Universitat Oberta de Catalunya

uoc.edu

Sistema de monitorización del estado de un contenedor de reciclaje

Daniel Alonso Rodrigo

Máster en Ingeniería de Telecomunicación

Área de electrónica

Nombre Consultor/a: Xavier Saura Mas

Nombre Profesor/a responsable de la asignatura: Carlos Monzo Sánchez

Junio 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	Sistema de monitorización del estado de los contenedores de reciclaje
Nombre del autor:	*Daniel Alonso Rodrigo*
Nombre del consultor/a:	*Xavier Saura Mas*
Nombre del PRA:	*Carlos Monzo Sánchez*
Fecha de entrega (mm/aaaa):	06/2020
Titulación::	*Máster en Ingeniería de Telecomunicación*
Área del Trabajo Final:	*Electrónica*
Idioma del trabajo:	*Castellano*
Palabras clave	*Monitorización contenedor, residuos urbanos, optimización ruta*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El presente proyecto tiene por objetivo, la monitorización del volumen de residuos urbanos y el nivel de dióxido de carbono (CO_2), en un contenedor de reciclaje.

A día de hoy, los problemas medioambientales son bien conocidos, por lo que la realización de proyectos con el fin de reducir la contaminación atmosférica, entre otros, están a la orden del día.

La monitorización de un contenedor de reciclaje permite saber el estado en el que se encuentra y por tanto, es posible valorar y decidir, de una manera objetiva, si debe ser o no, recogido durante la ruta de los camiones de reciclaje. Gracias a esta monitorización, es posible optimizar las rutas de recogida, puesto que así, no se recogen contenedores vacíos, con el consiguiente gasto de combustible y tiempo.

Para conocer el nivel de llenado de un contenedor, se utilizan sensores ultrasónicos, cuyo principio de funcionamiento es a través de ondas ultrasónicas. Dichas ondas son enviadas y recibidas por el propio sensor, calculando la distancia mediante el tiempo de transmisión-recepción.

Por otro lado, el nivel de dióxido de carbono (CO_2) se detecta mediante sensores electroquímicos, en los que su resistencia interna varía en función del nivel de concentración de gas.

Por último, los datos adquiridos en cada contenedor, son almacenados en una base de datos, para más tarde, ser procesados por una aplicación, cuyo objetivo es la generación de una lista con los contenedores que deben ser incluidos en la ruta de recogida.

Abstract (in English, 250 words or less):

The target of this project is to monitor the volume and the carbon dioxide (CO_2) level of urban waste stored in a recycling container.

Nowadays, environmental problems are known by everyone, so that is the reason of so many projects to reduce air pollution, among others, are being made.

Monitoring of a recycling container allows to know the state it is and, therefore, it is possible to assess and decide, in an objective way, whether or not it should be collected during the route of the recycling trucks. Thanks of it, it is possible to optimize the collection routes, since thus, empty containers are not collected with the consequent waste of fuel and time.

To know the fill level of a container, ultrasonic sensors are used, whose operating principle is through ultrasonic waves. These waves are sent and recieved by the sensor itself, calculating the distance by transmission-reception time.

On the other hand, carbon dioxide (CO_2) level is detected by electrochemical sensors, in which its internal resistance varies depending on the gas concentration level.

Finally, database is used in order to store the data acquired in every container. Later, this data is processed by an application, the objective of which is to generate a list of the containers that must be included in the collection route.

ÍNDICE

Lista de figuras

Lista de Tablas

1. INTRODUCCIÓN

1.1. Contexto y justificación del Trabajo

El presente proyecto tiene por objetivo, la monitorización del estado de los contenedores de residuos urbanos orgánicos, que se pueden encontrar en la vía pública de cualquier ciudad, pueblo o urbanización. Esta monitorización aportará información del estado de los contenedores, que podrá ser utilizada por los servicios municipales correspondientes con el objetivo de una mejor optimización del servicio.

La monitorización se divide en dos partes, la primera consiste en la detección de diferentes niveles en los que se encuentra el contenedor, como es el volumen de residuos, es decir, hasta qué nivel se han acumulado dentro del contenedor y el nivel de dióxido de carbono (CO_2) que es emitido por estos residuos. Esta detección se producirá mediante módulos instalados en cada uno de los contenedores y que contendrán sensores volumétricos y específicos para cada parámetro a detectar. Por tanto, dichos módulos serán los encargados de adquirir los datos de los sensores. Estos módulos, también se encargarán de la posterior transmisión de la información recogida a una base de datos, donde dicha información estará disponible para el servicio municipal correspondiente.
La segunda parte consiste en el control del volumen de residuos, ya que con los datos obtenidos y enviados por los módulos instalados en los contenedores, se valorará si el nivel de residuos ha superado un umbral previamente establecido, por el cual se considera que ese contenedor está o no lleno o si existe un nivel de CO_2 que exceda lo recomendable y por tanto, deba ser incluido en la ruta de recogida.

En este trabajo final, se propone utilizar la combinación de estos dos parámetros comentados (nivel de llenado del contenedor y CO_2), para poder tomar la decisión de incluir o no, un contenedor en la ruta de recogida, puesto que los residuos orgánicos deben ser estudiados más exhaustivamente que el resto (vidrio/plástico/papel), debido a los procesos de descomposición de la materia. Es precisamente esto lo que obliga a realizar una recogida de los residuos urbanos con una frecuencia de recogida fija, ya que no puede depender solo del nivel de llenado del contenedor, a diferencia del resto de tipos de contenedor. Lo que se propone aquí, es poder tener una frecuencia de recogida variable. Por tanto, el conocimiento de la relación de estos dos parámetros, se utilizará para optimizar la ruta de recogida de los residuos urbanos, por parte de la empresa propietaria de la flota de camiones, Además, con todos los datos recogidos, se podrá hacer un estudio de los diferentes parámetros por separado, ya que por ejemplo es interesante saber, el ritmo al que los contenedores se llenan o los niveles de dióxido de carbono emitidos por los residuos orgánicos.

Con la optimización de la ruta de recogida, se conseguirá reducir el tiempo de esta, que conlleva un ahorro de combustible de los vehículos y que por su parte, consigue reducir

un aspecto importante a día de hoy, como es la contaminación ambiental. Además de lo ya mencionado, se producirá una reducción de la contaminación acústica de la zona, que ayuda al descanso de la ciudadanía, puesto que en muchos casos, el horario de recogida es por la noche.

Un punto importante en este tipo de aplicaciones, en las que los dispositivos no están alimentados permanentemente, es el tiempo de vida de las baterías, que nos limita el consumo de energía de las mismas. Por esto, se deberá buscar una solución para, sobre todo, el protocolo de comunicación a utilizar para la transmisión de los datos y también, el método de localización de cada dispositivo, puesto que es necesario saber dónde se encuentra geográficamente el contenedor para realizar la optimización de la ruta de recogida.
Respecto a la adquisición de datos de los sensores volumétricos, esta no supondrá un problema crítico para el consumo de energía, como lo puede ser la comunicación entre el módulo y la base de datos o incluso el sensor de detección de gas. Se valorará qué nivel de prestaciones deberá tener el producto para poder definir especificaciones como, por ejemplo, la velocidad de transmisión, la cobertura del módulo o incluso la frecuencia con la que se adquieren los datos, que determinarán el consumo energético y limitará dichas prestaciones.
Como zona experimental, se utilizará una parte del área de Barcelona ciudad, puesto que es una zona con alta contaminación tanto atmosférica como acústica, debido a la alta densidad de población. Este proyecto está enfocado a la implementación del producto en contenedores de reciclaje orgánicos, pudiendo ser implementado, en el futuro, en otro tipo de contenedores como los de plástico, vidrio o papel/cartón.

1.1.1. Justificación del uso de un sensor de detección de dióxido de carbono

Se ha decidido detectar el valor de concentración del dióxido de carbono, que determinará si un contenedor debe ser recogido o no, puesto que como demuestran algunos estudios, el dióxido de carbono está ligado a la descomposición que sufre un material al oxidarse. Este proceso de descomposición de la materia produce formaciones de olores y emisiones de compuestos orgánicos volátiles que deben ser controlados, manteniéndolos a niveles aceptables para minimizar las quejas de la comunidad y sobre todo y más importante, para evitar que sean una amenaza para la salud, ya que en muchas ocasiones durante la descomposición de la materia se desprenden compuestos sulfurosos y nitrogenados [1].

Por tanto, para conseguir una recogida de los residuos con una frecuencia variable, habrá que asegurar que no se producen los problemas mencionados, ocasionados por la descomposición de la materia y asegurar así, la higiene de la zona.

1.2. Objetivos del Trabajo

Como se ha comentado en el apartado anterior, el objetivo de este proyecto final de máster, es poder implementar un sistema de monitorización del estado de un contenedor de residuos urbanos, para poder realizar una optimización o *'re-routing'* de la ruta de la flota de vehículos de recogida. Esto permitirá reducir los tiempos de ruta, importante para la empresa de recogida de residuos urbanos y por otro lado, se producirá una reducción de la contaminación ambiental, importante para el medioambiente.

Por tanto, como objetivos principales, serán los listados y descritos a continuación:

- Detección del nivel de llenado de residuos en un contenedor de reciclaje mediante un sensor volumétrico.
- Detección de los niveles de dióxido de carbono de un contenedor de reciclaje, a través de sensores específicos para ello.
- Comunicación entre el módulo instalado en el contenedor y la base de datos.
- Identificación y localización de los contenedores de reciclaje.
- Maximizar la autonomía del dispositivo.
- Detección de contenedores a incluir en la ruta de recogida.
- Gestión de una base de datos con la información recogida por los módulos.

1.3. Motivación

Los proyectos con objetivos de mejora medioambiental son, a día de hoy, muy importantes para la sociedad actual, que se encuentra sumida en una situación de alarma por la alta contaminación atmosférica, que afecta a todos y sobre todo, a las zonas con mayor densidad de población como puede ser Barcelona. En lo que a este proyecto se refiere, la gestión de los residuos urbanos juega un papel muy importante en el medioambiente, debido a los gases emitidos por estos residuos y para conseguir un reciclaje óptimo de los diferentes materiales para su uso en aplicaciones futuras. Por tanto, el bueno uso de los contenedores de reciclaje, gracias a una gestión eficiente de estos, ayudará a tener zonas más limpias.

Siempre es una motivación poder aportar algo que sirva para mejorar no solo la vida de los demás, sino, en este caso, también la vida del planeta, puesto que al final, las consecuencias las acabaremos pagando todos.

Dicho esto, también existe una motivación a nivel personal, puesto que es una oportunidad para proponerse a uno mismo un reto a superar, que no se llevaría a cabo si no fuera por la realización del trabajo final. Además, el mero hecho de intentar poder diseñar e implementar una solución tecnológica mejor a la actual, es en sí, una motivación más.

1.4. Planificación del Trabajo

La planificación del proyecto se ha basado en los entregables parciales que define la asignatura del trabajo final de máster, por lo que las fechas de inicio y límite de cada fase han sido las de las diferentes PEC.

A continuación se muestra un diagrama de Gantt con todas las fechas de inicio y final de cada fase.

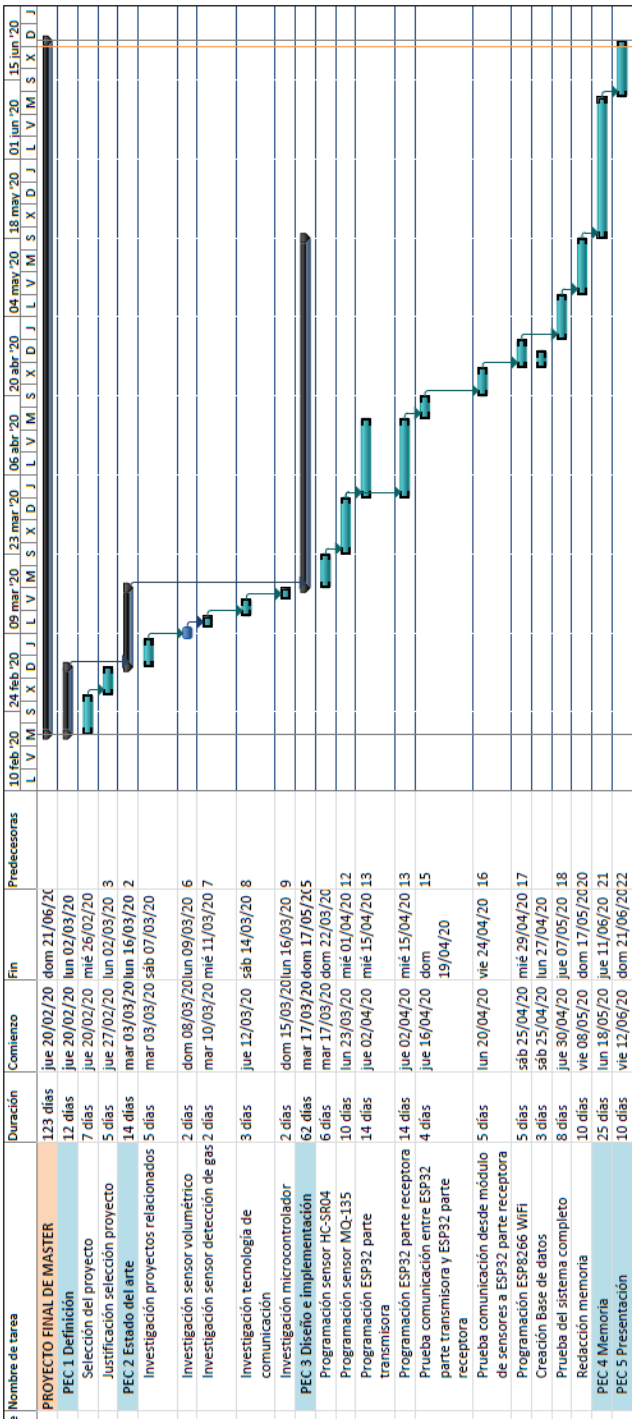


Figura 1. Diagrama de Gantt del proyecto.

1.5. Breve sumario de productos obtenidos

La siguiente figura muestra las diferentes partes de las que se compone el producto a entregar para alcanzar los hitos definidos en este proyecto.

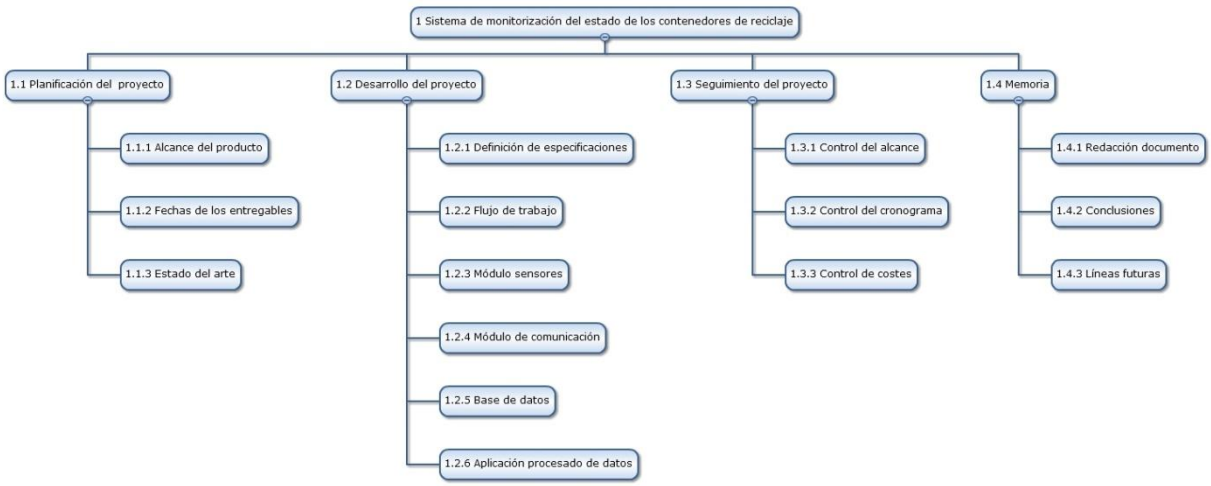


Figura 2. WBS de los entregables del proyecto.

1.6. Breve descripción de los otros capítulos de la memoria

Esta memoria se ha dividido en 9 capítulos, los cuales se listan y se describen brevemente a continuación:

1. Introducción: Se pone en contexto el tema de este proyecto y se explica el por qué de la elección de este proyecto, juntamente con la motivación que ha llevado al autor a realizarlo. También, se muestra la planificación de las fases del proyecto.
2. Estado del arte: Se describen proyectos similares que han inspirado al autor y las diferentes tecnologías existentes en el mercado para llevar a cabo este proyecto.
3. Diseño del proyecto: Contiene todo el diseño del sistema a implementar, a nivel de hardware y software. Incluye los cálculos matemáticos, simulaciones teóricas y partes importantes del código desarrollado.
4. Resultados del proyecto: Se muestran los resultados obtenidos del diseño realizado en el capitulo anterior, de los diferentes módulos que forman el sistema.
5. Conclusiones: Se escriben las conclusiones extraídas después de la realización del proyecto.
6. Versiones futuras: Se describen posibles mejoras en futuros proyectos.
7. Glosario: Se describen diferentes términos utilizados durante la memoria.
8. Referencias: Se listan las referencias utilizadas para la realización del proyecto.

9. Anexo: Se muestra los códigos generados para el módulo de sensores, comunicaciones, base de datos y la aplicación de procesado de datos.

2. ESTADO DEL ARTE

En esta sección se presenta el estado del arte del trabajo final de máster. El estado del arte de este trabajo se divide en varias partes. La primera tratará sobre proyectos relacionados con el concepto de monitorización del estado de contenedores, a través de la detección del nivel de volumen de llenado, temperatura o incluso movimiento. Esta parte servirá para conocer qué proyectos se han realizado con los mismos objetivos y cómo han logrado afrontar y superar la problemática presentada. La segunda parte, se enfocará más en el hardware a utilizar, es decir, los recursos de los que disponemos para poder realizar con garantías y alcanzar los objetivos de este proyecto. Por ejemplo, se buscará qué tipo de sensores utilizar para la monitorización de los diferentes parámetros y valorar cuáles de entre todos los tipos de sensores encontrados, se ajustan mejor a nuestro proyecto. Otro ejemplo es, el protocolo de comunicación a utilizar de entre todos los posibles, que cumpla con los requisitos mínimos.

2.1. Proyectos relacionados

1. BIN-CT: Urban Waste Collection based on Predicting the Container Fill Level

El primero de los proyectos encontrados es BIN-CT, implantado en Algeciras (Cádiz) por la Universidad de Málaga. Este proyecto es un sistema inteligente que prevé el nivel de llenado de los contenedores de recogida selectiva, través de sensores volumétricos, para calcular la ruta más eficiente de los camiones de recogida, a partir de los datos adquiridos diariamente y el histórico almacenado. Así, la empresa puede predecir cuándo deben recoger los residuos y también, tiene en cuenta factores como la estacionalidad, los días de la semana o días festivos [2].

Este proyecto usa la combinación de dos algoritmos, uno predice el nivel de llenado de los contenedores y con el otro, se generan las rutas de recogida eficientes, considerando la capacidad de los camiones, el nivel de llenado de cada uno de los contenedores por separado y las características de la ubicación de estos, entre otros parámetros. Para la visualización de estos datos, se utiliza una aplicación, que muestra las modificaciones de ruta.

El estudio de caso real, fue hecho con la monitorización de 217 contenedores de papel y consiguieron reducir un 20 por ciento la distancia recorrida por los camiones de recogida y un 33.2 por ciento los itinerarios, ya que antes se planificaba a partir de la experiencia humana.

La imagen siguiente muestra las cuatro fases en que este proyecto fue dividido. En la primera fase, se calculan la distancia de viaje y la duración de todos los pares de contenedores. En la segunda, se estiman los niveles de llenado y se descartan los contenedores correspondientes. La tercera fase es donde se generan las rutas y en la última, se muestra la solución en un navegador web.

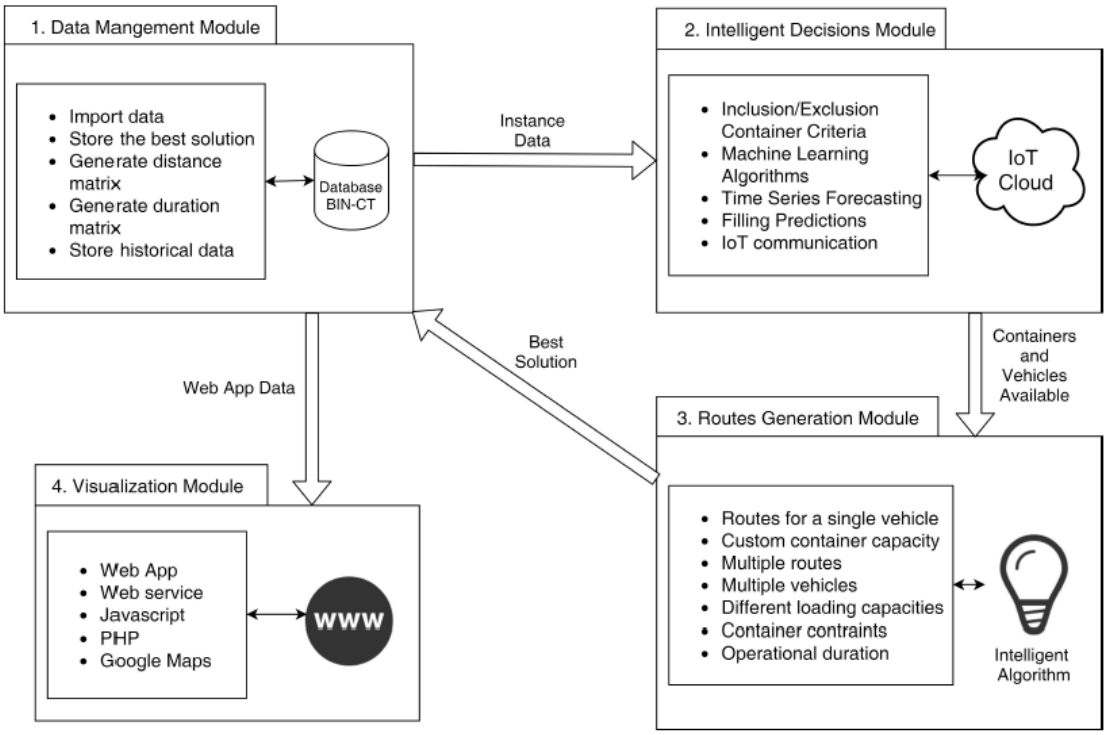


Figura 3. Diagrama de los módulos del proyecto BIN-CT.

2. Proyecto e-Garbage

El proyecto e-Garbage, es un sistema de detección del nivel de llenado de los contenedores de basura y determina rutas de recogida eficientes. Este proyecto fue implementado en tres municipios de la provincia de Badajoz, para monitorizar 49 contenedores de papel. Fue impulsado por la división Wellness Smart Cities & Solutions perteneciente a la empresa Wellness Telecom, juntamente con la Universidad de Sevilla y respaldado por Corporación Tecnológica de Andalucía (CTA) y la colaboración de Lipasam [3].

Al igual que en el primer proyecto comentado (BIN-CT), monitorea constantemente el nivel de llenado de los contenedores, mediante sensores volumétricos (concretamente sensores ultrasónicos) y también recoge datos de variación de temperatura para posibles casos de incendios. Los datos son transmitidos a una centralita, usando una red inalámbrica en una banda de frecuencia libre. Una vez transmitidos los datos, un software genera la ruta más apropiada, gracias a un algoritmo, antes de que los

camiones de recogida inicien su recorrido. La localización de los contenedores, necesaria para la generación de la ruta óptima, se determina gracias a la tecnología GPS.

Además de todo esto, el sistema se complementa con dispositivos de identificación RFID (Radio-frequency identification) en los contenedores, para transmitir datos al camión sobre la frecuencia de recogida, tipo de materiales, etc., para almacenarlos en una base de datos estadística. También, los datos recogidos por los sensores, están accesibles para el ciudadano a través de una aplicación Android.

3. Proyecto LIFE-EWAS

Proyecto implantado en Sevilla y Chania (Grecia), durante el año 2016, para la gestión de residuos urbanos, llevado a cabo por la empresa Wellness Smart Cities & Solutions y con la participación de Lipasam y de la Corporación de Empresas Municipales de Sevilla. Este proyecto logra una reducción de los costes asociados a la recogida de contenedores de vidrio, mediante un sistema que reporta el nivel de llenado de cada contenedor en tiempo real. Este sistema es conocido como Quamtra y fue desarrollado por la empresa Wellness Smart Cities & Solutions [4].

El proyecto se basa en la instalación de sensores en cada contenedor, los cuales se comunican con un software, que envían señales en tiempo real a la empresa de servicios municipales para informar sobre el estado de los contenedores, como el nivel de llenado, la temperatura o posibles daños (a través de vibraciones o la ubicación del contenedor, informando sobre un desplazamiento no controlado) y mejorar así, la planificación de los servicios de recogida. Al igual que el resto de proyectos anteriormente comentados, también optimiza las rutas de recogida y muestra información sobre el estado de los contenedores tanto a los servicios de recogida como a los ciudadanos, mediante una aplicación móvil que accede a los principales servicios e información de una plataforma web.

Con el fin de poder absorber todos los datos en una infraestructura de la nube, se utiliza una solución de Software as a Service (SaaS), para poder ofrecer una buena escalabilidad del sistema con un bajo coste inicial.

En Sevilla, se monitorizaron 268 contenedores, de los cuales 215 eran de vidrio, 29 soterrados de plástico y vidrio y 24 de distinta naturaleza para testear la validez del dispositivo. Por otro lado, en Chania se monitorizaron 400 contenedores.

4. Proyecto CleanTec Software

Proyecto para la optimización de la gestión y el uso de recursos, a través de la detección del volumen de llenado de un recipiente o contenedor. Disponen de dos módulos para controlar el volumen de llenado. El primero es ***Sensor Geombo Garbage Basic***, que es una solución destinada al seguimiento y monitorización del nivel de llenado, mientras que el segundo llamado ***Sensor Geombo Garbage Pro***, también proporciona la

temperatura interna e informa sobre posibles eventos de vandalismo, detectando un zarandeo del contenedor, a través de un sensor de movimiento, que en este caso es un acelerómetro [5][6].

Como características generales que ambos módulos ofrecen, están la amplia duración de la batería de hasta 5 años, el uso de la comunicación GPRS/3G, la tecnología de medida son los ultrasonidos, uso de las bandas 850/900/1800/1900 MHz, alarma de nivel de llenado configurable, envío de la alarma a la central para el cálculo de la ruta y tanto uno como otro, son instalados en la parte superior del contenedor.

El uso de la tecnología GPS, permite saber si el contenedor ha sido movido de su posición. También, nos aporta información sobre el contenedor, identificando cada uno de manera independiente. Otra forma de identificarlos, es mediante la tecnología RFID y se hace cuando el camión está realizando la carga de residuos.

En este proyecto se ofrece también, un mapa virtual donde ver el nivel de llenado de cada contenedor.

5. TSwasTe

TSwasTe es un proyecto para la gestión inteligente de residuos, utilizando un dispositivo de bajo coste con un sensor de ultrasonidos, un sensor de temperatura y un protocolo de comunicaciones compatible con tecnologías de bajo consumo como Sigfox, LoRa o Zigbee, sin embargo, el nuevo proyecto tiene el objetivo de utilizar NB-IoT como protocolo de comunicación. Además del nivel de llenado y la temperatura, también proporciona la posición del contenedor, la posibilidad de reprogramación remota a través de la red y la visualización de datos y gestión de nodos, alarmas e histórico con una plataforma software [7].

Este proyecto es uno de los primeros que ofrece compatibilidad con la tecnología de comunicación NB-IoT, y con esto, pretende reducir al máximo el consumo de su dispositivo y precio de este. El lanzamiento de este dispositivo ha sido posible con la ayuda de Vodafone y el fabricante de chipsets ublox, que además de la tecnología NB-IoT, utiliza una SIM. La programación del dispositivo se realiza vía comandos AT y las comunicaciones punto a punto con UDP. En cuanto al precio, TSwasTe prevé reducirlo hasta los 5 dólares.

Por ahora, tan solo existe un prototipo implantado en el municipio de Gorliz (Vizcaya), aunque TST, ya tenía una versión de este tipo de dispositivos, que utilizaba la tecnología 2G.

A continuación se muestra una tabla comparativa de diferentes parámetros, correspondiente a los proyectos descritos en esta sección.

	BIN-CT	e-Garbage	LIFE EWAS	CleanTec Software	TSwasTe
Tecnologías de comunicación compatibles	-	-	Zigbee / WiFi / GPRS	Sigfox / 3G / GPRS	NB-IoT / GPRS / Sigfox / LoRa WiFi / Zigbee
Sensores utilizados	volumétrico	volumétrico (ultrasónicos) / temperatura	volumétrico (ultrasónicos) / temperatura / vibración	volumétrico (ultrasónicos) / movimiento / temperatura	volumétrico (ultrasónicos) / temperatura
Nivel de implantación	217 contenedores de papel (Algeciras, Cádiz)	49 contenedores de papel (Badajoz)	668 contenedores de vidrio y plástico (Sevilla y Chania)	-	Municpio de Gorliz (Vizcaya)
Método de localización / identificación	-	GPS / RFID	GPRS	GPS / RFID / Código de barras	GPRS
Frecuencia de adquisición	-	-	1 hora (variable)	3 / 6 reportes diarios	Cada 4 horas

Tabla 1. Tabla comparativa de diferentes parámetros de los diferentes proyectos de referencia.

2.2. Recursos tecnológicos

2.2.1. Sensor volumétrico

Un sensor volumétrico es un sensor que es capaz de determinar el volumen de un objeto. En este proyecto interesa que este tipo de sensores trabajen sin contacto ni roces mecánicos con el material a detectar. A continuación, se hace un listado donde se exponen algunos que podrían ser útiles para la aplicación expuesta en este proyecto.

1.1. Sensor láser

Estos sensores miden distancia en función de la intensidad de la luz recibida. Esto lo consiguen, emitiendo luz en línea recta desde un emisor y recibiendo el haz en el receptor, donde se toma la medida de la intensidad de luz recibida. Para este proyecto, los sensores láser que podrían ser de interés, son los que tienen transmisor y receptor en el mismo módulo, puesto que solo es posible instalar un módulo en la parte superior del contenedor.

El sensor VL53L1X o VL53L0X (varía la distancia de detección), a diferencia de los sensores de infrarrojos convencionales que usan la intensidad de la luz reflejada para

estimar la distancia del objeto, sino que utilizan el tiempo de vuelo (TOF) a través del envío de pulsos de luz láser infrarroja [8]. Dependiendo del modelo, será posible detectar a mayor o menor distancia un objeto. En este caso, el modelo de este sensor que menor distancia detecta son 200 cm.

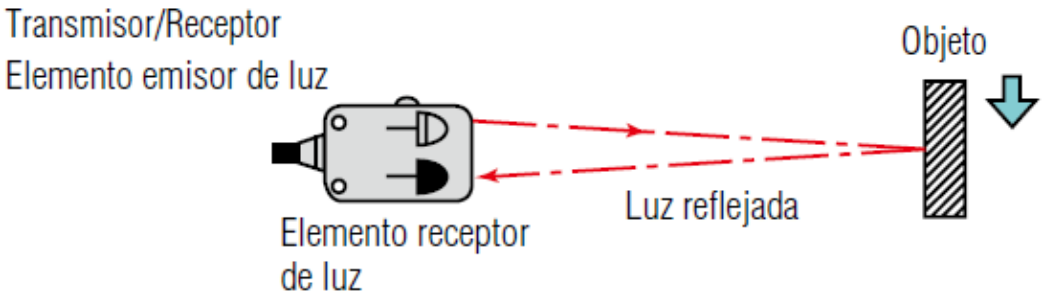


Figura 4. Funcionamiento sensor láser para medir distancia

1.2. Sensor ultrasonidos

Un sensor ultrasónico o de ultrasonidos, mide la distancia mediante ondas ultrasónicas. Este tipo de sensores son ideales sobre materiales translúcidos, irregulares, etc. [9] y son capaces de detectar cualquier tipo de material independientemente del color. Los sensores ultrasónicos ofrecen el valor de salida tanto de manera digital como analógica.

La distancia es medida a través del tiempo transcurrido entre el momento que se emite la onda y el momento en el que se recibe, después de haber sido reflejada en el objeto, por tanto, trabajan con el mismo principio de medición que los sensores láser, con el tiempo de vuelo (TOF) pero en este caso de ondas ultrasónicas. EL sensor HC-SR04 es un ejemplo de sensor ultrasónico ampliamente usado en este tipo de aplicaciones. La siguiente figura muestra el proceso descrito para la medida de la distancia.

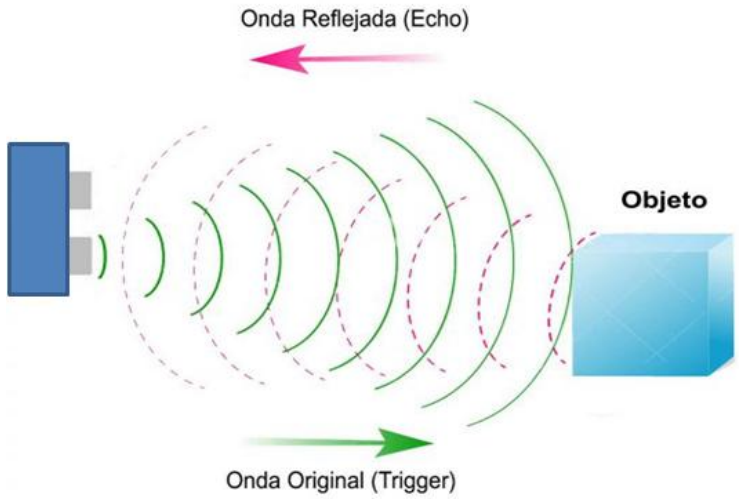


Figura 5. Funcionamiento sensor ultrasonidos

1.3. Sensor Inductivo

Al igual que los otros dos tipos de sensores para la medición de la distancia, los inductivos tampoco necesitan contacto con el material para detectar a qué distancia se encuentra. A diferencia de los sensores de tipo láser y los ultrasónicos, la gran mayoría de este tipo de sensores, necesitan una superficie metálica para realizar la medida. Contienen un devanado interno por el que circula una corriente y genera un campo magnético, que se contrapone al generado cuando se induce corriente al material. A medida que el objeto se aproxima, el flujo de corriente inducido aumenta y provoca que la carga en el circuito crezca y por tanto, la oscilación decrece, esta variación emite una señal de detección [10].

La desventaja de este tipo de sensores es su pequeño rango de detección.

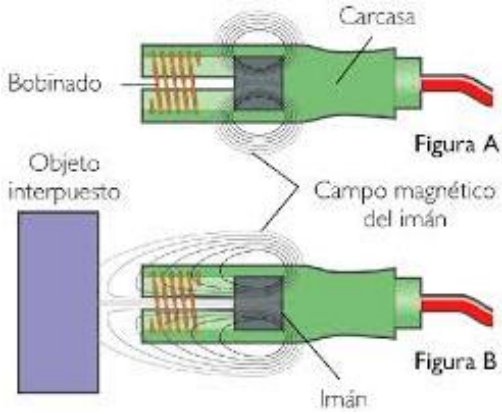


Figura 6. Campo magnético del sensor inductivo sin detección de un objeto (parte superior) y con detección de un objeto (parte inferior).

1.4. Sensor Capacitivo

Los sensores capacitivos tienen como principio de funcionamiento el cálculo de la constante dieléctrica entre el sensor y el material. Pueden detectar materiales metálicos y no metálicos pero tienen un muy bajo rango de detección [11].

1.5. Sensor microondas

Estos sensores se basan en la detección por movimientos mediante el efecto Doppler, emiten una frecuencia del orden de las microondas, que es reflejada en el objeto a detectar. Sin embargo, este tipo de sensores son cada vez menos usados debido a su gran probabilidad de falsa alarma [12]. Un ejemplo de sensor microondas es el NJR4265.

Se ha realizado una tabla para comparar los diferentes tipos de sensores mostrados en esta sección.

Tipo de sensor	Ventajas	Desventajas
Láser	• Alta distancia de detección > 2m • Bajo consumo en modo trabajo (16 mA aprox.) [8] • Detección de materiales metálicos y no metálicos • FOV 15º - 27º	• Resolución 1mm
Ultrasónico	• Alta distancia de detección de hasta 3.5 m • Bajo consumo en modo trabajo (15 mA aprox.) [13] • Detección de materiales metálicos y no metálicos • FOV 15º - 30º	• Resolución 3 mm -1 cm
Inductivo	• Alta precisión (µm) • Alta estabilidad de medida	• Bajo rango de detección < 25 mm • Detectan solo materiales metálicos
Capacitivo	• Alta precisión (nm) • Alta estabilidad de medida • Detección de materiales metálicos y no metálicos	• Bajo rango de detección < 10 mm
Microondas	• Alta distancia de detección • Consumo en modo trabajo de 60 mA [14] • Alto ángulo de visión (FOV) (60º - 70º)	• Alta vulnerabilidad a falsas alarmas

Tabla 2. Comparativa de los diferentes sensores volumétricos.

Por tanto, con esta tabla es posible observar que los sensores de láser y ultrasónicos, son los dos únicos tipos que serán útiles para este proyecto, puesto que los inductivos y los capacitivos tienen un rango de detección demasiado bajo para esta aplicación, mientras que los sensores de microondas, aun con una buena distancia para detectar un objeto, tienen un consumo significativamente mayor para una aplicación que ha de funcionar con baterías.

2.2.2. Sensor detección de gas

Los sensores presentados a continuación, son utilizados para medir el nivel de gas. En este proyecto, concretamente, interesa los que detectan el nivel de dióxido de carbono del aire en una zona determinada y así, obtener la calidad del aire. La unidad en la que miden estos sensores es en partes por millón (ppm). La concentración de CO_2 típica en el aire, es de aproximadamente 400 ppm.

Los tres principios de medición más utilizados para sensores de detección de gas, son los siguientes [15][16]:

2.1 Sensor de oxidación catalítica

El funcionamiento de este tipo de sensores es por la oxidación del gas vía catalítica y están compuestos por dos bobinas de hilo de platino (Pt), encapsuladas en un material cerámico. Una de las bobinas está tratada con un material catalizador y es por la que se hace fluir una corriente eléctrica que hace aumentar su temperatura y en consecuencia su resistencia. Esta variación de la resistencia con respecto a la resistencia en aire limpio es la que se utiliza para la evaluación electrónica. La otra bobina, no está tratada con material catalizador, ya que solo se utiliza para evitar influencia de la temperatura ambiente y junto con la primera, forman un puente de Wheatstone.

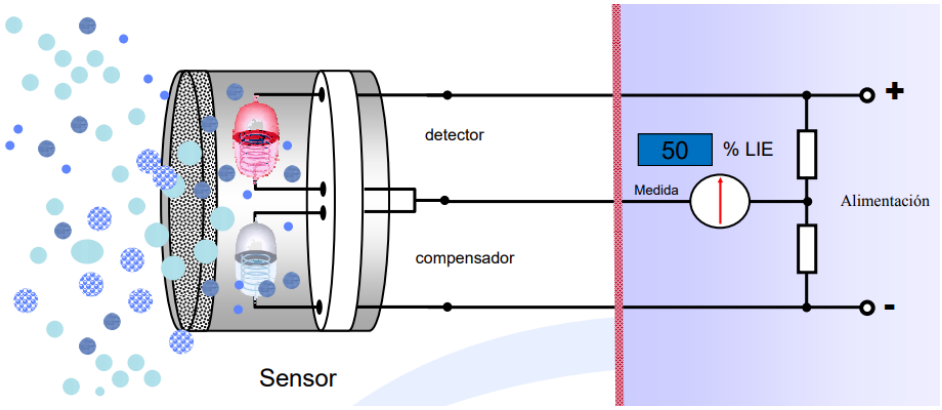


Figura 7. Sensor catalítico donde se aprecian los dos filamentos de Pt.

2.2 Sensor de espectroscópico infrarrojo

Este tipo de sensores se basan en el principio de absorción de energía de los compuestos a una determinada longitud de onda, como es el infrarrojo. El sensor tiene un emisor que manda un haz de luz infrarroja con la longitud de onda de absorción del gas, que es recibida por el receptor y se controla la atenuación que ha sufrido que nos dirá el nivel de gas. La atenuación es proporcional a la cantidad de gas en el aire. Un ejemplo de este tipo de sensores es el MH-Z19.

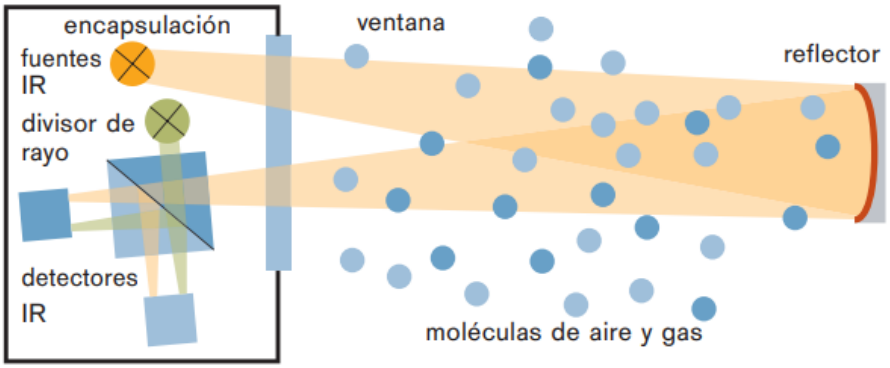


Figura 8. Sensores de detección de espectroscópico infrarrojo

2.3. Sensor electroquímico

Un sensor electroquímico se compone de como mínimo dos electrodos que están conectados eléctricamente de dos maneras diferentes: una mediante un electrolito y otra por medio de un filamento. Los dos electrodos tienen características catalíticas para que puedan reaccionar químicamente a la presencia de gas.

Utilizan un método basado en mediciones de corriente entre el electrodo detector y el electrodo contador. Los gases reaccionan electroquímicamente con el electrodo detector mediante reacción REDOX, generando una corriente eléctrica proporcional a la cantidad de moléculas de gas oxidadas. Un ejemplo de este tipo de sensores es la serie MQ.

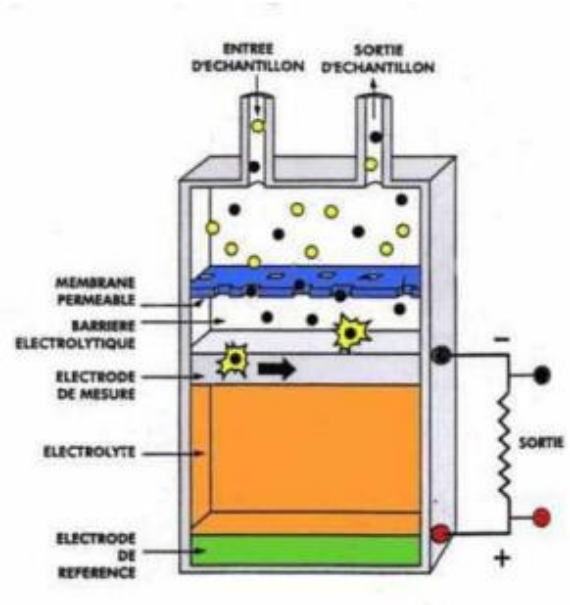


Figura 9. Sensor electroquímico de detección de gas

2.2.3. Tecnología de comunicación

La tecnología de comunicación a utilizar en esta aplicación será importante, porque como ya se ha comentado en la introducción del proyecto, deberá tener un consumo de energía lo más reducido posible para que la duración de las baterías se maximice, es decir, que el dispositivo tenga la mayor autonomía posible, ya que este no tendrá suministro eléctrico permanente.

Otro punto importante, es que la comunicación deberá ser inalámbrica, puesto que, al igual que con el suministro eléctrico, no es posible cablear los módulos instalados en los contenedores de reciclaje. También debe ser tenido en cuenta el rango de cobertura, por lo que se descartan tecnologías como bluetooth o Zigbee, que proporcionan distancias de máximo 10 y 75 metros, respectivamente.

A continuación se exponen diferentes comunicaciones inalámbricas de bajo consumo y largo alcance:

4.1. Sigfox

Es una tecnología de banda estrecha (192 KHz) con un ancho de banda para cada mensaje de 100Hz. Sigfox utiliza la banda de los 868 MHz en Europa (EEUU utiliza la banda de 915 MHz), con una velocidad de transmisión de entre 100 y 600 bits por segundo, que permite tener un largo alcance. El tamaño de cada mensaje es de máximo 12 bytes y utiliza la modulación BPSK [17].

Ofrece una alta eficiencia energética, puesto que los chips de Sigfox consumen entre 10 y 50 mA durante la transmisión, con una potencia de salida de 14 dBm. Existen otros factores por los que estos módulos tienen una buena autonomía, como que no se requiere emparejamiento entre el módulo y la estación base y el consumo durante el modo standby es de pocos nanoamperios.

Un aspecto importante, es la escalabilidad del sistema, gracias a la capacidad de la infraestructura de la red, por factores como la modulación de banda estrecha, la diversidad de frecuencia y tiempo introducida por el acceso aleatorio y también, por la diversidad espacial debido a la superposición de celdas de la red.

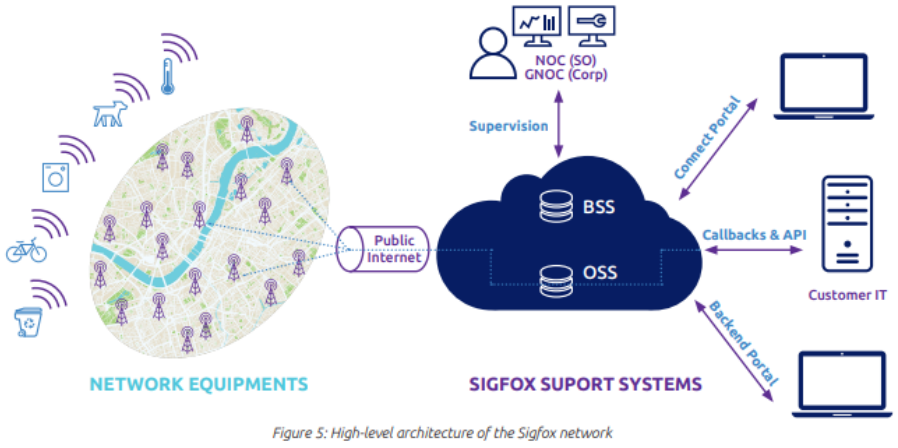


Figure 5: High-level architecture of the Sigfox network

Figura 10. Arquitectura red de Sigfox

4.2. LoRa

LoRa (Long Range) es una tecnología, estandarizada por LoRa-Alliance, de espectro ensanchado que utiliza la banda de los 433 MHz o la de los 868 MHz en Europa, con un coste de conectividad bajo. Tiene una velocidad de transmisión de datos baja, de entre 0.3 Kbps y 5.5 kbps, por lo que está enfocada a la transmisión de datos de sensores, con una potencia de transmisión de 20 dBm y una sensibilidad en el receptor de -140 dBm [18].

Utiliza una modulación similar a la FM, ya que modula la señal variando la frecuencia, pero a diferencia de la FM donde se cambia la frecuencia instantáneamente, aquí se varía lentamente a lo largo del tiempo, esto es conocido como modulación de espectro ensanchado chirp.

En esta tecnología de comunicación inalámbrica, además de LoRa, también juega un papel importante LoRaWAN, que describe los protocolos y el formato de datos de la red LoRa. Las redes LoRaWAN están organizadas en una tipología de estrella que utiliza el esquema de modulación LoRa [19]. Cada terminal LoRaWAN envía los datos a una puerta de enlace que transfiere esos datos a otras redes como Ethernet o WiFi.

El alcance de este tipo de comunicaciones pueden llegar a ser de 15 km en áreas abiertas y de 2 km en zonas urbanas. Ofrecen una autonomía de hasta 12 años y al igual que Sigfox, una buena escalabilidad.

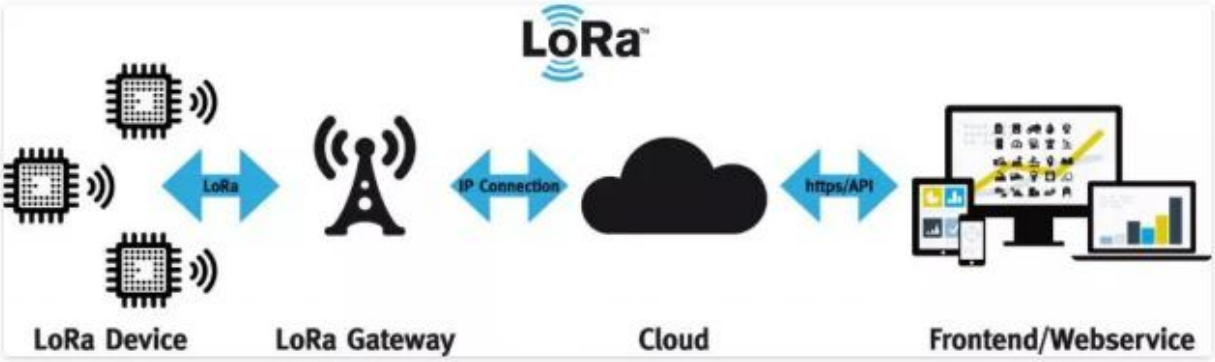


Figura 11. Arquitectura de red LoRa

4.3. NB-IoT

Al igual que las dos tecnologías anteriormente comentadas, NB-IoT está pensada para el concepto de Internet de las cosas (IoT) y las redes LPWAN. Puede desplegarse en varias bandas de frecuencia, y al ser un estándar creado por 3GPP, mantiene una compatibilidad con la infraestructura de red móvil y por tanto, puede utilizar el espectro tanto licenciado como no licenciado [20].

También ofrece una larga vida útil de las baterías con un gran alcance en interior y soportando múltiples conexiones, hasta 100.000. Al utilizar el mismo estándar que LTE, la cobertura es prácticamente global y el nivel de seguridad de la red es igual y por tanto, muy alto.

Otras características de esta tecnología son la potencia de transmisión de los dispositivos, que es de 23 dBm, el uso de un ancho de banda de 200 kHz y su velocidad máxima de transferencia de datos, que es de 100 kbps.

4.4. LTE-M

Long Term Evolution for Machines (LTE-M) es una tecnología para las redes LPWAN, también estandarizada por 3GPP, compatible con la infraestructura LTE y por tanto, con la misma cobertura [21].

El objetivo de esta tecnología es que conviva con las ya existentes redes móviles 2G, 3G y 4G con los mismos niveles de seguridad y privacidad.

Ofrece un ancho de banda mayor al resto de tecnologías expuestas con una velocidad de transmisión de hasta 1Mbps, permitiendo incluso enviar voz, pero con un mayor coste en los dispositivos compatibles con LTE-M.

A continuación se muestra una tabla comparativa de las diferentes tecnologías de comunicación presentadas en esta sección.

Tech	Data rate	Ancho de banda	Banda de frecuencia	Alcance
Sigfox	100-600bps	100 Hz	Gratuita (ISM) 868 MHz EU 902-928 MHz rest of the world	3 km urbano 20 km zona rural
LoRa	0.3 - 5.5 kbps	Depende del SF (125 kHz / 250 kHz / 500 kHz)	Gratuita (ISM) 433/868 MHz EU 915 MHz USA 433 MHz Asia	2 km urbano 15 km zona rural
NB-IoT	<100 kbps [22]	180 KHz	Licenciada 3GPP	1 km urbano 5 km zona rural
LTE-M	Up to 1Mbit	1.4 MHz	Licenciada 3GPP	1 km urbano 5 km zona rural

Tabla 3. Tabla comparativa de las diferentes tecnologías de comunicación.

Las tecnologías NB-IoT y LTE-M se descartarán para este proyecto, puesto que como se indica en la tabla, utilizan bandas de frecuencias con licencia y por tanto, el coste del sistema aumentaría.

2.2.4. Bases de datos

La base de datos será donde los datos, serán almacenados después de haber sido recogidos por los módulos instalados en los contenedores de reciclaje y posteriormente transmitidos por el módulo de comunicaciones. Un aspecto importante, es el coste que tiene esta BBDD para nuestro sistema y puesto que el mercado ofrece bases de datos de libre distribución, la investigación se centrará en este tipo de bases de datos.

A continuación, se muestra una tabla comparativa con las posibles soluciones para dicha base de datos a implementar.

Base de datos	Prestaciones	Limitaciones
MySQL [23]	• Base de datos relacional • Múltiples plataformas • API para C, C++, Java, Perl, PHP, Python, Ruby y TCL • Administración se basa en usuarios y privilegios • Gran estabilidad	• No inclusión de características de objetos como tipo de datos
PostgreSQL [24]	• Base de datos relacional • Múltiples plataformas • API para C, C++, Java, Perl, PHP, Python y TCL • Administración se basa en usuarios y privilegios • Gran estabilidad • Implementación de algunas extensiones de orientación de objetos	• Las transacciones se abortan por completo si se encuentra un fallo durante su ejecución • No soporta *tablespaces* para definir dónde almacenar la base de datos
MariaDB [25]	• Base de datos relacional • Múltiples plataformas • API para C, C++, Java, Perl, PHP, Python, Ruby y TCL • Administración se basa en usuarios y privilegios • Gran estabilidad	• No es posible restaurar un servidor eliminado
MongoDB [26]	• Base de datos no relacional o NoSQL • Base de datos documental • Velocidad mayor que una base de datos relacional • Flexibilidad en la variación de los campos predefinidos	• No existen mecanismos para hacer transacciones entre documentos

Tabla 4. Tabla comparativa de diferentes bases de datos de distribución libre.

2.2.5. Microcontrolador

El microcontrolador será una parte importante, ya que será el que deberá gestionar el módulo de sensores y la parte transmisora del módulo de comunicación. Al igual que el resto del hardware de la aplicación del proyecto, será necesario que el consumo de energía sea lo menor posible, pero asegurando un buen rendimiento y un bajo coste.

En el mercado es posible encontrar microcontroladores con diferencias entre unos y otros, a nivel de potencia del procesador, GPIOs, tipos de comunicación, memoria flash, memoria SRAM, etc. Para este proyecto no será necesario un nivel de computación alto, puesto que no se trata de una aplicación que trabaje en tiempo real, pero si necesitaremos un mínimo de 20 GPIOs y comunicación serie.

Debido a la alta cantidad de microcontroladores existentes, a día de hoy, que cumplen con los requisitos comentados, se han escogido algunos ejemplos que podrían ser útiles, integrados en placas de desarrollo que facilitan su implementación. Más tarde, se muestra una tabla comparativa de todos ellos.

6.1. STMicroelectronics

STMicroelectronics es una compañía de fabricación de componentes electrónicos semiconductores, que ofrece diferentes soluciones para aplicaciones de electrónica de consumo, automoción, periféricos informáticos y en el mundo de la industria [27].

- Núcleo STM32L031K6
- STM32F103C8
- Núcleo STM32L152RE

6.2. NXP Semiconductors

Empresa fabricante de semiconductores con diferentes series de microcontroladores y microprocesadores para aplicaciones de comunicaciones, automoción, SmartCities y aplicaciones industriales. Presentan las siguientes placas de desarrollo para la creación de prototipos, todas ellas están basadas en microcontroladores de 32 bits, con un consumo muy bajo, un buen rendimiento y a un bajo coste [28].

- OM13087 - LPCXpresso1115
- OM13080 - LPCXpresso1125 Board
- OM13077 - LPCXpresso5410

6.3. Texas Instruments

Empresa estadounidense que diseña y fabrica semiconductores y circuitos integrados. Gran parte de su mercado está enfocado a procesadores digitales de señal y semiconductores analógicos. Algunas placas de desarrollo de este fabricante son las siguientes [29]:

- MSP-EXP430F5529 LaunchPad
- MSP-EXP430FR2311 LaunchPad
- MSP-EXP430FR5994 LaunchPad

A continuación se muestra una tabla comparativa entre las placas de desarrollo descritas en esta sección.

Placa de desarrollo	Frecuencia CPU	Nº GPIO	Flash memory	Power consumption @ 1 MHz	Tipos de comunicaciones	Precio
Núcleo STM32L031K6	32 MHz	26	32 kB	76 µA	1xI2C 1xUSART 2xSPI	10 $
STM32F103C8	72 MHz	37	128 kB	1.02 mA	2xI2C 3xUSART 2xSPI 1xCAN	10 $
Núcleo STM32L152	32 MHz	51	512 kB	195 µA	2xI2C 5xUSART 8xSPI	13 $
OM13087 - LPCXpresso1115	50 MHz	42	64 kB	840 µA	1xI2C 1xUART 2xSPI	25 $
OM13080 - LPCXpresso1125	50 MHz	38	64 kB	700 µA	1xI2C 3xUART 2xSSP	29 $
OM13077 - LPCXpresso5410	150 MHz	50	512 kB	1.5 mA @ 12 MHz *	3xI2C 4xUSART 2xSPI	28 $
MSP-EXP430F5529LP	25 MHz	63	128 kB	290 µA	2xI2C 2xUART 4xSPI	12.99 $
MSP-EXP430FR2311LP	16 MHz	16	4 kB*[2]	126 µA	1xI2C 1xUART 2xSPI	13.99 $
MSP-EXP430FR5994LP	16 MHz	68	256 kB*[2]	118 µA	4xI2C 4xUART 8xSPI	16.99 $

Tabla 5. Comparativa de los distintos microcontroladores presentados.

*Datasheet no indica el consumo para frecuencias de 1 MHz. No es posible calcularlo, ya que el consumo no es lineal.

*[2] La memoria es de tipo FRAM.

3. DISEÑO DEL PROYECTO

Este proyecto, como ya se ha comentado anteriormente, tiene por objetivo la monitorización del estado de los contenedores de reciclaje, concretamente con la posibilidad de hacer un seguimiento de los que están compuestos por residuos orgánicos. Para llevar a cabo esto, será necesario diseñar un proyecto que contenga diferentes módulos , cada uno de ellos con una funcionalidad distinta, que más tarde se integrarán en un producto final.

Los diferentes módulos que compondrán este proyecto serán los siguientes:

1- Módulo de sensores

2- Módulo de comunicación

    2.1- Parte transmisora

    2.2- Parte receptora

3- Batería

4- Base de datos

5- Aplicación de procesamiento de datos

Como se observa, se diferencian dos partes en el módulo de comunicaciones, puesto que una, la parte transmisora, estará junto al módulo de sensores y será la encargada de transmitir los datos que reciba de este, desde el contenedor hasta la parte receptora, que se encuentra dentro del rango de cobertura de la red WiFi donde está la base de datos del sistema.

A continuación, se desarrollará más en detalle cada uno de los puntos arriba listados, explicando su funcionalidad y características y exponiendo los diferentes componentes de los que están compuestos, así como las comunicaciones utilizadas entre ellos.

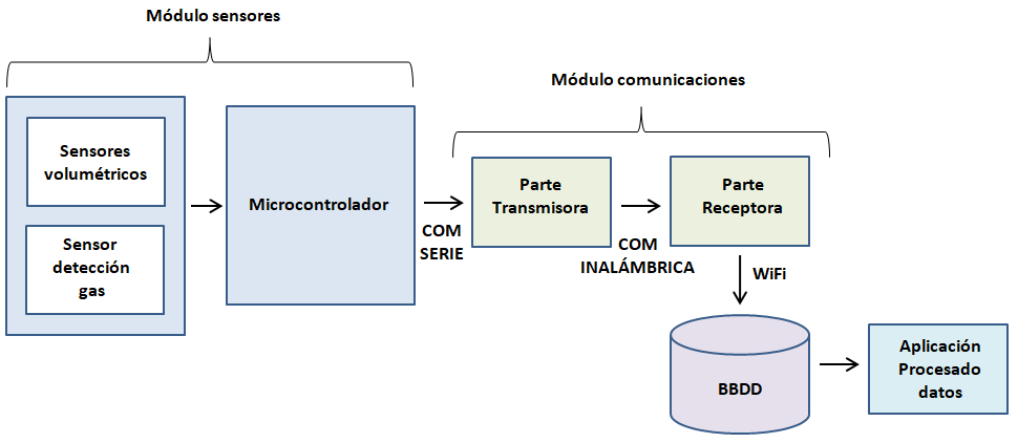


Figura 12. Esquema del flujo de datos del sistema completo de monitorización del estado de los contenedores de reciclaje.

3.1. - Módulo de sensores

Este módulo tiene por objetivo monitorizar y detectar el nivel de volumen de residuos y el de dióxido de carbono, que tiene un contenedor. Para ello, se han utilizado cinco sensores volumétricos, un sensor de detección de CO_2 y una placa de desarrollo con la que gestionar estos sensores. En esta sección se describe la función que tienen en esta aplicación, juntamente con el conexionado del módulo completo.

Para la elección de los distintos sensores, se ha tenido en cuenta lo siguiente:

- El principio de funcionamiento
- Rango de operación
- Precisión
- Estabilidad
- Disponibilidad
- Coste del sensor

Este módulo también deberá transmitir los datos adquiridos de los sensores al módulo de comunicaciones

3.1.1. Sensor volumétrico

En primer lugar, como sensores volumétricos, se han utilizado sensores ultrasónicos, por lo que su principio de funcionamiento es a través de ondas ultrasónicas. Estas ondas son enviadas por el sensor y reflejadas en el objeto, volviendo de nuevo al sensor. La medida de la distancia se realiza a través del tiempo que ha transcurrido en dicho proceso. Para este proyecto, se ha escogido el sensor HC-SR04, por los siguientes motivos:

1. Sencillez de implementación.

Como se explica con detalle en la siguiente subsección, funciona enviándole un pulso que el sensor recoge y responde con un pulso proporcional a la distancia, que es leído en el pin del microcontrolador sin necesidad de enviarle tramas de comunicación como en otro tipo de sensores.

2. Parámetros eléctricos

Este sensor tiene un muy bajo consumo en el modo trabajo, de tan solo 15 mA, por lo que le hace un buen candidato para una aplicación donde el tipo de alimentación será a través de una batería. Además, el rango de voltaje de trabajo es de 5V, un voltaje compatible con la mayoría de los microcontroladores del mercado.

3. Rango de detección

El rango de detección va desde los 2 cm hasta los 4 metros, valores similares a los sensores que utilizan el rango laser y suficiente para esta aplicación, ya que un contenedor convencional tiene una altura de 1600 mm [30].

4. Relación coste / rendimiento

Por último, pero no menos importante, es la relación calidad/precio. Este sensor ofrece un muy buen rendimiento por un coste increíblemente bajo y es por eso que es ampliamente usado en aplicaciones de medida de la distancia.

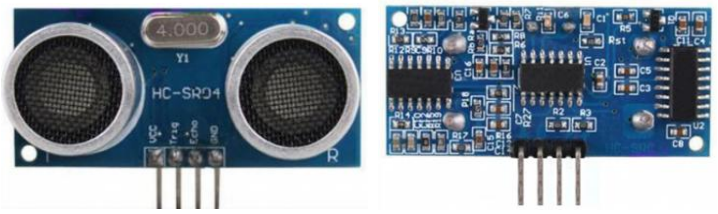


Figura 13. Sensor HC-SR04. A la izquierda se muestra la parte donde se encuentran el transmisor y receptor, mientras que la parte derecha muestra la cara BOTTOM con los componentes electrónicos.

La tabla siguiente muestra las características principales de este sensor ultrasónico [13]:

Parámetro	Valor
Power supply	5 VDC
Consumo de trabajo	15 mA
Consumo de reposo	< 2 mA
Rango de detección	2 cm - 4 m
Resolución	0.3 cm
Ángulo de medida	30º

Tabla 6. Parámetros del sensor ultrasónico HC-SR04

3.1.1.1. Sistema de medida del sensor HC-SR04

El sensor HC-SR04 consta de un transmisor y un receptor, así como cuatro pines que son, Vcc, GND, "Trigger" y "Echo". Estos dos últimos son los que se utilizan para la medición, siendo el pin de "Trigger" el que recibe el pulso generado por el microcontrolador, que debe ser de como mínimo 10 µs. Este pulso generado es enviado por el transmisor, que vuelve y es recogido por el receptor después de haber rebotado en un objeto. El otro pin usado para la medición, el pin "Echo", nos devuelve un pulso proporcional a la distancia a la que se encuentra el objeto [13].

Para calcular la distancia a partir del pulso proporcional devuelto, será necesario saber el tiempo de este pulso, así como la velocidad del sonido, que es de 343 m/s y que nos permite saber la distancia que ha recorrido la onda. Sin embargo, hay que tener en cuenta que esta distancia es de ida y vuelta, por lo que deberemos dividirla entre dos para saber la distancia a la que se encuentra el objeto.

Distancia [m] = (Tiempo del pulso [s] * Velocidad del sonido [343m/s]) / 2

La secuencia descrita se puede observar en la imagen siguiente.

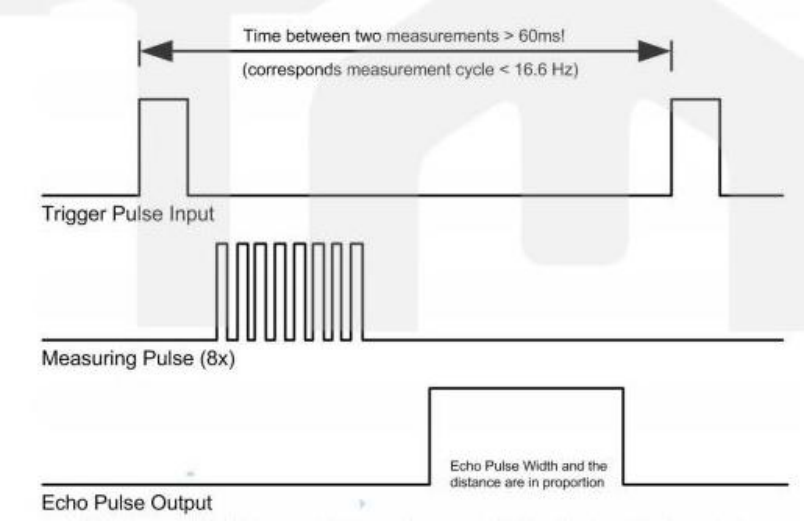


Figura 14. Proceso de medida del sensor HC-SR04.

Dependiendo de la distancia a la que se encuentre el objeto, el pulso enviado por el pin Echo, puede ser menor al milisegundo, por lo que será necesaria una frecuencia de reloj de 1 MHz para poder contar microsegundos. Para este fin, se utilizará un TIMER del microcontrolador [31].

Por otro lado, el ángulo de medida de este sensor es de 30 grados y consigue distancias de medida desde 2 cm hasta 400 cm. Se muestra su rendimiento en función del ángulo de medida en la figura 15.

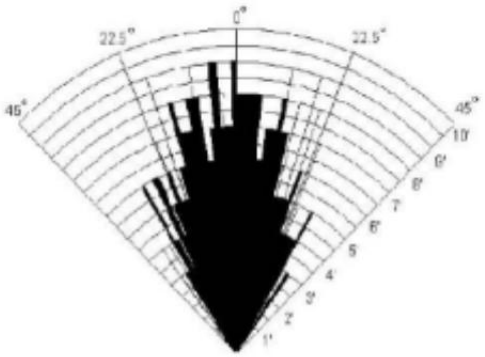


Figura 15. Rendimiento del sensor HC-SR04 en función de su ángulo de medida.

Con el fin de monitorizar con el mayor detalle posible el nivel de volumen del contenedor, se ha diseñado una matriz de este tipo de sensores, en forma de estrella. Esta configuración permite detectar el nivel de llenado en gran parte del espacio del contenedor y está formada por cinco sensores. Esta configuración es la que permite un mayor porcentaje de detección del espacio del contenedor respecto a otras topologías como anillo o en línea, con mayor eficiencia.

La configuración en estrella realizada, se puede ver en la siguiente imagen.

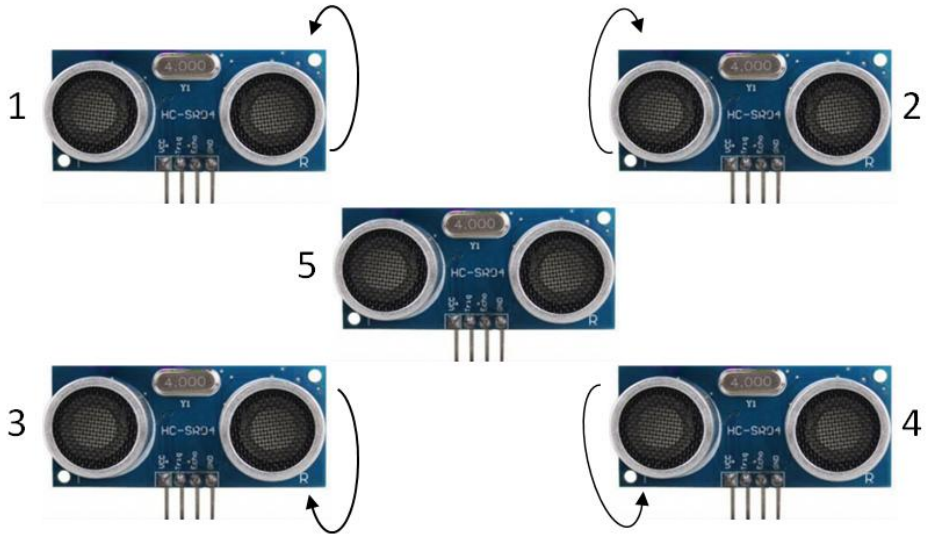


Figura 16. Configuración en estrella de los sensores ultrasónicos.

La imagen muestra dos detalles a destacar, el primero es que cada sensor tiene asignado un número en función de la posición en la que es instalado, esto es importante que se respete para poder enviar los valores ordenados a la base de datos y poder tratar los datos de manera correcta.

El segundo detalle es, la rotación a la que son instalados los sensores, esto es para conseguir detectar el mayor espacio posible del contenedor. El sensor que se encuentra en la posición central, estará perpendicular al plano x, es decir, al plano base del contenedor, mientras que el resto tendrá una rotación de aproximadamente 15º a este mismo plano, enfocando hacia cada una de las esquinas del contenedor. Para un mejor entendimiento, se muestra la figura 17, con los cinco sensores y sus rotaciones.

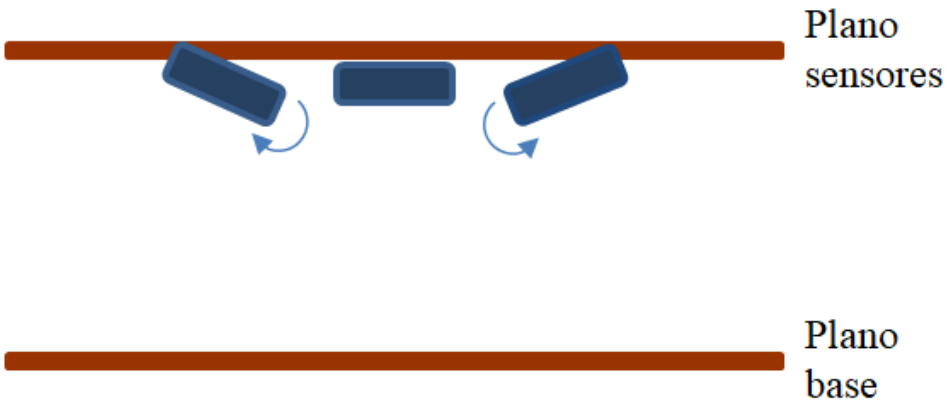


Figura 17. Representación ángulo de trabajo de los sensores ultrasónicos.

A continuación se detalla cómo se ha llegado a tomar la decisión de utilizar esta configuración descrita arriba, frente a otros tipos de topología, partir de cálculos

matemáticos, teniendo en cuenta el ángulo de medida del sensor ultrasónico y las medidas de un contenedor convencional [30].

Primero, se ha decidido rotar los sensores 1, 2, 3 y 4 (sensores que se encuentran en círculo exterior) unos 15º hacia el exterior del módulo para poder formar un ángulo de 90º con el plano base del contenedor, esto nos permite realizar el cálculo del espacio que abarca el sensor y demostrar que es la configuración más eficiente.

Entonces, sabiendo que el ángulo con el plano es de 90º y el ángulo A, correspondiente al del sensor es de 30º, es sencillo extraer el ángulo restante que será 180º - A - B = 60º. Se conoce también el lado b, puesto que es la distancia desde el sensor al plano base, y tomando el tamaño de un contenedor convencional [26] que es de 1600 mm.

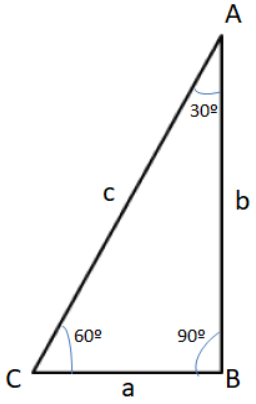


Figura 18. Haz de incidencia del sensor HC-SR04 con una rotación de 15º respecto al plano base.

Por el teorema del seno tenemos que, $\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$ y por tanto,

$$a = \frac{b * \sin A}{\sin B} = 923.7 \text{ mm}$$

$$c = \frac{b * \sin C}{\sin B} = 1847.5 \text{ mm}$$

Sabiendo el valor de '***a***' podemos calcular el área del círculo formado por los sensores como,

$$\text{Área círculo} = a * \pi^2 = 2.67\ m^2$$

Si ahora calculamos el área del plano base del contenedor, teniendo en cuenta las medidas de la figura 19, tenemos que:

$$\text{Área plano base contenedor} = a * f = 2.65\ m^2$$

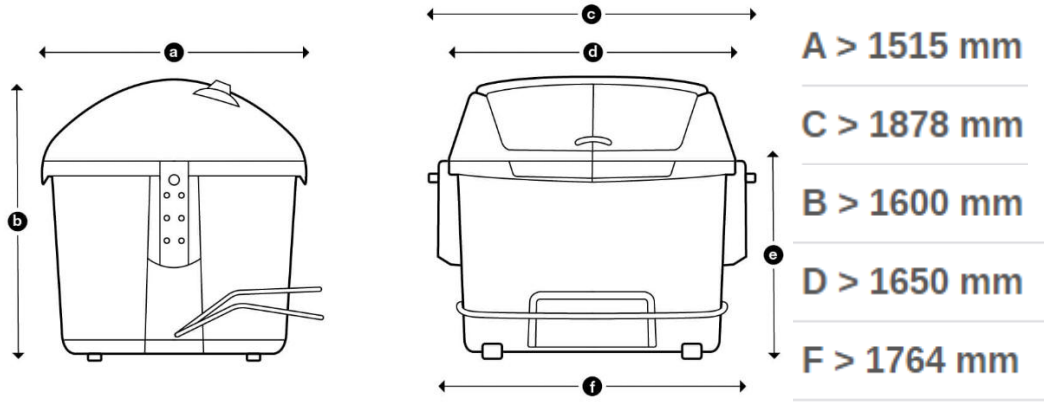


Figura 19. Medidas de un contenedor convencional de reciclaje. Fuente: [30].

Con estos cálculos confirmamos que con esta configuración se consigue abarcar prácticamente todo el área del contenedor. El quinto sensor se instala, como ya se ha comentado, en el centro a modo de refuerzo, para poder detectar con mayor precisión cada zona del contenedor.

A modo de demostración, si no se rotaran los sensores de la parte exterior de la estrella, debería haber mayor separación entre los sensores para igualar el área abarcada que se consigue rotándolos, ya que se solaparían los haces de incidencia de los sensores. La figura 20, muestra dicha separación, que hace que no sea viable su fabricación, como se demuestra con los cálculos siguientes donde se calcula cuánto deberíamos separarlos:

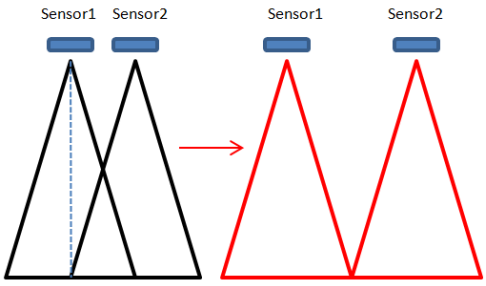
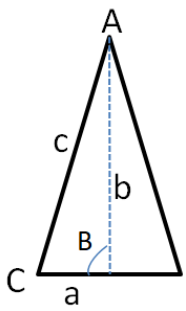


Figura 20. Separación requerida para abarcar todo el plano base del contenedor.



Al igual que antes, conocemos el lado ***b*** y el ángulo ***B***, mientras que el ángulo ***A***, ahora debemos tomar la mitad que es 15º para calcular el lado ***a*** que nos indicará la separación de los sensores.

Realizando los mismos cálculos que antes, obtenemos lo siguiente,

a = 0.428 m

Así, solo conseguiríamos abarcar un área de 0.71 m^2 y por tanto, la separación entre sensores debería ser de 0.428 m, una medida inviable para la fabricación del módulo.

Por último, el valor de cada uno de los sensores ultrasónicos es reportado a la base de datos, junto con un valor, al que se le ha denominado volumen, que es la suma de los sensores que tienen un valor por debajo de un umbral prefijado. Es decir, si hay 3 sensores que tienen un valor por debajo de dicho umbral, el valor de esta variable será de 3.

3.1.2. Sensor de detección CO_2.

Para la detección del nivel de CO_2 al que se encuentra el interior de un contenedor de reciclaje, se ha utilizado un sensor electroquímico. En este tipo de sensores, como se ha comentado en la *sección 2.2.2.*, sus electrodos reaccionan cuando se producen cambios de concentración de sustancias, es decir, cuando un gas concreto es detectado. Se generan señales entre estos dos electrodos cuando esto sucede.

El modelo escogido es el MQ-135, que permite medir CO_2 entre otro tipo de gases. Este sensor está compuesto por un tubo de cerámica con óxido de aluminio (Al2O3), una capa de dióxido de estaño (SnO2), un electrodo de medida y un calentador, este último es el encargado de aumentar la temperatura interna para que la resistencia del sensor varíe. Los datos del sensor se toman mediante cuatro de los seis pines de los que consta el sensor, con los otros dos se alimenta el sensor. Sin embargo, para tomar los datos, antes será necesario calibrar el sensor, puesto que la salida no nos muestra directamente los ***ppm (partes por millón - unidad de medida de un gas en relación con otro gas)***, sino que es un valor de voltaje proporcional a la variación de resistencia producida por el gas [32].

Figura 21. Sensor electroquímico MQ-135 para la detección de CO2.

La figura siguiente muestra las diferentes partes del sensor MQ-135, donde se puede ver los materiales de los que está compuesto cada parte y el puente de Wheatstone utilizado en este tipo de sensores para la generación de la señal a medir.

	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

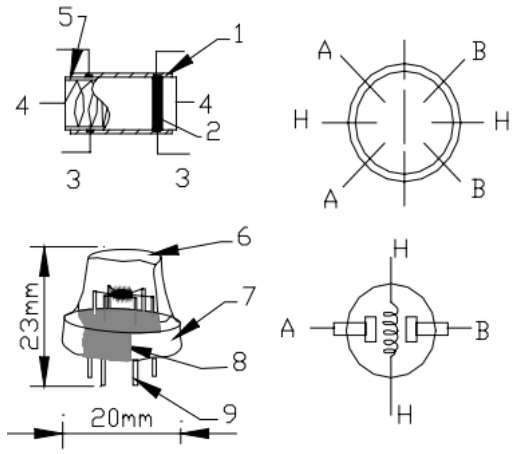


Figura 22. Características del sensor MQ-135

Para realizar el prototipo del proyecto, se utiliza una placa electrónica en la que está integrada el sensor, esta tiene cuatro pines que son, Vcc, GND, Dout y Aout. La salida Dout proporciona un nivel lógico 1 o 0, dependiendo del nivel de concentración de gas gracias a un comparador interno que previamente es calibrado con un umbral que fijaremos a través de un potenciómetro. Sin embargo, para este proyecto, se utilizará el pin de salida analógico (Aout), que es la equivalente a la salida comentada anteriormente, que nos proporciona un valor de voltaje en función de la resistencia del sensor Rs por la que pasa la corriente generada entre los electrodos.

La figura 23 muestra el sensor MQ-135 integrado en un módulo que contiene una resistencia RL, como la que se ve en el circuito de la parte derecha de la figura, para la medición del voltaje de salida del sensor.

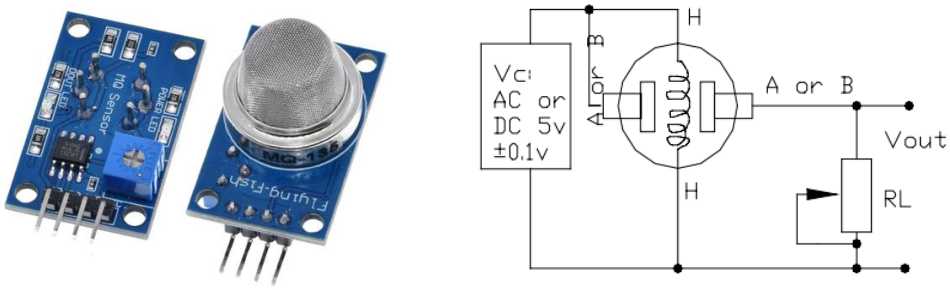


Figura 23. Sensor electroquímico MQ-135 integrado en una placa electrónica.

3.1.2.1 Calibración sensor MQ-135

Para calibrar este sensor, será necesario utilizar la gráfica del valor de ***ppm*** en función del valor ***Rs/Ro***, que nos proporciona el datasheet [33]. La recta que nos interesa es la azul con cruces, que corresponde al dióxido de carbono (CO_2).

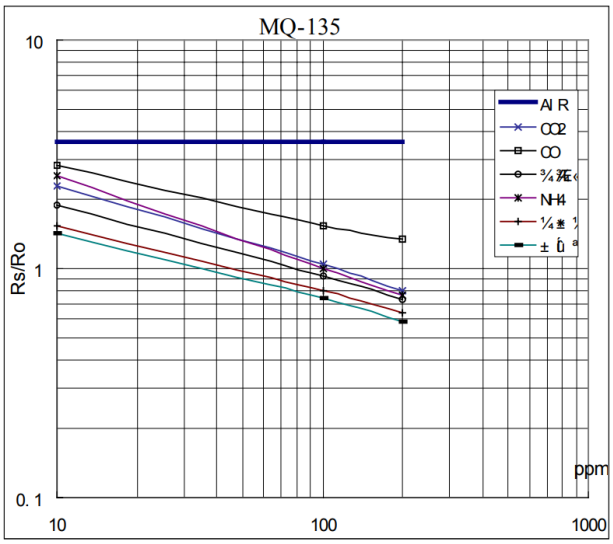


Figura 24. Gráfica log-log del sensor MQ-135 proporcionada por el datasheet.

La gráfica es logarítmica por ambos ejes, pero se puede ajustar a una función exponencial de la forma y = a · x^b donde ***a*** es la constante. Por tanto, sustituyendo en esta ecuación obtenemos lo siguiente:

$$\frac{Rs}{Ro} = a \cdot ppm^b$$

A partir de esta ecuación, podemos despejar el parámetro deseado, pero antes debemos calcular los valores de ***a*** y ***b*** mediante aproximación por mínimos cuadrados. Para ello, usaremos diferentes puntos de la gráfica correspondiente al CO_2 del datasheet. Los valores se calculan a partir de un código generado en lenguaje R [34][35].

```
#set input values
xlim = c(10, 1000)
ylim = c(0.1, 10)
minppm = 10
maxppm = 200
mres = 4184
mppm = 416
pointsdata = "
10.052112405371744, 2.283698378106183
20.171602728600178, 1.8052797165878915
30.099224396434586, 1.5715748803154423
50.09267987761949, 1.3195287228519417
80.38812026903305, 1.1281218760133969
90.12665922665023, 1.0815121769656304
100.52112405371739, 1.0430967861855598
199.62996638292853, 0.8000946404902397
"
setnames(points <- fread(pointsdata, sep=",", sep2="\n"), c("x","y"))

x <- as.vector(points[,x])
y <- as.vector(points[,y])
points = list(x=x, y=y)
xfirst = head(points$x, n=1)
```

```
xlast = tail(points$x, n=1)
yfirst = head(points$y, n=1)
ylast = tail(points$y, n=1)
bstart= log(ylast/yfirst)/log(xlast/xfirst)
astart = yfirst/(xfirst^bstart)
#perform the fit
fit <- nls("y~a*x^b", start=list(a=astart,b=bstart), data=points)
#estimate min Rs/Ro
cat("\nCorrelation function coefficients")
cat("\nEstimated a\n")
cat("  ")
cat(coef(fit)["a"])
cat("\nEstimated b\n")
cat("  ")
cat(coef(fit)["b"])
cat("\n")
```

La función ***nls*** es la que utilizamos para realizar la aproximación por mínimos cuadrados de la función exponencial. Los valores calculados para ***a*** y ***b*** son los siguientes:

$$a = 5.056743$$

$$b = -0.3434871$$

Ahora, será posible calibrar el sensor sabiendo que Rs es la resistencia del sensor, que es la que varía en función de la concentración de gas y Ro es la resistencia del sensor pero calculada con una concentración de gas conocida. En nuestro caso, la concentración conocida será la concentración de CO_2 al aire libre [36]. Se despeja Ro de la siguiente manera y se aplica un ppm de 416 como concentración de CO_2.

$$Ro = \frac{Rs}{a \cdot ppm^b}\Big|_{ppm=416}$$

En la ecuación de Ro, es necesario conocer el valor de Rs. Dicho valor se obtendrá del voltaje leído en el pin de la placa de desarrollo, y calculado de la forma siguiente:

$$\frac{Rs}{R_L} = \frac{V - Vs}{Vs}$$

$$Rs = R_L \cdot \frac{1023 - Vs}{Vs}$$

Una vez obtenidos tanto Rs como Ro y los valores de ***a*** y ***b***, es posible calcular la concentración del gas, despejando la ecuación exponencial obtenida.

$$ppm = \sqrt[b]{\frac{\left(\frac{Rs}{Ro}\right)}{a}}$$

3.1.3. Microcontrolador

Para la adquisición y procesado de los datos adquiridos de los sensores volumétricos y de detección de gas, es necesario utilizar un microcontrolador. Es importante que este consuma la mínima energía posible pero sin verse afectado el rendimiento del sistema. Además de la adquisición de los datos, también se deberá transmitir vía comunicación serie, los valores adquiridos al módulo de comunicaciones para poder ser enviados a la base de datos.

Después de haber definido el número de sensores que serán necesarios para esta aplicación y el tipo de comunicación, es posible especificar los requisitos mínimos que tiene que tener esta placa de desarrollo. Estos se listan a continuación.

- 5 GPIOs modo salida para los pines Trigger del sensor HC-SR04
- 5 GPIOs modo entrada para los pines Echo del sensor HC-SR04
- 1 GPIO de entrada para el pin Aout del sensor MQ-135
- ADC
- Frecuencia de CPU mínima de 1 MHz
- 1xTimer
- RTC
- Comunicación Serie (I2C, UART o SPI)

El microcontrolador escogido para este proyecto, pertenece a la familia STM32 y la serie de uC L1, este es el STM32L152 que está integrado en una placa de desarrollo conocida como Núcleo STM32L152RE (figura 25 y 26). Este tipo de PCB tienen la ventaja de que el programador está integrado en la misma PCB. En este caso, el programador es el ST-LINK V2.1 [37].

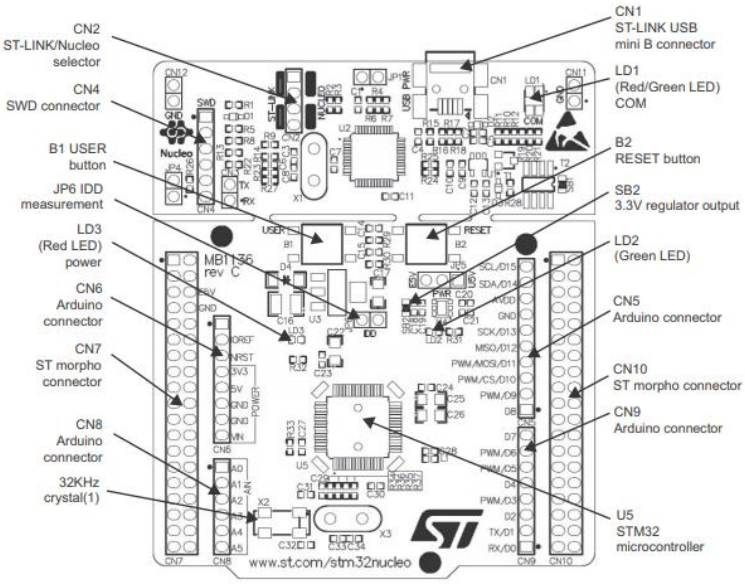


Figura 25. Cara TOP placa de desarrollo Núcleo STM32L152RE.

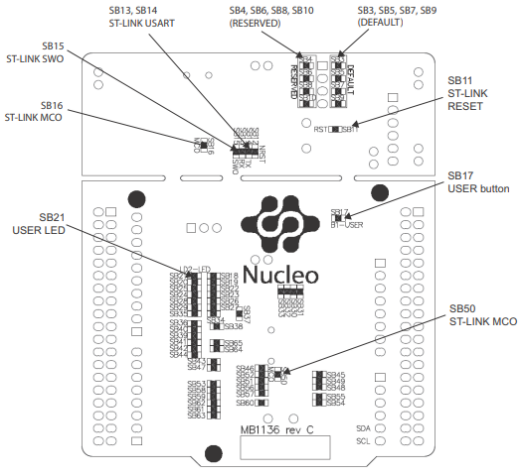


Figura 26. Cara TOP placa de desarrollo Núcleo STM32L152RE.

Esta serie de uC, utiliza un ARM Cortex-M3, que da la posibilidad de trabajar con modos de muy baja potencia sin comprometer el rendimiento. Un consumo muy bajo es necesario para poder alargar la autonomía de este módulo de la aplicación que debe alimentarse a través de una batería. Esto, juntamente con un bajo coste de mercado, la hace muy buena candidata para todo tipo de aplicaciones que no requieran de un alto nivel procesamiento. Las características principales son las siguientes [38]:

Peripheral		STM32L15xRE	STM32L15xVE	STM32L15xQE	STM32L15xZE
Flash (Kbytes)		512			
Data EEPROM (Kbytes)		16			
RAM (Kbytes)		80			
Timers	32 bit	1			
	General-purpose	6			
	Basic	2			
Communication interfaces	SPI	8(3)(1)			
	I^2S	2			
	I^2C	2			
	USART	5			
	USB	1			
GPIOs		51	83	109	115
Operational amplifiers		2			
12-bit synchronized ADC Number of channels		1 21	1 25	1 40	1 40
12-bit DAC Number of channels		2 2			
LCD (2) COM x SEG		1 4x32 or 8x28	1 4x44 or 8x40		
Comparators		2			
Capacitive sensing channels		23		33	34
Max. CPU frequency		32 MHz			
Operating voltage		1.8 V to 3.6 V (down to 1.65 V at power-down) with BOR option 1.65 V to 3.6 V without BOR option			
Operating temperatures		Ambient operating temperature: -40 °C to 85 °C / -40 °C to 105 °C Junction temperature: –40 to + 110 °C			
Packages		LQFP64	LQFP100, WLCSP104	UFBGA132	LQFP144

Figura 27. Sumario de las características principales del STM32L152RE.

Se aprecia en la figura que el STM32L152RE cumple con los requisitos necesarios para llevar a cabo esta aplicación, puesto que dispone de 51 GPIOs, comunicación serie I2C, UART y SPI, ADC de 12 bits, timer y una frecuencia de hasta 32 MHz.

Como se ha comentado anteriormente, el microcontrolador será el encargado de gestionar el módulo de sensores y por tanto, se deberá configurar con este propósito.

Tal y como se ha comentado durante la sección del sensor volumétrico, será necesario un TIMER [27] que cuente los microsegundos de los pulsos enviados por este tipo de sensor, para ello, primero se configurará el reloj del sistema a 1 MHz, ya que así, la base de tiempo será de 1µs y luego el TIMER.

Se utilizará el reloj MSI como reloj del sistema, que permite configurar la frecuencia del bus APB1 donde se encuentra el TIMER a utilizar (se ha escogido el TIMER3), tal y como muestra el diagrama de bloques de la figura 28. Además, el reloj MSI es interno, por lo que ofrece un menor consumo pero con la posibilidad de llegar hasta 1 MHz [4], frecuencia suficiente para un buen rendimiento del sistema. El lenguaje de programación escogido para programar el microcontrolador es el lenguaje C y juntamente con los manuales que el fabricante ST proporciona con las sentencias que permiten programar esta serie L1 de microcontroladores [38][39][40][41], se ha configurado el reloj de la manera siguiente:

```
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the APB busses clocks */

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_4;

  /** Initializes the CPU, AHB and APB busses clocks   */

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
```

Es importante establecer el "*MSI range*" a 4, ya que es el rango que permite alcanzar la frecuencia de 1.05 MHz, como indica el datasheet. Por otro lado, los relojes HCLK y PCLK son de los buses Advanced High-performance Bus y Advanced Peripheral Bus, respectivamente, mientras que el SYSCLK corresponde al reloj del sistema, que en este caso tendrá como referencia el MSI configurado. Estos relojes deben configurarse, puesto que son necesarios para el sistema.

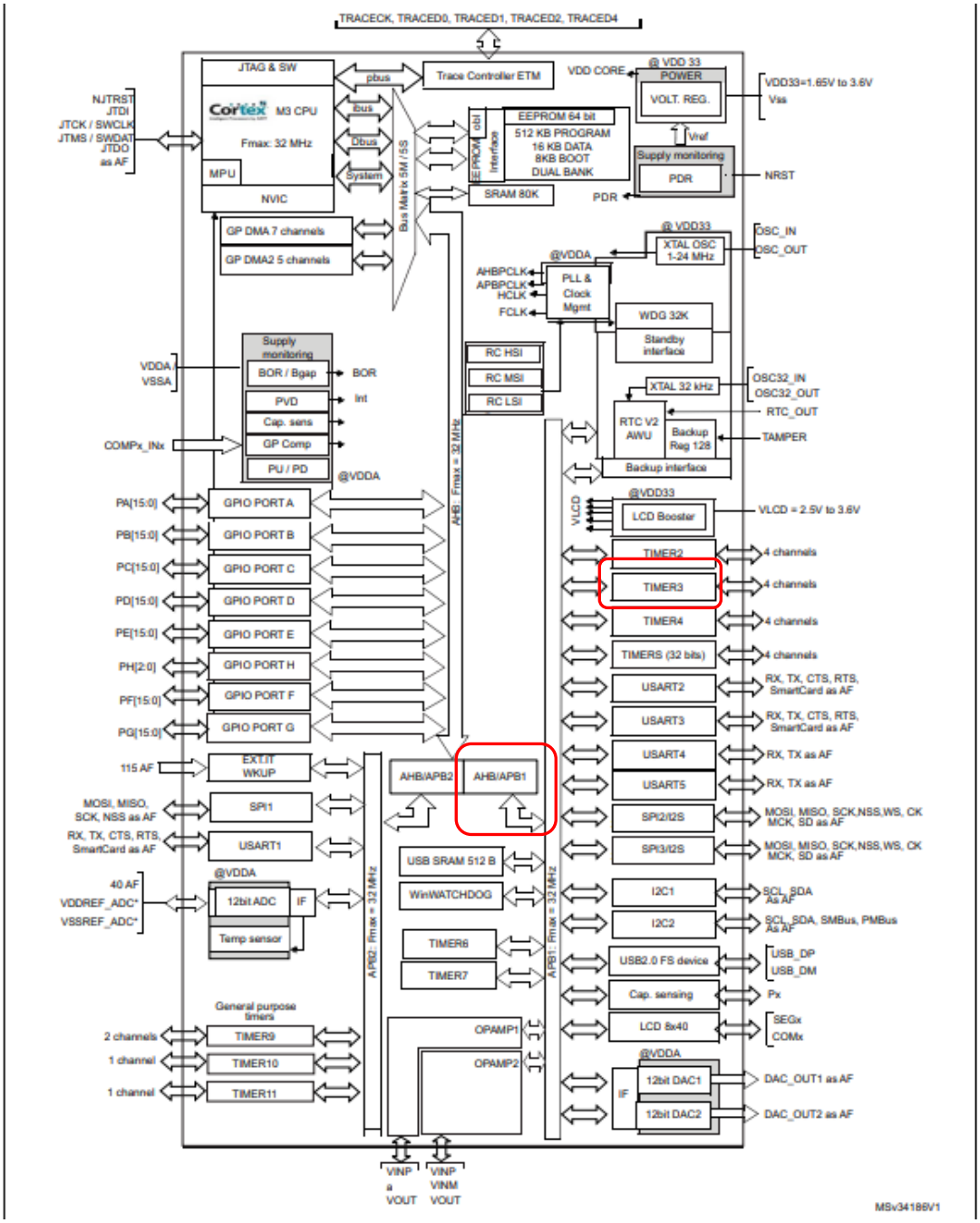


Figura 28. Diagrama de bloques del microcontrolador STM32L152xE.

Una vez configurado el reloj de la CPU, el siguiente paso es configurar el TIM3.

```
static void MX_TIM3_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 0;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 0xffff-1;
```

```
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
}
```

Ahora, una vez realizada la configuración del reloj del sistema y el TIMER, se necesitará una función que cuente microsegundos para poder contar el tiempo en el que el pulso del sensor ultrasónico está en estado alto.

```
void delay_us (uint16_t us)
{
	__HAL_TIM_SET_COUNTER(&htim3,0); //Set TIMER to 0
	while (__HAL_TIM_GET_COUNTER(&htim3)<us); //keep here until 'us'
times.
}
```

Por otro lado, el microcontrolador deberá enviar los datos a la parte transmisora del módulo de comunicaciones para que sean enviados a la parte receptora y posteriormente a la base de datos. Esta comunicación entre el STM32 y la parte transmisora se realizará vía UART y se configurará en el microcontrolador de la siguiente manera, utilizando los pines que corresponden a UART4.

```
static void MX_UART4_Init(void)
{

  huart4.Instance = UART4;
  huart4.Init.BaudRate = 9600;
  huart4.Init.WordLength = UART_WORDLENGTH_8B;
  huart4.Init.StopBits = UART_STOPBITS_1;
  huart4.Init.Parity = UART_PARITY_NONE;
  huart4.Init.Mode = UART_MODE_TX_RX;
  huart4.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart4.Init.OverSampling = UART_OVERSAMPLING_16;

}
```

Por último, este µC permite diferentes modos de bajo consumo que pueden ser útiles para esta aplicación y conseguir una mayor autonomía de la batería. A continuación, se muestra una tabla con los diferentes tipos, conocidos como *Low-power modes* [38].

Modos bajo consumo	Consumo de energía [µA]	Wake-up time [µs]	Características importantes
Sleep mode	51	0.4	- Solo CPU parada
Low-power run	11	NA	- MSI range 1 - Regulador de voltaje interno en low-power mode
Low-power sleep	4.6	46	- Regulador de voltaje interno en low-power mode - Frecuencia de reloj limitada a 32 kHz
Stop mode with RTC	1.4	7.7	- Es el modo con menor consumo manteniendo el contenido de la RAM y los registros - Todos los Vcore en off - Regulador de voltaje interno en low-power mode - PLL, MSI RC, HSI RC y HSE deshabilitados - LSI o LSE habilitados - RTC habilitado
Stop mode w/o RTC	0.56		- Ídem que Stop mode with RTC pero con el RTC deshabilitado - El dispositivo solo es capaz de despertarse con una interrupción externa
Standby with RTC	1.11	60	- Modo con menor consumo de potencia - Pérdida del contenido de RAM y registros, excepto *wakeup logic*, IWDG, RTC, LSI, LSE y registro RCC_CSR - Regulador de voltaje interno en off - Todos los Vcore en off - PLL, MSI RC, HSI RC y HSE deshabilitados - LSI o LSE habilitados - RTC habilitado
Standby w/o RTC	0.29		- Ídem que Standby mode with RTC pero con el RTC deshabilitado - El dispositivo solo es capaz de despertarse con una interrupción externa

Tabla 7. Tabla comparativa de los modos de bajo consumo del µC STM32L152

A partir de la tabla, se considerará utilizar el modo STOP, puesto que es el modo con un consumo menor pero manteniendo el contenido de la RAM y todos los registros. Para esta aplicación en la que no se podrán hacer interrupciones externas, será necesario el RTC para "despertar" al microcontrolador, por tanto, se utilizará el modo STOP + RTC.

El sistema se encontrará en este modo, todo el tiempo en el que no se deban hacer adquisiciones de los sensores para reducir el consumo y para ello, se configurará una función donde se calcula dicho tiempo y se deshabilitarán algunos de los componentes que no son deshabilitados por defecto en este modo, como son los GPIOs o el UART. También se deshabilita el *systick* para evitar que cree interrupciones que hagan despertar el µC.

```
void Enter_STOP_Mode(void)
{
/*## Configurar el timer ##*/
/*  Para generar la interrupción del RTC se debe calcular lo siguiente:
              Wake-up Time Base = (RTC_WAKEUPCLOCK_RTCCLK_DIV /(LSI))
              WakeUpCounter = Wake-up Time / Wake-up Time Base
Sabiendo que
        - RTC_WAKEUPCLOCK_RTCCLK_DIV = RTCCLK_Div16 = 16
        - LSI del STM32L152RE es de 37 kHz
        - Wake-up Time será el tiempo que decidamos poner a dormir el µC
Entonces,
        Wake-up Time Base = 16 /(37KHz) = 0.000432432 seconds
        WakeUpCounter = ~14375s/(16 /(37KHz)) = 33.242187E6 = 0x1FB3C4B */

   if (HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 0x1FB3C4B, RTC_WAKEUPCLOCK_RTCCLK_DIV16)
!= HAL_OK)
   {
        Error_Handler();
   }
// We have to suspend the systick before going into STOP mode to avoid an interrupt
that will wake the MCU up
   HAL_SuspendTick();
   //GPIOs disabled
   __HAL_RCC_GPIOA_CLK_DISABLE();
   __HAL_RCC_GPIOB_CLK_DISABLE();
   __HAL_RCC_GPIOC_CLK_DISABLE();
   __HAL_RCC_GPIOD_CLK_DISABLE();
   //UART disabled
   __HAL_RCC_UART4_CLK_DISABLE();

   //Enter to STOP mode
   HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
}
```

La sentencia *HAL_PWR_EnterSTOPMode* es la que permite entrar en el modo stop. El primer argumento es para indicar el modo en el que estará el regulador de tensión y el segundo argumento es para indicar qué instrucción se utilizará para despertar al microcontrolador, que puede ser por interrupción (WFI) o por evento (WFE), en este caso se ha escogido *WFI (Wait for interrupt)*.

Cuando el microcontrolador ha salido de este modo de bajo consumo, es necesario volver a inicializar todos los componentes que se han deshabilitado anteriormente y para esto, se crea otra función que será llamada después de la que permite entrar en el modo stop.

```
void Exit_STOP_Mode(void)
{
       SystemClock_Config();
       HAL_RTCEx_DeactivateWakeUpTimer(&hrtc);
       HAL_ResumeTick();
       MX_GPIO_Init();
       MX_UART4_Init();
       MX_ADC_Init();
       MX_RTC_Init();
}
```

Todos los códigos mostrados en esta sección, al igual que los que se mostrarán en secciones posteriores, se pueden encontrar en el capítulo 9 de Anexos, integrados en el código utilizado para esta aplicación.

3.2 Módulo de comunicación

Este módulo tiene por objetivo, realizar la comunicación entre el módulo de sensores y la base de datos a la cual se subirán los valores tomados por los sensores. Para llevar a cabo esta comunicación, podemos dividir este módulo en dos partes, la parte transmisora y la parte receptora. La parte transmisora se encuentra junto al módulo de sensores en el contenedor de reciclaje, mientras que la parte receptora se encuentra en una localización con cobertura a la red WiFi, donde se encuentra el host que contiene la base de datos. Ambas partes de este módulo serán programadas mediante el lenguaje de programación C++.

En los siguientes sub-apartados, se detalla el protocolo de comunicación utilizado y la parte transmisora y receptora de este módulo.

3.2.1. Protocolo de comunicación

Al igual que para el módulo de sensores, el consumo de energía es importante, puesto que la parte transmisora estará también alimentada por una batería. Sin embargo, es necesario que esta comunicación tenga un largo alcance, que permita llegar hasta la parte receptora.

La tecnología LoRa, introducida en la sección del estado del arte, cumple estos requisitos del sistema, ya que tiene un alcance de hasta 15 km en campo abierto y 2 km en zonas urbanas con consumos de energía extraordinariamente bajos.

LoRA es una capa física utilizada en redes LPWAN (Low-power wide-area network) para crear un enlace de comunicación de largo alcance, utilizando una modulación conocida como espectro expandido chirp y a diferencia de otros sistemas que utilizan FSK (modulación por desplazamiento de frecuencia) para lograr baja potencia, LoRa consigue mantener esa especificación pero aumentando significativamente el rango de alcance, además de una buena robustez frente a interferencias [18]. Además, una ventaja de la cobertura de LoRa respecto a otras tecnologías, es la capacidad de desplegar un *Gateway* extra sin la necesidad de depender de operadores como en el caso de NB-IoT o LTE-M, para aumentarla [42].

Esta muy enfocado al uso en aplicaciones del IoT, con redes de sensores que no necesitan enviar grandes cantidades de datos pero proporcionando un largo alcance. Algunas características importantes de esta tecnología son:

- Comunicación de largo alcance
- Bajo consumo para maximizar la vida de las baterías
- Basada en comunicación de espectro ensanchado chirp
- Baja transferencia de datos (250 bps)
- Robustez frente a interferencias

- Gran capacidad de la red para añadir nodos
- Seguridad de la red
- Comunicación one-way o two-way

Una vez conocido LoRa como la modulación, se debe presentar LoRaWAN, que es quien define el protocolo de comunicación y la arquitectura del sistema. LoRaWAN no es menos importante que LoRa, ya que es quien consigue maximizar la vida útil de la batería de un nodo, la calidad del servicio, la seguridad y la capacidad de la red. La figura siguiente muestra la estructura de una red LoRa/LoRaWAN.

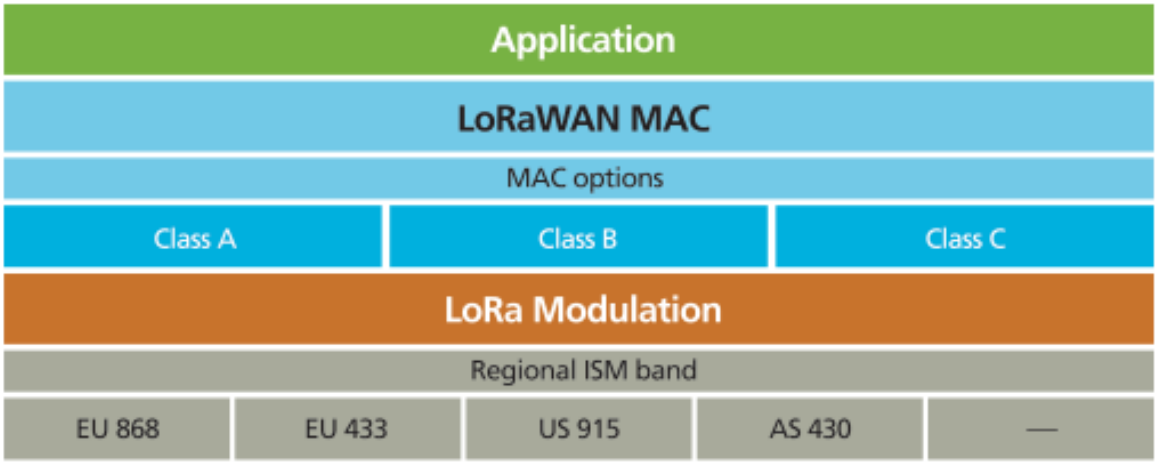


Figura 29. Estructura de una red LoRa/LoRaWAN.

Como se ha comentado, LoRa maximiza la vida de las baterías, y esto lo consigue con una arquitectura de red en estrella de largo alcance. Para que esto sea viable, el ***Gateway*** debe tener una alta capacidad para recibir datos de los nodos, esto se logra utilizando una velocidad de datos adaptativa y utilizando un transceptor multicanal [18].

Cada nodo se comunica con varios ***Gateway*** y estos con un servidor de red. Estos ***Gateways*** son transparentes entre los dispositivos finales y el servidor de red. Por tanto, en una red LoRaWAN, tenemos los siguientes elementos:

- Nodos: Sensores que adquieren los datos y los envían hasta el servidor de red. También reciben comunicación desde el servidor de aplicaciones.
- Gateway: Es el que se encarga de hacer de "puente" entre los nodos y el servidor de red.
- Servidor de red: Se conecta con los ***Gateways***, de manera inalámbrica o por cable, y se encarga de gestionar la red.
- Servidor de aplicación: Es la aplicación final donde el usuario puede elegir qué hacer con los datos recogidos por los sensores.

La Figura 30 muestra la arquitectura típica de una red LoRaWAN, donde se ven todos los elementos descritos arriba.

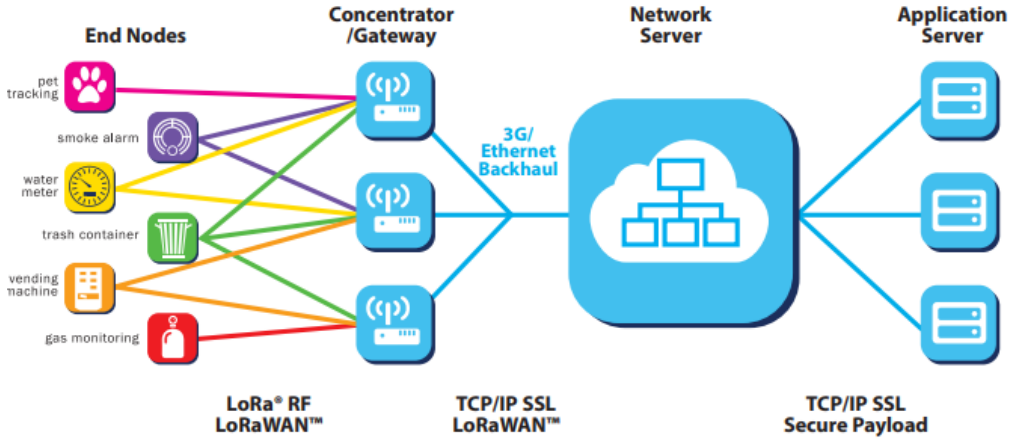


Figura 30. Arquitectura de una red LoRaWAN.

Como se ha comentado, esta tecnología utiliza como modulación el espectro ensanchado chirp, el cual está regulado por el *factor de propagación o spreading factor (SF)*. Este parámetro es una característica importante de esta tecnología, puesto que es el que decide cuántos *chirps* son enviados por segundo. La red decide el factor de propagación (SF), que puede ir entre 7 y 12, según las condiciones ambientales entre el dispositivo de comunicación y la puerta de enlace.

A medida que el SF es más bajo, se envían más *chirps* por segundo, y por lo tanto, puede codificar más datos por segundo. Un mayor SF implica menos *chirps* por segundo y por tanto, menos datos a codificar por segundo. Si se comparan ambos SF, se puede entender que a mayor SF, mayor tiempo de tiempo de transmisión (conocido como tiempo de aire o *Time on Air*), por lo que el dispositivo está funcionando durante más tiempo y consumiendo más energía. Un mayor SF tiene como beneficio un mejor muestreo de la potencia de la señal, lo que resulta en una mejor sensibilidad. Una mejor sensibilidad implica una mejor cobertura. Cada aumento del valor de este parámetro se correlaciona con un incremento de aproximadamente 2.5 dB.

Sin embargo, el consumo de energía entre el valor más alto de SF y el más bajo, puede llegar a ser de hasta 20 veces más por mensaje, con valores del orden de 2,78µWh para un SF7 y 55,1 µWh para un SF12 [43].

3.2.2. Parte transmisora

La parte transmisora es la encargada de enviar los datos de los sensores, que recibe del microcontrolador STM32, a través de la tecnología LoRa. Como se ha comentado anteriormente, la comunicación entre el microcontrolador STM32 y la parte transmisora se realiza mediante UART.

El hardware utilizado para llevar a cabo esta transmisión de los datos a la parte receptora es un módulo del fabricante Heltec, conocido como ESP32 WiFi LoRa V2. La figura siguiente muestra esta PCB.

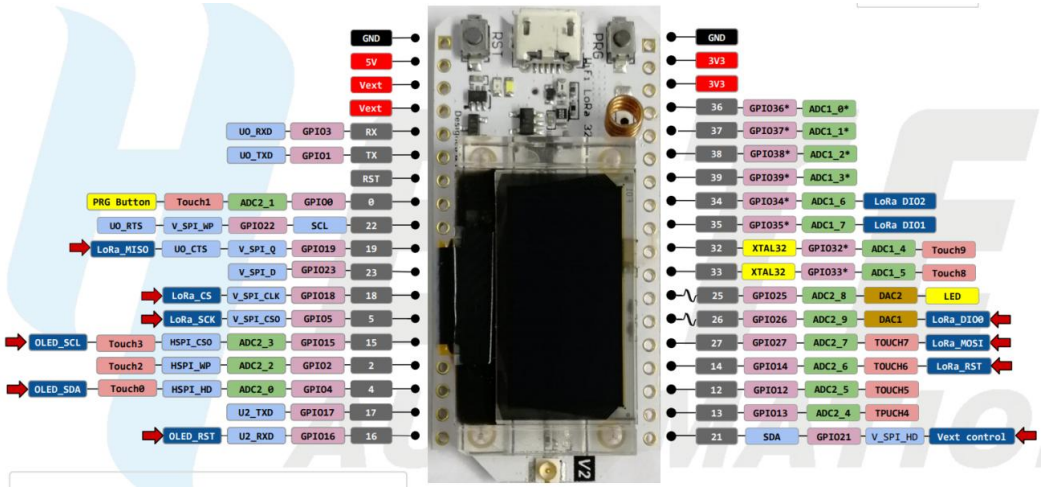


Figura 31. Heltec ESP32 WiFi LoRa V2

Este módulo permite la comunicación LoRa explicada en el punto anterior para frecuencias de 868-915 MHz, que corresponden a las permitidas por Europa y EEUU, respectivamente [18]. Para este proyecto, por tanto, se utilizará la frecuencia de 868 MHz, gracias a una antena con conector IPEX.

Para conseguir ser compatible con la tecnología LoRa, este módulo tiene un chip SX1276 del fabricante Semtech. Este chip, al igual que el resto de la familia SX, son transceptores con un módem de LoRa que proporciona comunicación de espectro ensanchado con un consumo de corriente bajo. Además, ofrece una sensibilidad de -148 dBm [44][45].

Las características principales de este módulo LoRa de Heltec son las siguientes [46]:

- MCU: ESP32(240MHz Tensilica LX6 dual-core + 1 ULP, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth)
- SX1276 LoRa chip
- Frequency bands: 868 - 915 MHz
- 18dB ± 2dB LoRa maximum output power
- Communication interfaces
    - UART x3
    - SPI x2
    - I2C x2
    - I2S x1
- FLASH 8MB (64 M-bits) SPI FLASH
- LoRa antenna interface IPEX
- Voltage operative range 3.3V - 5V

Estas características, juntamente con un bajo consumo y un precio de mercado muy bajo de hasta 11€, es una placa de desarrollo ampliamente utilizada para aplicaciones del internet de las cosas [46].

Para la programación de esta PCB, será necesario tener en cuenta el *Spreading Factor* comentado durante la descripción del protocolo LoRa. Se ha explicado que este parámetro sirve para variar el tiempo de transmisión a costa de una mejor o peor sensibilidad, dependiendo del valor de SF. En este proyecto se utilizará un valor de SF de 9, puesto que es un valor intermedio y es lo recomendado para una aplicación que no deba consumir mucha energía pero asegurando una buena recepción del mensaje [43].

```
uint8_t spreadingfactor = 9;
LoRa.setSpreadingFactor(spreadingfactor);
```

Otro aspecto a tener en cuenta, es la potencia de emisión de la comunicación LoRa, y para ello, primero se calculan las pérdidas que se podrían tener en una zona urbana, tomando el valor de sensibilidad en recepción (-138 dBm) del datasheet del chip SX1276 incorporado en esta PCBA [45] y suponiendo una distancia de 2 km entre dispositivos.

$$L = 20\log\left(\frac{4\pi d}{\lambda}\right) \text{ donde } \lambda \text{ es } c/f$$

$$L = 20\log\left(\frac{4\pi 2E3}{\frac{c}{868E6}}\right) = 97.23 \text{ dB}$$

Por defecto, estos módulos utilizan una potencia de emisión de 14 dB, por lo que la potencia en recepción será de,

$$P_{RX} = 14\text{dBm} - 97.23\text{dB} = -83.23 \text{ dBm}$$

Por tanto, vemos que esta potencia de 14dBm es suficiente. Sería posible valorar reducir esta potencia hasta el mínimo de 2 dB mediante la sentencia *Lora.setTxPower(txPower),* pero se mantendrán los 14 dBm, puesto que no se puede saber con exactitud las interferencias u obstáculos que puedan aparecer en una zona urbana, además de que el datasheet especifica que este valor es el que aporta una mayor eficiencia PA.

En cuanto a la comunicación vía UART con el STM32, será necesario leer los datos que este último ha enviado. Esto se realizará mediante la función siguiente:

```
if (Serial2.available()){

    String read = Serial2.readStringUntil('\n');
    int index = read.indexOf(separator);
    int index1 = read.indexOf(separator, index + 1);
    int index2 = read.indexOf(separator, index1 + 1);
    int index3 = read.indexOf(separator, index2 + 1);
    int index4 = read.indexOf(separator, index3 + 1);
    int index5 = read.indexOf(separator, index4 + 1);

    String Distance = read.substring(0, index);
```

```
    String Distance2 = read.substring(index + 1, index1);
    String Distance3 = read.substring(index1 + 1, index2);
    String Distance4 = read.substring(index2 + 1, index3);
    String Distance5 = read.substring(index3 + 1, index4);
    String filling_volume = read.substring(index4 + 1, index5);
    String CO2_sensor = read.substring(index5 + 1);
}
```

Se guarda en la variable ***index*** la posición en la que se encuentra el separador de los datos, que en este caso es una coma [,]. Luego, mediante la función *substring*, se extrae la parte deseada para almacenarla en cada variable.

Una vez obtenidos los valores para cada variable, se enviarán los datos, mediante la tecnología LoRa con los comandos siguientes:

```
Heltec.LoRa.beginPacket();
Heltec.LoRa.print(var);
Heltec.LoRa.endPacket();
```

3.2.3. Parte receptora

La parte receptora es la que se encarga de recibir los datos del módulo de sensores que han sido enviados por la parte transmisora. Este módulo estará en una ubicación con acceso a la red WiFi en la que se encuentra el host que contiene la base de datos.
Esta parte receptora está formada por dos módulos, uno de ellos recibe los datos del ESP32 de la parte transmisora a través de la comunicación LoRa, mientras que el segundo es el encargado de subir los datos, a través de WiFi, a la base de datos SQL. El primero es el mismo tipo de módulo Heltec ESP32 WiFi LoRa V2 usado en la parte transmisora, con las mismas características ya explicadas en la sección 2.2.2. El segundo módulo es un NodeMCU V3 ESP8266 del fabricante Lolin, con un chip transceptor WiFi [47].

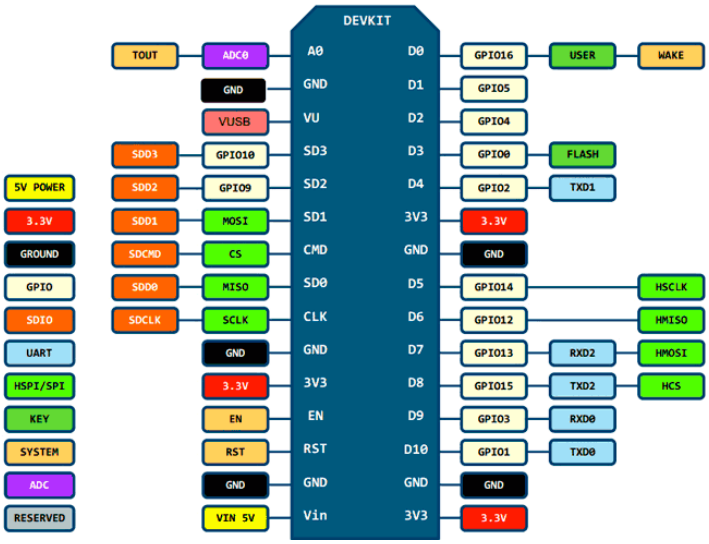
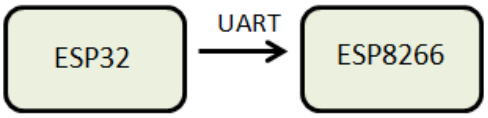


Figura 32. Pinout del módulo NodeMCU v3 ESP8266 WiFi.

Es un módulo con un muy bajo coste de mercado (es posible encontrarlo por 6€) y una fácil implementación. Es uno de los módulos más ampliamente usado para aplicaciones que requieren de una conexión WiFi. Como características principales podemos destacar las siguientes:

- MCU: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage 3.3 V
- GPIO: 17
    - Digital pin: 16
    - Analog pin: 1
- UART x1
- SPI x1
- I2C x1
- Flash Memory 4MB
- SRAM: 64 kB
- Clock speed: 80 MHz
- WiFi: IEEE 802.11 b/g/n:
    - Integrated TR switch, balun, LNA, power amplifier and matching network
    - WEP or WPA/WPA2 authentication or open networks.

En cuanto a la comunicación entre estos dos módulos de la parte receptora, ESP32 y ESP8266, también se ha utilizado la comunicación UART.



Después de haber extraído los datos enviados por el ESP32 de la parte transmisora, tal y como se ha mostrado en la sección anterior, el código utilizado para enviarlos al ESP8266, es el siguiente:

```
int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packets
    // read packet
    while (LoRa.available()) {
      String read = LoRa.readStringUntil('\n');
      int index = read.indexOf(separator);
      int index1 = read.indexOf(separator, index + 1);
      int index2 = read.indexOf(separator, index1 + 1);
      int index3 = read.indexOf(separator, index2 + 1);
      int index4 = read.indexOf(separator, index3 + 1);
      int index5 = read.indexOf(separator, index4 + 1);
      int index6 = read.indexOf(separator, index5 + 1);
      int index7 = read.indexOf(separator, index6 + 1);
```

```
      //int index8 = read.indexOf(separator, index7 + 1);
      ID = read.substring(0, index);
      Distance = read.substring(index + 1, index1);
      Distance2 = read.substring(index1 + 1, index2);
      Distance3 = read.substring(index2 + 1, index3);
      Distance4 = read.substring(index3 + 1, index4);
      Distance5 = read.substring(index4 + 1, index5);
      filling_volume = read.substring(index5 + 1, index6);
      CO2_sensor = read.substring(index6 +1, index7);
      localizacion = read.substring(index7 +1);

      String sent = (ID + separator + Distance + separator + Distance2 +
separator + Distance3 + separator + Distance4 + separator + Distance5 + s
eparator + filling_volume + separator + CO2_sensor + separator + localiza
cion + '\n');
      Serial2.print(sent);
      delay(5000);
    }
}
```

3.3. Batería

Como se ha comentado varias veces, la batería es un elemento importante en esta aplicación, puesto que deberá alimentar al módulo de sensores y a la parte transmisora del de comunicaciones. Ahora que se han detallado los diferentes módulos, es posible dimensionar la batería y para ello, en esta sección se calcularán los consumos de cada uno de los elementos del sistema. El mínimo voltaje a suministrar por la batería será de 5V, puesto que es el valor de voltaje que requieren los sensores ultrasónicos y de detección de gas. Sin embargo, los microcontroladores ESP32 y STM32L152 necesitan una alimentación de 3.3V, por lo que será necesario utilizar reguladores de tensión para poder suministrar dos voltajes diferentes.

Debido a la alta cantidad de hardware que deberá alimentar, un total de ocho elementos (5 sensores ultrasónicos, 1 sensor de detección de gas, la placa de desarrollo Núcleo STM32L152 y el ESP32 de la parte transmisora del módulo de comunicaciones), será necesario evitar que el consumo del sistema provoque una baja autonomía de la batería, y por tanto, se debe plantear una configuración que maximice la duración de esta.

Esta configuración se puede observar en la figura siguiente, donde por simplificar, se ha dibujado tan solo uno de los cinco sensores HC-SR04, aunque el resto debe conectarse de la misma manera. También se ha dibujado el sensor de detección de gas y el ESP32. La figura muestra la conexión de la batería, representada como BAT1, con los reguladores de tensión MCP1802T-33 y MCP1802T-50, basados en la tecnología CMOS, que proporcionan 3V3 y 5V, respectivamente [48], que alimentarán los sensores (MQ-135 y HC-SR04) y los microcontroladores (STM32L152 y ESP32). Cada uno de ellos estará conectado por el pin de tierra al drenador de un transistor MOSFET

IRL540, también basado en la tecnología CMOS [49], mientras que el terminal de puerta de este transistor estará conectado al µC STM32 (representado como la batería BAT2 de 3V) y el terminal de fuente a tierra. Esta configuración permitirá utilizar el NMOS como un conmutador, aplicando un 0 o 1 lógico desde el microcontrolador, para permitir alimentar los sensores y el ESP32 cuando sea necesario, y así poder reducir el consumo del sistema lo máximo posible.

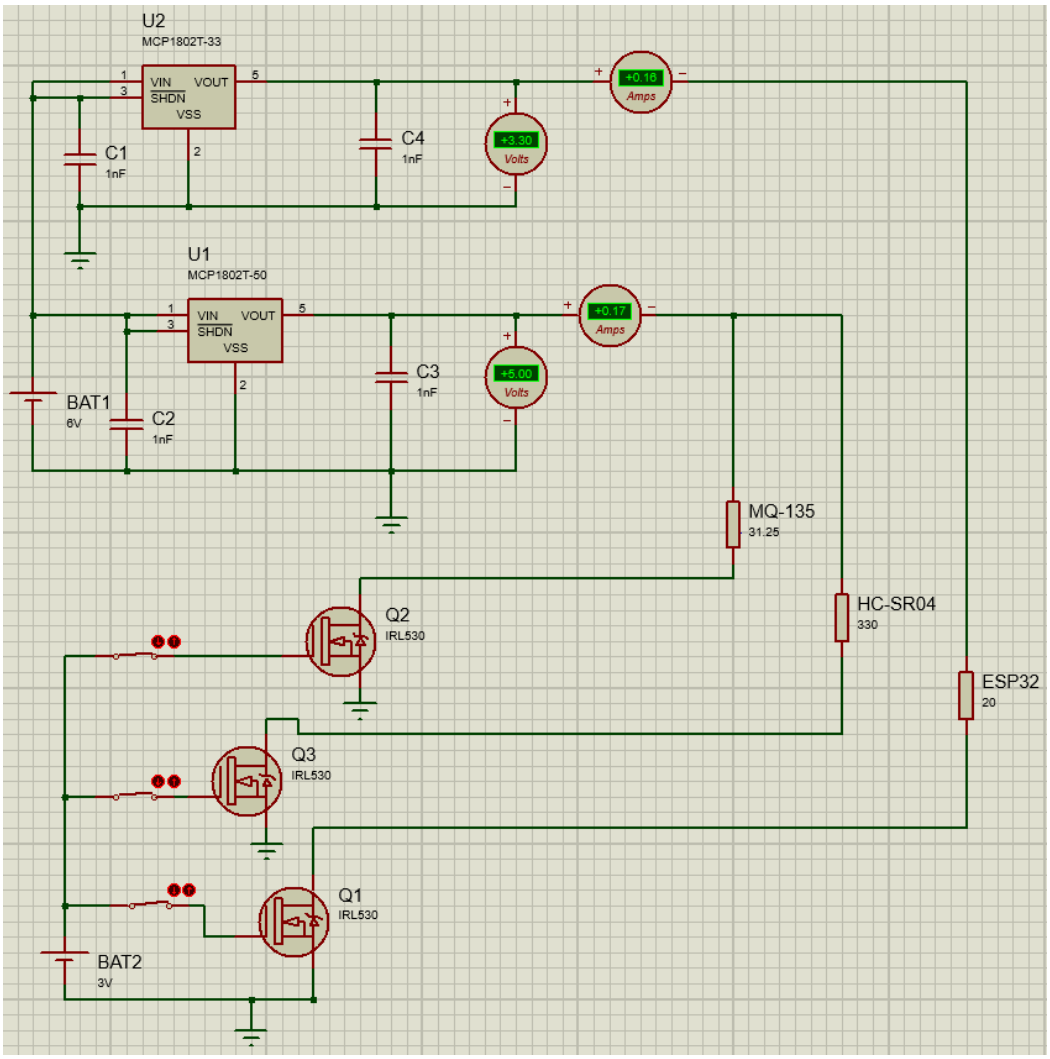


Figura 33. Conexión batería con los LDO MCP1802T-33 y MCP1802T-50 y los NMOS IRL540 como solución para reducir el consumo del sistema.

Un aspecto importante antes de realizar los cálculos, es establecer el período de tiempo adecuado entre cada adquisición de datos, se establece pues, un tiempo de cuatro horas, ya que no es necesario monitorizar el contenedor en tiempo real.

A continuación se presentan los cálculos del consumo de energía para cada elemento del sistema, teniendo en cuenta el escenario planteado.

3.3.1. Cálculo consumo sensor MQ-135

Este sensor se mantendrá encendido durante 21 segundos cada cuatro horas para realizar la adquisición del valor de CO_2. Los primeros 20 segundos son de pre-heating, mientras que el segundo restante, es para adquirir el valor y volver a apagarlo.

Según el datasheet de este sensor, se especifica que el consumo es menor a 800 mW [33], tomando este valor y sabiendo que el voltaje de trabajo de este sensor es 5 V, es sencillo calcular la corriente que pasará por él, haciendo uso de la fórmula P = V·I. Por tanto, el consumo de este sensor será de,

$$I = P / V = 800E\text{-}3W / 5V = 160 \text{ mA}$$

Con este consumo, se calcula el consumo por hora como sigue:

$$160 \text{ mA} * \left(\frac{21 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}}\right) = 160 \text{ mA} * 1.458E\text{-}3 = \mathbf{233.333\ \mu A}$$

3.3.2 Cálculo consumo ESP32 (Parte transmisora del módulo de comunicaciones)

Para el cálculo de consumo de esta PCBA, se debe tener en cuenta lo establecido durante la sección 3.2.2. correspondiente a la parte transmisora, en términos de potencia de emisión de la comunicación LoRa. Debido a que la hoja de especificaciones de Heltec para esta PCBA no especifica qué consumo tiene para una potencia de 14 dBm, se utilizará el consumo de 15 dBm que sí está especificado [46].

Mantendremos este microcontrolador durante 3 segundos cada vez que se active el sistema, es decir, estará trabajando 3 segundos cada cuatro horas.

Entonces, teniendo que $\frac{3 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}}$ = 208.333E-6, podemos calcular los siguiente:

Consumo = Consumo 3V3 powered * tiempo actividad + Consumo 15 dB LoRa emission * tiempo actividad

150 mA (3V3 powered) * 208.333E-6 + 110 mA (20dB LoRa emission) * 208.333E-6

Consumo = **54.166 µA**

3.3.3. Cálculo consumo sensor HC-SR04

Este sensor estará encendido, en modo trabajo, durante 1 segundo cada vez que se adquieran datos, es decir, 1 segundos cada cuatro horas. Según especificaciones del datasheet, su consumo en modo de trabajo es de 15 mA [13].

Este tiempo establecido de 1 segundo, es más que suficiente para asegurar el correcto funcionamiento del sensor, puesto que el tiempo de trabajo es alrededor de 39 milisegundos cada vez que se adquiere un dato de distancia, ya que el pulso enviado al

pin de Trigger dura 10 µs, el pulso enviado por el transmisor es de 200 µs (8 pulsos de 40 kHz) y el pulso enviado por el pin Echo al µC será de como máximo 38 milisegundos en caso de no encontrar un obstáculo.

Con esta configuración, podemos suponer que el sensor no estará nunca en modo standby. Por tanto,

$$\frac{2 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}} = 138.888\text{E-}6$$

15 mA * 138.888E-6 = 2.083 µA * 5 sensores = **10.415 µA**

3.3.4. Cálculo consumo STM32L152

Como se ha comentado en la sección 3.1.3., el microcontrolador entrará en modo STOP siempre que no se adquieran datos de los sensores. El tiempo en el que el sistema estará en modo *Run* para realizar la adquisición de los datos, será la suma de los tiempos en los que los sensores y el ESP32 estarán encendidos, es decir 21 + 3 + 1 = 25 segundos. Por tanto, todo el resto del tiempo del ciclo establecido de cuatro horas, se encontrará en el modo STOP, con lo que el consumo del µC se calculará de la siguiente manera.

$$\text{Consumo } \textit{Run mode} = 195 \text{ µA} * \frac{25 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}} = 0.339 \text{ µA}$$

$$\text{Consumo } \textit{STOP mode} = 1.4 \text{ µA} * \frac{14375 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}} = 1.398 \text{ µA}$$

Sin embargo, hay que tener en cuenta los periféricos, puesto que también consumen energía y deben ser considerados. Dependiendo del modo en el que se encuentre el microcontrolador, los periféricos a considerar no serán los mismos. Para el caso *Run mode*, estos serán la UART (4.8 µA) , el TIM3 (7 µA), el ADC (6.2 µA), el RTC (0.4 µA) y todos los GPIOx (4*4.1 µA = 16.4 µA). Por otro lado, durante el modo STOP solo el RTC estará operativo, puesto que los periféricos que no se deshabilitan por defecto al entrar en este modo, se hará de manera manual para reducir aún más el consumo.

$$\text{Consumo } \textit{Run mode} = 0.339 \text{ µA} + ((4.8 + 7 + 6.2 + 0.4 + 16.4) \text{ µA} * \frac{25 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}}) = 0.339 + 0.06 = 0.399 \text{ µA}$$

$$\text{Consumo } \textit{STOP mode} = 1.398 \text{ µA} + 0.4 \text{ µA} * \frac{14375 \text{ segundos} * 1 \text{ vez/4h}}{3600 \text{ segundos}} = 1.797 \text{ µA}$$

Consumo total = Consumo *Run mode* + Consumo *STOP mode*

Consumo total = **2.196 µA**

3.3.5. Consumos de los LDOs y MOSFETs

Por último, también se debe tener en cuenta el consumo de los LDO y los MOSFET. Los MCP1802 en situación de quiescencia tienen una Iq = 25 µA, mientras que los MOSFET IRL540 tienen un consumo de como máximo 25 µA cuando V_{GS} = 0V.

La tabla siguiente muestra el consumo total del sistema.

Elemento	Consumo
MQ-135	233.333 µA
ESP32 3'3V powered	31.249 µA
ESP32 emitiendo con LoRa durante 3 segundos cada cuatro horas	22.916 µA
5 sensores HC-SR04 en modo trabajo durante 39 milisegundos cada cuatro horas	10.415 µA
STM32L152 @ 1MHz	2.196 µA
2 x LDO MCP1802	50 µA
7 x IRL540	175 µA
TOTAL	**525.109 µA**

Tabla 8. Consumos del sistema para una adquisición de datos cada cuatro horas.

Con los datos de consumo conocidos, el siguiente paso es calcular la duración de la batería. Como se ha comentado, el mínimo voltaje que deberá suministrar la batería será de 5V, por lo que se ha escogido una batería compuesta por 4 pilas alcalinas tipo AA de 1.5V, que suministrarán un total de 6V y 3000 mAh.

Duración batería = 3000 mAh / 525.109 µA = 5713 horas

es decir,

5713 horas = **238 días**

Una vez calculada la autonomía que tendrá este sistema, es necesario incluir una rutina de recambio de baterías por parte de la empresa encargada de la recogida de residuos urbanos. Aunque de manera teórica se hayan obtenido los días que durará la batería del sistema, será adecuado incluir un reporte diario del estado en el que se encuentra. Para esto, se medirá el nivel de voltaje en el que se encuentra la batería antes de realizar la adquisición de los valores de los sensores, puesto que al suministrar corriente al sistema puede variar el valor de voltaje de la pila y no corresponderá al valor correcto.

La figura 34 muestra la curva de descarga de una pila alcalina, en este caso de la compañía Duracell, tipo AA [50]. Se aprecia que, a medida que el consumo de corriente es menor, la curva se va suavizando y por tanto, para un consumo aun menor como son los 525 µA calculados para esta aplicación, se puede considerar aproximarla a una recta y utilizar la ecuación de la recta para calcular el porcentaje de descarga de la pila. Otro aspecto importante es el nivel de voltaje al que se encuentra la pila cuando está descargada, se puede ver en la gráfica que este valor de voltaje es de 0.9 V. Por tanto, se considera que un valor de 1.5 V es una descarga del 0%, mientras que 0.9 V corresponde al 100% de la descarga de la pila.

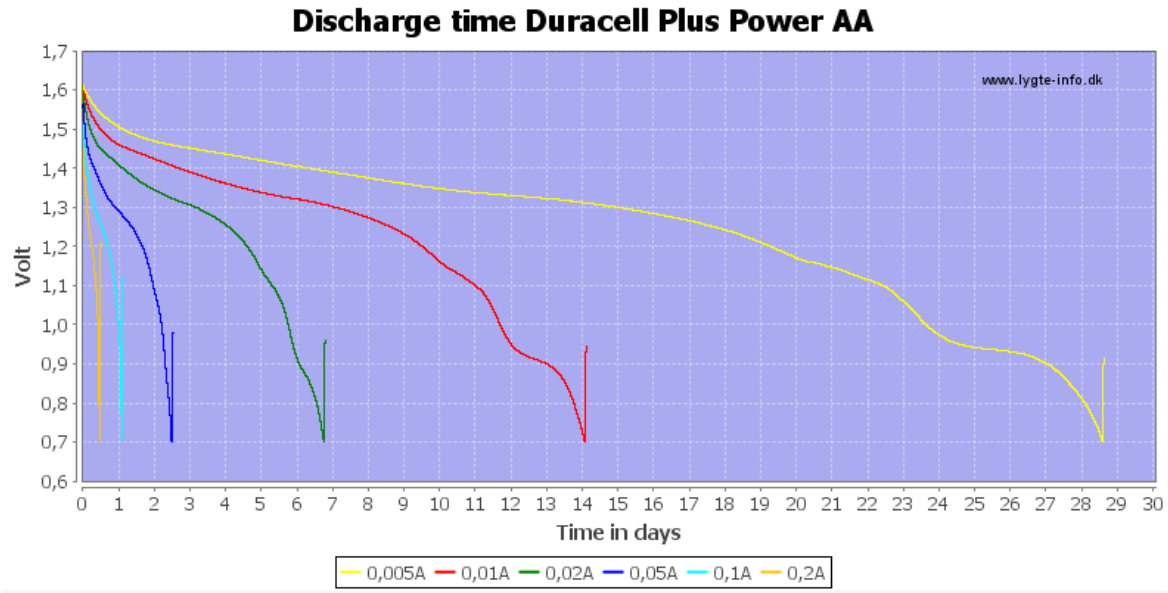


Figura 34. Curva descarga pila alcalina AA para diferentes consumos de corrientes [50].

Ecuación de la recta,

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

donde $x_1, y_1 \ / \ x_2, y_2$ son las ordenadas y abscisas respectivas de los puntos A y B de la recta $\overline{AB}$. Por otro lado, 'm' es la pendiente de la recta $\overline{AB}$.

Como se ha comentado anteriormente, la batería consta de cuatro pilas que proporcionan un voltaje de 6V. Este voltaje no es compatible con las entradas del microcontrolador, por lo que se debe ajustar este nivel de voltaje para poder leerlo. Según el datasheet del STM32L152, la tensión de referencia del ADC del microcontrolador, que es el tipo de pin que deberá ser usado para medir el nivel de la batería, es la tensión de alimentación [38], que para el caso de este proyecto es de 3.3V. Por tanto, se utilizará un divisor de tensión que proporcione un voltaje de 3.3V como el de la figura 35. Los valores de las resistencias son considerablemente elevados para evitar un consumo de potencia innecesario.

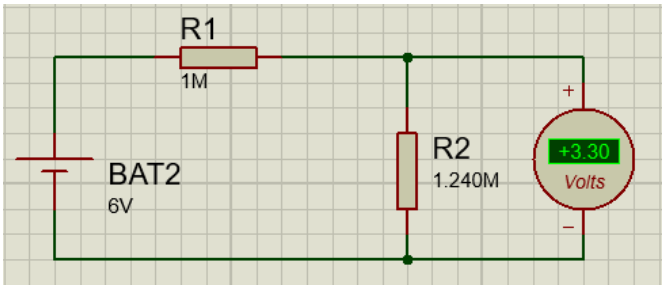


Figura 35. Divisor de tensión que permite leer el nivel de voltaje de la batería por parte del µC.

Debido a que la curva de descarga se conoce para una pila de 1.5V, se deberá escalar los 3.3V a este voltaje de 1.5V, esto nos dará un factor de 2.2, que sale de dividir 3.3V/1.5V = 2.2. Por tanto, el voltaje leído se dividirá entre 2.2.

Así pues, y teniendo en cuenta que el valor de voltaje de la batería será conocido cada vez que se adquiera, se calculará el porcentaje de descarga aislando la ecuación de la recta de la manera siguiente,

$x = \frac{y-y_1}{m} + x_1$, donde 'y' es el valor de voltaje leído dividido el factor 2.2.

A la base de datos, se reportará un 1 o un 0, en función de si el resultado del porcentaje de descarga supera el 80%, el valor 1 significará que este umbral de 80% ha sido superado, mientras que un 0 equivale a un porcentaje de descarga menor. Este umbral no deberá ser mayor a 80%, ya que al ser un cálculo aproximado, es necesario asegurarse de que el sistema no se quedará sin batería.

3.4. Conexionado del sistema

3.4.1. Conexionado módulo de sensores

La conexión del módulo de sensores con los cinco sensores ultrasónicos, el sensor de detección de gas CO_2, la placa de desarrollo y la batería, se representa en la figura 36. El cableado de color marrón, corresponde al pin ***Echo*** de los sensores HC-SR04, el verde al pin ***Trigger***, el morado es para la salida analógica del sensor MQ-135, el naranja al terminal de puerta del MOSFET IRL540 y por último, el negro corresponde a GND y el rojo a Vcc. Se ha puesto un solo MOSFET IRL540 a modo de ejemplo de cómo deben conectarse el resto.

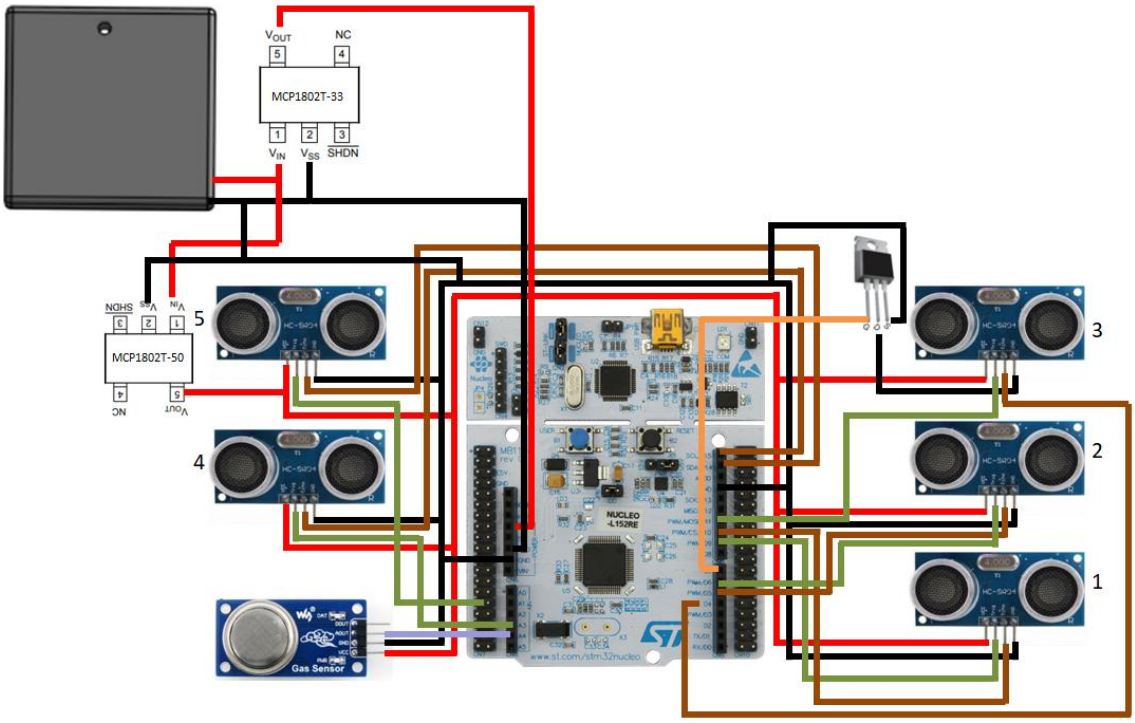


Figura 36. Conexión completa del módulo de sensores.

3.4.2. Conexionado módulo de comunicación

La comunicación de la parte transmisora con el módulo de sensores se puede ver en la figura siguiente. La comunicación es UART y por tanto, los pines TX se conectarán con los pines RX. El STM32 utiliza como TX la entrada PC_10 del uC e irá conectado al pin 16 (TX) del ESP32, mientras que el RX del STM32, corresponde a la entrada PC_11, que estará conectada al pin 17 (RX) del segundo. También se observa la conexión con el MOSFET IRL540 y el LDO.

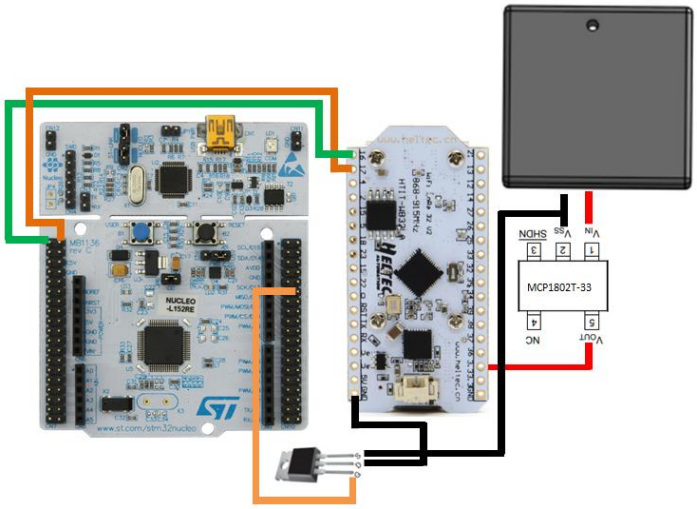


Figura 37. Conexión entre el uC STM32L152 con el ESP32, vía comunicación UART.

Para un mayor entendimiento y a modo de resumen, se listan en una tabla las conexiones representadas en la figura 36 y 37. También se tiene en cuenta el pin de lectura del nivel de tensión de la batería definido anteriormente.

STM32L152	Elemento	
PC_7 (GPIO)	**Trigger pin**	**HC-SR04 (1)**
PB_10 (GPIO)		**HC-SR04 (2)**
PA_7 (GPIO)		**HC-SR04 (3)**
PB_0 (GPIO)		**HC-SR04 (4)**
PA_1 (GPIO)		**HC-SR04 (5)**
PB_6 (GPIO)	**Echo pin**	**HC-SR04 (1)**
PB_4 (GPIO)		**HC-SR04 (2)**
PB_5 (GPIO)		**HC-SR04 (3)**
PB_8 (GPIO)		**HC-SR04 (4)**
PB_9 (GPIO)		**HC-SR04 (5)**
PC_1 (ADC)	**Salida analógica Aout**	**MQ-135**
PA_4 (GPIO)	**(Gate) Puerta**	**MOSFET 1 (HC-SR04 (1))**
PA_6 (GPIO)		**MOSFET 2 (HC-SR04 (2))**
PA_8 (GPIO)		**MOSFET 3 (HC-SR04 (3))**
PA_9 (GPIO)		**MOSFET 4 (HC-SR04 (4))**
PA_10 (GPIO)		**MOSFET 5 (HC-SR04 (5))**
PA_11 (GPIO)		**MOSFET 6 (MQ-135)**

PA_12 (GPIO)		**MOSFET 7 (ESP32)**
PC_11 (UART_RX)	**TX (Pin 17)**	**ESP32**
PC_10 (UART_TX)	**RX (Pin 16)**	
PC_0 (ADC)	**Vcc**	**Batería**

Tabla 9. Conexión pines módulo de sensores y parte transmisora con µC STM32

La comunicación para la parte receptora, entre el módulo ESP8266 y el ESP32 se realiza mediante UART, ya que no es necesaria una alta velocidad de comunicación y se prioriza la simplicidad de las conexiones. El cableado entre los dos módulos se puede ver en la figura 38, donde el TX del ESP8266 se conecta con el RX (pin 16) del ESP32 y el TX (pin 17) de este con el RX del ESP8266.

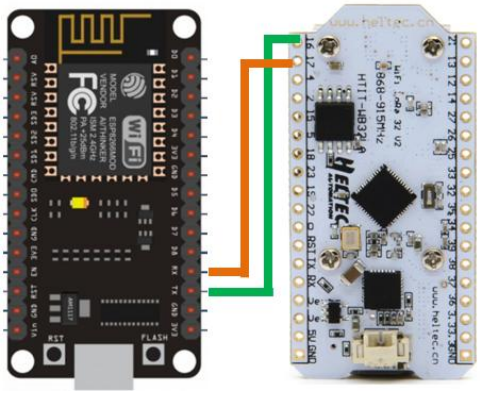


Figura 38. Conexión entre los módulos Heltec ESP32 y nodeMCU v3 ESP8266 vía UART.

3.5. Soporte y tapa para el módulo de sensores y la parte transmisora del módulo de comunicaciones.

Para la implementación de este sistema, es necesario poder sujetar los diferentes elementos del módulo de sensores en la parte superior del contenedor, por lo que se debe utilizar un soporte que permita instalarlo. La siguiente imagen muestra este soporte con los diferentes sensores ultrasónicos en la parte derecha, la placa de desarrollo en color rojo, el ESP32 en amarillo, el sensor de detección de gas MQ135 en color azul y por último en la parte izquierda la batería.

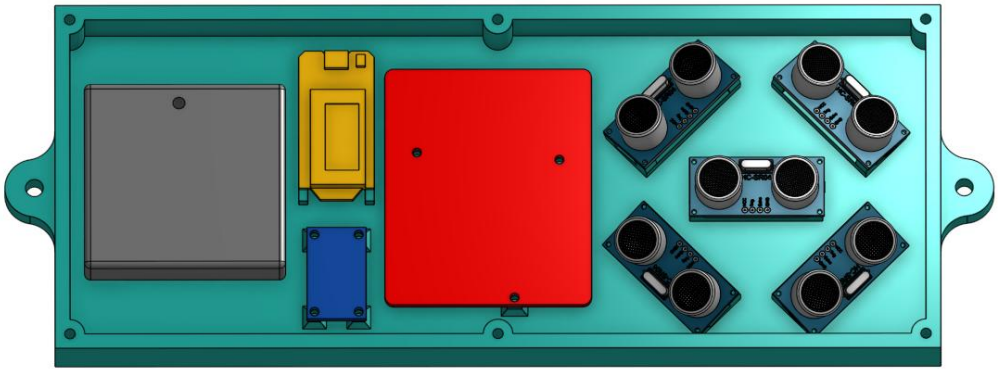


Figura 39. Soporte para la instalación del módulo de sensores y la parte transmisora del módulo de comunicaciones.

Los dos agujeros de los extremos, son los que se utilizan para anclar el soporte al contenedor, mientras que los seis agujeros del perfil, son los que se anclarán a una tapa. Debido a que el módulo de sensores estará expuesto dentro del contenedor a posibles golpes ocasionados por el lanzamiento de los residuos urbanos por parte del usuario, se decide diseñar una tapa que evite cualquier problema. Este diseño se muestra en la figura siguiente, donde se aprecia que la tapa es redondeada, precisamente para evitar que los golpes ocasionen una presión muy fuerte sobre el módulo.

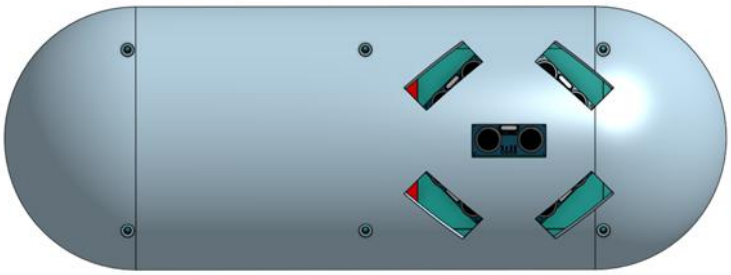


Figura 40. Tapa para la instalación del módulo de sensores y la parte transmisora del módulo de comunicaciones.

En cuanto al material, tanto del soporte como de la tapa, está pensado para poder ser impreso en una impresora 3D, este material es el conocido ácido poliláctico (PLA). El *drawing* del diseño se puede encontrar en el anexo 9.8.

3.6. Base de Datos

Una base de datos es necesaria para el almacenamiento de un conjunto de datos y poder así, ser tratados a posteriori. En este proyecto se usará para subir los datos adquiridos por el módulo de sensores, de los valores de cada uno de los sensores volumétricos y detección de gas, además de otros tipos de datos que se deseen y que más adelante se detallarán.

Con el objetivo de implementar la base de datos, se utiliza una herramienta de libre distribución, escrita con código PHP, llamada ***phpMyAdmin***, que permite la gestión de una base de datos MySQL o MariaDB a través de un navegador web [51]. Una gran ventaja de esta herramienta es la sencillez para la gestión de los datos, tablas, columnas, usuarios, permisos, etc., además de poder exportar los datos en muchos tipos de formatos.

Otras características, son la administración de múltiples servidores, la posibilidad de hacer consultas Query-by-Example (QBE) o la búsqueda en una base de datos o subconjunto de esta.

Se usará XAMPP para instalar la base de datos y el resto de software necesario. XAMPP es una distribución de Apache que contiene el servidor Apache, la base de datos MySQL y los intérpretes de los lenguajes de script PHP y Perl [52].

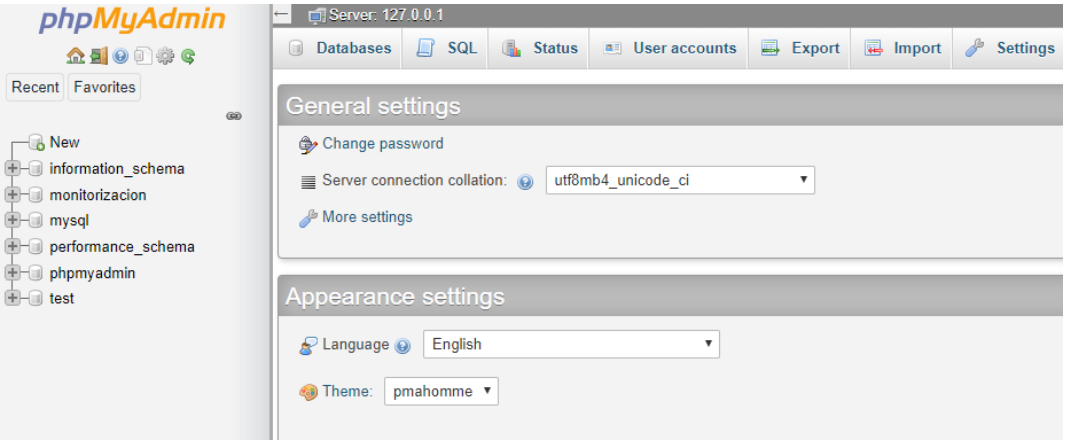


Figura 41. Interface de la herramienta phpMyAdmin.

Como se ha comentado, los datos de los sensores y otros parámetros se almacenarán en la base de datos y es importante respetar un orden a la hora de subirlos, ya que los campos se configuran de manera diferente dependiendo del tipo de dato que almacenará. Además, este orden también es importante para poder procesar los datos de manera correcta, puesto que la aplicación generada para este propósito, necesita conocer previamente la posición en que se encuentra cada dato.

A continuación se detallan los campos creados en la tabla que almacenará los datos, junto con el tipo de dato. Este orden es el que se debe respetar durante toda la comunicación desde el módulo de sensores hasta la BBDD. Los tipos de dato serán TIMESTAMP para la fecha, VARCHAR e INT.

"Campo1,Campo2,Campo3,Campo4,Campo5,Campo6,Campo7,Campo8,Campo9,Campo10,Campo11"

Nº campo	Tipo de Dato	Concepto
Campo1	TIMESTAMP	Fecha
Campo2	VARCHAR	ID del contenedor
Campo3	INT	Distancia sensor ultrasónico 1
Campo4	INT	Distancia sensor ultrasónico 2
Campo5	INT	Distancia sensor ultrasónico 3
Campo6	INT	Distancia sensor ultrasónico 4
Campo7	INT	Distancia sensor ultrasónico 5
Campo8	INT	Volumen: contador de los 5 sensores ultrasónicos
Campo9	INT	Nivel de CO2
Campo10	VARCHAR	Localización
Campo11	INT	Nivel de batería

Tabla 10. Detalle de los campos donde se almacenan los datos en la BBDD.

Para subir los datos a la base de datos, el módulo ESP8266 WiFi, deberá llamar a un archivo escrito en lenguaje PHP, que será el que permitirá registrar las entradas de los sensores. Este código se puede encontrar en el Anexo 9.5. Las funciones principales del código son las siguientes:

- ***mysqli_connect (host, user, password, database)*** -> encargado de conectarse a la base de datos.
- ***mysqli_query*** -> es el encargado de realizar un insert en la base de datos.

3.6.1 Exportar datos

Para poder procesar los datos, será necesario exportarlos desde la BBDD. El formato en el que nos interesa exportar es en CSV, ya que es un formato bastante extendido y que representa los datos en forma de tabla y separados por comas.

En la interface de la herramienta phpMyAdmin, es posible encontrar en la barra superior el menú Export que nos permitirá escoger cómo exportar los datos. El único requerimiento a parte de exportar en formato CSV, es eliminar el carácter dobles comillas (") en el campo *Columns enclosed with* que encierra el dato de cada columna.

3.7. Aplicación para procesamiento de datos

Después de haber adquirido los datos de los diferentes sensores, haber sido enviados por el módulo de comunicaciones desde la parte transmisora a la receptora y haber sido almacenados en la base de datos, el siguiente paso es procesar dichos datos para así, poder sacar conclusiones y valorar, en este caso, si un contenedor debe ser o no recogido/vaciado.

Para este propósito se utiliza una plataforma y entorno de desarrollo basado en un lenguaje de programación gráfico llamado LabView [53]. Este lenguaje de programación es propiedad de la empresa National Instruments (NI), la cual es propietaria de otros conocidos software como TestStand, que se utiliza para la creación de secuencias de test.

Figura 42. Logo de National Instruments para el lenguaje de programación LabView.

Con este lenguaje de programación se ha creado una aplicación que procesa los datos exportados, muestra los resultados y exporta a un archivo las localizaciones de los contenedores que deben ser incluidos en la ruta de los camiones de reciclaje. Además, muestra los contenedores que se encuentran en una área de Barcelona, indicando de manera más visual si un contenedor ha sido o no incluido en la ruta de recogida. Los contenedores que deben ser recogidos se muestran en color rojo, puesto que los niveles de llenado o concentración de gas han superado el límite establecido, mientras que los que no deben ser recogidos se muestran en color verde. También, muestra los contenedores en los que el sistema ha alcanzado el umbral preestablecido por el que se considera que tienen una batería baja y debe ser reemplazada.

A modo de resumen, los resultados que nos proporciona esta aplicación son los siguientes:

- Contenedores de reciclaje que deben ser incluidos en la ruta.
- Contenedores de reciclaje que deben ser excluidos de la ruta.
- Archivo con la localización de los contenedores incluidos en la ruta.
- Mapa de los contenedores de un área de Barcelona.
- Contenedores con batería baja.

Una ventaja de esta aplicación es que no es necesario conocer el número de contenedores previamente, por lo que hace posible que se incluyan o excluyan contenedores de forma dinámica, es decir, cualquier contenedor puede ser movido, reemplazado, añadido o incluso eliminado de la red de contenedores de reciclaje. Esto permite enfocar esta aplicación con un horizonte en el Internet de las cosas, ya que está abierto a que cualquier persona cree su propio dispositivo para añadirlo a la red, cumpliendo con los requisitos de los campos de la base de datos especificados anteriormente.

Para tomar la decisión de si un contenedor debe ser o no recogido, se evalúan todos los niveles de llenado y la concentración de CO_2 del contenedor. Si los niveles de llenado superan el umbral en 3 o más sensores o desde el módulo de sensores se ha reportado un 1 para el nivel de CO_2, entonces se incluirá en la lista de los contenedores a recoger. En caso de no superar ninguno de los dos umbrales establecidos, nivel de llenado o concentración de CO_2, el contenedor irá a la lista de los excluidos.

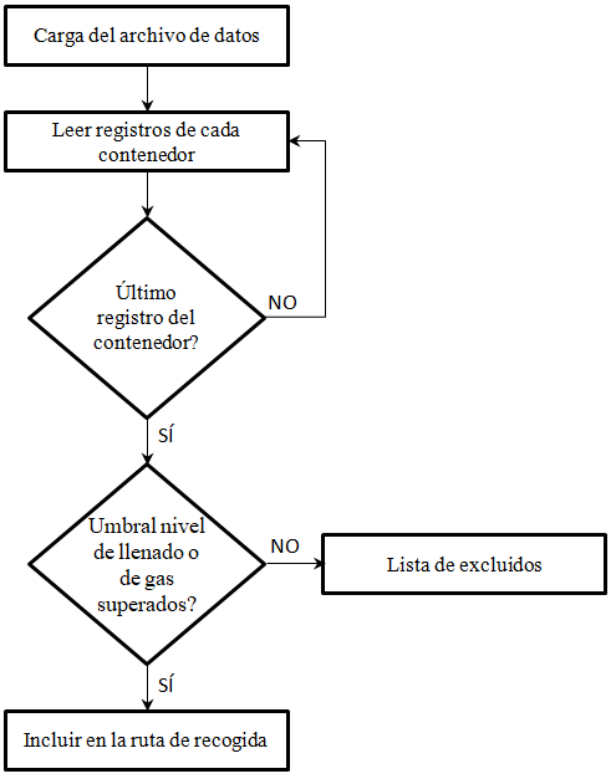


Figura 43. Diagrama de flujo de la aplicación de procesado de datos.

La figura siguiente muestra la interface de la aplicación, donde se pueden ver dos botones, el botón de ***PLAY***, que sirve para cargar el archivo exportado de la base de datos, generar las listas con los contenedores incluidos y excluidos de la ruta, exportar los incluidos a un archivo csv, mostrar un mapa con los contenedores y por último los contenedores con batería baja.

Figura 44. Interface de la aplicación Sistema de monitorización de un contenedor de reciclaje.

A continuación se muestra paso a paso, cómo utilizar la aplicación:

1. Se presiona el botón de ***PLAY*** que abre una ventana para seleccionar el archivo que se ha exportado de la base de datos.

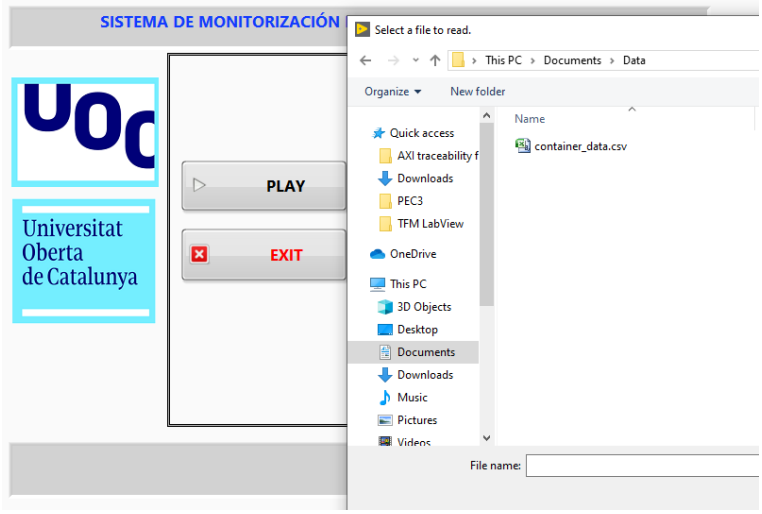


Figura 45. Selección del archivo exportado de la base de datos.

2. Una vez seleccionado el archivo, se muestra en la interface de la aplicación, el listado con los contenedores. Hay cuatro pestañas, una contiene los contenedores incluidos en la ruta, otra pestaña muestra los que están excluidos, la tercera es el mapa con los contenedores y la cuarta lista los contenedores con una batería baja

del sistema. La siguiente figura muestra la interface de las listas de incluidos y excluidos.

Figura 46. Interface de la aplicación con las listas de los contenedores incluidos y excluidos de la ruta.

3. Si se presiona en la pestaña *"MAPA"*, se mostrará una área con los contenedores que se encuentran en esa zona. Los puntos grises corresponden a los contenedores, se marcarán en color rojo cuando deban ser incluidos, puesto que significa que el límite ha sido superado, y en color verde cuando deban ser excluidos.

Figura 47. Interface de la aplicación con el mapa de los contenedores de una zona de Barcelona.

4. La siguiente imagen muestra la interface con la lista de los contenedores con una batería baja, es decir, por debajo del umbral preestablecido durante la sección de diseño de la batería.

Figura 48. Interface de la aplicación con la lista de los contenedores con una batería baja.

Además, también muestra la ruta en la que se ha guardado el archivo que contiene los contenedores a incluir en la ruta de los camiones de reciclaje.

Figura 49. Display con la ruta en la que se ha creado el archivo.

5. Por último, para salir de la aplicación, simplemente será necesario presionar el botón de ***EXIT.***

Figura 50. Botón para salir de la aplicación.

4. IMPLEMENTACIÓN Y RESULTADOS DEL PROYECTO

Después del diseño propuesto en la sección anterior, en esta se explica cómo implementar la fabricación de un prototipo para la puesta en marcha del sistema de monitorización del estado de un contenedor de reciclaje. Se verificará el correcto funcionamiento de este y se expondrán los resultados obtenidos. La figura siguiente muestra el diagrama de bloques detallado del sistema completo a implementar.

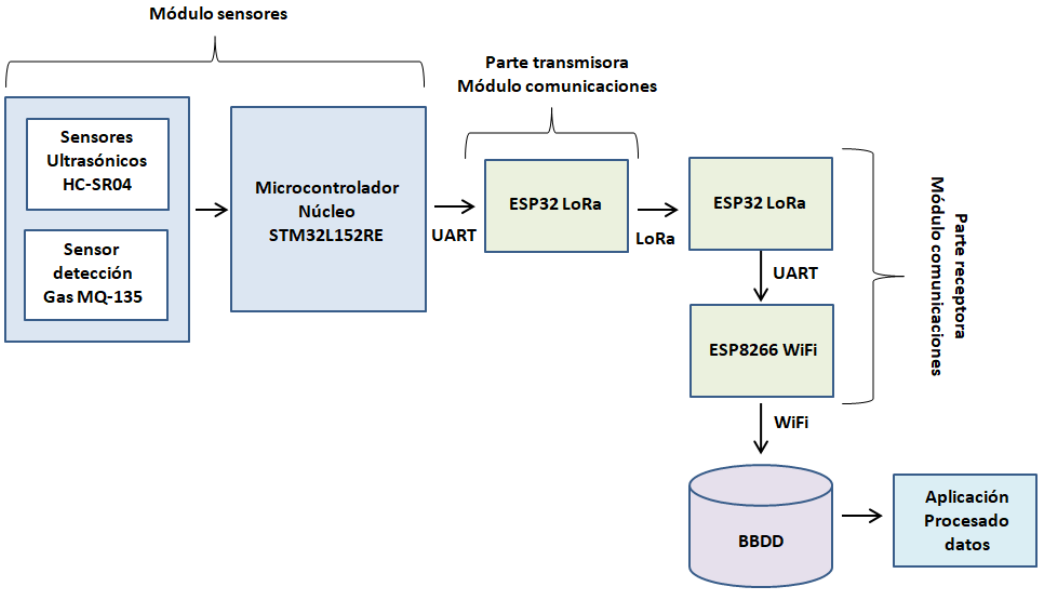


Figura 51. Diagrama de bloques detallado del sistema completo

4.1 Prototipo

Para la realización de este proyecto, en el que diferentes parámetros serán recogidos para monitorizar el estado de un contenedor de reciclaje, se implementará un prototipo que estará compuesto de lo siguiente:

Material	Cantidad
Sensor ultrasónico HC-SR04	5
Sensor de detección de gas CO2 MQ135	1
Núcleo STM32L152RE	1
Heltec ESP32 WiFi LoRa V2	2
NodeMCU v3 ESP8266 WiFi	1
Pila alcalina tipo AA 3000 mAh	4
Porta pilas	1
LDO MCP1802T	2
MOSFET IRL540	7

Tabla 11. Material utilizado para llevar a cabo el prototipo del proyecto.

La siguiente figura muestra el prototipo completo, con el módulo de sensores en la parte central derecha y el módulo de comunicación en la parte izquierda de la figura.

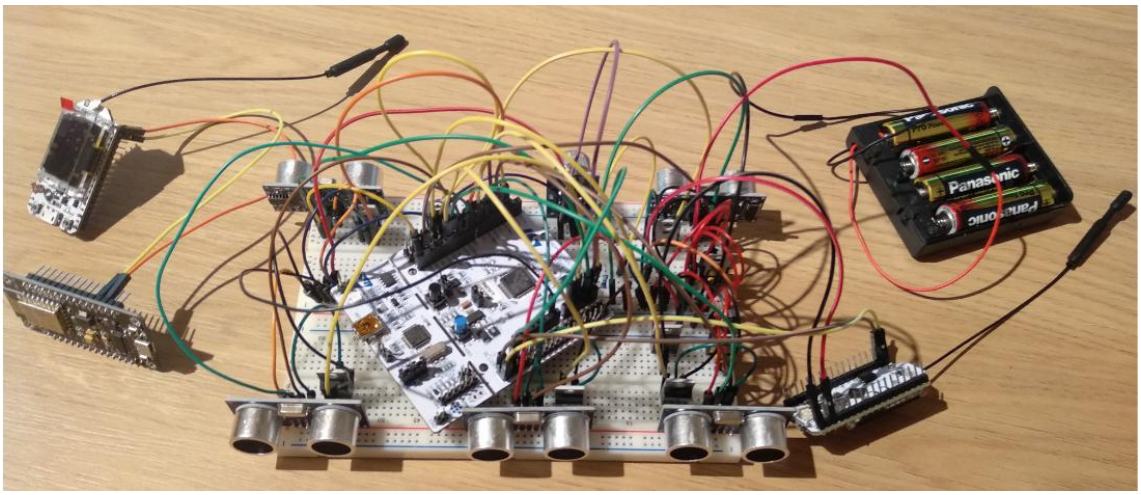


Figura 52. Sistema completo del proyecto.

La figura 53 muestra el conexionado del sistema completo presentado en la figura anterior, con todos los módulos (no se han incluido los LDO y MOSFETs por simplificar).

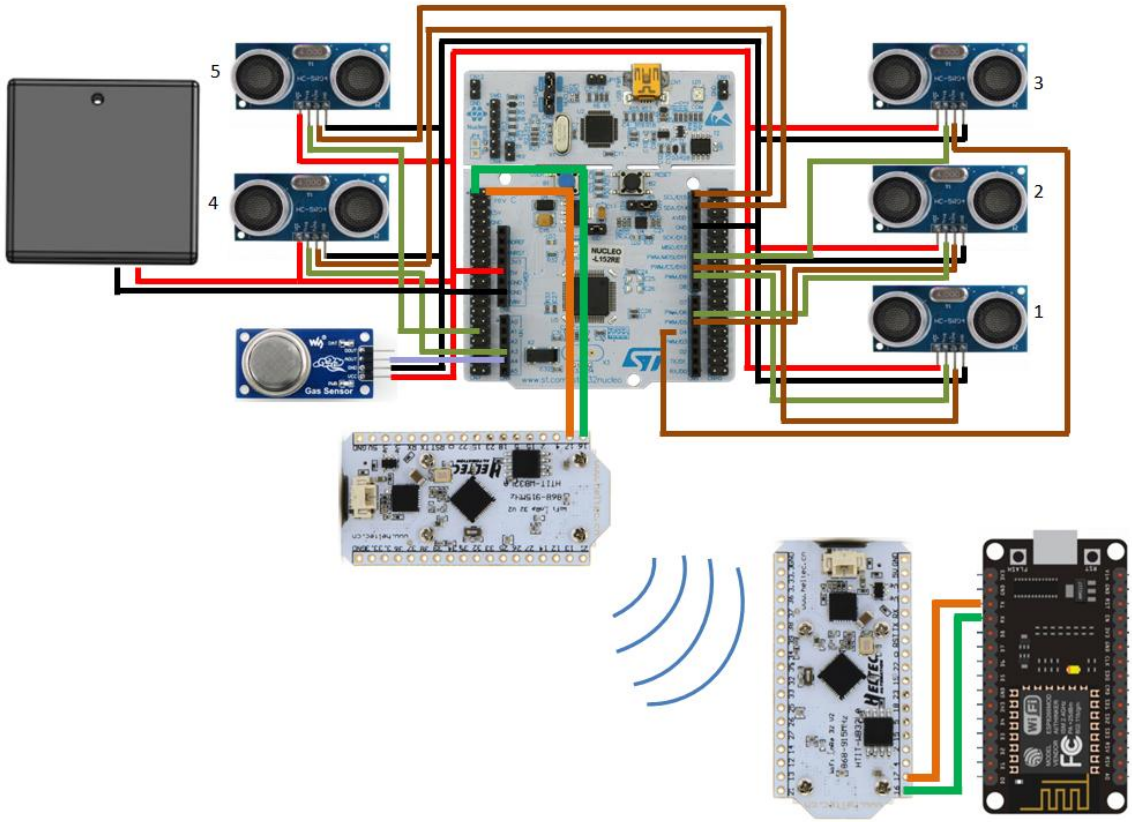


Figura 53. Conexionado completo del prototipo.

4.1.1. Módulo de sensores

Como ya se ha comentado en la *sección 3.1*, el módulo de sensores está compuesto por los sensores que detectan el nivel de llenado del contenedor y el nivel de concentración de CO_2 al que se encuentra el interior de este. A continuación, se muestra una imagen de esta parte del sistema, de manera individual, donde se ve la placa de desarrollo STM32L152RE junto con los sensores ultrasónicos HC-SR04, el sensor de detección de gas MQ-135 y la batería.

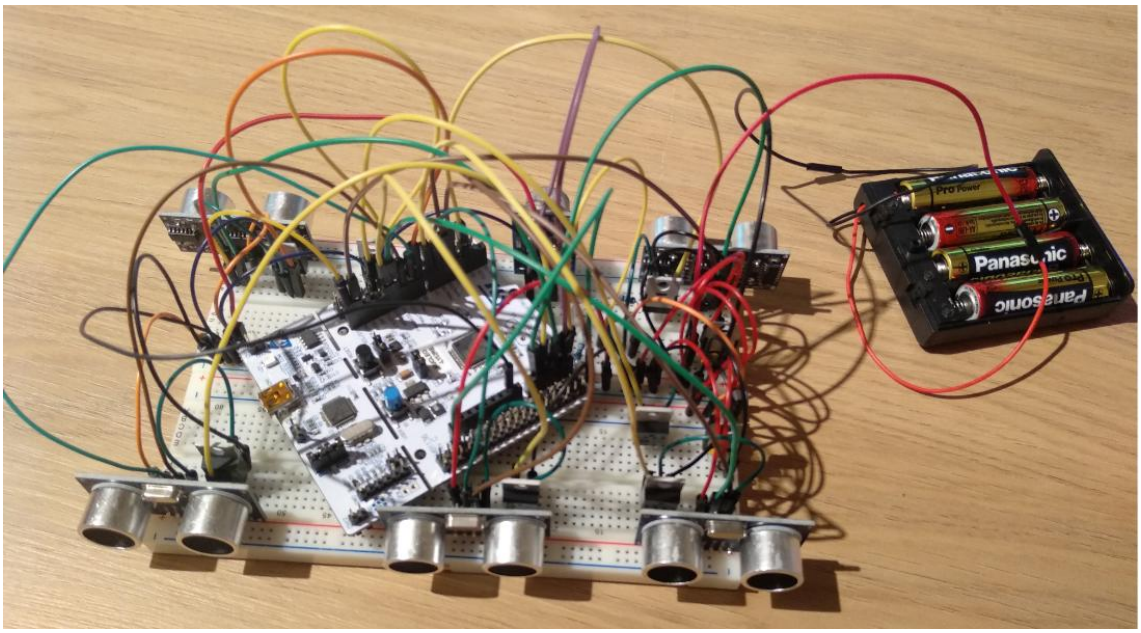


Figura 54. Módulo de sensores.

4.1.2. Módulo de comunicación

El módulo de comunicación está formado por dos partes, la transmisora y la receptora, que se encuentran en ubicaciones diferentes. En la figura siguiente, el módulo de la izquierda (ESP32) corresponde a la parte transmisora, que está junto al módulo de sensores, mientras que en la parte derecha de la figura, podemos ver la parte receptora (ESP32 y ESP8266) y encargada de subir los datos a la BBDD.

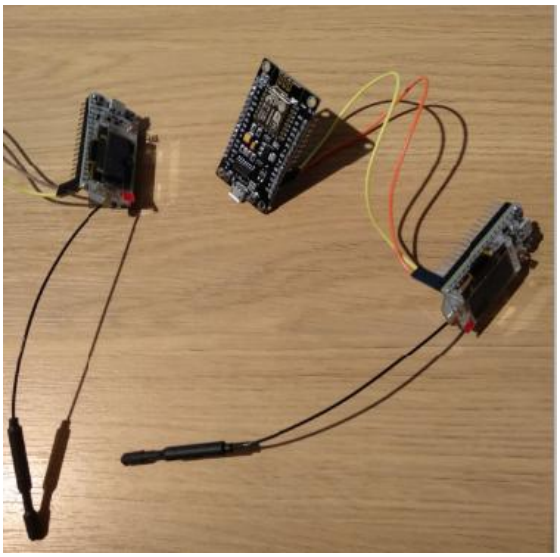


Figura 55. Módulo de comunicaciones. A la izquierda podemos ver la parte transmisora y a la derecha la parte receptora.

4.2. Coste del sistema

El presupuesto de un proyecto es importante a la hora de implementarse, ya que en muchas ocasiones es lo que determina si es viable o no, por tanto, esto se ha tenido en cuenta durante todo el proyecto. A continuación, se muestra una tabla del coste del material a utilizar para la realización de este prototipo.

Material	Precio	Cantidad	Total
Sensor ultrasónico HC-SR04	1,80 €	5	9,00 €
Sensor detección de gas MQ-135	5,79 €	1	5,79 €
Núcleo STM32L152RE	21,90 €	1	21,90 €
Heltec ESP32 WiFi LoRa v2	10,90 €	2	21,80 €
NodeMCU v3 ESP8266	6,00 €	1	6,00 €
Pilas alcalinas tipo AA	0,8 €	4	3,20 €
Portapilas	5,80 €	1	5,80 €
Cableado	5,79 €	1	5,79 €
MOSFETs IRL540	1,87 €	7	13,09 €
LDO MCP1802T	1,55 €	2	3,10 €
Soporte y tapa	20 €	1	20 €
TOTAL			**115,47 €**

Tabla 12. Presupuesto del prototipo del sistema.

4.3. Resultados

A continuación se muestran los resultados obtenidos de las diferentes partes del prototipo.

4.3.1. Sensor volumétrico HC-SR04

Como se ha comentado durante el apartado de diseño, este sensor envía un pulso por el pin *Echo*, proporcional a la distancia a la que se encuentra el objeto. Para verificar el correcto funcionamiento de este, se ha procedido a capturar mediante un osciloscopio el pulso enviado, con el fin de medirlo y contrastar los valores entre los obtenidos desde el microcontrolador con los obtenidos en el osciloscopio.

Se empezará por generar un pulso para una distancia de 10 cm. La figura siguiente muestra dicho pulso. Teniendo en cuenta que, el osciloscopio ha sido configurado para un time/division de 500 µs, se puede apreciar que el pulso tiene una duración de aproximadamente 600 µs.

Si realizamos los cálculos para obtener la distancia a la que se encuentra el objeto, recuperando la fórmula especificada durante la fase de diseño, tenemos que:

Distancia [m] = (Tiempo del pulso [s] * Velocidad del sonido [343m/s]) / 2

Distancia [m] = (600E-6 * 343) / 2

Distancia [m] = 0.102 m

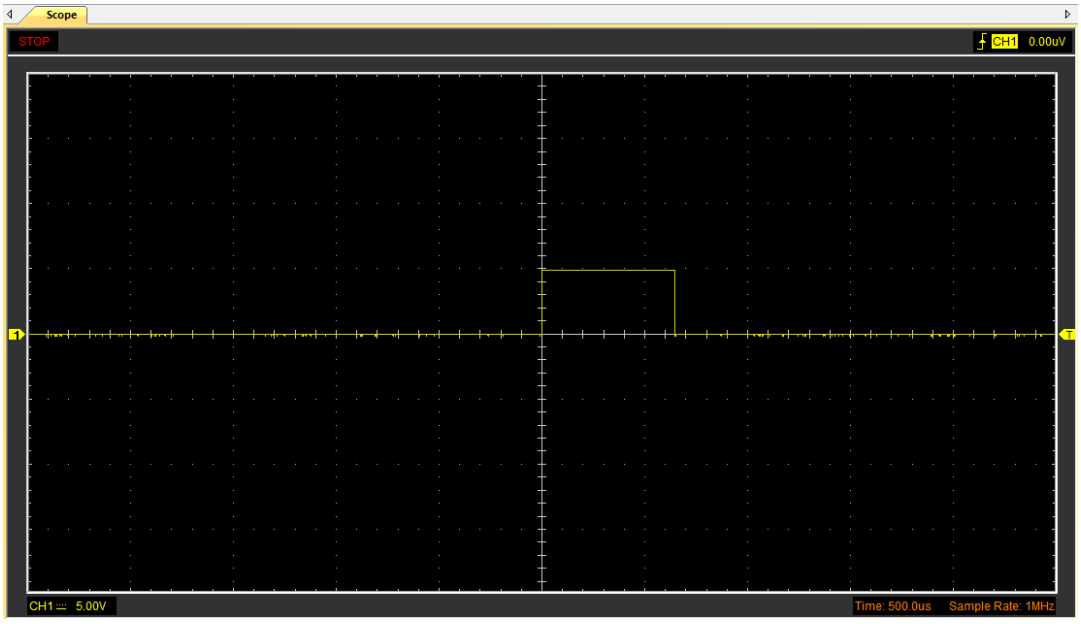


Figura 56. Pulso obtenido con el osciloscopio para un objeto que se encuentra a una distancia de 10 cm.

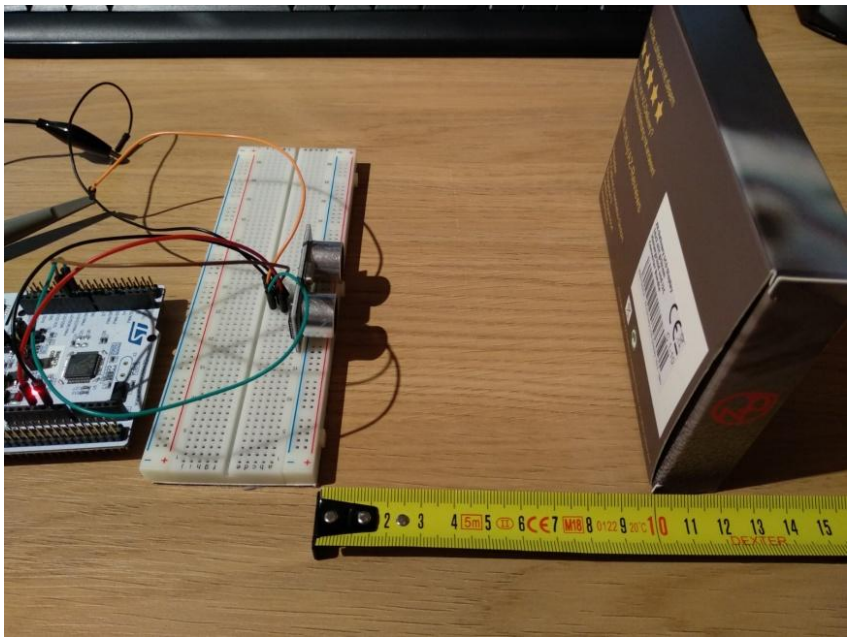


Figura 57. Sensor ultrasónico HC-SR04 con objeto a 10 cm.

Ahora, gracias a la herramienta PuTTY, es posible visualizar los datos, por el puerto serie, que han sido obtenidos por el microcontrolador. Se aprecia en la siguiente figura, que la adquisición de los datos es estable.

Figura 58. Valores de distancia, leídos por el puerto serie, para un objeto ubicado a 10 cm.

Para asegurar que las medidas son estables y correctas, se realiza otra medición, pero ahora con un objeto que se encuentra a 30 cm de distancia.

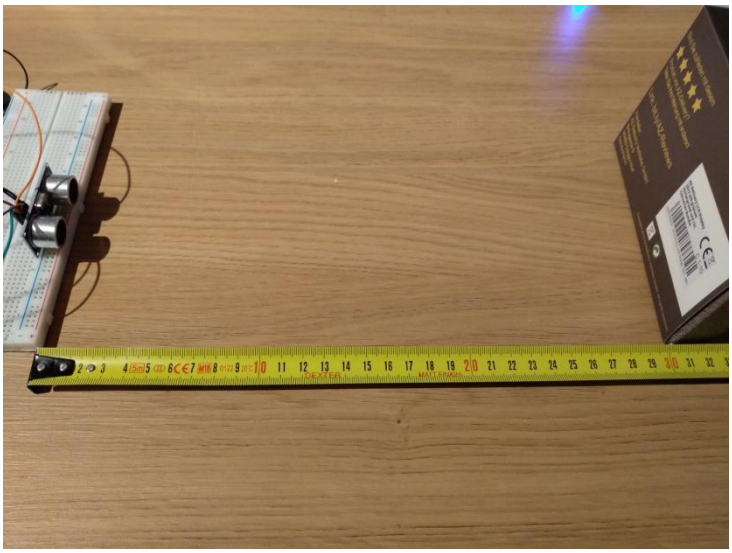


Figura 59. Sensor ultrasónico HC-SR04 con objeto a 30 cm.

La figura siguiente muestra el pulso recibido del sensor HC-SR04. Al igual que antes, el time/division es de 500 µs, por lo que se aprecia que el pulsa dura aproximadamente 1.77 ms. Realizamos los cálculos para obtener la distancia, tal y como se ha hecho antes.

Distancia [m] = (Tiempo del pulso [s] * Velocidad del sonido [343m/s]) / 2

Distancia [m] = (1.77E-3 * 343) / 2

Distancia [m] = 0.303 m

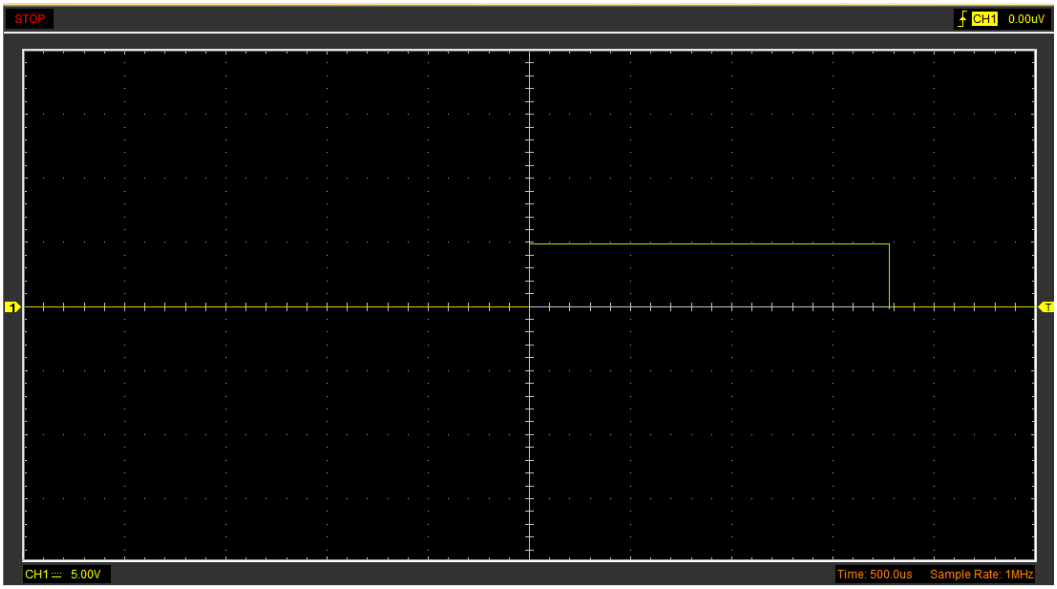


Figura 60. Pulso obtenido con el osciloscopio para un objeto que se encuentra a una distancia de 30 cm.

También aquí, se visualiza por el puerto serie los datos obtenidos desde el STM32. Se observa que la medida es estable y precisa.

Figura 61. Valores de distancia para un objeto ubicado a 30 cm.

Por último, como se ha comentado antes en este proyecto, es importante que este módulo consuma la menor energía posible, ya que deberá alimentarse con una batería. Aunque el módulo está en una localización de fácil acceso, siempre es importante optimizar el consumo para así, maximizar la vida útil del sistema.

La siguiente figura corresponde a la medida de la corriente, realizada con un multímetro, que pasa por el sensor ultrasónico. Se aprecia que tiene un consumo de 0.5 µA/s, por lo que si se hace un cálculo rápido, obtenemos lo siguiente:

Consumo por hora = 0.5 µA * 3600 s = 1.8 mA

Este consumo concuerda con las especificaciones para este sensor cuando se encuentra en standby, es decir, cuando no está en modo trabajo [13].

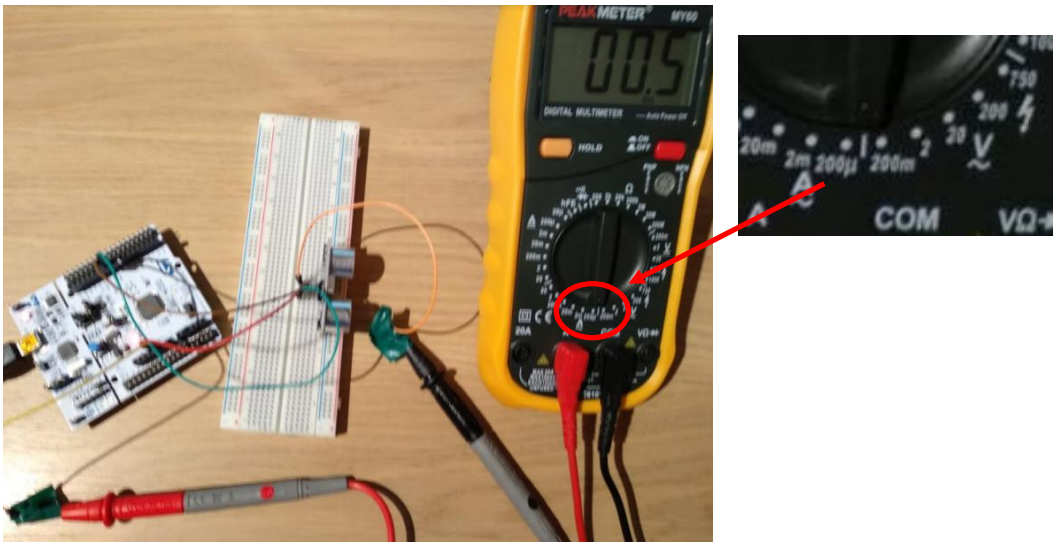


Figura 62. Medida de corriente para el sensor HC-SR04

4.3.2. Sensor de detección de gas CO2

El sensor de detección de gas CO2 se ha calibrado, tal y como se ha explicado en la sección 3.1.2.1. Para verificar el correcto funcionamiento, se leen los valores por el puerto serie con la herramienta PuTTY. Se ha dividido por un factor 10 los valores, para así poder ser enviados vía UART, y verlos por el PuTTY, ya que con la función HAL_UART_Transmit, solo permite enviar 8 bits de una vez por argumento.

Los valores que se observan en la figura 63, son los valores leídos por el ADC del microcontrolador, por el pin correspondiente al sensor de detección de CO_2. Se observan valores de 58-60, que corresponderían a valores de 580 - 600 ppm, debido al factor 10 aplicado, en una habitación con dos personas durante 5 horas.

Figura 63. Valores leídos por el puerto serie para el sensor de detección de gas CO2

4.3.3. Módulo de comunicaciones y Base de datos

Para verificar que el módulo de comunicaciones está funcionando correctamente, se muestra la figura siguiente, donde se aprecia que los datos han sido subidos a la BBDD, aproximadamente cada 7 segundos.

Fecha	ID	Distance	Distance2	Distance3	Distance4	Distance5	Volume	CO2sensor	Location	Battery
2020-06-10 18:21:11	44462151890428	50	52	52	61	60	0	0	Rambla de Prim 254	0
2020-06-10 18:21:18	44462151890428	29	29	29	22	22	5	0	Rambla de Prim 254	0
2020-06-10 18:21:25	44462151890428	29	29	29	21	22	5	0	Rambla de Prim 254	0
2020-06-10 18:21:32	44462151890428	21	21	22	21	22	5	0	Rambla de Prim 254	0
2020-06-10 18:21:39	44462151890428	22	22	21	21	22	5	0	Rambla de Prim 254	0
2020-06-10 18:21:46	44462151890428	21	21	21	21	21	5	1	Rambla de Prim 254	0
2020-06-10 18:21:54	44462151890428	21	21	21	21	21	5	1	Rambla de Prim 254	0
2020-06-10 18:22:01	44462151890428	21	21	21	21	21	5	1	Rambla de Prim 254	0
2020-06-10 18:22:08	44462151890428	51	52	52	21	21	2	1	Rambla de Prim 254	0
2020-06-10 18:22:15	44462151890428	73	75	76	21	21	2	0	Rambla de Prim 254	0
2020-06-10 18:22:22	44462151890428	33	33	33	22	21	5	0	Rambla de Prim 254	0
2020-06-10 18:22:29	44462151890428	33	33	34	23	23	5	0	Rambla de Prim 254	0
2020-06-10 18:22:36	44462151890428	33	33	34	33	33	5	0	Rambla de Prim 254	0
2020-06-10 18:22:43	44462151890428	10	10	10	10	9	5	0	Rambla de Prim 254	0
2020-06-10 18:22:50	44462151890428	9	9	9	10	9	5	0	Rambla de Prim 254	0
2020-06-10 18:22:57	44462151890428	10	10	9	14	14	5	0	Rambla de Prim 254	0
2020-06-10 18:23:04	44462151890428	9	9	9	14	15	5	0	Rambla de Prim 254	0
2020-06-10 18:23:12	44462151890428	9	9	9	13	15	5	0	Rambla de Prim 254	0
2020-06-10 18:23:19	44462151890428	9	9	9	14	15	5	0	Rambla de Prim 254	0
2020-06-10 18:23:26	44462151890428	9	9	9	13	14	5	0	Rambla de Prim 254	0
2020-06-10 18:23:33	44462151890428	9	9	9	14	15	5	0	Rambla de Prim 254	0

Figura 64. Registro de los datos enviados por el módulo de comunicaciones.

4.3.4 Aplicación para procesamiento de datos

A continuación, se muestran los resultados que genera la aplicación creada para el procesamiento de los datos. La siguiente figura muestra un archivo .csv ejemplo como el que nos descargamos de la base de datos. Se aprecia que existen diferentes ID y localizaciones para diferentes contenedores de reciclaje y sus respectivos valores.

```
Fecha,ID,Distance,Distance2,Distance3,Distance4,Distance5,Volume,CO2sensor,Location,Battery
2020-05-16 01:52:08,506034803045884,92,93,93,91,93,0,1,Cantabria 72,0
2020-05-16 02:00:35,206034803045884,89,83,87,83,81,0,0,Rambla de Prim 226,0
2020-05-16 02:03:15,306034803045884,73,74,74,74,74,0,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:03:27,106034803045884,82,82,82,82,82,0,0,Rambla de Prim 250,0
2020-05-16 02:03:51,206034803045884,79,72,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:04:14,806034803045884,75,75,75,74,75,0,1,Agricultura 232,0
2020-05-16 02:04:26,146034803045884,77,71,72,71,72,0,1,Julian Besteiro 10,0
2020-05-16 02:05:02,106034803045884,82,82,82,82,82,0,0,Rambla de Prim 250,0
2020-05-16 02:05:14,306034803045884,75,75,75,74,75,0,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:05:38,206034803045884,77,71,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:05:50,706034803045884,75,75,75,74,75,0,0,Maresme 220,0
2020-05-16 02:06:02,906034803045884,33,21,33,34,25,5,0,Guipuzcoa 73,0
2020-05-16 02:06:14,206034803045884,77,71,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:06:26,106034803045884,82,82,82,82,82,0,0,Rambla de Prim 250,0
2020-05-16 02:06:38,306034803045884,53,51,53,54,55,0,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:06:50,606034803045884,70,71,72,71,72,0,0,Cantabria 42,1
2020-05-16 02:07:02,106034803045884,42,42,42,42,42,0,0,Rambla de Prim 250,0
2020-05-16 02:07:14,306034803045884,37,41,43,44,35,2,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:07:26,406034803045884,70,71,72,71,72,0,0,Rambla de Prim 118,0
2020-05-16 02:07:38,106034803045884,30,30,29,31,30,5,0,Rambla de Prim 250,0
2020-05-16 02:07:50,116034803045884,37,41,37,44,35,3,0,Guipuzcoa 112,0
2020-05-16 02:08:02,206034803045884,70,71,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:08:14,106034803045884,30,30,29,31,30,5,0,Rambla de Prim 250,0
2020-05-16 02:08:26,306034803045884,37,41,37,44,35,3,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:08:38,206034803045884,70,71,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:08:50,136034803045884,30,30,29,31,30,5,0,Puigcerda 212,0
2020-05-16 02:09:02,306034803045884,47,41,37,44,35,2,1,GranVia de les corts catalanes 1098,1
2020-05-16 02:09:14,126034803045884,60,61,62,69,62,0,0,Treball 262,0
2020-05-16 02:09:26,106034803045884,30,30,29,31,30,5,0,Rambla de Prim 250,0
2020-05-16 02:10:08,406034803045884,92,93,93,91,93,0,0,Rambla de Prim 118,0
2020-05-16 02:10:35,206034803045884,89,83,87,83,81,0,0,Rambla de Prim 226,1
```

Figura 65. Archivo .csv ejemplo para el procesado de los datos.

En el archivo se diferencian 13 ID diferentes, estos ID corresponden a la ***MAC ID*** del módulo Heltec ESP32 de la parte transmisora. La localización del contenedor también es enviada por este mismo módulo ESP32 previamente configurado.

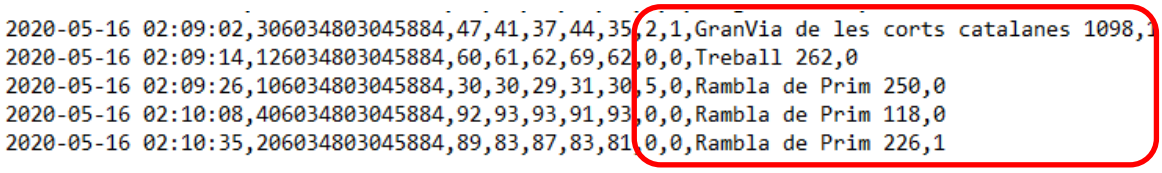
ID	Localización contenedor
106034803045884	Rambla de Prim 250
206034803045884	Rambla de Prim 226
306034803045884	GranVia de les corts catalanes 1098
406034803045884	Rambla de Prim 118
506034803045884	Cantabria 72
606034803045884	Cantabria 42
706034803045884	Maresme 220
806034803045884	Agricultura 232
906034803045884	Guipuzcoa 73
116034803045884	Guipuzcoa 112
126034803045884	Treball 262
136034803045884	Puigcerda 212
146034803045884	Julian Besteiro 10

Tabla 13. ID encontrados por la aplicación juntamente con su localización.

Ahora, se carga el archivo tal y como se ha explicado en la sección 3.7, y obtenemos lo siguiente. En la figura 66 se aprecia que se incluyen los contenedores siguientes:

- Cantabria 72
- GranVia de les corts catalanes 1098
- Rambla de Prim 250
- Agricultura 232
- Julian Besteiro 10
- Guipuzcoa 73
- Guipuzcoa 112
- Puigcerda 212

Esto es debido a que, por ejemplo, el contenedor de "Rambla de Prim 250" tiene un volumen por encima de 3, mientras que aunque el contenedor de "GranVia de les corts catalanes 1098" no tiene ese volumen, sí que sobrepasa los niveles de CO_2 permitidos, ya que tenemos un valor de 1 en la columna correspondiente al CO_2, y esto significa que el valor de CO_2, fijado para este ejemplo en 900 ppm, ha sido superado. Por otro lado, se encuentran los no incluidos como por ejemplo, "Rambla de Prim 226", puesto que no se ha superado ninguno de los dos umbrales.



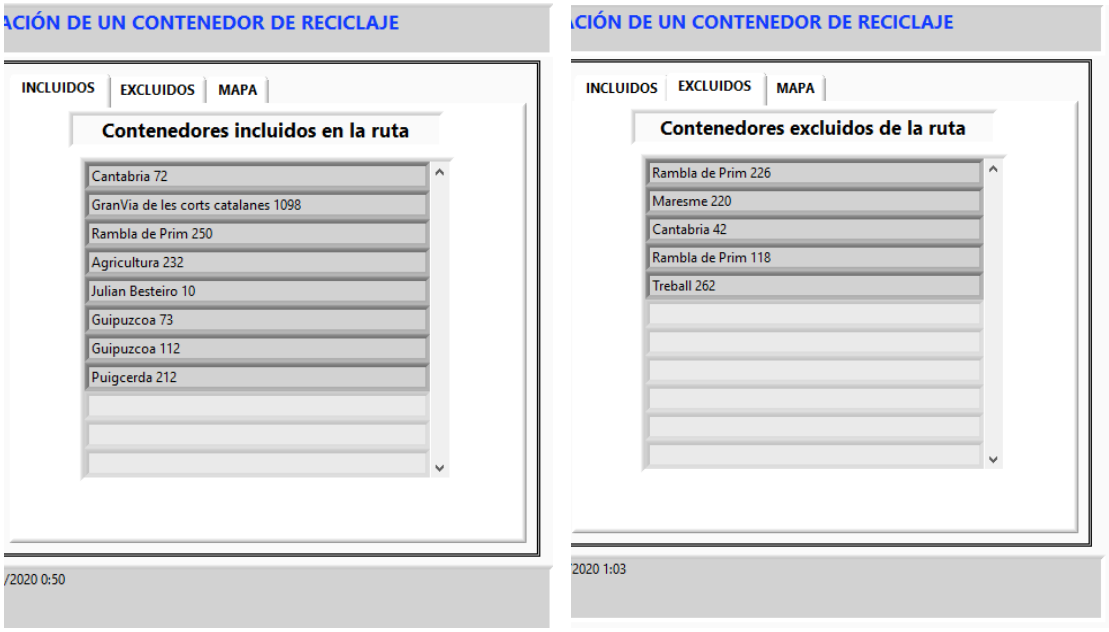


Figura 66. Listas de contenedores. En la parte izquierda se muestran los contenedores incluidos en la ruta, mientras que la parte derecha muestra los excluidos.

La figura siguiente muestra la pestaña correspondiente al mapa donde se encuentran los contenedores, en color rojo los que se han incluido en la ruta de reciclaje, mientras que los de color verde corresponden a los que no se han incluido.

Figura 67. Mapa que muestra los contenedores incluidos (color rojo) y los excluidos (color verde) de la ruta de recogida.

Con el mismo archivo también se obtienen los contenedores en los que el sistema tiene una batería baja, es decir, con un valor 1 en el campo de batería, y por tanto, tiene que ser reemplazada. En la siguiente figura se puede observar la lista para estos contenedores.

```
2020-05-16 02:06:50,606034803045884,70,71,72,71,72,0,0,Cantabria 42,1
2020-05-16 02:07:02,106034803045884,42,42,42,42,42,0,0,Rambla de Prim 250,0
2020-05-16 02:07:14,306034803045884,37,41,43,44,35,2,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:07:26,406034803045884,70,71,72,71,72,0,0,Rambla de Prim 118,0
2020-05-16 02:07:38,106034803045884,30,30,29,31,30,5,0,Rambla de Prim 250,0
2020-05-16 02:07:50,116034803045884,37,41,37,44,35,3,0,Guipuzcoa 112,0
2020-05-16 02:08:02,206034803045884,70,71,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:08:14,106034803045884,30,30,29,31,30,5,0,Rambla de Prim 250,0
2020-05-16 02:08:26,306034803045884,37,41,37,44,35,3,0,GranVia de les corts catalanes 1098,0
2020-05-16 02:08:38,206034803045884,70,71,72,71,72,0,0,Rambla de Prim 226,0
2020-05-16 02:08:50,136034803045884,30,30,29,31,30,5,0,Puigcerda 212,0
2020-05-16 02:09:02,306034803045884,47,41,37,44,35,2,1,GranVia de les corts catalanes 1098,1
2020-05-16 02:09:14,126034803045884,60,61,62,69,62,0,0,Treball 262,0
2020-05-16 02:09:26,106034803045884,30,30,29,31,30,5,0,Rambla de Prim 250,0
2020-05-16 02:10:08,406034803045884,92,93,93,91,93,0,0,Rambla de Prim 118,0
2020-05-16 02:10:35,206034803045884,89,83,87,83,81,0,0,Rambla de Prim 226,1
```

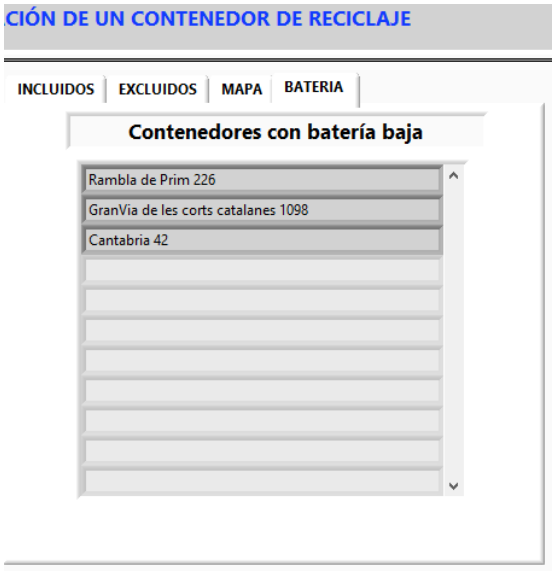


Figura 68. Lista de los contenedores con una batería baja.

También se muestra el archivo donde se guardan los contenedores que deben ser incluidos en la ruta.

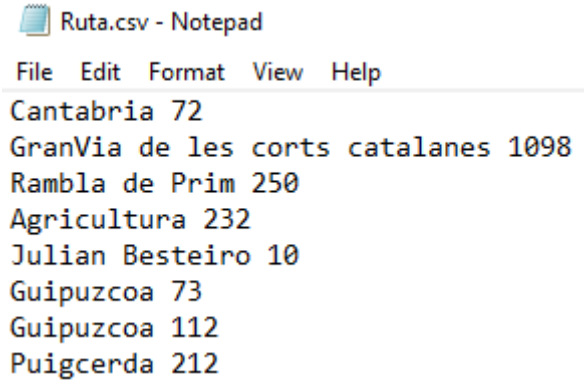


Figura 69. Archivo con los contenedores incluidos en la ruta.

5. CONCLUSIONES

Después del diseño e implementación de este proyecto, se ha visto que se han necesitado diferentes áreas de conocimientos, puesto que es una aplicación multidisciplinar. Han sido necesarios conocimientos de electrónica e informática para poder programar de la manera más optimizada posible, los sensores y las placas de desarrollo utilizadas en el proyecto. Además, se ha tenido que investigar sobre bases de datos y las herramientas necesarias para su gestión.

A día de hoy, existen aun pocos prototipos de sistemas de monitorización de contenedores que estén en funcionamiento, y sin embargo, son necesarios para evitar que los residuos se amontonen en los contenedores, provocando una mala higiene de la zona. Con este prototipo creado en este proyecto, es posible monitorizar un contenedor de una manera bastante fiable, puesto que se monitoriza todo el volumen interior. Por tanto, en base a los resultados obtenidos, se puede concluir que los resultados son satisfactorios.

Durante la realización del proyecto, han surgido problemas relacionados con el consumo del sistema, puesto que un alto consumo podría hacer que la aplicación no fuera viable, por tanto, es muy importante que en una aplicación alimentada con una batería, se maximice la autonomía de esta. Se ha recurrido a soluciones de apagado del sistema mediante electrónica, como MOSFETs activados mediante el microcontrolador y modos de ultra bajo consumo de dicho microcontrolador. Gracias a este tipo de soluciones se ha aumentado la batería considerablemente hasta los 238 días.

No se debe olvidar comentar, que es evidente que la planificación y la metodología de trabajo a la hora de enfocar un proyecto, son muy importantes, ya que puede ser decisivo para lograr los objetivos marcados al inicio de este.

A nivel personal, este proyecto ha sido gratificante para mí, ya que he podido aplicar diferentes áreas de las telecomunicaciones para poder alcanzar el objetivo del trabajo. El hecho de haber podido implementar en un solo producto, electrónica, informática, comunicaciones e incluso parte de bases de datos, me ha permitido obtener un mayor conocimiento de las telecomunicaciones.

6. VERSIONES FUTURAS

Después de la realización del trabajo, surgen diferentes ideas para el desarrollo de un proyecto futuro que permita mejorar las prestaciones del realizado para este trabajo. Como se ha comentado en las conclusiones, uno de los problemas encontrados ha sido la autonomía de la batería y en base a eso, se proponen las siguientes mejoras.

- Sustituir el sensor electroquímico de detección de gas por un sensor espectroscópico infrarrojo.
- Sustituir la placa de desarrollo por otra con un consumo menor. Una buena opción sería Núcleo STM32L031K6.
- Aumentar la capacidad de la batería al doble de la actual, utilizando un segundo porta pilas igual e instalando los dos en paralelo para proporcionar el doble de corriente al mismo valor de tensión. Con esto se consigue multiplicar por un factor dos la autonomía, que sería de 476 días.

Al inicio del proyecto, se planteó generar la trayectoria más optimizada que debía seguir el camión de recogida una vez conocidos los contenedores incluidos en la ruta. Sin embargo, esto requiere de un alto nivel de procesamiento de los datos, puesto que no solo se debe calcular la ruta más corta entre dos puntos, sino que se debe asegurar que se pasa por todos los puntos intermedios. Este nivel de procesado, hace inviable su realización en un período tan corto de tiempo. Esta sería otra interesante mejora para un futuro proyecto.

La impresión del soporte y la tapa para poder instalar el módulo de sensores y la parte transmisora del módulo de comunicaciones no se ha llevado a cabo, debido a que no ha sido posible disponer de una impresora 3D. Por tanto, para la implementación final en el siguiente proyecto, esta impresión podrá realizarse.

Por último, existe otra posible mejora en la aplicación para el procesado de datos. Esta mejora tiene que ver con el mapa que muestra los diferentes contenedores, ya que este es fijo y sería interesante poder hacer que fuera dinámico para poder navegar por él.

7. GLOSARIO

CO_2	Dióxido de carbono
µC	Abreviatura de microcontrolador
LoRa	(Long Range) Tecnología de comunicación de largo alcance y bajo consumo
LoRaWAN	Protocolo de comunicación de LoRa
LPWAN	(Low-Power Wide-Area network) Red de área amplia de baja potencia
SF	(Spreading factor) Factor de ensanchamiento de la tecnología LoRa
APB clock	Reloj del bus de periféricos (Advanced Peripheral Bus clock)
Ppm	Partes por millón. Unidad de medida con la que se mide la concentración de un gas en otro
MOSFET	Transistor de efecto de campo metal-óxido-semiconductor
CMOS	(Complementary metal-oxide-semiconductor) Semiconductor complementario de óxido metálico
GPIO	(General purpose I/O). Pin de entrada/salida de propósito general
UART	(Universal Asynchronous Receiver-Transmitter) Comunicación serie Transmisor-Receptor Asíncrono Universal
RTC	(Real Time Clock) Reloj de tiempo real
ADC	(Analogic to Digital conversor) Conversor analógico a digital
PHP	(Hypertext Preprocessor)
LDO	(Low-DropOut) Regulador de voltaje de baja caída

8. REFERENCIAS

[1] Marek Krzymien, Michael Day, Kathleen Shaw & Lillian Zaremba "An Investigation of Odors and Volatile Organic Compounds Released during Composting", Journal of the Air & Waste Management Association. 27 Dec 2011

[2] Javier Ferrer, Enrique Alba, "BIN-CT: Urban waste collection based on predicting the container fill level" 22 Abril 2019.

https://www.sciencedirect.com/science/article/abs/pii/S0303264718301333

[3] Página web interempresas.net donde se explica el proyecto e-Garbage. Fuente: Wellness Telecom.

https://www.interempresas.net/Equipamiento_Municipal/Articulos/49606-e-Garbage-un-sistema-inteligente-de-gestion-de-la-recogida-de-residuos.html

[4] Página web oficial del proyecto Life EWAS.

http://life-ewas.eu/es/dissemination/articles.html

[5] Página web del proyecto CleanTec.

http://www.cleantecsoftware.com/modulos-plataforma-grsu/modulos-externos-adicionales-grsu/deteccion-del-volumen-de-llenado-grsu

[6] Detección del volumen de llenado en contenedores. Documento explicativo del proyecto CleanTec.

http://www.cleantecsoftware.com/descargas/CleanTec%20GRSU/Sensor%20volumen%20de%20llenado/CLEANTEC_DETECCION_VOLUMEN_LLENADOv2.pdf/detail

[7] Página web esmartcity.es donde se explica el proyecto TSwasTe.

https://www.esmartcity.es/comunicaciones/comunicacion-gestion-inteligente-residuos-urbanos-tecnologia-narrow-band-iot-nb-iot

[8] Datasheet del sensor láser VL53L1X

https://www.pololu.com/file/0J1506/VL53L1X.pdf

[9] Página web dedicada a la venta de todo tipo de sensores industriales de medida.

https://sensores-de-medida.es/medicion/sensores-y-transductores/sensores-de-distancia/sensores-de-distancia-por-ultrasonidos/

[10] Página web keyence donde se explica el funcionamiento de los sensores inductivos.

https://www.keyence.com.mx/ss/products/sensor/sensorbasics/proximity/info/

[11] Datasheet de sensores capacitivos de waycon

https://www.waycon.es/fileadmin/capacitive-sensors/Capacitive-Sensor.pdf

[12] Blog Infaimon dedicados a la imagen y visión artificial y al control y optimización de procesos.

https://blog.infaimon.com/detector-volumetrico-usos-tiene-seguridad/

[13] Datasheet del sensor ultrasónico HC-SR04.

https://www.electronicoscaldas.com/datasheet/HC-SR04.pdf

[14] Artículo comparativo entre sensores microondas de la empresa Endrich.

https://www.redeweb.com/ficheros/articulos/endrich.pdf

[15] Documento de la empresa Dräger para los sensores de detección de gas.

https://www.draeger.com/Library/Content/9046703_infoflip_gds_es_l3.pdf

[16] Documento de la Universidad Complutense de Madrid para los sensores de detección de gas.

https://www.ucm.es/data/cont/docs/3-2015-03-16-Principios%20de%20detecci%C3%B3n.pdf

[17] Documento técnico de Sigfox.

https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf

[18] Lora Alliance, "LoRaWAN - What is it?" Noviembre 2015. Documento técnico.

https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf

[19] Artículo de Digikey "Desarrollar con LoRa aplicaciones IoT de baja tasa y largo alcance.

https://www.digikey.es/es/articles/develop-lora-for-low-rate-long-range-iot-applications

[20] Documento de la tecnología NB-IoT de la asociación GSMA.

https://www.gsma.com/iot/wp-content/uploads/2019/07/201906-GSMA-NB-IoT-Deployment-Guide-v3.pdf

[21] Documento de la tecnología LTE-M de la asociación GSMA.

https://www.gsma.com/iot/wp-content/uploads/2019/08/201906-GSMA-LTE-M-Deployment-Guide-v3.pdf

[22] Página web de accent-systems donde se muestran las diferencias entre NB-IoT y LTE-M.

https://accent-systems.com/es/blog/diferencias-nb-iot-lte-m/?gclid=EAIaIQobChMI6sipm8-a6AIVUtTeCh1bgADVEAAYASAAEgLNefD_BwE

[23] Luis Alberto Casillas Santillán, Marc Gibert Ginestà, Óscar Pérez Mora, Universitat Oberta de Catalunya "Bases de datos en MySQL"

[24] Marc Gibert Ginestà, Óscar Pérez Mora, Universitat Oberta de Catalunya "Bases de datos en PostgreSQL"

[25] Página web de Microsoft con documentación de la base de datos MariaDB

https://docs.microsoft.com/es-es/azure/mariadb/

[26] Página web de datacamp con información sobre las bases de datos MongoDB

https://www.datacamp.com/community/tutorials/introduction-mongodb-python

[27] Development tools ST. Página web del fabricante ST

https://www.st.com/en/evaluation-tools/stm32-mcu-mpu-eval-tools.html

[28] Development tools NXP. Página web del fabricante NXP

https://www.nxp.com/design/development-boards:EVDEBRDSSYS#/

[29] Development tools Texas Instruments. Página web del fabricante Texas Instruments

https://www.ti.com/design-resources/embedded-development/hardware-kits-boards.html

[30] Página web de Contenur. Ficha técnica del contenedor Oval 3000.

https://www.contenur.com/productos/oval-3000/

[31] Página web de *ControllersTech*. Crear microsegundos en STM32.

https://controllerstech.com/create-1-microsecond-delay-stm32/

[32] Antonio Pinzón , Miguel Castillo , Edwin González , José Araúz , Vladimir Villarreal, Licenciatura en Ingeniería de Sistemas y Computación, Centro Regional de Chiriquí, Universidad Tecnológica de Panamá 2 GITCE, Centro Regional de Chiriquí, Universidad Tecnológica de Panamá, "Sistema de detección de sustancias y partículas contaminantes para el ambiente a través de sensores Arduino". https://pdfs.semanticscholar.org/07ab/f1a1ae2b976e56dc0cd6e943e81232cfd267.pdf

[33] Datasheet del proveedor Olimez para el sensor de detección de gas MQ-135.

https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf

[34] Página web de Davide Gironi para la calibración de un sensor de gas MQ.

http://davidegironi.blogspot.com/2014/01/cheap-co2-meter-using-mq135-sensor-with.html#.Xrld_mgzaUl

[35] Página web de Davide Gironi para el cálculo de los coeficientes de la gráfica de un sensor de gas MQ.

http://davidegironi.blogspot.com/2017/05/mq-gas-sensor-correlation-function.html#.Xrle2mgzaUk

[36] Página web CO2 earth con la concentración en ppm de CO2 en la atmosfera.

https://www.co2.earth/

[37] User manual de las placas de desarrollo Núcleo STM32.

https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

[38] Datasheet del fabricante ST para el microcontrolador STM32L151xE STM32L152xE.

https://www.st.com/resource/en/datasheet/stm32l151qe.pdf

[39] Manual de usuario UM1816 de la librería HAL para micrcontroladores ST.

https://www.st.com/resource/en/user_manual/dm00132099-description-of-stm32l1-hal-and-lowlayer-drivers-stmicroelectronics.pdf

[40] Manual de referencia RM0038 para los microcontroladores ST.

https://www.st.com/resource/en/reference_manual/cd00240193-stm32l100xx-stm32l151xx-stm32l152xx-and-stm32l162xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

[41] Manual de usuario UM1718 para la programación de los microcontroladores ST con el software STM32CubeMx.

https://www.st.com/resource/en/user_manual/dm00104712-stm32cubemx-for-stm32-configuration-and-initialization-c-code-generation-stmicroelectronics.pdf

[42] Borja Martinez, Ferran Adelantado, Andrea Bartoli, Xavier Vilajosana, "Exploring the Performance Boundaries of NB-IoT". February 2019

[43] Artículo "How Spreading Factor affects LoraWAN device battery life". Página web de TheThingsNetwork.

https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life

[44] Página web del fabricante SEMTECH. Chips SEMTECH SX1276.

https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276

[45] Datasheet del fabricante SEMTECH para la serie de chips LoRa SX1276/77/78/79.

https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001OKs/Bs97dmPXeatnbdoJNVMIDaKDlQz8q1N_gxDcgqi7g2o

[46] Páginaweb del fabricante HELTEC. Módulo ESP32 WiFi LoRa v2.

https://heltec.org/project/wifi-lora-32/

[47] Página web de Zerynth para el uso de un ESP8266 WiFi.

https://docs.zerynth.com/latest/official/board.zerynth.nodemcu3/docs/index.html#

[48] Datasheet del Regulador de voltaje MCP1802T-XX.

https://micropython.org/resources/datasheets/MCP1802-22053C.pdf

[49] Datasheet del MOSFET IRL540

https://pdf1.alldatasheet.com/datasheet-pdf/view/251859/VISHAY/IRL540.html

[50] Desconocido. Página web con las curvas de descarga de diferentes tipos de pilas.

https://lygte-info.dk/info/BatteriesLowCurrentDischarge%20UK.html

[51] Página web oficial de la herramienta phpMyAdmin.

https://www.phpmyadmin.net/

[52] Página web de Apachefriends.

https://www.apachefriends.org/es/index.html

[53] Página web oficial de National Instruments.

https://www.ni.com/es-es.html

9. ANEXO

En esta sección se pueden encontrar los códigos utilizados.

9.1. Código STM32L152RE

```
#include "main.h"
#include "stdio.h"
#include "math.h"

/* Private variables ---------------------------------------------------------
-*/
ADC_HandleTypeDef hadc;
DMA_HandleTypeDef hdma_adc;
RTC_HandleTypeDef hrtc;
TIM_HandleTypeDef htim3;
UART_HandleTypeDef huart4;
UART_HandleTypeDef huart2;

/* Variables*/
uint16_t Distance;
uint16_t Distance2;
uint16_t Distance3;
uint16_t Distance4;
uint16_t Distance5;
uint16_t ultrasonic_sensor_time;
uint16_t pulsetime;
uint8_t calibration_factor = 100;
uint8_t Time;
float Battery;
uint8_t bat;
uint16_t DMA_memory[2];
float CO2_value;
float RL = 1000;
float Ro = 4184; //3730; //Resistencia del sensor a una concentración de gas
conocida, en el aire limpio (416 ppm)
uint8_t counter;
float Voltage_value;
float ppm_CO2_float;
uint8_t ppm_CO2;
float Rs;
float Ro;

/* Private function prototypes -----------------------------------------------
-*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_UART4_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM3_Init(void);
static void MX_RTC_Init(void);
static void MX_ADC_Init(void);
/* USER CODE */

void delay_us (uint16_t us)
{
	__HAL_TIM_SET_COUNTER(&htim3,0); //Set TIMER to 0
	while (__HAL_TIM_GET_COUNTER(&htim3)<us); //keep here until 'us'
```

```
times.
}

uint32_t Ultrasonic_sensor (uint8_t n_sensor)
{
	switch (n_sensor)
	{
		case 1:
			pulsetime=0;
			HAL_GPIO_WritePin(GPIOA, MOSFET_1_Pin, GPIO_PIN_SET);
			HAL_Delay(1);
			HAL_GPIO_WritePin(GPIOC, Trigger1_Pin, GPIO_PIN_RESET);
			delay_us(2);
			HAL_GPIO_WritePin(GPIOC, Trigger1_Pin, GPIO_PIN_SET);
			delay_us(10);
			HAL_GPIO_WritePin(GPIOC, Trigger1_Pin, GPIO_PIN_RESET);
			while(!(HAL_GPIO_ReadPin(Echo1_GPIO_Port, Echo1_Pin)));
			while(HAL_GPIO_ReadPin(Echo1_GPIO_Port, Echo1_Pin))
			{
				pulsetime++;
				delay_us(1);
			}
			HAL_GPIO_WritePin(GPIOA, MOSFET_1_Pin, GPIO_PIN_RESET);
			if(pulsetime == 0)
			{
				return Ultrasonic_sensor(1);
			}
			return pulsetime;
			break;

		case 2:
			pulsetime=0;
			HAL_GPIO_WritePin(GPIOA, MOSFET_2_Pin, GPIO_PIN_SET);
			HAL_Delay(1);
			HAL_GPIO_WritePin(GPIOB, Trigger2_Pin, GPIO_PIN_RESET);
			delay_us(2);
			HAL_GPIO_WritePin(GPIOB, Trigger2_Pin, GPIO_PIN_SET);
			delay_us(10);
			HAL_GPIO_WritePin(GPIOB, Trigger2_Pin, GPIO_PIN_RESET);

			while(!(HAL_GPIO_ReadPin(GPIOB, Echo2_Pin)));
			while(HAL_GPIO_ReadPin(GPIOB, Echo2_Pin))
			{
				pulsetime++;
				delay_us(1);
			}
			HAL_GPIO_WritePin(GPIOA, MOSFET_2_Pin, GPIO_PIN_RESET);
			if(pulsetime == 0)
			{
				return Ultrasonic_sensor(2);
			}
			return pulsetime;
			break;

		case 3:
			pulsetime=0;
			HAL_GPIO_WritePin(GPIOA, MOSFET_3_Pin, GPIO_PIN_SET);
			HAL_Delay(1);
			HAL_GPIO_WritePin(GPIOA, Trigger3_Pin, GPIO_PIN_RESET);
```

```
                delay_us(2);
                HAL_GPIO_WritePin(GPIOA, Trigger3_Pin, GPIO_PIN_SET);
                delay_us(10);
                HAL_GPIO_WritePin(GPIOA, Trigger3_Pin, GPIO_PIN_RESET);

                while(!(HAL_GPIO_ReadPin(GPIOB, Echo3_Pin)));
                while(HAL_GPIO_ReadPin(GPIOB, Echo3_Pin))
                {
                        pulsetime++;
                        delay_us(1);
                }
                HAL_GPIO_WritePin(GPIOA, MOSFET_3_Pin, GPIO_PIN_RESET);
                if(pulsetime == 0)
                {
                        return Ultrasonic_sensor(3);
                }
                return pulsetime;
                break;

        case 4:
                pulsetime=0;
                HAL_GPIO_WritePin(GPIOA, MOSFET_4_Pin, GPIO_PIN_SET);
                HAL_Delay(1);
                HAL_GPIO_WritePin(GPIOB, Trigger4_Pin, GPIO_PIN_RESET);
                delay_us(2);
                HAL_GPIO_WritePin(GPIOB, Trigger4_Pin, GPIO_PIN_SET);
                delay_us(10);
                HAL_GPIO_WritePin(GPIOB, Trigger4_Pin, GPIO_PIN_RESET);

                while(!(HAL_GPIO_ReadPin(GPIOB, Echo4_Pin)));
                while(HAL_GPIO_ReadPin(GPIOB, Echo4_Pin))
                {
                        pulsetime++;
                        delay_us(1);
                }
                HAL_GPIO_WritePin(GPIOA, MOSFET_4_Pin, GPIO_PIN_RESET);
                if(pulsetime == 0)
                {
                        return Ultrasonic_sensor(4);
                }
                return pulsetime;
                break;

        case 5:
                pulsetime=0;
                HAL_GPIO_WritePin(GPIOA, MOSFET_5_Pin, GPIO_PIN_SET);
                HAL_Delay(1);
                HAL_GPIO_WritePin(GPIOA, Trigger5_Pin, GPIO_PIN_RESET);
                delay_us(2);
                HAL_GPIO_WritePin(GPIOA, Trigger5_Pin, GPIO_PIN_SET);
                delay_us(10);
                HAL_GPIO_WritePin(GPIOA, Trigger5_Pin, GPIO_PIN_RESET);

                while(!(HAL_GPIO_ReadPin(GPIOB, Echo5_Pin)));
                while(HAL_GPIO_ReadPin(GPIOB, Echo5_Pin))
                {
                        pulsetime++;
                        delay_us(1);
                }
```

```
                    HAL_GPIO_WritePin(GPIOA, MOSFET_5_Pin, GPIO_PIN_RESET);
                    if(pulsetime == 0)
                    {
                          return Ultrasonic_sensor(5);
                    }
                    return pulsetime;
                    break;

      }
      return 0;

}


uint16_t CO2_sensor()
{
      ppm_CO2 = 0;
      float a = 5.056743;
      float b = -0.3434871;

      HAL_GPIO_WritePin(GPIOA, MOSFET_6_Pin, GPIO_PIN_SET);
      HAL_Delay(20000);
      HAL_ADC_Start_DMA(&hadc, (uint32_t *) DMA_memory, 2);
      CO2_value = DMA_memory[1];
      HAL_GPIO_WritePin(GPIOA, MOSFET_6_Pin, GPIO_PIN_RESET);

      Voltage_value = CO2_value/4095*5; //Obtener voltaje escalado
      Rs = ((float)RL*(4095-CO2_value)/CO2_value);
      ppm_CO2_float = pow(((Rs/Ro)/a),(1/b)); //Se eleva la operación
(RS/Ro)/a a la potencia (1/b). Ecuación obtenida de la fórmula descrita en la
memoria
      if (ppm_CO2_float >= 900)
      {
            ppm_CO2 = 1;
      }
      return ppm_CO2;
}

uint16_t Battery_level()
{
      bat = 0;
      HAL_ADC_Start_DMA(&hadc, (uint32_t *) DMA_memory, 2);
      Battery = DMA_memory[0];
      Battery = Battery * 3.3 / 4095;
      Battery = Battery / 2.2;
      Battery = (Battery - 1.5) / (-0.006); // Cálculo del porcentaje de
batería a partir de la ecuación de la recta
      if (Battery >= 80)
      {
            bat = 1;
      }
      return bat;
}
void Enter_STOP_Mode(void)
{
      /*## Configure the Wake up timer
###########################################*/
          /*  RTC Wake-up Interrupt Generation:
              Wake-up Time Base = (RTC_WAKEUPCLOCK_RTCCLK_DIV /(LSI))
```

```
               ==> WakeUpCounter = Wake-up Time / Wake-up Time Base

               To configure the wake up timer to 7175s the WakeUpCounter is
set to 0x1FB3C4B:
                 RTC_WAKEUPCLOCK_RTCCLK_DIV = RTCCLK_Div16 = 16
                 Wake-up Time Base = 16 /(37KHz) = 0.000432432 seconds
                 ==> WakeUpCounter = ~14375s/(16 /(37KHz)) = 33.242187E6 =
0x1FB3C4B */

        if (HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 0x1FB3C4B,
RTC_WAKEUPCLOCK_RTCCLK_DIV16) != HAL_OK)
          {
            Error_Handler();
          }
        // We have to suspend the systick before going into STOP mode to
avoid an interrupt that will wake the MCU up
        HAL_SuspendTick();
        //GPIOs disabled
        __HAL_RCC_GPIOA_CLK_DISABLE();
        __HAL_RCC_GPIOB_CLK_DISABLE();
        __HAL_RCC_GPIOC_CLK_DISABLE();
        __HAL_RCC_GPIOD_CLK_DISABLE();
        //UART disabled
        __HAL_RCC_UART4_CLK_DISABLE();
        //Enter to STOP mode
        HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);

}

void Exit_STOP_Mode(void)
{
      SystemClock_Config();
      HAL_RTCEx_DeactivateWakeUpTimer(&hrtc);
      HAL_ResumeTick();
      MX_GPIO_Init();
      MX_UART4_Init();
      MX_ADC_Init();
      MX_RTC_Init();
}

int main(void)
{
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_UART4_Init();
  MX_USART2_UART_Init();
  MX_TIM3_Init();
  MX_RTC_Init();
  MX_ADC_Init();

  HAL_TIM_Base_Start(&htim3);

  uint8_t tx_values[200];
```

```
  uint16_t stringSize;

  while (1)
  {
        HAL_Delay(100);
        Battery_level();
          ppm_CO2 = CO2_sensor();

          ultrasonic_sensor_time = Ultrasonic_sensor(1);
          Distance = ((ultrasonic_sensor_time*0.0343)/2)*calibration_factor;
          if (Distance <= 40)
          {
                counter++;
          }
          HAL_Delay(100);
          ultrasonic_sensor_time = Ultrasonic_sensor(2);
          Distance2 = ((ultrasonic_sensor_time*0.0343)/2)*calibration_factor;
          if (Distance2 <= 40)
          {
               counter++;
          }
          HAL_Delay(100);
          ultrasonic_sensor_time = Ultrasonic_sensor(3);
          Distance3 = ((ultrasonic_sensor_time*0.0343)/2)*calibration_factor;
          if (Distance3 <= 40)
          {
               counter++;
          }
          HAL_Delay(100);
          ultrasonic_sensor_time = Ultrasonic_sensor(4);
          Distance4 = ((ultrasonic_sensor_time*0.0343)/2)*calibration_factor;
          if (Distance4 <= 40)
          {
               counter++;
          }
          HAL_Delay(100);
          ultrasonic_sensor_time = Ultrasonic_sensor(5);
          Distance5 = ((ultrasonic_sensor_time*0.0343)/2)*calibration_factor;
          if (Distance5 <= 40)
          {
               counter++;
          }

          HAL_GPIO_WritePin(GPIOA, MOSFET_7_Pin, GPIO_PIN_SET);
          HAL_Delay(50);
          stringSize =
sprintf(tx_values,"%d,%d,%d,%d,%d,%d,%d,%d%c",Distance,Distance2,Distance3,Di
stance4,Distance5,counter,ppm_CO2,bat,'\n');
          HAL_UART_Transmit(&huart4, tx_values, stringSize, 100);
          HAL_GPIO_WritePin(GPIOA, MOSFET_7_Pin, GPIO_PIN_RESET);
          HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);

          Enter_STOP_Mode();
          Exit_STOP_Mode();

          counter = 0;
  }
}
```

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
  /** Configure the main internal regulator output voltage */
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the CPU, AHB and APB busses clocks */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_LSE
                                |RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_4;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  /** Initializes the CPU, AHB and APB busses clocks */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC;
  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
}

static void MX_ADC_Init(void)
{
  ADC_ChannelConfTypeDef sConfig = {0};

  /** Configure the global features of the ADC (Clock, Resolution, Data
Alignment and number of conversion)*/
  hadc.Instance = ADC1;
  hadc.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc.Init.Resolution = ADC_RESOLUTION_12B;
  hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc.Init.ScanConvMode = ADC_SCAN_ENABLE;
  hadc.Init.EOCSelection = ADC_EOC_SEQ_CONV;
  hadc.Init.LowPowerAutoWait = ADC_AUTOWAIT_DISABLE;
  hadc.Init.LowPowerAutoPowerOff = ADC_AUTOPOWEROFF_DISABLE;
  hadc.Init.ChannelsBank = ADC_CHANNELS_BANK_A;
  hadc.Init.ContinuousConvMode = DISABLE;
  hadc.Init.NbrOfConversion = 2;
  hadc.Init.DiscontinuousConvMode = DISABLE;
  hadc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc.Init.DMAContinuousRequests = ENABLE;
  /** Configure for the selected ADC regular channel its corresponding rank
in the sequencer and its sample time. */
  sConfig.Channel = ADC_CHANNEL_10;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_96CYCLES;
  /** Configure for the selected ADC regular channel its corresponding rank
in the sequencer and its sample time. */
  sConfig.Channel = ADC_CHANNEL_11;
```

```
  sConfig.Rank = ADC_REGULAR_RANK_2;
}

static void MX_RTC_Init(void)
{

  RTC_TimeTypeDef sTime = {0};
  RTC_DateTypeDef sDate = {0};

  /** Initialize RTC Only */
  hrtc.Instance = RTC;
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  /** Initialize RTC and set the Time and Date */
  sTime.Hours = 0x0;
  sTime.Minutes = 0x0;
  sTime.Seconds = 0x0;
  sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
  sTime.StoreOperation = RTC_STOREOPERATION_RESET;
  sDate.WeekDay = RTC_WEEKDAY_MONDAY;
  sDate.Month = RTC_MONTH_JANUARY;
  sDate.Date = 0x1;
  sDate.Year = 0x0;
}

static void MX_TIM3_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 0;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 0xffff-1;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;

}
static void MX_UART4_Init(void)
{

  huart4.Instance = UART4;
  huart4.Init.BaudRate = 9600;
  huart4.Init.WordLength = UART_WORDLENGTH_8B;
  huart4.Init.StopBits = UART_STOPBITS_1;
  huart4.Init.Parity = UART_PARITY_NONE;
  huart4.Init.Mode = UART_MODE_TX_RX;
  huart4.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart4.Init.OverSampling = UART_OVERSAMPLING_16;

}
```

```c
static void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;

}

static void MX_DMA_Init(void)
{
  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();
  /* DMA interrupt init */
  /* DMA1_Channel1_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, Trigger5_Pin|MOSFET_1_Pin|LD2_Pin|MOSFET_2_Pin
                          |Trigger3_Pin|MOSFET_3_Pin|MOSFET_4_Pin|MOSFET_5_Pin
                           |MOSFET_6_Pin|MOSFET_7_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, Trigger4_Pin|Trigger2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(Trigger1_GPIO_Port, Trigger1_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : Trigger5_Pin MOSFET_1_Pin LD2_Pin MOSFET_2_Pin
                           Trigger3_Pin MOSFET_3_Pin MOSFET_4_Pin
MOSFET_5_Pin
                           MOSFET_6_Pin MOSFET_7_Pin */
  GPIO_InitStruct.Pin = Trigger5_Pin|MOSFET_1_Pin|LD2_Pin|MOSFET_2_Pin

|Trigger3_Pin|MOSFET_3_Pin|MOSFET_4_Pin|MOSFET_5_Pin
                          |MOSFET_6_Pin|MOSFET_7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

```
  /*Configure GPIO pins : Trigger4_Pin Trigger2_Pin */
  GPIO_InitStruct.Pin = Trigger4_Pin|Trigger2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : Trigger1_Pin */
  GPIO_InitStruct.Pin = Trigger1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(Trigger1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : Echo2_Pin Echo3_Pin Echo4_Pin Echo5_Pin */
  GPIO_InitStruct.Pin = Echo2_Pin|Echo3_Pin|Echo4_Pin|Echo5_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : Echo1_Pin */
  GPIO_InitStruct.Pin = Echo1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(Echo1_GPIO_Port, &GPIO_InitStruct);
}
```

9.2 Código Heltec ESP32 WiFi LoRa. Parte transmisora del módulo de comunicaciones.

```
#include <heltec.h>
#include <lora/LoRa.h>
#include <Wire.h>
#include <SPI.h>
#include <Streaming.h>
#include <PriUint64.h>
#define WIFI_LoRa_32_V2
#define WIFI_LoRa_32
#define frequency 868E6
#define PABOOST true
#define SCK 5 // GPIO5 -- SX1278's SCK
#define MISO 19 // GPIO19 -- SX1278's MISO
#define MOSI 27 // GPIO27 -- SX1278's MOSI
#define SS 18 // GPIO18 -- SX1278's CS
#define RST 14 // GPIO14 -- SX1278's RESET
#define DI0 26 // GPIO26 -- SX1278's IRQ(Interrupt Request)
int tx = 17;
int rx = 16;
uint8_t spreadingFactor = 9;

uint64_t ID = ESP.getEfuseMac();
```

```
String localizacion = "Rambla de Prim 254";

void setup() {

  //Define inputs and outputs
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);

  //initialize Serial Monitor
  Serial.begin(9600);
  Serial2.begin(9600);

  //SPI LoRa pins
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DI0);

  if (!LoRa.begin(frequency, PABOOST)) {
    //Serial.println("Starting LoRa failed!");
    while (1);
  }
  LoRa.setSpreadingFactor(spreadingFactor);

  delay(100);
}

String separator = ",";
void loop() {

  if (Serial2.available()){

    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the vo
ltage level)
    delay(100);                       // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the
voltage LOW
    delay(100);


    String read = Serial2.readStringUntil('\n');
    int index = read.indexOf(separator);
    int index1 = read.indexOf(separator, index + 1);
    int index2 = read.indexOf(separator, index1 + 1);
    int index3 = read.indexOf(separator, index2 + 1);
    int index4 = read.indexOf(separator, index3 + 1);
    int index5 = read.indexOf(separator, index4 + 1);
    int index6 = read.indexOf(separator, index5 + 1);
    String Distance = read.substring(0, index);
```

```
    String Distance2 = read.substring(index + 1, index1);
    String Distance3 = read.substring(index1 + 1, index2);
    String Distance4 = read.substring(index2 + 1, index3);
    String Distance5 = read.substring(index3 + 1, index4);
    String filling_volume = read.substring(index4 + 1, index5);
    String CO2_sensor = read.substring(index5 + 1, index6);
    String Battery = read.substring(index6 + 1);

    //Send LoRa packet to receiver
    Heltec.LoRa.beginPacket();
    Heltec.LoRa.print(PriUint64<DEC>(ID));
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(Distance);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(Distance2);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(Distance3);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(Distance4);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(Distance5);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(filling_volume);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(CO2_sensor);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(localizacion);
    Heltec.LoRa.print(separator);
    Heltec.LoRa.print(Battery);
    Heltec.LoRa.print('\n');
    Heltec.LoRa.endPacket();

    delay(1000);
  }
}
```

9.3. Código Heltec ESP32 WiFi LoRa de la parte receptora del módulo de comunicaciones.

```
#include <heltec.h>
#include <lora/LoRa.h>
#include <Wire.h>
#include <SPI.h>
#include <Streaming.h>
#include <PriUint64.h>
#define WIFI_LoRa_32_V2
#define WIFI_LoRa_32
```

```
#define frequency 868E6
#define PABOOST true
#define SCK 5 // GPIO5 -- SX1278's SCK
#define MISO 19 // GPIO19 -- SX1278's MISO
#define MOSI 27 // GPIO27 -- SX1278's MOSI
#define SS 18 // GPIO18 -- SX1278's CS
#define RST 14 // GPIO14 -- SX1278's RESET
#define DI0 26 // GPIO26 -- SX1278's IRQ(Interrupt Request)
int counter = 0;
String separator = ",";
int tx = 17;
int rx = 16;
String ID;
String Distance;
String Distance2;
String Distance3;
String Distance4;
String Distance5;
String filling_volume;
String CO2_sensor;
String localizacion;
String Battery;

void setup() {
  // put your setup code here, to run once:
 //Initialize of port Serial to send data to ESP8266
  Serial2.begin(9600);
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
  Serial.begin(9600);
  SPI.begin(5,19,27,18);
  LoRa.setPins(SS,RST,DI0);

  if (!LoRa.begin(frequency,PABOOST)) {
    //Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packets
    // read packet
    while (LoRa.available()) {
      String read = LoRa.readStringUntil('\n');
      int index = read.indexOf(separator);
```

```
      int index1 = read.indexOf(separator, index + 1);
      int index2 = read.indexOf(separator, index1 + 1);
      int index3 = read.indexOf(separator, index2 + 1);
      int index4 = read.indexOf(separator, index3 + 1);
      int index5 = read.indexOf(separator, index4 + 1);
      int index6 = read.indexOf(separator, index5 + 1);
      int index7 = read.indexOf(separator, index6 + 1);
      int index8 = read.indexOf(separator, index7 + 1);
      ID = read.substring(0, index);
      Distance = read.substring(index + 1, index1);
      Distance2 = read.substring(index1 + 1, index2);
      Distance3 = read.substring(index2 + 1, index3);
      Distance4 = read.substring(index3 + 1, index4);
      Distance5 = read.substring(index4 + 1, index5);
      filling_volume = read.substring(index5 + 1, index6);
      CO2_sensor = read.substring(index6 +1, index7);
      localizacion = read.substring(index7 +1, index8);
      Battery = read.substring(index8 + 1);

      String sent = (ID + separator + Distance + separator + Distance2 +
separator + Distance3 + separator + Distance4 + separator + Distance5 + s
eparator + filling_volume + separator + CO2_sensor + separator + localiza
cion + separator + Battery + '\n');
      Serial2.print(sent);
      delay(2000);
    }
  }
}
```

9.4. Código ESP8266 WiFi del módulo de comunicaciones

```
#include <HardwareSerial.h>
#include <ESP8266WiFi.h>
#include <HTTPClient.h>

char* ssid = "MiRed_wiFi";
const char *password = "Contraseña_WiFi";
const char* host = "192.168.1.41";

#define D4 2

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(D4, OUTPUT);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED){
```

```
    delay(500);
    //Serial.print(".");
  }

  for (size_t i = 0; i < 5; i++)
  {
      digitalWrite(D4, LOW);
      delay(50);
      digitalWrite(D4, HIGH);
      delay(50);
      digitalWrite(D4, LOW);
  }
}

int count = 0;
String separator = ",";

void loop() {
  // put your main code here, to run repeatedly:

  if (Serial.available()>0){
    String read = Serial.readStringUntil('\n');
    int index = read.indexOf(separator);
    int index1 = read.indexOf(separator, index + 1);
    int index2 = read.indexOf(separator, index1 + 1);
    int index3 = read.indexOf(separator, index2 + 1);
    int index4 = read.indexOf(separator, index3 + 1);
    int index5 = read.indexOf(separator, index4 + 1);
    int index6 = read.indexOf(separator, index5 + 1);
    int index7 = read.indexOf(separator, index6 + 1);
    int index8 = read.indexOf(separator, index7 + 1);
    String ID = read.substring(0, index);
    String Distance = read.substring(index + 1);
    String Distance2 = read.substring(index1 + 1, index2);
    String Distance3 = read.substring(index2 + 1, index3);
    String Distance4 = read.substring(index3 + 1, index4);
    String Distance5 = read.substring(index4 + 1, index5);
    String Volume = read.substring(index5 + 1, index6);
    String CO2sensor = read.substring(index6 + 1, index7);
    String Location = read.substring(index7 + 1, index8);
    String Battery = read.substring(index8 + 1);

    for (size_t i = 0; i < 5; i++)
    {
      digitalWrite(D4, LOW);
      delay(50);
      digitalWrite(D4, HIGH);
      delay(50);
      digitalWrite(D4, LOW);
```

```
    }
    delay(100);
    ++count;

    WiFiClient client;
    const int httpPort = 80;
    if(!client.connect(host, httpPort)){
      delay(500);
      //Serial.println("connection failed");
      return;
    }

    String url = "http://192.168.1.41/TFM/enviardatos.php";
    String data = String("&ID=" + ID + "&Distance=" + Distance + "&Distan
ce2=" + Distance2 + "&Distance3=" + Distance3 + "&Distance4=" + Distance4
 + "&Distance5=" + Distance5
    + "&Volume=" + Volume + "&CO2sensor=" + CO2sensor + "&Location=" + Lo
cation + "&Battery=" + Battery);

    client.print(String("POST ") + url + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Accept: *" + "/" + "*\r\n" +
                 "content-Length: " + data.length() + "\r\n" +
                 "Content-Type: application/x-www-form-urlencoded\r\n" +
                 "\r\n" + data);

    delay(500);
    Serial.flush();
  }
}
```

9.5. Código PHP

```
<?php
$link = mysqli_connect('localhost', 'root', 'Dan1234alo','monitorizacion');

if(!$link){
        echo "No se ha podido conectar a MySQL";
}
mysqli_set_charset($link, "utf8");

$ID = $_POST ['ID'];
$Distance = $_POST ['Distance'];
$Distance2 = $_POST ['Distance2'];
$Distance3 = $_POST ['Distance3'];
$Distance4 = $_POST ['Distance4'];
$Distance5 = $_POST ['Distance5'];
$Volume = $_POST ['Volume'];
$CO2sensor = $_POST ['CO2sensor'];
$Location = $_POST ['Location'];
$Battery = $_POST ['Battery'];
```

```
$sql = "INSERT INTO container_data (Fecha, ID, Distance, Distance2, Distance3, Distance4,
Distance5, Volume, CO2sensor, Location, Battery) VALUES
(CURRENT_TIMESTAMP,'$ID','$Distance','$Distance2','$Distance3','$Distance4','$Distance5'
,'$Volume','$CO2sensor','$Location','$Battery')";

mysqli_query($link,$sql) or die (mysql_error());


mysqli_close($link);

echo "Datos ingresados correctamente.";
?>
```

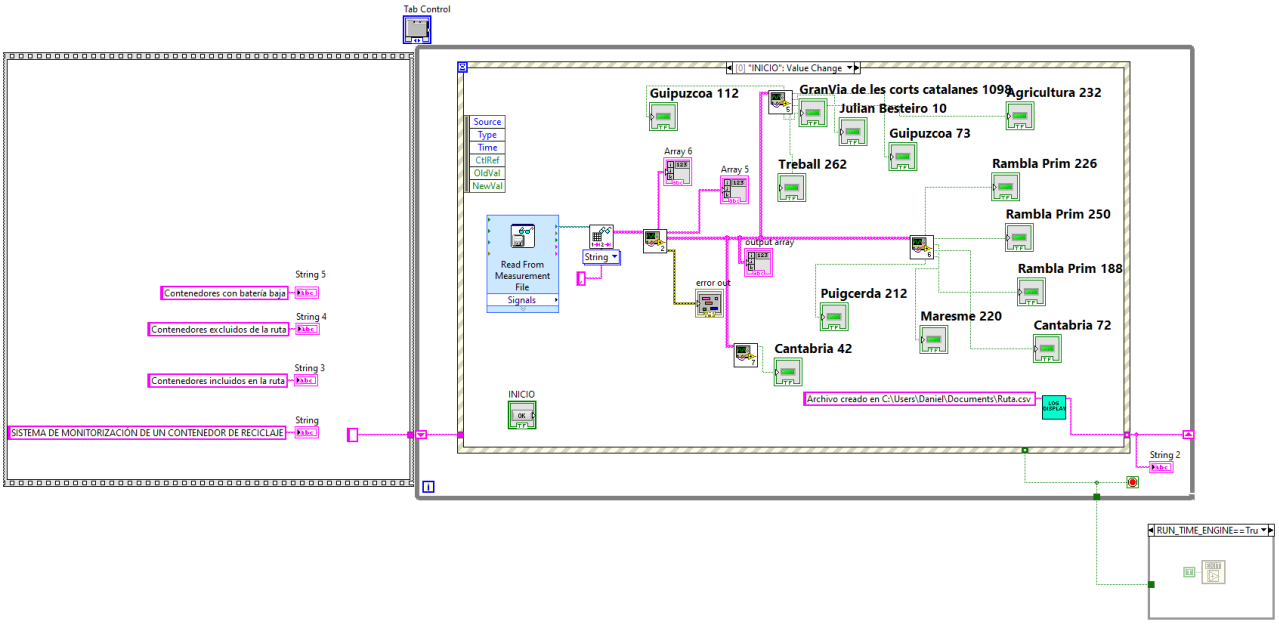
9.6. Código R para el cálculo de los factores *a* y *b* necesarios para la calibración del sensor MQ-135

```
library(data.table)

#remove old variables
rm(list=ls())

#set input values
xlim = c(10, 1000)
ylim = c(0.1, 10)
minppm = 10
maxppm = 200
mres = 4184
mppm = 416
pointsdata = "
10.052112405371744, 2.283698378106183
20.171602728600178, 1.8052797165878915
30.099224396434586, 1.5715748803154423
50.09267987761949, 1.3195287228519417
80.38812026903305, 1.1281218760133969
90.12665922665023, 1.0815121769656304
100.52112405371739, 1.0430967861855598
199.62996638292853, 0.8000946404902397
"

#load points using fread
setnames(points <- fread(pointsdata, sep=",", sep2="\n"), c("x","y"))

#set named list of points, and swapped list of points
#points will be used to plot and compute values as datasheet figure
x <- as.vector(points[,x])
y <- as.vector(points[,y])
points = list(x=x, y=y)


#the nls (Nonlinear Least Squares) it's used to perform the power regression on points
#in order to work, nls needs an estimation of staring values
#we use log-log slope estimation to find intitial values
```

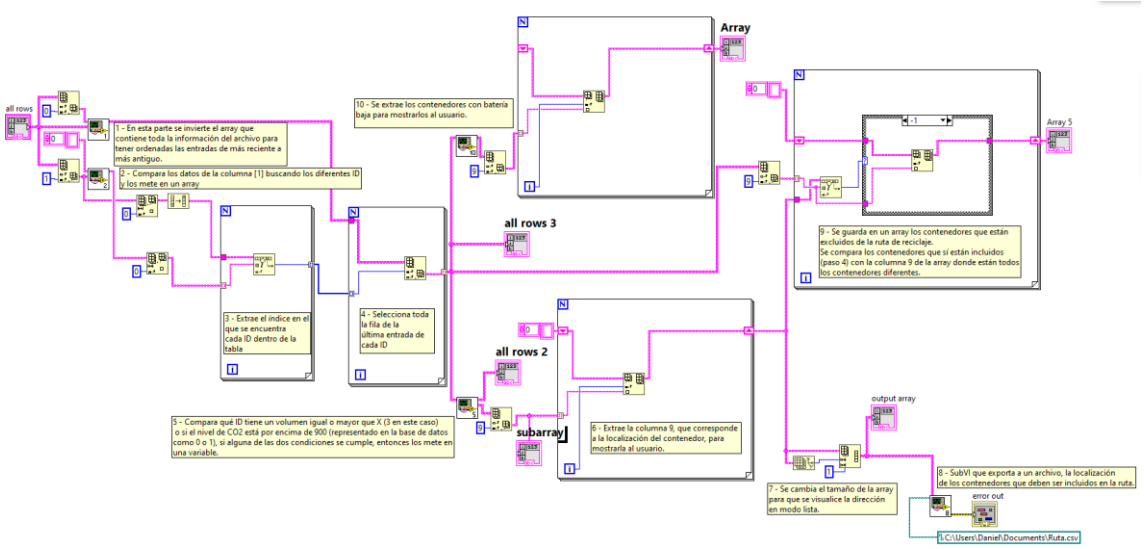
```
#estimate fit curve initial values
xfirst = head(points$x, n=1)
xlast = tail(points$x, n=1)
yfirst = head(points$y, n=1)
ylast = tail(points$y, n=1)
bstart= log(ylast/yfirst)/log(xlast/xfirst)
astart = yfirst/(xfirst^bstart)
#perform the fit
fit <- nls("y~a*x^b", start=list(a=astart,b=bstart), data=points)

#estimate min Rs/Ro
cat("\nCorrelation function coefficients")
cat("\nEstimated a\n")
cat("  ")
cat(coef(fit)["a"])
cat("\nEstimated b\n")
cat("  ")
cat(coef(fit)["b"])
cat("\n")
```

9.7. Código LabView para la aplicación de procesamiento de datos

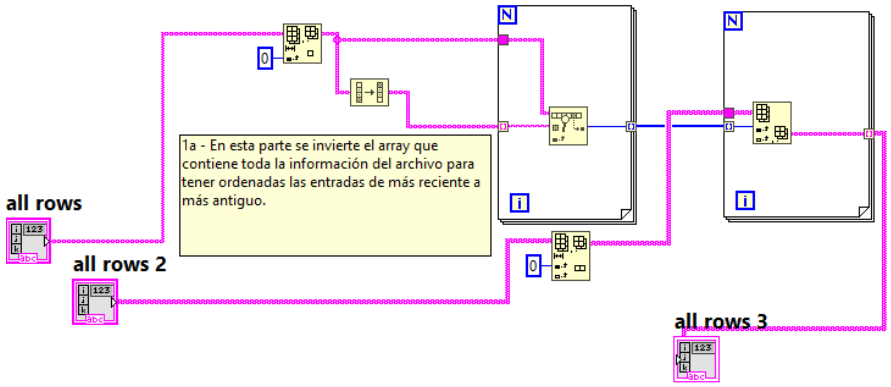
VI interface de la aplicación



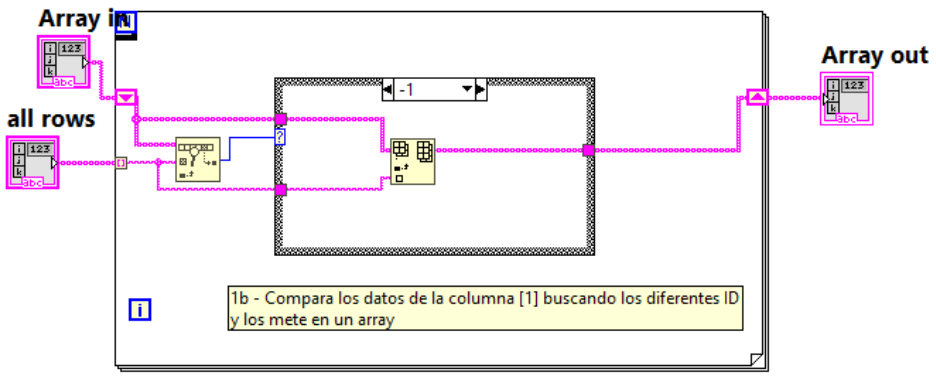
VI código general del cálculo de listas



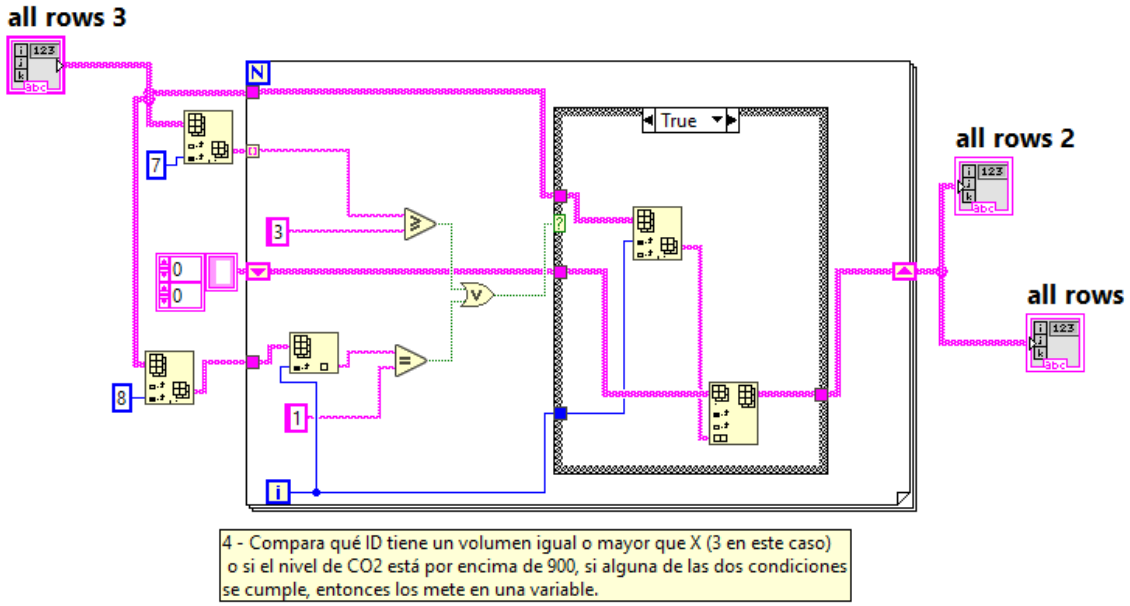
VI para ordenar los registros de los contenedores de más reciente a más antiguo



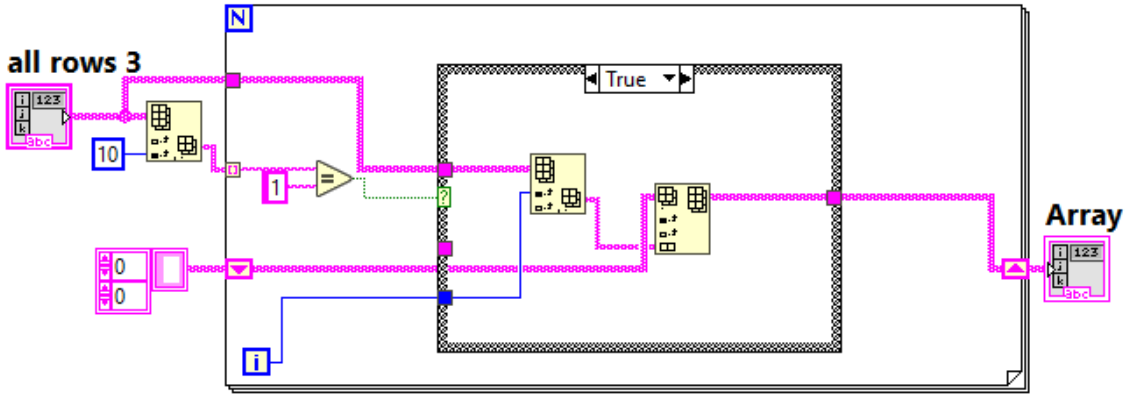
VI donde se extraen los diferentes ID y se meten en una array



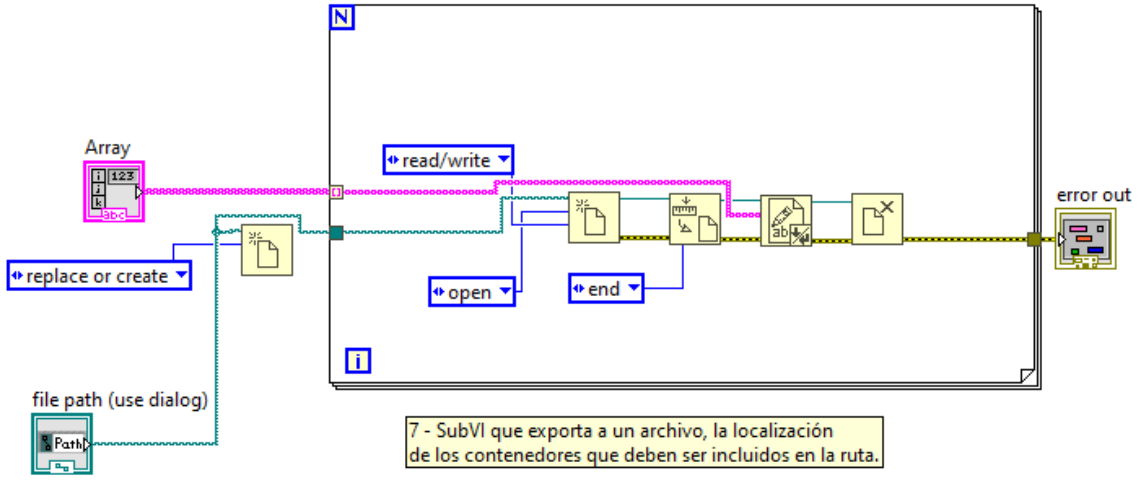
VI para evaluar los niveles de los sensores ultrasónicos y de detección de gas



VI para detectar los contenedores con batería baja



VI para exportar la lista de contenedores incluidos a un archivo



9.8. Drawing soporte y tapa para el módulo de sensores y parte transmisora del módulo de comunicaciones