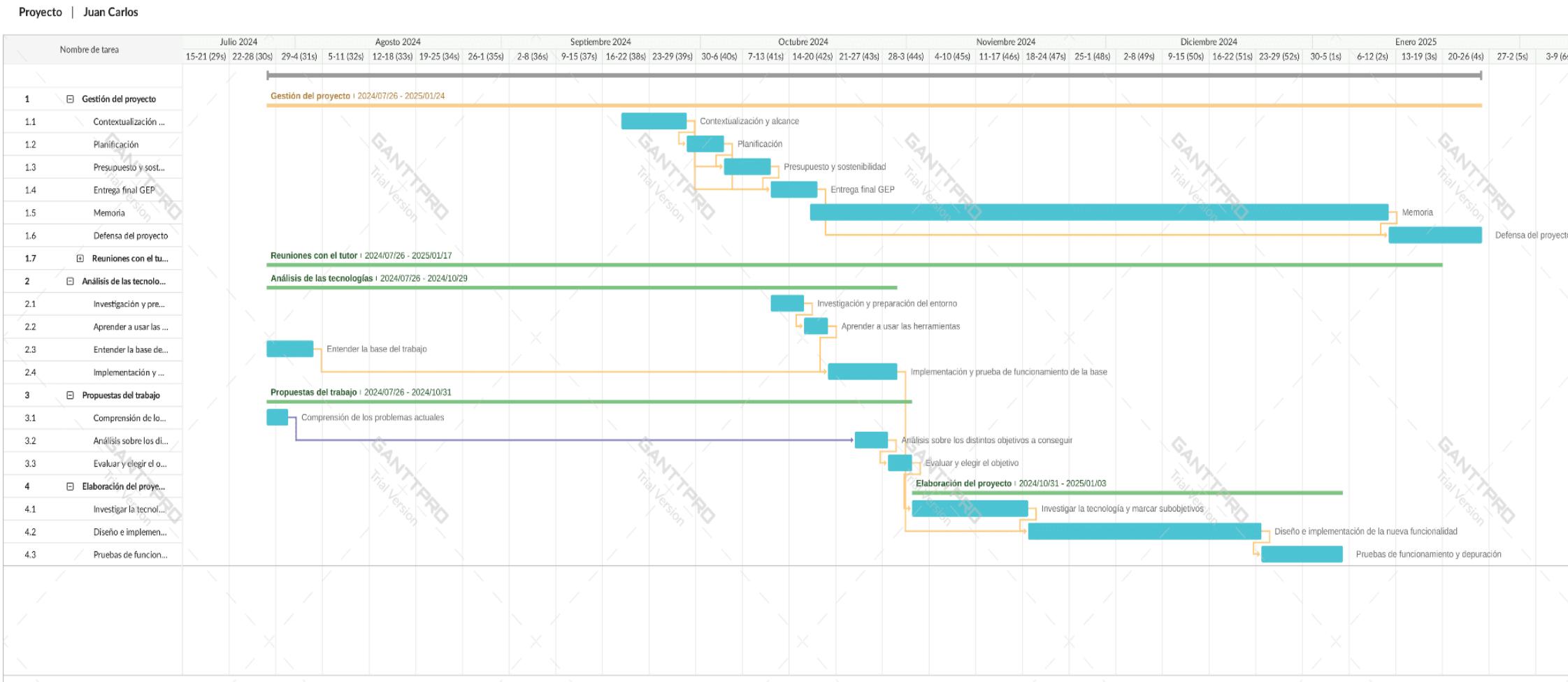


Figura 2: Diagrama de Gantt inicial del proyecto. Elaboración propia usando GanttPro.

2.4 Gestión del riesgo: Planes alternativos y obstáculos

Tal como dice en el apartado 2.3 Riesgos y obstáculos de la anterior entrega:
Para el proyecto se han identificado los siguientes obstáculos y/o riesgos que pueden afectar al desarrollo de este, para evitar que sucedan o solucionarlos expongo lo siguiente:

- Fecha límite: tener en cuenta el diagrama de *Gantt* especificado e ir adaptando las fases del proyecto a medida que se vayan haciendo.
- Falta de experiencia y disponibilidad de la información: para minimizar este impacto, ya comencé en julio a tratar de entender el trabajo que ya había hecho.
- Errores o retrasos: se intentarán minimizar, pero en este caso, el problema es inevitable. En caso de fallo de las aplicaciones, se intentará avanzar en paralelo con otra tarea en medida de lo posible.
- Fallos inesperados o incontrolables: como ya dice el nombre son incontrolables. Aunque se van a preparar varios pc listos para la ejecución del trabajo en caso de que uno falle. Si fallan las placas, se contactará de inmediato con el tutor.
- Desconocimiento del final: el trabajo puede explicar y justificar el por qué esa solución no es viable. Aun así se podría intentar, en caso de que haya tiempo suficiente, de cambiar la propuesta de implementación.

2.5 Planificación definitiva

2.5.1 Diagrama de Gantt final

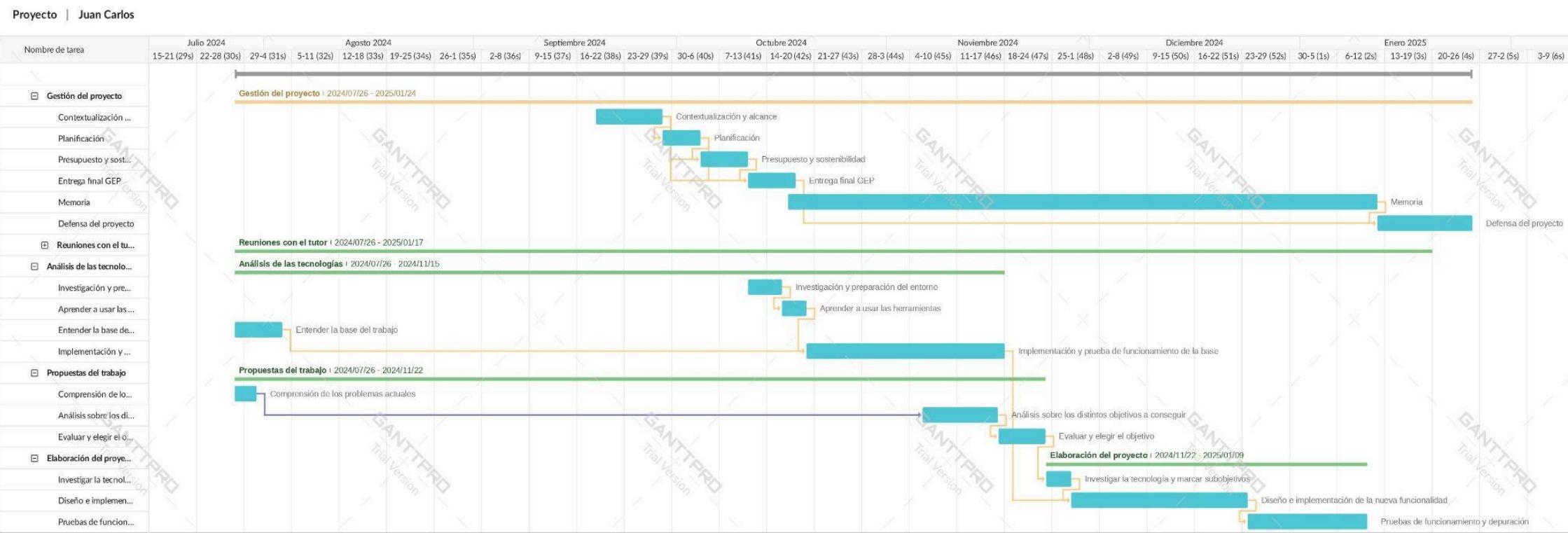


Figura 3: Diagrama de Gantt final del proyecto. Elaboración propia usando GanttPro.

2.5.2 Problemas

Como se puede ver en el diagrama, y en comparación con el anterior, se ha aumentado la franja de “Implementación y prueba de funcionamiento de la base” en 3 semanas. Hubo 2 problemas que requirieron mucho tiempo:

- Primero, al ser una de las 3 placas proporcionadas, más antigua que las otras 2, el ordenador no la detectaba, por lo que, no se podía cargar el programa dentro de esta.
- Segundo, yo, el autor, tenía el pensamiento de que estaba haciendo algo de forma incorrecta, al ver que no podía acceder al servidor *EMQX*. Resulta que la combinación de usuario/contraseña que había indicado Jack en su documento eran incorrectas, por lo que, no pude acceder hasta preguntar al tutor por la contraseña nueva.

Hablé con Felix sobre el proyecto y decidimos ir por la opción que parecía que más se ajustaba a mi experiencia adquirida hasta ahora, además de ser la que más me motivaba (implementar sensores). Después de ir a buscar los materiales, se hizo un trabajo intensivo para poner en marcha esta mejora, cosa que, se consiguió antes de lo esperado (el 8/12/2024 el sistema ya mostraba los valores de los sensores).

Dado que la implementación de la base se alargó 3 semanas y el diseño/implementación de una nueva funcionalidad se acorta aproximadamente la misma cantidad de tiempo, el cómputo global de horas y coste podría dejarse como está (recordar también que estas 2 funciones las hace el supuesto programador, de modo que hace las mismas horas, pero repartidas de forma distinta) .

2.5.3 Estado actual del proyecto (9/12/2024)

Como ya se ha dicho, actualmente el proyecto tiene integrada la mejora de los sensores funcionando correctamente. El tener mucho tiempo por delante, nos lleva a intentar poner en marcha otra de las propuestas (hacer el *deployment* a Internet), que se prevé que esté en funcionamiento antes de acabar el año. A partir de ese momento, solo habría que hacer pruebas de funcionamiento.

3. Gestión económica y sostenibilidad

3.1 Presupuesto

3.1.1 Recursos humanos

En la parte de recursos humanos se calculan los sueldos de los empleados. En realidad, el proyecto lo llevará a cabo únicamente una persona, yo, pero, supondremos 3 roles diferentes desarrollados por 3 personas diferentes: jefe de proyecto, analista y programador. Si nos fijamos en el diagrama de *Gantt* proporcionado y asociamos tareas a empleados, a grandes rasgos, el jefe de proyecto se encargaría de toda la parte de gestión de proyecto, el analista se preocupa por el estudio de las tecnologías y propuestas viables, y el programador en hacer el código.

A continuación, en la tabla detallamos el trabajador indicando su salario bruto y neto, es decir, antes y después de aplicar los impuestos correspondientes. Además, se tiene en cuenta el coste de la seguridad social con un factor de 1.3 del salario de cada empleado que se indica en la última columna.

| | Salario Bruto | Salario neto | Salario del personal (salario bruto x 1.3) |
|--------------------------|---------------|--------------|--------------------------------------------|
| Jefe de proyecto TI (JP) | 36.00€/h | 28.80€/h | 46.80€/h |
| Analista (A) | 18.75€/h | 15.00€/h | 24.78€/h |
| Programador (P) | 14.76€/h | 11.81€/h | 19.19€/h |

Tabla 2: Tabla de sueldos elaborada manualmente. Fuente: página web de indeed. [12]

La tabla siguiente muestra en detalle la repartición de las tareas a realizar en el proyecto con su coste. Se supondrá que el analista conoce la base del trabajo teniendo algo de base en programación, pero no se encargará de ninguna tarea de escribir código.

En la columna “empleados”, se marcan las horas que supone para cada uno. Por ejemplo, la memoria, es un trabajo compartido, por lo que se repartirán las horas, pero el tiempo de reunión supone la comunicación entre jefe-analista, jefe-programador, analista-programador, y la comunicación entre los 3, por lo que la suma de las horas, supera el tiempo de la tarea. Dicho de otra manera, 1 reunión de los 3 empleados de 1 hora, en tiempo de tarea es 1 hora, pero si miramos desde el

punto de vista del empleado es 1 h cada uno, por lo que, si miramos el coste, está indicado la suma del precio de la hora de cada uno de los agentes.

| ID | Tarea | Tiempo | Empleados | Coste |
|-----------|------------------------------------------------------|-------------|-------------------------------|----------------|
| GP | GESTIÓN DEL PROYECTO | 185h | | 7.280 € |
| GP1 | Contextualización y alcance | 32h | JP | 1.498 € |
| GP2 | Planificación | 27h | JP | 1.264 € |
| GP3 | Presupuesto y sostenibilidad | 25h | JP | 1.170 € |
| GP4 | Entrega final GEP | 1h | JP | 47 € |
| GP5 | Memoria | 50h | JP(15h), A(15h), P(20h) | 1.458 € |
| GP6 | Defensa del proyecto | 20h | JP | 936 € |
| GP7 | Reuniones con el tutor | 30h | JP, A, P (20h/cada uno) | 907 € |
| AT | ANÁLISIS DE LAS TECNOLOGÍAS | 75h | | 1.977 € |
| AT1 | Investigación y preparación del entorno | 15h | A, P (7.5h/cada uno) | 330 € |
| AT2 | Aprender a usar las herramientas | 10h | P | 192 € |
| AT3 | Entender la base del trabajo | 20h | A, P (20h/cada uno) | 879 € |
| AT4 | Implementación y prueba de funcionamiento de la base | 30h | P | 576 € |
| PT | PROPUESTAS DEL TRABAJO | 35h | | 868 € |
| PT1 | Comprensión de los problemas actuales | 10h | A | 248 € |
| PT2 | Análisis sobre los distintos objetivos a conseguir | 15h | A | 372 € |
| PT3 | Evaluar y elegir el objetivo | 10h | A | 248 € |

| EP | ELABORACIÓN DEL PROYECTO | 185h | | 3.690 € |
|-----------|---------------------------------------------------|-------------|-----------------------|-----------------|
| EP1 | Investigar la tecnología y marcar subobjetivos | 50h | A,P (25h/cada uno) | 1.099 € |
| EP2 | Diseño e implementación de la nueva funcionalidad | 100h | P | 1.919 € |
| EP3 | Pruebas de funcionamiento y depuración | 35h | P | 672 € |
| - | TOTAL | 480H | | 13.815 € |

Tabla 3: Tabla con los costes estimados por tarea. Fuente: elaboración propia.

3.1.2 Recursos hardware

Los dispositivos *hardware* necesarios para el proyecto se detallan a continuación. Ahora vamos a suponer el caso real, solo 1 persona, el autor, trabaja en el proyecto, por lo que se contabilizará el material para únicamente esta persona. Se tiene en cuenta el portátil para la elaboración del proyecto y el pc de torre de sistema de uso en caso de fallo, indicado como *hardware* secundario. Los precios de cada producto son en el momento de adquisición, actualmente, podemos encontrar algunas variaciones.

| Hardware | Coste |
|-----------------------------------------|----------------|
| Acer Nitro AN515-58 Intel Core i7-12700 | 779 € |
| Raton Technet | 15 € |
| Placas T-Beam (3 unidades) | 44 x 3 = 132 € |
| Hardware Secundario | |
| Torre Asus Intel Core i7-7700 | 1021 € |
| Monitor Gaming Koorui | 157 € |
| Raton+teclado BAKTH | 32 € |
| Total | 2136 € |

Tabla 4: Tabla de precios en el momento de adquisición. Fuente: elaboración propia.

Según la tabla de amortización simplificada [13], podemos amortizar equipos informáticos hasta en un máximo de 4 años. Suponemos 220 días laborables en 1 año y 8 h/día de trabajo. Aquí no se tiene en cuenta el coste de haber de renovar el equipo, dado que, por ejemplo, un ratón, es poco probable que aguante 4 años. Teniendo un coste total de 2136 € y amortizando en 4 años y 480 horas de proyecto:

$$\text{NUMERADOR} = \text{Coste total del hardware} * \text{Número de horas de uso durante el proyecto}$$

$$\text{DENOMINADOR} = \text{Vida útil del equipo autorizada por Hacienda (4 años)} * \text{Días laborables por año (220)} * \text{Horas de uso del equipo por día (8 horas)}$$

$$\frac{2136 \text{ €} * 480 \text{ horas}}{4 \text{ años} * 220 \text{ días} * 8 \text{ horas/día}} \simeq 145.63 \text{ €}$$

Entonces, el coste estimado total de *hardware* es 145.63 €

3.1.3 Recursos software

Se usará *software* libre (*Google Meet* para las reuniones y *Visual Studio* como entorno de desarrollo) para la realización del proyecto, por lo que el coste será nulo.

3.1.4 Costos genéricos

Los costes de alquiler de local, agua, luz, Internet y material de oficina son también contabilizables. El proyecto no necesita de un local específico para llevarse a cabo, por lo que, para hacer el cálculo del local, usaremos el caso real. El autor hará el proyecto desde su habitación la cual mide aproximadamente 16 m², en su casa, en Vilanova del Camí. Consultando el precio del metro cuadrado en la zona [14], vemos un coste de 1223 €/m² a la hora de la compra y 7.3€/m² en alquiler. Dado que la casa está en propiedad, el símil será la compra de la habitación: 1223 €/m² * 16 m² = 19568 €. Sabiendo que esto dispararía el coste del proyecto, se asumirá que se trabajará en una habitación de las mismas condiciones, pero estando en alquiler:

$$\frac{7.3 \text{ €}}{1 \text{ mes } 1 \text{ m}^2} * 16 \text{ m}^2 = 116.80 \text{ €/mes}$$

Al ser un alquiler destinado a uso profesional, deberíamos tener en cuenta el IVA del 21%, por tanto:

$$116.80 \text{ €/mes} * 1.21 \text{ IVA} = 141.33 \text{ €/mes}$$

El coste de agua en el pueblo es de 3.66 €/m³ [15] y el de luz tiene una media de 113.39 €/MWh [16]. Además se utilizará una tarifa de Internet de fibra óptica de O2, que ofrece 300Mb por 27 €/mes [17].

Suponiendo 20 días laborables al mes y teniendo en cuenta que:

- Una persona usa una media de 136 litros/día [18] de agua, de los cuales usaremos:

$$\frac{136 \text{ litros}}{1 \text{ día}} * \frac{3.66 \text{ €}}{1 \text{ m}^3} * \frac{1 \text{ m}^3}{1000 \text{ litros}} * \frac{1 \text{ día}}{24 \text{ horas}} * 8 \text{ horas de trabajo} \simeq 0.17 \text{ € cada jornada laboral}$$

- Un ordenador portátil consume 60 kWh en 8 horas de trabajo [19].

$$\frac{113.39 \text{ €}}{1 \text{ MWh}} * \frac{1 \text{ MW}}{1000 \text{ kW}} * 60 \text{ KWh} \simeq 6.80 \text{ € cada jornada laboral}$$

Haciendo las matemáticas, obtenemos que cada día, entre luz y agua, gastamos $0.17 + 6.8 = 6.97 \text{ €}$, por lo que al mes, sería un total de 139.40 €/mes. Sabiendo ahora que el proyecto dura 5 meses:

$$(141.33 \text{ €/mes alquiler} + 27.00 \text{ €/mes internet} + 6.97 \text{ €/mes agua y luz}) * 5 \text{ meses} \simeq 876.50 \text{ €}$$

3.1.5 Contingencia

El hablar sobre el coste de contingencia es con el fin de cubrir costes no previstos. Para facilitar los cálculos y agilizar el proceso, fijamos el porcentaje en 10%. En la tabla siguiente se muestran los costes calculados anteriormente con su coste de contingencia total.

| Recursos | Coste | Contingencia |
|---------------|----------|--------------|
| Humanos | 13.815 € | 1.381 € |
| Hardware | 146 € | 15 € |
| Genéricos | 877 € | 88 € |
| Total: | | 1.484 |

Tabla 5: Tabla de costes estimados por contingencias. Fuente: elaboración propia.

3.1.6 Imprevistos

Tal como se ha indicado anteriormente (ver en el apartado 2.1 Objetivo y propuestas o en el apartado 5. Gestión del riesgo), hay la posibilidad de que aparezcan imprevistos que puedan no solo afectar al tiempo y fechas de entrega, sino que, también pueden cambiar el presupuesto por uno mayor en caso de que sucedan. Los imprevistos descritos se pueden resumir en 2: inexperiencia y fallas de hardware.

- Inexperiencia: en el caso de la inexperiencia y desconocimiento del temario que envuelve este proyecto, nos encontramos en que es bastante probable que se haya de aumentar las horas de trabajo. Al haber una fecha que marca el final del proyecto, no se podría retrasar, por tanto, no se trabajaría después de este punto, y habría la posibilidad de hacer horas extra. Como ya se ha dicho, la probabilidad de que ocurra un retraso por inexperiencia es alta, y se le asignará un valor de un 25%, no del todo arbitrario, ya que, el autor, ya ha trabajado anteriormente con proyectos *Arduino* y el protocolo *MQTT*. Nos ponemos en el caso del programador (aunque también podría haber sido el analista, pero este último no haría tantas horas): la persona encargada de hacer el código invierte 67.5 horas de su tiempo para el “Análisis de las Tecnologías”. Suponiendo que necesita 1.5 veces el tiempo previsto, obtenemos un total de 101.25 horas, o lo que es lo mismo, 33.75 horas extra. Cogiendo el valor bruto de lo que se ha de pagar a este programador (19.19 €/h) y multiplicando por estas 33.75 horas extra, tenemos que pagar 647.66 € extra por este trabajo.
- Fallas de *hardware*: aunque sea menos probable, también se tiene en cuenta los problemas que puedan derivar del *hardware*. Se ha de decir que, ya que se tiene un “ordenador de emergencia” en caso de que tengamos una falla, no habrá de reemplazar un ordenador, sino, comenzar a usar el otro. Independientemente del uso, se intuye que se habrá de arreglar/comprar otro ordenador para su sustitución. Para las fallas de *hardware* usaremos el valor total del *hardware* necesario. Esto incluye el portátil, ratón y placas, dejando fuera el ordenador de sobremesa. Se supondrá una probabilidad de suceso del 5%.

A continuación, se expresa en la tabla el coste que tendrían estos imprevistos:

| Imprevisto | Coste | Probabilidad del suceso | Coste final |
|---------------------------------|-------|-------------------------|--------------|
| Aumento en el tiempo de trabajo | 648 € | 25% | 162 € |
| Avería en el <i>hardware</i> | 924 € | 5% | 46 € |
| Total: | | | 208 € |

Tabla 6: Tabla de costes estimados por imprevistos. Fuente: elaboración propia.

3.1.7 Presupuesto final

Seguidamente se presenta un resumen de todos los costes necesarios para el desarrollo del trabajo.

| Tipo de coste | Coste |
|--------------------------|-----------------|
| Recursos humanos | 13.815 € |
| Recursos <i>hardware</i> | 2.136 € |
| Costes genéricos | 877 € |
| Contingencia | 1.484 € |
| Imprevistos | 208 € |
| Presupuesto final | 18.520 € |

Tabla 7: Tabla del presupuesto final estimado. Fuente: elaboración propia.

Sabemos que el desarrollo funcional de este proyecto puede provocar una gran eficiencia económica, por lo que, en el caso de que salga bien, podría ahorrar una gran cantidad de dinero a algunas empresas. Por otra parte, en el caso de que el proyecto se quede en una fase meramente académica o, fracase, podríamos argumentar la no viabilidad de este sistema, lo que conlleva a explorar otras opciones. De todas formas, independientemente del resultado, podemos decir que el proyecto es viable, dadas las dimensiones a las que se dirige, y como se suele decir, “quien no arriesga, no gana”.

3.2 Control de gestión

El objetivo de este apartado es comparar y evaluar desviaciones entre el presupuesto y el coste real que ha necesitado el proyecto. A medida que el proyecto vaya avanzando, se podrá ir poniendo en marcha calculándolo. Para calcular estas desviaciones, en general, se utiliza el dato estimado y se le resta el dato real. Por lo tanto tenemos las siguientes desviaciones (en totales):

- En tiempo de realización:

$$\text{horas estimadas tarea}_i - \text{horas reales tarea}_i$$

- En el total de horas del proyecto:

horas totales estimadas - horas totales reales

- En el coste de realización de las tareas:

*(horas estimadas - horas reales) * coste de las tareas (desvío en eficiencia)*

o

coste estimado de las tareas - coste real de las tareas

- En el coste de recursos *hardware*:

coste estimado del hardware - coste real del hardware

- En el coste de los imprevistos:

coste estimado de imprevistos - coste real de imprevistos

- En la desviación total del presupuesto:

coste total estimado - coste total real

Se utilizarán las anteriores desviaciones de costes para hacer el control de gestión del proyecto.

3.3 Informe de sostenibilidad

El informe de sostenibilidad se ha convertido en un requisito común en cualquier proyecto actual. En este documento, analizaremos los aspectos de sostenibilidad ambiental, económica y social en relación con el proyecto.

3.3.1 Autoevaluación

Empezamos la parte de sostenibilidad haciendo un resumen sobre la encuesta contestada [20]. En esta se evalúan las competencias usando afirmaciones y habiendo de responder si: no estás nada informado, lo estás poco, bastante o mucho. A continuación, se van a explicar los conocimientos previos del autor sobre este tema.

En cuanto a conocimientos generales se refiere, estoy bastante informado sobre los conceptos de sostenibilidad y desarrollo sostenible y conozco varios enfoques económicos, aunque desconozco los roles y derechos que tienen los agentes que se dedican a estos temas en mi ámbito.

En la encuesta también se pregunta sobre problemáticas sociales, económicas y/o ambientales de la sociedad actual. En este caso, estoy bastante capacitado, ya que aunque conozca poco/nada sobre iniciativas internacionales tales como la Agenda 2030, conozco bien las principales causas, consecuencias y agentes implicados, además de ser capaz de reflexionar y relacionar sobre los problemas de sostenibilidad con los métodos para afrontarlos.

En relación al impacto ambiental de los productos y servicios (segundo punto), también estoy bastante informado desde mi punto de vista, al conocer técnicas para medir el impacto medioambiental, reducir, reutilizar y reciclar. Soy consciente de que pasa con los productos de mi ámbito en el mundo, por lo que esta competencia podría darse como superada.

Pasamos a hablar de los conceptos de salud, seguridad y justicia social, aunque este sea uno de los primeros proyectos que tenga en cuenta estos criterios, desde mi perspectiva, podría decirse que estoy muy informado sobre este tema. También son parte de este temario los indicadores de impacto social. Es la parte que menos domino, indicando que conozco poco o nada sobre métricas, medidas o indicadores de este tema.

Seguimos con viabilidad económica, el cuarto tema. Gracias a asignaturas como Entorn Econòmic y Empresarial, Negoci Electrònic o Projecte Aplicat a l'Enginyeria (abreviados EEE, NE y PAE, respectivamente) he aprendido sobre metodologías para estimar la viabilidad económica de los proyectos, por lo que soy conocedor de herramientas como CANVAS, DAFO, y los planes de negocios. En este contexto me autoevaluaría como muy conocedor. En la parte de gestión de recursos, también gracias a lo anterior, me evaluaría como muy conocedor o bastante, al menos.

Cuando se habla de proyectos, productos y servicios relacionados con mi ámbito profesional, podría decirse que estoy entre poco y bastante conocedor del tema, estaría en un punto medio, estando al tanto de los enfoques en producción, consumo y reciclaje, pero en contraparte, no ser capaz de realizar el desmantelamiento del proyecto con tal que sea sostenible.

El punto 6 habla sobre interacciones producidas con otros agentes en los proyectos. En este caso, también me evaluaría como punto medio, es decir, no soy bastante conocedor del tema, pero tampoco poco conocedor de este.

Por último, se pregunta por principios deontológicos y principios éticos. Esta parte la conozco bastante bien debido a cursar la asignatura Aspectes Socials y Mediambientals de la Informàtica. En esta se explican todo lo que concierne a estos temas tales como equidad, justicia, o derechos humanos.

3.3.2 Análisis de la sostenibilidad del proyecto

Seguidamente se presentarán una serie de preguntas y respuestas sacadas de la Figura 3.Preguntas de la matriz de Sostenibilidad del TFG del documento “Mòdul 2.6 - El informe de sostenibilidad.pdf”.

3.3.2.1 Ambiental

PPP - ¿Has estimado el impacto ambiental que tendrá la realización del proyecto? ¿Te has planteado minimizar el impacto, por ejemplo, reutilizando recursos?

Se estima que el proyecto en desarrollo tendrá un impacto ambiental mínimo durante su ejecución. Los recursos de *hardware* necesarios son el pc con sus componentes (teclado, ratón) y las placas *T-Beam*. En mi caso, usaré un portátil para la realización del trabajo, por lo que no es necesario el teclado, ya que viene incorporado. También usaré un pc de torre como sistema auxiliar en caso de fallada. En este podemos contar también, además de la torre, un teclado, ratón y pantalla. Se utilizarán sus equipos personales (tanto el portátil como la torre) además de las placas proporcionadas por el tutor. Sabiendo que ya se disponía de las placas (que también usó Jack en el anterior proyecto) y, dado que los componentes *hardware* ya pertenecían al autor en un principio, es bastante difícil minimizar más el impacto ambiental.

Vida Útil - ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará ambientalmente tu solución a las existentes?

Se podría llegar a argumentar que la implementación de estas placas *T-Beam* tendría un impacto positivo, aunque no demasiado grande. Actualmente, hay sistemas implementados en campos de cultivo, por ejemplo, que se dedican a medir variables tales como temperatura, luz o humedad. Este trabajo podría reducir el número de componentes necesarios y, por tanto, la energía consumida para este fin.

3.3.2.2 Económico

PPP- ¿Has estimado el coste de la realización del proyecto (recursos humanos y materiales)?

El presupuesto estimado para este proyecto está detallado posteriormente en este documento (sección 2.1 Presupuesto). En cuanto a los recursos humanos, se suponen 3 trabajadores: jefe de proyecto, analista y programador. Respecto a los recursos materiales, como *hardware* y otros gastos, el coste es bastante reducido. La suma total no es elevada si se consideran los cinco meses de trabajo y los posibles beneficios de implementar nuestro sistema una vez finalizada la investigación.

Vida Útil - ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará económicamente tu solución a las existentes?

Como ya se ha dicho anteriormente, este trabajo podría ayudar a diversos ámbitos mediante los dispositivos *IoT*. Dado que al montar el sistema (en caso de que sea funcional y aplicable), se necesitará un menor número de placas, las empresas comprarán un menor número de dispositivos, por lo que, se gastará menos en el hecho de comprar el dispositivo, además del uso energético que tienen estos. El problema viene con las empresas que ya tengan un método implementado. Estas, en caso de adoptar nuestra solución, habrán de cambiar todo el *hardware* existente, por lo que supondría un gasto posiblemente grande e innecesario, dependiendo de las características.

3.3.2.3 Social

PPP - ¿Qué crees que te va a aportar a nivel personal la realización de este proyecto?

Desde mi punto de vista, el proyecto generará un impacto positivo en mi desarrollo personal y profesional. Al realizar este trabajo de fin de grado obtendré evidentemente, experiencia, por lo que servirá de guía en caso de futuros proyectos. De la misma manera como está pensado el trabajo, se han puesto en práctica las tareas de planificación del tiempo, además de la parte autodidáctica del proyecto en la que tendré que buscar, estudiar e investigar los recursos necesarios para llevarlo a cabo. Se podría resumir lo anteriormente mencionado como esfuerzo, tiempo y dedicación invertida para obtener un crecimiento personal y académico.

Vida Útil - ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará socialmente (calidad de vida) tu solución a las existentes?

Actualmente, el enfoque de las redes *LoRa* tiene limitaciones de cobertura, especialmente en áreas remotas o con obstáculos. Las redes en malla (*mesh*) *LoRa* permiten la comunicación y extensión de la cobertura. Sin embargo, las implementaciones existentes, como *LoRaWAN* o proyectos como *Meshtastic*, aún están en desarrollo y no ofrecen soluciones completamente optimizadas para redes en malla.

Este proyecto puede ofrecer una red *LoRa mesh* más robusta y eficiente. Por tanto, puede facilitar la comunicación en situaciones de emergencia, mejorar la productividad en áreas rurales a través de sensores distribuidos en agricultura, y potenciar el desarrollo de ciudades inteligentes con conectividad para dispositivos

IoT. Además, en el ámbito ecológico, podría mejorar el monitoreo ambiental y la respuesta ante desastres.

Vida Útil - ¿Existe una necesidad real del proyecto?

En cuanto a la necesidad real del proyecto, sabemos que, muchas zonas rurales carecen de una infraestructura de comunicación adecuada, por lo que, las redes *LoRa mesh* ofrecerían una solución asequible.

Además, si tenemos en cuenta problemas como el cambio climático, esta solución es bastante relevante para monitorizar valores ambientales.

Por último mencionar que no solo podría servir para mejorar la calidad de vida, sino también para garantizar la seguridad y la sostenibilidad.

4. Conjunto tecnológico

En esta sección se hará hincapié en una visión general del conjunto de tecnologías que se usarán en el proyecto. Es esencial para este, entender el funcionamiento de las herramientas y componentes que se utilizan para apreciar la solución propuesta (figura 3). A continuación, por apartados se explicará cada una de las tecnologías: las que usó Jack en su momento para reproducir el trabajo, y las que he usado yo, el autor, para replicar y expandir su trabajo.

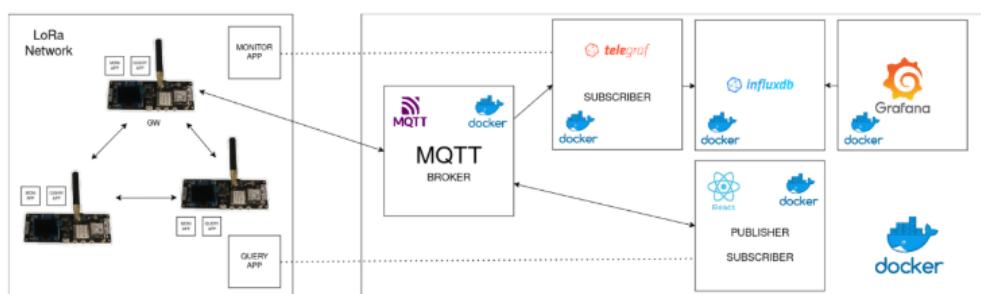


Figura 4: Stack tecnológico del proyecto de Jack. Fuente: TFG de Jack Griffiths.

4.1 Placas *T-Beam*

Lo primero que quiero mencionar es el *hardware*, es decir, las placas *T-Beam*. Este proyecto usará las *Lilygo T-Beam*, producidas por la misma empresa *Lilygo*, una compañía que se dedica a producir dispositivos que son usados en *IoT*.

A continuación se muestra una imagen del *T-Beam* en funcionamiento, conectado a una fuente de alimentación (en este caso al pc), y con la librería *LoRaMesher* cargada. Entre otras cosas, la placa dispone de una pantalla *OLED* en la que se muestra información de la conexión como su identificador de la red en hexadecimal

(“número identificador”), la *routing table* del nodo, y, en caso de que el nodo sea un *gateway*, su *IPv4*.

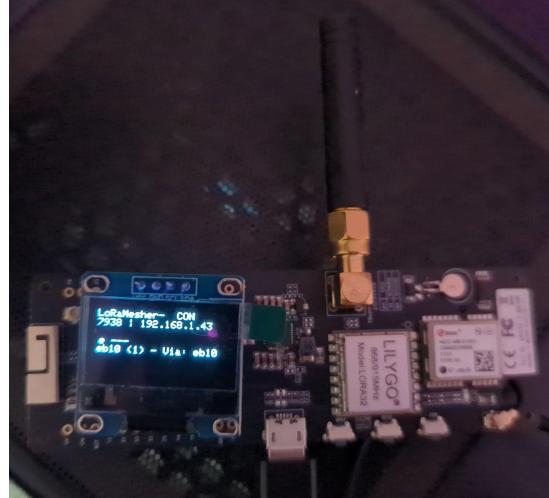


Figura 5: T-Beam con el programa cargado y en funcionamiento. Fuente: Elaboración propia.

4.2 LoRa

Pasamos a la parte de *LoRa*, aquí pretendo diferenciar los conceptos de redes *LoRa Mesh* que tiene que ver más con la parte *hardware* y la librería *LoRaMesher* que es relativo al *software*.

4.2.1 Redes *LoRa Mesh*

La red *LoRa Mesh* que se utiliza en este proyecto está compuesta por las 3 placas *T-Beam* como se muestra en la imagen anterior (figura 3). Estas placas se comunican entre ellas usando el protocolo de *LoRa* mencionado y se conectan a Internet mediante un *gateway*, función que hará una de las placas mediante el *EMQX broker*. Este *gateway* se podrá comunicar con las aplicaciones web directamente mediante el *broker* de *MQTT*, mientras que, las otras 2 placas deberán pasar por esta tercera con el fin de hacer esta comunicación. Para conseguir esto, se cargará en cada una de las placas el mismo programa (con la librería *LoRaMesher* incluida), pero, a una de ellas se le indicará una combinación SSID/contraseña existente para poder conectarse a Internet, mientras que a las otras 2, esta combinación no existirá o será incorrecta.

4.2.2 Librería *LoRaMesher*

4.2.2.1 Introducción y diseño

La librería *LoRaMesher*[5] fue desarrollada por un estudiante de la Universidad Politécnica de Cataluña (UPC), Joan Miquel Solé. Su propósito principal es controlar

cómo se envían los paquetes entre nodos en una red *LoRa*, en nuestro caso, con dispositivos *TTGO-Tbeam*. Esta librería es esencial para el proyecto, ya que se busca ampliar sus funcionalidades para mejorar la comunicación entre nodos.

LoRaMesher funciona en placas integradas con un sistema en chip (SoC) y un radio *LoRa* de un canal (*single-channel*). Usa *FreeRTOS* para la gestión de tareas y *RadioLib* para la configuración del protocolo *LoRa*. Además, aprovecha la estructura de los paquetes *LoRa* para añadir un "encabezado *LoRaMesher*" al valor de carga útil, que incluye:

- La dirección de destino.
- La dirección del nodo emisor.
- El tipo (enrutamiento o datos).
- Tamaño del mensaje.

4.2.2.2 Tipos de mensajes

El protocolo permite dos tipos de mensajes: de enrutamiento (para actualizar tablas de rutas en los nodos) y de datos (para el intercambio de información entre aplicaciones):

- Mensajes de enrutamiento: se envían periódicamente a todos los nodos, con la dirección de destino 0xFF (difusión). Cada nodo receptor revisa si el emisor ya está en su tabla de rutas; si es así, actualiza la entrada, y si no, lo agrega. Luego procesa la tabla de rutas del emisor y actualiza o añade entradas en su propia tabla antes de eliminar el paquete.
- Mensajes de datos: Estos contienen la información de la capa de aplicación. Si el nodo receptor es el destino, el mensaje se envía a la aplicación; si es solo un paso intermedio, reenvía el mensaje al siguiente nodo. Si no se aplica ninguna de estas condiciones, el mensaje es eliminado.

4.2.2.3 Herramientas de depuración

Para probar *LoRaMesher*, se instalaron herramientas como *Mosquitto* para suscribirse a temas del nodo de *gateway*, y *PlatformIO* para monitorizar la salida de nodos conectados a puertos específicos. Estas herramientas permiten detectar

errores en la comunicación y el funcionamiento interno de *LoRaMesher*, como problemas de conexión.

4.3 Visual Studio & PlatformIO

El primer paso del proyecto fue instalar un *IDE*. Para ello, se eligió *Visual Studio* por su capacidad de añadir extensiones. En este caso, se utiliza *PlatformIO*, una extensión de *Visual Studio Code* encargada de gestionar la carga de la librería *LoRaMesher* en las placas *T-Beam*.

Para poder usar el *PlatformIO*, es necesario un archivo llamado *platformio.ini*, que contiene toda la información relativa a la configuración de distintas opciones, como la estructura del proyecto o la información de las placas a utilizar. Esta extensión permite no solo crear proyectos, sino también importar y abrir proyectos ya existentes junto a las librerías y placas con soporte en este entorno, por lo que resulta ideal para este trabajo. Como mejora en términos de calidad de vida del usuario, *PlatformIO* dispone de un sistema de detección de puertos *USB*, con el cual el desarrollador puede ver en todo momento los dispositivos conectados al pc y su información con el fin de cargar el programa en un dispositivo u otro.

Como cabría esperar de un *IDE*, *PlatformIO* también ofrece funcionalidades bastante útiles para compilar y cargar el proyecto, borrar el código existente en una placa o probar y monitorear esta a través de los botones *Build* (ícono de un “*tick*”), *Upload* (ícono de una flecha), *Clean* (ícono de papelera), *Test* (ícono de un matraz de laboratorio) y *Monitor* (ícono de un enchufe), respectivamente.

4.4 MQTT

Con el fin de establecer un “esqueleto” para la comunicación entre nuestras aplicaciones *LoRaMesher* y las aplicaciones web, se hará uso de un *broker MQTT*. Un *MQTT broker* es un sistema *backend* que gestiona el flujo de mensajes entre sus clientes *MQTT*. El funcionamiento de este es bastante sencillo de comprender: los denominados “*publisher*” publican mensajes a un cierto tema (llamado “*topic*”), que serán recibidos por todos los “*client*” suscritos a este tema. Por ejemplo, si tenemos un sensor de temperatura, le asignaremos un *publisher* que vaya publicando cada 5 segundos el valor que marca el sensor a un *topic* llamado “temperatura”. Desde nuestro *backend*, colocaremos un *subscriber* a este *topic*, de modo que cada 5 segundos, nos llegará el valor que registre el sensor. Gracias a este valor obtenido, podremos realizar tareas interesantes según lo requiera el proyecto, como enviar alertas en caso de que la temperatura supere ciertos límites.

Una vez entendido su funcionamiento, también es importante saber que nos ofrece. Los *MQTT broker* proporcionan:

- Enrutamiento de mensajes: Cada dispositivo funciona como un nodo con tablas de enrutamiento, lo que permite redirigir los mensajes a través de la ruta más óptima.
- Escalabilidad: La mayoría de estos *brokers* pueden permitir millones de conexiones a la vez, característica esencial para el mundo del *IoT*.
- Seguridad: según el funcionamiento de *MQTT*, este utiliza medidas para encriptar y autenticar mensajes, lo que evita filtraciones de datos.
- Integración. La conexión de las aplicaciones web con el *MQTT broker* es crucial para nuestro proyecto, ya que permitirá la comunicación entre nodos que, de otro modo, no serían compatibles.

A continuación se indican los *brokers* que se utilizarán para el proyecto.

4.4.1 Mosquitto

Mosquitto (también abreviado a veces como *MQTT*) es un *broker MQTT* “open source” que se utilizará para el proyecto. Esta tecnología es óptima por tener, entre otras cosas, un diseño ligero y su compatibilidad multiplataforma.

Al tener este diseño tan ligero conlleva ciertas desventajas tales como la limitación de la escalabilidad o la ausencia de componentes esenciales como una interfaz web, lo cual es fundamental para el proyecto al aportar la monitorización de los mensajes.

Como el proyecto hace uso de solo 3 placas, usaremos *mosquitto* con la finalidad de monitorizar las placas y comprobar su uso, funcionamiento y envío de mensajes en entornos de prueba.

4.4.2 EMQX

EMQX es uno de los *brokers* de código abierto más utilizados en la actualidad. Sus principales características incluyen su escalabilidad (tanto horizontal como vertical), alta disponibilidad, rendimiento y fiabilidad.

Debido a su popularidad, se cuenta con un soporte masivo que resulta muy beneficioso a nuevos usuarios. Además, *EMQX* nos aporta una interfaz web

integrada que permite gestionar las conexiones y monitorizar la red de manera simultánea, lo que lo convierte en una solución ideal para el proyecto.

4.5 Servicios para monitorizar

La finalidad es visualizar la información enviada por la aplicación de monitoreo de *LoRaMesher*. Para esto es necesario un tipo de servicio que se encargue de proporcionar las funcionalidades que nos interesa: mostrar la información de manera gráfica y la capacidad de verla en tiempo real.

4.5.1 *InfluxDB & Telegraf*

InfluxDB y *Telegraf* son usados de forma combinada para servicios de monitoreo. Se basa en integrar una base de datos de código abierto (*InfluxDB*) con un complemento de recolección de datos (*Telegraf*). Esto permite visualizar, recopilar y procesar datos en tiempo real. *Telegraf* actúa como un complemento para la fuente de información, que en este caso es *MQTT*. Una vez procesados los datos, *Telegraf* ofrece complementos de salida para enviarlos a *InfluxDB*, donde se visualizan.

4.5.2 *Grafana*

La herramienta encargada de la visualización de datos en el servicio de monitoreo de este proyecto se llama *Grafana*, una aplicación web de código abierto que permite mostrar datos en tiempo real. Aunque ofrece diversas fuentes de datos, en este proyecto solo se utilizará *InfluxDB*. Esta herramienta genera gráficos y muestra toda la información que recibe de *InfluxDB*, que obtiene los datos procesados por *Telegraf*. La combinación de *Telegraf*, *InfluxDB* y *Grafana*, conocida como la pila *TIG*, es muy popular ya que permite tanto el almacenamiento como la visualización de datos en tiempo real.

4.5.3 *React*

Por último, es necesario una aplicación para enviar las consultas a los nodos. Por lo que para este proyecto se utiliza *React*. *React* es una librería escrita en *JavaScript* para desarrollar aplicaciones web centradas en interfaces de usuario. Además, permite sitios web capaces de procesar datos en tiempo real, lo cual, es necesario para que la respuesta del nodo se muestre inmediatamente después de hacer la consulta.

5. Implementación del estado inicial

Este apartado describe paso a paso lo que se ha hecho para probar el funcionamiento del sistema en su totalidad. A continuación, se muestra una especie de guía que también servirá más adelante para el proyecto una vez que hayamos integrado las mejoras.

Primeramente se han de conectar mediante *USB* las placas *T-Beam* al pc para cargar el programa desde el *Visual Studio*. En mi caso se ha utilizado un *USB bridge* (dispositivo circular de color blanco en la Figura 5) para hacer la conexión, aunque no es necesario si se conecta cada cable en un puerto *USB* distinto del ordenador.

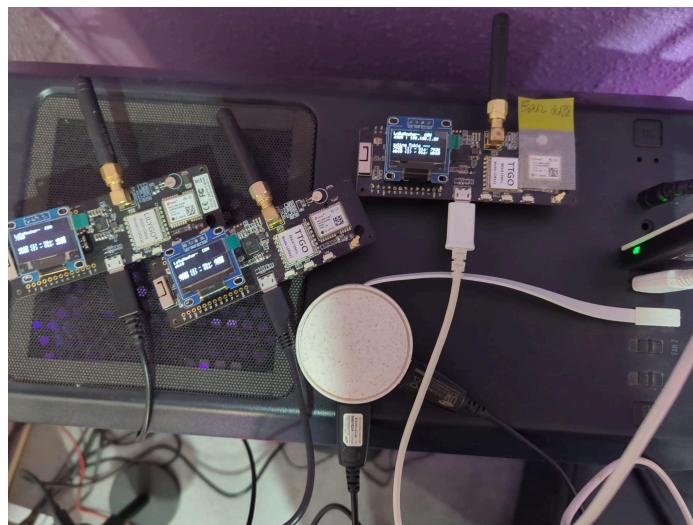


Figura 6: conexión de las placas mediante cables *USB*. Fuente: *Elaboración propia*.

El segundo paso es abrir el *Visual Studio* y comprobar que el pc detecta las placas. Para esto, se abre el terminal que nos ofrece el *IDE* y escribimos lo siguiente:

```
Unset  
pio device list
```

Este comando lista los dispositivos conectados. En nuestro caso, las placas son los dispositivos */dev/ttyACM0*, */dev/ttyACM1* y */dev/ttyUSB0*. Nótese que hay una de las placas que es diferente al detectarla como *USB* y no como *ACM*. Esto se debe a que una de las placas es de un modelo anterior.

```

gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoRaMe
● sher-main/LoRa/LoRaChat_2023_10_30$ pio device list
/dev/ttyS0
-----
Hardware ID: PNP0501
Description: ttyS0

/dev/ttyUSB0
-----
Hardware ID: USB VID:PID=10C4:EA60 SER=0200EA5E LOCATION=1-1
.3
Description: CP2104 USB to UART Bridge Controller - CP2104 U
SB to UART Bridge Controller

/dev/ttyACM1
-----
Hardware ID: USB VID:PID=1A86:55D4 SER=52D4003599 LOCATION=1
-9:1.0
Description: USB Single Serial

/dev/ttyACM0
-----
Hardware ID: USB VID:PID=1A86:55D4 SER=576F000191 LOCATION=1
-1.1:1.0
Description: USB Single Serial

```

Figura 7: Captura del terminal del Visual Studio al listar dispositivos. Fuente: Elaboración propia.

Una vez ya hemos comprobado que nuestro ordenador detecta las 3 placas, lo que debemos hacer es darle los permisos a los dispositivos para que se puedan leer/escribir/ejecutar. Para ello, en este mismo terminal introducimos:

```

Unset
sudo chmod 777 /dev/ttyUSB0
sudo chmod 777 /dev/ttyACM0
sudo chmod 777 /dev/ttyACM1

```

Este comando nos pide introducir la contraseña del pc. Una vez la pongamos, nos dará los permisos para las placas.

```

gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoRaMe
● sher-main/LoRa/LoRaChat_2023_10_30$ sudo chmod 777 /dev/ttyU
SB0
[sudo] contraseña para gelowwakkai:
gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoRaMe
● sher-main/LoRa/LoRaChat_2023_10_30$ sudo chmod 777 /dev/ttyA
CM0
gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoRaMe
● sher-main/LoRa/LoRaChat_2023_10_30$ sudo chmod 777 /dev/ttyA
CM1

```

Figura 8: Captura del terminal del Visual Studio al cambiar los permisos. Fuente: Elaboración propia.

En este punto ya se puede cargar el código. Para que este programa funcione, debemos modificarlo ligeramente. Primero debemos obtener la *IP* de nuestro ordenador. Abrimos el terminal de nuestro pc y escribimos:

```
Unset
ifconfig
```

Se indicarán unas cuantas interfaces e *IPs*. Buscamos la de nuestro pc (en este caso, 192.168.1.33). Volvemos al *Visual Studio* y cambiamos el valor actual de “*MQTT_SERVER*” por la *IP* obtenida. De esta manera podemos trabajar en local, haciendo de nuestro ordenador como si fuera el servidor *MQTT*. Además también hemos de cambiar el Wifi que vayamos a usar. Vamos a localizar el define de “*WIFI_SSID*” y el de “*WIFI_PASSWORD*” y los cambiamos por el nuestro.

```
wlxb0487a877030: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.33 netmask 255.255.255.0 broadcast 192.168.1.255
inet6 fe80::a42c:c712:5a48:9ee%1 prefixlen 64 scopeid 0x2<link>
    > simulator
    > time
    > utilities
    > wallet
    > wifi
    C BuildOptions.h
    C config.h
    G display.cpp
    G display.h
    I echo.zip
    G LoraMesher.cpp
    C LoraMesher.h
    G main.cpp
    G resetGPS.cpp
    > test
    > Testing
    .gitignore
    LICENSE
    platformio.ini
    > OUTLINE
    > TIMELINE
        94 #define WIFI_PASSWORD "E6W7aQ9PMxm48tpRy3b"
        95 // #define WIFI_SSID "invent"
        96 // #define WIFI_PASSWORD "fakepassword"
        97
        // MQTT configurations
        98 #define MQTT_ENABLED
        99 #define MQTT_SERVER "192.168.1.33" //wlxb0487a877030
        100 #define MQTT_PORT 1883
        101 #define MQTT_USERNAME "admin"
        102 #define MQTT_PASSWORD "public"
        103 #define MQTT_TOPIC_SUB "from-server/"
        104 #define MQTT_TOPIC_OUT "to-server/"
        105 #define MQTT_MAX_PACKET_SIZE 2048 // 025 // 128, 256 or 512
        106 #define MQTT_MAX_QUEUE_SIZE 10
        107 #define MQTT_STILL_CONNECTED_INTERVAL 90000 //30000 // In minutes
        108 #define MQTT_STILL_CONNECTED_INTERVAL 90000 //30000 // In minutes
        109
        110
        111 #if defined(BLUETOOTH_ENABLED)
        112 #undef WIFI_ENABLED
        113 #undef MQTT_ENABLED
    PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoRaMe
sher-main/LoRa/LoRaChat_2023_10_30$ pio run -e ttgo-t-beam -t upload --upload-port /dev/ttyACM0
```

Figura 9: Captura del código con los cambios realizados. Fuente: Elaboración propia.

Guardamos los cambios y cargamos el programa usando la terminal que ofrece el *Visual Studio*:

```
Unset
pio run -e ttgo-t-beam -t upload --upload-port /dev/ttyACM0
```

A continuación se reescribirá el código de la placa (en este caso, el *ACM0*), con el nuevo código. Este dispositivo es el que hará de *gateway* al indicarle una

combinación `WIFI_SSID+WIFI_PASSWORD` válidos. Para las otras placas, hemos de volver a cambiar el código. Para evitar errores, no podemos borrar los “*define*” de estos valores, por lo que cambiaremos estos por unos que no existan. En mi caso, he puesto “invent” como valor de `SSID` y “fakepassword” como valor de contraseña. Esta red es inexistente, por lo que, las placas que contengan este código no serán capaces de conectarse a Internet por ellas solas, sinó que, tendrán que conectarse con el *gateway*.

```
88 // WiFi credentials
89 #define WIFI_SSID "invent"
90 #define WIFI_PASSWORD "fakepassword"
91
92 // MQTT configurationsuod
93 #define MQTT_ENABLED
94
95 #define MQTT_SERVER "192.168.1.33" //wlxb0487a877030
96 #define MQTT_PORT 1883
97 #define MQTT_USERNAME "admin"
98 #define MQTT_PASSWORD "public"
99 #define MQTT_TOPIC_SUB "from-server/"
100 #define MQTT_TOPIC_OUT "to-server/"
101 #define MQTT_MAX_PACKET_SIZE 2048 // 025 // 128, 256 or 512
102 #define MQTT_MAX_QUEUE_SIZE 10
103 #define MQTT_STILL_CONNECTED_INTERVAL 90000 //300000 // In milis
104
105
106
107
108
109
110
111 #if defined(BLUETOOTH_ENABLED)
112 #undef WIFI_ENABLED
113 #undef MQTT_ENABLED
114 #endif
115
116 //Wallet Configuration // FF: from previous version
117 #ifndef A0
118 #define A0 A0
119 #endif
```

Figura 10: Captura con el código a cargar en la placa ACM1. Fuente: elaboración propia.

Volvemos a guardar los cambios y cargamos el código:

```
pio run -e ttgo-t-beam -t upload --upload-port /dev/ttyACM1  
pio run -e ttgo-t-beam -t upload --upload-port /dev/ttyUSB0
```

Una vez tenemos las 3 placas con el código correspondiente, y después de un breve período de tiempo, veremos algo como en la Figura 11 en las pantallas de las *T-Beam*.

```
88 // WiFi credentials
89 #define WIFI_SSID "invent"
90 #define WIFI_PASSWORD "fakepassword"
91
92 // MQTT configurations
93 #define MQTT_ENABLED
94 #define MQTT_SERVER "192.168.1.33" //wlxb0487a877030
95 #define MQTT_PORT 1883
96 #define MQTT_USERNAME "admin"
97 #define MQTT_PASSWORD "public"
98 #define MQTT_TOPIC_SUB "from-server/"
99 #define MQTT_TOPIC_OUT "to-server/"
100 #define MQTT_MAX_PACKET_SIZE 2048 // 025 // 128, 256 or 512
101 #define MQTT_MAX_QUEUE_SIZE 10
102 #define MQTT_STILL_CONNECTED_INTERVAL 90000 //30000 // In minutes
103
104 #if defined(BLUETOOTH_ENABLED)
105 #undef WIFI_ENABLED
106 #undef MQTT_ENABLED
107#endif
108
109
110 //Wallet Configuration // FF: from previous version
111 #ifndef A0
112 #define A0 A0
113#endif
114
115
116 //Wallet Configuration // FF: from previous version
117 #ifndef A0
118 #define A0 A0
119#endif
```

Figura 11: Captura con el código a cargar en la placa USB0. Fuente: elaboración propia.

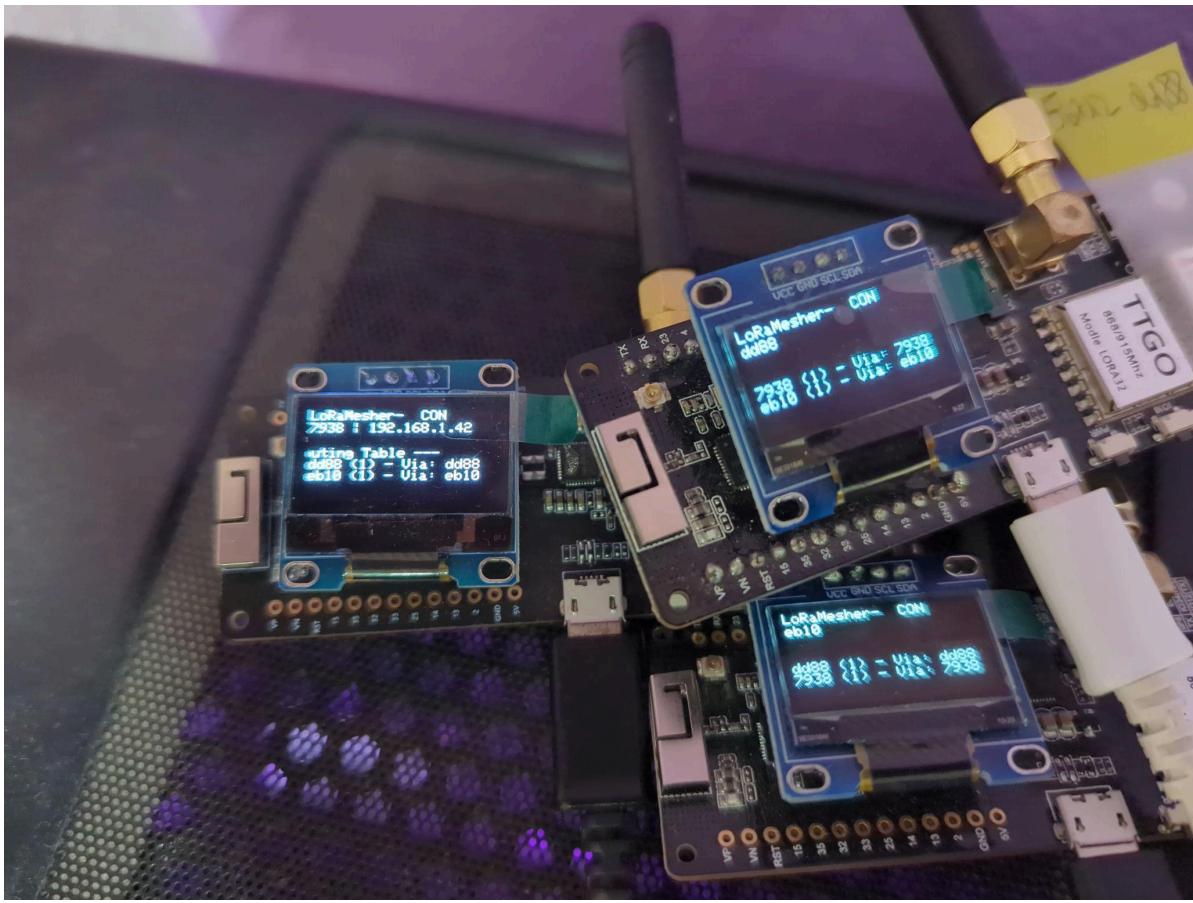


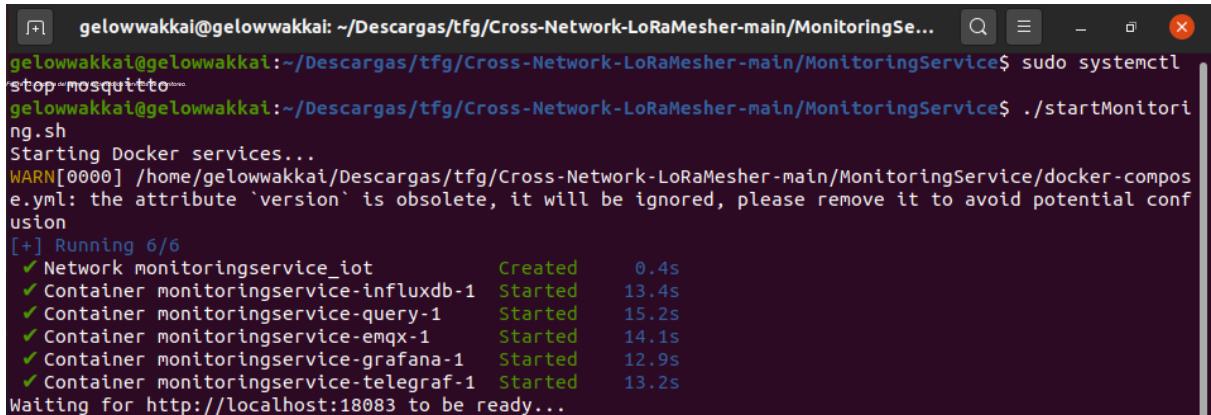
Figura 12: T-Beams con el código cargado y funcionando. Fuente: Elaboración propia.

Si nos fijamos en las pantallas de las placas, vemos que una es ligeramente diferente. En la figura 11, la placa que está más a la izquierda muestra una dirección IP: 192.168.1.42. Este dispositivo es el que hace de *gateway*, por lo que es el único que está conectado directamente a la red (el que hemos cargado el programa con el Wifi existente). Las otras 2 placas no tienen IP, pero vemos sus identificadores (7938, dd88 y eb10) . Si nos fijamos en las tablas de enruteamiento, vemos que las placas están conectadas entre sí.

Volvemos al terminal del ordenador. Ahora pondremos en marcha los servicios de monitoreo, navegamos por los directorios hasta entrar en “*MonitoringService*”. Dentro de la carpeta ejecutamos:

```
Unset
sudo systemctl stop mosquitto
./startMonitoring.sh
```

Con el primer comando estamos parando el funcionamiento de *mosquitto* porque tiene reservado de forma predeterminada el puerto 1883. Debido a que queremos hacer uso de ese puerto para desplegar el *EMQX*, primero hemos de dejarlo libre.



```
gelowwakkai@gelowwakkai: ~/Descargas/tfg/Cross-Network-LoRaMesher-main/MonitoringService$ sudo systemctl stop mosquitto
gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoRaMesher-main/MonitoringService$ ./startMonitoring.sh
Starting Docker services...
WARN[0000] /home/gelowwakkai/Desktop/tfg/Cross-Network-LoRaMesher-main/MonitoringService/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 6/6
 ✓ Network monitoringservice_iot          Created      0.4s
 ✓ Container monitoringservice-influxdb-1 Started    13.4s
 ✓ Container monitoringservice-query-1   Started    15.2s
 ✓ Container monitoringservice-emqx-1    Started    14.1s
 ✓ Container monitoringservice-grafana-1 Started   12.9s
 ✓ Container monitoringservice-telegraf-1 Started   13.2s
Waiting for http://localhost:18083 to be ready...
```

Figura 13: Captura del terminal iniciando los servicios de monitoreo. Fuente: Elaboración propia.

Esperamos unos minutos y automáticamente se abrirán los servicios en el navegador como se muestra en la siguiente imagen:

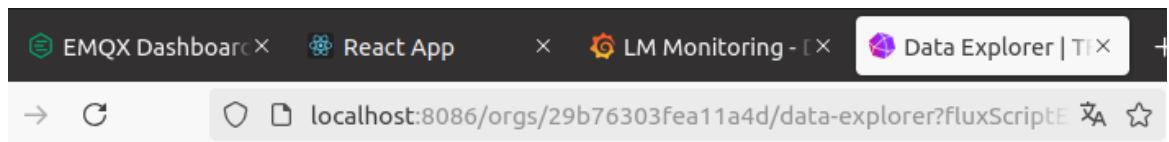
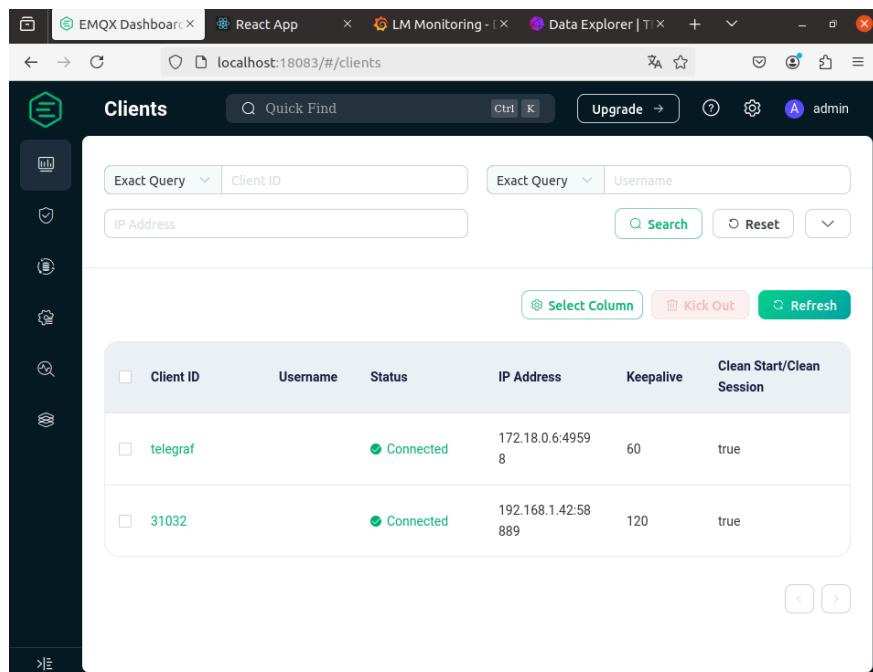


Figura 14: Captura de las pestañas abiertas al empezar a monitorear. Fuente: Elaboración propia.

Al clicar sobre la pestaña de *EMQX*, veremos los siguientes servicios conectados:



A screenshot of the 'Clients' section of the EMQX Dashboard. The URL is 'localhost:18083/#/clients'. The interface includes search filters for 'Client ID', 'Username', and 'IP Address', and buttons for 'Search', 'Reset', 'Select Column', 'Kick Out', and 'Refresh'. A table lists connected clients:

| Client ID | Username | Status | IP Address | Keepalive | Clean Start/Clean Session |
|-----------|----------|-----------|-------------------|-----------|---------------------------|
| telegraf | | Connected | 172.18.0.6:49598 | 60 | true |
| 31032 | | Connected | 192.168.1.42:5889 | 120 | true |

Figura 15: Captura del servicio EMQX inicialmente. Fuente: Elaboración propia.

El *broker* en este momento detecta los servicios de *Telegraf* y un “31032”. Si nos fijamos en la dirección *IP* de este último, vemos que es la misma *IP* que tenía la *T-Beam* que hace de *gateway*. Ahora falta conectar el servicio para las *query*. Sencillamente vamos a la app de *React* y clicamos en “connect” en el bloque de “Connection” con los valores predeterminados.

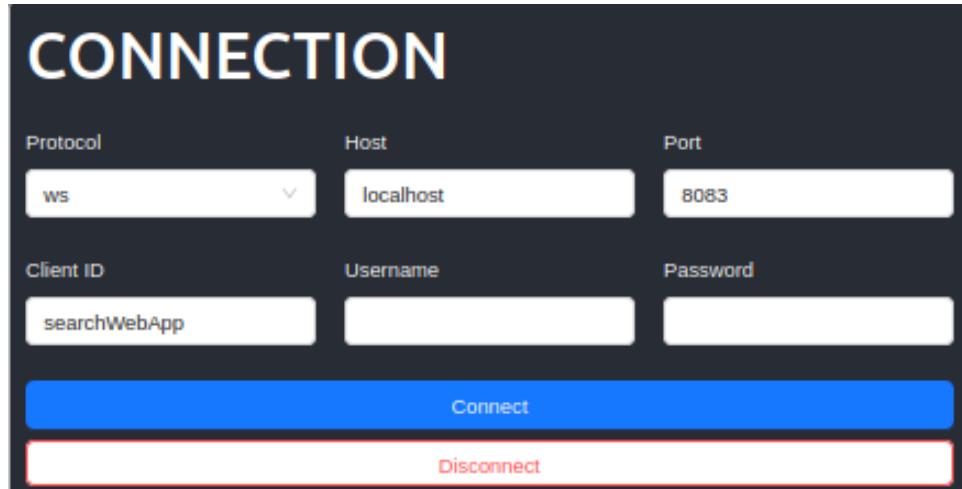


Figura 16: WebApp de *React* sin estar conectado. Fuente: Elaboración propia.

Y se actualizará a lo siguiente, ahora detectando las placas en la sección de “AVAILABLE NODES”.

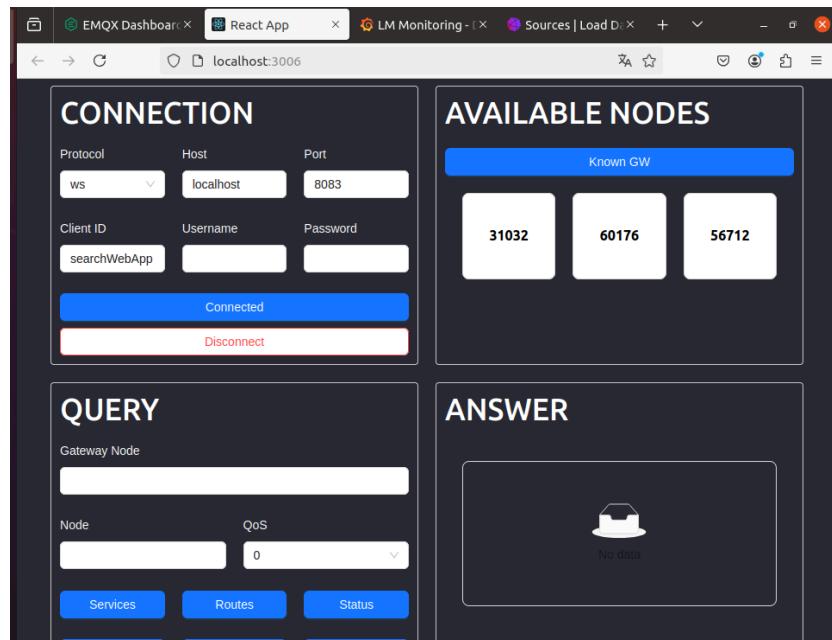


Figura 17: Captura de *React* cuando el servicio está conectado. Fuente: Elaboración propia.

Si volvemos a *EMQX*, ahora habrá un servicio nuevo: *searchWebApp*, el servicio que se encarga de los *query*.

The screenshot shows the EMQX Dashboard interface with the 'Clients' tab selected. The left sidebar has icons for clients, topics, subscriptions, and gateway. The main area has search filters for Client ID, Username, and IP Address, along with buttons for 'Search', 'Reset', 'Select Column', 'Kick Out', and 'Refresh'. A table lists three clients:

| <input type="checkbox"/> Client ID | Username | Status | IP Address | Keepalive | Clean Start/Clean Session |
|---------------------------------------|----------|-----------|-------------------|-----------|---------------------------|
| <input type="checkbox"/> telegraf | | Connected | 172.18.0.6:49598 | 60 | true |
| <input type="checkbox"/> searchWebApp | | Connected | 172.18.0.1:57128 | 60 | true |
| <input type="checkbox"/> 31032 | | Connected | 192.168.1.42:5889 | 120 | true |

Figura 18: Captura de EMQX con React activo. Fuente: Elaboración propia.

Para ver qué mensajes se mandan, vamos a ir al terminal y ejecutar lo siguiente:

```
Unset
mosquitto_sub -t to-server/#
```

Con esto hemos hecho que nuestro terminal se suscriba al *topic*, y por lo tanto, podemos observar los mensajes que se envían. Volvemos a la aplicación web de *React* para hacer algunos *query* y comprobar que funcionen correctamente. Para esto, vamos al campo “*QUERY*” de la aplicación de *React* e introducimos el nodo *gateway* en su campo correspondiente. En este caso, el 31032 es el *gateway*, ya que lo hemos podido ver en la aplicación *EMQX*. Seguidamente indicamos el nodo con el que nos queremos comunicar y clicamos en “la pregunta” que le queremos hacer.

A modo de ejemplo, he preguntado al nodo que hace de *gateway* (31032) los servicios activos. Este ha respondido con: “10 Services”, como se muestra en la siguiente captura.

The terminal window shows the command `mosquitto_sub -t to-server/#` being run, with the output of the MQTT subscription. The output includes JSON messages with fields like `messageId`, `addrSrc`, `addrDst`, `messageSize`, `queryCommand`, and `query`. The React application has three main sections: **NODES** (Known GW: 31032, 60176, 56712), **QUERY** (Gateway Node: 31032, Node: 31032, QoS: 0), and **ANSWER** (31032: 10 Services, 31032: 0 Status, 31032: 10 Services).

```

gelowwakkai@gelowwakkai:~/Descargas/tfg/Cross-Network-LoraMesher-main/MonitoringService$ mosquitto_sub -t to-server/#
{"data":{"messageId":2,"addrSrc":31032,"addrDst":0,"messageSize":37,"queryCommand":0,"query":"Services","queryAns":"10"}}
{"data":{"messageId":5,"addrSrc":60176,"addrDst":31032,"messageSize":85,"monitoringState":1,"nServices":10,"nRoutes":2,"ledStatus":0,"outMessages":6,"inMessages":0}}
  
```

Figura 19: Captura de React al hacer la query “Services”. Fuente: Elaboración propia.

En caso de querer preguntar a otro nodo, solo hay que cambiar el valor del campo “Node” por uno de los valores que nos indica el apartado “NODES”, es decir, cambiarlo por 60176 o 56712 en este caso. También es posible preguntar por otros valores como el estado del nodo o la tabla de enrutamiento (“Status” y “Routing Table”, respectivamente). Por último, hay una captura de la aplicación web de *Grafana* que muestra de forma gráfica, el tráfico que ha habido en los nodos.

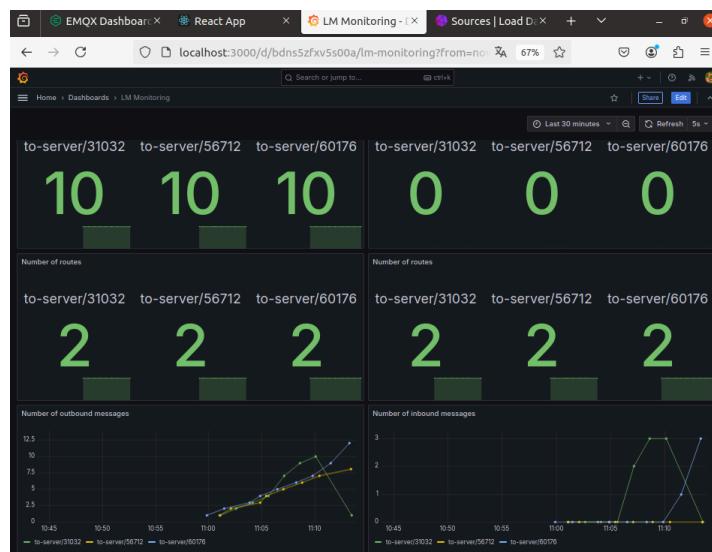


Figura 20: Captura de Grafana después de generar tráfico de red. Fuente: Elaboración propia.

En el tablero se muestran (de izquierda a derecha y de arriba a abajo) el número de servicios de cada placa, el estado, el número de rutas y 2 gráficos que muestran el número de mensajes salientes y entrantes, respectivamente.

Ya habiendo hecho todo esto, tenemos el proyecto funcionando tal y como lo tenía el autor original en su momento, por lo que ahora se procederá a implementar las mejoras a este.

6. Análisis de la base

Este proyecto tiene como base o punto de partida el TFG de Jack, como ya se ha comentado anteriormente. De manera sencilla, este capítulo está dedicado a: saber qué se tiene, hacia dónde va dirigido el trabajo y cómo se va a llegar hasta ese punto.

- Investigar lo que hay como inicio.
- Encontrar limitaciones y decidir cuál superar.
- Evaluar el desarrollo.

6.1 Investigar la base

Teniendo la base ya implementada (capítulo 5), se va a analizar, a grandes rasgos, qué es lo que se tiene:

Actualmente el proyecto desarrolla una solución de monitoreo para redes *LoRaMesher*, que consiste en una infraestructura combinada de aplicaciones y servicios web para gestionar y observar el comportamiento de los nodos dentro de una red. Esta solución se centra en permitir la comunicación entre nodos *LoRaMesher* y usuarios externos a través de un broker *MQTT*.

En cuanto a su utilidad: el proyecto tiene como finalidad monitorear y gestionar los nodos en redes *LoRaMesher*. En este punto, es posible obtener datos de los nodos y observar el estado de la red de forma dinámica sin necesidad de estar conectados directamente.

Por último, comentar las deficiencias que encontramos en el sistema actual. Podemos resumir sus limitaciones en 2:

- Pérdida de datos: en caso de que los servicios se detengan, los datos se perderán sin posibilidad de recuperación.

- Poca velocidad: debido a como está montado ahora, las consultas a los nodos *gateway* (los que no están conectados a Internet directamente) pueden llegar a ser bastante lentas al hacer los saltos de nodos.

6.2 Objetivo del trabajo

Como ya se ha dicho en la justificación, hay 4 propuestas interesantes para este trabajo (ver apartado 2.2 Soluciones posibles). En esta sección se expondrán las 4 ideas principales y se evalúa y analiza cada una de ellas, con el fin de ver la factibilidad de estas. Se explica la opción, qué significa llevarla a cabo, y los beneficios y desafíos (o problemas) al implementarla.

6.2.1 Propuesta 1: *Deployment*

La primera opción es desplegar el entorno local a Internet. De momento la base utiliza un entorno local para la red usando los servicios asignados a nuestra propia máquina a través del *localhost*. Esto quiere decir que, si queremos acceder a los nodos, debemos conectarnos desde un dispositivo que esté conectado a la misma red que el *gateway*.

Hacer el *deployment* del proyecto significa trasladar la infraestructura actual a un entorno en la nube o servidor accesible por Internet. Esto permitiría acceder a los nodos y controlarlos de forma remota, lo que conlleva a una mejora en disponibilidad y facilidad de mantenimiento. Al tener un servidor dedicado, también sería posible hacer una base de datos con el fin de solventar el problema mencionado anteriormente de pérdida de datos.

6.2.1.1 Beneficios

- Accesibilidad remota: El principal beneficio de la propuesta es la posibilidad de acceder al sistema de monitorización y a los datos de los nodos desde cualquier lugar con conexión a Internet, sin necesidad de estar presente en la red local.
- Escalabilidad: En caso de que la red *LoRaMesher* crezca y haya más nodos que monitorizar, será bastante sencillo añadir más recursos fácilmente.
- Integración: Una vez se despliegue en Internet, se podrán integrar otros servicios web externos que pueden mejorar la funcionalidad del proyecto.

6.2.1.2 Desafíos

- Seguridad: Un posible problema al exponer nuestra red a Internet es la seguridad. Es muy común en el mundo de los dispositivos *IoT* los ataques informáticos, por lo que se tendrá que configurar los *brokers* para evitar accesos no autorizados o ataques de denegación de servicio.
- Latencia: Uno de los problemas que tiene el sistema actual es su poca velocidad. Al hacer *deployment* a Internet, no solo no solucionamos este problema, sino que se agravaría, al tener que navegar por la red. Además, si el servidor que aloja el *broker* cae o tiene problemas, se perdería la conexión con los nodos.
- Configuración compleja: Llevar a cabo la configuración del entorno para que sea seguro, confiable, escalable y resistente a las caídas resulta un trabajo con la combinación de tiempo y dificultad demasiado elevada.

6.2.2 Propuesta 2: Sensores

La siguiente idea a comentar se basa en la implementación de sensores en el proyecto. Llevar a cabo esta acción significa integrar estos dispositivos con el fin de recopilar datos del entorno en tiempo real, como temperatura o humedad, y transmitirlos a través de la red. Al conseguir realizar esta mejora, se expanden las capacidades del proyecto, siendo más funcional y útil para aplicaciones *IoT*.

6.2.2.1 Beneficios

- Expansión de funcionalidad: Al hacer la incorporación de sensores se amplía el alcance del proyecto al recolectar estos datos que pueden ayudar en aplicaciones *IoT* como agricultura, ciudades inteligentes o monitoreo ambiental.
- Valor de los datos: Como ya se ha dicho, y se puede suponer, los datos que ahora se manejan son más valiosos en comparación con lo que había en un inicio. La información servirá no solo para monitorear en tiempo real, sino también para analizar, mejorar procesos o tomar decisiones.
- Escalabilidad: Al igual que en el caso anterior, podemos agregar más sensores y nodos al tener la red con capacidad de ser escalable.
- Integración y uso con otras herramientas: Los datos que obtengamos de los sensores se pueden usar para alimentar modelos de IA, optimizar sistemas o generar alertas automatizadas.

6.2.2.2 Desafíos

- Compatibilidad del *hardware*: Algunos sensores pueden necesitar adaptadores o configuraciones específicas para funcionar con las placas que ya se tienen.
- Gestión de energía: Si se hace uso de estos sensores adicionales, aumentaremos el consumo energético de los nodos, cosa que afectará al impacto ambiental del proyecto.
- Capacidad de la red: Al funcionar el sistema con datos más significativos y en caso de agregar muchos sensores, aumentaría el tráfico de la red. Esto podría requerir una optimización de la configuración de los nodos con el fin de evitar congestiones.
- Calibración y precisión: Los sensores pueden necesitar calibración para garantizar la precisión de los datos recolectados.

6.2.3 Propuesta 3: Actualización

La tercera de las propuestas, es quizá, la más necesaria para el sistema en sí, aunque también, la menos útil para “el mundo”. Actualizar los componentes al *LoRaMesher* actual es un paso esencial para que el proyecto pueda aprovechar las mejoras y características que se han implementado en las últimas versiones de la librería.

Hacer este cambio en el trabajo implica adaptar el código, las aplicaciones y la infraestructura existente a la versión nueva optimizando, así, compatibilidad y rendimiento.

6.2.3.1 Beneficios

- Compatibilidad, estabilidad y fiabilidad: Hacer la implementación de esta idea nos garantiza que el proyecto funcione con un código más reciente y depurado, lo que evita problemas de compatibilidad, y fallos en las versiones anteriores, aumentando también, la fiabilidad del sistema.
- Rendimiento: Las mejoras al código de nuestra librería incluyen optimizaciones que pueden mejorar el rendimiento de la red y reducir el consumo energético.
- Nuevas funcionalidades: Al actualizar el sistema, se podrán aprovechar nuevas funcionalidades de la biblioteca que pueden llegar a simplificar la implementación.

6.2.3.2 Desafíos

- Tiempo de desarrollo: la posible incompatibilidad del código existente puede implicar que las aplicaciones actuales necesiten modificaciones significativas, lo que conlleva mucho tiempo.
- Aprendizaje y estudio: A diferencia de las otras ideas que se basan en “mejorar el sistema”, esta se dedica a cambiarlo. Debido a esto, se ha de estar muy familiarizado con todo el código actual, además del código nuevo del *LoRaMesher* con el fin de llevar a cabo la actualización.
- Pruebas de funcionamiento: Aunque esta propuesta no sea la única con pruebas de funcionamiento, si es la que es crucial hacerlas muy exhaustivas. Al hacer cambios en la aplicación con el fin de actualizarla, podría llevarnos a un caso donde nuestro proyecto sea parcial o totalmente inservible.

6.2.4 Propuesta 4: *ESP32*

Por último, tenemos la propuesta de validar con otras placas. El objetivo sería evaluar el funcionamiento del sistema actual con las placas *T-Beam*, y ver si también se puede desplegar y funcionar correctamente con otros tipos de nodos, en este caso con *ESP32*. Para esto se usarían las placas a disposición del tutor de este TFG y se intentarían integrar en el sistema sustituyendo, o, complementando las *T-Beam* ya programadas.

6.2.4.1 Beneficios

- Coste: Las placas *ESP32* tienen un coste de unos 10€ [21], por lo general, son más baratas que las *T-Beam*, cuyo precio ronda los 50€. Para este proyecto subiría el precio del *hardware* a usar, pero, en un caso real comercializable, bajaría significativamente su coste.
- Flexibilidad: Las *ESP32* son unos dispositivos muy utilizados debido a su versatilidad, flexibilidad y recursos de procesamiento.
- Experiencia: Otro de los beneficios es la experiencia que tengo yo, el autor, con estas placas. Anteriormente ya elaboré un proyecto usando este modelo, por lo que estoy familiarizado con el entorno.

- Escalabilidad: En el caso de que la red sea capaz de permitir distintos tipos de nodos, se habrá convertido entonces en una muy escalable, al adaptarse, también a distintos tipos de consumos de energía y funcionalidades.
- Redundancia y robustez: Sabiendo que, en algún momento, el sistema tiene que fallar, y poniéndonos en el caso de que un tipo de nodos falle o presente problemas, el otro tipo podría continuar su funcionamiento.

6.2.4.2 Desafíos

- Compatibilidad y tiempo de desarrollo: Al igual que en el caso anterior, las aplicaciones y configuraciones existentes para los *T-Beam* pueden necesitar cambios o ajustes significativos para su funcionamiento con los *ESP32*, lo que requeriría de mucho tiempo.
- Eficiencia: Aunque los *ESP32* tienen ciertas ventajas sobre los *T-Beam*, lo cierto es que, al ser tan generales, no son tan eficientes. La implementación de estos dispositivos podría afectar a la cobertura, alcance y velocidad de la red.
- Energía: Los *T-Beam* suelen consumir menos energía que los *ESP32*, lo que conlleva a un coste energético mayor y un impacto ambiental negativo. Además, si se tiene planeado implementar este sistema para que opere automáticamente y sin intervención humana, se necesitará un control de funcionamiento mayor.
- Integración: No es seguro que todas las partes de nuestro sistema (servicios de monitoreo web y aplicaciones) puedan soportar los datos de todo tipo de nodos sin perder funcionalidad o precisión.

6.3 Evaluación del desarrollo

Después de tener la base implementada y de poner en común los puntos de vista de Felix y el mio (el tutor y el autor de este proyecto, respectivamente), propusimos 2 objetivos.

El objetivo principal del proyecto es hacer la implementación de sensores en las placas. Como ya se ha descrito en el apartado anterior (6.2), tiene sus beneficios y desafíos, pero nos decantamos por esta propuesta por tener el tiempo muy limitado. En caso de que se complete esta propuesta con bastante margen de tiempo, se intentará hacer también una segunda mejora: hacer el *deployment* a Internet.

Con el fin de la realización de estos pasos, se dispone de materiales extra, proporcionados por el tutor. Este *hardware* nuevo no está contemplado en el

apartado 3.1.2 Presupuesto (recursos *hardware*), para mantener la coherencia con el informe presentado en GEP.

Aún así, a continuación, expongo un poco estos nuevos materiales que se usarán y sus respectivos costes monetarios.

Se ha dispuesto de 3 unidades adicionales de placas *T-Beam*, 1 *kit* de sensores y sus cables para hacer la conexión con las placas. Todo esto tiene un coste de adquisición aproximado de 180€ extra.

7. Integración de las mejoras

7.1 Cambios en el JSON & React

Al observar detenidamente el código de Jack, me he percatado de una ineficiencia o redundancia, en la *App* de *React*, los botones de “*Routes*” y “*Routing Table*” llaman a la misma función, por lo que independientemente cual de los 2 pulses, se generará la misma respuesta. Por eso se ha decidido sustituir uno de estos botones por un “*Sensor Value*”, que proporcione el valor que recolecta la placa del sensor conectado a ella.

Con el fin de llevar a cabo esta mejora, se ha de cambiar el contenido de varios ficheros dentro de las carpetas “*monitoring*” y “*query*”, que se encuentran dentro del path “*Cross-Network-LoRaMesher-main\LoRa\LoRaChat_2023_10_30\src*”:

- *monitoring.h / query.h*: Se modifican para incluir la cabecera de la función anterior.
- *monitoringMessage.h*: Se incluye una nueva variable con la información del sensor para generar un archivo *.json* que la contenga.
- *monitoring.cpp*: Se ha de crear una nueva función que lea y devuelva el valor del sensor.
- *query.cpp*: Se cambia el condicional de “*processReceivedMessage*”, para que, se pueda diferenciar una *query* dirigida al “*sensorValue*”.

A continuación, se muestra un fragmento de código del archivo *monitoringMessage.h* con la variable “*sensorValue*” añadida y utilizada para crear el *.json* en la función “*serialize*”.

C/C++

```
class MonitoringMessage: public DataMessageGeneric{
public:
    MonitoringState monitoringState;
    int monitoringSendTimeInterval;
    uint16_t nAddress;
    uint16_t nServices;
    uint16_t nRoutes;
    String routeTable;
    uint16_t ledStatus;
    uint16_t outMessages;
    uint16_t inMessages;
    float sensorValue;
    GPSMessage gps;

    uint32_t helloPacket[5] = {0, 1, 2, 3, 4};

    void serialize(JsonObject& doc) {
        //ESP_LOGV(MONITORING_TAG, "in class MonitoringMessage: void
        serialize");
        int i;
        ((DataMessageGeneric*)(this))->serialize(doc);
        doc[ "monitoringState" ] = 1; // FF: for short monitoring message
        doc[ "nServices" ] = nServices;
        doc[ "nRoutes" ] = nRoutes;
        doc[ "ledStatus" ] = ledStatus;
        doc[ "outMessages" ] = outMessages;
        doc[ "inMessages" ] = inMessages;
        doc[ "sensorValue" ] = sensorValue;
    }
}
```

Es importante que la modificación del *json* para enviar el valor del sensor, dado que es este mismo mensaje lo que está enviando nuestra placa *T-Beam*. Además, este archivo, llega hasta los servicios de monitorización, que necesitan del valor para poder representarlo en tiempo real y realizar consultas.

En cuanto a la información del sensor, esta se obtiene mediante la función que se implementa en los archivos “*monitoring.cpp*” que se muestra:

C/C++

```
/*NO SENSOR*/
/*float Monitoring::getSensorValue(){
    return 0;
}*/
```

```

/*DIGITAL TEMPERATURE*/
float Monitoring::getSensorValue(){
    OneWire oneWire(4); //El número 4 indica cual es el pin conectado a data
    DallasTemperature sensors(&oneWire);
    sensors.requestTemperatures();
    float temp = sensors.getTempCByIndex(0);
    return temp;
}

/*Button*/
/*float Monitoring::getSensorValue(){
    int buttonState = digitalRead(4); // Lee el estado del botón
    buttonState = !buttonState;
    float buttonStateFloat = (float)buttonState;
    return buttonStateFloat;
}*/

```

En la función que obtiene el valor de la temperatura se ha tenido que importar las librerías de “*OneWire*” y de “*DallasTemperature*”. Esto se hace desde el *Visual Studio*, en la pestaña “librerías” y buscando e instalando las que tienen estos nombres.

Hay varias funciones diferentes que se comentan o descomentan dependiendo del sensor que tenga conectado a la placa. En este caso, la función descomentada es la que se carga en una placa con un sensor de temperatura digital conectado.

Se ha de tener en cuenta que la cabecera de la función “*getSensorValue()*” ha de estar declarada como función de tipo *public* para que sea accesible desde “*query.cpp*”.

En el archivo “*query.cpp*”, con la finalidad de obtener también los mismos datos que en el archivo anterior, se crea una instancia de *Monitoring* y se llama a la función “*getSensorValue()*” de esa clase, es decir, la anterior.

C/C++

```

float Query::getSensorValue(){
    Monitoring& monitoring = Monitoring::getInstance();
    return monitoring.getSensorValue();
}
void Query::processReceivedMessage(messagePort port, DataMessage* message) {
    inQuery++;
    QueryMessage* queryMessage = (QueryMessage*) message;

```

```

        Log.infoln(F("FF: Query::processReceivedMessage  perform local actions"))
    );

    queryCommandS = queryMessage->queryCommand; //FF added
    queryvalueS = queryMessage->queryValue; //FF added

    Log.verboseln(F("FF in Query::processReceivedMessage queryCommands
%d"),queryCommandS );
    Log.verboseln(F("FF in Query::processReceivedMessage queryvalueS
%d"),queryvalueS );

    ESP_LOGV(QUERY_TAG, "services 2");

    switch (queryMessage->queryValue) {
        case QueryCommand::services:
            queryPetitionS = "Services";
            queryAnswerS = getServices();
            break;
        case QueryCommand::routes:
            queryPetitionS = "Routes";
            queryAnswerS = getRoutingTable();
            break;
        case QueryCommand::status:
            queryPetitionS = "Status";
            queryAnswerS = getLEDstatus();
            break;
        case QueryCommand::outMessages:
            queryPetitionS = "Outgoing messages";
            queryAnswerS = getOutMessages();
            break;
        case QueryCommand::inMessages:
            queryPetitionS = "Incoming messages";
            queryAnswerS = getInMessages();
            break;
        case QueryCommand::sValue:
            queryPetitionS = "Sensor Value";
            queryAnswerS = getSensorValue();
            break;
        default:
            break;
    }
    Log.infoln(F("FF: Query::processReceivedMessage  createAndSendQuery()"))
);
    createAndSendQuery();
}

```

También es necesario hacer un cambio en el funcionamiento de las *query*. En el archivo “*query.cpp*” se gestionan las peticiones de entrada, por lo que, en nuestro caso, vamos a tener en la función “*processReceivedMessage*” un nuevo caso, cuando se haga una pregunta dirigida al valor del sensor.

Por último hay que hacer una pequeña modificación en “*queryMessage.h*”, que es donde se inicia la variable a evaluar en el condicional. Una vez hecho esto y guardados los cambios, al utilizar la aplicación web de *React*, y haciendo clic en el botón, obtendremos el valor que devuelva la función “*getSensorValue()*”, o lo que es lo mismo, lo que nos proporcione el sensor.

7.2 Cambios en *Grafana*

Por otra parte, en *Grafana*, la aplicación que muestra la información en tiempo real, tiene 2 secciones que muestran el número de rutas, a lo que, se ha sustituido también, una de estas secciones, por otro gráfico en tiempo real que muestre el valor que marca el sensor. Esta es una tarea bastante sencilla y a la vez, con un resultado visualmente muy satisfactorio.

Primero he abierto el archivo ya existente que se encuentra en el *path*: *Cross-Network-LoRaMesher-main\Cross-Network-LoRaMesher-main\MonitoringService\grafana\dashboard\GrafanDashboard.json*

Este archivo contiene todo lo relacionado a la generación del *dashboard* de *Grafana*. Para hacer la modificación, he reutilizado el código ya existente, copiando el código de una de las gráficas de los mensajes (*inMessages*) y cambiando los valores de “*title*” y “*r.field*”. Dado que el código de generar el gráfico es muy extenso, he utilizado “...” para indicar que entre líneas, hay más código.

```
C/C++
{
  "datasource": {
    "type": "influxdb",
    "uid": "loramesher"
  },
  "fieldConfig": {
    "defaults": {
      "color": {
        "mode": "palette-classic"
      }
    }
  }
  ...
  "query": "from(bucket: \"loramesher\")\n    |> range(start: v.timeRangeStart, stop:v.timeRangeStop)\n    |> filter(fn: (r) =>\n      r.value > 0\n    )\n    |> count()\n    |> yield(name: \"r\")"
}
```

```

r._measurement == \"mqtt_consumer\" and \n      r._field ==
\"data_sensorValue\"\n    ),\n    \"refId\": \"A\"\n  },\n],\n  \"title\": \"Sensor value\",  
...\n  \"type\": \"timeseries\"\n}

```

La línea de *query* se lee del *json* que habíamos creado antes, por lo que, el *r._field* está leyendo el valor “*sensorValue*” que hemos incluido anteriormente en el *json*.

A continuación se muestra una nueva captura de “*Grafana*” con el nuevo gráfico en funcionamiento.

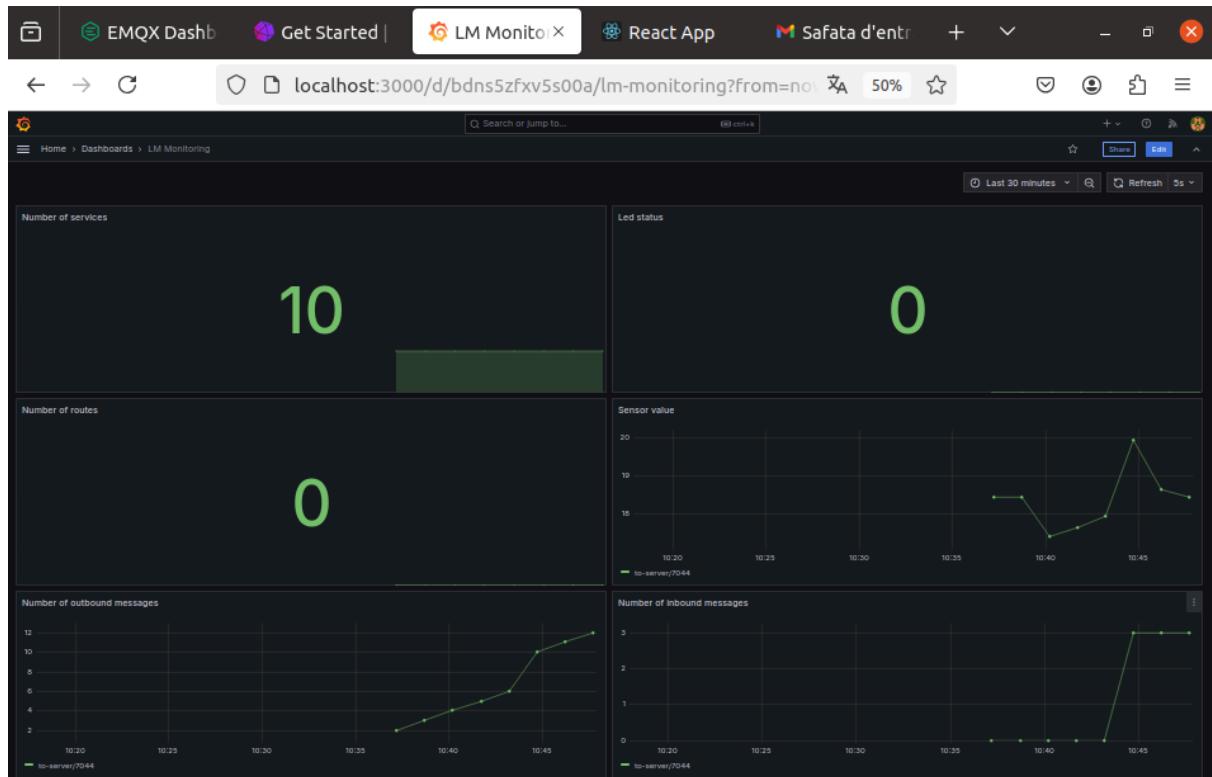


Figura 21: Captura de *Grafana* con el gráfico que muestra el valor del sensor. Fuente: Elaboración propia.

7.3 Conexiones en las placas

Aunque se tenga el código adaptado para leer valores de los sensores y poder mostrarlos, hay que tener en cuenta lo indispensable para esto, la conexión del sensor.

A continuación se muestra una imagen sobre cómo está hecha la conexión en las placas en este momento. El cable de datos lo he puesto en amarillo por motivos visuales, aunque en la realidad sea el blanco.

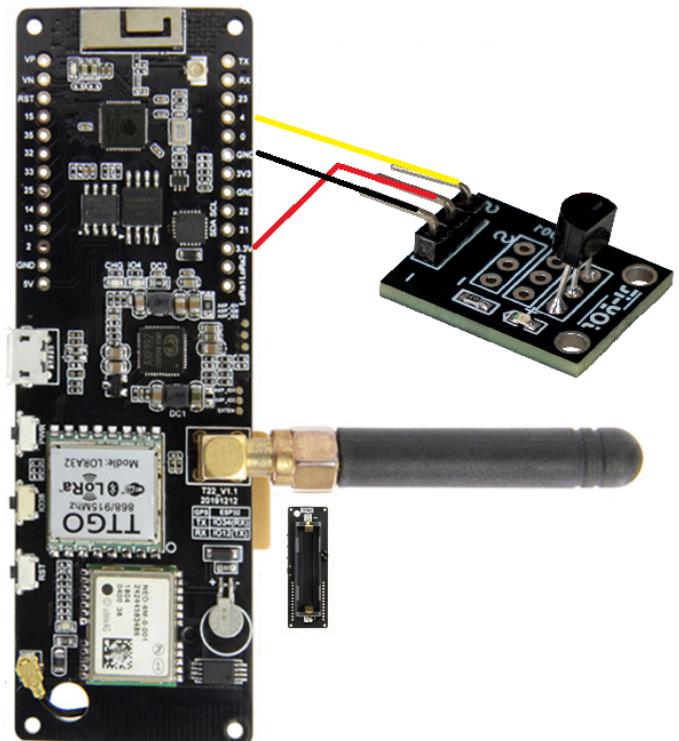


Figura 22: Conexión de los sensores a las T-Beam. Fuente: Elaboración propia.

Para hacer la conexión, se han usado los pines 4, 3.3V y GND, que son pines de datos, corriente y tierra, respectivamente. De manera arbitraria, se ha elegido el pin 4 entre todos los pines digitales disponibles en la placa, para hacer la conexión. En la siguiente figura, se indican los pines de las placas y su función.

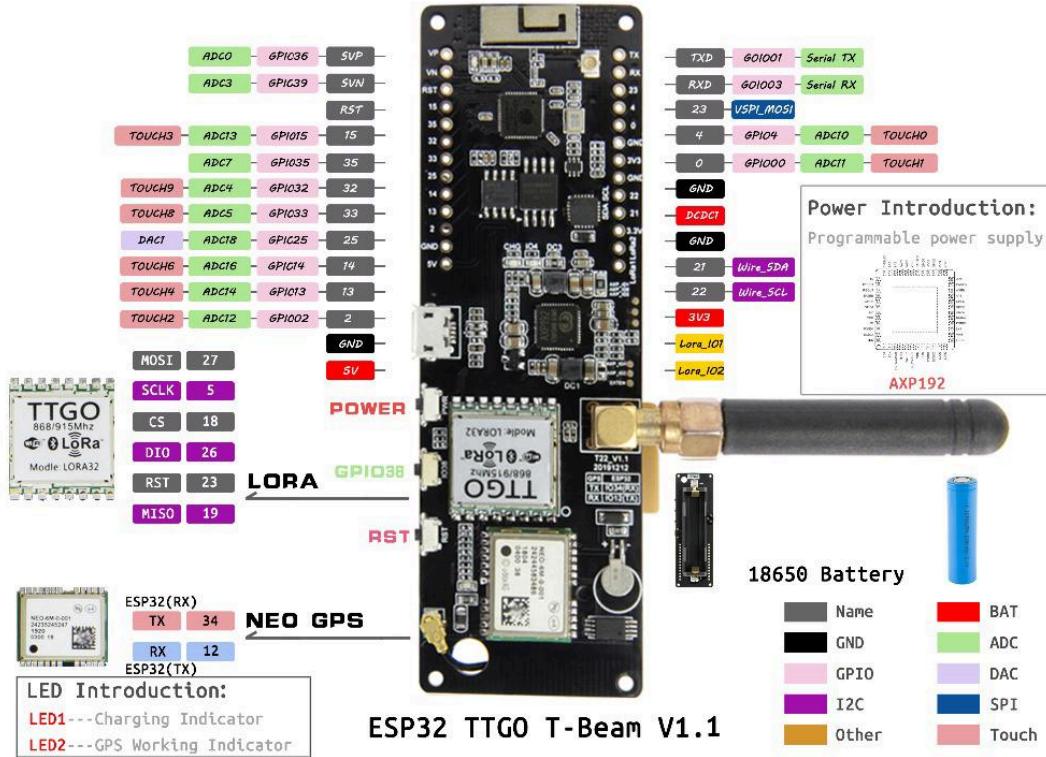


Figura 23: Pines de las placas T-Beam. Fuente: página web de thethingsnetwork [22].

En el caso de conectar un sensor analógico, la conexión del pin de datos, necesariamente, se ha de conectar en uno de los pines analógicos, mientras que un sensor digital, se conecta a cualquiera de los pines digitales.

En el caso del sensor de temperatura que disponemos, al ser un sensor digital, está conectado también a un pin digital.

Se han evaluado otros tipos de sensores tanto analógicos como digitales para hacer la implementación de estos. Se me proporcionó una caja con un total de 38 sensores diferentes, pero se descartaron algunos por varios motivos. El primero y más común de estos, fue por la información que podían aportar.

Muchos de los sensores son *LEDs*, cosa que no interesa ya que su única utilidad es aportar luz. Estos sensores pueden llegar a ser un poco más aptos en caso de que se quiera utilizar un sistema de alarma o similar, pero se debería usar una extensión de los cables y unos *LEDs* bastante más potentes.

Otro tipo de sensores son los de humedad, cosa que tampoco interesan en gran medida a no ser que se tenga algo que necesite de una humedad específica (por ejemplo, poner el sensor en la tierra de un huerto).

También se dispone de sensores de gravedad y presión, que no aportan demasiado valor para este trabajo, y, con los cuales, no se puede comprobar el funcionamiento de estos (al no poder cambiar la presión o gravedad).

Por último, estaban los sensores de sonido. Aquí entran tanto *buzzers* (dispositivos que emiten sonido) como detectores que captan el sonido. Se ha descartado el uso debido a diversos factores, tales como la utilidad que aportan o la dificultad de la implementación.

7.4 Deployment

La siguiente mejora implementada es sobre hacer el *deployment* del trabajo a Internet. Para esto se usa una máquina virtual proporcionada por el director. Para comenzar, tuve que generar una clave pública para que se me diera acceso a la máquina virtual por *ssh*. Se ha seguido el tutorial de [esta web](#) para la generación.

A continuación, desde nuestro terminal se introduce

```
Unset  
ssh juancarlos@cloudy4.pc.ac.upc.edu
```

para conectarnos a la *VM*. Una vez dentro de este pc, se hizo la instalación de las herramientas necesarias, en nuestro caso, el *net-tools*, para mirar la *IP* del pc, *mosquitto-clients* para el *MQTT* y el *docker-compose* para poder generar el programa. Además, también he clonado mi repositorio de [github](#), donde se encuentra el código actualizado del proyecto.

```
Unset  
sudo apt install net-tools  
  
sudo apt install mosquitto-clients  
  
sudo curl -L  
"https://github.com/docker/compose/releases/latest/download/docker-compose-$(  
(uname -s)-$(uname -m))" -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose  
  
git clone https://github.com/Gelowwakkai/Cross-Network-LoRaMesher.git
```

Para ejecutar poner en marcha el proyecto, igual que en la implementación de la base, se usa:

```
Unset  
docker-compose up -d
```

Una vez hecho esto, cambiamos en el archivo *config.h* la dirección *IP* del *MQTT server* a la *IP* de la máquina virtual en las placas y cargamos el nuevo programa. Para comprobar que el funcionamiento, igual que anteriormente, se usa:

```
Unset  
mosquitto_sub -t to-server/#
```

Y se pueden ver los mensajes que llegan en algo similar como la captura siguiente:

```
juancarlos@cloudy4:~/Cross-Network-LoRaMesher/MonitoringService$ mosquitto_sub  
-t to-server/#  
{"data": {"messageId": 1, "addrSrc": 7044, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 0, "ledStatus": 0, "outMessages": 2, "inMessages": 0, "sensorValue": 22.1875}}  
{"data": {"messageId": 2, "addrSrc": 7044, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 0, "ledStatus": 0, "outMessages": 3, "inMessages": 0, "sensorValue": 22.125}}  
{"data": {"messageId": 3, "addrSrc": 7044, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 0, "ledStatus": 0, "outMessages": 4, "inMessages": 0, "sensorValue": 22.125}}  
{"data": {"messageId": 4, "addrSrc": 7044, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 0, "ledStatus": 0, "outMessages": 5, "inMessages": 0, "sensorValue": 22}}  
{"data": {"messageId": 5, "addrSrc": 7044, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 0, "ledStatus": 0, "outMessages": 6, "inMessages": 0, "sensorValue": 22.5}}
```

Figura 24: Captura de los mensajes MQTT que llegan a la MV. Fuente: Elaboración propia.

A continuación, dado que se podrían provocar problemas de seguridad, se ha limitado el acceso público a los puertos del *MQTT broker*. Para esto se ha utilizado el programa *ufw*.

```
Unset  
sudo apt update  
sudo apt install ufw  
sudo ufw enable
```

Por último se han definido las reglas de acceso como cualquier *access list*, permitiendo las *IP* de la UPC y la *IP* pública de mi casa con el fin de comprobar el funcionamiento.

```
Unset
sudo ufw allow from 147.83.42.0/24 to any port 1883
sudo ufw allow from 147.83.30.0/24 to any port 1883
sudo ufw allow from 79.157.12.65 to any port 1883
sudo ufw deny 1883
```

8. Resultados

En esta sección se discute todo tipo de resultados obtenidos del proyecto explicando qué objetivos se han conseguido y cuáles no.

Inicialmente se pensaron 4 propuestas a implementar. De estas, se descartaron 2, la de actualizar el proyecto a una versión actual y la de cambiar el tipo de placas a *ESP32*. Después de elegir la propuesta en la que centraría mis esfuerzos, estas 2 quedaron descartadas.

En cambio, sí se puede mostrar lo conseguido en las otras 2 propuestas: implementación de sensores y *deployment* a Internet.

En cuanto a la implementación de sensores, se ha conseguido satisfactoriamente hacer la conexión de los sensores de temperatura y de presión (botón/pulsador). Estos sensores digitales se han conseguido conectar correctamente a la placa, cargar un programa que lea sus valores, enviarlos a través de los mensajes y mostrarlos en las aplicaciones web. Gracias a los cambios provocados, desde la aplicación web, también se puede comunicar con las placas para recolectar esta información. Seguidamente se muestran capturas con el funcionamiento del sistema. En este caso, con 5 placas conectadas, 2 de ellas a Internet, y de las 3 restantes, 1 tiene conectada el sensor de temperatura y otra el pulsador.

```

{"data": {"messageId": 6, "addrSrc": 7044, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 9, "inMessages": 2, "sensorValue": 18}}
{"data": {"messageId": 1, "addrSrc": 56712, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 2, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 27, "addrSrc": 11028, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 28, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 0, "addrSrc": 4920, "addrDst": 56712, "messageSize": 37, "queryCommand": 0, "query": "Sensor Value", "queryAns": "0.00"}}
{"data": {"messageId": 1, "addrSrc": 31032, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 2, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 2, "addrSrc": 56712, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 3, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 28, "addrSrc": 11028, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 29, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 3, "addrSrc": 56712, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 14, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 29, "addrSrc": 11028, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 30, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 2, "addrSrc": 31032, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 3, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 2, "addrSrc": 4920, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 5, "inMessages": 2, "sensorValue": 1}}
{"data": {"messageId": 1, "addrSrc": 4920, "addrDst": 56712, "messageSize": 37, "queryCommand": 0, "query": "Sensor Value", "queryAns": "1.00"}}
{"data": {"messageId": 7, "addrSrc": 7044, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 10, "inMessages": 2, "sensorValue": 18}}
{"data": {"messageId": 4, "addrSrc": 56712, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 5, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 0, "addrSrc": 4920, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 1, "inMessages": 0, "sensorValue": 1}}
{"data": {"messageId": 8, "addrSrc": 7044, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 11, "inMessages": 2, "sensorValue": 18.125}}
{"data": {"messageId": 30, "addrSrc": 11028, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 31, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 3, "addrSrc": 31032, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 4, "inMessages": 0, "sensorValue": 0}}
{"data": {"messageId": 1, "addrSrc": 4920, "addrDst": 56712, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 4, "ledStatus": 0, "outMessages": 37, "inMessages": 0, "sensorValue": 1}

```

Figura 25: Captura que muestra mensajes MQTT con los valores del sensor. Fuente: Elaboración propia.

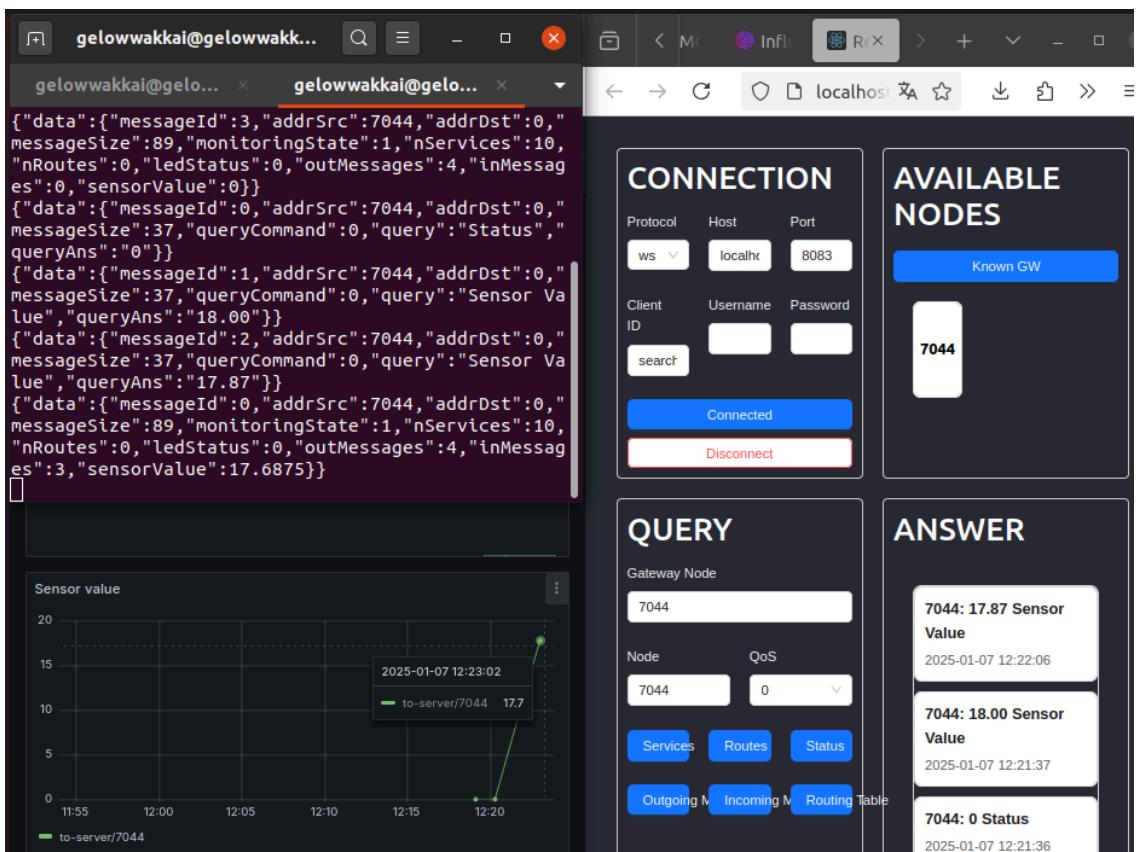


Figura 26: Captura de React al hacer la query "Sensor Value". Fuente: Elaboración propia.

En esta sección de implementación de sensores, no fue todo a la perfección, por lo que a continuación, explico que se probó a hacer y no se consiguió. Se intentó conectar un sensor de luz a las placas, pero al hacer la conexión y cargar el programa, los valores del sensor no cambiaban independientemente de la cantidad de luz a la que estaban expuestos. A esto se supuso que: o bien, los pines analógicos de las placas no funcionaban (improbable), o bien, que el único sensor de luz del que disponemos, no funcionase. Dado que disponemos de muy pocos sensores analógicos para comprobar, no fué posible determinar la causa real del problema.

Por último en esta parte de sensores, se ha probado a tener el sistema en marcha durante un período de tiempo “extenso”. En este caso se ha dejado funcionar al sistema durante 2 días (48 horas), y se han obtenido los siguientes resultados mostrados en el gráfico.

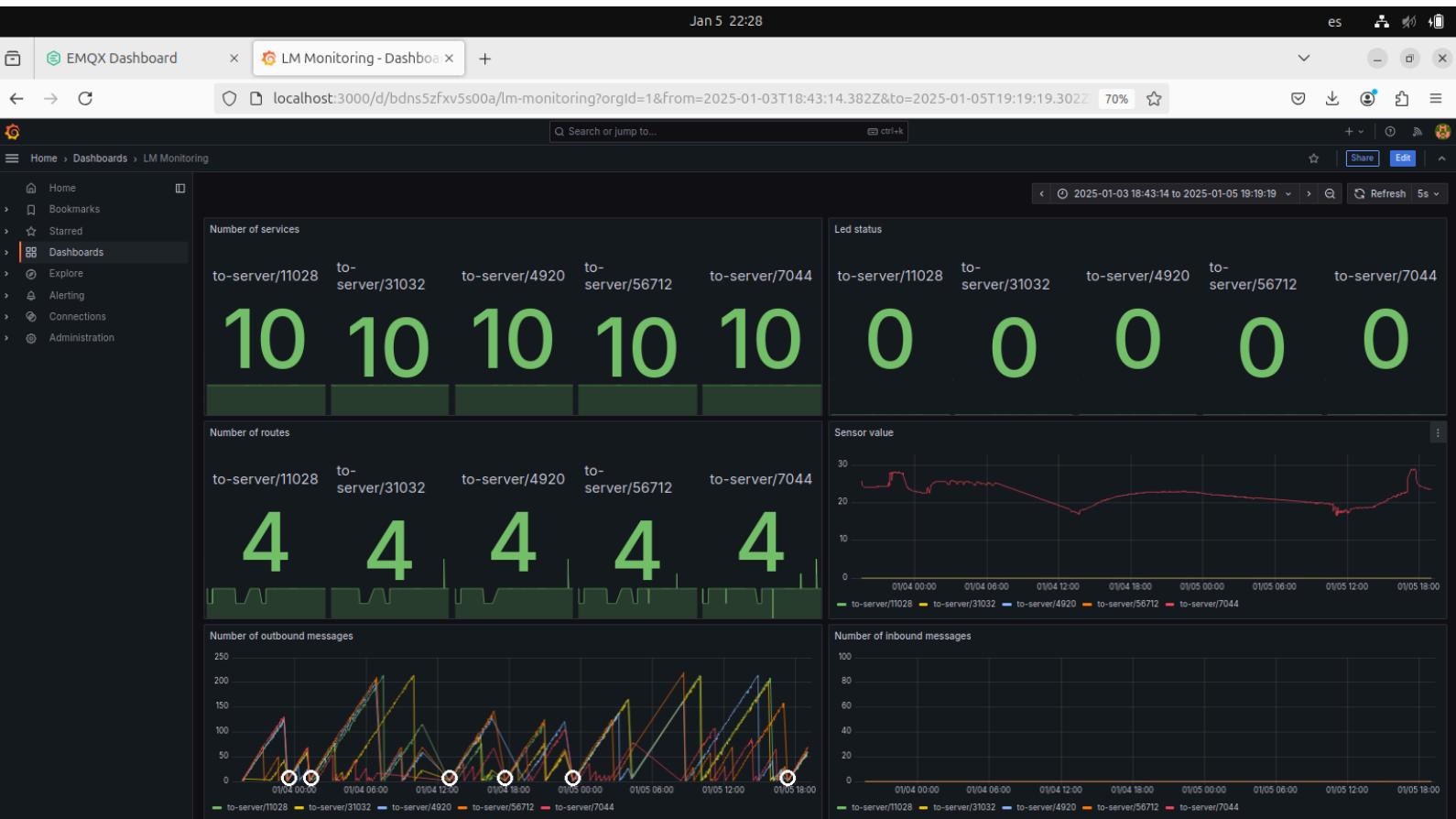


Figura 27: Captura de Grafana con los valores recogidos durante 48 horas. Fuente: Elaboración propia.

Se ha de tener en cuenta que se ha modificado levemente el programa, para evitar la desconexión constante de las placas. En la figura 27, se muestran con los círculos blancos estas desconexiones.

Tal como estaba definido anteriormente, las placas, al no recibir/enviar información durante un periodo de tiempo de 95 segundos, procedían a desconectarse automáticamente. Este período de tiempo se ha incrementado a 30 minutos cambiando la línea en el archivo *VisNodes.js* que se encuentra en el directorio: *Cross-Network-LoRaMesher/MonitoringService/query/src/components/Hook*

```
C/C++  
...  
if (currentTime - lastTimestamp <= 95000)  
...
```

Modificando el valor 95.000 por 1.800.000 conseguimos cambiar este 1.50 minutos por 30 minutos.

Nótese también que, las marcas que se indican en la imagen significan que las placas se desconectaron, por lo que se tuvieron que volver a conectar manualmente haciendo el reset de la placa desconectando y conectando de nuevo el *USB* que les proporcionaba la alimentación.

En cuanto a la parte de llevar el sistema a la máquina virtual, se han conseguido poner en marcha los servicios, tal como se ha explicado anteriormente, y mostrado en la Figura 24.

Seguidamente se ha realizado un experimento en la máquina virtual con las reglas de *ufw* definidas. Se han conectado 2 placas indicando como *MQTT_SERVER* la *IP* de la máquina virtual, y, se observa el funcionamiento.

Vemos que una de las placas (4920) se conecta a través de la otra mediante *MQTT*, y es la otra (7044) la que hace la conexión con el servidor. Al desconectar la placa con la *IP* permitida en las reglas, el servidor deja de recibir mensajes. Esto se puede ver en la siguiente imagen.

```
{"data": {"messageId": 96, "addrSrc": 4920, "addrDst": 7044, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 1, "ledStatus": 0, "outMessages": 97, "inMessages": 0, "sensorValue": 0}}  
{"data": {"messageId": 40, "addrSrc": 7044, "addrDst": 0, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 1, "ledStatus": 0, "outMessages": 41, "inMessages": 0, "sensorValue": 22.25}}  
{"data": {"messageId": 97, "addrSrc": 4920, "addrDst": 7044, "messageSize": 89, "monitoringState": 1, "nServices": 10, "nRoutes": 1, "ledStatus": 0, "outMessages": 98, "inMessages": 0, "sensorValue": 0}}  
^C^C  
^Z juan carlos@cloudy4:~/Cross-Network-LoRaMesher/MonitoringService$ mosquitto_sub -t to-server/#
```

Figura 28: Captura que muestra mensajes *MQTT* en la máquina virtual. Fuente: Elaboración propia.

En la imagen se observa que la placa con *addrSrc* 7044 envía mensajes “independientes” a la *addrDst* 0. Esto significa que está conectado directamente, en este caso, al servidor *MQTT*. En el caso de la placa con *addrSrc* 4920, se detectan los mensajes que salen de esta, pero tienen como *addrDst* a 7044, que reenvía estos mensajes al servidor.

A continuación, se ha desconectado de la alimentación la placa con identificador 7044 y se ha vuelto a ejecutar el comando:

```
Unset  
mosquitto_sub -t to-server/#
```

Ha estado alrededor de media hora sin recibir ningún mensaje.

También podemos observar que, físicamente, en la pantalla *LED* de la placa, ha desaparecido la línea que indica la *IP* de esta placa, como se muestra en la siguiente imagen.

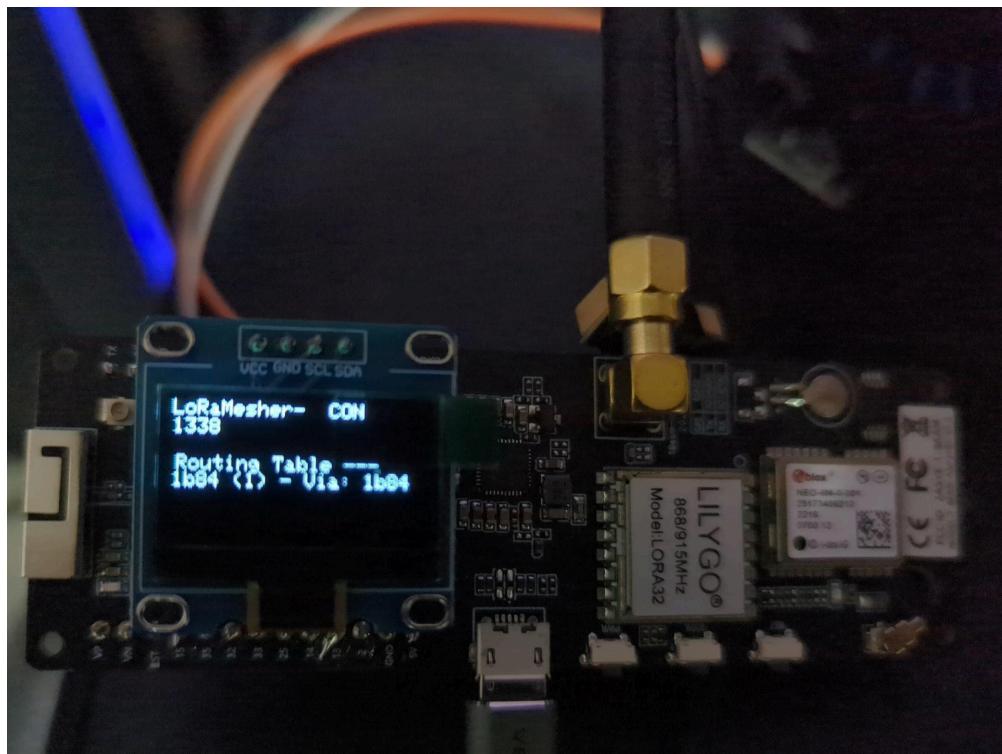


Figura 29: T-Beam con el código cargado conectado a la MV. Fuente: Elaboración propia.

9. Conclusiones

La realización del Trabajo de Final de Grado ha supuesto una experiencia muy interesante a nivel académico, personal y profesional. En este se han puesto a prueba cualidades esenciales tales como la disciplina, organización y comunicación, por encima del nivel de conocimientos. El cumplimiento de plazos y objetivos, aunque no haya sido del todo realista en cuanto a tiempo, ha supuesto el desarrollo de la metodología para el desarrollo de forma estructurada y eficiente.

Por otra parte, el proyecto ha fomentado mi capacidad de autoaprendizaje y resolución de problemas. En mi caso, no ha habido imprevistos graves que hayan ralentizado el proyecto, sin embargo, han habido algunos problemas y complicaciones, los cuales no tenían una respuesta inmediata o de fácil solución, por lo que he debido solucionarlos como bien haya podido. El afrontar estos problemas ha sido una constante durante el proyecto, que no solo ha ampliado mis conocimientos sobre el tema tratado, sino que, también, me ha dado las herramientas necesarias para que pueda afrontar futuros retos.

Otro aspecto destacable ha sido la mejora considerable en análisis. Dado que la base de mi trabajo era un código demasiado extenso conteniendo cientos de archivos y miles de líneas del programa, hacer cualquier cambio ha sido el gran problema a afrontar. La necesidad de interpretar toda esta gran cantidad de información compleja, la toma de decisiones y adaptación a imprevistos ha fortalecido mi propia capacidad de adaptabilidad y reflexión.

Por último, el tema tecnológico. Este proyecto me ha permitido profundizar en una temática interesante para mí, lo que ha consolidado mis conocimientos adquiridos durante el grado, siendo capaz de aplicarlos a un entorno práctico y real. Esta experiencia no solo me ha proporcionado una confianza mayor en mis propios conocimientos, sino que, también, ha desarrollado un interés en mí por seguir la formación en este ámbito.

10. Referencias

- [1] *Desenvolupament d'una infraestructura experimental per a xarxes LoRa en malla.* (n.d.).
<https://raco.fib.upc.edu/projectes/veure-competencies-tfq.jsp?projecte=192799>
- [2] *Projectes.* (n.d.). El Raco Fib.
<https://raco.fib.upc.edu/projectes/el-meu-projecte>
- [3] Salazar, Ó. (2023, 29 marzo). *¿Qué es la escala de madurez tecnológica (TRL)?* Euro Funding. Retrieved September 25, 2024, from
<https://euro-funding.com/es/blog/que-es-la-escala-de-madurez-tecnologica-trl/>
- [4] *Haciendo IoT con LoRa: Capítulo 2.- Tipos y Clases de Nodos.* (2017, October 2). Medium. Retrieved September 25, 2024, from
<https://medium.com/@Sabasacustico/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be>
- [5] Griffiths, J. (2024, 06 27). *Exploring Cross-network applications for a LoRa mesh network.*
- [6] Machado, A. (1998). *Caminante.* Planeta.
- [7] (n.d.). *The Things Network.* Retrieved December 11, 2024, from
<https://www.thethingsnetwork.org/>
- [8] (n.d.). *Helium - Own the Air.* Retrieved December 11, 2024, from
<https://www.helium.com/>
- [9] *Serval Project.* (2024, 03 19). Serval Project.
https://en.wikipedia.org/wiki/Serval_Project
- [10] (n.d.). *Dragino :: Open Source WiFi, Linux Appliance.* Retrieved December 11, 2024, from <https://www.dragino.com/>

- [11] ZigBee. (2024, 11 23). ZigBee. <https://ca.wikipedia.org/wiki/ZigBee>
- [12] Sueldos. (n.d.). Indeed. Retrieved October 9, 2024, from <https://es.indeed.com/career/salaries>
- [13] Agència Tributària: 3.5.4 Taula d'amortització simplificada. (2022, April 29). Agencia Tributaria. Retrieved October 9, 2024, from https://sede.agenciatributaria.gob.es/Sede/ca_es/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html
- [14] Mercado inmobiliario en Vilafranca del Camí. (n.d.). Indomio. Retrieved October 9, 2024, from <https://www.indomio.es/mercado-inmobiliario/cataluna/vilafranca-del-cami/>
- [15] Aigua de Rigat Vilafranca del Camí: Teléfono de Agua. (n.d.). Tarifas de agua. Retrieved October 9, 2024, from <https://tarifasdeagua.es/oficinas/barcelona/vilafranca-del-cami>
- [16] Precio Luz » Precio de la luz hoy en Barcelona » Precio luz hoy en Vilafranca del Camí Ahorra en tus repostajes con Repsol Precio luz hoy en Vilafranca del Camí. (2024, Octubre 09). Retrieved October 9, 2024, from <https://precioluz.info/precio-luz-hora-hoy/barcelona/vilafranca-del-cami/>
- [17] Tarifas de Fibra desde 27€ al mes. (n.d.). O2. Retrieved October 9, 2024, from <https://o2online.es/fibra/>
- [18] El consumo de agua por persona al día. (n.d.). Fundación Aquae. Retrieved October 9, 2024, from <https://www.fundacionaqua.org/sabes-cuanta-agua-consumes-a-diario/>

- [19] ¿Cuánto consume realmente un ordenador? (n.d.). Repsol. Retrieved October 9, 2024, from <https://www.repsol.es/particulares/asesoramiento-consumo/cuanto-consume-ordenador/>
- [20] PROJECTE EDINSOST2-ODS. (n.d.). Qüestionari Genèric Estudiants d'Enginyeria. <https://bit.ly/3wZjPLw>
- [21] Placas ESP32. (2024, November 25). Amazon. https://www.amazon.es/s?k=esp32+placa&adgrpid=120677549978&hvadid=601382925032&hvdev=c&hvlocphy=9221607&hvnetw=g&hvqmt=e&hvrand=9288466066221195423&hvtargid=kw-443240809396&hydadcr=19377_2264613&tag=hydes-21&ref=pd_sl_4oda95mhdqe
- [22] bluejedi. (2020, May). TTGO T-Beam topic. <https://www.thethingsnetwork.org/forum/t/ttgo-t-beam-topic/15297/271>