Grado en Ingeniería Informática

Trabajo de Final de Grado

Redes LoRa para la monitorización de datos en entornos industriales

Autor:
Jose Carlos Torró Belda

Supervisor:
Toni Mas Vicent
Tutor académico:
Maria Isabel Castillo Catalán

Fecha de lectura: 21 de Junio de 2021
Curso académico 2020/2021

Resumen

El proyecto que se desarrolla en esta memoria se enmarca dentro de ámbitos industriales para aplicaciones de captación de datos, además de utilizarse en lo que se conoce como agricultura inteligente, ya que actualmente está cobrando más importancia. El objetivo del proyecto por tanto es la monitorización de datos en tiempo real. Este documento recoge los aspectos técnicos y explica como se ha llevado a cabo la tarea de investigación y desarrollo. Para la realización del proyecto primero se hace un estudio de los comportamientos de diferentes tecnologías como WiFi, Bluetooth en sus nuevas versiones y LoRa bajo el contexto de plantas industriales. La monitorización de datos consiste en almacenar datos por lo que se utilizan técnicas actuales aplicadas a IoT como el protocolo MQTT y el uso de bases de datos no relacionales.

Palabras clave

ESP32, red en malla, LoRA, Raspberry, monitorizar, agricultura inteligente, industria inteligente, InfluxDB, MQTT

Keywords

ESP32, Mesh network, LoRa, Raspberry, monitor, smart-crops, smart-industry, InfluxDB, MQTT

Agradecimientos

Llegar hasta este proyecto no hubiese sido posible sin tener el apoyo de mis padres y mi hermana, ya que durante mi vida académica en los momentos más difíciles han estado apoyándome en todo momento y siempre han lidiado con que estropeara los aparatos de casa por toquetearlos.

Agradezco al grupo de investigación de HPC&A por abrirme sus puertas a la investigación durante los dos últimos años del grado dándome la oportunidad de aprender nuevas herramientas y formas de trabajar, además de poder programar en diferentes plataformas, en especial agradecer a Manel Dolz por haberme tutorizado desde el primer día en el grupo. Por otra parte, agradecer especialmente a Germán Fabregat la ayuda recibida en el proyecto y Maribel Castillo como tutora y profesora por todo el apoyo dado

Por último, agradecer a mis compañeros que hemos compartido esta etapa, desde risas hasta los peores momentos. En especial a Jonatan Tomás Caro y a Samir Haganu por haber formado parte de este camino.

Índice general

Capítulo 1

Introducción

Contents

En este capítulo contextualizamos el proyecto que llevamos a cabo en la empresa Más Ingenieros dentro las prácticas curriculares del grado, por ello, especificamos los objetivos del proyecto teniendo en cuenta que requisitos debe cumplir.

1.1. Contexto y motivación del proyecto

El proyecto que se propone se realiza durante las prácticas formativas como proyecto de final de grado. En las prácticas se aplican los conocimientos recibidos durante los cursos pasados, y, concretamente, del itinerario de Ingeniería de Computadores. Para el desarrollo de las prácticas se aplican los conocimientos recibidos durante el estudio del grado.

La empresa Mas Ingenieros, situada en Vila Real, se dedica a ofrecer sistemas de control y supervisión para optimización de procesos integrando *Smart Automation*. Mas Ingenieros trabaja en el campo de la automatización industrial, gestión técnica de edificios y el Internet de las cosas (IoT).

La empresa tiene el objetivo de obtener información útil y accesible que permita la toma de decisiones, desde cualquier lugar, momento y usuario. Los sectores que principalmente trabaja son la industria, edificios e infraestructuras para procesos y energía y edificios. Los campos que explota mayoritariamente son Big Data, sistemas de control de acceso, sistemas de gestión energética y programación avanzada de dispositivos móviles.

Más Ingenieros es una empresa pequeña por lo que no tiene un gran número de departa-

mentos. La oficina técnica especializada en la instalación y mantenimiento de los controladores lógicos programables (PLCs) conectados a las plantas industriales y que permiten manejar el comportamiento de determinadas partes de la misma. También, el departamento de administración que se encarga de las tareas de carácter administrativo en la empresa como son: recibir material de los proveedores y emitir facturas a los clientes; emitir las nóminas a los trabajadores; administrar los recursos humanos y financieros de la empresa; es decir, como su nombre indica se realiza las funciones de carácter administrativo en la empresa.

El departamento de Innovación y Desarrollo cobra una especial relevancia, puesto que debido a la temática en la que se enmarca la empresa necesita estar a la vanguardia de las tecnologías que se estén utilizando. El principal objetivo de este departamento, por ello, es estudiar las nuevas tecnologías que haya en el mercado y analizar como aplicarlas a sus propios productos o a la creación de otros nuevos más adaptados a sus clientes.

El proyecto que se presenta esta enmarcado dentro de esta última línea y se desarrolla dentro del departamento de innovación. Este trabajo se relaciona con los equipos de control de acceso a una planta industrial. En el contexto de la pandemia de COVID-19, la empresa ha observado que los sistemas de accesos a las plantas son muy rudimentarios respecto a otros procesos industriales. Es decir, cuando un trabajador o trabajadora llega a una planta industrial debe firmar una hoja donde se registra su entrada y salida. Si además se quisiera controlar los valores de temperatura, humedad o gases o cualquier otro parámetro de estas características en la propia planta, un inspector o inspectora debería apuntarlo manualmente. Esta forma de trabajar hace que las condiciones de trabajo del operario u operaria que lleva a cabo este control no estén garantizadas en todo momento.

Por otro lado, si en una inspección de trabajo se quisiera conocer a que temperatura ha estado sometido un operario u operaria en un momento concreto debería haber alguien que se encargase de captar esos valores por toda la planta industrial.

Teniendo en cuenta toda esta problemática, este proyecto se centra en que la captación de datos de estas características sea automática y se tengan valores en tiempo real de qué está ocurriendo en el contexto de la planta industrial en todo momento. Más tarde, se pueden cruzar con los trabajadores y trabajadoras que han estado trabajando y bajo que condiciones lo han hecho. Los datos se almacenarán directamente sobre una base de datos y en cualquier momento se puede disponer de ellos para realizar cualquier consulta o estudio.

Dado el paralelismo funcional del control ambiental de una planta industrial con el de una plantación agrícola, la empresa pretende explotar este proyecto en el entorno del mundo rural en grandes plantaciones agrícolas. Actualmente, en este entorno se están aplicando nuevas técnicas de agricultura inteligente, donde los datos captados en tiempo real están cobrando especial importancia para poder hacer cultivos más sostenibles y óptimos con el medio ambiente. En este caso, los datos recogidos se almacenan en una base de datos y mostrarían la información del estado de una planta agrícola y sus cultivos. En un momento dado, el agricultor podría visualizar estos datos y tomar decisiones sobre qué cuidados requiere la plantación. También se podrían utilizar para la realización de estudios posteriores que ayudase a conseguir mejores cultivos.

Este proyecto se incluye dentro de las operaciones de I+D de Más Ingenieros para en un futuro poder ofrecerlo e implantarlo a los clientes.

1.2. Objetivos del proyecto

El objetivo principal del proyecto es la automatización de la captación y almacenamiento de datos contextuales de una planta industrial del tipo temperatura ambiental, humedad, nivel de gases o temperatura del suelo. Para ello se debe conseguir comunicar dispositivos sin interferencias en plantas industriales y enviar los datos capturados hasta un servidor para que los almacene sobre una base de datos sin que se produzcan pérdidas.

Este objetivo se puede dividir en tres sub objetivos:

1. En primer lugar, se debe conseguir hacer una red que permite enviar información a través de ella sin pérdidas, teniendo en cuenta que el uso final será industrial, por lo que no puede haber interferencias con otros dispositivos que formen la planta.

2. La red se deberá conectar con un dispositivo que haga de puente para poder almacenar los datos en una base de datos.

3. Almacenamiento de la información para poder ser procesada posteriormente.

1.2.1. Alcance del proyecto

El alcance del proyecto se divide en el alcance funcional, organizativo e informático.

Alcance funcional

El producto que se desarrolla debe almacenar todos los datos contextuales recogidos en una planta industrial en una base de datos. Para la recolección de datos se crea una red de dispositivos donde uno o varios nodos principales se conectarán con el dispositivo que almacene toda esa información.

Alcance organizativo

El producto que se desarrolla se encuentra en el departamento de I+D de la empresa por lo que está dentro de la fase de investigación y desarrollo para futuras aplicaciones en otras empresas y entornos.

El objetivo es que este producto sea utilizable para venderlo a futuros clientes implementando nuevos sistemas de captación de datos ya utilizados en contextos como *Smart Cities*. En este caso, aplicado a niveles industriales y agrónomos.

Alcance informático

El sistema debe almacenar todos los valores recolectados en una base de datos. Los valores se transportarán usando una red reduciendo el coste de *gateways*.

1.3. Descripción del proyecto

El objetivo de este proyecto por parte de la empresa es desarrollar un prototipo que sea capaz de capturar información desde una planta industrial, enviarla a través de una red de interconexión y almacenarla en una base de datos que se encontrará localizada en un servidor. A partir de este prototipo se pretende desarrollar un producto nuevo que la empresa pueda vender a sus clientes.

Así pues, la realización de este proyecto ha partido desde cero y ha requerido de todas las fases de desarrollo, comenzando por una parte considerable de estudio de las distintas tecnologías que se pueden utilizar para implementarlo.

En primer lugar, se han estudiado los distintos redes de comunicación que existen. El objetivo es utilizar una tecnología que permita la recogida de datos y el envío de los mismos dentro de entornos industriales. Este aspecto es muy importante, ya que en una planta industrial hay mucha maquinaria que genera interferencias y ruido y puede provocar que la información no se envíe de la forma correcta produciendo perdidas y errores. Así pues, la primera parte del proyecto ha consistido en estudiar los tipos de redes de comunicación que mejores resultados ofrecen desde el punto de vista de tolerancia a fallos e interferencias para recopilar información en una planta industrial. Las tecnologías estudiadas han sido WiFi, LoRa o Bluetooth con el objetivo de evaluar y seleccionar aquella que mejor se adapte a las condiciones indicadas.

Después de analizar las ventajas e inconvenientes de las diferentes tecnologías y, teniendo en cuenta las necesidades de la empresa, se decidió que el sistema utilizado sería LoRa. A continuación, se diseña la red de comunicación junto con los dispositivos que actuarán como nodos de la misma. La red se diseñará con una topología tipo malla. Los nodos deben recoger información desde los sensores y/o enviarla a otros nodos asegurando que no haya pérdida de información, ni se transmitan errores y asegurando que la información llegue a su destino. Esta red también debe permitir que, a medida que se añadan sensores, se pueda capturar y enviar esta nueva información.

La información recogida se enviará a un nodo final de la red de interconexión. Este nodo estará conectado al servidor. Mediante el protocolo MQTT el nodo enviará al servidor los datos, que se tratarán y almacenarán en una base de datos o relacional.

Resumiendo, el prototipo diseñado será una red en malla formada por distintos nodos del tipo ESP32 que serán capaces de recoger la información proporcionada por los sensores y enviarla a través de ellos, a partir de una red de interconexión LoRa, hasta un destino final que se comunicará con el servidor. Este servidor estará formado por una Raspberry Pi que contiene la base datos donde se almacenará la información. La base de datos será del tipo no relacional, llamada InfluxDB.

1.4. Estructura de la memoria

La memoria que se presenta contiene 6 capítulos.

- El capítulo 1 introduce el proyecto realizado en el contexto y ámbito de uso, así como el objetivo que persigue.
- En el Capítulo 2 se describe la planificación seguida en el proyecto y que metodología se ha seguido para el desarrollo. También se estima el coste total del sistema y como se ha realizado el seguimiento de la empresa.
- En el capítulo 3 se analiza el sistema y especifica como se ha diseñado, es decir, qué arquitectura se ha seguido.
- El capítulo 4 contiene la implementación del sistema, además se entra en detalle en como se ha validado y verificado.
- En el capítulo 5 se concluye el proyecto explicando los aspectos técnicos concluidos y el análisis personal del desarrollo e implementación.

Capítulo 2

Planificación del proyecto

Contents

2.1. Metodología

La metodología utilizada para la planificación del proyecto es la predictiva que es la que utiliza la empresa Mas Ingenieros para el desarrollo de los proyectos.

En esta metodología se lleva a cabo una planificación de tareas inicial y se estima el tiempo de su realización. Después se valora si la situación es correcta a medida que se lleva a cabo cada una de las tareas. Semanalmente se ha realizado una reunión con el tutor de la empresa que valora y comenta como se ha desarrollado cada una de las etapas del proyecto, también se plantean todas las dudas que hayan surgido durante el desarrollo semanal del proyecto y se decide si los plazos de las tareas se amplían o no.

El seguimiento del proyecto se lleva a cabo mediante Trello. Esta herramienta permite organizar las tareas a realizar mediante tableros como se muestra en la figura 2.1. El objetivo de su utilización es mejorar la rutina de trabajo organizando las prioridades, tiempos, avisos y otras opciones.

Teniendo en cuenta el proyecto y los objetivos a conseguir, en el siguiente apartado se muestran las tareas a desarrollar y el tiempo en el que se espera conseguir cada una.

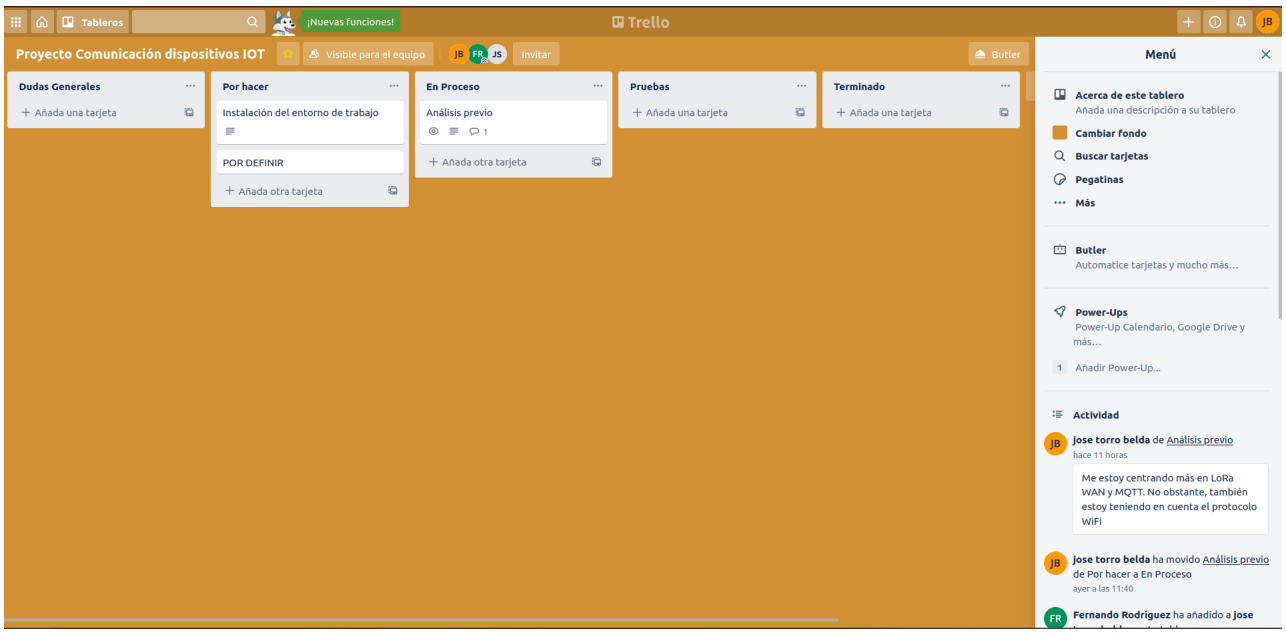


Figura 2.1: Herramienta de organización de tareas Trello.

2.2. Planificación

En la figura 2.2 se enumeran las tareas en las que se ha dividido y llevado a cabo para la realización del proyecto. También se realiza la planificación temporal de las mismas teniendo en cuenta que el proyecto consta de 300 horas para la realización de las prácticas en empresa.

1. **Estudio previo del tipo de redes y protocolos**: Se tienen en cuenta los caso de usos del proyecto y los requisitos para hacer un análisis detallado de los diferentes protocolos y tipos de red que existen.

2. **Instalación de software y hardware para teletrabajo**: Debido a la situación actual la empresa opta por el teletrabajo, por ello, una vez recibido todo el material se instala en el lugar de trabajo.

3. **Implementación de sensores de temperatura, humedad y gases**: Se programa en las ESP32 el software de lectura de los datos captados por los sensores.

4. **Conexión de microcontrolador a la base de datos**: Usando los protocolos estudiados en el punto 1, se conecta el microcontrolador a la base de datos instalada en la Raspberry Pi que actúa como servidor.

5. **Programación del protocolo de red**: A partir de los protocolos estudiados en el punto 1 se elige el que más se adapte al contexto industrial.

6. **Establecer formato de paso de mensajes por la red y programar la replicación de mensajes**: Se establece el formato en que los mensajes se replican por la red y añadir información por cada nodo programado.

7. **Conexión de red con la base de datos**: Esta tarea se diferencia de la 4 en que conectar la red se refiere a la replicación de mensajes con todos los datos recogidos por los sensores.

8. **Refinar la red y conexión con la base de datos**: Reducir la tasa de errores.

9. **Depuración de código**: Limpiar y depurar código.

2.3. Estimación de recursos y costes del proyecto

En este apartado se detallan los datos del proyecto. Por un lado se calcula el hardware necesario para desarrollo. A continuación el software utilizado para que ese hardware funcione correctamente. Finalmente, los recursos humanos necesarios para la implementación de dicho proyecto. El coste temporal para la realización de este documento se calcula aparte.

Recursos software

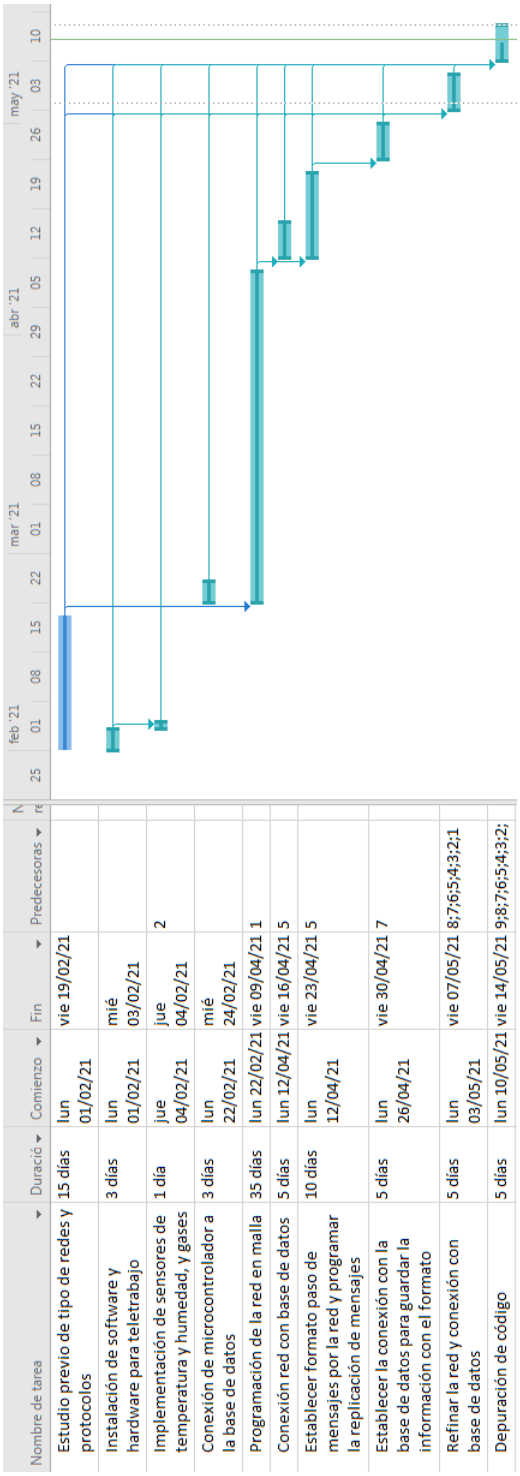
El software que se utiliza tiene un coste de 0€ puesto que todo el software es de libre utilización. Se utiliza el IDE de Arduino para el desarrollo de código, grabar el código en el microcontrolador y poder ver los logs en un ordenador al que esté conectado por Serial. También, utilizamos Visual Code proporciona un entorno de programación más cómodo y sencillo para el desarrollo de código. Además, se utiliza Node-RED que es una herramienta de programación por bloques de código de forma visual. También, se utiliza Chronograf para la visualización de datos en tiempo real de la base de datos. Por último, para el seguimiento del código y control de versiones utilizamos GitHub.

En cuanto al software utilizado para el seguimiento se ha usado Slack, esta es una aplicación de mensajería para empresas que conecta a las personas con la información que necesitan con lo que se tiene comunicación. Para las reuniones online hemos utilizado Google Meet. Por último, para dudas de manejo de código y tener mayor fluidez resolviendo problemas se ha usado Team Viewer[1].

Recursos hardware

El hardware utilizado añade coste económico al proyecto, por ello se estudia y analiza cuál es el más barato y eficiente para esta implementación, así pues se elige los microcontroladores ESP32 LoRa TTGO. Luego, al coste debemos añadir los sensores que se utilizan para captar datos del entorno como temperatura y humedad. También se utiliza una fuente de alimentación que alimenta el sistema que está conectado en una protoboard, ya que el proyecto está en la fase de investigación y desarrollo. El sistema debe tener una red internet local por lo que se hace uso de un router TP-Link.

[1]Team Viewer es una herramienta que permite controlar un ordenador de forma remota desde otro equipo, a priori es un un software privado pero que tiene su versión gratuita para no profesionales.



Nombre de tarea	Duració	Comienzo	Fin	Predecesoras
Estudio previo de tipo de redes y protocolos	15 días	lun 01/02/21	vie 19/02/21	
Instalación de software y hardware para teletrabajo	3 días	lun 01/02/21	mié 03/02/21	
Implementación de sensores de temperatura y humedad, y gases	1 día	jue 04/02/21	jue 04/02/21	2
Conexión de microcontrolador a la base de datos	3 días	lun 22/02/21	mié 24/02/21	
Programación de la red en malla	35 días	lun 22/02/21	vie 09/04/21	1
Conexión red con base de datos	5 días	lun 12/04/21	vie 16/04/21	5
Establecer formato paso de mensajes por la red y programar la replicación de mensajes	10 días	lun 12/04/21	vie 23/04/21	5
Establecer la conexión con la base de datos para guardar la información con el formato	5 días	lun 26/04/21	vie 30/04/21	7
Refinar la red y conexión con base de datos	5 días	lun 03/05/21	vie 07/05/21	8;7;6;5;4;3;2;1
Depuración de código	5 días	lun 10/05/21	vie 14/05/21	9;8;7;6;5;4;3;2;

Figura 2.2: Estimación de planificación de tareas realizadas para el proyecto.

El coste de recursos software y hardware se muestra en la tabla 2.1.

Recursos			
Software	**Cantidad**	**Coste (€)**	**Total (€)**
IDE Arduino	1	0	
Node-RED	1	0	
Visual Studio	1	0	
Chronograf	1	0	
GitHub	1	0	0
Hardware			
ESP32 LoRa TTGO	5	13.48	67.4
Raspberry Pi	1	60.95	60.95
Protoboad	2	4.99	9.98
Sensor DHT11	2	5.49	10.98
Sensor Gas MQ-2	1	4.99	4.99
Fuente alimentación MB102	1	2.22	2.22
Cable alimentación 12V	1	4.90	4.90
Router TP-LINK Archer C6- AC1200	1	39.05	39.05
Total			**200.47**

Tabla 2.1: Coste de recursos hardware y software utilizados.

Recursos humanos

En cuanto a coste de recursos humanos, según el convenio de las TIC de 2021, un programador junior, que se encuentra en el Área 3 Grupo E Nivel 1 según las tablas salariales[2], el sueldo es de 12.701,05€ anuales, lo que equivale a 1.058,42€/mes.

A estos costes, hay que añadir el coste mensual de Internet y Electricidad usada para la creación del proyecto, es decir, los costes indirectos variables serían los que se muestran en la tabla 2.2.

Recurso	**precio/mes**	**total (3 meses) (€)**
Electricidad	10	30
Internet	20	60
Total		**90**

Tabla 2.2: Costes variables.

Por ello, los costes totales de la realización del proyecto son los mostrados en la tabla 2.3. Por tanto, el coste total es de 3465,73€.

[2]Nóminas en convenios de las TIC[2]

Recurso	total (€)
Recursos software y hardware	200.47
Recursos mensuales	90
Recursos humanos	3175.26
Total	**3465.73**

Tabla 2.3: Costes totales del proyecto.

2.4. Seguimiento del proyecto

El seguimiento del proyecto por parte de superior de la empresa se ha realizado semanalmente. Así, cada semana se ha realizado una reunión online para resolver dudas, revisar los hitos realizados y observar posibles mejoras en lo realizado.

Para el seguimiento se ha usado Slack. Esta es una aplicación de mensajería para empresas que conecta a las personas con la información que necesitan. Así se tiene comunicación diaria durante el horario de trabajo marcado y se ha conseguido que dudas surgidas en el trabajo diario, pudiesen ser resueltas por el supervisor de forma rápida. En cuanto las dudas de software, se ha utilizado la herramienta TeamViewer. El tutor se podía conectarse al ordenador donde se estaba desarrollando el proyecto desde la oficina y así se resolvieron diferentes problemas surgidos en el desarrollo del software, sobre todo con la implementación de la comunicación con las bases de datos.

Aunque haya primado el teletrabajo, también se hicieron diferentes reuniones presenciales en la oficina de Mas Ingenieros en Vila Real. En estas se hizo un seguimiento más exhaustivo comprobando que el sistema funcionaba con la captación de datos con los sensores.

Capítulo 3

Análisis y diseño del sistema

Contents

En este capítulo se realiza el estudio del sistema que se ha desarrollado indicando las tecnologías utilizadas. Así mismo, se definen los requisitos que necesita el sistema y qué casos de uso tiene teniendo en cuenta los usuarios finales.

3.1. Análisis del sistema

Este proyecto desarrolla un sistema que implica la utilización de diferentes tecnologías. En primer lugar, se hace un estudio de tecnologías y protocolos que se utilizan en ámbitos industriales para las comunicaciones teniendo en cuenta los objetivos del proyecto. También se estudian los sistemas de almacenamiento de información que propone la empresa Mas Ingenieros. Para la realización del proyecto se estudian las diferentes redes que se aplican en entornos industriales para la comunicación con tecnologías sin cableado.

3.1.1. Redes de comunicación

El primer paso de la realización del proyecto es el estudio previo de las diferentes redes de comunicación que se aplican en entornos industriales.

LoRa

LoRa[4] es una tecnología inalámbrica como el WiFi, Bluetooth. Es decir, LoRa es un tipo de modulación en radiofrecuencia como AM o FM, patentado por Semtech[16]. Aunque fue desarrollada por esta empresa, ahora mismo está gestionada por "LoRa Alliance", por ello, cualquier fabricante de hardware que quiera trabajar con esta tecnología se certifica en esta alianza.

La gran ventaja de LoRa es que permite comunicaciones de largas distancias (distancias kilométricas), es sólida ante interferencias y de bajo consumo.

Esta tecnología se basa en "chirp modulation"[18] o frecuencia modulada pulsada. Según la documentación, se modula con un periodo fijo y una secuencia de ciclo de trabajo fija. Al principio de cada pulso transmitido, la frecuencia de la portadora se modula en frecuencia, causando un ensanchado adicional. El patrón de la modulación en frecuencia dependerá de la función de ensanchado que se elija.[18]

Las ventajas de la tecnología LoRa son las siguientes:

- Alta tolerancia a interferencias.
- Alta sensibilidad para recibir los datos (-168dB).
- Basada en "chirp modulation".
- Es una tecnología de largo alcance, en condiciones óptimas[1] entre 10 y 20 km de radio.
- Bajo consumo.

La principal ventaja de LoRa es que el consumo para comunicar dispositivos es bajo, si a esto se le añade que los microcontroladores que la usan tiene también bajo consumo, esto permite que en algunos momentos no se conecte corriente eléctrica y el sistema pueda estar alimentado con baterías, alargando la vida útil de las baterías se alarga. Como desventaja encontramos que tan solo se pueden emitir hasta 255 Bytes. Las características principales de LoRa a nivel físico son las que se muestran en la tabla 3.1.

LoRa trabaja en la frecuencia 868 MHz o 433 MHz, dependiendo del territorio se utiliza una u otra. Por ejemplo en Europa se utiliza 868, ya que está regulada y se estipula que los nodos no pueden usar más del 1 % de la carga de red por canal, por esta razón no hay carga permanente en la red de radio. De igual forma las bandas están reguladas según el país:

[1] Por condiciones óptimas se entiende situaciones donde no hay interrupciones, pérdida de paquetes

- 868 MHz en Europa.
- 915 MHz en Estados Unidos.
- 920 MHz en Japón.
- 470-510 MHz en China.

	Europa	América del Norte
Banda de frecuencia	**867-869 MHz**	**902-928 MHz**
Canales	10	64+8+8
Canal banda ancha ascendente	125/250 kHz	125/500 kHz
Canal banda ancha descendente	125 kHz	500 kHz
TX Encendido	+14 dBm	+20 dBm
SF Up	7-12	7-10
Velocidad de datos	250 bps - 50 kbps	980 bps - 21.9 kbps
Link Budget UP	155 dB	154 dB

Tabla 3.1: Tabla características principales LoRa a nivel físico [7]

Teniendo en cuenta estas ventajas, LoRa se aplica a redes IoT o para conexiones de largas distancias. Por ello, actualmente se utiliza en Smart Cities o en plantas agrícolas o ganaderas, también para construir redes privadas de sensores, que es el caso que nos ocupa este proyecto.

Wi-Fi/802.11

El WiFi [22] es una tecnología que permite conectar dispositivos entre sí, además mediante un punto de acceso o router da acceso a Internet. Esta tecnología cumple el estándar 802.11 [21] de redes inalámbricas de área local.

El estándar 802.11 son normativas creadas por Instituto de Ingenieros Eléctricos y Electrónicos (IEEE). Este estándar utiliza el protocolo *half duplex* utilizando el aire como medio de transmisión. Los conceptos a tener en cuenta con el estándar son los siguientes:

- Estaciones: Son dispositivos que deben tener una interfaz de red para poder usar el estándar.
- Medio: Sea radiofrecuencia o infrarrojos. El WiFi funciona bajo radiofrecuencias por lo que se transmite por el aire.
- Sistema de distribución: Indica donde está el punto de acceso.
- Conjunto de Servicio Básico (BSS): Grupo de estaciones que se intercomunican. Estas pueden ser independientes si se comunican entre ellas; o de infraestructura si tienen un punto de acceso.
- Conjunto de Servicio Extendido (ESS): Unión de varios BSS.

- Área de servicio básico: Indica la capacidad que se tiene de cambiar de ubicación variando el BSS.

El WiFi opera a 2.4 GHz, al igual que el Bluetooth por lo que pueden presentar interferencias. No obstante, las nuevas versiones de WiFi se hizo para que se usaran ambas tecnologías de forma simultánea sin presentar interferencias.

Una desventaja a tener en cuenta en el contexto espacial del proyecto es que las conexiones WiFi se pueden ver afectadas por los agentes físicos que se encuentran a su alrededor, es decir, otras máquinas, paredes o metales. Esto hace que la potencia de la conexión disminuya considerablemente.

Dentro de Wi-Fi existe la tecnología IEEE 802.11ah[6] designada como Wi-Fi HaLow[19]. Esta posibilidad mejora el Wi-Fi al trabajar sobre una frecuencia de 1 GHz para ofrecer un mayor alcance y una conectividad con menor consumo de energía. Como la frecuencia a la que trabaja es baja tiene una conectividad de largo alcance y las paredes no son un impedimento para el funcionamiento. Este tipo de WiFi tiene un consumo bajo de energía, por lo que tiene una alta eficiencia energética. También proporciona conectividad con IP nativa por lo que no se necesita una puerta de enlace propietaria y utiliza el cifrado WPA3.

Bluetooth

El Bluetooth "es una especificación industrial para redes inalámbricas de área personal (WPAN)"[20]. Esta tecnología sirve para la comunicación en voz y datos en la banda ISM de 2.4 GHz. Por esta razón, se facilita la comunicación entre equipos móviles eliminando cables y se pueden crear redes inalámbricas.

Existen diferentes versiones de Bluetooth donde se han actualizado los protocolos de seguridad, el alcance al que puedan llegar o diferentes aspectos técnicos. En nuestro caso nos centramos en las últimas versiones para realizar el estudio.

El rango físico que ocupan las comunicaciones Bluetooth dependen de la versión en la que se trate. A partir de 2016 se presenta la versión V5.0 de Bluetooth con la cual se consigue un rango de alcance de entre 40 y 400 metros, dependiendo de las condiciones y el contexto. Las novedades que acapara son:

- Con el estándar 5.0 el consumo energético de los dispositivos que utilizan esta tecnología desciende, por esta razón se le denomina Bluetooth LE o de bajo consumo.
- Se añade más velocidad y distancia. Según la documentación oficial con el nuevo estándar se pueden alcanzar velocidades de hasta 2 Mbps y distancias máximas (en condiciones idóneas) de hasta 200 metros, sin embargo esto se reduce considerablemente con obstáculos en medio.

El hardware que contiene la tecnología Bluetooth un dispositivo de radio que modula y transmite la señal y un controlador digital que procesa las señales digitales, este contiene una

CPU. Un dispositivo de radio Bluetooth genérico realiza tareas como envío y recepción de datos, paginación y peticiones, establecer conexiones, autenticar, negociar y establecer los enlaces, establecer el tipo de cuerpo de cada paquete.

3.1.2. Protocolo de paso de mensajes

Message Queuing Telemetry Transport (MQTT)

Para conectar la red mallada con el servidor establecemos que el protocolo de paso de mensajes será el Message Queuing Telemetry Transport (MQTT) donde un nodo de la red que actúa como cliente se encarga de enviar los paquetes al servidor.

MQTT[11] es un protocolo de paso de mensajes por subscripción, es decir, uno o varios clientes se suscriben a un "topic" y pueden enviar mensajes al nodo principal. Dicho de otra forma, hay un servidor y varios clientes, estos últimos se suscriben a un "topic" y ya pueden enviar mensajes a ese "topic". Estos "topic" se organizan jerárquicamente, además el "broker" se utiliza como filtro. El cliente publica un mensaje en el broker y este lo filtra para los "topics".

Con este protocolo se pueden crear topologías de red como en la figura 3.1 donde N clientes pueden suscribirse a los "topics". Para realizar la conexión, por defecto, MQTT funciona sobre el puerto 1883. Este protocolo, a diferencia de HTTP donde se tienen muchas cabeceras y reglas, tan solo utiliza la regla de *publish-subscribe*, por lo que es utilizado para redes congestionadas como es el caso de captación de datos.

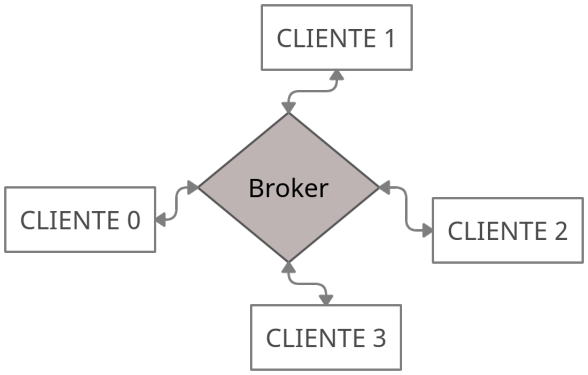


Figura 3.1: Topología con un solo broker para MQTT.

Radio Head

La implementación de la red de interconexión se hace utilizando la biblioteca de RadioHead[3] que incluye métodos y protocolos ya desarrollados con LoRa a nivel físico.

RadioHead[13] es una biblioteca para microcontroladores que permite empaquetar mensajes y está orientada para enviar y recibir mensajes empaquetados a través de diferentes tecnologías.

Los programas que utilizan esta biblioteca deben inicializar un "driver" que da un acceso de bajo nivel a los paquetes de radio y otros transportes de mensajes; luego un "manager" que proporciona los servicios de envío y recepción de mensajes de alto nivel. En nuestro caso usaremos el driver RH_RF95 que da soporte LoRa con un espectro de frecuencias extendido o uso de cargas útiles. Cuando se configura una red usando este driver, debemos especificar en cada uno de ellos la misma frecuencia y el esquema de modulación.

Todos los paquetes que se envían o reciben con el driver RH_RF95 tienen el siguiente formato:

- Modo de funcionamiento LoRa.
- 8 símbolos de preámbulo.
- Cabecera explícita.
- 4 octetos de cabecera que incluye:
  1. Destino.
  2. Quién lo envía.
  3. ID.
  4. Flags.
- Datos: 0 - 251 octetos.
- CRC.

Por otra parte esta biblioteca incluye un protocolo propio de algoritmo en malla que también usaremos para la realización de la red.

3.1.3. Tratamiento de mensajes

Node-RED

La herramienta Node-RED[8] se utiliza en el servidor para comunicar y tratar los mensajes recibidos mediante MQTT y almacenarlos en la base de datos.

Node-RED[8] es una herramienta de programación visual, esta muestra las relaciones y funciones, y permite al usuario programar sin tener que escribir una línea. Por tanto, Node-RED tiene como objetivo que conectar hardware, APIS y servicios en línea sea sencillo. Además permite que el proceso de paso de información entre productores y consumidores sea sencillo de implementar.

Esta herramienta es un editor de flujo basado en el navegador donde se puede añadir o eliminar nodos y conectarlos entre sí con el fin de hacer que se comuniquen entre ellos. Luego, Node-RED se encarga de formatear los datos y subirlos a la base de datos tal como se muestra en la figura 3.2. Se utiliza esta forma para montar el servidor porque resulta sencilla y visible

para depurar errores. Node-RED es utilizada en IoT (Internet de las cosas) que ejecuta Node.js[2], además aprovecha el modelo sin bloqueos para ejecutarlo en hardware de bajo coste.

En la figura 3.2 se muestra una visión general del servidor, en el capítulo siguiente se profundizará en la explicación.

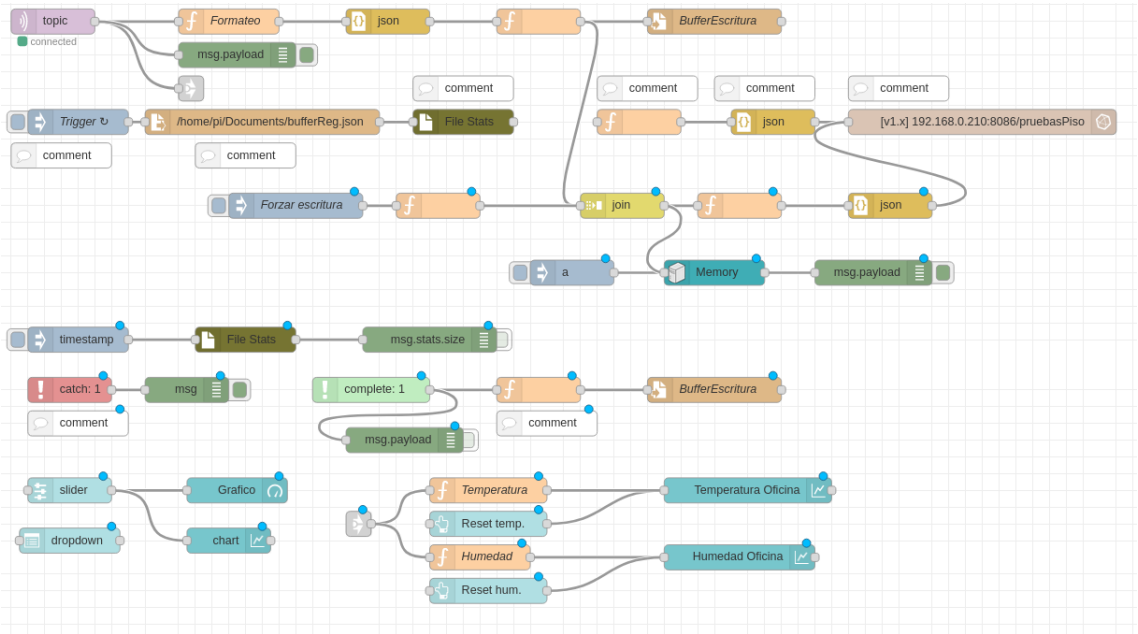


Figura 3.2: Ejemplo de proyecto realizado con node-RED.

3.1.4. Almacenamiento de datos

InfluxDB

InfluxDB es una base de datos no relacional, es decir, está diseñada con el objetivo de almacenar datos en el tiempo. Al contrario de las bases de datos relacionales como SQL pueden manejar datos en el tiempo, pero no están optimizadas para esa carga de trabajo. Dicho de otra forma, InfluxDB puede almacenar grandes volúmenes de datos en el tiempo y analizarlos en tiempo real de forma rápida y eficiente.

A diferencia de SQL, en InfluxDB el tiempo identifica un valor en cualquier serie de datos. Mientras, en SQL se necesita una clave principal está preestablecida por el sistema.

En InfluxDB la organización de los datos puede variar con el tiempo, es decir, no es necesario realizar un diseño de tablas por adelantado para almacenar los datos. Dicho vulgarmente, "las tablas se van creando conforme se necesiten". Por ejemplo, si queremos almacenar los valores de temperatura y humedad de un sensor 1 y luego queremos almacenar valores de gases de un sensor 2 no será necesario crear una tabla en el diseño para dichos valores, sino que tan solo

[2]Node.js es un entorno de tiempo de ejecución de JavaScript.

hacemos la orden para guardar los datos y si no hay tabla para estos valores el sistema la crea automáticamente.

3.1.5. Visualización de datos

Chronograph

La visualización de los datos de InfluxDB[1] se hace mediante una aplicación web con la que se pueden crear alertas y reglas de automatización.

En la Figura 3.3 se muestra como se visualizan los datos que se están monitorizando, además de las posibles configuraciones que puede tener.

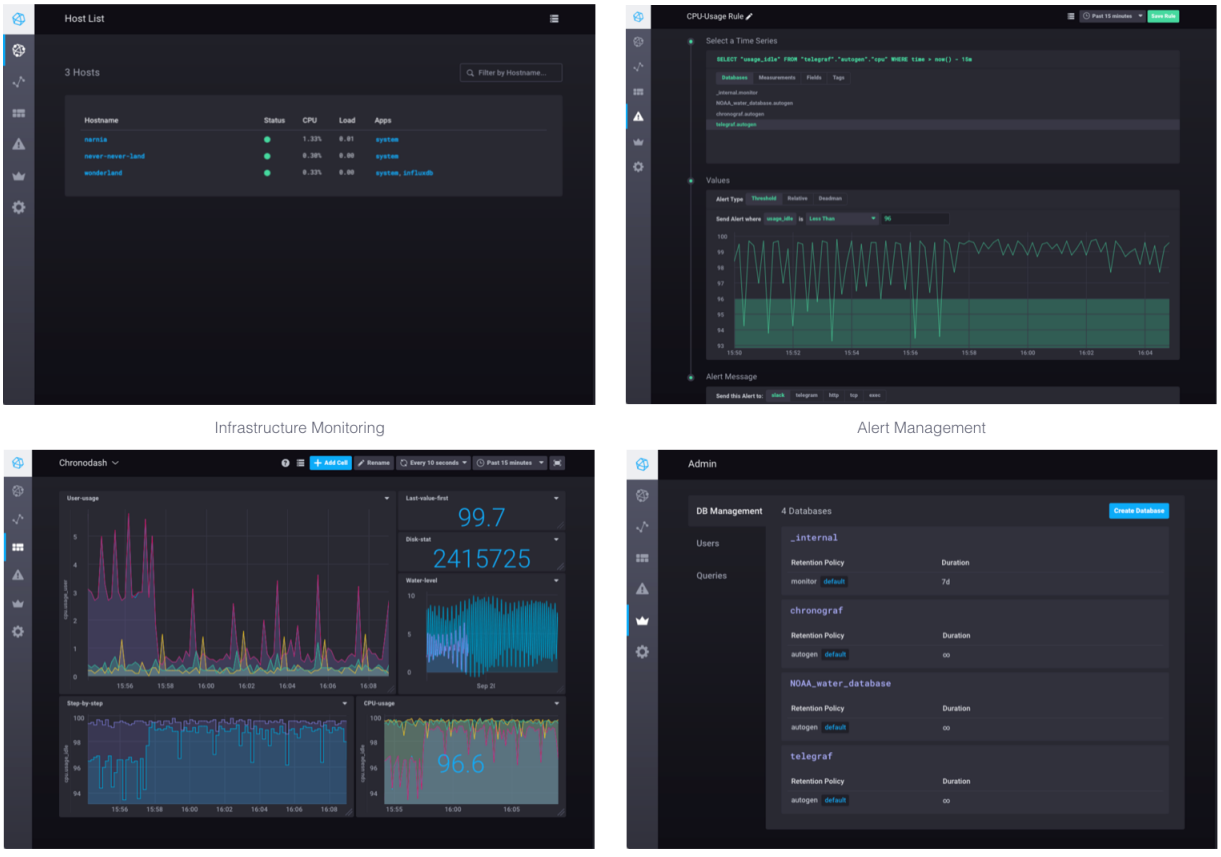


Figura 3.3: Visualización de datos en web con Chronograph. *Imagen extraída de la documentación* [1]

3.1.6. Material hardware utilizado

El material hardware que se ha utilizado para la implementación del proyecto ha sido determinado por las tecnologías utilizadas y descritas en el apartado anterior.

Rasberry Pi

El servidor para almacenar y tratar los datos captados por los microcontroladores se implementa en una Rasberry Pi 4. Este dispositivo actúa como "broker" para el protocolo MQTT y contiene la base de datos.

Raspberry Pi es un ordenador de placa reducida o placa simple de bajo coste. El hardware que contiene aunque no se especifique es libre, es decir, es un producto con propiedad registrada que mantiene el control de la plataforma, pero se permite el uso libre. Por otra parte, el software sí es *open source*[3]. De hecho, el sistema operativo que se suele utilizar es Linux por lo que la creación de un servidor es sencillo.

En el caso de este proyecto utilizamos el modelo Raspberry Pi 4 cuyas características más importantes que nos convienen son las siguientes:

- Quad core cortex A72 (ARM v8) con 1.5 GHz.
- 4GB de RAM LPDDR4.
- 2.4 y 5 GHz IEEE 802.11 wirless. Soporta BLuetooth 5.0.
- Conexión Gigabit Ethernet.
- 2 USB 30.0 y 2 USB 2.0.
- 2x micro-HDMI.
- 5V DC USB-C con un mínimo de 3A.
- Un slot para micro-SD.

TTGO LoRa ESP32

Los nodos de la red en malla están formados por los microcontroladores TTGO LoRa ESP32. Algunos de estos leen datos de los sensores como la temperatura y humedad. Otros únicamente utilizan el nodo como una pasarela para transmitir la información. Por último hay un microcontrolador ESP32 (nodo final o destino) que es quien se comunica con el servidor.

ESP32 es un chip integrado de bajo coste diseñado por Espressif Systems y fabricado por TSMC. Este chip ofrece una conectividad 802.11, además se tienen las siguientes ventajas:

[3] *El software open source es un código diseñado de manera que sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente.*[15]

- ESP32 se integra en entornos industriales y soporta temperaturas entre 40º C y 125ºC.
- Estos chips tienen un bajo consumo energético, es por ello que se utilizan en dispositivos móviles o aplicaciones IoT.
- El chip ESP32 incluye diferentes modos de energía y escalado dinámico de energía.
- Utiliza un chip híbrido para tecnologías Wi-Fi y Bluetooth. ESP32 también sirve de interfaz para estas tecnologías mediante SPO, SDIO o I2C.

En este proyecto vamos a utilizar ESP32 LoRa TTGO, que es una tarjeta basada en ESP32. Este dispositivo cuenta con una pantalla de 1.14 pulgadas, una interfaz para cargar baterías de litio y dos botones, tal como se observa figura 3.4.

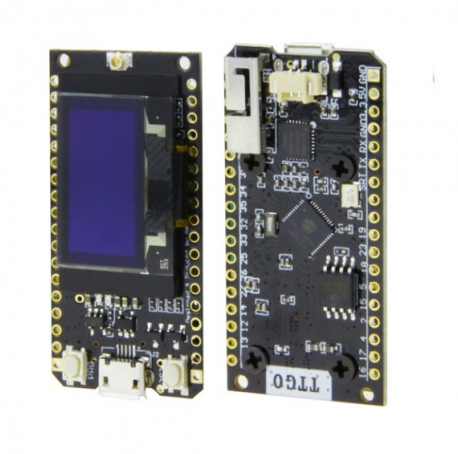


Figura 3.4: ESP32 LoRa TTGO.

LoRa TTGO incluye 36 pines de los cuales 28 son tipo GPIO, estos se pueden usar como "input" o "output" si no se utilizan para otra función, tal como se observa en el esquema de pines de la figura 3.5. En el siguiente capítulo se especificará el esquema de conexiones con los sensores elegidos.

Sensores

Para realizar pruebas con valores reales se eligen los siguientes sensores:

- **Sensor DHT11[5]**: Es un sensor de temperatura y humedad utilizado en proyectos con microcontroladores como Arduino o ESP32. Este sensor, según la documentación, puede medir temperaturas entre 0 a 50º C (± 2º C). Luego, en cuanto a humedad se puede medir entre un 20 a un 80 % (± 5 %) con una frecuencia de muestreo de un 1 Hz. Este sensor lo podemos observar en la figura 3.6.
- **Sensor gases MQ**[12]: Es un sensor de gases, la familia de sensores MQ es grande por ello, en este proyecto usaremos el sensor MQ-2 que permite medir valores como metano,

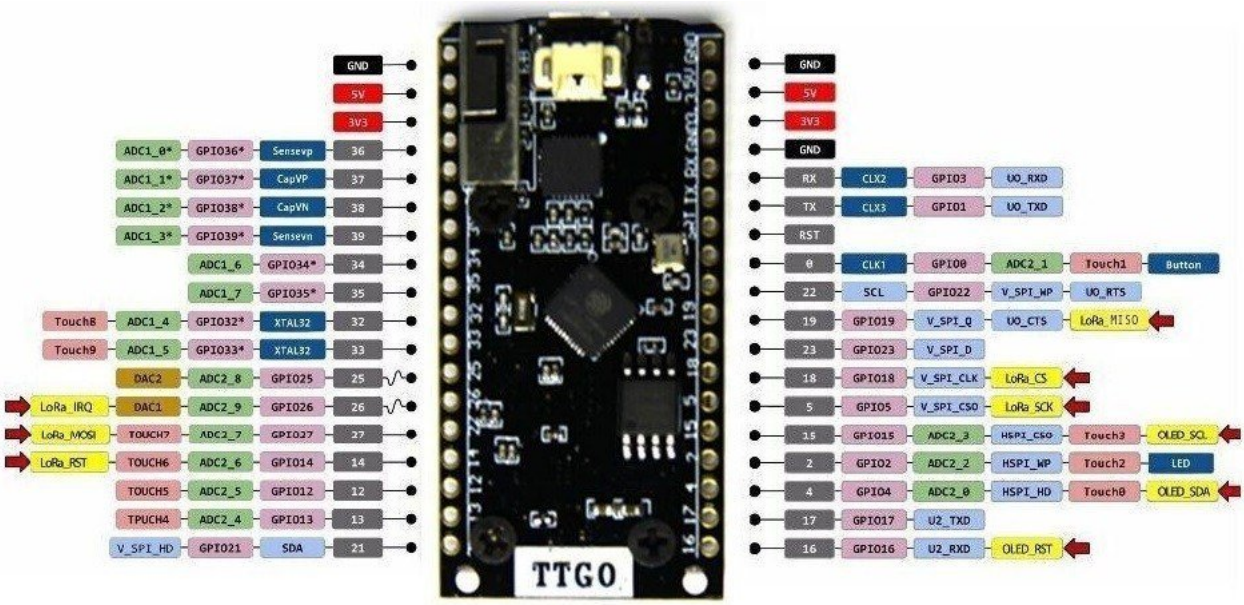


Figura 3.5: Esquema pines ESP32 LoRa TTGO.

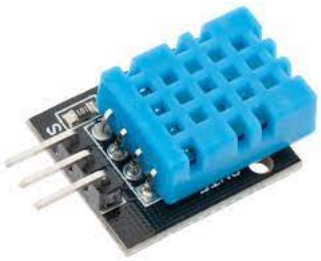


Figura 3.6: Sensor de temperatura y humedad DHT11.

butano, gas licuado de petróleo (GLP) y humo, este funciona a 5V. Este sensor lo podemos observar en la figura 3.7.

Estos sensores tan solo los utilizamos para obtener datos del entorno y alimentar la base de datos así como testear el funcionamiento de la red. En la fase de desarrollo, los valores que tengamos no son de especial interés, no implica que en una segunda fase de implementación en entornos reales los datos cobren más valor empresarial.

3.1.7. Definición de requisitos

En esta sección se definen qué requisitos debe cumplir el prototipo desarrollado para garantizar el funcionamiento atendiendo a los objetivos del proyecto. El sistema debe ser capaz de:

- Leer datos en cada nodo que se configure como lectura.

Figura 3.7: Sensor de gases MQ-2.

- Enviar los datos recogidos por los nodos a un nodo final que se comunique con un servidor a partir de una red en malla donde se reenvían evitando ciclos.

- Almacenar los datos en una base de datos de forma ordenada para que sean fáciles de analizar.

- Reconfigurarse en caso de que un nodo pierda la conexión.

- Reconfigurarse en caso de que se añada un nuevo nodo a la red.

- Ser escalable a nuevas necesidades de la empresa.

- Comunicar los nodos en grandes distancias.

3.1.8. Casos de uso

Atendiendo a los requisitos del sistema los casos de uso que lo forman se pueden observar en la figura 3.8.

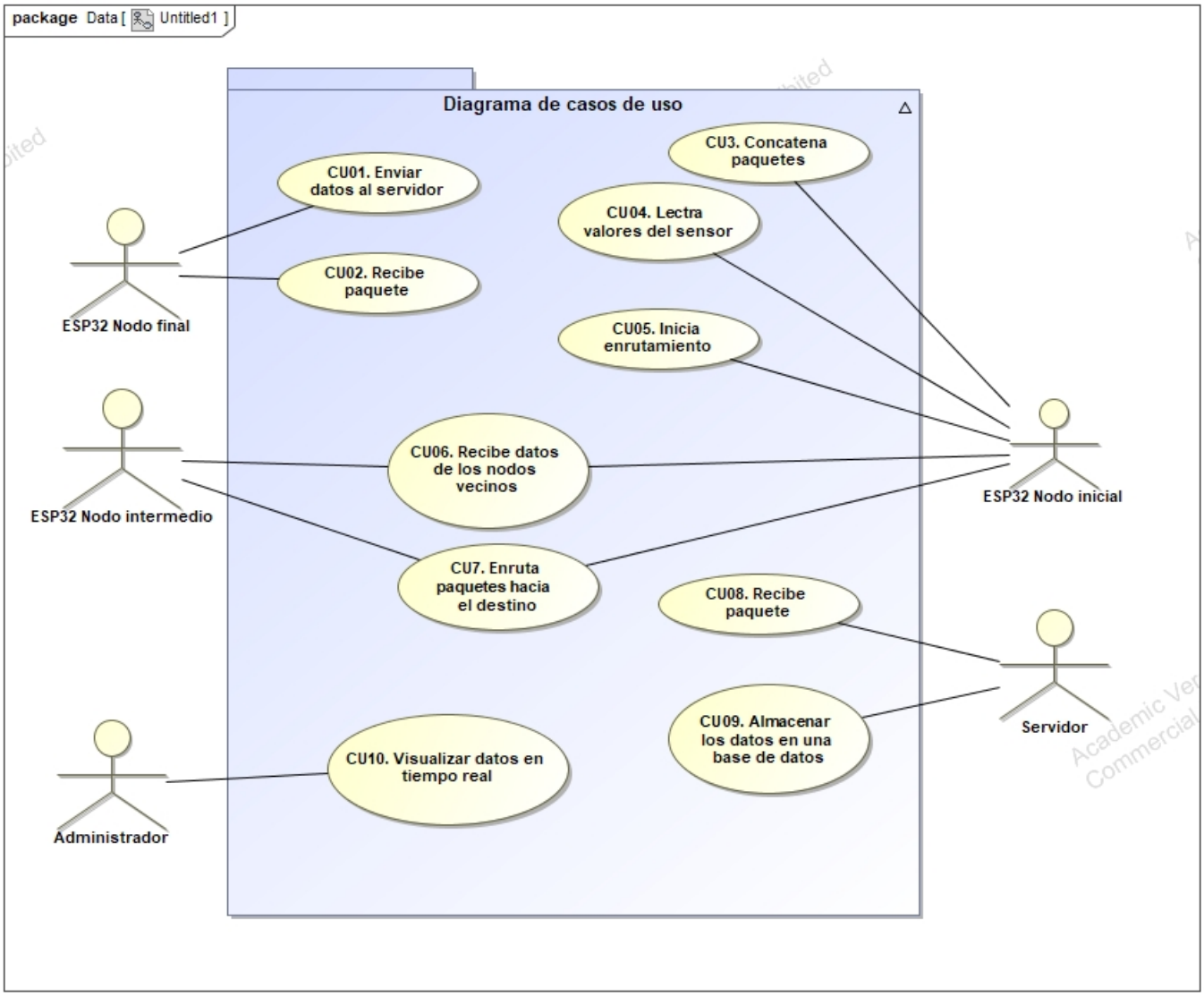


Figura 3.8: Casos de uso del sistema.

A continuación se detallan cada uno de ellos en las tablas que se muestran a continuación desde la 3.2 a 3.11.

CU01. Enviar datos al servidor	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente:Mas Ingenieros	Fecha creación: 19/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El sistema debe enviar los datos leídos de los sensores al servidor. Como empieza: Recibe los datos de los nodos vecinos.	
Pasos: 1. El microcontrolador recibe los datos de los nodos vecinos. 2. El microcontrolador envía los datos al servidor.	
Comentarios: La ESP32 espera a recibir valores de los nodos vecinos.	

Tabla 3.2: Caso de uso 01.

CU02. Recibe paquete	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente:	Fecha creación: 19/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021
Descripción: Los nodos pueden recibir datos de los nodos vecinos para reenviarlos. Como empieza: El nodo está a la espera de recibir datos del vecino.	
Pasos: 1. El microcontrolador espera recibir datos del vecino. 2. El microcontrolador envía al servidor todo el paquete que recibe.	
Comentarios: Se envía todo el paquete ya que no se trata en el microcontrolador la información.	

Tabla 3.3: Caso de uso 02.

CU03. Concatena paquetes	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Mas Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El sistema debe replicar los datos por la red para llegar al nodo final. Como empieza: Recibe datos de un vecino sin ser este el destino	
Pasos: 1. El nodo recibe datos del vecino. 2. El nodo comprueba si es el destino. 3. Si el nodo no es el destino final, lee los datos de su sensor. 4. El nodo concatena el paquete que ha recibido donde añade los datos que ha leído. 5. El nodo envía al nodo vecino para enrutar.	
Comentarios: Los datos leídos dependerá del sensor que se haya incorporado.	

Tabla 3.4: Caso de uso 03.

CU04. Lectura valores del sensor	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Mas Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El microcontrolador lee los valores del sensor. Como empieza: El nodo de inicio lee los valores del sensor.	
Pasos: 1. El microcontrolador lee los datos del sensor. 2. El microcontrolador crea el paquete para enviar.	
Comentarios: El tipo de dato que lee dependerá del tipo de sensor	

Tabla 3.5: Caso de uso 04.

CU05. Iniciar enrutamiento	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Mas Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El microcontrolador lee los valores del sensor. Como empieza: El nodo envía su paquete al nodo vecino.	
Pasos: 1. El microcontrolador crea el paquete con los datos leídos. 2. El microcontrolador envía su paquete al nodo vecino para llegar al destino.	
Comentarios: El paquete se monta	

Tabla 3.6: Caso de uso 05.

CU6. Recibe dato de los nodos vecinos.	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Más Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El nodo espera la recepción de paquetes. Como empieza: Espera la recepción de paquete.	
Pasos: 1. Espera la recepción de paquete. 2. Recibe el paquete. 3. Procesa el paquete.	
Comentarios: Se hace una espera activa.	

Tabla 3.7: Caso de uso 06.

CU7. Enruta paquetes hacia el destino.	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Más Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: Los nodos intermedios se utilizan de salto para llegar al destino final. Como empieza: El nodo recibe el paquete y enruta al destino.	
Pasos: 1. Recibe el paquete. 2. Procesa el paquete. 3. Extrae la información del paquete para ver donde está el destino. 4. Envía el paquete.	
Comentarios: Los paquetes tienen el formato especificado.	

Tabla 3.8: Caso de uso 07.

CU8. Recibe paquete.	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Más Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: Recibe paquete en el servidor para almacenar la información. Como empieza: Espera recibir un paquete.	
Pasos: 1. Se recibe el paquete en el server. 2. Se procesa el paquete. 3. Se alamcena la información.	
Comentarios: Los paquetes tienen el formato especificado.	

Tabla 3.9: Caso de uso 08.

CU9. Almacenar datos en una base de datos.	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Mas Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El sistema debe almacenar los datos en una base de datos. Como empieza: Recibe datos de un nodo.	
Pasos: 1. Recibe el bufer de datos. 2. Trata el bufer de datos según el formato. 3. Alamcena los datos en la base de datos.	
Comentarios: Debe tratar los datos antes de almacenarlos en la base de datos.	

Tabla 3.10: Caso de uso 09.

CU10. Visualizar datos en tiempo real	
Autor: Jose Carlos Torró Belda Revisor: Jose Salvador Fuente: Mas Ingenieros	Fecha creación: 18/05/2021 Fecha de revisión: 19/05/2021 Fecha de aprobación: 19/05/2021 Versión: 0.1
Descripción: El sistema muestra los datos en tiempo real en un host. Como empieza: Broker almacena datos.	
Pasos: 1. Lee los datos de la base de datos. 2. Muestra las gráficas.	
Comentarios: Los datos se muestran con la herramienta Chronograf.	

Tabla 3.11: Caso de uso 10.

3.2. Diseño de la arquitectura del sistema

Es esta sección se diseña la arquitectura del sistema con la que listamos todos los componentes y cómo interactúan entre ellos.

3.2.1. Justificación tecnología utilizada

Las redes de comunicación industrial existen tres niveles de diseño según la necesidad de volumen de datos, transmisión de datos o la seguridad que requiere el sistema. Por tanto los niveles son los siguientes:

1. **Nivel de dispositivo**: dispositivos que proporcionan la información y elementos técnicos de proceso como los PLCs.
2. **Nivel de control**: Está formado por los controladores industriales como PLCs, las unidades de control distribuidas y sistemas informáticos.
3. **Nivel de información**: Es el nivel del sistema de automatización industrial que recoge la información del nivel de control, es por ello que en este nivel es donde se tratan grandes volúmenes de datos.

En este proyecto que abarcamos nos encontramos en el nivel de información, ya que los datos que usaremos no son críticos, sino que su objetivo es usarlos para la realización de estudios, es decir, ningún otro sistema crítico[4] depende de este.

Por otro lado el contexto donde se va a aplicar el sistema es en entornos industriales como puede ser una planta de una refinería o una planta de fabricación de cerámica. En estos contextos tenemos una gran cantidad de metales, paredes, maquinaria y cableado, que pueden llegar a generar grandes cantidades de ruido o interferencias que hacen que la red en malla sea inestable. Así mismo, este sistema también se va a aplicar a la "agricultura inteligente", por ello debe ser capaz de llegar a largas distancias en espacios totalmente abiertos.

En resumen se debe crear una red IoT, por ello los criterios para crear estas redes son:

- Bajas velocidades de datos.
- Baja frecuencia de transmisión.
- Conexiones bidireccionales.
- Bajo consumo de energía.
- Largo alcance de comunicación.

[4]Por sistema crítico nos referimos por ejemplo a sistemas de alarmas ó incendios. Es decir, sistemas de los cuales la seguridad de una planta dependan.

La tabla 3.12 recoge la comparativa entre las redes estudiadas. La tabla muestra que tanto desde el punto de vista del alcance, las interferencias y el consumo la mejor opción es LoRa para el entorno donde se quiere aplicar.

Tecnología	Frec.	Alcance (m)	Interferencias	Consumo	Tasa de datos	Disponibilidad
WiFi	2.4 GHz	45	Alta	Alto	Alta	Alto
Bluetooth LE 5.0	2.4 GHz	200	Alta	Bajo	Bajo	Bajo
LoRa	868 MHz	20K	Baja	Bajo	Bajo	Bajo

Tabla 3.12: Comparaciones tecnologías estudiadas.

3.2.2. Arquitectura del sistema

El sistema que se desarrolla en este proyecto es una red en malla compuesta por un conjunto de nodos que conectan con un servidor. A través de esta red se transferirá información que llegará al servidor y la almacenará en una base de datos contenida en él.

Tal como se observa en la figura 3.9, los nodos se comunican entre ellos mediante una red tipo LoRa a nivel físico formando una malla. Los nodos son microcontroladores del tipo ESP32 LoRa TTGO. Algunos de estos nodos se encargan de leer los datos de los sensores, en este caso capaces de medir temperatura y humedad DHT11 que son los parámetros que se quieren controlar. La información capturada será enviada a través de los nodos de la red hasta un nodo final (destino) que está conectado con el servidor. Este último dispositivo está implementado sobre una Rasberry que contiene la base de datos de tipo InfluxDB donde se almacenará la información recogida. El nodo final de la red será el encargado de transmitir al servidor los datos que le llegan, mediante el protocolo MQTT (apartado 3.1.2).

Por tanto, el sistema a desarrollar se compone de:

- Diferentes sensores de captación de distintos parámetros como temperatura, humedad o nivel de gases.
- Nodos formados por microcontroladores ESP32 que recogen la información de los sensores.
- Nodos formados por microcontroladores ESP32 que envían los datos a través de la red para que lleguen al nodo destino.
- Nodo destino que será el que se comunique con el servidor.
- Servidor que almacena y trata los datos.
- El sistema de comunicación está estructurado en una topología en malla.

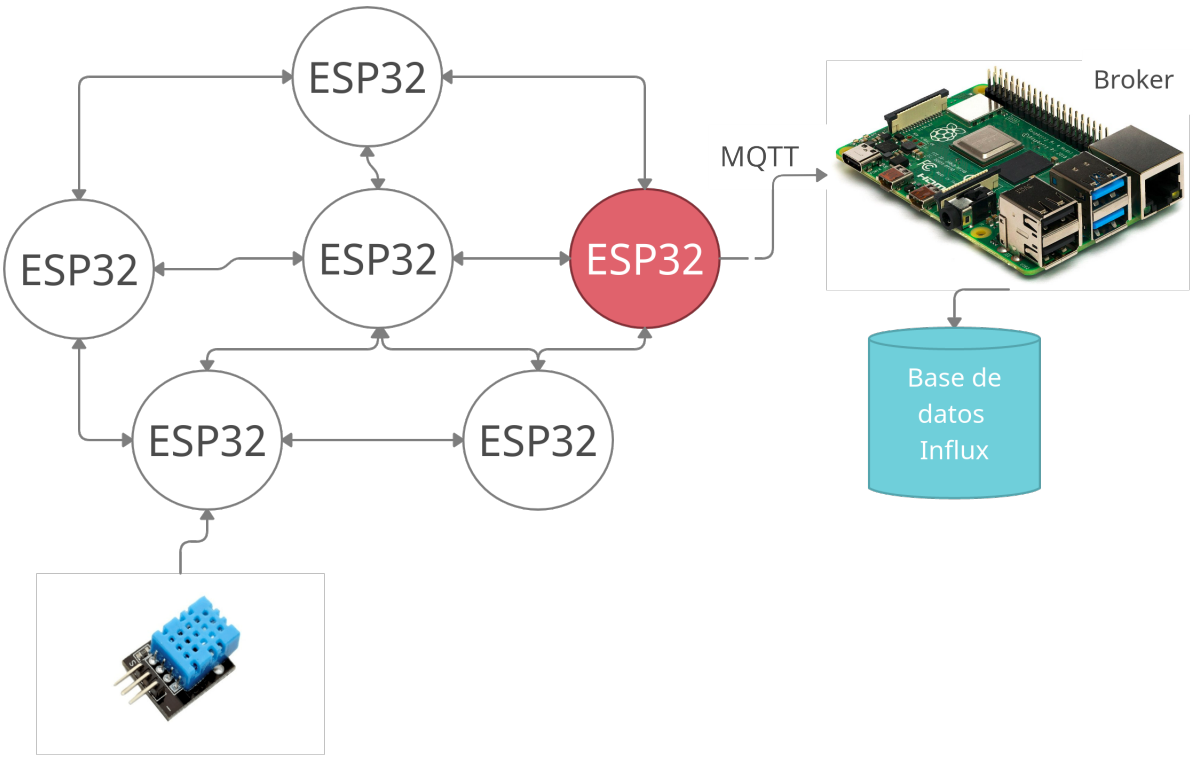


Figura 3.9: Diagrama de la arquitectura.

Capítulo 4

Implementación

Contents

En el capítulo de implementación y pruebas especificamos como se ha desarrollado el sistema y el testeo del mismo.

El desarrollo del proyecto se aborda desde tres contextos distintos: la red en malla para la captación y transmisión de los datos, la conexión con el servidor y el propio servidor que alberga la base de datos donde se almacenará la información. Y esto se tiene que tener en consideración a la hora de la implementación, dividiéndose esta en dos partes según los dos principales componentes del sistema:

- Programación de los microcontroladores para que sean capaces de leer la información desde los sensores y enviarla a través de los nodos de la red hasta uno final. Esto implica implementar el software necesario para la captura de datos de los sensores y el algoritmo de encaminamiento con LoRa como red de interconexión.
- Programación del servidor para que sea capaz de reunir los datos desde un nodo de la red utilizando el protocolo MQTT. Este tiene que ser capaz de almacenar los datos dentro de la base de datos.

4.1. Programación del microcontrolador

4.1.1. Lectura sensores

En esta fase de desarrollo tan solo hemos usado el sensor de temperatura y humedad DHT explicado en el apartado 3.1.6. Para la lectura de los valores hacemos lo siguiente:

1. Definición de los pines de lectura. En este caso, utilizamos el pin 13 como entrada digital para la lectura de los datos. También definimos el tipo de sensor, en este caso se trata de DHT11.

```
#define DHTPIN 13
#define DHTTYPE DHT11
```

2. Inicialización de la biblioteca DHH11. En este proyecto utilizamos la propia biblioteca que Arduino proporciona para la lectura de los valores del sensor DHT11.

```
DHT dht(DHTPIN, DHTTYPE);
```

3. Lectura de los valores. En este caso, a partir de la inicialización de la biblioteca, se obtienen los métodos de lectura de temperatura y humedad. Estos pueden acarrear errores de lectura por lo que se trata.

```
humidity = dht.readHumidity();
temperature = dht.readTemperature();

if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
}
```

Con los valores leídos generamos el paquete correspondiente para el envío al nodo final tal como se indica en el formato que especificamos en el código 4.1.

4.1.2. Especificación del formato de los datos

Para la implementación del protocolo en malla primero debemos especificar el formato de los mensajes, también se debe especificar y configurar la biblioteca RadioHead para utilizarse en el proyecto. El fichero de ejemplo del búfer se muestra en el código 4.1. En este formato primero especificamos cuantos sensores hay en total, luego tenemos la información de cada nodo en la malla.

El formato del buffer es de tipo JSON(JavaScript Object Notation). Este formato lo utilizamos debido a que los ficheros son sencillos de tratar, es decir, es sencillo extraer los datos de ellos por lo que la programación se facilita en Node-RED. Este tipo de formato es un subconjunto de notación literal en el lenguaje JavaScript, sin embargo, ya se considera que está totalmente separado del lenguaje.

En la información de cada nodo en la malla, se especifica el ID de cada uno, el destino[1] y los valores leídos de los sensores, en este caso el primer nodo ha leído Temperatura (Temp) y Humedad (Hum); el siguiente nodo ha leído los valores de gases.

```
{
    "sensores":2,
    "red": [
      {
        "id": 4,
        "data": {
          "destino": 1,
          "valores": {
            "Temp": 25,
            "Hum": 65
          }
        }
      },
      {
        "id": 3,
        "data": {
          "destino": 1,
          "valores": {
            "gas": 256
          }
        }
      }
    ]
}
```

Código 4.1: Formato buffer.

4.1.3. Configuración RadioHead

La biblioteca RadioHead está orientada a enviar y recibir mensajes empaquetados a través de diferentes tecnologías en nuestro caso utilizamos LoRa a nivel físico.

En primer lugar, debemos incluir las siguientes bibliotecas:

1. "RHRouter.h"[9]: Con RHRouter, las rutas están cableadas. Esto significa que cada nodo debe tener programado en él cómo llegar a cada uno de los otros nodos con los que intentará comunicarse. Esto significa que debe especificar la dirección del nodo del siguiente salto para cada uno de los nodos de destino.

2. "RHMesh.h"[9]: Utilizado para el descubrimiento de vecinos de forma automática.

[1]El destino es el nodo final a quién va dirigido.

3. "RH_RF95.h"[10]: Controlador para enviar y recibir datagramas no confiables y no direccionados a través de LoRa.

A continuación debemos especificar la frecuencia con la que trabajaremos con LoRa, teniendo en cuenta la documentación, LoRa opera con la frecuencia 868.0 en Europa.

```
#define RF95_FREQ 868.0
```

En la parte de "setup" del código debemos hacer diferentes acciones para la configuración de la biblioteca.

1. Iniciamos el controlador; en caso de que obtengamos error paramos la ejecución en este punto, ya que no funcionará el protocolo.

```
while(!rf95.init()) { Serial.println("[RFM95] init failed");}
```

2. Cambiamos la frecuencia a la especificada en cada territorio para LoRa, tal como se ha visto en la documentación. En caso de que esto falle, paramos la ejecución porque no podremos seguir operando.

```
if (!rf95.setFrequency(RF95_FREQ)) {
 Serial.println("setFrequency failed");
 while (1);
}
```

3. Instanciamos la clase RMesh para controlar la entrega y recepción de mensajes utilizando el controlador. Para instanciarla debemos especificarle el controlador y el ID de nodo que tenemos. Luego, inicializamos el controlador.

```
manager = new RHMesh(rf95, nodeId);
if (!manager->init()) {
    Serial.println(F("init failed"));
} else {
    Serial.println("done");
}
```

4. Por último configuramos el controlador variando la potencia de la antena así como la configuración del modem de la siguiente forma; aquí especificamos la configuración de los registros donde se indica Bw125Cr48Sf4096 que significa lo siguiente:

   - **Bw = 125 kHz**: Se utiliza un ancho de banda (Bw) de 125 kHz.
   - **Cr = 4/8**: El "code rate"[23] es de 4/8, por lo que estamos transmitiendo el doble de bits que los que contienen información.

- **Sf = 4096 chips/symbol**: Se utiliza un "spreding factor" de 4096 chips por símbolo. Este valor indica el número de bits sin procesar que pueden codificarse con ese símbolo.
- **low data rate**
- **CRC on**: Cyclic Redundancy Check[24] el cual es un código de detección de errores que se utiliza para detectar cambios en accidentales en los datos sin procesar.
- **Slow+long range**: Se indica que se utiliza el máximo rango de distancias.

```
rf95.setTxPower(20, false);

boolean longRange = true;
if (longRange) {
    RH_RF95::ModemConfig modem_config = {
      0x78, // Reg 0x1D: BW=125kHz, Coding=4/8, Header=explicit
      0xC4, // Reg 0x1E: Spread=4096chips/symbol, CRC=enable
      0x08  // Reg 0x26: LowDataRate=On, Agc=Off.  0x0C is LowDataRate=ON,
                ACG=ON
    };
    rf95.setModemRegisters(&modem_config);
    if (!rf95.setModemConfig(RH_RF95::Bw125Cr48Sf4096)) {
      Serial.println(F("set config failed"));
    }
}
```

4.1.4. Implementación de la red en malla

La red en malla de dos dimensiones es una topología en la que los nodos se comunican entre sí directamente (si están dentro del alcance) o indirectamente a través de nodos intermedios.

En primer lugar para la especificación de la red en malla, obviamos como interactúa cada nodo con el sistema y hacemos que todos funcionen como nodos intermedios para verificar que existe conexión entre todos, además de comprobar que el protocolo se está cumpliendo. En esta verificación se imprime el RSSI e ID del nodo con el que tienen comunicación.

Una vez establecido el protocolo en malla se inicia la fase de implementación de información captada por los sensores a través de la malla.

El desarrollo de la red en malla en los microcontroladores se realiza en tres escenarios donde se hacen las siguientes acciones:

- **Nodo final**: Espera paquetes de los nodos vecinos. Indicamos un tiempo para que el nodo no quede bloqueado si ha perdido la conexión con la malla, luego, a partir del "manager" que hemos inicializado con RadioHead utilizamos los métodos que nos proporciona la librería para recibir mensajes. Para la recepción utilizamos el método `recvfromAckTimeout` donde indicamos donde se escribirá el búfer que se reciba (`(uint8_t *)buf`), el tamaño

del paquete (`len = sizeof(buf)`), el tiempo de espera en la recepción y por último de quién lo recibe (`from`). Este nodo también se encarga de enviar la información al servidor mediante el protocolo MQTT, esto se ve más adelante en el apartado 4.2.

```
unsigned long nextTransmit = millis() + random(3000, 5000);
while (nextTransmit > millis()) {
  int waitTime = nextTransmit - millis();
  uint8_t len = sizeof(buf);
  uint8_t from;
  if (manager->recvfromAckTimeout((uint8_t *)buf, &len, waitTime, &from)) {
    buf[len] = '\0'; // Indica la terminación del búffer.
    sendMQTT(buf);
  }
```

- **Nodo de lectura de datos**: Este nodo puede actuar de dos formas. Primero se encarga de leer los datos desde el sensor como se ha descrito en el apartado 4.1.1. También, este mismo nodo puede actuar como nodo intermedio, por tanto realiza las siguientes acciones:

  1. Lectura valores de los sensores.
  2. Crear paquete para enviar.
  3. Verifica si se recibe un paquete de los nodos vecinos.
  4. Si se recibe un paquete del vecino, se concatena el paquete del nodo que se acaba de crear y se envía hacia el nodo destino.

- **Nodo intermedio**: Espera la recepción de los datos como en el punto 1; luego los replica al siguiente nodo. Esta parte se divide en dos partes:

  1. El nodo envía el paquete al destino final.
  2. En caso de que no haya conexión, se realiza un Broadcast a todos los nodos vecinos y se busca la mejor conexión teniendo en cuenta el parámetro de RSSI que indica la potencia que se recibe en el vecino.
  3. Una vez encontrado, se establece conexión y envía el paquete al nodo vecino.

  En el algoritmo que hemos definido en redes como WiFi o Bluetooth se podría llegar a tener ciclos. Sin embargo, LoRa tiene un gran alcance de distancia por lo que es poco probable que se lleguen a producir ciclos.

4.2. Conexión red con el servidor

El siguiente paso de la implementación es la conexión de la red con el servidor que contiene la base datos. Esto se realiza utilizando el protocolo MQTT. En este protocolo la conexión TCP/IP se mantiene abierta y se reutiliza en cada comunicación.

Con el protocolo MQTT debemos "subscribirnos" a un "topic" para poder enviar datos y que queden almacenados. Por ello, el primer paso es configurar la conexión al broker. Para esta implementación haremos uso de las bibliotecas "WiFi.h"[17] y "PubSubClient.h"[14]. Por tanto, en primer lugar iniciamos las constantes que usaremos para poder realizar la conexión, como se muestra en el código 4.2. En este código indicamos la IP del servidor MQTT y los "topic" a los que nos subscribimos. También indicamos el usuario y contraseña que usamos en el protocolo para poder enviar datos y el "topic" principal. Por último identificamos que nodos está enviando la información con un ID.

```
const char* mqtt_server = "192.168.0.210";    // IP of the MQTT broker
const char* humidity_topic = "ADBEEFFEEE/Hum/";
const char* temperature_topic = "ADBEEFFEEE/Temp/";
const char* mqtt_username = "pi"; // MQTT username
const char* mqtt_password = "pi"; // MQTT password
const char* clientID = "ADBEEFFEEE"; // MQTT client ID
const char* mainTopic = "topic";

// Initialise the WiFi and MQTT Client objects
WiFiClient wifiClient;
// 1883 is the listener port for the Broker
PubSubClient client(mqtt_server, 1883, wifiClient);
```

Código 4.2: Configuración variables MQTT.

Una vez indicadas las variables que vamos a usar e inicializadas las bibliotecas que usaremos para la conexión MQTT realizamos los siguientes pasos:

1. Hacemos un broadcast a todas los nodos que podemos alcanzar buscando la que tiene el broker. Código 4.3.

```
String getSSID(){
    int n = WiFi.scanNetworks();

    if (n == 0) {
        Serial.println("no networks found");
    } else {
        Serial.print(n);
        Serial.println(" networks found");
        for (int i = 0; i < n; ++i) {
            if(WiFi.SSID(i).equals("tplink"))
            {
              Serial.println("Descubierta red");
              return WiFi.SSID(i);
            }
        }
    }
}
```

Código 4.3: Escaneo de redes.

2. Una vez encontrada la red, nos conectamos a esta mediante la ejecución del código 4.4, aquí especificamos el nombre de la red y la contraseña.

```
    WiFi.begin(ssid, wifi_password);
```

Código 4.4: Conexión a la red.

3. Una vez el nodo lo tenemos conectado a la red, entonces es cuando iniciamos la conexión al servidor mediante la ejecución del código 4.5 donde especificamos el ID del dispositivo que envía, luego el usuario y contraseña del servidor MQTT.

```
client.connect(clientID, mqtt_username, mqtt_password)
```

Código 4.5: Conexión al servidor.

Una vez realizados estos pasos ya podemos empezar a enviar el búfer al servidor como se muestra en el código 4.6. Aquí, en primer lugar, publicamos el búfer en el topic principal. En caso de que tengamos error, nos volvemos a conectar al broker como hemos hecho anteriormente, luego enviamos el búffer de nuevo. En caso de que esto falle nos devuelve error y perdemos el mensaje.

```
void sendMQTT(char* buf){
    if (client.publish(mainTopic,buf))
    {
      Serial.println("Buffer enviado");
    } else {
      client.connect(clientID, mqtt_username, mqtt_password);
      delay(10);
      if(!client.publish(mainTopic,buf))
      {
          Serial.println("Buffer no enviado despues de reconectar.");
      }else{
        Serial.println("Buffer enviado");
      }
    }
}
```

Código 4.6: Envío del bufer al broker.

4.3. Implementación servidor

Para realizar este paso utilizamos Node-RED que es una herramienta de programación visual. Es decir, Node-RED facilita la conexión de elementos hardware con el software. También, se utiliza para realizar aplicaciones de visualizaciones de datos en tiempo real.

En el proyecto que realizamos el flujo dentro de Node-RED es el siguiente, de forma gráfica lo podemos observar en la figura 4.1:

1. Un bloque MQTT espera la entrada de datos al topic indicado.
2. Convertimos el búfer que entra en el sistema a formato JSON.
3. Tratamos el fichero JSON para poder almacenar los datos en la BBDD Influx tal como indica la documentación.
4. Volvemos a convertir la salida del formateo de datos a JSON.
5. Escribimos en la BBDD.

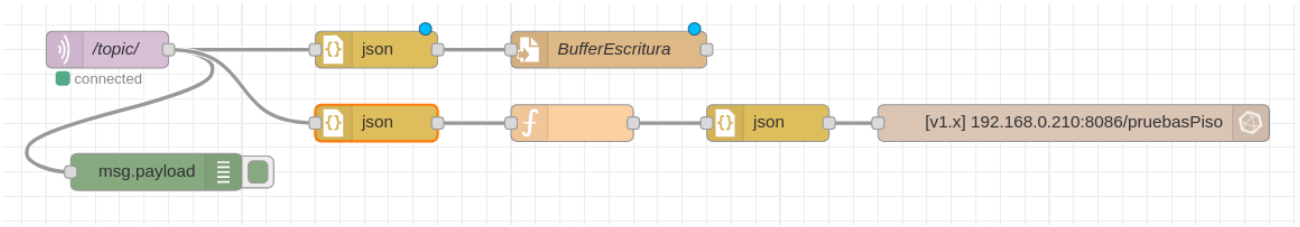


Figura 4.1: Flujo de ejecución de los nodos en Node-RED.

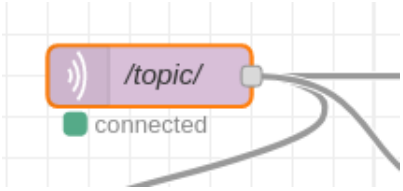


Figura 4.2: Nodo MQTT.

4.3.1. Configuración receptor MQTT

En esta parte debemos configurar el bloque MQTT (figura 4.2) para que acepte peticiones y reciba datos, para ello se indica la IP del servidor y el "topic" principal. En la figura 4.3 podemos observar la configuración del servidor MQTT, donde se indica:

- **IP Servidor**: 192.168.0.210.
- **Puerto**: 1883
- **Topic**: Topic; Es el topic principal donde es nodo ESP32 se subscribe para publicar los datos.
- **Quality of Service (QoS)**: 0; esta calidad del servicio indica que la conexión es estable.
- **Output**: Es la salida del nodo, en este caso se extrae como String o Buffer.
- **Name**: Nombre del nodo.

4.3.2. Convertidor JSON

Antes de tratar el buffer con Node-RED debemos convertirlo a formato JSON, ya que la que salida del nodo MQTT es un String, esta conversión se hace en el nodo mostrado 4.4. Como se muestra en la figura 4.5 la acción que se hace es convertir de String a Json.

4.3.3. Extracción y tratado de datos

Para almacenar los datos en la base de datos de InfluxDB, primero debemos extraer los valores y tratarlos para dejarlos en el formato correspondiente tal como podemos ver en el

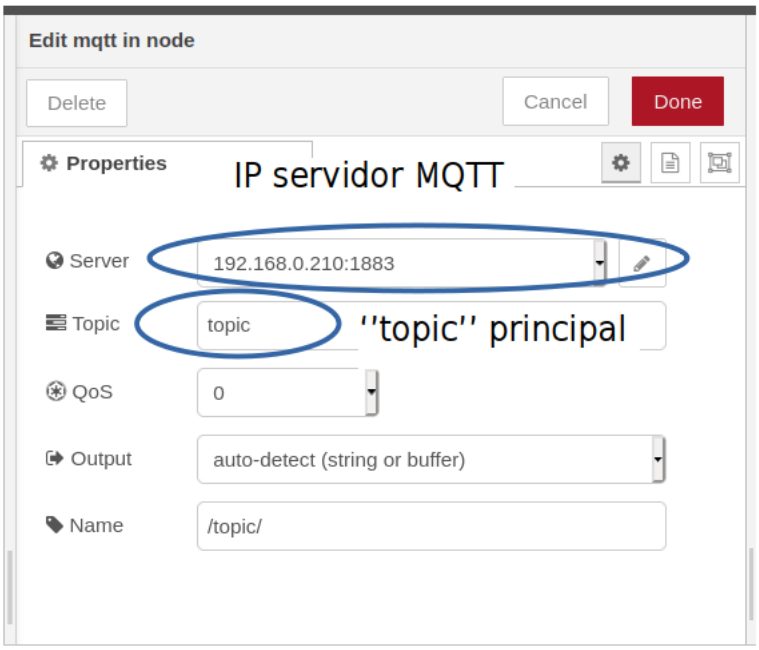


Figura 4.3: Especificaciones del servidor MQTT en Node-RED.

Figura 4.4: Nodo json.

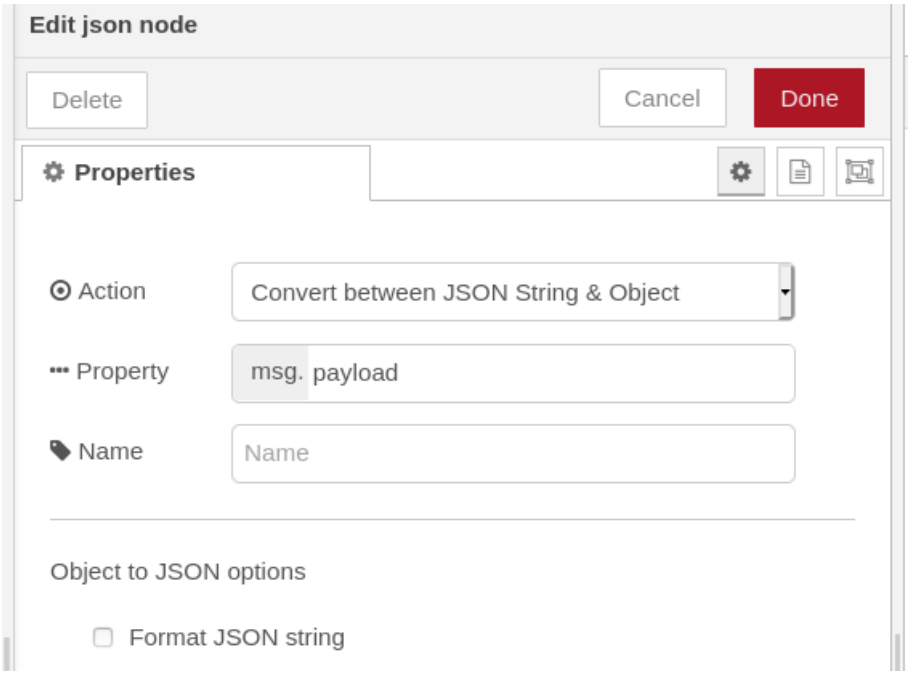


Figura 4.5: Configuración del conversor a formato JSON.

código 4.7. Para ello debemos tener en cuenta el formato del fichero visto en el apartado 4.1.2. La especificación del formato JSON debe contener lo siguiente:

1. **“tags”**: Sirve de etiqueta para saber el nodo que ha captado los datos, es por ello que especificamos el ID del microcontrolador.
2. **“fields”**: Contiene la información que se almacena.
3. **“timestamp”**: Indica en que momento se almacena la información.

```
let val, res, x, tam;
let meas, valor, id
let mensaje = "";
x = msg.payload;

mensaje += "[";
msg.payload.red.forEach(elem => {

val = Object.keys(elem.data.valores);
    for (var prop in val) {
        valor = elem.data.valores[val[prop]];
        id = elem.id;
        meas = val[prop];
        mensaje +=
    `
    {
    "measurement":  "${meas}",
    "fields": {
        "value": ${valor}
    },
    "tags":{
        "device": ${id}
    },
    "timestamp": ${new Date().getTime()}
    },`;
    }
});
tam = mensaje.length;
mensaje = mensaje.substring(0,tam-1) + "\n]";

msg.payload = mensaje;

return msg;
```

Código 4.7: Tratado de los datos en formato JSON y preparado para almacenar en la base de datos.

Con el código 4.7 lo que se realiza es la extracción de los datos del paquete que se recibe y se crea un nuevo fichero en formato JSON en el formato especificado por InfluxDB,

4.3.4. Almacenamiento en la base de datos

El almacenamiento en la base de datos la realizamos utilizando un nuevo nodo de Node-RED con el que indicamos la IP del servidor que actúa como base de datos, el puerto y la base de

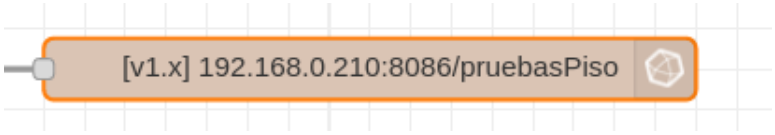


Figura 4.6: Nodo de Node-RED de conexión con la base de datos.

datos donde almacenamos los datos, este nodo es el referente en la figura 4.6.

A continuación, en la configuración de este nodo especificamos la IP del servidor, en este caso 192.168.0.210; el puerto (8086). Por último se indica la base de datos en la que almacena los valores. El servidor al que nos conectamos es `192.168.0.210:8086/pruebasPiso`, que está en local en la misma Raspberry. Estos valores se indican en la figura 4.7.

4.3.5. Visualización de la base de datos

Para la visualización de los datos utilizamos Chronograf donde podemos ver los datos en tiempo real del sistema. Con esta tecnología se crean gráficas a partir de los valores que se almacenan. Por esta razón es importante que los datos en el fichero JSON cumplan los requisitos que se especifican en el formato de InfluxDB.

La vista general de la interfaz de Chronograf es la mostrada en la figura 4.8, en ella podemos controlar que valores podemos visualizar de cada dispositivo. Por ejemplo para la visualización de la Humedad ("Hum") de dos nodos se marcan ambos tal como aparece en la figura 4.9. En la figura 4.10 se muestra la gráfica que se crea. En este caso, son los valores tomados entre el 18 y 32 de marzo. Con estos valores se puedo verificar el funcionamiento del protocolo MQTT y que los datos se almacenaran correctamente en la base de datos.

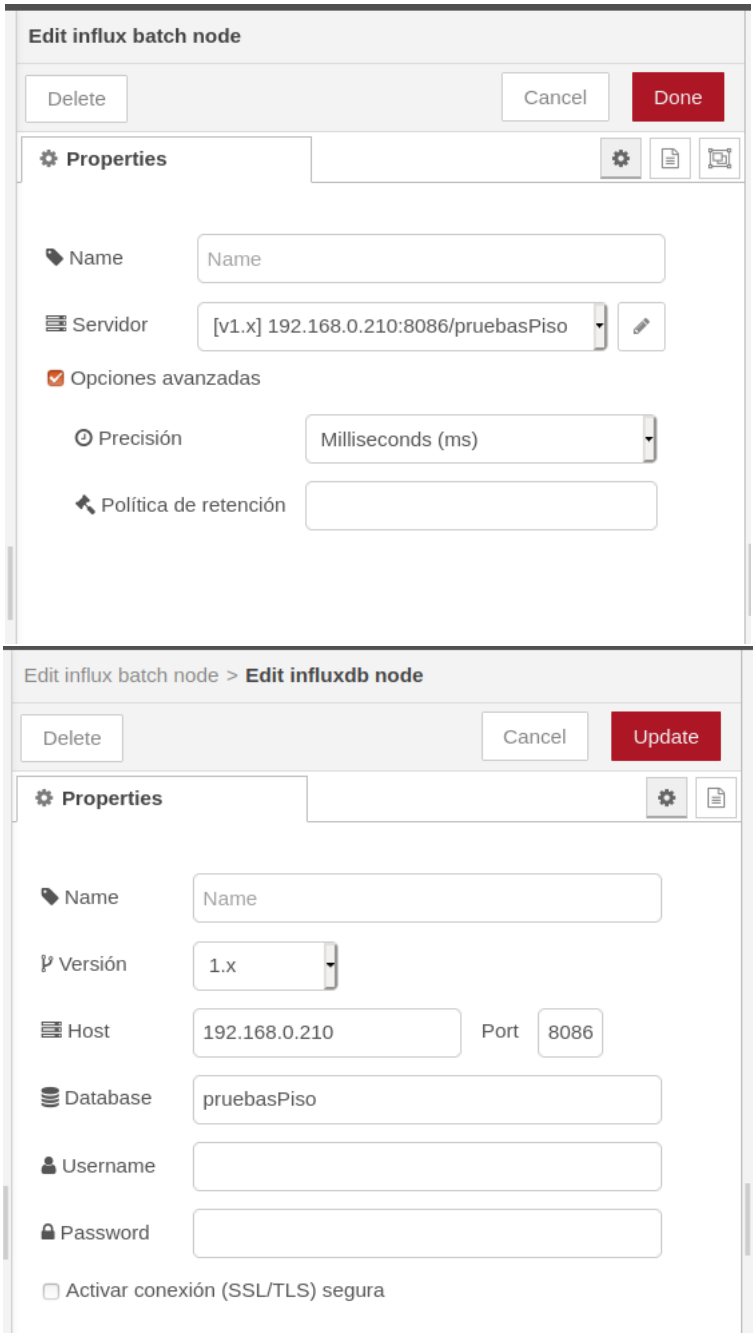


Figura 4.7: Configuración conexión con la base de datos

.

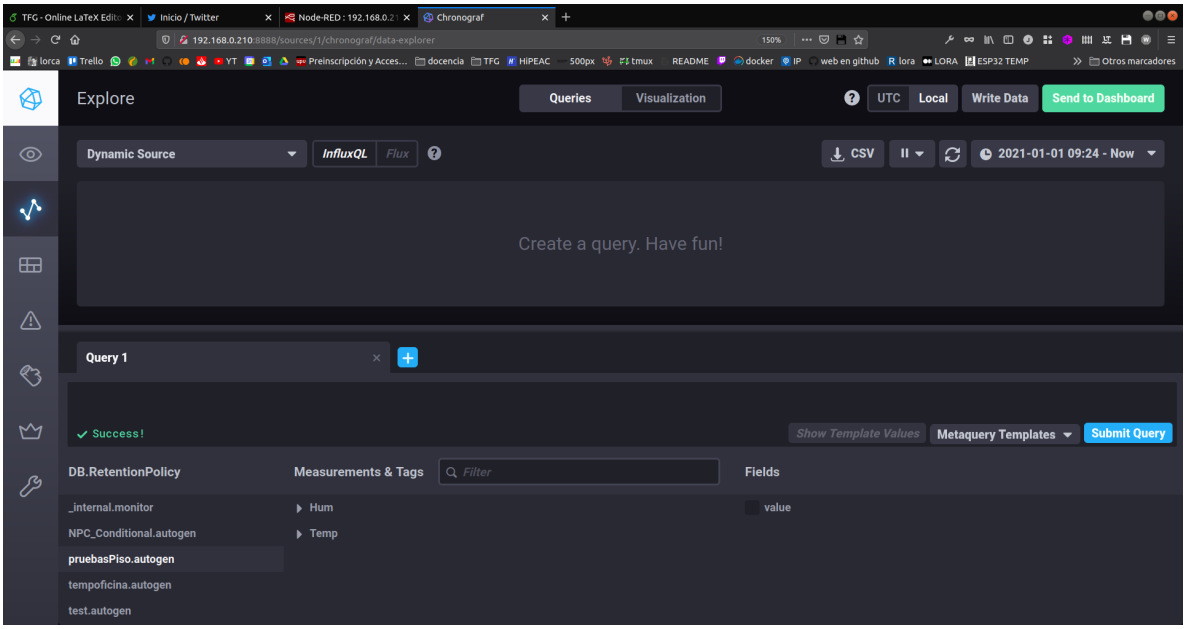


Figura 4.8: Vista general interfaz Chronograf.

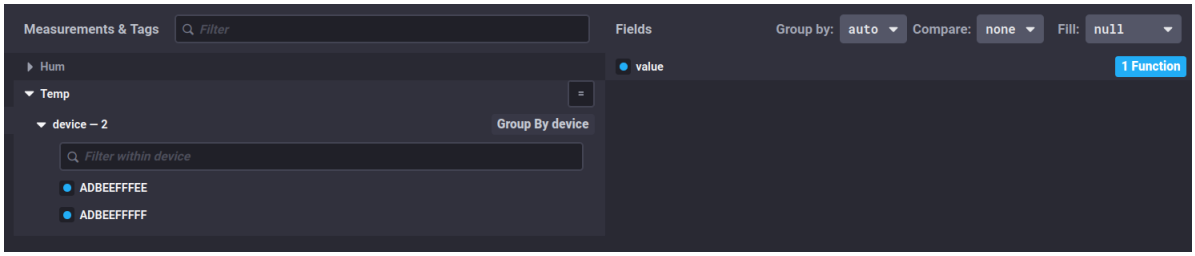


Figura 4.9: Elección de visualización de datos en Chronograf.

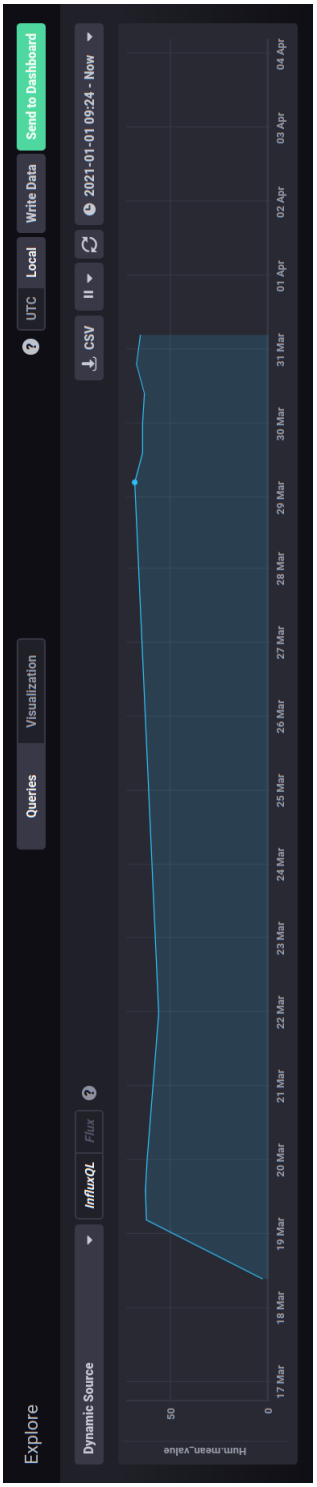


Figura 4.10: Gráfica de valores de humedad y temperatura.

Capítulo 5

Conclusiones

Contents

Este proyecto enmarcado dentro del departamento de Investigación y Desarrollo de Más Ingenieros tiene el objetivo de ofrecer a los clientes nuevas soluciones inteligentes a los problemas actuales donde todavía no hay digitalización, como pueda ser la captación de datos y su posterior tratamiento de una planta industrial hasta aplicaciones de "agricultura inteligente".

El prototipo desarrollado es capaz de capturar información de una planta industrial y a través de una red tipo LoRa se envía a un servidor donde almacena esa información.

Mas Ingenieros con este prototipo pretende terminar de desarrollarlo y venderlo a los clientes que ya tiene en plantas industriales así como explorar nuevos nichos de mercado como pudiese ser la agricultura inteligente cada vez más en auge.

5.1. Ámbito formativo

En el ámbito formativo he podido aplicar las técnicas utilizadas a lo largo de mi paso por el grado de ingeniería informática, así como gracias a la experiencia recibida dentro del grupo de investigación de HPC&A.

En general, el desarrollo del proyecto he aprendido a buscar fuentes de información; a realizar una investigación previa y conocer nuevas tecnologías y técnicas de programación que desconocía totalmente. También he podido saber lo que es teletrabajar y aprender a ser autónomo en el aprendizaje, ya que no podía preguntar dudas con tanta facilidad como en la empresa.

El proyecto me ha permitido poder programar para ESP32, algo que ya venía haciendo de

forma autónoma en casa, por lo que he podido asimilar conceptos de LoRa a nivel físico así como aplicar conocimientos dados durante el grado. Por una parte, he podido aplicar conocimientos de programación y entender cómo aplicar algoritmos vistos a otras tecnologías, como pueda ser el algoritmo en malla.

5.2. Ámbito profesional

La experiencia en la empresa ha sido buena, gracias a las reuniones semanales hemos podido avanzar en el proyecto. Además me han enseñado tecnologías que desconocía, también a como funciona un proyecto de I+D dentro del ámbito empresarial. Por otra parte, en cuanto al proyecto la empresa me ha facilitado el material correspondiente para poder realizar las prácticas online.

5.3. Ámbito personal

Personalmente, en un futuro si puedo elegir un trabajo no online será mi preferencia. El teletrabajo está bien cuando tienes un espacio y lugar adecuados para ello, sin embargo, en cuanto a prácticas he tenido que costearme yo mismo los gastos provocados por las prácticas, además de no tener un espacio fijo para teletrabajar. Por esta razón, en un futuro si puedo elegir entre un tipo de trabajo online o presencial, elegiré presencial.

Por otra parte, las dudas no se resuelven con facilidad, ya que debo esperar que respondan para una duda que en persona podrían ser menos de 5 minutos.

Cabe destacar que se trata de un proyecto hardware, por lo que a estas prácticas se le suma no tener a veces material para poder desarrollarlas correctamente, como fuentes de alimentación.

5.4. Mejoras del proyecto

Como mejoras se aplica poder hacer una red en malla más escalable estudiando más a fondo la biblioteca de RadioHead, así como poder crear niveles de redes malladas para poder organizar los datos mejor.

Por otra parte, en lo que respecta a seguridad al utilizar LoRa a nivel físico no estamos encriptando los mensajes por lo que se tendría que desarrollar un protocolo de seguridad donde solo los dispositivos reconocidos en la red puedan enviar y recibir. En lo que respecta al servidor deberíamos crear un firewall que no permitiera que cualquier dispositivo no reconocido pudiese enviar datos a la base de datos.

En lo que respecta al envío de datos al broker, se debería investigar la forma de hacer que el mensaje no se pierda en caso de que se reconecte al broker tal como se observa en el código 4.6.

En conclusión, este proyecto está en una fase muy inicial por lo que se debería estudiar mucho más las tecnologías actuales y realizar muchas más pruebas en entornos reales, además se debería hacer más estudios sobre las interferencias en frecuencias en estos entornos. Por último, analizar hasta que punto es mejor usar solo LoRa a nivel físico o el protocolo LoRa WAN que ya está implementado.

Bibliografía

[1] Chronograf 1.8 documentation. [Consulta: 05 de mayo de 2021].

[2] NÓmina convenio tic: Tablas salariales 2021. [Consulta: 13 de mayo de 2021].

[3] Rhmesh class reference. `http://www.airspayce.com/mikem/arduino/RadioHead/classRHMesh.html`. [Consulta: 18 de Febrero 2021].

[4] Tecnología de radio. [Consulta: 13 de mayo de 2021].

[5] Aosong. Temperature and humidity module. [Consulta: 15 de mayo de 2021].

[6] Department of ECE and INMC, Seoul National University, Seoul, Korea; email:weiping, mhchoi@mwnl.snu.ac.kr, schoi@snu.ac.kr. Ieee 802.11ah: A long range 802.11 wlan at sub 1 ghz. `https://www.riverpublishers.com/journal_read_html_article.php?j=JICTS/1/1/5`. [Consulta: 17 de Febrero 2021].

[7] Digi-Key. Desarrollar con lora para aplicaciones iot de baja tasa y largo alcance. [Consulta: 18 de mayo de 2021].

[8] OpenJS Foundation. Node-red user guide. [Consulta: 14 de mayo de 2021].

[9] Adafruit Industries. Rhmesh class reference. [Consulta: 15 de marzo de 2021].

[10] Adafruit Industries. Rhmesh class reference. [Consulta: 15 de marzo de 2021].

[11] Luis Llamas. ¿qué es mqtt? su importancia como protocolo iot. urlhttps://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/, 2019. [Consulta: 03 de mayo de 2021].

[12] Luís Llamas. Detector de gases con arduino y la familia de sensores mq. [Consulta: 15 de mayo de 2021].

[13] Mike McCauley. Radiohead. `http://www.airspayce.com/mikem/arduino/RadioHead/`. [Consulta: 18 de Mayo 2021].

[14] Nick O'Leary. Arduino client for mqtt. [Consulta: 10 de marzo de 2021].

[15] RedHat. ¿qué es el open source? [Consulta: 14 de mayo de 2021].

[16] Semtech. What is lora? `https://www.semtech.com/lora/what-is-lora`. [Consulta: 15 de Abril de 2021].

[17] Espressif Systems. Arduino core for the esp32. [Consulta: 10 de marzo de 2021].

[18] Lorenzo Vangelista. Frequency shift chirp modulation: The lora modulation. *IEEE Signal Processing Letters*, 24(12):1818–1821, 2017.

[19] Wi-Fi HaLow, el estándar Wi-Fi para IoT. José antonio lorenzo. `https://www.redeszone.net/tutoriales/redes-wifi/wi-fi-halow-estandar-que-es-iot/`. [Consulta: 17 de Febrero 2021].

[20] Wikipedia. Bluetooth — wikipedia, la enciclopedia libre, 2021. [Internet; descargado 30-abril-2021].

[21] Wikipedia. Ieee 802.11 — wikipedia, la enciclopedia libre, 2021. [Internet; descargado 14-mayo-2021].

[22] Wikipedia. Wifi — wikipedia, la enciclopedia libre, 2021. [Consulta: 14 de mayo de 2021].

[23] Wikipedia contributors. Code rate — Wikipedia, the free encyclopedia, 2019. [Online; accessed 20-May-2021].

[24] Wikipedia contributors. Cyclic redundancy check — Wikipedia, the free encyclopedia, 2021. [Online; accessed 20-May-2021].

Índice de cuadros

Índice de figuras