UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Tranmisión de multimedia vía LoRa

Autor

Fernando Tejero Rodríguez

Directores

Jorge Navarro Ortiz y Natalia Chinchilla Romero

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, noviembre de 2022

Tranmisión de multimedia vía LoRa

Autor
Fernando Tejero Rodríguez

Directores
Jorge Navarro Ortiz y Natalia Chinchilla Romero

Transmisión de multimedia vía LoRa

Fernando Tejero Rodríguez

Palabras clave: IoT, LoRa, LoRaWAN, LBT.

Resumen

En la actualidad, nos encontramos ante un continuo crecimiento y una demanda cada vez mas grande de dispositivos IoT, motivo que provoca el surgimiento de nuevas tecnologías en el cada vez más importante sector de las telecomunicaciones.

Gran parte de estos dispositivos IoT serán interconectados a través de redes de area amplia y baja potencia (LPWAN) donde destacamos LoRa, Sigfox y NB-IoT. El presente proyecto se centrará en LoRa para la transmisión de multimedia, de la que podemos resaltar su gran alcance y cobertura, su baja potencia y, por ende, su bajo consumo energético. Esto último implicará una gran duración de la vida de sus baterías.

El proyecto tiene como objetivo la transmisión de multimedia entre motas LoRa, mejorando la velocidad de transmisión entre estas. A la hora de transmitir debemos cumplir la normativa de la ITU, la cual impone el uso del 1 % del ciclo de trabajo. No obstante, también permite utilizar el algoritmo de escucha antes de transmitir (LBT, Listen Before Talk), que será el que utilicemos para cumplir el objetivo del trabajo.

Streaming media via LoRa

Fernando Tejero Rodríguez

Keywords: IoT, LoRa, LoRaWAN, LBT.

Abstract

Currently, we are faced with continuous growth and a increasing demand for IoT devices, which is causing the emergence of new technologies in the increasingly important sector of telecommunications.

Most of these IoT devices will be interconnected through low power wide area networks (LPWAN), where the most prominent are LoRa, Sigfox and NB-IoT. This project will focus on LoRa for multimedia transmission. Where we can highlight as its main characteristics its great range and coverage, its low power and, therefore, its low energy consumption, where the latter implies a long life of its batteries.

The project aims to transmit multimedia between LoRa motes, improving the transmission speed between them. When transmitting, we must comply with ITU regulations, which impose the use of 1 % of the duty cycle. However, ITU also allows to employ the Listen Before Talk (LBT) algorithm, which will be our choice to meet this project's objectives.

Yo, **Fernando Tejero Rodríguez**, alumno de la titulación TITULACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 74538406Y, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Fernando Tejero Rodríguez

Granada, noviembre de 2022.

D. **Jorge Navarro Ortiz**, Profesor Titular de Universidad del Área de Ingeniería Telemática del Departamento Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada, y

Dª. **Natalia Chinchilla Romero**, estudiante de doctorado en el Área de Ingeniería Telemática del Departamento Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada

Informan:

Que el presente trabajo, titulado ***Transmisión de multimedia vía LoRa***, ha sido realizado bajo su supervisión por **Fernando Tejero Rodríguez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada, noviembre de 2022.

Los directores:

Jorge Navarro Ortiz **Natalia Chinchilla Romero**

Agradecimientos

Han sido cuatro años muy rápidos y a la vez duros, y quisiera dar las gracias a:

En primer lugar a Jorge y Natalia, por su paciencia y por todos los consejos y ayuda que me han dado para poder sacar el trabajo adelante.

A mis padres, por todo el esfuerzo y sacrificio para hacer posible que haya llegado hasta aquí.

A Iván y Pepelu, por ser los dos mejores compañeros, y sobre todo amigos, que me han dado estos cuatro años.

Y en especial a Cristina, por haberme dado un empujón cada vez que lo he necesitado, por hacerme creer en mí mismo y por animarme hasta el final.

Índice general

Índice de figuras

Índice de cuadros

Siglas

CAD Channel Activity Detection.

CRC Cyclic Redundancy Check.

CSMA Carrier Sense Multiple Access.

CSS Chirp Spread Spectrum.

DSSS Direct Secuence Spread Spectrum.

ETSI European Telecommunications Standards Institute.

FSK Frequency Shift Keying.

FUOTA Firmware Updates Over the Air.

Gp Ganancia de procesamiento.

IoT Internet of Things.

ISM Industrial Scientific and Medical.

LBT Listen Before Talk.

LoRa Long Range.

LoRaWAN Long Range Wide Area Network.

LPWAN Low Power Wide Area Network.

M2M Machine to Machine.

MAC Medium Access Control.

NB-IoT Narrow Band IoT.

OSI Open Systems Interconnection.

SF Spreading Factor.

SNR Signal to Noise Ratio.

ToA Time over the Air.

Capítulo 1

Introducción

1.1. Contexto y motivación

En los últimos años, el Internet de las Cosas (Internet of Things (IoT)) se ha convertido en una de los paradigmas más importantes a nivel mundial, ya que nos permite conectar objetos cotidianos, electrodomésticos, coches, termostatos, a Internet a través de dispositivos integrados. En cuanto a las cifras, para 2020 se esperaban un total de 10 mil millones de dispositivos conectados, y para 2025 este valor se estima que se duplicará [1].

IoT supone un salto muy importante en la calidad de vida de las personas en la sociedad, ofreciendo una gran cantidad de nuevas oportunidades en diversos campos como pueden ser el acceso a los datos, servicios específicos en educación, seguridad, asistencia sanitaria, transporte, entre otros.

Con la llegada del IoT desempeñan un papel fundamental las redes Low Power Wide Area Network (LPWAN), que se tratan de unas redes de telecomunicaciones que permiten el envío de paquetes de datos pequeños a cambio de un bajo consumo. Entre los distintos ejemplos de redes LPWAN destacan LoRa, Sigfox, Narrow Band IoT (NB-IoT) etc. LoRa, cuyas siglas significan largo alcance (*Long Range*), se trata de una tecnología patentada por la empresa Semtech, que emplea una modulación de radio frecuencia que proporciona comunicaciones de largo alcance (hasta 15 km en áreas rurales). Opera en la banda de frecuencia sin licencia Industrial Scientific and Medical (ISM) y es una de las tecnologías más usadas a nivel mundial debido a sus requisitos de energía tan bajos, permitiendo el uso de dispositivos con baterías de duración de hasta diez años [2]. LoRaWAN, como se comentará más adelante, define el protocolo de comunicación y la arquitectura del sistema para la red [3].

En cuanto al campo de aplicación de LoRa, este es muy extenso. Entre

los distintos usos que le podemos dar destacan la agricultura inteligente, las ciudades inteligentes, la medición inteligente de electricidad, agua, gas, el control industrial, etc. [4].

Una de las desventajas que posee LoRa es su limitada tasa de transferencia de datos. Al operar en una banda de frecuencia sin licencia, la transmisión de datos por LoRa debe cumplir con las restricciones impuestas por la European Telecommunications Standards Institute (ETSI) para utilizarla. El dispositivo final LoRa debe cumplir con un *duty cycle* o ciclo de trabajo de un 1 %, es decir, podrá hacer uso de una determinada sub-banda dentro de la banda de frecuencia en la que trabaje durante 36 segundos por cada hora de operación o aplicar una técnica de acceso al medio Listen Before Talk (LBT).

Dicha regulación perjudica gravemente en los casos en los que se requiere mandar una gran cantidad de datos, como por ejemplo en aplicaciones de videovigilancia. Por lo contrario, otros nodos, como podrían ser los sensores de medio ambiente, comparten un único servidor LoRa para enviar cantidades de información pequeñas y subutilizan el tiempo autorizado del 1 % del duty cycle. De ahí surge la motivación del proyecto, que será hacer uso de un mecanismo de compartición del tiempo de actividad de los nodos, es decir, utilizar un algoritmo que optimice el tiempo de acceso al medio. En la solución propuesta se utilizará el algoritmo LBT, el cual consiste en escuchar al medio antes de transmitir datos y, si está libre de transmisiones, entonces se procede al envío de los datos. De esta forma, el tiempo durante el cual se está transmitiendo será mayor, ya que lo haremos siempre y cuando el medio esté libre.

1.2. Objetivos del proyecto

Los objetivos principales a abordar en el proyecto son los siguientes:

1. Diseño e implementación de un programa para una mota LoRa que permita el envío de datos a otra mota a una velocidad mayor a la habitual en esta tecnología. Esto se conseguirá mediante el uso del algoritmo LBT, como se mencionó anteriormente.

2. Introducción de un módulo con cámara en una mota LoRa que permita capturar información multimedia.

3. Diseño e implementación de un servidor web que permita almacenar la información multimedia capturada por la mota LoRa receptora y mostrar por una interfaz web el contenido multimedia actualizado.

1.3. Estructura de la memoria

En esta sección se enumeran los diferentes capítulos que constituyen esta memoria, así como una breve descripción de cada uno de ellos.

1. **Capitulo 1. Introducción:** Consta de las siguientes partes:

   a) Contexto y motivación: Se presenta el problema a abordar en este proyecto así como su importancia en la actualidad.

   b) Objetivos del proyecto: Se detallan los objetivos a cumplir en el proyecto.

   c) Estructura del proyecto: Se comenta la información que contendrá cada capítulo de la memoria.

2. **Capítulo 2. Estado del arte:** En este capítulo se compara el proyecto con el panorama actual en cuanto a mejoras en la tasa de transferencia de datos para la tecnología LoRa.

3. **Capítulo 3. Fundamentos teóricos:** Aquí se explica todo el contenido teórico en el que se ha basado el proyecto. Se profundizará en la tecnología LoRa, que será la que utilizaremos en el proyecto. Se expondrá como funciona y sus características más importantes.

4. **Capítulo 4. Planificación de recursos y coste estimado:** La planificación de las tareas y los recursos empleados junto al coste total estimado son presentados en este capítulo.

5. **Capítulo 5. Diseño e implementación:** Se describen todos los pasos seguidos para el diseño e implementación de la solución presentada en este proyecto. Se desarrollará el programa que permitirá la transmisión de multimedia, así como el diseño del servidor web para mostrar este contenido.

6. **Capítulo 6. Prueba de concepto:** Tras haber diseñado e implementado el proyecto, se realiza la puesta en marcha de todos los elementos y se realizan pruebas para comprobar los resultados obtenidos.

7. **Capítulo 7. Conclusiones y líneas futuras:** En este capítulo final se presentan las principales conclusiones tras la elaboración del presente proyecto así como sus líneas futuras.

Capítulo 2

Estado del arte

En este capítulo se realizará un análisis de la literatura científica que tiene como objeto la mejora de velocidad en LoRa. También se expondrán sus mejoras, y su comparación con las ventajas del proyecto a abordar.

En este primer artículo [5], se pretende conseguir una transmisión eficiente de imágenes usando LoRa en sistemas IoT de monitoreo agrícola. Esto, como se comenta en la motivación del proyecto, supone un problema debido a la baja velocidad de datos de LoRa y al tamaño de carga útil insuficiente. La solución propuesta en el artículo consiste en el uso de un nuevo protocolo de entrega confiable, denominado Multi-Packet LoRa, con el fin de enviar mensajes grandes (imágenes). Las pruebas realizadas afirman que el mencionado protocolo reduce el tiempo de transmisión en una media de prácticamente el 25 % del tiempo cuando no hay pérdidas de paquetes, y de un 30 %, 42 % y 49 % con una tasa de pérdidas del 2 %, 5 % y 10 % respectivamente. Para que la transmisión utilizando el protocolo mencionado sea eficiente, se propone además el uso de otro protocolo de reserva del canal de datos, que se usarán en los casos en que haya múltiples nodos LoRa transmitiendo simultáneamente a una sola puerta de enlace. Este uso combinado de ambos protocolos consigue un mayor éxito de envío de imágenes que en el caso de uso ALOHA con parada y espera.

Como bien se comenta en el párrafo anterior, la transmisión de imágenes utilizando LoRa siempre se plantea como un problema, debido a su bajo rendimiento y su reducida velocidad de envío de datos. En el siguiente artículo [6], se propone el uso de Carrier Sense Multiple Access (CSMA), adaptando este a la capa física LoRa para mejorar la robustez de las transmisiones de largo alcance. Dicho problema es aún mayor cuando se transmiten grandes cantidades de datos, lo que provoca un mayor tiempo de transmisión de paquetes. Este aumento del tiempo de transmisión de una imagen es proporcional al aumento de la probabilidad de colisiones (a pesar de la limitación del 1 % del *duty cycle*). Los autores proponen como solución a este problema determinar qué enfoque de CSMA se adapta mejor a LoRa. Para

ello, utilizan un algoritmo para detectar actividad en el canal (CAD). Este presenta problemas de confiabilidad a medida que aumenta la distancia de transmisión, por ello, habrá que modificar algunos parámetros de CSMA. El DIFS se extenderá hasta el Time over the Air (ToA) del paquete más grande y durante este tiempo, el algoritmo CAD se ejecuta periódicamente. El propósito de elegir este DIFS reside en maximizar la probabilidad de detectar una transmisión activa que podría ser un mensaje largo con muchos CAD fallidos, que aparecerán como errores de mensajes cortos.

Siguiendo con el objetivo de optimizar la transmisión de imágenes a través de LoRa, el siguiente artículo [7] plantea la transmisión utilizando distintosSpreading Factor (SF)s y antenas, es decir, usar distintos SFs como método de multiplexación para reducir el tiempo de transmisión de las imágenes. Esta solución se implementa utilizando el formato de imagen JPG, ya que suelen tener un tamaño pequeño. Por tanto, en primer lugar se comprime la imagen a formato JPG, y luego se transmite. La solución consiste, por tanto, en que se asignará un número de paquetes determinado de la imagen a N nodos emisores, estos transmitirán de forma simultánea estos paquetes, y los N nodos receptores los recibirán y reconstruirán la imagen. En cuanto a los resultados experimentales obtenidos, efectivamente se aprecia una reducción del tiempo de transmisión de las imágenes.

La solución propuesta en el presente proyecto a esta problemática es, en lugar de la utilización del requisito del 1 % del duty cycle, utilizar un algoritmo de escucha al medio. En nuestro caso será el algoritmo LBT. Este nos permitirá optimizar el tiempo de espera y, por tanto, reducir de forma considerable el tiempo de transmisión de una imagen, ya que podremos transmitir siempre que el medio no este ocupado.

Capítulo 3

Fundamentos teóricos

3.1. Qué son LoRa y LoRaWAN

LoRa se trata de una técnica de modulación de radiofrecuencia utilizada en redes LPWAN. Es capaz de soportar transmisiones a través de largas distancias, de ahí su nombre (Long Range). Es importante mencionar que una de sus características más importantes reside en sus requisitos de energía muy bajos, lo que permitirá el uso de dispositivos con una vida de batería muy larga.

La regulación impone restricciones en cuanto al uso del canal en LoRa, en concreto al 1 % del tiempo de ciclo de trabajo. Esto perjudica a los casos en los que se requiere el envío de datos durante largos periodos de tiempo, como podría ser videovigilancia. Por otra parte, otros nodos, como podrían ser los sensores ambientales, son capaces de compartir un mismo servidor LoRa para enviar paquetes pequeños de datos, consiguiendo así la subutilización del tiempo de actividad impuesto por la regulación.

De ahí surge la idea de implementar un mecanismo de compartición del tiempo de actividad entre nodos. Este tipo de mecanismos permitirá a los dispositivos utilizar el tiempo de actividad sobrante de otros.

Otro punto importante a destacar es la diferencia entre LoRaWAN y LoRa. Por un lado, LoRaWAN se encarga de definir el protocolo de comunicación así como la arquitectura del sistema para la red. Por otro lado, LoRa, a nivel físico, creará la conexión que permitirá la comunicación a través de largas distancias. Por este motivo las funciones de LoRaWAN son indispensables para el funcionamiento de la red, ya que dependiendo de su arquitectura y del protocolo usado, se determinarán los distintos parámetros de la red, como el tiempo de las baterías en los nodos, la seguridad...

En cuanto a la topología que presenta LoRaWAN, se denomina topología en estrella, formada por los nodos finales y el nodo central. El nodo central o gateway es el lugar donde van a parar todos los datos provenientes de los nodos finales para ser redirigidos hacia su destino.

En este tipo de red, todos los nodos finales están conectados al nodo final. Esta característica permite escalar la topología y formar lo que se denomina topología en estrella extendida. Esta se diferencia de la primera en que algunos nodos de una subred en estrella se conectan como si fueran un nodo más de otra subred.

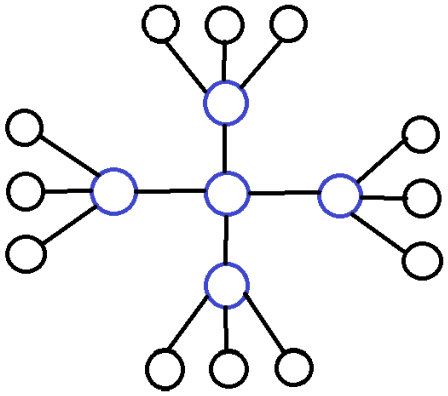

Figura 3.1: Topología de estrella extendida.

Respecto a los tipos de dispositivos finales que podemos encontrar en una red LoRaWAN destacamos tres clases [8]:

- **Clase A:** Permiten comunicaciones bidireccionales. A cada transmisión de enlace ascendente le siguen dos ventanas de recepción de enlace descendente. Esta clase de dispositivos son los que menos energía consumen ya que pasan la mayor parte del tiempo en inactividad. Este bajo consumo se debe a que disponen de dos modos de funcionamiento: transmitir datos cada período de tiempo (ejemplo: cada 10 minutos) o bien enviar datos con eventos.

- **Clase B:** Los dispositivos de esta clase permiten añadir más ventanas de recepción programadas en el tiempo. Cada vez que un dispositivo final entra en modo escucha, recibe una baliza sincronizada con la hora (*beacon*). Este hecho permite al servidor conocer cuando un dispositivo final está escuchando.

- **Clase C:** Esta clase de dispositivos están siempre encendidos, es decir, tienen sus ventanas de recepción siempre abiertas. Implementan las mismas dos ventanas de recepción que los de Clase A pero se diferencian de estos en que no cierran la segunda ventana hasta que transmiten. Por este motivo son los dispositivos que más energía consumen.

A modo de resumen, la especificación de LoRaWAN define la capa de control de acceso al medio (Medium Access Control (MAC)) para la red LPWAN. Según el modelo OSI, dicha capa se sitúa justo por encima de la capa física definida por LoRa y está desarrollada por la LoRa-Alliance, a diferencia de LoRa, que pertenece a la empresa Semtech [3].

A continuación, en la Figura 3.2, se exponen las diferencias entre LoRa y otras tecnologías utilizadas en IoT o soluciones de conectividad máquina a máquina (M2M).

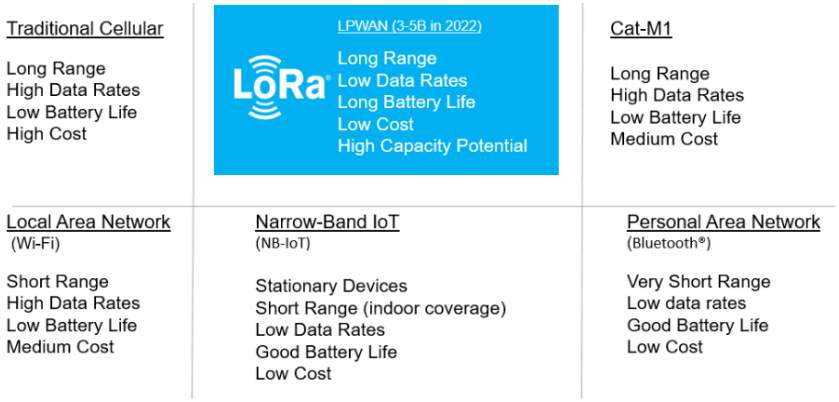

Figura 3.2: Comparativa de LoRa con otras tecnologías M2M [2].

Los operadores de móviles Europeos pueden hacer uso de LoRaWAN. Esto es interesante ya que satisface la necesidad de una gran duración de la batería a cambio de el envío de paquetes más pequeños y una mayor latencia.

Algunas de las ventajas que conlleva el despliegue de una red LoRaWAN se aprecian en la figura 3.3:

1. Largo alcance.
2. Gran duración de las baterías.
3. Alta capacidad.
4. Bajo coste.
5. Geolocalización.
6. Firmware Updates Over the Air (FUOTA).
7. Roaming.
8. Seguridad.

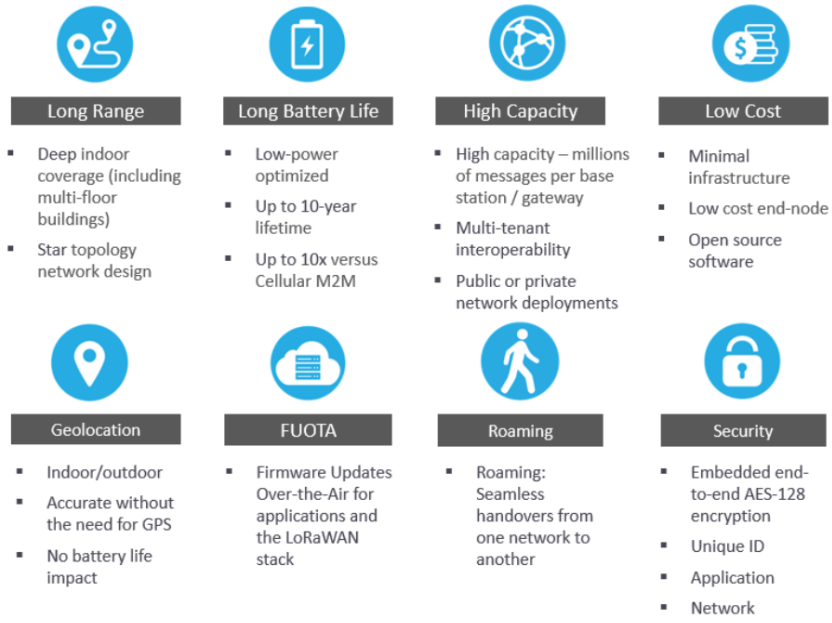

Figura 3.3: Ventajas del uso de LoRaWAN [2].

Un único *gateway* o también denominado pasarela de red LoRaWAN, es capaz de transmitir y recibir señales a una distancia de más de 15 kilómetros en áreas rurales, y hasta 5 kilómetros en áreas urbanas (esto se debe a los distintos obstáculos que se presentan en el camino).

Relativo a la duración de las baterías de estos dispositivos, la energía necesaria para transmitir un paquete de datos es insignificante ya que los paquetes de datos transmitidos son muy pequeños y solo se transmiten unas pocas veces al día. Sumado a la ventaja de que consumen poca energía a la hora de transmitir, cabe resaltar que cuando los dispositivos finales están apagados, la energía que consumen es insignificante, lo que permitirá que sus baterías duren muchos años.

Hablando de su capacidad, una red LoRaWAN puede soportar hasta millones de mensajes, sin embargo, el número de mensajes que podrá soportar dependerá siempre del número de *gateways* instalados, es decir, el número de mensajes admitidos es proporcional a los *gateways* utilizados. Un *gateway* puede soportar cientos de miles de mensajes al día, si cada dispositivo envía 10 mensajes al día, un solo *gateway* será capaz de admitir 10000 dispositivos. Por tanto, cuando se necesite aumentar la capacidad de la red solo habrá que añadir más *gateways*.

Y por último, en cuanto al coste se refiere, dadas las capacidades de los nodos finales y *gateways* LoRa, solo se necesitan unos pocos *gateways* para servir a una multitud de nodos finales. Esto conlleva que los gastos para el

funcionamiento serán bastante bajos. Además, cuando los módulos de RF de LoRa integrados en nodos finales se utilizan junto con el estándar LoRaWAN, el retorno de inversión será considerable, ya que como se menciona en el párrafo anterior, con unos pocos *gateways* se podrá servir a una multitud de dispositivos.

3.2. Radio Modulación y LoRa

LoRa utiliza una técnica patentada de modulación de espectro ensanchado derivada de la tecnología existente Chirp Spread Spectrum (CSS). Esta ofrece un equilibrio entre velocidad de datos y sensibilidad mientras opera en un canal de ancho de banda fijo de 125 KHz o 500 KHz (para canales de enlace ascendente), y 500 KHz (para canales de enlace descendente). El hecho de que LoRa utilice SFs adaptativos permite a la red optimizar la duración de las baterías de los dispositivos ya que esto permite adaptar de forma dinámica los niveles de velocidad de datos y potencia de cada nodo. Por ejemplo, un dispositivo final que se encuentra cerca de un gateway deberá transmitir con un SF bajo, por lo contrario, un dispositivo final que se encuentre a varios kilómetros de un *gateway* deberá hacerlo con un SF mucho mayor, lo que permitirá una mayor sensibilidad en el receptor, aunque el tiempo de transmisión será mayor, debido a ese ensanchamiento en el tiempo.

LoRa se trata de una implementación de capa física pura, o "bits", como se define en el modelo de red de siete capas de OSI y que se muestra en la Figura 3.4. El medio de transmisión utilizado será el aire, en lugar del cableado, ya que lo que se transmitirá serán ondas de radio LoRa desde un transmisor de RF ubicado en un dispositivo IoT hacia un receptor de RF ubicado en un *gateway*, y viceversa.

En un sistema Direct Secuence Spread Spectrum (DSSS), la portadora de la señal del transmisor cambia de acuerdo a una secuencia de código como se muestra en la Figura 3.5. Al multiplicar la señal de datos por un patrón de bits definido a una velocidad mayor, surge una señal más rápida con componentes en frecuencia más altos en comparación con la señal original, lo que se traduce en que el ancho de banda de la señal resultante se extiende más allá del de la señal original. Cuando la señal que se transmite llega al receptor de RF, se multiplica por una copia del patrón de bits utilizado en el transmisor, por lo tanto, tenemos como resultado una réplica idéntica de la señal de datos original.

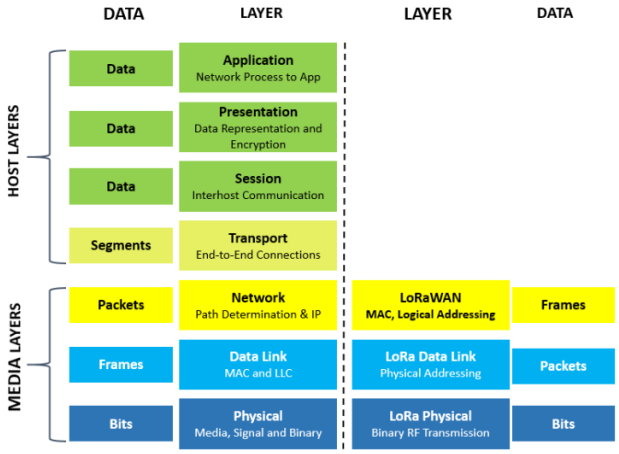

Figura 3.4: LoRa y LoRaWAN en el modelo de 7 capas OSI [2].

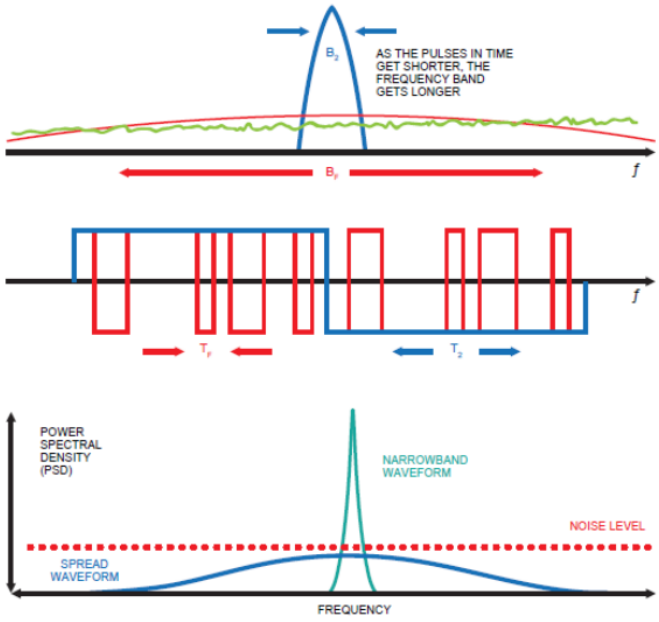

Figura 3.5: Sistema DSSS [2].

La relación logarítmica de la velocidad de la secuencia de código y la velocidad de bits de la señal de datos se denomina Ganancia de procesamiento (Gp). Dicha ganancia es la que permite recuperar la señal de datos original en el receptor, incluso en los casos en los que la relación señal/ruido (Signal to Noise Ratio (SNR)) es negativa. LoRa presenta una Gp superior a la modulación Frequency Shift Keying (FSK), lo cual permitirá un nivel de potencia de salida del transmisor baja a la vez que se mantiene una

velocidad de datos de la señal constante [2].

Una de las desventajas a mencionar sobre los sistemas DSSS es el hecho de que requiere un reloj de referencia de muy alta precisión. La tecnología LoRa CSS ofrece una alternativa DSSS de bajo coste y consumo, pero muy robusta, y que no necesita de un reloj de referencia de alta precisión. En la modulación LoRa la difusión del espectro de la señal se consigue generando una señal chirp (chirrido) que va variando en frecuencia, como se observa en la figura 3.6.

La ventaja del método mencionado es que las compensaciones en tiempo y frecuencia entre el transmisor y receptor son equivalentes, lo que provoca una reducción de la complejidad en el diseño del receptor. El ancho de banda de este "chirrido" es igual al ancho de banda espectral de la señal, la cual transporta los datos de un dispositivo final a un *gateway* a una velocidad de datos más alta. La modulación LoRa dispone también de un esquema de corrección de errores variable que implica una mejora de la robustez de la señal transmitida, por cada cuatro bits de información transmitidos, se envía un quinto bit de información de paridad.

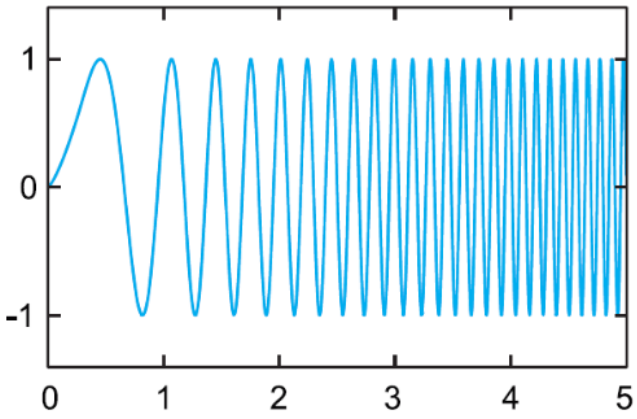

Figura 3.6: Sistema DSSS [2].

3.3. Propiedades clave de la modulación LoRa

Algunas de las propiedades más importantes de LoRa son las comentadas a continuación [9]:

3.3.1. Ancho de banda escalable

La modulación LoRa es escalable tanto en ancho de banda como en frecuencia, y puede usarse en modulación de banda estrecha y para banda ancha. A pesar de que existen esquemas de modulación de banda ancha y banda estrecha, LoRa se puede adaptar para cada modo de operación con unos simples cambios en su configuración.

3.3.2. Envolvente constante y bajo consumo

LoRa se trata de una modulación de envolvente constante, lo que implica un bajo coste y consumo, y una alta eficiencia. Además, debido a la ganancia de procesamiento asociada a LoRa, la potencia de salida del transmisor se puede reducir en comparación a un enlace FSK con el mismo o mayor presupuesto de enlace (distancia).

3.3.3. Alta robustez

Debido a su naturaleza asíncrona, las señales LoRa son robustas y, por tanto, muy resistentes a las interferencias dentro y fuera de banda.

3.3.4. Resistente a efectos multipath y al desvanecimiento

La modulación LoRa ofrece también inmunidad a los efectos multipath, que son aquellos que se producen cuando la señal llega al receptor proveniente de más de dos caminos distintos, y esto se debe a los rebotes de la señal a lo largo del trayecto. También es inmune al desvanecimiento, lo que las hace ideal para su uso en entornos urbanos y suburbanos.

3.3.5. Resistente al efecto Doppler

El efecto Doppler provoca un pequeño cambio de frecuencia en cuanto al eje de tiempo de la señal. Esta tolerancia al desplazamiento en frecuencia elimina el requisito ya mencionado del reloj de referencia, y, por lo tanto, hace que LoRa sea ideal para comunicaciones de datos desde dispositivos móviles.

3.3.6. Capacidad de largo alcance

Para una potencia de salida y un *throughput* fijo, el presupuesto de enlace de LoRa supera al de FSK. Esto se debe a que cuando se combina con la

robustez ante interferencias y desvanecimientos ya mencionada, supone una mejora del presupuesto de enlace del cuádruple e incluso más.

3.3.7. Mayor capacidad de red

Cabe mencionar que los SF de la modulación LoRa son ortogonales entre sí, lo que se traduce en que la señales moduladas con distintos SF y transmitidas a través del mismo canal de frecuencia en el mismo período de tiempo no interfieren entre sí. En cambio, las señales moduladas con distintos SF aparecen como ruido para las otras señales.

3.3.8. Alcance y Localización

Una propiedad inherente de LoRa es la capacidad de discriminar linealmente entre los errores de tiempo y de frecuencia. LoRa es la modulación ideal para aplicaciones de radar y, por lo tanto, es la que mejor encaja en aplicaciones de alcance y localización como pueden ser los servicios de localización en tiempo real.

3.3.9. Factor de ensanche

La elección del factor de ensanche tendrá relevancia en la capacidad de una celda LoRa. Las colisiones al transmitir dependerán de la relación de potencia de las señales, y esta dependerá de la posición de los dispositivos.

El factor de ensanche se define como la relación entre la tasa de símbolo Rs y la tasa de chip Rc [10].

$$SF = \log_2 (Rc/Rs) \tag{3.1}$$

LoRa dispone de un total de seis factores de ensanche distintos, que van del SF7 al SF12, a medida que aumenta este, más lejos se podrá transmitir la señal y podrá aun así ser recibida sin errores en el receptor. Estos factores de ensanche tienen la propiedad de que son ortogonales entre sí, es decir, señales moduladas con diferentes factores de ensanche y transmitidas en el mismo canal de frecuencia al mismo tiempo, no interfieren entre sí. Esto es así porque cada factor de ensanche da lugar a una tasa de chirrido distinta. La tasa de chirrido se define como el cambio en frecuencia respecto al tiempo, por lo tanto, será distinta para cada factor de ensanche.

En la figura 3.7 se muestran la tasa de bits,la distancia alcanzada (que dependerá del terreno, obstáculos...) y el ToA obtenidos para distintos SF con una carga de 11 bytes, en concreto desde el SF7 al SF10, en un canal de enlace ascendente (Up Link) de 125KHz.

Spreading Factor (For UL at 125 KHz)	Bit Rate	Range (Depends on Terrain)	Time on Air for an 11-byte payload
SF10	980 bps	8 km	371 ms
SF9	1760 bps	6 km	185 ms
SF8	3125 bps	4 km	103 ms
SF7	5470 bps	2 km	61 ms

Figura 3.7: Spreading Factor [2].

Como conclusión, al observar los datos de la tabla, y corroborando lo expuesto sobre el SF, vemos que para un SF menor, se reduce la distancia alcanzada por el enlace, pero conseguimos una tasa de bits mayor, así como un ToA más reducido.

3.4. Colisiones de datos y ortogonalidad del Spreading Factor

Con LoRa, los paquetes que utilizan diferentes SF son ortogonales, lo que significa que son invisibles entre sí, simplemente aparecen como ruido entre ellos. Por tanto, dos paquetes que lleguen a la vez en el tiempo y al mismo canal de recepción con diferentes SF no colisionarán entre sí y ambos serán demodulados por el chip módem del *gateway*. Sin embargo, dos paquetes con el mismo SF que lleguen a la vez y al mismo canal podrían colisionar. Como única excepción, si uno de los dos paquetes es 6 dB mayor que el otro, sobrevivirá.

La capacidad de una red LoRaWAN dependerá de su número de gateways, como se menciona en apartados anteriores, el número de dispositivos soportados será directamente proporcional al número de gateways desplegados. Para maximizar la capacidad de la red, es importante utilizar un mecanismo de velocidad de datos adaptable (ADR). El objetivo principal de este mecanismo reside en ahorrar la energía de las baterías de los dispositivos finales. Esto se consigue haciendo que los dispositivos finales más cercanos a los *gateways* transmitan con el SF más bajo para reducir el ToA, así se conseguirá prolongar la vida de las baterias. Con esto se pretende mantener un equilibrio entre la energía de la bateria y la distancia, ya que utilizando un SF mayor, los *gateways* podrán conectarse a dispositivos que se sitúan más lejos.

3.5. Estructura de un paquete LoRa

La estructura que presenta un paquete LoRa es la siguiente [11]:

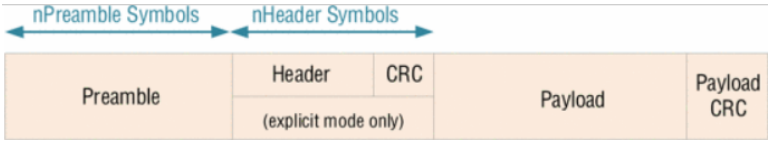

Figura 3.8: Estructura de un paquete LoRa.

Donde podemos diferenciar cinco campos distintos:

- Preámbulo: se utiliza para detectar la señal LoRa antes de comenzar la transmisión. Permite sincronizar el emisor con el receptor.
- Cabecera: contiene campos como pueden ser el origen, destino y el número de secuencia.
- Cyclic Redundancy Check (CRC): es un código de detección de errores que permite detectar cambios accidentales en los datos.
- Payload: conjunto de datos transmitidos.
- Payload CRC: código de detección de errores del payload.

3.6. Channel Activity Detection (CAD)

El modo de detección de actividad de canal (CAD) [12] es un mecanismo que nos permitirá reducir el número de colisiones entre paquetes, con la consiguiente mejora de la escalabilidad. Al usar el protocolo ALOHA, los nodos transmiten al medio en el momento que ellos quieran y esto provoca un gran número de colisiones. Para solucionar esta deficiencia, se utilizará este mecanismo para realizar una escucha del medio previa a la transmisión.

En el momento en el que se entra en el modo CAD, el nodo actuará como receptor, este comprobará la señal de interferencia para verificar si el canal está ocupado antes de comenzar la transmisión. Si el canal está ocupado, se implementa la función de retroceso (*backoff*) para retrasar la transmisión y así evitar colisiones.

En general, el modo CAD es utilizado para detectar el preámbulo de LoRa. Así, cuando un nodo activa el modo CAD y encuentra un canal que está ocupado, el nodo es capaz de retrasar la transmisión del paquete para evitar la colisión.

Según la especificación de Semtech, el modo CAD se clasifica en tres etapas principales: activación, recepción y procesamiento.

En primer lugar, el nodo solicita un tiempo TCADwakeup para activar el modo de recepción.

$$TCAD_{wakeup} = 32/BW \tag{3.2}$$

Después de esto, el nodo abre una ventana de recepción. El tiempo empleado en esta etapa se puede calcular como:

$$T_s = 2^{SF}/BW \tag{3.3}$$

Por último, el nodo procesa los datos entrantes para comprobar la presencia de un preámbulo. El tiempo de procesamiento será:

$$TCAD_p = (2^{SF} \cdot SF)/(1750 \cdot 10^3) \tag{3.4}$$

Cabe mencionar que el tiempo de duración del modo CAD dependerá del factor de ensanche y del ancho de banda, como se puede apreciar en las expresiones para el cálculo de los tiempos en cada etapa.

Capítulo 4

Planificación, recursos y coste estimado

En este capítulo, en primer lugar se expondrá la planificación que se ha llevado a cabo para desarrollar el proyecto, después se comentarán los recursos utilizados para este fín y, por último, se estimará el coste total del proyecto.

4.1. Planificación de tareas

- **Fase de estudio y análisis**

    1. *Estudio de LoRa y sus características:* será primordial conocer como funciona LoRa, en cuanto a requisitos de funcionamiento se refiere, como el tipo de modulación usado, el tiempo máximo permitido por transmisión, que en este caso sera del 1 %. Dicho conocimiento nos permitirá elegir con criterio los parámetros de transmisión, así como conocer la manera en la que funciona esta tecnología.
    2. *Estudio de la librería Pycom:* es imprescindible ya que nos permitirá programar las dos motas FiPy. En nuestro caso, la programación de las motas se llevará a cabo utilizando el editor de código Atom.
    3. *Estudio del lenguaje MicroPython*: será la base de nuestro proyecto, ya que el código con el que programaremos las dos motas estará escrito en dicho lenguaje de programación. Para ello necesitaremos un software que nos permita editar, depurar y modificar los scripts que posteriormente se subirán a ambas motas FiPy para ejecutarlos, que como se comenta en el punto anterior, será Atom.

4. *Estudio de las peticiones PHP:* es importante conocer como funcionan las peticiones PHP. Esta será la forma en la que se envíe el contenido multimedia al servidor local donde se mostrará posteriormente.

- **Fase de diseño e implementación:**

  1. *Pruebas de transmisión de datos entre motas:* en primer lugar tomaremos dos scripts ya elaborados cuya función es mandar una cadena de caracteres entre ambas motas. Comprobaremos que se transmite sin ningún problema para verificar que las motas funcionan correctamente.
  2. *Implementación de librería para tomar una foto:* el siguiente paso será, tomando como base los scripts de la anterior tarea, extender ambos de manera que además de transmitir la cadena de caracteres, se tome una foto gracias a la cámara incorporada en la mota emisora y se almacene en la tarjeta micro-SD de esta.
  3. *Transmisión de la imagen desde la mota emisora a la receptora:* se trata de la parte más importante del proyecto, en la cual tendremos que mandar la imagen almacenada en la tarjeta micro-SD de la mota emisora hacia la mota receptora. Para ello, debemos tener en cuenta que debido al bajo ancho de banda del que dispone LoRa, tendremos que fragmentar la imagen para poder transmitirla, mandar los distintos fragmentos de uno en uno y reconstruir la imagen una vez tengamos todos los datos en la mota receptora.
  4. *Creación de un servidor web Apache y envío de imágenes mediante PHP:* para visualizar las imágenes tendremos primero que enviarlas desde la mota receptora hacia un servidor, para posteriormente mostrarlas en pantalla. Para la creación del servidor web se hará uso de un sistema operativo Ubuntu, lanzado mediante el software VirtualBox.

- **Fase de prueba de concepto:** esta parte consiste en comprobar el funcionamiento del proyecto, es decir, verificar que el diseño e implementación son correctos.

- **Fase de escritura de la memoria:** Esta será la última parte que compone el proyecto. Se trata de recoger por escrito toda la información recopilada previa al diseño, así como los resultados que se han ido obteniendo a lo largo del proceso. Esta parte culmina con la elaboración de una conclusión en base a todo lo realizado y aprendido, así como las líneas futuras del proyecto.

En la tabla 4.1 se recogen las distintas tareas junto con la temporización de cada una de ellas. Dicha tabla irá seguida de un diagrama de Gantt, donde se mostrará de forma visual la planificación inicial de todo el proyecto.

Tarea	**Días**
Estudio de LoRa y sus características	12
Estudio de librería Pycom	6
Estudio de las peticiones PHP	7
Pruebas de transmisión de datos entre motas	12
Implementación de librería para tomar una foto	8
Transmisión de la imagen	20
Creación servidor local Apache	14
Fase de prueba y concepto	6
Fase de escritura de la memoria	75

Cuadro 4.1: Planificación temporal.

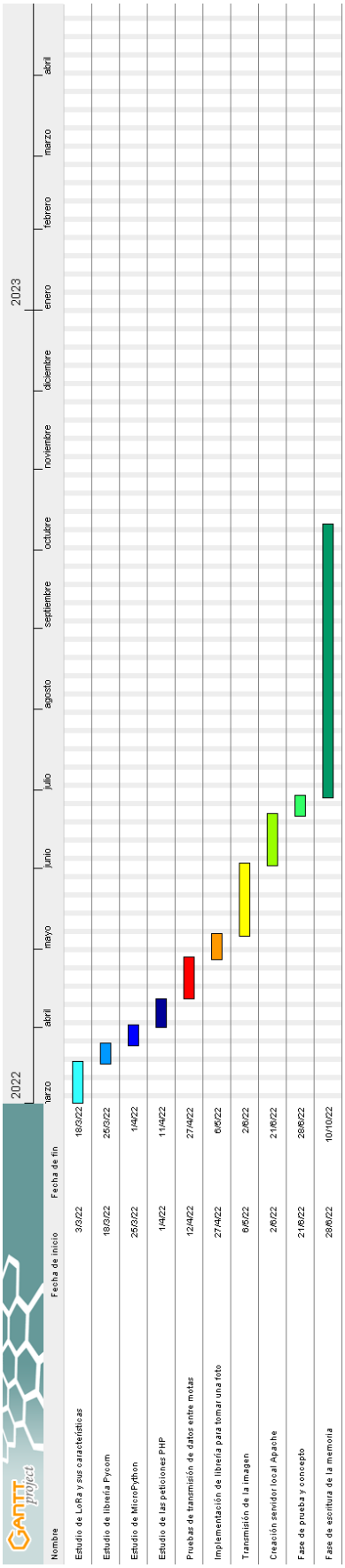

Figura 4.1: Diagrama de Gantt.

4.2. Recursos empleados y coste

En este apartado se expondrán tanto los recursos hardware, como los software y humanos. En última instancia se mostrará una tabla con el coste total del proyecto.

4.2.1. Recursos Hardware

El primer y principal recurso utilizado se trata de un ordenador portátil de uso personal. Un HP Pavilion Power 15-cb009ns [13].

Figura 4.2: HP Pavilion Power 15 [13].

Procesador	Intel Core i7-7700HQ
Memoria RAM	8GB
Almacenamiento	1024 GB HDD y 256GB SDD
Tarjeta Gráfica	Nvidia GeForce GTX 1050 2GB

Cuadro 4.2: Especificaciones ordenador portátil [13].

El segundo recurso se trata de dos placas de desarrollo Pycom FiPy 1.0, cuyas características principales se exponen en la siguiente tabla:

Potente CPU
WiFi, BLE, cellular LTE-CAT M1/NB1, LoRa y Sigfox
Alcance WIFI de 1Km
Lenguaje MicroPython soportado
Muy baja potencia

Cuadro 4.3: Especificaciones placas Pycom FiPy 1.0 [14].

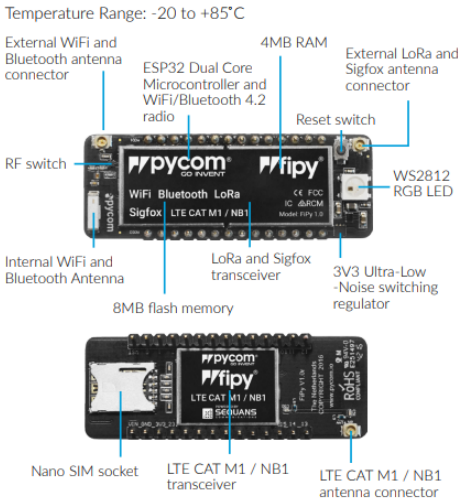

Figura 4.3: Pycom FiPy 1.0.

El tercer elemento, que va en conjunto con las placas de desarrollo Pycom FiPy, se trata de dos placas de expansión universales Pycom 3.0. Sus características principales son:

Batería USB y LiPo
Cargador de batería LiPo
Ranura MicroSD
TPS2115A con proteción de voltaje inverso
Pines para la conexión de distintas placas, en nuestro caso la Pycom FiPy 1.0

Cuadro 4.4: Especificaciones Expansion Board 3.0 [15].

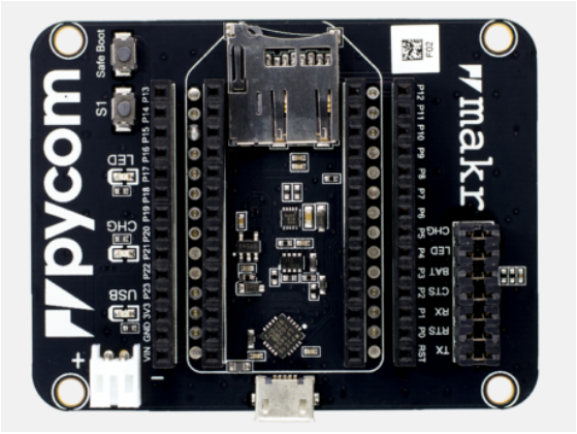

Figura 4.4: Expansion Board 3.0.

El cuarto elemento se trata del Módulo de Cámara ArduCAM c/ 2MP Plus OV2640. Esta será la cámara integrada en la mota Pycom que se encargará de tomar las imágenes.

Figura 4.5: Mini módulo de cámara OV2640 2MP [16].

También se usarán cables macho-hembra para la conexión de la cámara OV2640, como bien se comentará más adelante en el capítulo de diseño e implementación, donde se indicarán los pines a conectar entre la mota y el módulo de la cámara.

Figura 4.6: Cables macho-hembra.

El quinto elemento se trata de cuatro antenas LoRa, dos para cada mota Pycom LoPy.

Figura 4.7: Kit Antena LoRa [17].

La tabla 4.5 recoge el coste total en cuanto a recursos hardware.

Recurso Hardware	**Unidades**	**Precio unidad (€)**	**Precio total (€)**
Ordenador portatil	1	700	100
Placa Pycom FiPy 1.0	2	65	130
Expansion board 3.0	2	18	36
Modulo OV2640	1	48	48
Antenas LoRa Pycom	4	14	56
Cables macho-hembra	8	0.05	0.4
Coste Total			**370.4**

Cuadro 4.5: Recursos hardware totales.

4.2.2. Recursos Software

Los programas utilizados han sido los siguientes:

1. **Overleaf:** se trata de un editor LaTeX colaborativo en la nube que se utiliza para escribir, editar y publicar documentos científicos [18].

2. **Atom:** es un editor de código fuente de código abierto con soporte para múltiples plug-in escritos en Node.js y control de versiones Git integrado, desarrollado por GitHub [19].

3. **Python:** se trata de un lenguaje de alto nivel de programación multiparadigma, que soporta la orientación a objetos, programación imperativa y, en menor medida, programación funcional [20].

4. **Gantt Project:** [21] se trata de un programa de código abierto cuya función es la administración de proyectos usando el diagrama de Gantt.

5. **Oracle VM VirtualBox:** [22] consiste en un software de virtualización multiplataforma de código abierto, permite a los desarrolladores entregar código más rapido, ya que permite ejecutar múltiples sistemas operativos en un solo dispositivo.

En cuanto al coste de los recursos software, solo incluiremos el coste de la licencia del sistema operativo utilizado, que se trata de Windows 10, su precio es de 22€.

4.2.3. Recursos Humanos

Aquí se incluye el tiempo dedicado al proyecto por parte del tutor, la estudiante de doctorado y el alumno.

En el caso del tutor, con el título de Profesor Titular de la Universidad, el precio por hora dedicada se estima que es de 50€, así como el de la estudiante de doctorado. Por otro lado, el precio por hora dedicada del alumno es de 25€, cuyo título es Graduado en Ingeniería de Tecnologías de Telecomunicación.

La siguiente tabla 4.6 muestra una aproximación de las horas dedicadas por parte de los tutores y el alumno.

Persona	**Total horas**
Tutor	15
Tutora	15
Alumno	320

Cuadro 4.6: Recursos Humanos.

4.2.4. Coste Total del proyecto

Aquí se realiza la suma de los costes de todos los recursos utilizados, que se mostrará en la siguiente tabla:

Persona	**Precio/hora(€)**	**Coste(€)**
Hardware		370.4
Software		22
Tutor	50	750
Tutora	25	375
Alumno	25	8000
Coste total		**9571.4**

Cuadro 4.7: Coste total del proyecto.

Capítulo 5

Diseño

5.1. Diseño del hardware

Una de las partes más importantes e imprescindibles del proyecto se trata del diseño hardware. Para ello, los pasos que se han seguido son los comentados a continuación.

Disponemos, como bien se comenta en la planificación, de dos motas FiPy, una de ellas la transmisora y la otra la receptora. Además de estas, también es primordial una cámara, que como se explica en el apartado anterior será el mini módulo OV2640.

Sumado a estos tres elementos, será imprescindible disponer de un ordenador, con sus correspondientes puertos USB, que serán los que nos permitan conectar las motas para poder programarlas con ayuda de Atom.

Una vez cosultada la documentación correspondiente a los pines, tanto de la mota FiPy como del módulo OV2640, procederemos a la conexión de ambos dispositivos utilizando cables de tipo macho-hembra. En cuanto a la conexión de cada par de pines, se ha tomado como referencia la librería [23], donde la asignación que siguen es la siguiente:

- **Cable azul:** Pin **P9** de la mota con **CS** de la cámara.
- **Cable naranja:** Pin **P10** de la mota con **SCK** de la cámara.
- **Cable verde:** Pin **P11** de la mota con **MOSI** de la cámara.
- **Cable rojo:** Pin **GND** de la mota con **GND** de la cámara.
- **Cable marrón:** Pin **3V3** de la mota con **VCC** de la cámara.
- **Cable blanco:** Pin **P22** de la mota con **SCL** de la cámara.
- **Cable negro:** Pin **P21** de la mota con **SDA** de la cámara.
- **Cable amarillo:** Pin **P14** de la mota con **MISO** de la cámara.

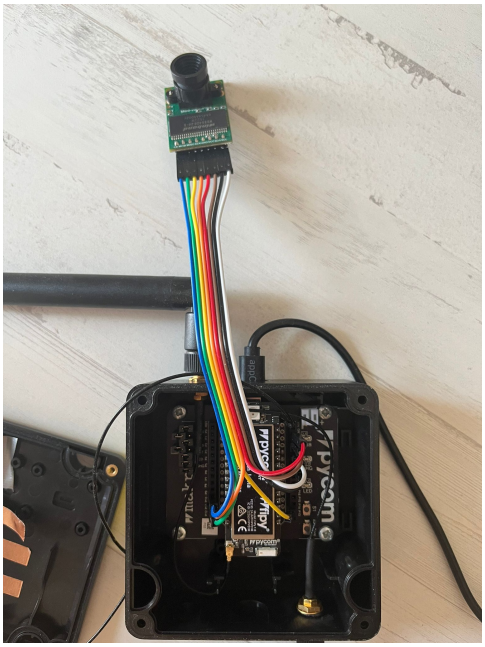

Figura 5.1: Conexión Física OV2640 con mota FiPy.

En la figura 5.1 se aprecian las conexiones físicas reales.

Por último, la identificación de pines de cada uno de los dos elementos, extraído de su datasheet es incluida en la figura 5.2. Los pines del módulo OV2640 pueden visualizarse en la figura 5.3.

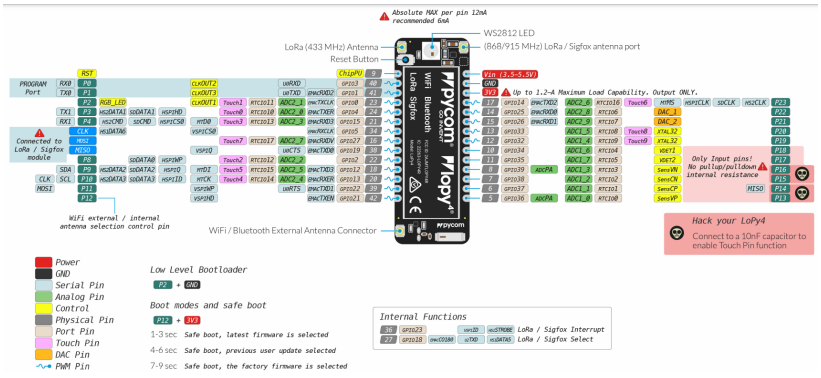

Figura 5.2: Pines FiPy [24].

Figura 5.3: Pines OV2640 [25]

Una vez tengamos correctamente conectada la cámara a la mota FiPy transmisora, el único paso restante sería la conexión de ambas motas al ordenador personal utilizado para la realización del proyecto. Es importante mencionar que las motas disponen de un conector de tipo Micro-USB, por lo tanto, haremos uso de un cable de tipo USB/Micro-USB para llevar a cabo la conexión de los dispositivos mencionados.

A su vez, cabe destacar el detalle de que las motas también disponen de un puerto de conexión micro-SD, que se utiliza para almacenar las imágenes de forma temporal en las motas FiPy.

5.2. Diseño del software

Como primer paso, es imprescindible tener instalada la extensión "Pymakr"de Atom, que, como bien se comenta en el capítulo de planificación, será el software que nos permitirá editar nuestro código escrito en MicroPython. Pymakr es una extensión que proporciona una interfaz para conectarnos a la placa Pycom y nos permite comunicarnos con ella utilizando la línea de comandos REPL. Entre las funcionalidades más usadas se encuentra el hecho de subir los scripts a la mota FiPy para que pueda ejecutarlos.

5.2.1. Pruebas simples

La elaboración del código final que nos permitirá la transmisión/recepción de la imagen vía LoRa se ha llevado a cabo por pequeñas fases, siguiendo los pasos comentados a continuación.

Los pasos que se han seguido han sido, en primer lugar, la comprobación de transmisión de datos entre motas. El otro paso ha consistido en la toma de una imagen y su posterior almacenamiento en la tarjeta micro-sd. Estos dos pasos mencionados nos permiten verificar la correcta conexión de las motas y la cámara, así como su funcionamiento.

5.2.2. Prueba simple de transmisión entre motas

Para comprobar la correcta transmisión y recepción de datos entre ambas motas FiPy, se hará uso de un repositorio ya existente, tanto para la mota emisora como para la receptora. Este repositorio contiene dos scripts de extensión *".py"*, ya que, como se comenta anteriormente, se trabajará con MicroPython. Dichos scripts nos permitirán enviar una cadena de bytes procedente de la mota transmisora hacia la mota receptora.

Para una correcta transmisión de los datos, y mediante el uso del módulo LoRa, debemos establecer previamente los parámetros que caracterizarán la transmisión, que serán los siguientes:

- **Región:** EU868
- **Potencia de transmisión:** TXPOWER = 10dBm
- **Frecuencia:** CHANNEL = 868100000 Hz
- **Ancho de banda:** BANDWIDTH = 500KHz
- **Spreading Factor:** SF = 7
- **CodingRate:** CODINGRATE = CODING4_5
- **Preámbulo:** PREAMBLE = 8

Una vez establecidos todos los parámetros de transmisión, se comprueba que las motas envían y reciben la cadena de bytes correctamente.

5.2.3. Prueba de envío y almacenamiento de la imagen

La otra prueba consistirá en utilizar el código del punto anterior, y combinarlo con aquel que permita hacer una foto y almacenarla en la tarjeta micro-SD. Para que esto sea necesario es imprescindible incorporar los módulos *OV2640*, *OV2640_constants*, *OV2640_hires_constants* y *OV2640_lores_constants*.

5.2.4. Mota Emisora

El código que permite a la mota FiPy emisora tomar la foto, fragmentarla y enviarla a la receptora se estructura tal y como se muestra en la figura 5.4:

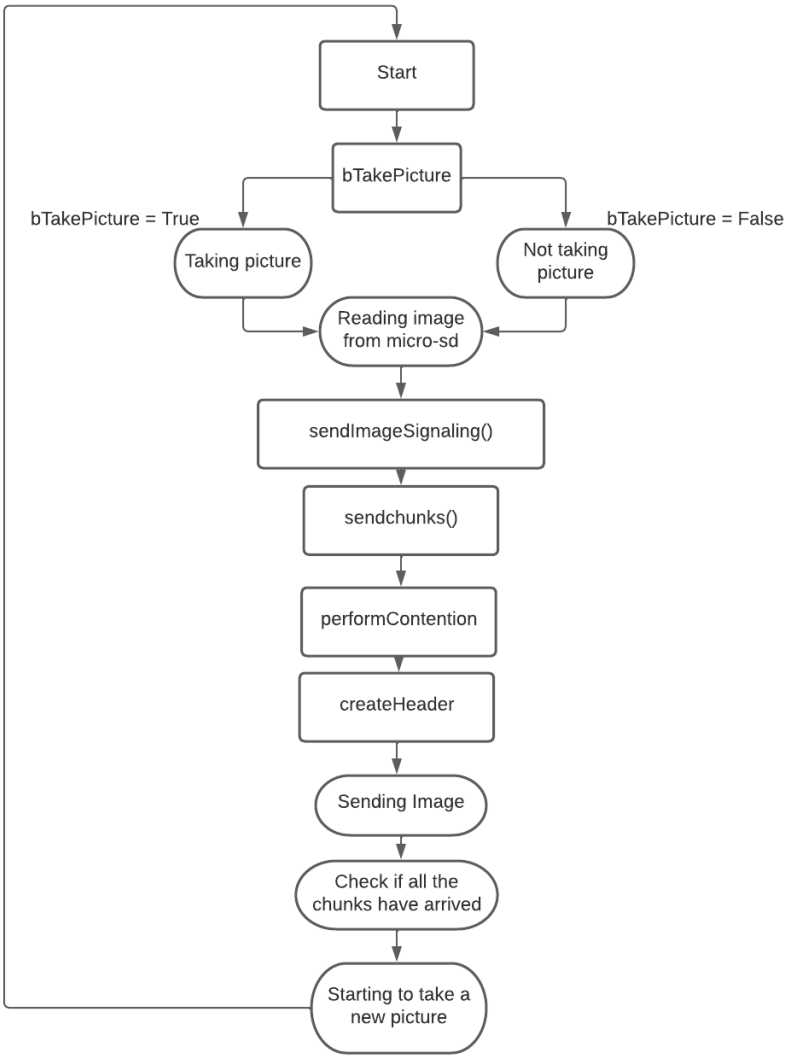

Figura 5.4: Diagrama de estados mota emisora.

Las funciones principales que componen el proceso de toma, fragmentación y envío de la imagen son las comentadas a continuación.

Funciones relacionadas con la cabecera de los datos:

- Función **createHeader**.Consiste en crear una cabecera que precederá

a cada paquete de datos enviado (cada fragmento de la imagen), cuyos parámetros de entrada son src (origen), dst (destino), transID (sirve para identificar la imagen, en el hipotético caso de que haya varias motas distintas transmitiendo a la vez) y, por último, seq (número de secuencia, que indica el primer byte del paquete en cuestión, esto nos ayudará con la retransmisión de los paquetes perdidos durante toda la transmisión, ya que nos dará la posibilidad de colocarlos en su lugar correspondiente).

- Funciones **getHeaderFromPacket** y **getDataFromPacket**. Ambas funciones reciben el paquete de datos como argumento de entrada, separan respectivamente la cabecera y la parte de datos útiles para poder analizarlos posteriormente.

- Funciones **getSrc**, **getDst**, **getTransID** y **getSeq**. Una vez extraída la cabecera del paquete con la función anterior, se utilizará esta como argumento de entrada de las cuatro funciones mencionadas para poder extraer el origen, destino, el número de transacción y el número de secuencia del paquete.

Funciones relacionadas con la señalización:

- Función **sendImageSignaling**. Recibe como entrada el destino, el número de transacción y el tamaño de la imagen y crea una cabecera con estos datos para enviarla al receptor junto con el tamaño de la imagen, que será necesario conocerlo al otro lado de la transmisión para recibirla correctamente.

- Función **receiveBlockNACK**. Esta función se utiliza para retransmitir aquellos paquetes que se han perdido. En el caso del emisor, este recibirá el mensaje "blockNACK", que es enviado por la mota receptora cuando ha recibido el último byte de la imagen o bien lleva 5 segundos sin recibir nada. En este mensaje se indican los paquetes que se han perdido y el emisor lo analiza para extraer los paquetes que faltan, y los pueda volver a enviar, hasta un máximo de tres (este número se puede variar).

Funciones relacionadas con la transmisión de los datos:

- Función **performContention**. Implementa la contención en la transmisión, es decir, escucha para ver si el canal está libre para transmitir. Si está libre, transmite, y, si está ocupado, aumenta CW al doble de su tamaño y vuelve a preguntar. Este proceso se utiliza ya que las motas no tienen en cuenta el 1 % del tiempo para transmitir. De esta forma, utilizando la contención, cumplimos el requisito de transmisión impuesto por la regulación. Para el funcionamiento de este algoritmo

es necesario especificar los parámetros de contención previamente, que son los siguiente:

- lbtPower = -80 dBm, será el umbral a partir del cual se considere el canal libre
- DIFS = 50 ms, periodo de tiempo durante el cual se escuchará el medio
- slotTime = 5 ms, ranuras de tiempo en las cuales se dividirá el periodo de backoff, el cual tendrá la finalidad de evitar las colisiones de transmisiónes simultáneas
- cwMin = 16
- cwMax = 128

El funcionamiento del algoritmo es el siguiente:

- Se escucha el medio durante un tiempo DIFS.
- Si el medio está libre, se inicia el periodo de backoff (tiempo de cesión del medio), que será un tiempo aleatorio entre 0 y CW - 1*SlotTime en el que la mota sigue escuchando el medio. Si tras estos tiempos el medio sigue libre, la mota transmitirá un paquete.
- Al tamaño de la ventana de contención CW se le asigna el valor inicial de CWmin. Este valor aumentará cada vez que haya un intento de transmisión fallido.
- El nuevo valor de CW cada vez que se escucha al canal y esta ocupado será el doble del valor anterior y podrá tener como valor máximo CWmax.
- Después de cada transmisión exitosa el valor de CW se restablece a CWmin.

- Función **sendChunks**. Recibe como entrada el destino, el número de transacción, el primer y último byte a enviar de cada paquete, la variable de estado que indica si se tiene en cuenta la contención y la CW. Esta es la más importante ya que se encarga de enviar los distintos fragmentos de la imagen uno por uno. Va calculando el primer y último byte de cada paquete a enviar gracias a los parámetros de entrada firstByteChunksToBeSent y lastByteChunksToBeSent. A continuación, en el caso de que la variable bContention esté a True, se llama a la función que implementa el mecanismo de contención. Por último, crea la cabecera gracias a la función mencionada anteriormente, la concatena con los datos correspondientes al paquete actual y los envía.

En primer lugar, calcula el número de paquetes en los que se divide la imagen dividiendo número de bytes de la imagen (lastByteChunksToBeSent - firstByteChunksToBeSent) entre el tamaño del paquete. El primer paquete a enviar sería aquel que va desde el primer byte de la imagen hasta el byte número "packetSize". Este primer paquete se concatena con la cabecera y se envía (utilizando contención o no, según lo especificado en la variable bPerformContention). Cuando se haya enviado el primer paquete, se calcula el segundo paquete, sumando packetSize a la variable beginByte y endByte.

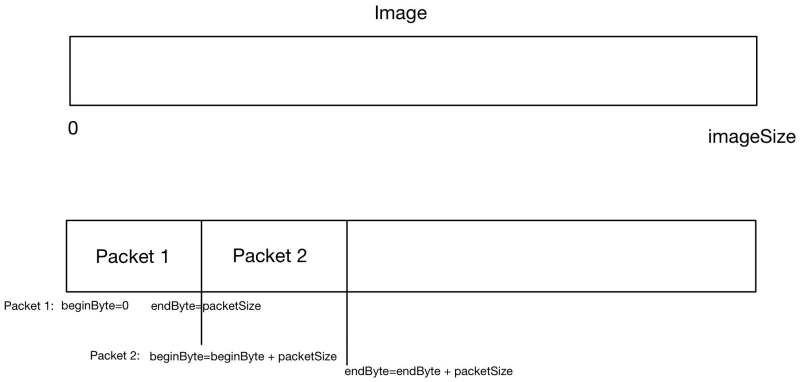

Figura 5.5: Diagrama sendChunks.

Y por último, la función **main**. En esta se establecen los distintos parámetros de transmisión LoRa y se llama a todas las funciones mencionadas anteriormente siguiendo la estructura mostrada en el diagrama de estados (figura 5.4).

5.2.5. Mota Receptora

Por otra parte, el script de la mota receptora sigue la estructura de la imagen 5.6:

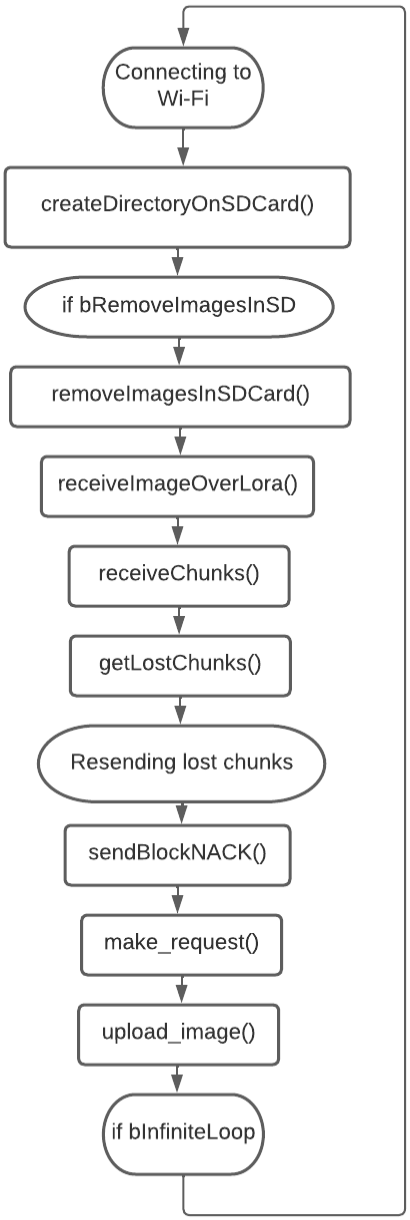

Figura 5.6: Diagrama de estados mota receptora.

Al igual que en el receptor, tendremos funciones relacionadas con las cabeceras, tanto para la creación de estas como para la extracción de los distintos campos que contiene, lo cual nos permitirá identificar la fuente, el destino, el identificador del paquete y otros parámetros.

Funciones relacionadas con la señalización:

- Función **waitForImage**. Esta función se encarga, en primer lugar, de recibir el tamaño total de la imagen, para conocer el número de fragmentos a recibir. La otra tarea que realiza se trata de comprobar que la fuente receptora de la imagen coincide con el destino de la mota emisora, es decir, extrae el destino de la cabecera que acompaña al mensaje donde se envía el tamaño de la imagen, y comprueba que esta se trate de la fuente receptora, es decir, de ella misma.
- Función **sendBlockNACK**. Se encarga de enviar el mensaje **#BLOCK-NACK#**, esto sucede cuando la mota receptora termina de recibir todos los paquetes, o bien lleva sin recibir un paquete más de 5 segundos, el mensaje contiene el número de paquetes perdidos con su identificación, hasta un máximo de tres paquetes.

Funciones relacionadas con la recepción de datos:

- Función **receiveChunks**. Esta se trata de la función principal en la mota receptora, se encarga de recibir los paquetes uno por uno. Para ello, va comprobando que la mota emisora es la deseada (en el caso de que hubiese varias), verificando que el origen coincide con el destino de la mota receptora, y, además de esto, también se verifica que los números de transacción son iguales. Una vez se cumplan estas dos condiciones, se irán concatenando los datos en un vector, con el que se obtendrá la imagen completa.
- Función **getLostChunks**. Función que se encarga de obtener los fragmentos de la imagen que se han perdido en la recepción. Para ello, va recorriendo la variable donde se almacenó la imagen y va comprobando uno por uno si se encuentran todos los paquetes presentes. En caso de que no estén, se almacena en una variable el primer byte del paquete que se ha perdido, y en otra variable el último byte de dicho paquete.

Por último, la función **main** tiene el mismo propósito que en la mota receptora. En primer lugar, comprueba que existe un directorio en el que ir almacenando las imágenes recibidas e inicializa la tarjeta SD para hacer posible este almacenamiento. Como bloque principal, podemos destacar un bucle infinito que se encarga de almacenar la imagen en la tarjeta SD. Este espera a recibir el mensaje de señalización proveniente de la mota emisora, en el que se indica el tamaño de la imagen a recibir a continuación. Una vez lo recibe, se reciben en primer lugar los fragmentos de la imagen y en el momento en que se recibe el último, le siguen las retransmisiones de los paquetes perdidos.

5.2.6. Visualización del contenido multimedia y página web

Para concluir con el diseño a nivel de software, crearemos un servidor local Apache que nos permitirá visualizar todas las imágenes tomadas por

la cámara. La mota receptora enviará una petición POST con el contenido de la imagen al servidor via WiFi. Para ello, harán falta varios archivos php de los cuales cada uno desempeñará una función distinta.

En nuestro servidor hacemos uso de dos ficheros php. El primero de ellos, denominado *'upload.php'*, que tendrá la función de recibir las peticiones POST y subir las imágenes al servidor web. El otro fichero, *'gallery.php'*, nos permitirá acceder a todas aquellas imágenes que ya se encuentran almacenadas en el servidor, proporcionando la fecha y hora de subida. Además de esto, también permite eliminar las imágenes de la base de datos del servidor.

Sumado a estos dos ficheros mencionados, se hará uso de otro archivo php denominado *'streaming.php'*. Este se encargará de, utilizando un filtro para quedarse con la imagen tomada más reciente, mostrar por pantalla dicha imagen. El filtro mencionado se actualizará cada cierto tiempo (1 segundo por defecto), comprobando en cada iteración si hay una imagen más reciente que la actual. Cabe destacar que este fichero hace uso del Ajax/jQuery, el cual permite actualizar el contenido de la página web sin necesidad de recargar la página entera.

Una vez implementado lo mencionado en los párrafos anteriores, cada vez que la mota emisora tome una foto y la envíe, se mostrará en nuestro servidor web. Este servidor irá actualizándose periódicamente, comprobando a su vez si hay una imagen más reciente para mostrar.

Capítulo 6

Análisis de resultados

En este apartado se pondrá en funcionamiento el proyecto para comprobar que todo lo diseñado funciona correctamente, y para mostrar y analizar los resultados obtenidos. En primer lugar, se harán las pruebas con la imagen de un limón que ya esta almacenada en la tarjeta sd de la mota. Se concluirá haciendo esto mismo pero utilizando el módulo de la cámara.

6.1. Puesta en marcha y conexión de dispositivos

El primer paso de todos será conectar la cámara a la mota emisora. Una vez hecho esto, conectaremos ambas motas al ordenador, e iniciamos Atom, que, como se comenta en el capítulo cuarto, será el programa que nos proporcione la interfaz para controlar las motas FiPy. El escenario de trabajo se muestra en la figura 6.1.

Figura 6.1: Escenario de trabajo.

Una vez iniciado Atom, tendremos un proyecto correspondiente a la mota emisora y otro a la mota receptora. El primer proyecto a ejecutar debe ser el de la mota receptora, ya que antes de ejecutar el de la emisora, la receptora debe estar esperando para recibir datos. Estos datos serán los fragmentos de la primera imagen. A la hora de transmitir tenemos dos opciones permitidas por el código elaborado, o bien transmitir una imagen simple, o bien transmitir de forma indefinida. Sumado a esto, se nos permite transmitir utilizando el algoritmo LBT comentado en capítulos anteriores, o bien no teniéndolo en consideración, y esto se controlará mediante una variable booleana.

La apariencia del servidor web antes de recibir ninguna imagen es la mostrada en la figura 6.2:

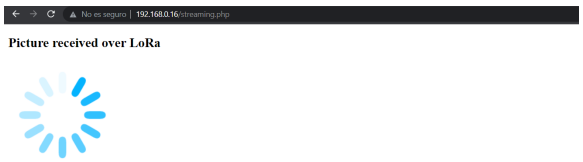

Figura 6.2: Página web esperando para recibir imagen.

Y las comunicaciones proporcionadas por cada una de las dos motas en la transmisión de una imagen simple sin utilizar el mecanismo LBT (contención) se muestran a continuación.

Para la mota emisora:

```
(c) Jorge Navarro and Fernando Tejero, 2022

----------------------------------
Parameters for image transmission:
----------------------------------
Infinite loop: False
Packet size: 200 bytes
Source address: 111
Destination address: 222
Time between pictures: 0.0 s

-----------------------------------
```

```
13 [INFO] Processing image number 1
14 -----------------------------------
15 [INFO] Sending image to destination 222
16 [INFO] LoRa initialized!
17 Content of SD card: ['test.OV2640_1024x768_JPEG.jpg', 'System Volume
       Information', 'images']
18 Content of images directory on SD card: ['lemons-10kb.jpg', 'test.
       OV2640_1024x768_JPEG.jpg']
19 Size of /sd/images/lemons-10kb.jpg: 10470 bytes
20 [INFO] Signaling message "#IMAGE# 10470" with transaction ID 37715
       sent!
21 [INFO] 10 packets sent
22 [INFO] 20 packets sent
23 [INFO] 30 packets sent
24 [INFO] 40 packets sent
25 [INFO] 50 packets sent
26 [INFO] Number of packets sent (image or retransmissions): 53
27 [INFO] Waiting for signaling message #BLOCKNACK#
28 [INFO] Signaling message received: #BLOCKNACK# 0 199 400 599 2600 2799
        4800 4999 7000 7199 9200 9399
29 [INFO] Signaling message #BLOCKNACK# received (transaction ID 37715),
       6 chunks to be retransmitted
30 [INFO] firstByteChunksToBeSent: [0, 400, 2600, 4800, 7000, 9200]
31 [INFO] lastByteChunksToBeSent:  [199, 599, 2799, 4999, 7199, 9399]
32 [INFO] Number of packets sent (image or retransmissions): 6
33 [INFO] Waiting for signaling message #BLOCKNACK#
34 [INFO] Signaling message received: #BLOCKNACK# 400 599
35 [INFO] Signaling message #BLOCKNACK# received (transaction ID 37715),
       1 chunks to be retransmitted
36 [INFO] firstByteChunksToBeSent: [400]
37 [INFO] lastByteChunksToBeSent:  [599]
38 [INFO] Number of packets sent (image or retransmissions): 1
39 [INFO] Waiting for signaling message #BLOCKNACK#
40 [INFO] Signaling message received: #BLOCKNACK#
41 [INFO] #BLOCKNACK# message received (transaction ID 37715), all
       packets were correctly received
42 [INFO] Image has been sent!
43 [INFO] Time spent sending picture: 7.226812 s
44 [INFO] Time between pictures: 0.0 s
45 [INFO] Starting to take a new picture immediately (after 1.0 s)!
46 Pycom MicroPython 1.20.2.rc11 [v1.20.1.r2-306-gd574024b7] on
       2020-09-27; FiPy with ESP32
```

Los distintos sucesos que tienen lugar en la mota emisora son:

- En primer lugar se inicializa LoRa con sus correspondientes parámetros de transmisión. Además, se muestra el contenido actual de la tarjeta sd así como el tamaño de la imagen a enviar (líneas 13-19).

- Se envía el mensaje de señalización, que contiene el ID de la imagen y su tamaño. Seguido a este mensaje de señalización se envían todos los paquetes que componen la imagen y se indica de cuantos se tratan (líneas 20-26).

- Una vez se han enviado todos los paquetes, la mota espera la recepción del mensaje BLOCKNACK, que indicará que la imagen se ha enviado con éxito, o bien los paquetes perdidos en caso de que los haya.

A continuación, se retransmitirán todos los paquetes que indica este mensaje de señalización hasta que nos indique que la imagen se ha enviado entera correctamente (líneas 27-42).

- Por último, se indica el tiempo empleado en la transmisión (línea 43).

Y para la mota receptora:

```
1  (c) Jorge Navarro and Fernando Tejero, 2022
2
3  ------------------------------
4  Parameters for image reception:
5  ------------------------------
6  Infinite loop: False
7  Source address: 222
8  Destination address: 111
9  Max. no. retransmissions: 3
10 Packet reception timeout: 5 s
11 Save to SD card: True
12 Remove previous images from SD card: True
13
14 [INFO] Trying to connect to the Wi-Fi network sagemcomED80...
15 [INFO] Wi-Fi connected succesfully
16 [INFO] Wi-Fi configuration: ('192.168.0.24', '255.255.255.0',
       '192.168.0.1', '212.166.132.118')
17 [INFO] LoRa initialized!
18 [INFO] /sd/images content: ['image00001.jpg']
19 [INFO] Removing images in the SD card...
20
21 ------------------------------------------------
22 [INFO] Next image name: /sd/images/image00001.jpg
23 [INFO] Processing image number 1
24 ------------------------------------------------
25 [INFO] Receiving image (to address 222)
26 [INFO] #IMAGE# message received (transaction ID 37715), image size
       10470 bytes
27 [INFO] First reception of the image 1
28 [INFO] Received packet 0
29 [INFO] Received packet 10
30 [INFO] Received packet 20
31 [INFO] Received packet 30
32 [INFO] Received packet 40
33 [INFO] Some LoRa packets were lost (bytes [0-199, 400-599, 2600-2799,
       4800-4999, 7000-7199, 9200-9399])
34 [INFO] Signaling message "#BLOCKNACK# 0 199 400 599 2600 2799 4800
       4999 7000 7199 9200 9399" sent!
35 [INFO] Retransmission 1 of LoRa packets
36 [INFO] Received packet 0
37 [INFO] Some LoRa packets were lost (bytes [400-599])
38 [INFO] Signaling message "#BLOCKNACK# 400 599" sent!
39 [INFO] Retransmission 2 of LoRa packets
40 [INFO] Received packet 0
41 [INFO] No LoRa packet was lost
42 [INFO] Signaling message "#BLOCKNACK#" sent!
43 [INFO] Image received! Time spent receiving picture (53 packets, after
        initial signaling message): 7.184331 s
44 [INFO] Sending image (10470 bytes) to web server (http://192.168.0.16/
       upload.php)...
45 [INFO] Uploaded image (HTTP POST request)
```

Los sucesos de los que se informa son los siguientes:

- En primer lugar la mota se conecta a la red Wi-Fi (líneas 14-16).
- Se inicializa LoRa, se muestra el contenido de la tarjeta SD, y se borra su contenido si la variable booleana correspondiente lo indica (líneas 17-19).
- Se asigna un nombre a la imagen a recibir (líneas 22-23).
- Se recibe el ID de la imagen, su tamaño y todos los fragmentos que la componen (líneas 25-32).
- Se indican qué paquetes se han perdido, y se envía el mensaje BLOCKNACK indicándolos para poder retransmitirlos (líneas 33-42).
- Una vez se ha recibido la imagen, se indica el número de paquetes que la componen, el tiempo empleado en transmitirla y la dirección del servidor hacia el que se ha enviado, así como un mensaje indicando que la petición POST ha sido exitosa (líneas 43-45).

En las figuras 6.3 y 6.4 se muestram capturas de pantalla en las que aparecen ambos proyectos en ejecución.

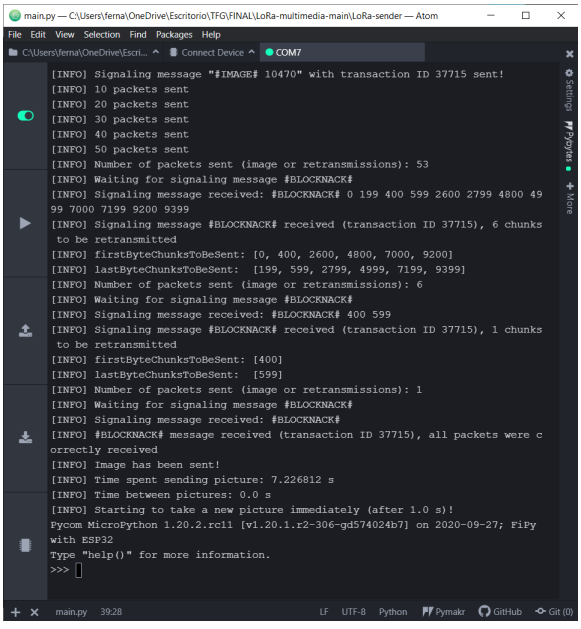

Figura 6.3: Captura comunicaciones mota emisora.

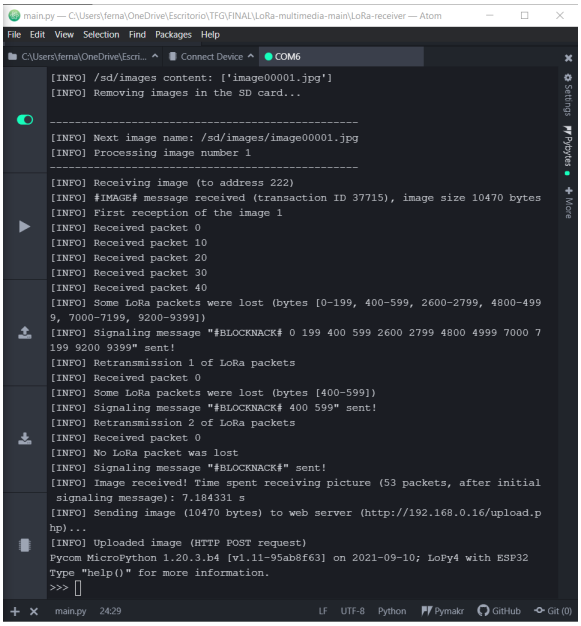

Figura 6.4: Captura comunicaciones mota receptora.

El tiempo empleado en enviar una imagen simple es de 7.1 segundos, esto se debe a que no se está usando LBT, y las motas envían los datos sin tiempo de espera entre paquetes.

Ahora seguiremos el mismo proceso pero utilizando LBT.

Las comunicaciones tanto en la mota emisora como en la receptora serán idénticas al caso anterior, el único parámetro que variará será el tiempo de envio, que se presentará algo mayor debido al LBT.

Las comunicaciones de la mota emisora son:

```
(c) Jorge Navarro and Fernando Tejero, 2022

----------------------------------
Parameters for image transmission:
----------------------------------
Infinite loop: False
Packet size: 200 bytes
Source address: 111
Destination address: 222
Time between pictures: 0.0 s

-----------------------------------
[INFO] Processing image number 1
-----------------------------------
[INFO] Sending image to destination 222
[INFO] LoRa initialized!
```

```
Content of SD card: ['test.OV2640_1024x768_JPEG.jpg', 'System Volume Information', 'images']
Content of images directory on SD card: ['lemons-10kb.jpg', 'test.OV2640_1024x768_JPEG.jpg']
Size of /sd/images/lemons-10kb.jpg: 10470 bytes
[INFO] Signaling message "#IMAGE# 10470" with transaction ID 9339 sent!
[INFO] 10 packets sent
[INFO] 20 packets sent
[INFO] Channel is busy! Listen again! (CW=32)
[INFO] 30 packets sent
[INFO] Channel is busy! Listen again! (CW=32)
[INFO] Channel is busy! Listen again! (CW=64)
[INFO] Channel is busy! Listen again! (CW=128)
[INFO] Channel is busy! Listen again! (CW=128)
[INFO] Channel is busy! Listen again! (CW=128)
[INFO] Channel is busy! Listen again! (CW=128)
[INFO] Channel is busy! Listen again! (CW=128)
[INFO] 40 packets sent
[INFO] 50 packets sent
[INFO] Number of packets sent (image or retransmissions): 53
[INFO] Waiting for signaling message #BLOCKNACK#
[INFO] Signaling message received: #BLOCKNACK#
[INFO] #BLOCKNACK# message received (transaction ID 9339), all packets were correctly received
[INFO] Image has been sent!
[INFO] Time spent sending picture: 10.52706 s
[INFO] Time between pictures: 0.0 s
[INFO] Starting to take a new picture immediately (after 1.0 s)!
```

No hay diferencias en las comunicaciones con respecto al caso anterior. La única diferencia es que se emplea un poco más de tiempo en la transmisión debido al empleo de la contención. Cabe destacar que en este ejemplo concreto no se pierde ningún paquete.

Las de la mota receptora:

```
(c) Jorge Navarro and Fernando Tejero, 2022

--------------------------------
Parameters for image reception:
--------------------------------
Infinite loop: False
Source address: 222
Destination address: 111
Max. no. retransmissions: 3
Packet reception timeout: 5 s
Save to SD card: True
Remove previous images from SD card: True

[INFO] Trying to connect to the Wi-Fi network sagemcomED80...
[INFO] Wi-Fi connected succesfully
[INFO] Wi-Fi configuration: ('192.168.0.24', '255.255.255.0', '192.168.0.1', '212.166.132.118')
[INFO] LoRa initialized!
[INFO] /sd/images content: ['image00001.jpg', 'image00002.jpg', 'image00003.jpg']
[INFO] Removing images in the SD card...

------------------------------------------------
```

```
22 [INFO] Next image name: /sd/images/image00001.jpg
23 [INFO] Processing image number 1
24 ------------------------------------------------
25 [INFO] Receiving image (to address 222)
26 [INFO] #IMAGE# message received (transaction ID 9339), image size
       10470 bytes
27 [INFO] First reception of the image 1
28 [INFO] Received packet 0
29 [INFO] Received packet 10
30 [INFO] Received packet 20
31 [INFO] Received packet 30
32 [INFO] Received packet 40
33 [INFO] Received packet 50
34 [INFO] No LoRa packet was lost
35 [INFO] Signaling message "#BLOCKNACK#" sent!
36 [INFO] Image received! Time spent receiving picture (53 packets, after
       initial signaling message): 10.48438 s
37 [INFO] Sending image (10470 bytes) to web server (http://192.168.0.16/
       upload.php)...
38 [INFO] Uploaded image (HTTP POST request)
```

Sucede lo mismo que para el caso en el que no se utiliza contención, la única diferencia es el aumento del tiempo de transmisión.

Las imágenes 6.5 y 6.6 muestran ambos proyectos en ejecución.

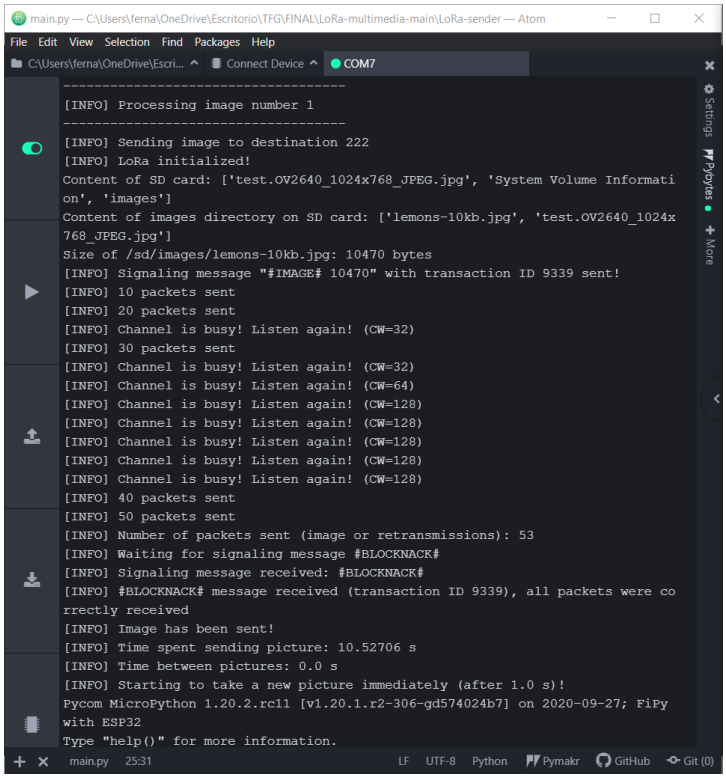

Figura 6.5: Captura comunicaciones mota emisora.

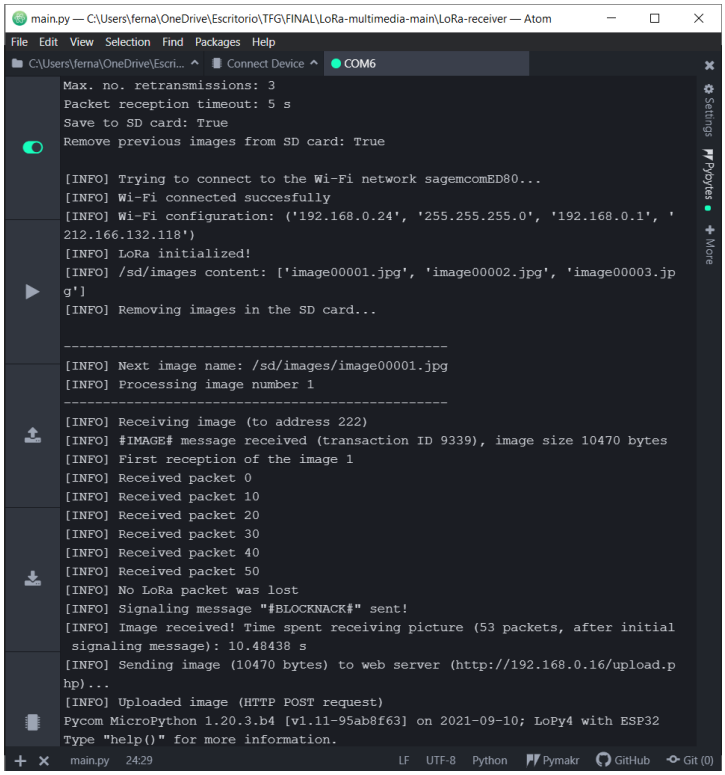

Figura 6.6: Captura comunicaciones mota emisora.

Si nos fijamos en el tiempo empleado en enviar la imagen en estos dos supuestos, vemos cómo la transmisión es algo más rápida si no se aplica contención. Esto se debe a que no tenemos como requisito escuchar el medio cada vez que se vayan a transmitir datos para comprobar si ya hay una transmisión en curso.

Una vez la mota receptora nos indique que la imagen se ha recibido con éxito, podremos acceder a "streaming.php", mostrado en la figura 6.7, para observar el contenido recibido por el servidor web.

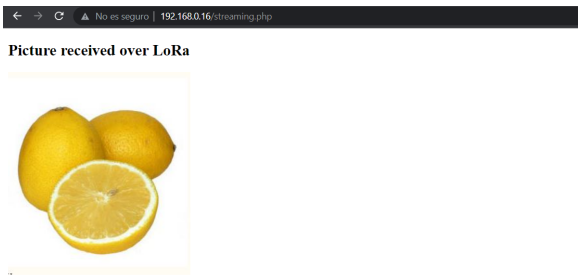

Figura 6.7: Contenido de streaming.php.

Y además, comprobamos que se almacena en "gallery.php", en la figura 6.8.

Figura 6.8: Contenido de gallery.php.

La siguiente prueba será aquella en la que enviaremos varias imágenes, dejando a la mota transmisora y receptora en un bucle infinito (bInfiniteLoop = True). Si observamos la imagen 6.9, se mandan un total de 23 imágenes. Comprobamos que estan se almacenan en la figura 6.10.

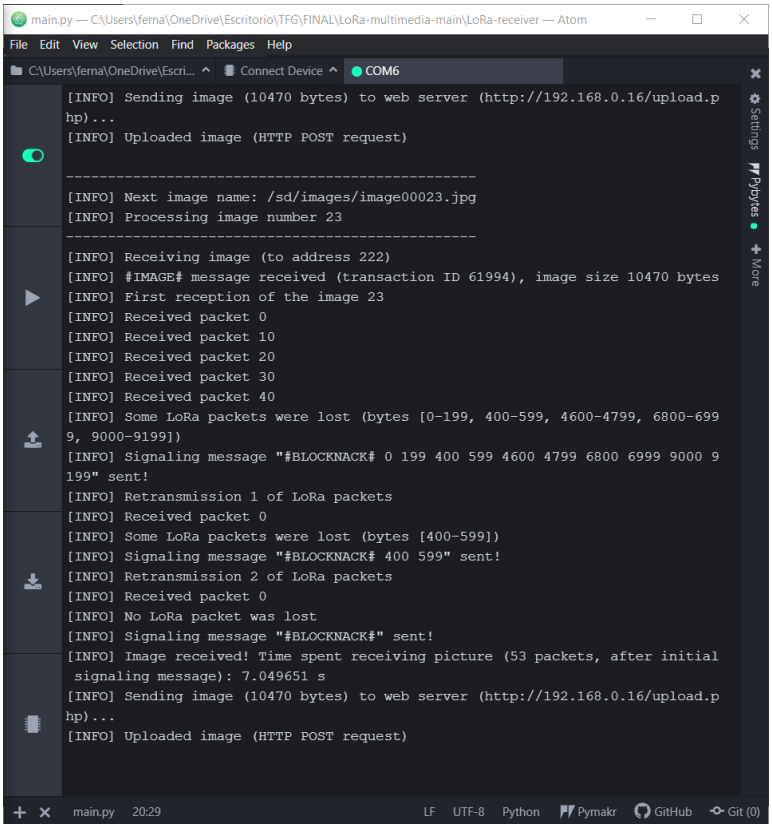

Figura 6.9: Comunicaciones mota receptora en bucle infinito.

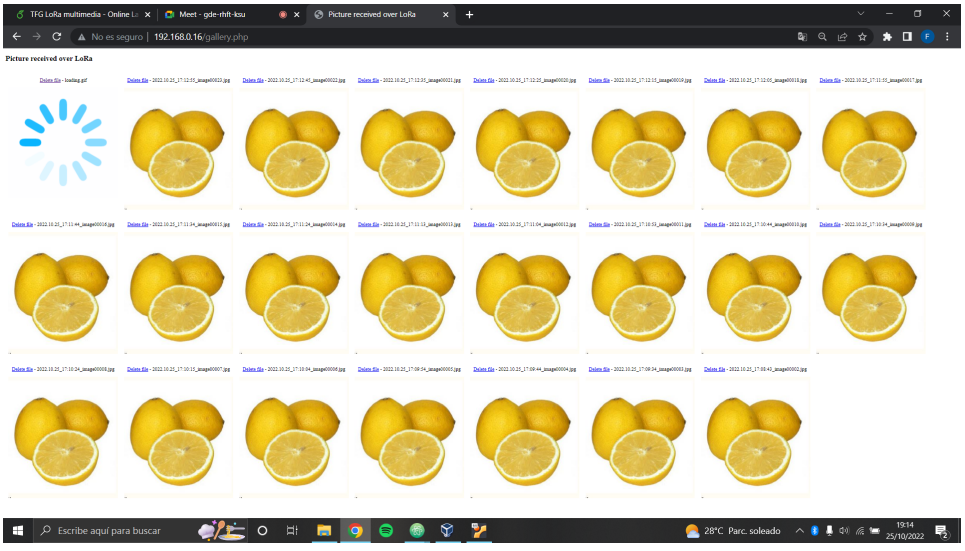

Figura 6.10: Contenido de gallery.php.

En la última prueba realizada se mostrará el funcionamiento del módulo de la cámara utilizado(figura 6.11):

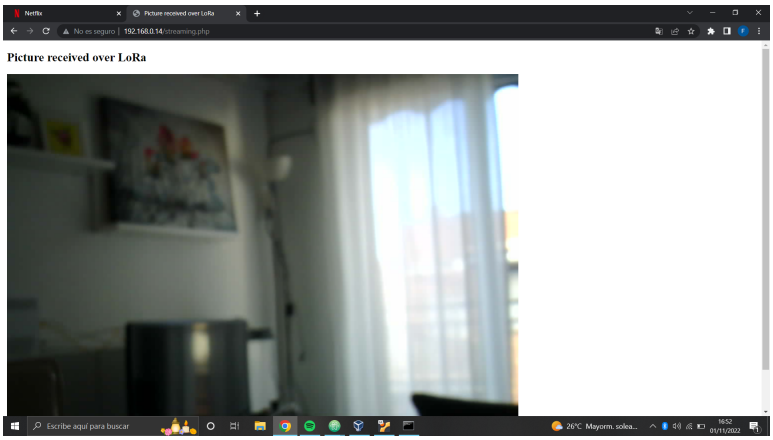

Figura 6.11: Imagen tomada con la cámara

Y el contenido de *gallery.php* con varias imágenes tomadas en el mismo lugar (figura 6.12):

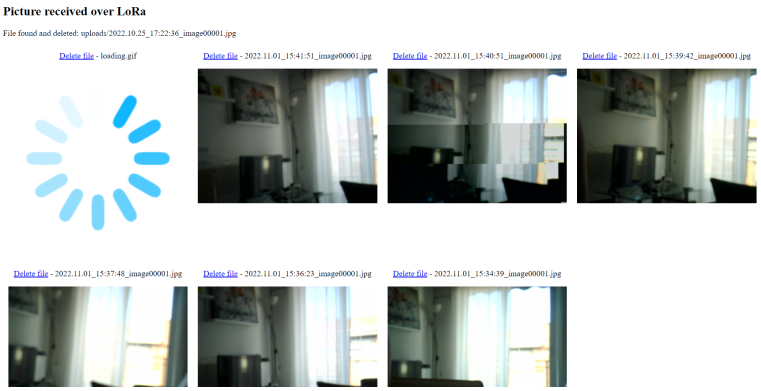

Figura 6.12: Contenido de gallery.php.

Capítulo 7

Conclusiones y líneas futuras

En este último capítulo se recogen las conclusiones extraidas al finalizar el proyecto, así como los objetivos conseguidos. Además, se hablará sobre las líneas futuras que pueda seguir el proyecto para mejorar sus funciones e implementación.

7.1. Conclusiones

El propósito de este proyecto consistía en diseñar un sistema de transmisión multimedia de bajo consumo.

El propósito se ha conseguido utilizando dos motas FiPy, con requisitos de potencia muy bajos, y por tanto, bajo consumo. Las motas mencionadas han funcionado como transmisora-receptora, y se han comunicado a través de la tecnología LoRa, cosa que se traduce en un largo alcance y una baja potencia/consumo.

Para ello, en primer lugar se estudiaron las características y el funcionamiento de LoRa. Una vez recopilada suficiente información sobre LoRa, se dispuso a elaborar dos ficheros de configuración para cada mota, emisora y receptora. En el fichero de la mota emisora se configura el módulo de la cámara de fotos, se establecen los distintos parámetros de transmisión de LoRa y se envía la imagen. En el de la mota receptora se recibe la imagen, se reconstruye, y una vez reconstruida se manda una petición POST a un servidor web con dicha imagen. Este servidor se encargará de recibir las peticiones POST con las imágenes para mostrarlas por pantalla en tiempo real.

Una vez diseñado e implementado el sistema propuesto, el último paso será comprobar que todo funciona de la manera deseada, y se pasará a la extracción de resultados y recopilación de conclusiones.

En cuanto a las ventajas que puede presentar este proyecto, son las siguientes:

- Nuestra solución está basada en el uso de un mecanismo de escucha

del medio. En nuestro caso el algoritmo LBT (Listen Before Talk), esto nos permite obviar la limitación del 1 % del ciclo de trabajo, lo que se traduce en una mejorar de la velocidad de transmisión.

- Al tener el código desarrollado al completo, se puede modificar en cualquier momento para incluir nuevas funciones u optimizaciones.
- Otra ventaja importante a comentar es la presencia en algunos entornos de motas LoRa, por ejemplo para realizar mediciones en zonas agrícolas o rurales. Esto permite enviar imágenes cada pocos segundos a cambio de un muy bajo consumo haciendo uso de un hardware que ya está disponible. Esto nos permitiría evitar añadir otras tecnologías inalámbricas como podría ser Wi-Fi, 4G/5G... que incluso podrían no tener cobertura en estos entornos mencionados.

7.2. Líneas Futuras

En este proyecto se ha conseguido la mejora de la velocidad en la transmisión de imágenes usando LoRa como medio de transmisión. Sin embargo, encontramos algunas mejoras o alternativas a esta solución:

- En lugar de utilizar sólo LoRa, conseguir que esta solución funcione en una red LoRaWAN. Esto permitiría usar redes LoRaWAN ya existentes.
- Implementar medidas de seguridad al transmitir los datos. En la presente solución los datos son enviados sin encriptar. Se podría implementar algún mecanismo de cifrado para garantizar autenticación y la integridad de los datos.
- Evitar el uso del almacenamiento en la tarjeta SD, tanto antes de enviarla a la mota receptora, como antes de mandarla en la petición POST.

Bibliografía

[1] IoT Analytics, *Number of connected IoT devices in 2025.* Disponible en:`https://iot-analytics.com/number-connected-iot-devices/` (último acceso el 5/03/22)

[2] LoRa and LoRaWAN: Technical overview. Disponible en:`https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/` (último acceso el 10/03/22)

[3] LoRa Alliance, "LoRaWAN® Specification v1.1". Disponible en: `https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/`(último acceso el 15/03/22)

[4] LoRa applications. Disponible en: `https://www.semtech.com/lora/lora-applications` (último acceso el 17/03/22)

[5] T. Chen, D. Eager and D. Makaroff,"Efficient Image Transmission Using LoRa Technology In Agricultural Monitoring IoT Systems", 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2019, pp. 937-944, doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00166.

[6] C. Pham,"Robust CSMA for long-range LoRa transmissions with image sensing devices", 2018 Wireless Days (WD), 2018, pp. 116-122, doi: 10.1109/WD.2018.8361706.

[7] C. C. Wei, S. T. Chen and P. Y. Su,"Image Transmission Using LoRa Technology with Various Spreading Factors", 2019 2nd World Symposium on Communication Engineering (WSCE), 2019, pp. 48-52, doi: 10.1109/WSCE49000.2019.9041044.

[8] Cheong, Phui San, Bergs, Johan Hawinkel, Chris Famaey, Jeroen. (2017). Comparison of LoRaWAN Classes and their Power Consumption. 10.1109/SCVT.2017.8240313.

[9] Semtech Corporation, AN1200.22 "LoRa™ Modulation Basics", 2015. Disponible en: https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf (último acceso el 15/03/22)

[10] LoRa Spreading Factor Disponible en: https://www.sciencedirect.com/topics/engineering/spreading-factor (último acceso el 17/03/22)

[11] Z. Wang, Z. Jiang, J. Hu, T. Song and Z. Cao, "Research on Agricultural Environment Information Collection System Based on LoRa,"2018 IEEE 4th International Conference on Computer and Communications (ICCC), 2018, pp. 2441-2445, doi: 10.1109/CompComm.2018.8780762.

[12] K. Kamonkusonman and R. Silapunt,"Channel Activity Detection with the Modified Backoff Algorithm for LoRaWAN", 2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2021, pp. 86-89, doi: 10.1109/ECTI-CON51831.2021.9454714.

[13] Especificaciones ordenador portatil HP Pavilion Power 15-cb009ns. Disponible en: https://support.hp.com/es-es/document/c05530815 (último acceso el 08/09/22)

[14] Especificaciones Pycom FiPy 1.0. Disponible en: https://pycom.io/wp-content/uploads/2018/08/fipySpecsheetAugust2017n2-1.pdf (último acceso el 08/09/22)

[15] Especificaciones Pycom Expansion Board 3.0. Disponible en: https://pycom.io/product/expansion-board-3-0/ (último acceso el 09/09/22)

[16] Mini módulo cámara OV2640 2MP. Disponible en: https://www.kubii.es/camaras-sensores/2326-mini-modulo-camara-shield-ov2640-2mp-kubii-3272496012479.html (último acceso el 09/09/22)

[17] Antena LoRa para placa Pycom FiPy. Disponible en: https://pycom.io/product/lora-868mhz-915mhz-sigfox-antenna-kit/ (último acceso el 09/09/22)

[18] Overleaf. Disponible en: https://es.overleaf.com/ (último acceso el 10/09/22)

[19] Atom. Disponible en: https://atom.io/ (último acceso el 10/09/22)

[20] Python. Disponible en: https://www.python.org/downloads/ (último acceso el 10/09/22)

[21] Gantt Project. Disponible en: `https://www.ganttproject.biz/` (último acceso el 10/09/22)

[22] Oracle VM VirtualBox. Disponible en: `https://www.virtualbox.org/` (último acceso el 10/09/22)

[23] MicroPython-OV2640-Pycom Library `https://github.com/grahamPatico/micropython-ov2640-Pycom` (último acceso el 28/04/22)

[24] FiPy 1.0 Pinout. Disponible en: `https://pycom.io/wp-content/uploads/2018/03/Pycom_002_Specsheets_FiPy_v2.pdf` (último acceso el 28/04/22)

[25] Disponible en: `https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/` (último acceso el 29/04/22)