Universidad Zaragoza
1542

Trabajo Fin de Grado

Monitorización del consumo de agua mediante IoT

Water Consumption Monitoring Using IoT

Autor

Lucas Mallén Zaera

Directores

Julio A. Sangüesa Escorihuela

Pablo Doñate Navarro

Escuela Universitaria Politécnica de Teruel

2024

RESUMEN

La monitorización del consumo del agua en las viviendas es clave para hacer un uso responsable de este recurso e impedir que se desperdicie. Esto se realiza de manera manual mediante los contadores de agua, pero la monitorización remota es un campo para el que hay pocas alternativas, y las que hay son caras o requieren la modificación del sistema de tuberías.

El propósito principal de este proyecto es desarrollar un sistema eficiente y económico para la monitorización del consumo de agua en viviendas. Este sistema tiene como objetivo ofrecer una alternativa asequible a las soluciones comerciales existentes, permitiendo la monitorización remota del gasto del agua, que es esencial para la gestión sostenible de este recurso.

Para ello se ha desarrollado un prototipo de bajo coste que captura una imagen del contador de agua cada cierto tiempo. Este prototipo pretende solucionar los problemas que se pueden encontrar en los lugares donde están los contadores, como la falta de iluminación, de espacio o de conexión a una corriente eléctrica, así como las malas comunicaciones con las redes inalámbricas.

Además se ha realizado un sistema que, mediante el uso de detección de objetos y el reconocimiento de caracteres en imágenes, permite obtener el valor del contador de agua a partir de la imagen capturada.

Los resultados demuestran que es posible utilizar sistemas diferentes a los que se utilizan de manera comercial para la monitorización del agua en los hogares. Además, estos sistemas pueden realizarse con un bajo coste y adaptándose a los posibles problemas que existen en los entornos en los que están los contadores.

PALABRAS CLAVE

Monitorización del consumo de agua, Contadores de agua, LoRa, LoRaWAN, IoT, Diseño 3D, OCR, Inteligencia Artificial, AIOT.

ABSTRACT

Monitoring water consumption in homes is key to promoting responsible use of this resource and preventing waste. Currently, this is done manually through water meters, but remote monitoring is a field with few alternatives, and those that exist are expensive or require modifications to the existing water supply system.

The primary goal of this project is to develop an efficient and cost-effective system for monitoring water consumption in homes. This system aims to provide an affordable alternative to existing commercial solutions, enabling remote control of water usage, which is essential for the sustainable management of this resource.

To achieve this, a low-cost prototype has been developed that captures an image of the water meter at regular intervals. This prototype addresses the challenges often encountered in meter locations, such as poor lighting, limited space, lack of electrical power, and unreliable wireless network connectivity.

Furthermore, a system has been implemented that utilizes object detection and optical character recognition (OCR) to extract the water meter reading from the captured image.

The results demonstrate the feasibility of employing alternative systems for household water monitoring, distinct from those commercially available. These systems can be developed at a low cost and adapted to the specific challenges posed by meter locations.

KEYWORDS

Water consumption monitoring, Water meters, LoRa, LoRaWAN, IoT, 3D design, OCR, Artificial Intelligence, AIOT.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción, motivación y objetivos

1.1. Introducción

El agua es uno de los recursos más importantes para la vida humana, y su gestión eficiente ha sido un factor crucial en la evolución de la sociedad hasta el día de hoy. Gracias a esta gestión, se han conseguido grandes avances que han mejorado la vida de las personas, proporcionando agua potable en los hogares, ayudando a superar sequías a través de los sistemas de almacenamiento o permitiendo su uso en sectores como la agricultura y la ganadería, entre muchos otros.

La importancia de controlar el consumo de agua reside en que, aunque el agua es un recurso muy abundante en el planeta, el agua potable es relativamente escasa y hay muchos factores que pueden llevar a una escasez de esta. Su conservación y uso eficiente son esenciales para garantizar su disponibilidad en todo momento.

Para la correcta gestión del agua es imprescindible llevar a cabo un control de su consumo, ya sea de manera individual, controlando lo que consume cada persona a lo largo del tiempo, o por zonas, controlando el agua consumida en un cierto lugar. De esta manera, se puede saber si este recurso se está utilizando de manera adecuada.

La manera más común mediante la que se monitoriza hoy en día en los hogares son los contadores de agua. Estos dispositivos se instalan en las tuberías y miden el volumen de agua que pasa por ellos, permitiendo así conocer la cantidad que se ha consumido en un periodo de tiempo.

Una de las características de estos sistemas es que se implementan directamente en la tubería por la que entra el agua a las viviendas. Esto provoca que, por lo general, están situados en lugares con muy baja iluminación, falta de corriente eléctrica y a menudo en espacios reducidos y de difícil acceso.

Estos dispositivos son esenciales para poder saber el consumo de agua de cada vivienda, pero tienen un problema: la gran mayoría no permiten conocer el consumo de manera remota, lo que implica que sea necesario desplazarse físicamente para conocerlo.

El hecho de poder ver los datos a distancia implica una serie de beneficios tanto para las personas como para las empresas que manejan el agua, ya que permite, por ejemplo, identificar rápidamente anomalías que indiquen fugas en el sistema de abastecimiento. Esto permitiría tomar medidas para la detección y la reparación de la fuga y evitar que se desperdicie agua y dinero innecesariamente. Además, se puede hacer una idea clara del gasto que implica en la factura de agua, permitiendo ajustar el comportamiento y adoptar medidas para controlar y reducir el consumo si es necesario, generando así ahorros significativos a largo plazo.

Una de las formas que se podría aplicar para conseguir monitorizar el consumo del agua de forma remota sería haciendo uso del internet de las cosas (en inglés Internet of Things, Iot). Este se define como objetos físicos con sensores y capacidad de procesamiento que son capaces de intercambiar datos a través de una red de comunicaciones, como puede ser LoRa [4].

LoRa es una tecnología de comunicación inalámbrica que se ha diseñado para permitir la transmisión de datos a largas distancias con un consumo de energía mínimo. Esto es lo que la posiciona por encima de las demás tecnologías de comunicación, como podrían ser Wifi o Bluetooth, ya que, a la hora de utilizar un dispositivo que se implante en un contador de agua, se van a tener en cuenta las características del entorno en el que está el contador.

Dicho entorno suele ser parecido en todos los casos, y esta caracterizado por tener un difícil acceso a la corriente eléctrica y estar ubicado lugares con muchos obstáculos y en el que difícilmente llegan otras redes de comunicaciones, como puede ser un sótano o una alcantarilla. El bajo consumo de LoRa, junto a su buena capacidad de atravesar objetos físicos la hacen ideal para este trabajo.

Por su parte, el IoT es un sector que ha tenido un gran crecimiento durante la última década. En los últimos años ese crecimiento se ha multiplicado, hasta el punto

que se considera uno de los sectores de mayor impacto para 2025 [5].

Hoy en día el Internet de las Cosas se utiliza en casi todos los ámbitos de la vida, desde el hogar hasta la Industria 4.0. Los sistemas que se utilizan en los hogares tienen un propósito claro, mejorar la vida diaria de las personas. Por ello se suelen enfocar en temas como la seguridad, entretenimiento, accesibilidad o, como es el caso de este trabajo de fin de grado (TFG), la gestión de recursos.

Gracias al IoT en los hogares se puede lograr una gestión más eficiente y precisa de los recursos. Mediante la implementación de dispositivos IoT, se pueden instalar sensores en los sistemas de tuberías o contadores de agua, que obtengan y transmitan datos en tiempo real sobre el consumo de agua a una plataforma centralizada. Esta plataforma puede ser accesible tanto para los usuarios como para las empresas de suministro de agua, permitiendo una monitorización constante y detallada.

Una de las formas más conocidas de mejorar un sistema de IoT es implementarlo junto a un sistema de Inteligencia Artificial (IA) [6] que ofrezca más comodidad en el día a día, por ejemplo, ofreciendo reconocimiento de voz, de imágenes o ayudando a reducir el espacio de almacenamiento del sistema. Una IA es la capacidad que tiene una máquina de realizar tareas para las que normalmente haría falta inteligencia humana, como aprendizaje, adaptación, resolución de problemas etc.

Es de esta unión de donde sale el término Inteligencia Artificial de las Cosas (en inglés Artificial Intelligence of Things, AIoT), el cual se define como la combinación del IoT con la Inteligencia Artificial, donde los dispositivos no solo recopilan datos, sino que también los analizan y actúan sobre ellos de manera inteligente.

Uno de los campos más avanzados de la IA es el reconocimiento óptico de caracteres, o mejor conocido como OCR (del inglés Optical Character Recognition). Este consiste en [7] un método cuyo propósito es el de reconocer caracteres de imágenes y convertirlo a formato digital. Hoy en día se ha extendido en todo el mundo con usos como la traducción automática de imágenes, los radares de velocidad o la digitalización de documentos entre otros. El OCR hace uso de técnicas de aprendizaje automático para el procesamiento de imágenes y reconocimiento de patrones para la identificación del texto.

Asimismo, el aprendizaje automático [8] es un área de la Inteligencia Artificial que permite a los sistemas informáticos mejorar su rendimiento en una tarea específica mediante el análisis de datos, sin necesidad de ser programados explícitamente para

cada situación particular, permitiendo a los sistemas aprender de diferentes patrones para diferentes situaciones.

1.2. Motivación

Los contadores de agua son comunes y eficaces, pero su limitado acceso a los datos de manera remota dificulta su monitorización. Aunque a día de hoy existen contadores más modernos que sí que lo permiten, no es el caso de la gran mayoría de estos, y si se quisiera usar uno se tendría que cambiar todo el sistema, algo costoso tanto en tiempo como en dinero.

Cada vez más ciudades han empezado a sustituir los contadores antiguos por unos modernos con el objetivo de facilitar su monitorización, pero esto es un proceso muy invasivo, ya que es necesario ir a cada uno de los contadores y modificar todo el sistema de tuberías, dejando al propietario sin agua durante un tiempo. Además este es un proceso largo y que lleva un gran gasto económico.

Por ello, otra posible solución es utilizar un dispositivo externo que, implementado junto al contador de agua, obtenga el valor de este y la envíe a través de la red para que se pueda monitorizar. El problema de estos dispositivos comerciales es que tienen un precio muy elevado, lo que hace que no sean muy utilizados.

La monitorización del agua es muy importante por muchas razones, pero el alto coste de llevarla a cabo y los procesos que hay que realizar para implantar algunos de los sistemas hacen que no sea tan interesante. Es por eso que este trabajo de fin de grado se centra en desarrollar una solución que aproveche los beneficios del AIoT para ofrecer un sistema más económico de fabricar e instalar.

1.3. Objetivos

El objetivo principal de este TFG es la monitorización del consumo del agua en las viviendas mediante la combinación de IoT e IA, permitiendo ver el consumo que se ha llevado a cabo a lo largo del tiempo. Se busca desarrollar un prototipo económico, de fácil implementación y que se adecúe a los posibles problemas que ofrecen los entornos en los que están instalados los contadores. Para lograr este propósito será necesario abarcar una serie de objetivos específicos:

- Desarrollo de un sistema que permita obtener una imagen del contador de agua periódicamente mediante un dispositivo de bajo coste.

    - El sistema ha de permitir obtener imágenes en condiciones de baja luminosidad.
    - El dispositivo deberá tener un bajo consumo energético.
    - Se hará uso de redes LoRaWAN (Long Range Wide Area Network) para la comunicación entre el dispositivo y la aplicación final.

- Diseño de una caja o soporte que permita instalar el dispositivo en los contadores de agua. Se tendrá que diseñar esta caja para que sea lo más compacta posible. También deberá permitir la sustitución de la batería de manera sencilla.

- Realización de un sistema que, mediante técnicas de OCR, permita obtener el valor del consumo de agua a partir de una imagen del contador y almacenarlo para su posterior visualización.

- Realización de una plataforma de visualización mediante la que se muestren los valores del consumo de agua que se han reconocido a lo largo del uso del dispositivo, con el objetivo de poder monitorizarlos. La plataforma también permitirá ver el estado de la batería de los dispositivos.

Capítulo 2

Estado del arte

Para poder realizar un sistema que permita la monitorización de los contadores de agua mediante el uso del internet de las cosas es necesario entender primero la forma en la que se están monitorizando a día de hoy los contadores de agua. Es por ello que se ha realizado una búsqueda sobre los principales dispositivos IoT existentes que sirven para monitorizar el agua, así como contadores de agua inteligentes u otros métodos. Además se han explorado algunos de los productos comerciales más populares que permiten la monitorización del agua.

El primer producto "Sensus iPERL" [9] se trata de un contador de agua inteligente que utiliza tecnología electromagnética para medir el consumo del agua de manera precisa. Además permite el envío de datos de alta granularidad en intervalos de 15 minutos, lo que ayuda a la detección de fugas y a la eficiencia en la gestión de las redes de distribución.

Otro producto es el "Contador de pulsos Sigfox" [10] , que no es un contador de agua, sino un dispositivo IoT que, mediante el uso de sensores, es capaz de leer los pulsos emitidos por los contadores de agua, lo que le permite saber el consumo exacto. Gracias a ello puede enviar los datos para su monitorización en tiempo real. Es una solución eficaz para los contadores que cuenten con salida de pulsos.

El artículo "Smart water meter for automatic meter reading" [11] describe el diseño y desarrollo de un medidor de agua inteligente basado en Internet de las Cosas (IoT) para la lectura automática del consumo de agua. El medidor propuesto utiliza un sensor de flujo de agua que permite a los consumidores determinar y controlar la cantidad de agua utilizada, además de enviar datos en tiempo real a través de una aplicación móvil. Este sistema reduce la necesidad de lecturas manuales y mejora la precisión de la

medición con un error máximo de 0.03 litros. El diseño incluye una válvula que se cierra automáticamente cuando se alcanza la cantidad de agua preestablecida, optimizando así la gestión del agua en las ciudades inteligentes.

Otro artículo que se ha encontrado es "Design and Implementation of a Digital Water Meter for Remote Monitoring"[12]. En él se describe el diseño y desarrollo de un medidor de agua digital enfocado en la durabilidad y eficiencia energética. El medidor utiliza sensores HALL para detectar el consumo de agua mediante la medición del campo magnético generado por la rotación de un impulsor. Los datos de consumo se transmiten de manera inalámbrica utilizando ZigBee, lo que permite una comunicación efectiva incluso desde ubicaciones subterráneas. El diseño se centra en mantener el consumo de energía lo suficientemente bajo como para que el medidor funcione durante más de 8 años sin necesidad de reemplazar la batería.

Por último el artículo "Compact Smart Water Meter Development for Smart City" [13] presenta el desarrollo de un medidor de agua inteligente compacto diseñado para ser implementado en ciudades inteligentes. Utilizando un microcontrolador ESP32-CAM, el dispositivo se coloca encima del medidor de agua mecánico existente sin necesidad de reemplazarlo. Este dispositivo captura imágenes del medidor y las envía a través del wifi a un servidor web donde se procesan utilizando Google Vision para la lectura de caracteres ópticos (OCR). Los resultados se muestran en una aplicación web que permite a los usuarios ver sus lecturas y facturas, así como realizar pagos en línea. Este sistema reduce el tiempo y los costes asociados con la gestión de facturas de agua y ayuda a los usuarios a monitorizar su consumo y detectar fugas.

Con todo esto, se puede llegar a la conclusión de que existen tanto productos comerciales como estudios que se centran en la monitorización remota del consumo del agua, ya sea mediante contadores inteligentes o mediante productos que permiten el control del consumo al integrarlos junto con el contador. Sin embargo, ninguno de ellos tiene un enfoque en el que contemplen al mismo tiempo el coste reducido, la falta de iluminación, la falta de corriente eléctrica y la dificultad de conexión a redes.

Capítulo 3

Funcionamiento del sistema

En este capítulo se va a explicar en detalle el funcionamiento del sistema. Para ello, se ha dividido la sección en dos apartados en los que se explicará el hardware y software que se ha utilizado, así como la arquitectura y el funcionamiento del sistema que se ha desarrollado.

3.1. Elementos Hardware y Software

En esta sección se van a presentar los elementos hardware y software más relevantes que se han utilizado en este proyecto, explicando cada uno de ellos y justificando su elección para este trabajo. Por motivos de espacio se puede observar más en detalle cada uno de los elementos mostrados en este apartado en el Anexo I.

3.1.1. Hardware

En el caso del hardware se han utilizado tres dispositivos: el microprocesador con el que se manejan y envían los datos, la cámara con la que se obtiene la imagen y un relé que se utiliza para optimizar el consumo de energía del dispositivo.

El primer dispositivo hardware que se ha utilizado ha sido el Heltec Wifi LoRa 32 v3 [14], el cual es un módulo de comunicaciones diseñado para aplicaciones en el IoT que tiene integrado un microcontrolador ESP32, un módulo LoRa y soporte tanto para el uso de wifi como de bluetooth. También cuenta con una pantalla Oled integrada que facilita la visualización de los datos. Este dispositivo permite la conexión con diferentes módulos mediante los que puede obtener información, como puede ser un

módulo de cámara. Además permite procesar dicha información y enviarla a través de LoRa requiriendo un consumo de energía reducido. j

Otro elemento hardware que se utiliza es el módulo de la cámara, utilizado para capturar las imágenes del contador. Para la elección de este componente se han realizado varias pruebas, tanto con diferentes cámaras como con diferentes lentes en la misma cámara.

Las cámaras que se han evaluado son: Arducam Mini 2mp Plus [15], VGA OV7670 con FIFO [16] y VGA OV7670 sin FIFO [17]. Entre ellas se ha optado por utilizar la Arducam Mini 2mp Plus, debido a que ofrece mayor variedad de resoluciones, comprendidas desde 160x120 píxeles hasta 1600x1200 píxeles. También permite elegir entre formatos de imagen JPEG, BMP y RAW, lo que hace que no sea necesario procesar la imagen para transformarla a alguno de estos formatos en el microcontrolador. Además, ofrece librerías que facilitan el uso de la cámara desde múltiples lenguajes, entre ellos Arduino IDE.

En el caso de las lentes se han realizado pruebas con las que venían integradas con las cámaras, en concreto se han utilizado la lente por defecto del Arducam Mini 2mp Plus, la del VGA OV7670 con FIFO, y una lente infrarroja CCTV [18]. Tras realizar las pruebas se optó por utilizar la lente que venía integrada con la Arducam Mini 2mp Plus debido a que la calidad de la imagen era muy superior a las demás. Además, gracias a que se ha podido utilizar el led del microcontrolador como si fuera un flash, se ha optado por no utilizar una cámara infrarroja, ya que para ello sería necesario un led especial que implicaría un mayor coste económico.

Por último, con la intención de reducir el consumo, dado que la cámara consumía energía independientemente del estado del microcontrolador, se ha utilizado un relé. Este se utiliza para conectar el microcontrolador con el módulo de la cámara. Un relé funciona como un interruptor digital, el cual permite, o no, el paso de la corriente dependiendo de la señal que le llegue. El relé que se ha utilizado ha sido un relé IM 01 [19], el cual permite alimentar la cámara únicamente cuando se activa el GPIO correspondiente. De esta manera cuando se quiera encender la cámara únicamente se tiene que activar dicho GPIO y la corriente pasará del microcontrolador a la cámara, permitiendo su uso.

3.1.2. Software

En el caso del software hay más cantidad de elementos que han de ser mencionados, como los programas que se han utilizado para programar en el dispositivo, las librerías de OCR más relevantes, el servidor de red que se ha utilizado para la comunicación LoRa, MQTT, la base de datos que se ha realizado y la plataforma utilizada para la visualización de los resultados.

El entorno de desarrollo utilizado ha sido el Arduino IDE (Integrated Development Environment) [20], este es una plataforma de desarrollo de software diseñada específicamente para la programación de microcontroladores Arduino. Proporciona un entorno intuitivo y accesible que permite a desarrolladores escribir, compilar y cargar código en placas Arduino de manera eficiente. Este entorno se ha elegido frente a otros como MicroPython [21] o PlatformIO [22] ya que tiene un gran soporte y comunidad que lo respalda, lo que ha generado una gran cantidad de librerías que simplifican la integración del software y el uso de elementos como la antena LoRa. Entre estas librerías se pueden destacar sobretodo dos: la de "ArduCAM" [23] y la de "Heltec_ESP32_Lora_V3" [24].

Otro elemento importante han sido las librerías utilizadas para llevar a cabo el reconocimiento óptico de caracteres, entre dichas librerías se pueden destacar dos: Pytorch [25] y Tensorflow [26]. Ambas se utilizan para la ejecución de las redes neuronales realizadas para el reconocimiento de los caracteres del contador de agua.

Por un lado, PyTorch es una biblioteca de aprendizaje profundo de código abierto desarrollada principalmente por Facebook's AI Research lab (FAIR) [27], que permite la construcción y modificación dinámica de modelos, facilitando la experimentación y la depuración.

Por otro lado, TensorFlow es una plataforma de aprendizaje automático de código abierto desarrollada por Google AI [28]. El funcionamiento de TensorFlow se basa en la construcción y ejecución de gráficos de flujo de datos, donde los nodos representan las operaciones matemáticas y los datos, en forma de tensores, fluyen entre ellos.

Como servidor de red para la comunicación LoRa se ha optado por utilizar ChirpStack [29]. Este es una solución de código abierto diseñado para implementar y gestionar redes LoRaWAN. Su principal función es actuar como una infraestructura middleware entre los gateways LoRaWAN y las aplicaciones que consumen los datos de los dispositivos.

ChirpStack es necesario para gestionar los paquetes que se envían por la red, ya que este los procesa, elimina los duplicados, los decodifica y los publica en MQTT (Message Queuing Telemetry Transport) [30] entre muchas otras cosas.

En el caso de la base de datos que se ha utilizado para almacenar los resultados del OCR y las imágenes, se ha optado por InfluxDB [31]. Esta es una base de datos de series temporales (TSDB) diseñada para gestionar grandes volúmenes de datos generados a lo largo del tiempo. Organiza los datos en series temporales, donde cada punto tiene un tiempo asociado. Estos puntos se agrupan en "measurements"(similares a tablas) y se pueden etiquetar para facilitar las búsquedas. Utiliza un modelo de almacenamiento eficiente para escrituras rápidas y ofrece un lenguaje de consulta (InfluxQL) para analizar los datos temporales de manera efectiva.

Se ha elegido esta base de datos debido a varias razones: la primera es que se basa en series temporales, lo cual es ideal para almacenar eventos, como es el caso del sistema que se está realizando. Otra razón es su facilidad de visualización y monitorización, ya que InfluxDB se integra perfectamente con sistemas de visualización como puede ser Grafana. La última razón es que ofrece una fácil integración con herramientas como Chipstack o lenguiajes de programación como Python, lo que facilita el proceso de añadir datos.

Por último, se ha utilizado Grafana [32] para la visualización de la información. Grafana es una plataforma de software libre de código abierto diseñada para la visualización y el análisis de datos en tiempo real. Se utiliza principalmente para crear gráficos, paneles interactivos y alertas, permitiendo a los usuarios monitorizar métricas provenientes de diversas fuentes de datos. Grafana cuenta con una integración nativa con InfluxDB, lo que facilita el proceso de comunicación entre el sistema de visualización y el de almacenamiento de los datos. Además, cuenta con un sistema de alertas, el cual permitiría avisar a los usuarios en caso de que se detecte alguna anomalía, permitiendo así actuar lo antes posible ante posibles fugas de agua u otros problemas que se puedan detectar.

3.2. Arquitectura del sistema

Una vez detallados los elementos hardware y software más importantes que se van a utilizar en este proyecto, se va a proceder a explicar el funcionamiento del sistema que se ha realizado.

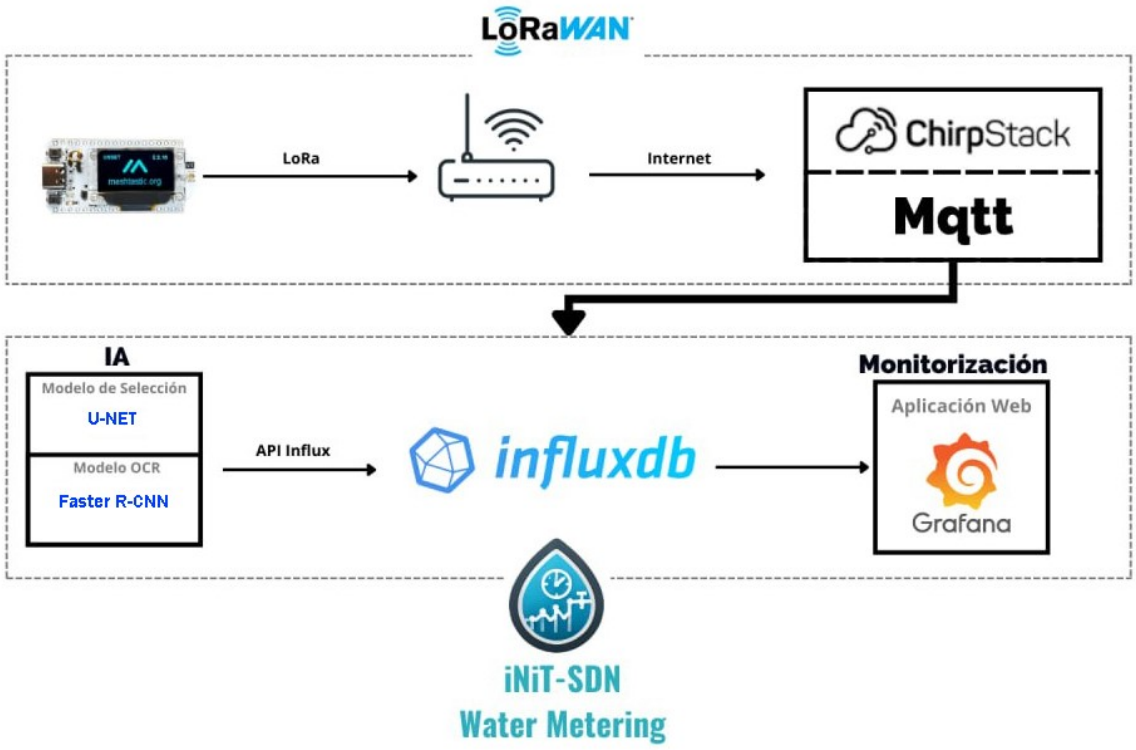

Figura 1: Arquitectura del sistema propuesto

Como se puede observar en la arquitectura mostrada en la Figura 1, el sistema empieza con el dispositivo obteniendo una imagen del contador y procesándola para preparar el envío de los paquetes por LoRa. A continuación, envía la imagen y los gateways que la reciben la reenvían al servidor de ChirpStack. En este servidor se procesan los paquetes decodificándolos, eliminando duplicados, etc. y son publicados en MQTT.

La aplicación, que está suscrita a esa comunicación MQTT, obtiene la imagen y la procesa mediante los modelos implementados para realizar el OCR. Tras obtener el valor reconocido, almacena en la base de datos de InfluxDB tanto la imagen como el valor asociado a la misma.

Al mismo tiempo, la aplicación web utiliza el framework de Grafana para mostrar los datos reconocidos del consumo del agua a lo largo del tiempo, permitiendo así su monitorización.

El funcionamiento del sistema se va a abordar más en detalle en las siguientes subsecciones.

3.2.1. Comunicación LoRaWAN

Para entender el funcionamiento del sistema, es necesario empezar explicando el tipo de red de comunicaciones que se ha utilizado, ya que de esta depende el rendimiento del resto del sistema.

Para este proyecto se va a utilizar LoRaWAN (Long Range Wide Area Network), que es [3] un protocolo de red de área amplia que está estandarizado por la LoRa Alliance. Se caracteriza principalmente por ofrecer una comunicación de largo alcance y tener un bajo consumo energético. Esto lo hace ideal para dispositivos que necesitan mandar datos de manera intermitente. Se puede ver más información en el Anexo II.

El uso de LoRaWAN en este proyecto se debe a varias razones. Para comenzar, es un sistema de bajo consumo de energía, lo que lo hace perfecto para los lugares en los que se van a introducir estos dispositivos, sin conexión eléctrica y de difícil acceso. Gracias a ello se podrá ahorrar energía y alargar más el tiempo de uso de la batería. Además, tiene una gran capacidad para sobrepasar objetos físicos, lo que facilita la comunicación de los dispositivos ubicados en lugares como sótanos o alcantarillas.

Uno de los problemas que se podían dar a la hora de utilizar una comunicación LoRa era que, al utilizar una banda abierta, se ha de hacer un uso responsable de la misma, por lo que se ha establecido que no se puede ocupar la red más del 1 % del tiempo en el que se utiliza el dispositivo [33]. Esto quiere decir que si se envía una imagen al día, el tiempo en el que se están enviando paquetes no puede superar los 864 segundos al día (1 % de 24 horas).

Para comprobar la cantidad de imágenes que se pueden enviar en un día, se han tenido en cuenta dos factores: la cantidad de paquetes que se necesitan para enviar una imagen y el tiempo de transmisión de cada uno de ellos (se detalla más el funcionamiento de este sistema en los siguientes apartados).

Para saber la cantidad de paquetes que se pueden enviar se ha de tener en cuenta que en la comunicación LoRaWAN es posible ajustar el Spreading Factor (SF), lo que determina el tamaño del paquete, la velocidad del envío y la distancia de este. Contra menor sea el Spreading Factor, mayor será el tamaño de los paquetes y la velocidad del envío, pero menor será la distancia que recorra.

Los dispositivos que se han empleado en este proyecto cuentan con un SF de 7 y un ancho de banda de 125 kHz. Esto quiere decir que se pueden llegar a transmitir 222

Bytes de “payload” por paquete. Se ha decidido utilizar este Spreading Factor ya que, en estos entornos urbanos, las distancias no van a ser muy grandes.

Además, se ha observado que el tamaño de las imágenes de 160x120 píxeles nunca ha superado los 3500 Bytes, por lo que, como se pueden enviar hasta 222 Bytes de payload en cada paquete, se ha supuesto que el máximo número de paquetes necesario para enviar una imagen es de 16.

Por otra parte, se ha utilizado una calculadora [34] con la que se ha obtenido el tiempo de transmisión de cada uno de estos paquetes, los cuales contienen 222 Bytes de payload y se envían en SF 7. El resultado ha sido que cada envío de paquete ocupa la red 368.9 ms, por lo que se pueden llegar a enviar 81 paquetes por día. Con todo ello se puede ver que, a lo largo de un día, se podrían llegar a enviar aproximadamente cinco imágenes cumpliendo con el uso responsable de la red.

3.2.2. Funcionamiento del dispositivo

El dispositivo es la parte del sistema que se coloca en el contador de agua y permite obtener el valor del consumo. En concreto, el dispositivo realiza una imagen del contador para que sea procesada y se pueda obtener dicho valor.

La primera opción que se planteó a la hora de realizar este trabajo fue la de realizar el procesamiento del OCR directamente dentro del dispositivo, lo que se conoce como ”Edge Computing”. Este método tenía varias ventajas, como que únicamente sería necesario enviar un número por la red LoRa y, por ello, tardaría menos en enviar los datos, ahorrando tiempo de ejecución. Además, no sería necesario toda la lógica del servidor, en el que se reconstruye la imagen, ya que este solo debería de almacenar el valor recibido.

Desde un principio se intentó realizar de esta manera, pero surgió un problema ya que el almacenamiento del dispositivo era únicamente de 8 MB. Esto implicaba que el dispositivo no pudiera ni siquiera almacenar la imagen en un formato que no fuera JPEG. Además, ambas redes neuronales que se han necesitado para el reconocimiento de caracteres en los contadores de agua, que se explicarán más adelante, superan ese tamaño por separado, lo cual hace que ni siquiera se haya podido reducir la imagen antes de enviarla por la red.

Es por ello que se ha acabado optando por la opción de enviar la imagen con la

menor resolución posible dividida en varios paquetes y reconstruirla en el servidor.

Como se puede ver en la figura 2, el dispositivo cuenta con cuatro módulos conectados entre ellos para realizar las funciones de toma, procesamiento y envío de la imagen. Estos componentes son:

- Heltec Wifi Lora 32 V3: es el microprocesador que tiene el programa que se está utilizando y controla los demás componentes, cuenta con una antena LoRa para el envío de mensajes y está conectado al módulo de la cámara mediante el que obtiene la imagen.
- Arducam Mini 2MP Plus: es el módulo de la cámara que permite tomar fotos, que son recogidas por el microprocesador. Existe la posibilidad de configurarlo para modificar valores como la resolución o el formato en el que se van a enviar las imágenes al microprocesador. No tiene la opción de apagarlo, por lo que se utiliza un relé para activarla y desactivarla.
- Relé: es un dispositivo que se utiliza para controlar el encendido y apagado de la cámara. Funciona como un interruptor pero que se enciende y se apaga de manera electrónica.
- Batería: se utiliza una batería de 2000 mAh para alimentar el dispositivo y permitir que se utilice sin estar conectado a la corriente.

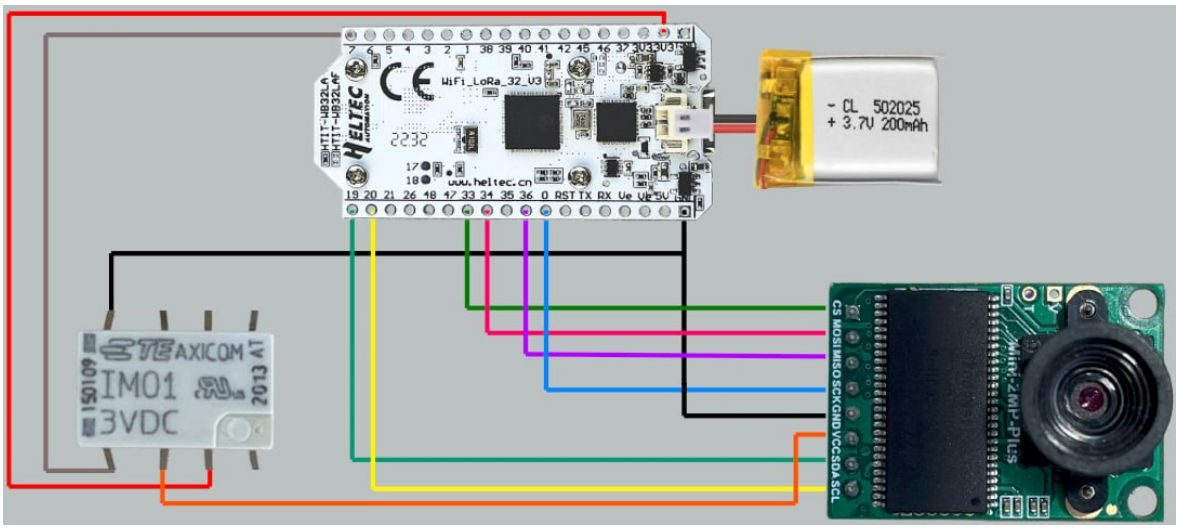

Figura 2: Arquitectura del dispositivo

En un primer momento el dispositivo no iba a contar con un relé, pero tras realizar una serie de pruebas del consumo de la batería (que se pueden ver en el Anexo IV), se

llegó a la conclusión de que la mejor manera de controlar el consumo de la cámara e impedir que se malgaste la batería era haciendo uso de este. Esto se debe a que, si no se hace uso del relé, el microprocesador está en todo momento dando energía al módulo, y esto provoca que, pese a estar en el modo de ahorro de batería, se siga consumiendo batería. Con el uso de el relé, se puede elegir en qué momento el microprocesador le da energía al módulo de la cámara, por lo que se puede encender únicamente para obtener la imagen e inmediatamente después apagarse para ahorrar energía.

Una vez se ha visto la parte física del dispositivo, se va a pasar a ver el funcionamiento del software desarrollado.

Al iniciar el dispositivo, este va a iniciar la cámara y la va a configurar para que se realicen las imágenes de la manera requerida. Además va a inicializar la comunicación LoRaWAN y va a enviar un mensaje "JoinRequest" para confirmar que la comunicación funciona correctamente. Una vez iniciadas correctamente tanto la cámara como la comunicación, se procede a empezar con el envío de las imágenes.

Como se ha visto en el apartado anterior, las comunicaciones LoRaWAN que se utilizan sólo soportan el envío de 222 bytes de datos, utilizando un "Spreading Factor" de 7. Esto afecta a la forma de realizar el sistema ya que, aunque se consiguiera hacer el OCR de las imágenes más pequeñas que pueda obtener el módulo de la cámara, superarían el peso máximo de envío de datos de LoRaWAN. Es por ello que es necesario realizar una segmentación de la imagen para poderla enviar en diferentes mensajes.

Como se puede ver en la Figura 3, esta segmentación se realiza en bloques de 221 bytes, ya que el primer byte de cada paquete se utiliza para indicar el número de paquete. Con ese byte inicial, el servidor puede reconstruir la imagen y, en caso de que se haya perdido algún paquete, pedirle al dispositivo el reenvío de ese paquete.

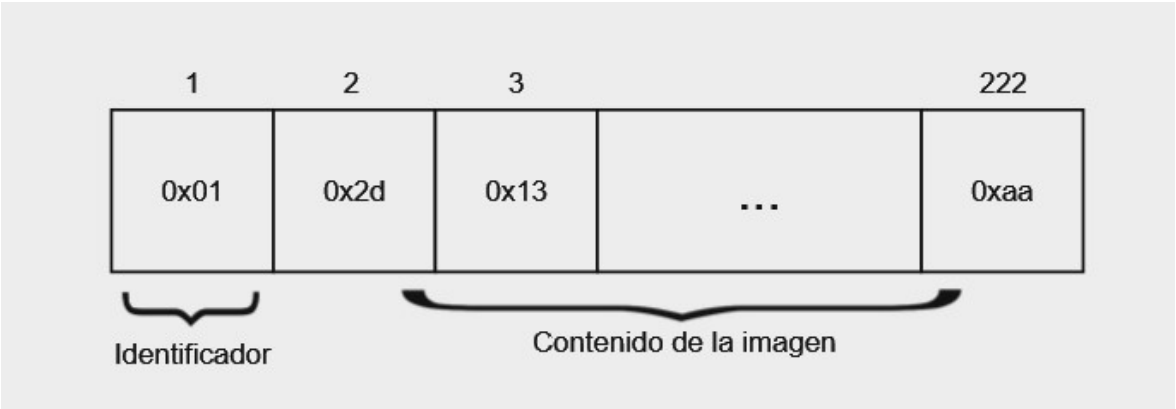

Figura 3: Estructura de los paquetes enviados mediante LoRa

Una vez se ha visto cómo se organizan los paquetes que se envían hacia el servidor se va a explicar el funcionamiento básico del dispositivo. Este funciona en un bucle constante que se repite cada cierto tiempo. En este bucle lo primero que hace es capturar la imagen, para ello el microcontrolador envía la señal de tomar una foto a la cámara y esta le devuelve la foto en formato JPEG.

Tras obtener la imagen, la secciona, añade el número de paquete y lo envía. Una vez enviado se queda en una pausa que le permite esperar a que se reciba correctamente el paquete sin gastar un exceso de batería. Tras esa pausa se despierta y procede a enviar el siguiente mensaje, añadiendo el número de paquete correspondiente. Realiza esta acción hasta que finaliza el envío de la imagen.

La manera en la que funciona la comunicación en este dispositivo es que él, nada más enviar un mensaje por la red, escucha por si se le ha enviado alguno a él. De esta forma existe una manera de comunicarse con él. Esto es esencial en un sistema en el que se necesita recibir de manera correcta todos los mensajes que se envían, ya que se utiliza este sistema para que, en caso de que el servidor no haya recibido alguno de los paquetes de la imagen, le haga una petición al dispositivo para que se lo vuelva a enviar.

Una vez enviado el último paquete realiza una pausa más larga, esperando a que pase el tiempo necesario para enviar otra imagen.

Durante ese tiempo se despierta una última vez, poco después de finalizar el envío de la imagen y envía un mensaje de confirmación. Este mensaje es necesario ya que permite al servidor informar al dispositivo si se ha perdido algún paquete. En caso de que el dispositivo haya finalizado el envío de la imagen y el servidor no la haya recibido entera, por ejemplo por que falta el último paquete, este aprovecha el envío del paquete de confirmación, ya que, tras enviarlo, el dispositivo se queda escuchando, lo que permite que se le pida el paquete que se ha perdido.

Por otro lado, se aprovecha el envío de este mensaje para enviar el estado de la batería, permitiendo así estimar el tiempo que durará el dispositivo sin necesitar un cambio de batería.

Para que el servidor no confunda este mensaje con los mensajes de imágenes se utiliza un primer byte especial, el 255. En el sistema explicado anteriormente, este número haría referencia a que es el paquete número 255, pero como se ha visto anteriormente, las imágenes no van a contener más de 16 paquetes.

3.2.3. Funcionamiento del servidor

El servidor es la parte del sistema que se encarga de leer los paquetes enviados por el dispositivo y procesarlos, consiguiendo así obtener los datos de la imagen o el estado de la batería.

La aplicación se conecta mediante MQTT para poder leer los paquetes enviados por el dispositivo y hace una conexión a una base de datos de InfluxDB.

Cada vez que se recibe un paquete lo procesa. Este puede ser de dos tipos: paquete de envío de imagen o paquete de confirmación. Para seleccionar cuál de los dos tipos es, se divide el primer byte y se comprueba que número es. Se utiliza el número 255 en el primer byte del paquete para representar que es un paquete de confirmación, ya que una imagen nunca va a necesitar tantos paquetes. Si se recibe, se obtiene el número del estado de la batería y se almacena. En caso de que la imagen no esté completa se envía una petición para que se vuelva a mandar el paquete que falte.

En caso contrario, se utiliza ese primer byte para comprobar el número de paquete que es y se almacenan los datos en la posición correspondiente. Una vez almacenados, se comprueba si se ha perdido algún paquete para pedir que se vuelva a enviar. En caso de que se hayan obtenido todos los paquetes de la imagen se procede a reconstruirla para su procesamiento.

Para ello se colocan los datos de la imagen en el orden correspondiente y se almacena. Una vez almacenada se realiza el reconocimiento de los números del consumo del agua a partir de la fotografía obtenida. Finalmente, se almacenan en la base de datos tanto la imagen como el valor reconocido.

La base de datos que se utiliza contiene únicamente dos tablas. En la primera contiene el id del dispositivo utilizado y el estado de la batería a lo largo del tiempo y en la segunda tiene el id del dispositivo utilizado, la imagen y el valor de esa imagen a lo largo del tiempo.

3.2.4. Visualización de la información

Una vez visto el funcionamiento principal del sistema, se va a explicar la aplicación final que utilizarían los usuarios para ver la información de sus contadores de agua. Esta es una aplicación web en la que se mostrarían los datos obtenidos de todos los

dispositivos asociados a cada usuario.

Para la realización de esta aplicación se han creado previamente dos dashboards de Grafana, uno que contiene una gráfica con los datos obtenidos de la predicción del consumo de agua, y otro que contiene otra gráfica con los datos obtenidos de la batería del dispositivo. Para obtener los datos de cada uno de esos dashboards se ha realizado una conexión entre InfluxDB y Grafana, mediante el uso de la API de InfluxDB, se han introducido las credenciales necesarias para el acceso a los datos, así como el "token"de la aplicación, el cual sirve para identificar el bucket del que se quieren obtener los datos. Tras ello, se han realizado dos consultas en InfluxQL para obtener una gráfica con los valores del reconocimiento de los caracteres y otra gráfica con los valores del consumo de la energía.

Tras crear los dashboards con las consultas necesarias se han utilizado para integrar dichas gráficas para realizar la visualización de los datos.

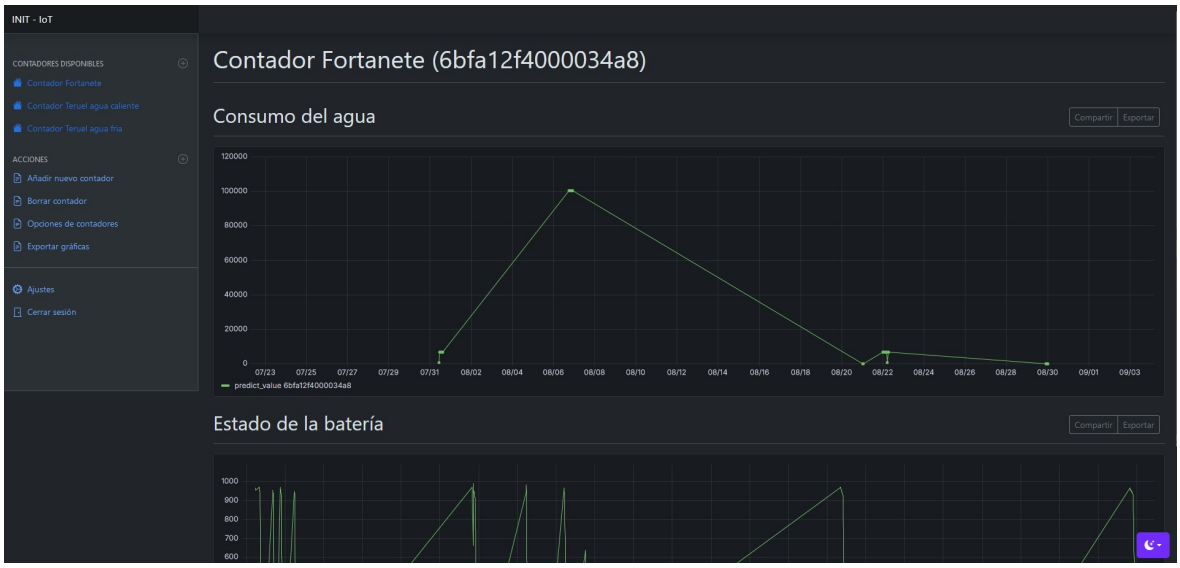

Figura 4: Web de visualización de los datos

Como se observa en la Figura 4, gracias a esta aplicación se puede monitorizar el consumo de agua, ya que permite ver en tiempo real el valor que se está registrando. Además, gracias a la gráfica del estado de la batería, se puede predecir el momento en el que será necesario reemplazarla o cargarla. Gracias a ello, se puede conseguir que el dispositivo esté el mínimo tiempo posible apagado.

En el caso de ambas gráficas existe la posibilidad de ampliar o reducir la sección de tiempo en la que se muestran los resultados, permitiendo así elegir los días para los que se quieren ver los resultados.

Existen dos mejoras relevantes que se podrían implementar para una posible futura versión de la aplicación que son:

- La posibilidad de conectar múltiples dispositivos en una misma cuenta. Esto sería necesario ya que cada persona no tiene únicamente un contador de agua, sino que puede tener uno o varios por casa. La posibilidad de conectar varios dispositivos en la misma cuenta facilitaría la monitorización del consumo de agua.
- El uso de un sistema de alertas que avisen a los usuarios en caso de detectar anomalías. Esto es esencial para el propósito de la monitorización del agua, ya que avisaría a los usuarios en caso de que el sistema detectara alguna anomalía como puede ser un cambio muy grande del consumo en poco tiempo. Gracias a este sistema se podrían detectar fallos en las tuberías o robos de agua.

Capítulo 4

Resultados

Una vez se ha visto cómo funciona la arquitectura del sistema que se ha propuesto, se va a comprobar su correcto funcionamiento. Para ello, lo primero que se ha hecho ha sido obtener los modelos para realizar el OCR y comprobar cuál es el mejor modelo para el reconocimiento del valor del consumo de agua en los contadores.

Tras obtener el mejor modelo para conseguir saber el valor del consumo del agua a partir de la imagen de un contador, se ha puesto a prueba en un entorno real, construyendo un modelo 3D y realizando pruebas sobre un contador de agua.

En este apartado se van a ver las pruebas realizadas, así como los resultados que se han obtenido en cada caso, concluyendo con unas pruebas que pretenden simular el entorno real en el que se instalaría el dispositivo.

4.1. Mecanismo de reconocimiento de caracteres

Uno de los retos principales de este proyecto ha sido el de crear un sistema que permita el reconocimiento de los datos del contador a partir de una imagen.

Como se ha podido observar en uno de los apartados anteriores, las redes LoRaWAN que se han utilizado durante el proyecto solo permiten el paso de 222 Bytes por cada paquete. Esto condiciona el trabajo y genera un nuevo objetivo, el de crear un sistema de reconocimiento de caracteres que funcione con imágenes del menor tamaño posible.

Para llevar a cabo el OCR y aumentar las posibilidades de acierto se han realizado dos modelos diferentes que trabajan en conjunto. El primer modelo se encargará de seleccionar la zona de la imagen en la que están ubicados los números del contador, a

este modelo se le introducirá la fotografía completa y devolverá una máscara de ésta con la sección en la que están los números. El segundo modelo será el encargado de realizar el reconocimiento de los dígitos, se le pasará una imagen, que será la original recortada en función de lo que haya devuelto el primer modelo, y devolverá el valor del consumo del agua. En la Figura 5 se puede ver el funcionamiento de este sistema.

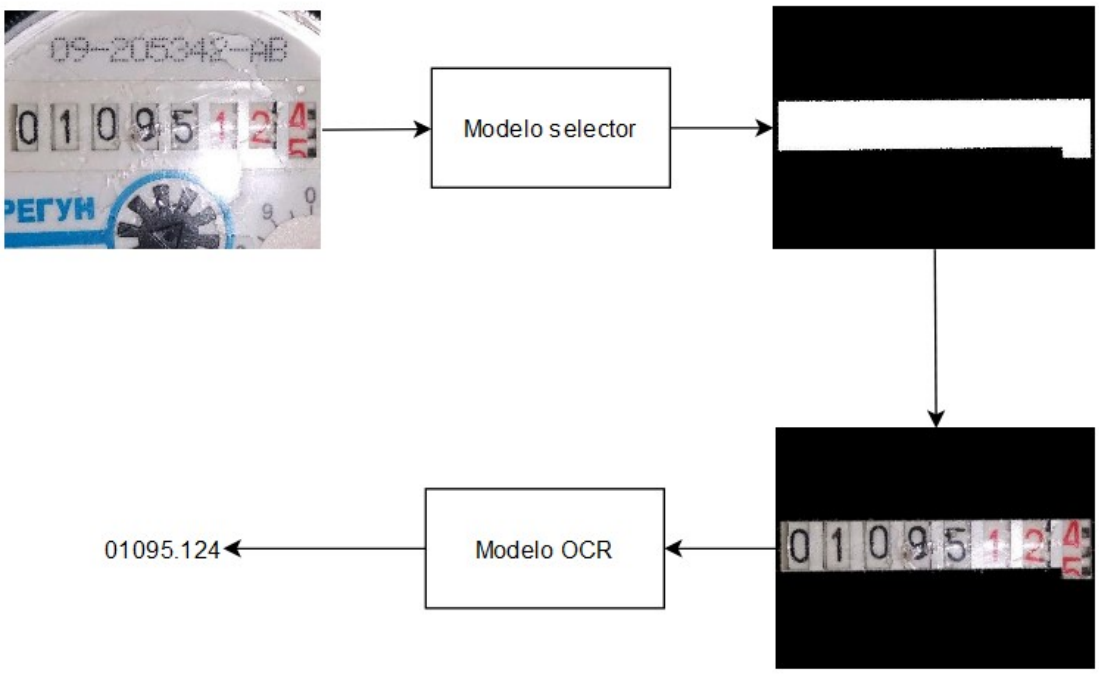

Figura 5: Muestra del funcionamiento del reconocimiento de caracteres

Para explicar con más detalle todas las partes del OCR se va a dividir esta sección en tres apartados, una para explicar el preprocesamiento de los datos, otra para explicar el modelo de selección y una última parte para explicar el OCR que se ha realizado.

4.1.1. Preprocesamiento de los datos

El dataset que se ha utilizado para entrenar los modelos, tanto el de selección como el de OCR, ha sido un conjunto [35] que contenía imágenes de contadores de agua de varios tipos. En concreto este dataset contiene 1244 imágenes de contadores con sus respectivas máscaras de la localización de los datos.

A continuación se muestra una imagen, y su respectiva máscara, del dataset para que se pueda apreciar el tipo de datos con el que se ha estado trabajando.

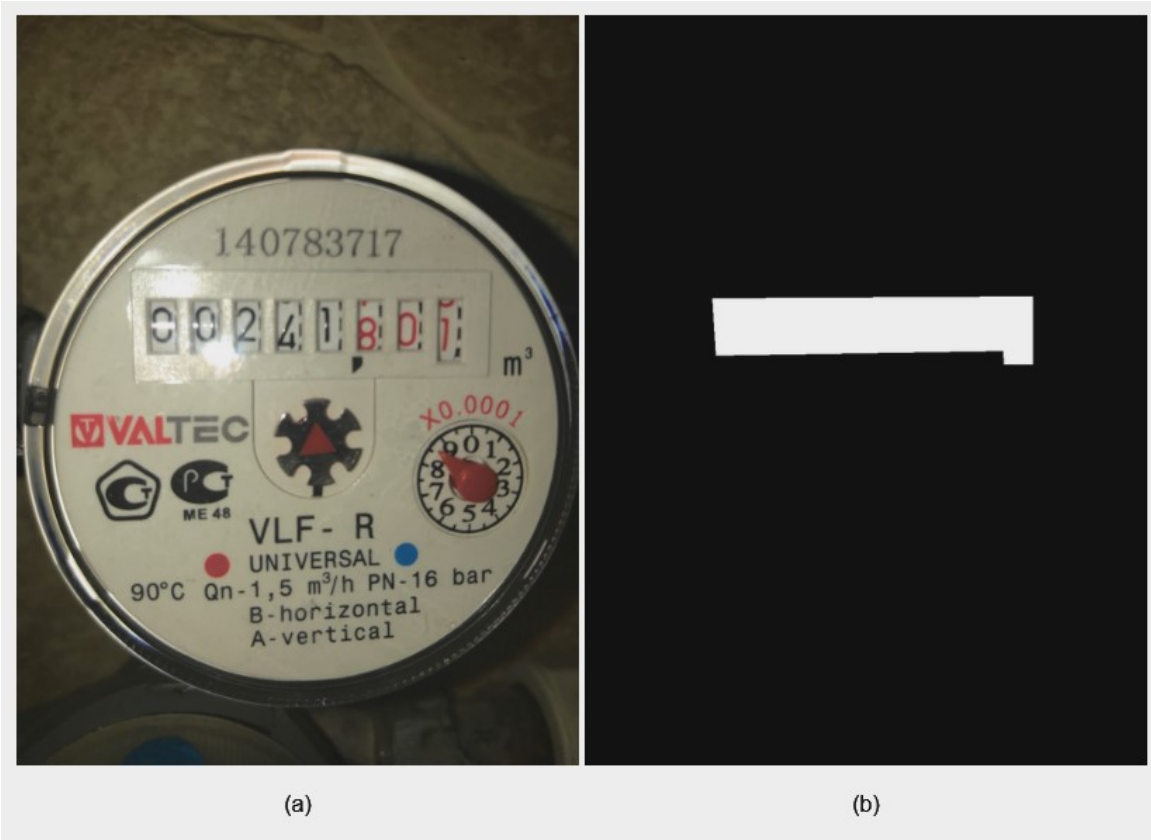

Figura 6: Imagen muestra del dataset: (a) Imagen contador (b) Imagen máscara

Como se puede observar en la Figura 6, las imágenes cuentan con una muy buena calidad. Pese a que en el dataset existen algunas con diferentes dimensiones y calidad entre ellas, por lo general son de 1300x1000 píxeles. Además, como se puede ver en la Figura 6, están tomadas desde distancias muy alejadas.

Como las imágenes del dataset no se parecen a las imágenes que se van a acabar tomando por la cámara, se optó por realizar un preprocesado de estas para que se parecieran lo máximo posible.

Lo primero que se hizo fue comprobar la calidad de imagen de la cámara. La Arducam Mini 2mp Plus puede ofrecerlas en las siguientes resoluciones: 160x120, 176x144, 320x240, 352x288, 640x480, 800x600, 1024x768, 1280x1024 y 1600x1200. Como se necesitaba pasar por LoRaWAN el mínimo número de datos posible se decidió utilizar imágenes de 160x120 píxeles.

Tras la realización de una serie de pruebas para comprobar que tamaño ocupaban este tipo de imágenes al ser comprimidas en JPEG, se pudo observar que dependiendo de la imagen este variaba entre 1500 bytes y 3500 bytes. Que las imágenes tuvieran un

tamaño máximo de 3500 bytes quería decir que para el paso de una de ellas a través de la red se necesitan, como máximo, once paquetes. Esto era un número asequible de paquetes, teniendo en cuenta que el número de imágenes que se envían cada día no va a ser mayor que dos y que, como se ha visto anteriormente, se podrían llegar a enviar hasta cinco imágenes.

Para convertir las imágenes del dataset en posibles imágenes reales, tomadas con la cámara, se realizaron tres pasos. En primer lugar, se recortó la imagen para acercarla al número que se pretendía leer, dejándola en cuatro tercios, que es el formato de la cámara. Tras eso se redimensionó para dejarla en calidad 160x120 píxeles. Por último se realizó la misma conversión para las máscaras de cada imagen, haciendo así que coincidieran con estas.

En la Figura 7 se muestra un ejemplo de la misma foto que se ha mostrado en la Figura 6 pero después de recibir el preprocesamiento.

Figura 7: Imagen muestra del dataset procesada: (a) Imagen contador (b) Imagen máscara

Por último, con el fin de aumentar los datos del dataset, se realizaron modificaciones en las imágenes, guardando tanto la original como la modificada en el dataset final. Las modificaciones que se llevaron a cabo fueron dos.

La primera fue rotar la imagen, donde se observó que, como se estaba trabajando con fotografías a baja resolución, si la rotación era diferente a $180^{\underline{o}}$ la imagen perdía demasiada calidad, haciendo que fuera contraproducente para el entrenamiento. Esta modificación se le hizo también a la máscara, para que coincidiera.

La segunda modificación que se hizo fue añadir ruido a las imágenes, al igual que en el caso anterior si se añadía demasiado ruido la foto quedaba inservible, por lo que se tuvo que ajustar. En este caso, no se añadió ruido a la máscara, dejándola como la original, ya que el propósito de añadirlo en las imágenes es que al utilizar el modelo de selección, aunque la imagen tenga ruido, encuentre de manera correcta la máscara.

Con estas dos modificaciones se quedó un dataset final de 3732 imágenes, el cual se dividió de manera aleatoria en datos de entrenamiento (70 % de las imágenes), validación (25 % de las imágenes) y prueba (5 % de las imágenes). Con este conjunto de imágenes ya se podían entrenar los modelos seleccionados para obtener el valor del consumo del agua a partir de la imagen del contador.

4.1.2. Modelo de selección

Una vez se modificó el dataset para conseguir uno que asemejara las imágenes que se iban a obtener de la cámara, se procedió a la búsqueda y entrenamiento del primer modelo. Este debía recibir una imagen y devolver la posición de los números sobre los que se necesita realizar el OCR posteriormente.

Para este modelo se seleccionaron dos redes neuronales que cumplían con las necesidades dichas. Estas redes eran la red neuronal VGG16 y la red neuronal U-Net. Ambas redes reciben una imagen de poco tamaño y policromática y devuelven las características que se quieren encontrar en ellas, en este caso, los números del contador.

Ambas redes neuronales se caracterizan por sus buenos resultados a la hora de obtener algún tipo de característica de las imágenes, como puede ser seleccionar una zona en concreto de fotos diferentes. Para determinar cuál de las dos se utilizará en el modelo final se entrenaron ambas y se hicieron pruebas de funcionamiento.

Modelo VGG16:

VGG [36] es un modelo de red neuronal convolucional propuesto por K. Simonyan y A. Zisserman, del "Visual Geometry Group" de la universidad de Oxford, que se hizo famosa al ganar el Desafío de Reconocimiento Visual a Gran Escala de ImageNet (ILSVRC), obteniendo una precisión del 92.7 %. El ILSVRC es un desafío que evalúa algoritmos de detección de objetos y clasificación de imágenes a gran escala. Este porcentaje de precisión supuso una grán mejora con respecto a los modelos anteriores.

El modelo VGG16 es el nombre que se le dio al modelo VGG que cuenta con 16

capas, trece capas convolucionales y de max-pooling, 3 capas totalmente conectadas y una capa de softmax.

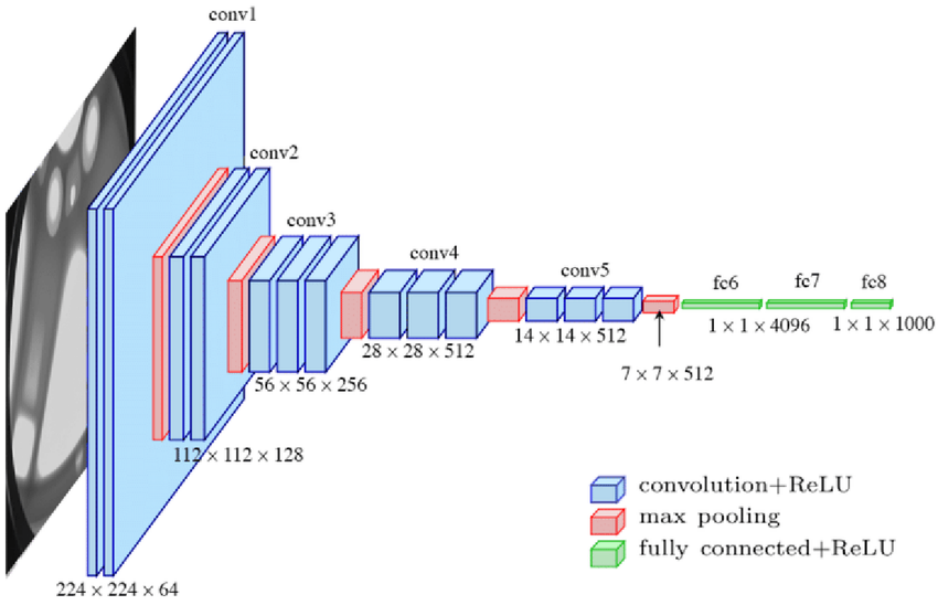

Figura 8: Arquitectura de la red VGG16 [1]

Como se observa en la Figura 8, estas capas están seguidas por una capa de agrupación (max-pooling) de 2x2 que reduce la dimensión espacial de las características extraídas, manteniendo la información relevante.

Posteriormente, se aplican dos capas convolucionales más, ambas con filtros de 3x3 y produciendo 128 mapas de características cada una. Estas capas también están seguidas por una capa de max-pooling de 2x2. Este patrón de dos capas convolucionales seguidas de una capa de max-pooling se repite en las siguientes fases del modelo, pero con un número creciente de filtros.

En la tercera etapa, se utilizan tres capas convolucionales con filtros de 3x3, generando 256 mapas de características cada una, seguidas por una capa de max-pooling. Este incremento gradual en el número de filtros permite al modelo capturar características más complejas y abstractas de las imágenes.

La cuarta etapa del VGG16 consiste en tres capas convolucionales adicionales, cada una con filtros de 3x3 y produciendo 512 mapas de características. Nuevamente, esto se sigue por una capa de max-pooling de 2x2. Finalmente, la quinta y última etapa convolucional incluye tres capas convolucionales más, también con filtros de 3x3 y generando 512 mapas de características, seguidas por una capa de max-pooling.

Después de las capas convolucionales, la arquitectura de VGG16 incorpora capas completamente conectadas. La primera capa completamente conectada tiene 4096 unidades, seguida por otra capa completamente conectada también de 4096 unidades. Estas capas están diseñadas para realizar una integración completa de las características extraídas y para permitir la clasificación de las imágenes. La última capa completamente conectada del modelo tiene 1000 unidades, correspondientes a las 1000 clases de la base de datos ImageNet, sobre la cual el modelo fue entrenado originalmente. Esta capa utiliza una función de activación softmax para producir probabilidades de clasificación para cada clase.

En el caso de este modelo, la salida no es una máscara que refleje las características de la imagen, sino que son los puntos que forman la figura en el que están las características que se han obtenido. En este caso concreto, la salida del modelo son cuatro pares de posiciones, "x" e "y", que representan las esquinas.

Como se puede observar en la Figura 8, el modelo recibe una imagen en formato RGB de 224x224 píxeles, por lo que las imágenes del dataset utilizado se tuvieron que redimensionar a 224x224 píxeles para este entrenamiento.

Para el correcto entrenamiento del modelo se utilizó el error cuadrático medio (en inglés Mean Square Error, MSE) para el análisis de la función de pérdida. El MSE es una medida estadística que evalúa la calidad de un modelo de predicción al calcular el promedio de los cuadrados de los errores, donde el error es la diferencia entre la predicción del modelo y el valor observado.

A continuación se puede ver la evolución de la pérdida tanto durante el entrenamiento como durante la validación.

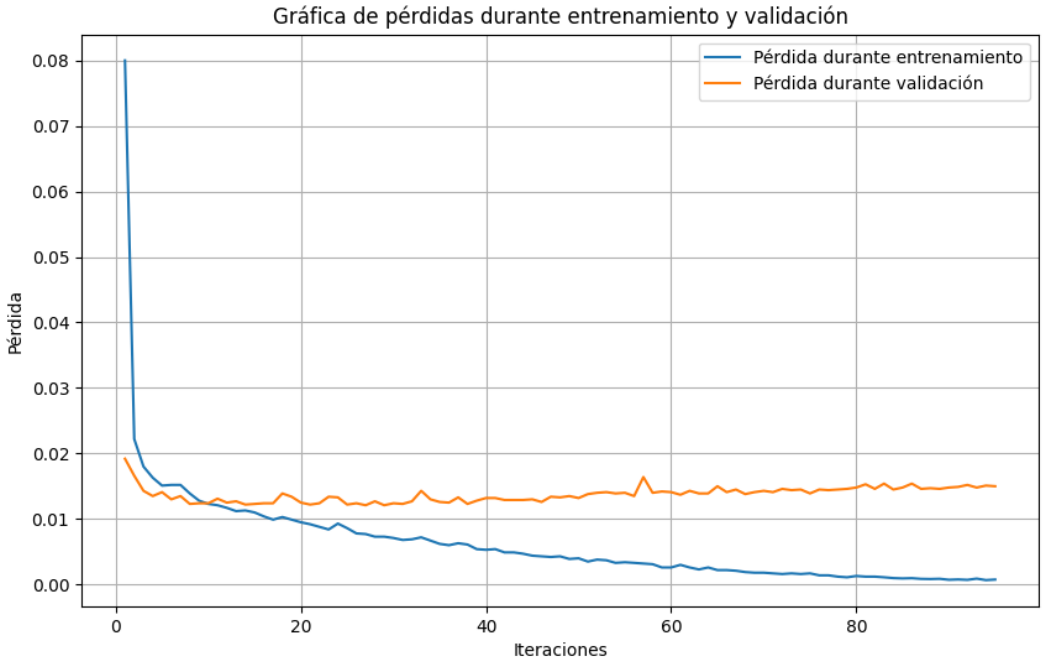

Figura 9: Gráfica de la pérdida del entrenamiento de la red VGG16

Como se puede observar en la Figura 9, el error cuadrático medio se reduce significativamente en el caso del entrenamiento, llegando a valores muy pequeños. Pese a eso, el error en el conjunto de validación solo se reduce al inicio, manteniéndose en valores muy parecidos durante el resto del entrenamiento.

Al realizar las predicciones sobre el modelo de prueba se pudo observar que los resultados no coincidían con los esperados.

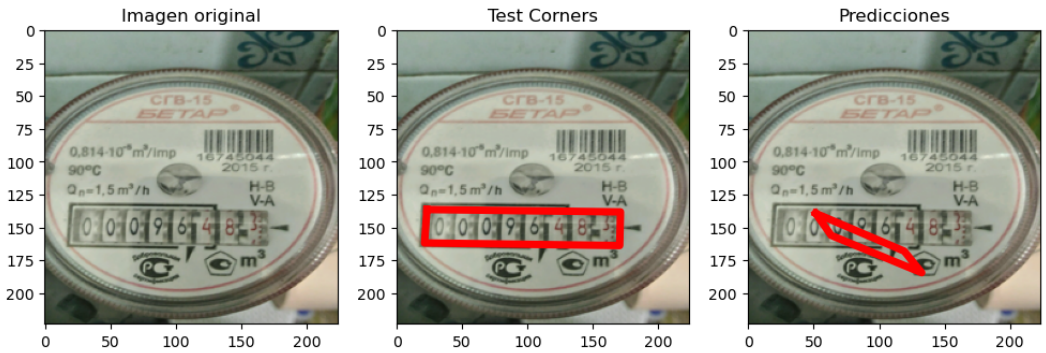

Figura 10: Ejemplo del resultado de la predicción de la red VGG16

Como se puede ver en la Figura 10, el modelo es capaz de predecir la zona sobre la

que están los números que se quieren reconocer, pero no es capaz de reconocerla bien. Únicamente predice la posición aproximada de esta, pero no llega a predecir la posición exacta de todas las esquinas.

Debido a esto se decidió entrenar otra red neuronal que también había dado buenos resultados a la hora de analizar imágenes y obtener características de ellas. La red U-Net.

Modelo U-Net:

El modelo U-Net es una arquitectura de red neuronal convolucional introducida en 2015 por Olaf Ronneberger, Philipp Fischer y Thomas Brox en su artículo "U-Net: Convolutional Networks for Biomedical Image Segmentation" [37]. Esta red fué diseñada principalmente para tareas de segmentación de imágenes.

El U-Net fue pensado inicialmente para segmentación de imágenes biomédicas, facilitando la identificación de estructuras celulares en imágenes de microscopía. No obstante, su aplicación se ha extendido a múltiples ámbitos como la segmentación de imágenes satelitales, la detección de objetos en vídeos e imágenes, y la segmentación de imágenes en la conducción autónoma, entre otros.

En la Figura 11 se puede ver la arquitectura de este modelo. Esta se caracteriza por su estructura simétrica en forma de U, compuesta por un camino de contracción y un camino de expansión. Esta configuración permite la captura de características de manera eficiente y la combinación de información espacial con precisión.

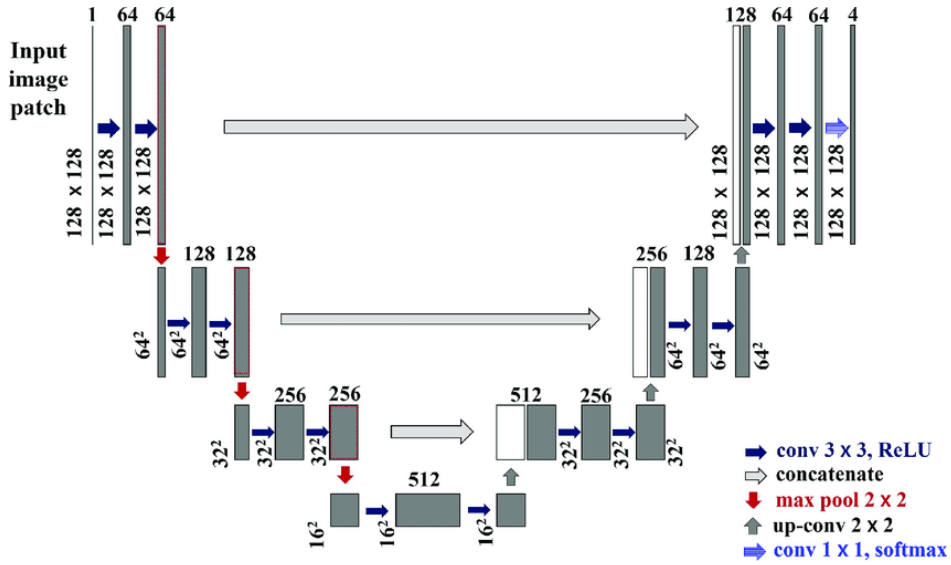

Figura 11: Arquitectura de la red U-Net [2]

El camino de contracción se encarga de extraer características de la imagen de entrada y reducir su resolución para capturar información más abstracta. Está compuesto por varias etapas que incluyen capas convolucionales que aplican múltiples filtros a la imagen para extraer características locales como bordes, texturas y patrones.

Estas convoluciones suelen utilizar filtros de 3x3 píxeles, seguidas de una función de activación no lineal, como ReLU (Rectified Linear Unit). Posteriormente, se aplica una capa de max pooling, que reduce la resolución espacial de las características extraídas a la mitad. Este proceso de reducción permite capturar información a diferentes niveles de abstracción y disminuye la carga computacional. El max pooling selecciona el valor máximo en una ventana de 2x2 píxeles, resumiendo las características más prominentes.

Con cada etapa de pooling, el número de filtros se duplica, permitiendo a la red aprender un conjunto de características más complejo y representativo. Este proceso de convolución, activación y pooling se repite varias veces, profundizando la red y reduciendo la resolución de las características hasta alcanzar el cuello de botella de la red.

El cuello de botella representa la transición entre el camino de contracción y el camino de expansión. En este punto, la red tiene la menor resolución espacial pero la mayor cantidad de canales de características, lo que permite representar la información de manera muy abstracta y compacta. Las operaciones en el cuello de botella son similares a las del encoder, con capas convolucionales y activaciones, pero sin reducción adicional de resolución.

El camino de expansión se encarga de reconstruir la imagen segmentada a partir de las características extraídas por el encoder. Este camino es simétrico al de contracción y consta de varias etapas.

Cada etapa del decoder comienza con una capa de upsampling que incrementa la resolución espacial de las características. Esto se puede hacer mediante interpolación (upsampling) o mediante convoluciones transpuestas (deconvolución), como es el caso de la red que se ha utilizado. Estas operaciones recuperan la resolución espacial perdida durante el pooling.

Una característica clave del U-Net es el uso de conexiones de salto que concatenan las características del encoder correspondiente con las características del decoder en cada etapa. Estas conexiones ayudan a preservar la información espacial detallada que se perdió durante el pooling, mejorando la precisión de la segmentación.

Después de la concatenación, se aplican capas convolucionales adicionales para refinar las características combinadas y reducir la dimensionalidad, integrando la información de alta y baja resolución de manera efectiva. Este proceso de upsampling, concatenación y convoluciones se repite varias veces, aumentando la resolución de las características gradualmente hasta que se alcanza la resolución original de la imagen de entrada.

La última capa del U-Net es una capa convolucional que genera el mapa de segmentación final. Esta capa tiene tantos canales de salida como clases haya en la tarea de segmentación. Se utiliza una función de activación sigmoidal para segmentación binaria, para producir una probabilidad para cada clase en cada píxel.

Para el trabajo que se tenía que realizar, obtener las características de una foto de 160x120 píxeles, se decidió que la mejor opción era utilizar una red U-Net con una imagen de entrada de 128x128 píxeles. Esto provocaría una disminución de la calidad de la imagen de entrada, pero también una disminución de las capas necesarias, y por ende, una disminución del tamaño de la red.

En el caso del entrenamiento de esta red se utilizó la función de pérdida "Binary Cross-Entropy". Esta función mide la diferencia entre las funciones de probabilidad predichas por la red neuronal y las reales. Se utiliza en problemas de clasificación binaria, como la red U-Net, y su objetivo es clasificar cada píxel de la imagen en una de dos clases: pixel de interés o píxel de fondo.

Para el entrenamiento de esta red se hizo uso de las máscaras mencionadas anteriormente, las cuales funcionaban de manera idónea para representar los datos de salida de la red ya que utilizaba los píxeles a cero para los datos de fondo y los píxeles a uno para los datos que eran relevantes.

A continuación se puede ver la evolución de la pérdida tanto durante el entrenamiento como durante la validación:

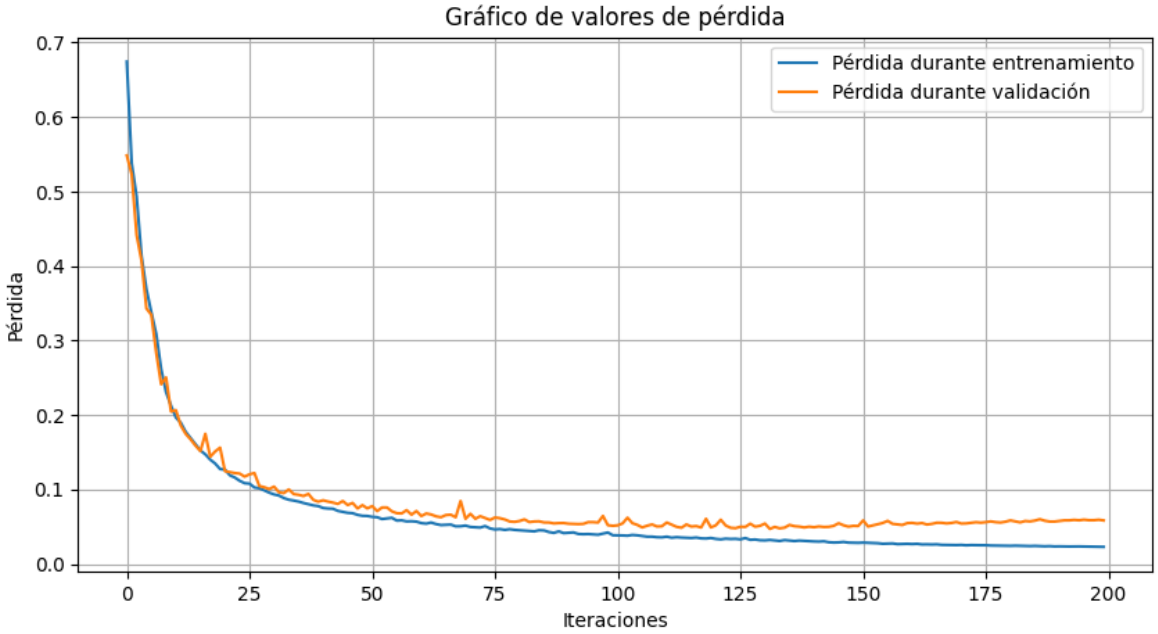

Figura 12: Gráfica de la pérdida del entrenamiento de la red U-NET

Como se puede observar en la Figura 12, y a diferencia del modelo VGG16, tanto la pérdida del entrenamiento como la pérdida de la validación disminuyen hasta ser valores muy cercanos a cero.

Al realizar las pruebas sobre el conjunto de imágenes de prueba se pudo observar que la red devolvía unos resultados que se asemejan en gran medida a las máscaras reales. Pese a no ser perfectas, las máscaras que devolvía la red eran completamente funcionales para el propósito que se les pretendía dar. El porcentaje de acierto que se obtuvo fue de un 90,79 %.

En la Figura 13 se muestra un ejemplo de resultado de la predicción:

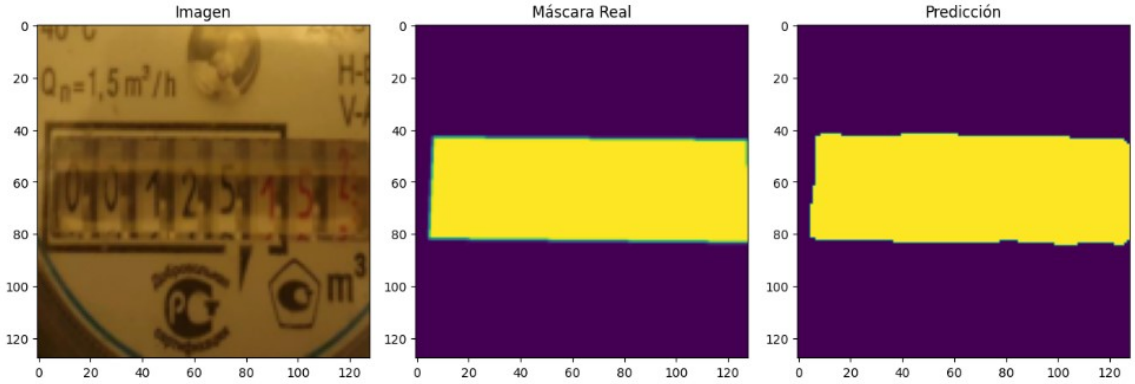

Figura 13: Ejemplo del resultado de la predicción de la red U-NET

Con esta red neuronal completamente funcional solo quedaba un paso para poder empezar a realizar el OCR, redimensionar la máscara y aplicarla a la imagen original. Para ello se creó un script que obtenía la máscara y la redimensiona de 128x128 a 160x120 utilizando interpolación bilineal, y una vez se tienen la foto original y la máscara en las mismas dimensiones se mantiene el color de los píxeles de la imagen original que coincidan con unos en la máscara y se eliminan los que coincidan con ceros.

Con este procedimiento ya se tiene todo lo necesario para realizar el OCR sobre la imagen.

4.1.3. Modelo de reconocimiento de carácteres

Una vez se ha procesado la imagen y se ha conseguido obtener únicamente una imagen de los dígitos del contador de agua, se necesita obtener los números de dicha imagen. Para ello se realizó una búsqueda de diferentes métodos de OCR que pudieran servir para este propósito.

En este apartado se va a explorar el funcionamiento y resultados de los diferentes modelos seleccionados, con el fin de seleccionar uno de ellos para su uso en el prototipo final. Se va a dividir esta sección en un apartado de explicación de los modelos más relevantes y otro de comparación de los resultados obtenidos.

Modelos utilizados

Los modelos que han sido probados para el reconocimiento de los carácteres son de tres tipos diferentes. Los primeros son librerías de python que contienen modelos dedicados al OCR, estos modelos no han sido entrenados con las imágenes obtenidas por el modelo de selección visto anteriormente, sino que han sido entrenados por los creadores de las librerías y son utilizados directamente. En concreto se han realizado pruebas sobre las librerías Aspore [38], EasyOCR y PyTesseract [39]. El funcionamiento interno de estos modelos se especifica en el Anexo III.

El segundo tipo de modelo que se ha probado son APIs de webs dedicadas al reconocimiento de caracteres. Al igual que el modelo anterior, estas no han sido entrenadas con mis imágenes. Las aplicaciones que se han utilizado han sido OCRbest y 2OCR.

Por último se ha utilizado un tipo de modelo que ha sido directamente entrenado con las imágenes del dataset, trás ser convertidas por el modelo anterior. En este caso se ha entrenado una red neuronal Faster R-CNN realizando un ajuste fino.

Faster R-CNN:

Las Faster R-CNN son [40] una evolución avanzada dentro de la serie de Redes Neuronales Convolucionales diseñadas para abordar la tarea de detección de objetos en imágenes. A diferencia de sus predecesoras, las Faster R-CNN introducen una mejora significativa en la eficiencia y precisión mediante la incorporación de una Red de Propuestas de Regiones (RPN, por sus siglas en inglés).

En términos de funcionamiento, las Faster R-CNN operan en dos fases principales. Primero, una RPN se encarga de generar propuestas de regiones. Esta red toma como entrada una imagen y produce un conjunto de regiones candidatas que probablemente contengan objetos. La RPN está diseñada para ser extremadamente rápida, ya que reutiliza las características extraídas por las primeras capas de la red convolucional.

La segunda fase consiste en refinar estas propuestas utilizando una red de detección de objetos. Aquí, las propuestas generadas por la RPN son evaluadas y clasificadas en diferentes categorías de objetos mediante una red neuronal que también ajusta con mayor precisión las cajas delimitadoras alrededor de los objetos detectados. Esta red realiza una clasificación fina y una regresión de las coordenadas de las cajas delimitadoras, mejorando tanto la exactitud en la identificación de los objetos como en

la delimitación espacial de los mismos.

En el caso que nos ocupa, se ha utilizado una red neuronal de Detectron2 [41], una plataforma de código abierto creada por Facebook AI Research que permite la implementación de algoritmos de detección de objetos o segmentación de imágenes. Esta plataforma ofrece múltiples opciones de redes neuronales entrenadas previamente para la detección de objetos en imágenes o la segmentación de estas.

En este caso se ha utilizado una de estas redes entrenadas y se ha realizado un ajuste fino (en inglés, fine tuning) con los datos del dataset tras ser procesados por la red neuronal de selección. El ajuste fino es [42] un término que se refiere a utilizar una red neuronal que ya ha sido entrenada con un gran conjunto de datos, como pueden ser imágenes con palabras y textos, y entrenarla adicionalmente con un conjunto de datos más específicos, como pueden ser imágenes de contadores.

El modelo que se ha utilizado es el Faster R-CNN X101-FPN. Este modelo añade a la arquitectura Faster R-CNN una red piramidal de características (en inglés Feature Pyramid Network, FPN) y una red residual ResNet. Una FPN es una arquitectura de red neuronal muy similar a el modelo U-Net, visto anteriormente, pero con la diferencia de que en este caso se realiza una predicción por cada capa del camino de expansión.

El generador de propuestas de regiones es responsable de identificar posibles regiones de interés en la imagen. Este componente está compuesto por una cabeza RPN estándar que incluye una capa convolucional para el mapeo de características y capas adicionales para la predicción de las regiones.

Las propuestas de regiones generadas por el RPN son procesadas por los cabezales de regiones de interés(en inglés Regions of Interest, ROI) estándar, que realizan una alineación precisa de las regiones mediante el uso de un conjunto de niveladores ROI. Estos niveladores ajustan las regiones propuestas a una resolución estándar, facilitando la posterior clasificación y regresión de las cajas delimitadoras por la red.

La predicción de las categorías de los objetos de las cajas delimitadoras es llevada a cabo por las capas de salida. Estas capas incluyen unidades lineales y funciones de activación ReLU para asegurar una clasificación precisa y la regresión de las coordenadas de las cajas delimitadoras.

Estas últimas capas se han ajustado de manera que coincidan con las categorías de los números del uno al nueve. De esta manera, la red se puede entrenar para

detectar números y establecer sus ubicaciones en la imagen, permitiendo así reconstruir el número completo del contador de agua.

Con el objetivo de obtener una red neuronal que acertara lo máximo posible se ha entrenado este modelo con las imágenes del dataset que se ha utilizado durante el desarrollo de este proyecto.

Para ello se ha procesado el dataset con la red de selección, obteniendo un dataset de imágenes en las que se ven únicamente los números del contador que representan el valor del consumo del agua. Este dataset contiene 3732 imágenes de diferentes tamaños, por lo que se decidió establecer un tamaño para todas y rellenar las imágenes con fondo negro para que se ajustaran a él, el tamaño se estableció en 160x160 píxeles.

Además se decidió realizar un último procesado de las imágenes antes de entrenar la red, se optó por reajustar las imágenes que tenían los números girados o torcidos para que estuvieran todos rectos. De esta manera se ayudaba a la red ya que solo debía de reconocer los números en una posición.

En la Figura 14 se muestra una imagen del dataset resultante:

Figura 14: Ejemplo de imágen procesada

Tras el entrenamiento, salieron los siguientes resultados:

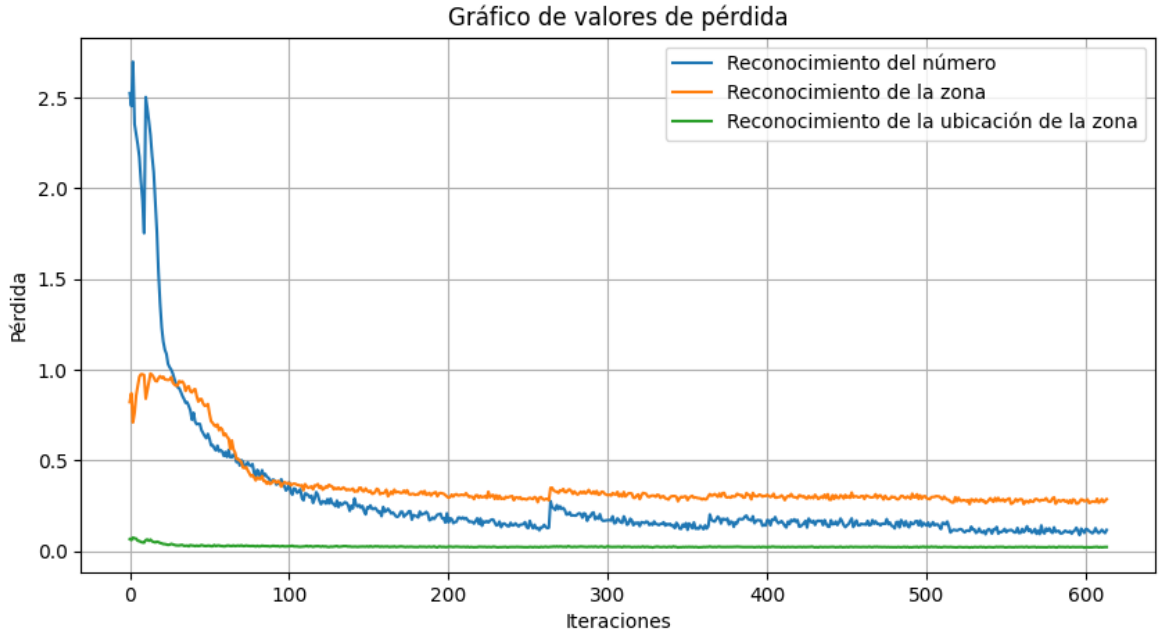

Figura 15: Gráfica de la pérdida del entrenamiento de la red Faster R-CNN

Como se puede observar en la Figura 15, existen tres datos relevantes que varían durante el entrenamiento. Esto se debe al funcionamiento de la red, ya que esta primero reconoce las zonas en las que puede haber números y luego reconoce los números de cada una de esas zonas.

El valor de "Reconocimiento de la zona" indica la pérdida que se ha producido durante el entrenamiento a la hora de reconocer las zonas que existen en cada imagen. El valor de "Reconocimiento de la ubicación de la zona" indica la pérdida que ha habido en el momento de ubicar las zonas predichas anteriormente. Y el valor de "Reconocimiento de número" se refiere a la pérdida a la hora de seleccionar el número que hay en cada zona.

Comparación de resultados

Tras mostrar los modelos con los que se iban a realizar las pruebas para determinar el algoritmo que se utilizará en el modelo final, se va a pasar a comparar los resultados obtenidos por ellos.

Para ello se va a explicar las pruebas de funcionamiento que se han llevado a cabo para determinar el modelo que mejor funciona.

La primera prueba que se ha realizado ha sido sobre el dataset utilizado durante el proyecto. Para llevarla a cabo se han seleccionado varias imágenes de manera aleatoria y se han redimensionado cada una de ellas para obtener las imágenes de las cuatro resoluciones más pequeñas que puede generar la cámara (160x120, 176x144, 320x240 y 352x288). Utilizar estas cuatro resoluciones es un método que se va a llevar a cabo en el resto de las pruebas ya que permite comprobar el tamaño mínimo que se puede utilizar para la captura de la imagen.

Tras la realización de las pruebas los resultados fueron los siguientes:

Método	**160x120**	**176x144**	**320x240**	**352x288**
Aspore	0 %	0 %	0 %	0 %
PyTesseract	0 %	0 %	0 %	0 %
EasyOCR	37.53 %	39.12 %	42.77 %	41.75 %
OCRbest	65.15 %	64.25 %	70.09 %	69.12 %
2OCR	60.12 %	63.42 %	66.30 %	68.78 %
Faster R-CNN	83.17 %	82.64 %	83.66 %	85.13 %

Tabla 1: Resultados de la primera prueba

Como se puede observar en la Tabla 1, el reconocimiento correcto del valor de los contadores de agua no es sencillo. Los modelos de Aspire y PyTesseract no han sido capaces de reconocer ningún número a partir de las imágenes proporcionadas.

En el caso de EasyOCR, se realiza de manera correcta el reconocimiento de los números en algunas imágenes, pero en la mayoría de los casos se realiza una lectura incorrecta de los dígitos. Ambos modelos basados en una APIs de webs dedicadas al reconocimiento de caracteres han dado unos resultados positivos, pero se ve que son muy afectados por la calidad de la imagen, ya que contra mayor es esta, menor es el porcentaje de acierto.

Por último, la red Faster R-CNN ha dado unos resultados de acierto de más del 80 %. Se consideraría el mejor modelo de los que se han probado si no fuera porque esta red ha sido entrenada con las mismas imágenes con las que se ha realizado la prueba.

Es por ello que se ha optado por realizar una segunda prueba, en la que en vez de utilizar imágenes del dataset se van a utilizar unas tomadas directamente con la cámara. Esta prueba tiene como objetivo utilizar las mismas imágenes que se van a usar en el proyecto final.

Para la realización de esta prueba se decidió no utilizar ninguna imagen del dataset

que se había creado, sino utilizar imágenes de contadores realizadas directamente con la cámara, para conseguir información real. Al igual que en el caso anterior, para cada contador se hicieron fotos en cuatro resoluciones diferentes.

La Figura 16 de las imágenes que se tomaron para esta prueba es el siguiente:

Figura 16: Ejemplo de imagen tomada con la cámara

Los resultados de esta prueba fueron los siguientes:

Método	**160x120**	**176x144**	**320x240**	**352x288**
Aspore	0 %	0 %	0 %	0 %
PyTesseract	0 %	0 %	0 %	0 %
EasyOCR	33.82 %	35.23 %	35.98 %	36.18 %
OCRbest	57.34 %	59.87 %	58.76 %	61.02 %
2OCR	56.43 %	60.25 %	57.98 %	59.65 %
Faster R-CNN	87.54 %	85.45 %	85.63 %	86.33 %

Tabla 2: Resultados de la segunda prueba

Como se puede observar en la Tabla 2, los valores obtenidos con respecto a la prueba anterior no varían en gran manera, bajando el porcentaje de acierto en todos ellos con excepción del modelo Faster R-CNN, que no sólo no ha disminuido, sino que ha aumentado el porcentaje de acierto.

Con esta prueba ya se puede observar que el modelo que mejor resultados está dando es el que se ha entrenado directamente con imágenes de contadores.

Como en este último caso se han tomado las imágenes con la mejor iluminación posible, se ha realizado una última prueba para ver cómo cambia el porcentaje de acierto en función de la iluminación que existe a la hora de realizar la imagen. Esta última prueba se realiza ya que, en caso de necesitar una luz muy potente para el correcto funcionamiento de la red Faster R-CNN, esto implica un gasto mayor, por

lo que se tendrán que buscar nuevas soluciones, ya sea utilizando otro de los modelos vistos en caso de que funcionen bien con baja iluminación o buscando nuevas soluciones.

Para ello se obtienen imágenes de varios contadores con tres tipos de iluminación, una intensa desde un punto muy cercano al contador, otra con iluminación más leve y una última con una iluminación muy tenue. Todas las imágenes con las que se realizó esta prueba estaban en una resolución de 160x120 píxeles.

En este caso se va a comprobar únicamente la tasa de acierto de los modelos que más aciertos han dado en las pruebas anteriores, por lo que se realizó sobre OCRbest, 2OCR y la red Faster R-CNN.

Una vez realizada se obtuvieron los siguientes resultados:

Método	**Alta iluminación**	**Iluminación neutra**	**Baja iluminación**
OCRbest	56.93 %	57.12 %	43.82 %
2OCR	56.88 %	55.49 %	46.16 %
Faster R-CNN	86.45 %	86.78 %	57.38 %

Tabla 3: Resultados de la tercera prueba

Viendo los resultados obtenidos en la Tabla 3, existe una gran bajada de los aciertos al utilizar imágenes con poca iluminación, esto se debe a que hay menos contraste entre los números en la imagen y el fondo. Pese a eso no se ve ninguna diferencia entre el uso de una iluminación muy intensa y el uso de una iluminación más leve.

Tras la realización de las pruebas que han permitido observar el funcionamiento de los diferentes modelos vistos a la hora de reconocer los carácteres de los contadores, se puede deducir que el mejor modelo y el que se va a utilizar para el proyecto final es el modelo Faster R-CNN. Esto se debe a su grán diferencia de acierto con respecto a los demás modelos y a que no se ve afectado por una falta de luminosidad.

A parte de obtener el modelo que se va a utilizar finalmente, estas pruebas han servido para comprobar que se van a poder realizar la toma de las imágenes en la menor calidad posible, lo que permite un menor paso de bytes por la red.

4.2. Pruebas reales

Una vez se tiene un sistema funcional que permite registrar el consumo del agua a partir de una imagen enviada por un dispositivo colocado en el contador, se han realizado unas pruebas para comprobar su correcto funcionamiento en un entorno real. Se han realizado unas pruebas del consumo de batería, que se pueden ver en el Anexo IV y unas pruebas de funcionamiento que se verán a lo largo de este apartado.

Para ello se ha creado un prototipo en 3D el cual es utilizado para ayudar al módulo de la cámara y al microprocesador a realizar la toma de la imagen y enviarla a través de la red. En la figura 17 se puede ver el diseño 3D del prototipo y el prototipo impreso se puede ver en la Figura 18. Los detalles de este diseño y la creación de este prototipo se pueden ver en el Anexo V

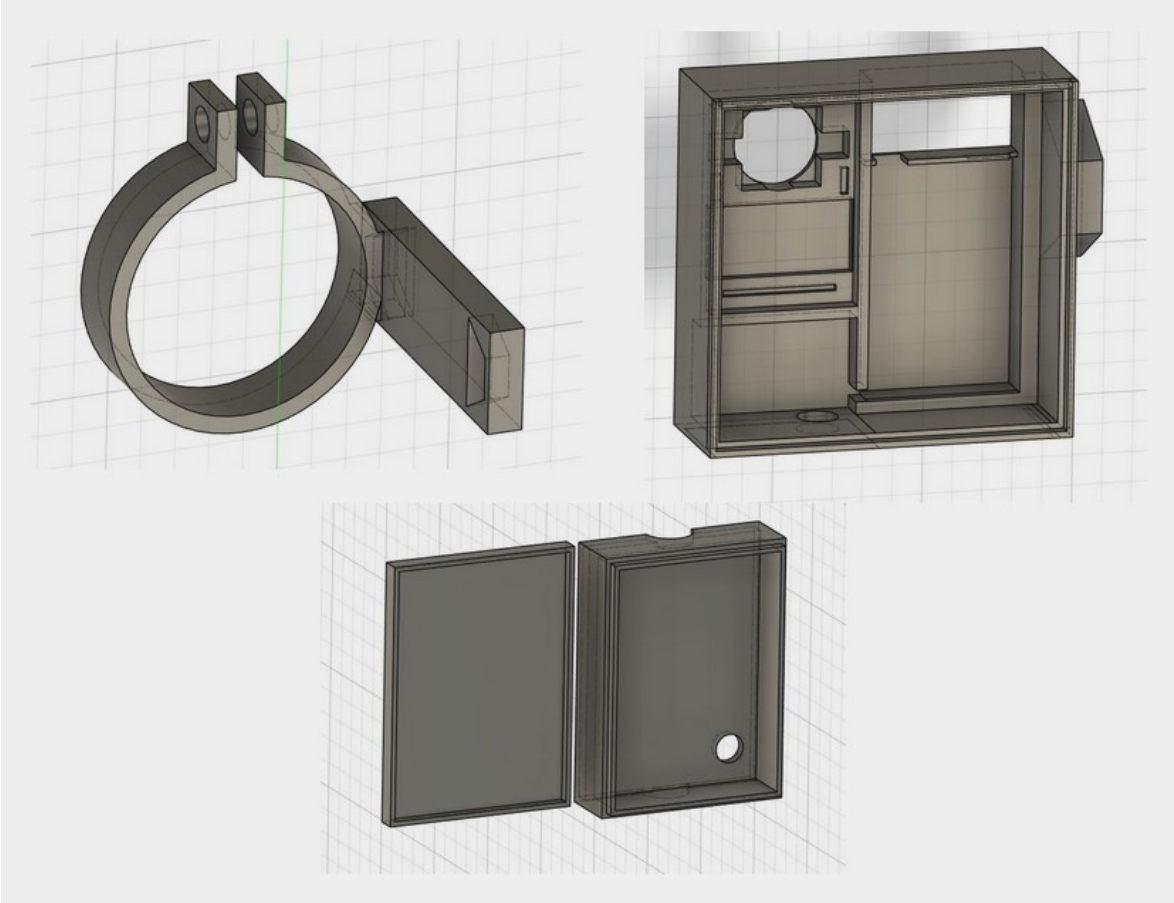

Figura 17: Diseño 3D de las piezas del prototipo

Figura 18: Prototipo impreso en 3D

Una vez se tiene un prototipo funcional, con un sistema que permite registrar el consumo del agua a partir de un módulo que se coloca en el contador, se han realizado unas pruebas para comprobar su correcto funcionamiento en un entorno real.

Los contadores sobre los que se han realizado las pruebas no contaban con números decimales, lo que quiere decir que para observar una variación se tenían que consumir 1000 litros de agua. Es por ello que se han recogido datos durante un mes en uno de los contadores (Contador B), mientras que en el otro únicamente se han recogido durante una semana (Contador A).

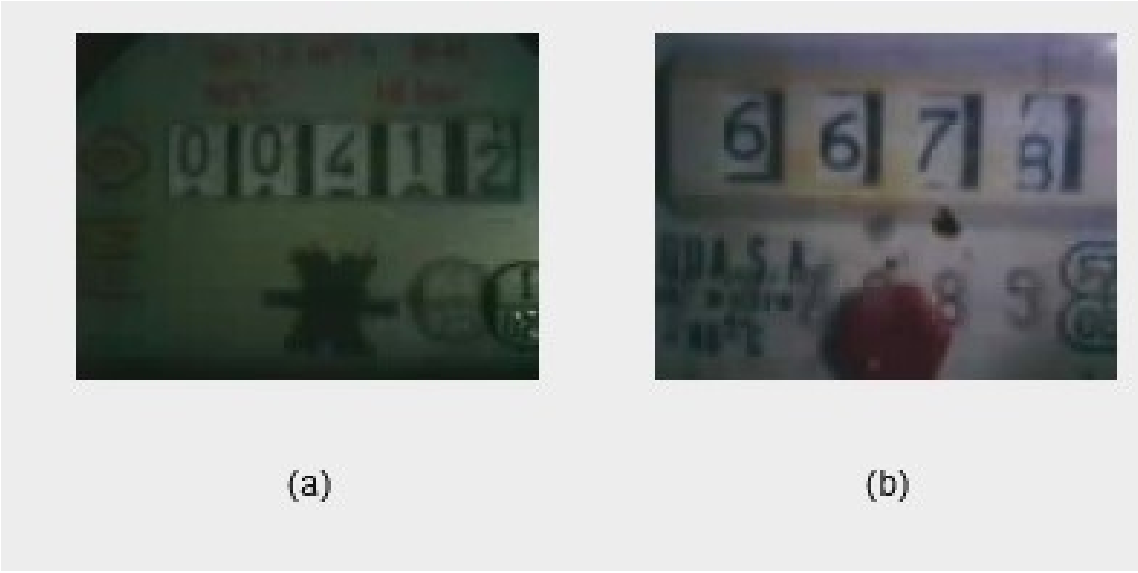

Figura 19: Ejemplo de los contadores: (a) Contador A (b) Contador B

En la Figura 19 se observan dos imágenes, una tomada del Contador A y otra del Contador B.

Una vez mostrados los contadores sobre los que se van a hacer pruebas se van a mostrar las tablas con los resultados obtenidos.

Días	1	2	3	4	5	6	7
Predicción	100412	412	412	412	41	413	413

Tabla 4: Resultado de la predicción del Contador A con respecto al tiempo

Como se puede observar en la Tabla 4, en el caso del Contador A da un pequeño error en la primera imagen que toma, prediciendo un valor "1" al inicio de los dígitos que provoca un gran cambio en la predicción real. Además, tiene otro error en el quinto valor, en el que no detecta bien el último número, esto se debe a que el modelo de OCR no funciona tan bien en las imágenes en las que el dígito está a mitad camino, ya que no reconoce ni el número anterior ni el siguiente. Pese a eso, en el resto de imágenes da el valor exacto que marca el contador.

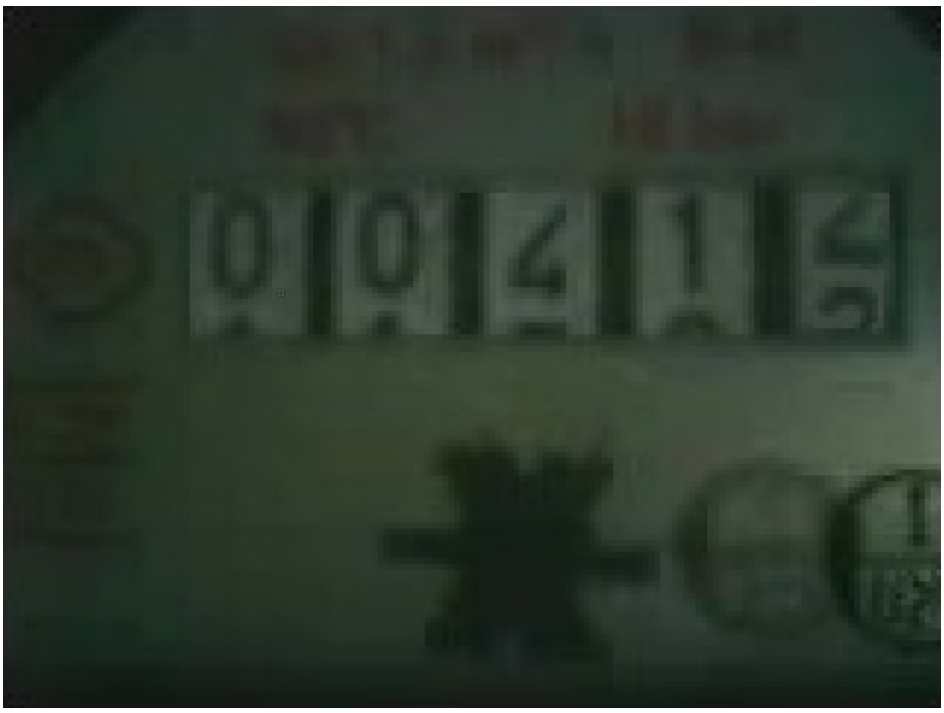

Figura 20: Imagen capturada del contador de agua en la que el número no está completo.

La imagen del contador mostrada en la Figura 20 muestra un ejemplo en el que el OCR no funciona del todo bien debido a que en el último dígito únicamente se ve la mitad del número anterior y la mitad del número siguiente.

Para las pruebas en el Contador B se obtuvieron los siguientes resultados:

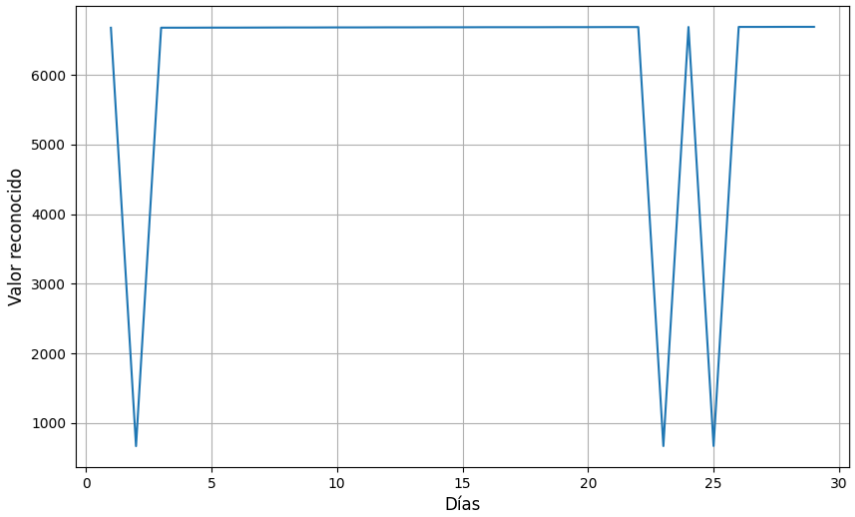

Figura 21: Gráfica que muestra el avance del valor del Contador B con respecto al tiempo

Como se puede observar en la gráfica 21, al ser una cantidad mayor de números reconocidos, falla más a menudo. Estos fallos que se obtienen son, al igual que en el caso anterior, por culpa de que intenta reconocer un valor de una imagen en la que se ve la mitad de dos números, pero ninguno completo.

Para observar la gráfica que se hubiera obtenido sin esos errores se ha realizado un procesado manual en el que se han eliminado los valores erróneos.

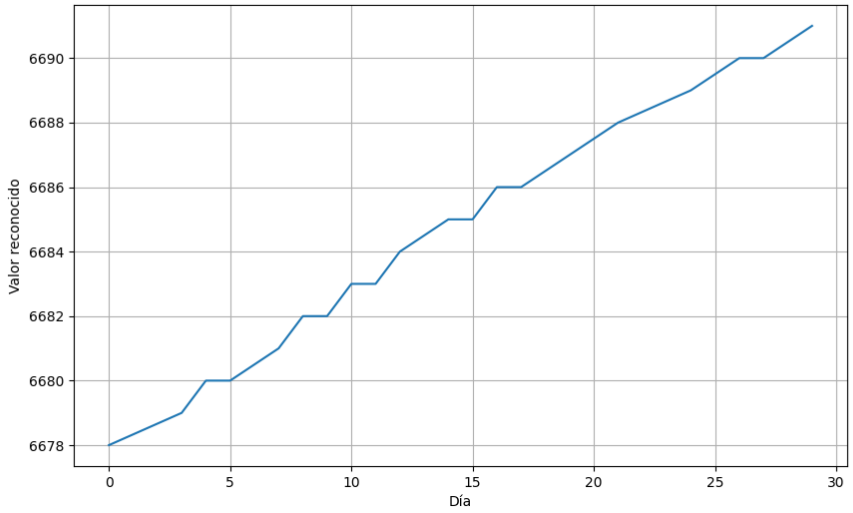

Figura 22: Gráfica que muestra el avance del valor del Contador B con respecto al tiempo, eliminando los valores erróneos.

Como se puede observar en la Figura 22, el consumo de agua que se ha registrado en este contador es de unos 15 metros cúbicos de agua en todo el mes, lo que supondría unos 500 litros de agua al día.

Tras la realización de estas pruebas se puede decir que el sistema funciona correctamente. Pese a que tiene algún fallo, son fallos que tienen una fácil solución.

Una de las propuestas de mejora que se podría realizar a la hora de implementar este sistema en un entorno real sería la de que el técnico que instale el dispositivo en el contador anote y envíe al servidor de alguna manera el valor del consumo del agua que hay en ese momento. De esta manera se podrían detectar posibles fallos del modelo y corregirlos. De manera que si el resultado del reconocimiento de caracteres devuelve un resultado que es imposible se podría realizar un procesado del resultado para eliminar los datos erróneos y obtener así el dato real.

Otra propuesta de mejora relacionada con el error explicado sería cambiar la lógica del sistema para que, en caso de que de un resultado que no tuviera sentido, comparándolo con los datos recibidos hasta el momento, se envíe un mensaje desde el servidor hasta el dispositivo para que vuelva a realizar la captura de la imagen y la vuelva a enviar.

Capítulo 5

Conclusiones

En este trabajo se ha desarrollado un nuevo sistema para facilitar la monitorización del consumo del agua. Este sistema utilizaba un módulo con una cámara y un microcontrolador para realizar la toma de una imagen, la cual se enviaba a través de una red LoRaWAN para ser procesada en el servidor. En este servidor se utilizan diferentes técnicas de detección de objetos y OCR para obtener el valor del consumo del agua, y poder así mostrárselo al usuario.

Para la realización de este sistema se ha buscado desarrollar un prototipo económico, de fácil implementación, y que se adecúe a los posibles problemas que ofrecen los entornos en los que están instalados los contadores.

Tras realizar el proyecto se puede determinar que se han cumplido los objetivos especificados, pese a no haber sido posible realizar el OCR directamente en el dispositivo por la falta de almacenamiento.

Por una parte, se ha conseguido solventar dicho problema gracias a la implementación de un sistema que permite obtener la imagen de un contador de agua y la envía a través de una red LoRaWAN hacia el servidor. Además, se han conseguido enviar imágenes de manera consistente y sin provocar una gran pérdida de datos.

Por otra parte, se ha creado un sistema mediante el cual se ha podido obtener el valor del contador de agua mediante la imagen proporcionada por el prototipo. Consiguiendo obtener dicho valor de imágenes lo más pequeñas posibles, en este caso de 160x120 píxeles.

El prototipo que se ha creado para este proyecto cuenta con soluciones a los principales problemas que existen en los entornos en los que están los contadores de

agua, como pueden ser falta de espacio, falta de iluminación o falta de conexión a una corriente eléctrica.

Por último, se han conseguido cumplir los objetivos con elementos de bajo coste, lo que era fundamental para la realización de este proyecto.

Gracias al uso de AIoT se puede confirmar que se pueden utilizar diferentes tecnologías para la monitorización del agua, como un sistema que utiliza inteligencia artificial para procesar una imagen tomada. Además confirma que esta monitorización se puede realizar con un coste económico más bajo que las soluciones que existen actualmente en el mercado.

Capítulo 6

Bibliografía

[1] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015.

[2] Bumshik Lee, Nagaraj Yamanakkanavar, and Jae Choi. Automatic segmentation of brain mri using a novel patch-wise u-net deep architecture. *PLOS ONE*, 15:e0236493, 08 2020.

[3] Mehmet Ali Ertürk, Muhammed Ali Aydın, Muhammet Talha Büyükakkaşlar, and Hayrettin Evirgen. A survey on lorawan architecture, protocol and technologies. *Future internet*, 11(10):216, 2019.

[4] Somayya Madakam, Ramya Ramaswamy, and Siddharth Tripathi. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3(5):164–173, 2015.

[5] Juan Manuel Cueva Lovelle, Jose Ignacio Rodriguez Molano, and Carlos Enrique Montenegro Marin. IntroducciÓn al internet de las cosas. *Redes de Ingeniería*, 6, sep. 2015.

[6] Subhas Chandra Mukhopadhyay, Sumarga Kumar Sah Tyagi, Nagender Kumar Suryadevara, Vincenzo Piuri, Fabio Scotti, and Sherali Zeadally. Artificial intelligence-based sensors for next generation iot applications: A review. *IEEE Sensors Journal*, 21(22):24920–24932, 2021.

[7] Sánchez Fernández, Carlos Javier, and V Sandonís Consuegra. Reconocimiento óptico de caracteres (ocr). *Univ. Carlo*, 3(7):2008, 2008.

[8] Jorge Díaz-Ramírez. Aprendizaje automático y aprendizaje profundo. *Ingeniare. Revista chilena de ingeniería*, 29(2):180–181, 2021.

[9] Sensus contadores de agua iperl. https://www.xylem.com/es-es/products--services/metrology-equipment-for-utilities/meters/iperl-international-water-meter/. Accedido el 11 de septiembre de 2024.

[10] Contador de pulsos sigfox. http://productos-iot.com/monitorizacion-del-contador-fiscal-de-agua/. Accedido el 11 de septiembre de 2024.

[11] A Amir, R Fauzi, and Yusnaini Arifin. Smart water meter for automatic meter reading. In *IOP Conference Series: Materials Science and Engineering*, volume 1212, page 012042. IOP Publishing, 2022.

[12] Young-Woo Lee, Seongbae Eun, and Seung-Hyueb Oh. Wireless digital water meter with low power consumption for automatic meter reading. In *2008 International Conference on Convergence and Hybrid Information Technology*, pages 639–645, 2008.

[13] Luthfi Muhammad Ramadhan, Rina Pudji Astuti, and Hanif Fakhrurroja. Compact smart water meter development for smart city. In *2023 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, pages 677–682, 2023.

[14] Heltec wifi lora 32 v3. https://heltec.org/project/wifi-lora-32-v3/. Accedido el 11 de septiembre de 2024.

[15] Arducam mini 2mp plus. https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/. Accedido el 11 de septiembre de 2024.

[16] Cámara ov7670 fifo al422 (ref: 0180). https://electronperdido.com/shop/sensores/luz/camara-ov7670-fifo/. Accedido el 11 de septiembre de 2024.

[17] Cámara ov7670 (ref: 0011). https://electronperdido.com/shop/sensores/luz/camara-ov7670/. Accedido el 11 de septiembre de 2024.

[18] Lente cctv de 8 mm con montura de 1/2,5 " lente de 5mp de alta definición compatible con visión nocturna por infrarrojos con alta definición. https://www.amazon.es/Montura-definici%C3%B3n-Compatible-Nocturna-Infrarrojos/dp/B08YNJNL8Q/. Accedido el 11 de septiembre de 2024.

[19] Te relay products. https://www.mouser.com/datasheet/2/418/ENG_SS_108-98001_S[1]-1210543.pdf?srsltid=AfmBOorbxjpbyCyMcu_yiZQ_GdSnkfLWk-uD7fFEZ5-RMf-EymeRpQgK. Accedido el 11 de septiembre de 2024.

[20] Arduino ide. https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/. Accedido el 11 de septiembre de 2024.

[21] Micropython. https://micropython.org/. Accedido el 11 de septiembre de 2024.

[22] Plataformio. https://platformio.org/. Accedido el 11 de septiembre de 2024.

[23] Librería arducam. https://www.arduino.cc/reference/en/libraries/arducam/. Accedido el 11 de septiembre de 2024.

[24] Librería lorawan. https://www.arduino.cc/reference/en/libraries/heltec_esp32_lora_v3/. Accedido el 11 de septiembre de 2024.

[25] Pytorch. https://pytorch.org/. Accedido el 11 de septiembre de 2024.

[26] Tensorflow. https://www.tensorflow.org/?hl=es-419. Accedido el 11 de septiembre de 2024.

[27] Noelia Hernández. Facebook pone su inteligencia artificial de código abierto al servicio de la ciencia. https://www.elespanol.com/invertia/disruptores/grandes-actores/tecnologicas/20210701/facebook-inteligencia-artificial-codigo-abierto-servicio-ciencia/593191230_0.html. Accedido el 11 de septiembre de 2024.

[28] Google ai. https://es.wikipedia.org/wiki/Google_AI. Accedido el 11 de septiembre de 2024.

[29] Chirpstack, open-source lorawan® network server. https://www.chirpstack.io/. Accedido el 11 de septiembre de 2024.

[30] Mqtt. https://mqtt.org/. Accedido el 11 de septiembre de 2024.

[31] Influxdb time series database. https://www.influxdata.com/. Accedido el 11 de septiembre de 2024.

[32] Grafana. https://grafana.com/. Accedido el 11 de septiembre de 2024.

[33] Lora y el duty cycle. https://lpwan.es/lora/lora-y-el-duty-cycle/. Accedido el 11 de septiembre de 2024.

[34] Airtime calculator for lorawan. https://avbentem.github.io/airtime-calculator/ttn/eu868/222. Accedido el 11 de septiembre de 2024.

[35] Water meters dataset. https://www.kaggle.com/datasets/tapakah68/yandextoloka-water-meters-dataset. Accedido el 11 de septiembre de 2024.

[36] Sheldon Mascarenhas and Mukul Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1, pages 96–99, 2021.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[38] Aspose.words for python. https://docs.aspose.com/words/python-net/. Accedido el 11 de septiembre de 2024.

[39] Pytesseract. https://pypi.org/project/pytesseract/. Accedido el 11 de septiembre de 2024.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[41] Detectron 2. https://ai.meta.com/tools/detectron2/. Accedido el 11 de septiembre de 2024.

[42] Saheb Chhabra, Puspita Majumdar, Mayank Vatsa, and Richa Singh. Data fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):8223–8230, Jul. 2019.

[43] Arducam shield rev.c. https://www.arducam.com/hardware-2/. Accedido el 11 de septiembre de 2024.

[44] Python. https://www.python.org/. Accedido el 11 de septiembre de 2024.

[45] Assefaw H Gebremedhin and Andrea Walther. An introduction to algorithmic differentiation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(1):e1334, 2020.

[46] Torchvision. https://pytorch.org/vision/stable/index.html. Accedido el 11 de septiembre de 2024.

[47] Easyocr: A comprehensive guide. https://medium.com/@adityamahajan.work/easyocr-a-comprehensive-guide-5ff1cb850168. Accedido el 11 de septiembre de 2024.

[48] Roberto Omar Andrade and Sang Guun Yoo. A comprehensive study of the use of lora in the development of smart cities. *Applied Sciences*, 9(22), 2019.

Anexos

Anexo I

Hardware y Software empleado

En este Anexo se va a realizar una explicación más detallada tanto del hardware como del software que se ha utilizado para llevar a cabo este proyecto.

I.1. Hardware

En este apartado se exponen los elementos de hardware que se han utilizado en el proyecto, mostrando una descripción detallada de ellos, así como el motivo por el que se han seleccionado.

Heltec Wifi LoRa 32 V3:

El Heltec Wifi LoRa 32 v3 es un módulo de comunicaciones diseñado para aplicaciones en el internet de las cosas que tiene integrado un microcontrolador ESP32, un módulo LoRa y soporte tanto para el uso de wifi como de bluetooth. También cuenta con una pantalla oled integrada que facilita la visualización de los datos.

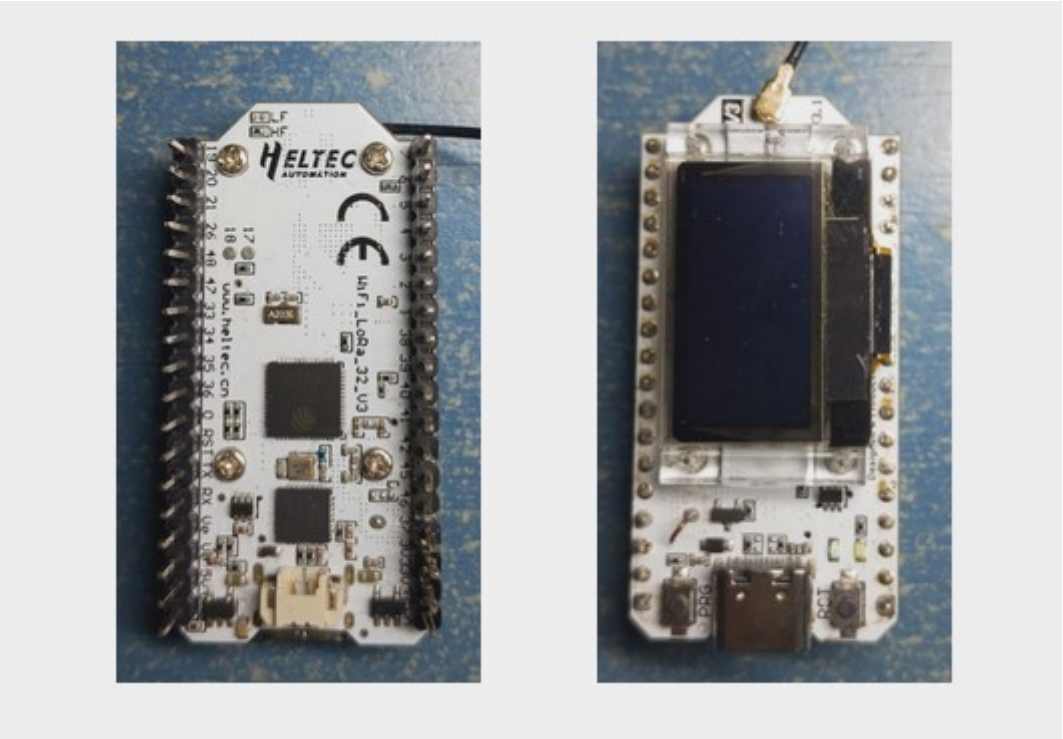

Figura 23: Heltec Wifi Lora 32 V3

Este dispositivo cuenta con múltiples pines que son imprescindibles para su propósito, permitiendo una gran variedad de comunicaciones y conexiones con otros dispositivos. Entre ellos destacan los pines de entrada salida de propósito general (GPIO) que proporcionan interfaces flexibles que pueden configurarse tanto para entrada como para salida, facilitando la interacción con sensores, actuadores y otros dispositivos electrónicos. Otro tipo de pines necesarios para este proyecto son los de circuito integrado (I2C), que incluyen el SDA, para el paso de datos y el SCL, para el reloj, y permiten la comunicación con múltiples dispositivos en un bus de datos compartido, como pantallas OLED, sensores ambientales y módulos de almacenamiento. Por último, destacar los pines de la interfaz periférica en serie (SPI), que son esenciales para la comunicación rápida y eficiente entre el microcontrolador ESP32 y el módulo LoRa, así como otros periféricos compatibles con SPI, como la cámara que se utiliza en este proyecto.

Especificaciones técnicas del Heltec Wifi LoRa 32 V3:

Parámetros	**Descripción**
Chip maestro	ESP32-S3FN8 (Xtensa® 32-bit lx7 dual core processor)
Nodo del chip LoRa	SX1262
USB a chip serie	CP2102
Frecuencia	470 510MHz, 863 928MHz
Máxima potencia de transmisión	21±1dBm
Máxima sensibilidad de recibo	-136dBm@SF12 BW=125KHz
Wi-Fi	802.11 b/g/n, up to 150Mbps
Bluetooth	Bluetooth 5 (LE)
Especificaciones Hardware	7*ADC1 + 2*ADC2; 7*Touch; 3*UART; 2*I2C; 2*SPI; etc.
Memoria	384KB ROM; 512KB SRAM; 16KB RTC SRAM; 8MB SiP Flash
Interfaz	Type-C USB; 2*1.25 lithium battery interface; LoRa ANT(IPEX1.0); 2*18*2.54 Header Pin
Batería	3.7V lithium battery power supply and charging
Dimensiones	50.2 * 25.5 * 10.2 mm

Tabla 5: Especificaciones del Heltec Wifi LoRa 32 V3

Las especificaciones técnicas del Heltec Wifi LoRa 32 V3 (vistas en la Tabla 5) son útiles en el propósito de este proyecto debido a que:

- El microprocesador ESP32-S3FN8 cuenta con un procesador Xtensa® de 32 bits con doble núcleo LX7 y una estructura de pipeline de cinco etapas, operando a una frecuencia de hasta 240 MHz. Este procesador ofrece un gran rendimiento y capacidad de procesamiento.

- Cuenta con un chip SX1262, el cual permite las comunicaciones LoRa, ideales para los entornos en los que se pretende establecer el dispositivo, y que permite transmitir datos con un poder de salida de 21 ± 1 dBm a una frecuencia de entre 470 510MHz y 863 928MHz.

- La interfaz de batería SH1.25-2 es un sistema de gestión de batería de litio que permite la carga y descarga, protege contra la sobrecarga y permite la carga automática de la batería desde el USB.

- Cuenta con una sensibilidad de recepción de hasta 125 KHz, lo que le permite captar señales débiles y asegurar una comunicación fiable incluso en entornos con alta interferencia y a largas distancias. Esta alta sensibilidad es fundamental para

aplicaciones IoT, ya que optimiza el rendimiento en escenarios donde la robustez y la eficiencia energética son cruciales, permitiendo a los dispositivos operar con una potencia de transmisión más baja y prolongando la vida útil de la batería.

Arducam Mini 2mp Plus:

El Arducam Mini 2mp Plus es un módulo de cámara diseñado para ser utilizado con microcontroladores. Cuenta con 2mp, lo que permite capturar imágenes en una resolución de hasta 1600x1200 píxeles. También permite la toma de imágenes en tres formatos diferentes: JPEG, BMP y RAW.

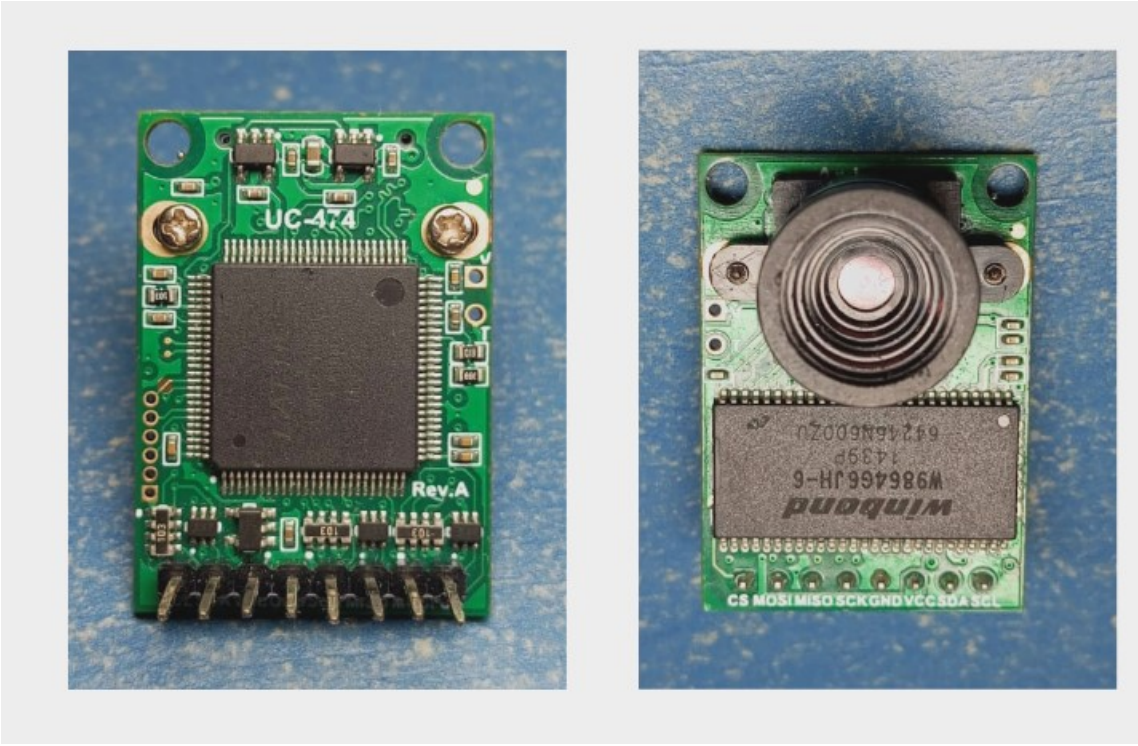

Figura 24: ArduCam Mini 2mp Plus

Este módulo es una versión optimizada del ArduCAM shield Rev.C [43]. Es una cámara SPI de alta definición con una resolución de 2MP, diseñada para simplificar la interfaz de control de la cámara. Incluye un sensor de imagen CMOS OV2640 de 2MP y tiene un tamaño compacto. El ArduCAM mini puede funcionar con múltiples plataformas como Arduino, Raspberry Pi o Maple, siempre que tengan interfaces SPI e I2C y sean compatibles con las placas Arduino estándar. El ArduCAM mini no solo permite añadir una cámara a microcontroladores de bajo coste que no tienen esta capacidad, sino que también permite conectar múltiples cámaras a un solo microcontrolador.

Este dispositivo se caracteriza por:

- Contar con un sensor de imagen de 2mp OV2640 el cual permite capturar imágenes con una resolución de hasta 1600x1200 píxeles. El OV2640 es conocido por su bajo consumo de energía y su alta calidad de imagen, incluso en resoluciones muy bajas, lo que lo hace perfecto para este proyecto.
- Todos los puertos de entrada y salida (E/S) del sensor pueden manejar señales tanto de 5V como de 3.3V. Esto hace que el sensor sea compatible con una amplia gama de microcontroladores y dispositivos electrónicos.
- Permite la compresión en JPEG, lo que reduce significativamente el tamaño de las imágenes y las hace más fáciles para mandarlas por la red.
- Tiene modo de disparo único o múltiple, lo que permite tomar una sola foto o una serie de fotos en rápida sucesión.
- Tiene la opción de lectura en ráfaga, que permite la captura y lectura continua de imágenes.
- Utiliza un modo de bajo consumo, lo que es esencial en este trabajo para que se ahorre la energía y se pueda utilizar una batería por más tiempo.
- Cuenta con una librería de código abierto para plataformas como STM32, Chipkit, Raspberry Pi y BeagleBone Black.
- Tiene un tamaño compacto, lo que es imprescindible para que el tamaño del prototipo sea lo más pequeño posible.
- La lente cuenta con un campo de visión horizontal (HFOV) de 60 grados y una distancia focal efectiva (EFL) de 4.9mm. Esto implica que la imagen se pueda tomar lo suficientemente cerca del contador sin que pierda calidad.

Esta pieza se conecta al dispositivo mediante dos interfaces, I2C y SPI. Utiliza la interfaz I2C para la configuración del sensor, con él se pueden ajustar la configuración de la cámara y los parámetros de esta, como la exposición, el balance de blancos, el brillo, el contraste y otras configuraciones de imagen. La interfaz SPI se utiliza principalmente para la transferencia de datos de la fotorgafía entre el módulo de la cámara y el microcontrolador conectado, a parte, a través de esta interfaz se controlan las funciones de la cámara, como iniciar y detener la captura de imágenes o configurar la resolución.

Los pines que utiliza son:

- CS: Entrada de selección de chip esclavo SPI. Este pin se usa para seleccionar el dispositivo esclavo específico al que el maestro quiere comunicarse en una configuración SPI con múltiples esclavos.
- MOSI: Salida del maestro y entrada del esclavo en la comunicación SPI. Los datos se envían desde el maestro al esclavo a través de este pin.
- MISO: Salida del maestro y entrada del esclavo en la comunicación SPI. Los datos se envían desde el esclavo al maestro a través de este pin.
- SCLK: Entrada del reloj serie SPI. Este pin proporciona la señal de reloj que sincroniza la transferencia de datos entre el maestro y el esclavo.
- GND: Conexión a tierra. Este pin se utiliza para establecer un punto de referencia común para el voltaje y completar el circuito eléctrico.
- +5V: Suministro de energía de 5V. Este pin proporciona la alimentación necesaria para el funcionamiento del dispositivo.
- SDA: Línea de datos de la interfaz serie de dos hilos (I2C). Este pin se utiliza para la transferencia de datos bidireccional entre los dispositivos en la comunicación I2C.
- SCL: Línea de reloj de la interfaz serie de dos hilos (I2C). Este pin proporciona la señal de reloj que sincroniza la transferencia de datos entre los dispositivos en la comunicación I2C.

Relé:

Se ha utilizado un relé para controlar el momento en el que la cámara está encendida. Esto se ha hecho ya que, de no utilizarlo, la cámara consumía una gran cantidad de energía pese a no estar enchufada ni en uso. El relé que se ha utilizado es un relé IM 01, este cuenta con varias entradas necesarias para su correcto funcionamiento. La primera, es la entrada de la corriente que se quiere llevar a la salida, en caso de que la cámara requiera de 3,3 V para funcionar, se conectará el pin 3V3 del microprocesador a esta entrada y el pin VCC de la cámara a una de las salidas. Dichas salidas son dos, una que permite el paso de la corriente cuando el interruptor está apagado y otra que permite el paso de la corriente cuando el interruptor está encendido. Por último,

la manera en la que se enciende y se apaga el interruptor es encendiendo o apagando el relé, de manera que cuando el relé está encendido el interruptor está enchufado y cuando el relé no está encendido el interruptor esta cerrado.

Este componente cuenta con las siguientes especificaciones:

- Voltaje nominal: 3 VDC
- Voltaje operativo: 2.25 VDC
- Voltaje de liberación; 0.45 VDC
- Resistencia a la bobina: 64 Omios
- Potencia: 140 mW
- Intensidad: 42 mA

Se ha seleccionado este relé debido a que necesita menos de 3.3 V para poder funcionar y, a su vez, menos de 42 mA. Estos datos son importantes ya que la manera en la que se va a utilizar va a ser conectando su alimentación a un GPIO para poder apagar y encender la cámara cuando se necesite. Los GPIOs del Heltec Wifi Lora 32 V3 pueden ofrecer un voltaje máximo de 3.3 V y una intensidad de hasta 45 mA, es por ello que el relé se va a poder utilizar mediante un GPIO.

I.2. Software

En este apartado se va a explicar el software que ha sido necesario para llevar a cabo la realización de este proyecto. Se va a explorar el software que se ha utilizado dentro del microcontrolador, el que se ha utilizado para realizar el OCR de las imágenes y la base de datos en la que se han guardado los resultados.

Arduino IDE:

El Arduino Integrated Development Environment (IDE) es una plataforma de desarrollo de software diseñada específicamente para la programación de microcontroladores Arduino. Proporciona un entorno intuitivo y accesible que permite a desarrolladores escribir, compilar y cargar código en placas Arduino de manera eficiente.

El IDE incluye una variedad de características útiles, como un editor de código con resaltado de sintaxis, autocompletado y verificación de errores en tiempo real, permite buscar e importar librerías de una forma sencilla o lo posibilidad de ver los datos que se transmiten a través de la interfaz serial desde la propia aplicación.

A pesar de que el Heltec WiFi LoRa v3 no es una placa oficial de Arduino, es compatible con el Arduino IDE gracias a la comunidad de desarrolladores que ha creado y mantenido una biblioteca específica para esta placa, permitiendo a los usuarios programarla y configurarla de manera sencilla utilizando el entorno del Arduino IDE.

La preferencia de este entorno frente a otros como MicroPython o PlatformIO se debe a varias razones clave, entre las que se puede destacar su amplio soporte y la gran comunidad que lo apoya. Esta base de usuarios activa ha contribuido al desarrollo de una gran colección de bibliotecas, documentación y recursos. Estas librerías simplifican la integración de hardware y la implementación de funcionalidades complejas. Las principales librerías por las que se ha decidido utilizar Arduino IDE son la de "ArduCAM" y la de "Heltec_ESP32_Lora_V3".

La librería de "ArduCAM" es una herramienta especializada diseñada para interactuar con los módulos de cámaras desarrollados por Arduino. Gracias a ella se puede manejar de manera sencilla todos los parámetros de la cámara, incluyendo configuraciones avanzadas de resolución, formato de salida, control de exposición, y otras características específicas del sensor de imagen. Mediante métodos definidos en la API, se puede gestionar la inicialización de la cámara, el inicio y detención de la captura de imágenes y vídeo, la calidad que se desea obtener, así como la manipulación y transmisión de los datos capturados a través de interfaces de comunicación como SPI o I2C.

La librería de "Heltec_ESP32_Lora_V3" sirve para facilitar la implementación y gestión de comunicaciones utilizando el protocolo LoRaWAN en dispositivos Arduino compatibles con módulos LoRa. Esta librería permite realizar la configuración de los parámetros necesarios para la comunicación, como la frecuencia de operación, el factor de esparcimiento (spreading factor), la potencia de transmisión y otros ajustes de red. Además facilita la gestión de sesiones de conexión y autenticación con el servidor LoRaWAN. También proporciona métodos para enviar y recibir mensajes de datos en formato LoRaWAN.

Reconocimiento óptico de caracteres:

Para la realización del reconocimiento óptico de caracteres se ha utilizado el lenguaje de programación Python [44], ya que ofrece una amplia variedad de bibliotecas y herramientas especializadas que facilitan el desarrollo de aplicaciones de procesamiento de imágenes y análisis de textos.

En este apartado se van a mostrar y explicar todas las bibliotecas que se han utilizado para realizar el OCR en este proyecto, estas son tanto PyTorch como Tensorflow. Ambas se utilizan para la ejecución de las redes neuronales realizadas para el reconocimiento de los caracteres del consumo del agua. Pese a que en el trabajo final tan solo se utiliza PyTorch, se ha visto conveniente explicar ambas ya que alguno de los modelos que se han probado se han realizado con Tensorflow.

- PyTorch:

  PyTorch es una librería de python que se utiliza para la creación y entrenamiento de modelos de aprendizaje automático y aprendizaje profundo. Desarrollada por Facebook's AI Research lab (FAIR), PyTorch ofrece una serie de herramientas que facilitan la creación y el entrenamiento de las redes neuronales.

  Una de las funciones más importantes que ofrece esta librería es el "tensor computation", la cual es una API que se utiliza para realizar cálculos numéricos con tensores, que son los objetos básicos que representan los datos y los cálculos. Esta librería permite aprovechar la GPU y el "autograd", que se refiere a la capacidad de PyTorch para realizar diferenciación automática. La diferenciación automática es [45] un conjunto de técnicas que se utilizan para evaluar de manera eficiente las derivadas de las funciones. En el contexto del aprendizaje profundo, se calculan las derivadas de las funciones de pérdida para poder actualizar los valores de la red y que, de esa manera, aprenda con cada iteración del entrenamiento.

  Además esta librería ofrece una extensión llamada TorchVision [46], la cual ofrece conjuntos de datos, modelos predefinidos y funciones de transformación de imágenes que son comúnmente utilizados.

- TensorFlow:

  TensorFlow es una librería de software de código abierto desarrollada por Google Brain, una sección dentro de Google AI, para el aprendizaje automático y la inteligencia artificial. Se utiliza principalmente para la construcción y el entrenamiento de redes neuronales profundas, aunque también es adecuada para una amplia variedad de tareas de aprendizaje automático.

El funcionamiento de TensorFlow se basa en la construcción y ejecución de gráficos de flujo de datos, donde los nodos representan las operaciones matemáticas y los datos, en forma de tensores, fluyen entre ellos.

Además TensorFlow permite aprovechar al máximo las Unidades de Procesamiento Gráfico (GPU) para acelerar el entrenamiento y la inferencia de modelos de aprendizaje automático. Las GPU están diseñadas para manejar cálculos matemáticos en paralelo, lo que las hace ideales para tareas de aprendizaje profundo que involucran operaciones intensivas con matrices y tensores.

ChirpStack:

ChirpStack es una plataforma de código abierto diseñada para implementar redes LoRaWAN (Red de Área Amplia de Baja Potencia). Actúa como el sistema encargado de gestionar estas redes, ofreciendo una infraestructura para poder desplegar y administrar dispositivos IoT que utilizan LoRaWAN.

El funcionamiento de ChirpStack se basa en una arquitectura modular, lo que permite una integración flexible con diferentes componentes de la red. En su núcleo tiene el ChirpStack Network Server. Este es el componente central de la plataforma, donde se procesa la mayor parte de la lógica de la red. Su función principal es gestionar las conexiones de los dispositivos LoRaWAN, verificar la autenticidad y la integridad de los mensajes recibidos, y garantizar la correcta encriptación y autenticación de los datos, conforme a las especificaciones del protocolo LoRaWAN. El servidor de red es responsable de gestionar el espectro de frecuencias y programar la retransmisión de mensajes, optimizando el uso de los recursos de red.

Otro componente fundamental es el ChirpStack Application Server, que se encarga de recibir los datos procesados por el Network Server y traducirlos en un formato útil para las aplicaciones que los necesitan. Este servidor es el que finalmente distribuye los datos a las aplicaciones de usuario final mediante el uso de protocolos como MQTT.

De esta manera, cuando se envía un mensaje mediante LoRaWAN los gateways lo envían a ChirpStack, el cual filtra y traduce los mensajes para hacerlos más accesibles para las aplicaciones que lo necesiten. A su vez los publica mediante protocolos como MQTT para que estas aplicaciones lo obtengan y lo utilicen.

MQTT es un protocolo de comunicación ligero diseñado para la transmisión de datos entre dispositivos con baja potencia y conectividad limitada. Es ampliamente

utilizado en aplicaciones del Internet de las Cosas.

Una de las principales características de MQTT es su modelo de publicación/suscripción. Esto significa que los dispositivos, conocidos como clientes, pueden publicar mensajes en temas específicos y, a su vez, otros dispositivos pueden suscribirse a estos temas para recibir la información relevante. Este enfoque reduce la necesidad de conexiones punto a punto entre los dispositivos, lo que lo hace especialmente adecuado para redes con muchos dispositivos interconectados.

InfluxDB:

InfluxDB es una base de datos de series temporales (TSDB) diseñada para gestionar grandes volúmenes de datos generados a lo largo del tiempo. Esta base de datos se distingue por su capacidad para almacenar, consultar y analizar datos que son recolectados en intervalos regulares, lo cual es particularmente útil en aplicaciones de monitorización, analítica y gestión de métricas. Este tipo de datos es común en entornos donde se requiere el seguimiento continuo de eventos, como en sistemas de monitorización de infraestructuras, Internet de las Cosas (IoT), aplicaciones financieras y análisis de rendimiento de aplicaciones.

La forma en la que trabaja InfluxDB es empleando un motor de almacenamiento optimizado para añadir y consultar datos temporales. Los datos se organizan en series temporales compuestas por puntos de datos, cada uno de los cuales tiene un tiempo asociado. Los puntos de datos se almacenan en una estructura denominada "measurement", qué es análoga a una tabla en una base de datos relacional, y pueden ser etiquetados con "tags"para facilitar su consulta. InfluxDB utiliza un modelo de almacenamiento basado en LSM (Log-Structured Merge) para optimizar las operaciones de escritura, y ofrece una API de consulta basada en el lenguaje InfluxQL, que permite ejecutar consultas eficientes sobre los datos temporales almacenados.

La elección de InfluxDB como base de datos para series temporales en este proyecto se basa en tres razones clave. En primer lugar, InfluxDB es ideal para aplicaciones de monitorización debido a su enfoque en series temporales. Las bases de datos de series temporales están diseñadas específicamente para gestionar datos que cambian con el tiempo, lo que las hace perfectas para almacenar y analizar métricas de rendimiento, registros de eventos y datos de sensores.

Otra razón para utilizar InfluxDB es su facilidad de visualización y monitorización de los datos. InfluxDB se integra perfectamente con sistemas de visualización

como Grafana, lo que permite crear dashboards interactivos y personalizables. Estos dashboards son esenciales para monitorizar y analizar datos en tiempo real, proporcionando una visión clara y concisa del estado y rendimiento de los sistemas.

Finalmente, InfluxDB ofrece una fácil integración con herramientas como Chipstack y lenguajes de programación como Python. Esta capacidad de integración es crucial para el desarrollo de aplicaciones de monitorización y análisis de datos, ya que permite a los desarrolladores aprovechar las librerías y frameworks existentes en Python, facilitando así el procesamiento y análisis de datos en tiempo real.

Grafana:

Grafana es una plataforma de software libre de código abierto diseñada para la visualización y el análisis de datos en tiempo real. Se utiliza principalmente para crear gráficos, paneles interactivos y alertas, permitiendo a los usuarios monitorizar métricas provenientes de diversas fuentes de datos.

Grafana destaca por su capacidad para conectarse a una amplia variedad de fuentes de datos, tales como bases de datos SQL, NoSQL, sistemas de monitoreo como Prometheus, o incluso APIs externas. Esto permite que los usuarios puedan centralizar la visualización de sus métricas en un único lugar, sin importar la diversidad de sus orígenes.

Otra funcionalidad de Grafana es su capacidad de personalización de los dashboards. Se pueden construir dashboards a medida, combinando distintos tipos de visualizaciones como gráficos de líneas, barras, tablas, mapas de calor, entre otros. Además, Grafana ofrece opciones avanzadas de configuración, permitiendo modificar aspectos visuales, definir filtros globales o variables, y organizar las visualizaciones en múltiples capas o paneles.

Una característica importante es el sistema de alertas. Grafana permite configurar alertas basadas en condiciones específicas de las métricas monitorizadas. Estas alertas pueden ser enviadas a través de distintos canales de comunicación, como correo electrónico, Slack, o integraciones personalizadas mediante webhook. Esto asegura que los equipos responsables sean notificados inmediatamente en caso de que alguna métrica supere umbrales predefinidos, facilitando la toma de decisiones rápidas.

La elección de Grafana para la visualización de los datos se debe a dos razones principales:

– Su integración nativa con InfluxDB. Esta integración facilita el proceso de conexión y extracción de datos, permitiendo que la información almacenada en InfluxDB sea visualizada de manera rápida y eficiente en los dashboards de Grafana.

– Su sistema de alertas. El cual es esencial para avisar a los usuarios en caso de que se detecte alguna anomalía, permitiendo así actuar lo antes posible ante posibles fugas de agua u otros problemas que se puedan detectar.

Anexo II

Arquitectura LoRaWAN

LoRaWAN es una capa de enlace que organiza cómo los dispositivos LoRa se comunican entre sí y con los servidores. LoRa (Long Range) por su parte, es la capa física de la tecnología, desarrollada por Semtech Corporation. Utiliza una modulación por espectro ensanchado que permite comunicaciones de largo alcance con bajo consumo de energía, a costa de una velocidad de datos relativamente baja.

La arquitectura de LoRaWAN sigue un modelo en estrella donde los dispositivos finales (sensores, actuadores, etc.) se comunican directamente con puertas de enlace (gateways) mediante el protocolo LoRa. Estas puertas de enlace actúan como intermediarios, retransmitiendo los datos a un servidor de red central a través de una conexión de backhaul (que puede ser Internet, 3G, Ethernet, etc.). El servidor de red procesa y gestiona los datos, aplicando políticas de red, autenticación y administración de dispositivos. Finalmente, los datos son enviados a servidores de aplicaciones donde se procesan y se utilizan para fines específicos.

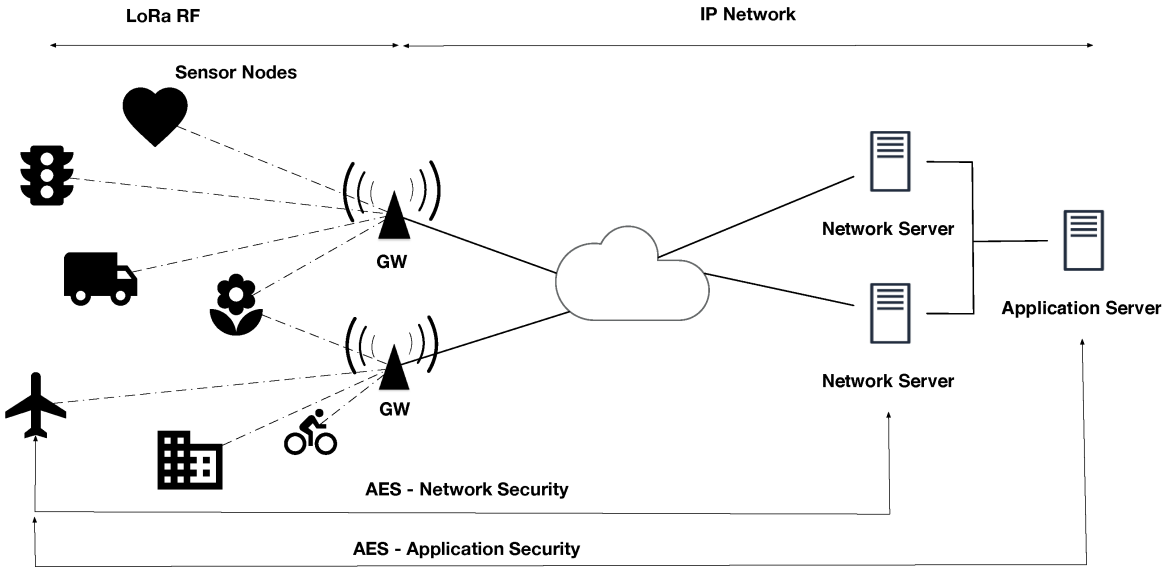

Figura 25: Arquitectura LoRaWAN [3]

Como se puede ver en la Figura 25, el proceso de comunicación sigue una serie de pasos. Lo primero que ocurre es que el dispositivo final, equipado con un módulo LoRa, transmite los datos. La transmisión se realiza en frecuencias sub-GHz, que varían según la regulación regional (por ejemplo, 868 MHz en Europa y 915 MHz en América del Norte).

Tras eso, las puertas de enlace, también llamadas gateways, están ubicadas estratégicamente para cubrir áreas amplias. Cada transmisión de un dispositivo final puede ser recibida por múltiples puertas de enlace si están dentro del alcance. Estas puertas de enlace no realizan ningún procesamiento de datos, sino que simplemente actúan como repetidores, enviando las señales recibidas al servidor de red. La redundancia en la recepción de señales por múltiples puertas de enlace aumenta la fiabilidad del sistema.

Una vez llegan los datos a las puertas de enlace, estas los transmiten al servidor de red utilizando conexiones de backhaul, que pueden ser a través de Internet, redes celulares (3G, 4G) o conexiones Ethernet. El servidor de red es el corazón del sistema LoRaWAN, encargado de coordinar y gestionar toda la comunicación entre los dispositivos finales y las aplicaciones.

El servidor de red recoge los datos de todas las puertas de enlace y elimina los duplicados y verifica la autenticidad de los mensajes. Una vez que los datos son validados y procesados en el servidor de red, son enviados a los servidores de aplicación específicos a través de una interfaz de aplicación.

Para gestionar de manera eficiente la red LoRaWAN, se ha utilizado el servidor de red ChirpStack [29], una plataforma de código abierto que facilita la administración, monitorización y configuración de dispositivos LoRaWAN. ChirpStack actúa como el intermediario que recoge los datos de los dispositivos LoRaWAN a través de las gateways y los procesa antes de enviarlos a las aplicaciones de los usuarios finales.

En la comunicación LoRa existen tres términos esenciales que describen como va a ser la comunicación. Estos son el "Spreading Factor", el "Data Rate" y el ancho de banda.

- Spreading Factor (SF): Determina la relación entre la velocidad de transmisión y el alcance de la señal. LoRa permite utilizar SF entre 6 y 12, y contra más alto el valor mayor, más se reduce la cantidad de Bytes que se pueden enviar pero mayor es el alcance de la comunicación. Con SF10, SF11 y SF12 se pueden enviar 51 Bytes de datos, con SF9 123 Bytes y con SF8, SF7 y SF6 hasta 222 Bytes de datos.
- Data Rate (DR): Es la velocidad a la que se envían los datos, generalmente se mide en bits por segundo (bps). Una tasa de datos más alta reduce la sensibilidad del receptor, lo que puede limitar el alcance de la comunicación. Se pueden utilizar desde DR0 hasta DR5. Contra menor sea el Data rate, menor será la velocidad y, a su vez, mayor será la distancia.
- Ancho de banda (BW): El ancho de banda es el rango de frecuencias utilizado para la transmisión de la señal LoRa. Un ancho de banda mayor permite una tasa de datos más alta.

$$DR = SF \times \frac{1}{\frac{|2 \times SF|}{BW}} bits/sec$$

Anexo III

Funcionamiento de los modelos de OCR basados en librerías

En este anexo se van a explicar en detalle el funcionamiento de los modelos basados en librerías de Python con los que se han realizado las pruebas del OCR. Estos modelos son Aspore, EasyOCR y PyTesseract.

Aspore:

Aspore (Adaptive Sparse Online Regression) es un método avanzado de aprendizaje automático diseñado para resolver problemas de regresión y clasificación. Utiliza técnicas de regresión escasa (en inglés, sparse regresion) y adaptación en línea, lo que permite que el modelo se ajuste continuamente a nuevos datos sin necesidad de ser entrenado desde cero. Esto es particularmente útil en contextos donde los datos llegan en tiempo real y el modelo debe mantenerse actualizado de manera eficiente.

La regresión escasa es una técnica que, a diferencia de los métodos tradicionales de regresión que pueden incluir una gran cantidad de variables predictoras, se centra en seleccionar un subconjunto pequeño de variables relevantes. Esta selección se hace de tal manera que el modelo resultante sea más interpretable y menos propenso al sobre ajuste.

Una de las aplicaciones de Aspore, y la que se le ha dado en este proyecto, es el reconocimiento de dígitos, un problema clásico en el campo de la visión por computadora y el procesamiento de imágenes. Este problema implica identificar correctamente los dígitos del 0 al 9 a partir de imágenes digitalizadas.

EasyOCR:

EasyOCR es [47] una biblioteca de código abierto diseñada para realizar el reconocimiento óptico de caracteres. Desarrollada por Jaided AI, EasyOCR permite a los usuarios extraer texto de imágenes con alta precisión. Es compatible con más de 80 idiomas, incluyendo aquellos que utilizan alfabetos no latinos, como el chino, japonés, coreano y árabe.

El reconocimiento de caracteres en EasyOCR se basa en una combinación de técnicas avanzadas de procesamiento de imágenes y aprendizaje profundo, específicamente redes neuronales convolucionales (CNN) y modelos secuenciales como las redes neuronales recurrentes (RNN).

PyTeseract:

PyTesseract es una biblioteca de código abierto que proporciona una interfaz para el motor Tesseract OCR, desarrollado por Google. Esta herramienta permite extraer texto de imágenes utilizando las capacidades de Tesseract. PyTesseract es altamente valorada por su capacidad para manejar una amplia variedad de idiomas y su fácil integración con aplicaciones de Python.

El reconocimiento óptico de caracteres en PyTesseract se basa en una combinación de técnicas avanzadas de procesamiento de imágenes y algoritmos de aprendizaje automático. El proceso se inicia con el preprocesamiento de la imagen, que incluye la conversión a escala de grises, la normalización del contraste y la eliminación de ruido. Luego, Tesseract analiza la imagen para identificar patrones de texto, utilizando técnicas de segmentación para dividir la imagen en bloques de texto, líneas y caracteres individuales.

Tesseract utiliza redes neuronales para reconocer cada carácter, comparándolo con patrones almacenados en su base de datos interna. Al igual que EasyOCR, el modelo que utiliza esta librería se basa en una RNN, en concreto utilizan un modelo de memoria a largo plazo (Long Short-Term Memory, LSTM) el cual, a parte de la función de las RNN que permiten acordarse de valores anteriores y utilizarlos, permite recordar durante largos periodos de tiempo valores que considera importantes.

Este enfoque permite una alta precisión en la detección de texto, incluso en imágenes con baja calidad o con fondos complejos. Una vez que se ha extraído el texto, PyTesseract puede aplicar técnicas de postprocesamiento para corregir errores

y mejorar la legibilidad del resultado final.

Anexo IV

Pruebas Batería

Uno de los requisitos que se pretenden cumplir en este proyecto es el realizar un prototipo el cual se adapte a las dificultades de los lugares en los que suelen estar los contadores de agua. Una de esas dificultades es el difícil acceso a la corriente eléctrica y para ello era necesario que el dispositivo consumiera muy poca batería, ya que en caso de que se tuviera que cambiar la batería a menudo, el prototipo no sería útil.

En este anexo se van a explicar las pruebas realizadas y los problemas que se han encontrado con respecto al consumo de la batería.

Al enviar muy pocos mensajes al día y únicamente realizar una o dos fotos por día, que es lo que más batería consume, se puso el objetivo de que la batería utilizada durara entorno a medio año o un año entero. Ya que de este modo no es necesita que el usuario esté cambiándola constantemente.

Lo primero que se ha hecho ha sido comprobar el tiempo que dura el dispositivo encendido, y para ello se han utilizado dos baterías diferentes. Una de ellas era de 700 mAh y la otra de 2000 mAh. Se han obtenido los siguientes resultados:

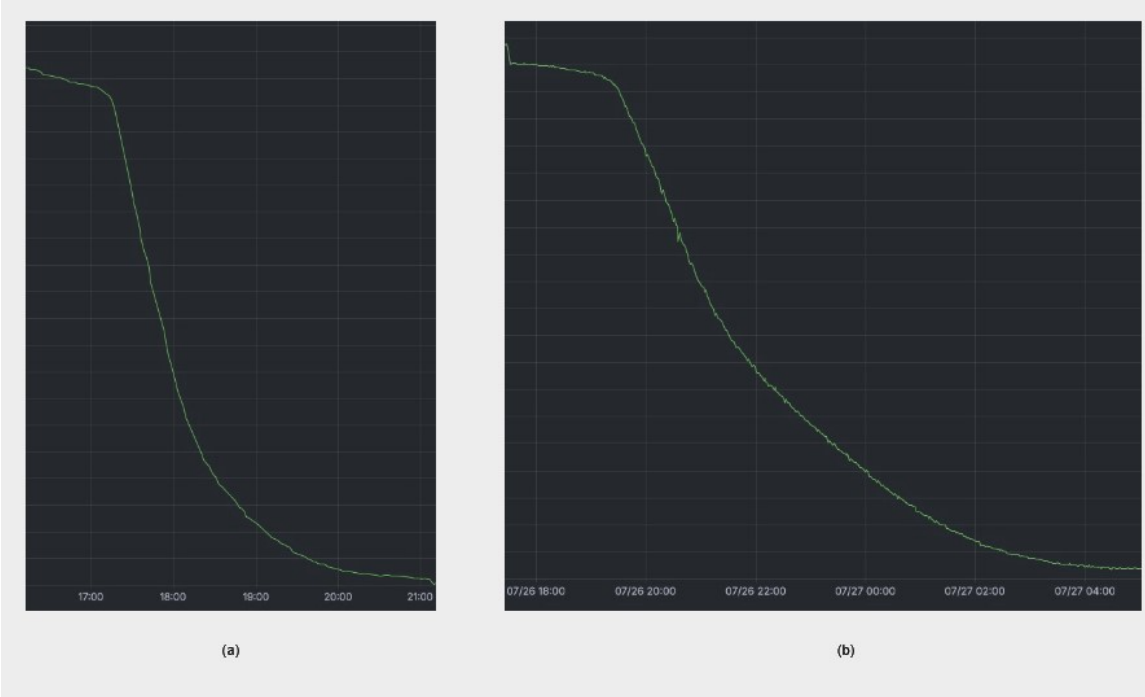

Figura 26: Gráficas del consumo de batería: (a) 700 mAh (b) 2000 mAh

Como se puede ver en la Figura 26 ninguna de los dos consumos cumple con los requerimientos. En el caso de la batería de 700 mAh se puede ver que dura entorno a las cinco horas con el programa encendido, mientras que en el caso de la batería de 2000 mAh este tiempo aumenta hasta doce horas.

Estos resultados no solo no se acercan a los resultados que se quieren obtener, sino que ni siquiera se asemejan a los resultados realizados con otro programa que no hacían uso del módulo de la cámara. Este programa que se menciona envía mensajes LoRa de manera intermitente cada minuto, lo que quiere decir que está mucho más tiempo despierto que el programa que se utiliza en este proyecto. Pese a eso, con la batería de 700 mAh, llega a durar varios días encendido.

Tras ver que la batería no duraba lo que debería, se optó por utilizar un medidor de consumo para observar cuanto consumía en cada momento. Para ello se probaron con los dos programas mencionados y se hicieron diferentes pruebas. Los resultados fueron los siguientes:

Prueba	**Consumo (miliamperios)**
Programa final con la cámara conectada, enviando	280 - 320
Programa final con la cámara conectada, dormido	150 - 170
Programa final sin la cámara conectada, dormido	0
Programa final sin la cámara conectada, enviando	200 - 230
Programa de prueba con la cámara conectada, enviando	250 - 300
Programa de prueba con la cámara conectada, dormido	150
Programa de prueba sin la cámara conectada, enviando	190 - 230
Programa de prueba sin la cámara conectada, dormido	0

Tabla 6: Tabla consumo de la batería

Como se puede observar en la Tabla 6 la diferencia de consumo cuando la cámara está conectada y cuando no lo está no se nota en gran medida mientras el programa está en funcionamiento. El problema viene cuando el programa está dormido, ya que cuando la cámara no está conectada al dispositivo, este consume prácticamente cero Amperios, mientras que al estar conectada consume entre 15 y 17.

Este problema se debe a que el pin utilizado para la alimentación de la cámara en el Heltec Wifi Lora 32 V3 (uno de los pines 3V3) está directamente conectado a la batería de este, lo que provoca que no se pueda controlar su apagado y, por ello, que siempre que haya un dispositivo conectado le esté dando energía.

Se han probado diferentes soluciones para este problema mediante software como poner el dispositivo en "deep sleep" o cambiar el pin de alimentación a un GPIO, y controlarlo. Pero ninguna ha dado resultados.

Es por ello que se acabó optando por la opción de utilizar un relé, el cual, controla la corriente que se le da a la cámara mediante el GPIO 7. De esta manera cuando el GPIO 7 está activado, la cámara recibe corriente, por lo que se puede encender y ser utilizada. Cuando se deja de utilizar la cámara, el relé corta la corriente e impide que la cámara consuma energía.

Con este cambio en el diseño se consiguió que la cantidad que consumía el dispositivo cuando estaba en estado de reposo pasara de entre 15 y 17 Amperios a 0 Amperios, mejorando en gran medida su rendimiento y permitiendo alargar la duración de la batería.

Prueba con el uso de un relé	**Consumo (miliamperios)**
Cámara conectada, enviando	290
Cámara conectada, dormido	0
Cámara desconectada, dormido	0
Cámara desconectada, enviando	230 - 250

Tabla 7: Tabla consumo de la batería

En la Tabla 7 se puede ver el consumo de batería de los diferentes estados del dispositivo tras añadirle el relé.

Anexo V

Diseño 3D

Los entornos en los que están situados los contadores de agua, sobre todo en zonas rurales, tienden a tener unas condiciones muy limitadas, estando en zonas pequeñas, sin acceso a la corriente eléctrica, con ningún tipo de iluminación y difícil acceso.

Para la correcta realización de este proyecto se ha tenido que desarrollar un prototipo personalizada, la cual se ha producido con una impresora 3D. En concreto se ha utilizado una Bambu Lab X1-Carbon, la cual destaca por su alta velocidad, precisión y capacidad para producir piezas detalladas con diversos materiales.

Este prototipo se ha creado teniendo en cuenta los problemas que se han visto durante el desarrollo del proyecto, como la iluminación a la hora de realizar una foto o la distancia a la que debería estar la cámara. Además se han tenido en cuenta las condiciones del entorno en el que se va a utilizar este modelo.

En este apartado de la memoria se va a abarcar el desarrollo que se ha llevado a cabo para la creación del prototipo, tanto necesidades que tiene que cubrir el prototipo, como la explicación de este.

V.1. Requisitos técnicos

En este apartado se detallan los requisitos técnicos necesarios para el correcto funcionamiento del proyecto final, abordando tanto los necesarios para la realización del reconocimiento de los dígitos, como los que lo son para facilitar su uso en entornos de condiciones limitadas.

Primero, hay que tener en cuenta que la cámara ha de estar centrada con el contador,

para poder realizar la foto de manera correcta sin perder ninguno de los dígitos de este. Además esta ha de estar a una distancia lo más reducida posible de los dígitos, ya que al hacer fotos de muy baja calidad no se puede permitir perder píxeles en elementos que no son relevantes.

Otro requisito está relacionado con el OCR es el de la iluminación. Como se ha podido observar en uno de los apartados anteriores la iluminación no afecta en gran medida al OCR, pero es necesario que exista. Es por eso que se decidió utilizar el led que lleva integrado el Heltec Wifi Lora 32 V3 con el fin de ahorrar costes. El problema viene dado a la hora de la posición de este led, ya que si está centrado puede llegar a deslumbrar la cámara y provocar que no se pueda ver bien la imagen.

Debido a que los entornos en los que se va a utilizar este prototipo son pequeños, se requiere crear una dispositivo lo suficientemente compacto y pequeño para que se pueda adaptar a ellos.

Por último, como en la mayoría de los casos no existe una forma de mantenerlos cargados continuamente, se requiere crear un módulo de batería fácil de cambiar. Permitiendo al usuario cambiar la batería sin necesidad de entender en detalle el funcionamiento del dispositivo.

V.2. Modelado 3D

En esta subsección se explica el prototipo que se ha llevado a cabo, y cómo se han implementado los requisitos vistos en el apartado anterior.

Para la realización de este prototipo se ha utilizado el software llamado Autodesk Fusion 360. Este software es una plataforma de diseño 3D basada en la nube ofrece capacidades de modelado paramétrico, directo, de superficies y de malla, facilitando el diseño de piezas y ensamblajes complejos.

Para explicar el prototipo realizado se va a dividir este en tres partes, cada una de las cuales tiene una función en específico dentro del prototipo.

La primera, y la principal, es la parte que contiene tanto la cámara como el Heltec Wifi Lora 32 V3.

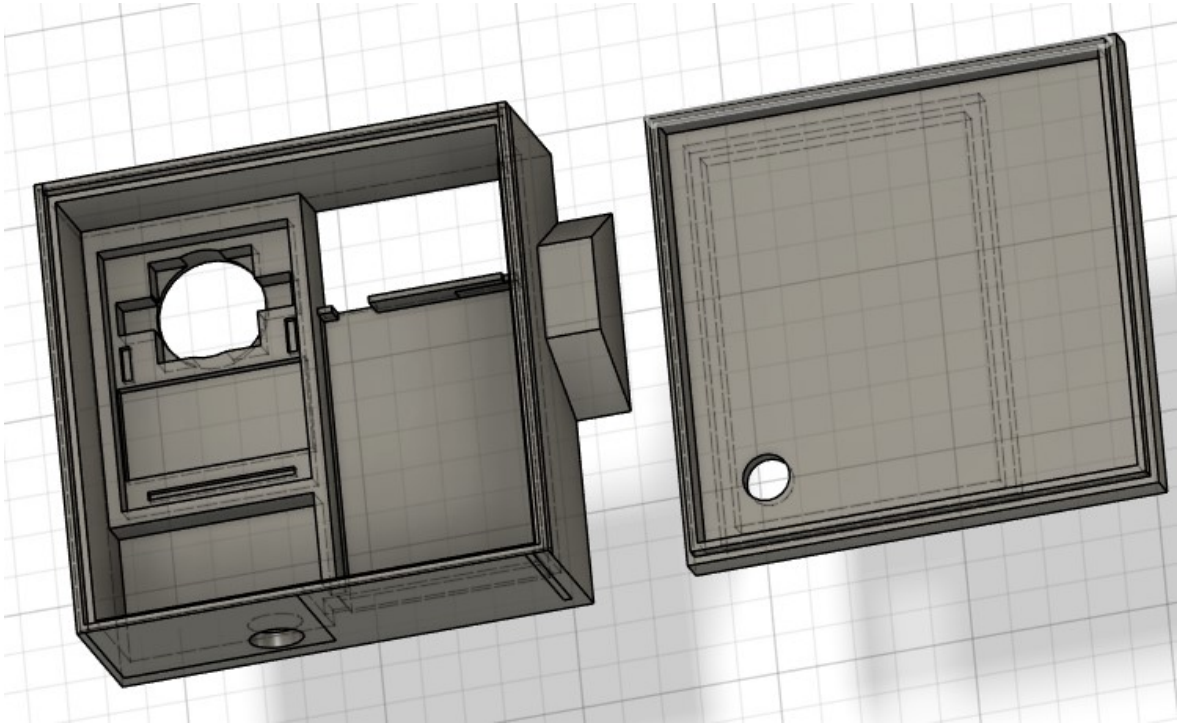

Figura 27: Pieza principal 3D

En la Figura 27 se muestra esta parte del prototipo, la cual está compuesta por una caja y una tapa. En la caja se pueden observar varios elementos, a la izquierda cuenta con un módulo para incrustar la cámara y que se quede fija en esa posición. Justo a la derecha de este módulo tiene otro, este sirve para el Heltec Wifi Lora 32 V3 en vez de para la cámara. Está organizado de esta manera y no al revés, con el módulo Heltec a la izquierda, debido a que de hacerlo así el led que se utiliza como flash daría directamente a los dígitos del contador de agua y los deslumbrará, dificultando enormemente el OCR.

En la parte inferior de la caja se puede ver un agujero, este es el que se utiliza para colocar la antena de LoRa, facilitando así las conexiones.

Finalizando con la caja, en la parte exterior derecha de esta se puede ver un saliente, el cual es utilizado para unir esta pieza con la parte que agarra el contador.

La segunda parte es, como se ha mencionado antes, el agarre con el contador. Su función es la de mantener la pieza anterior a cierta distancia del contador y la de someterla a él.

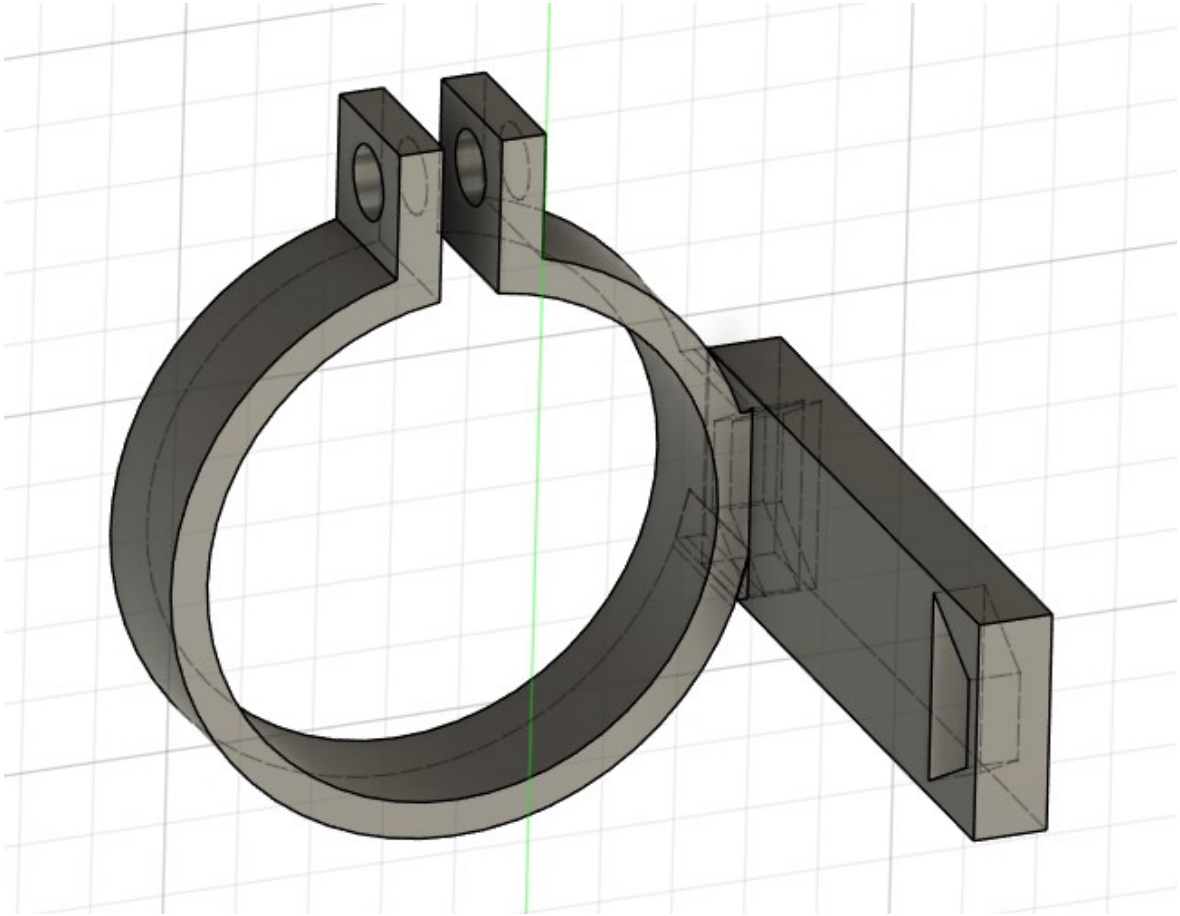

Figura 28: Pieza de agarre al contador 3D

En la Figura 28, se puede ver el diseño 3D de esta pieza, la cual se basa en un cilindro partido, este rodeará el contador de agua y se utiliza la parte rota para ajustar la medida, permitiendo hacerlo a medida de cada contador. Cuenta con un brazo que será el que se agarra a la cámara.

Por último, está el módulo de la batería, cuyo diseño 3D se muestra en la Figura 29. Este es una simple caja que aguanta la batería. Cuenta con un pequeño agujero para conectarla con la pieza principal. Este módulo se agarra a la parte superior de esta pieza, donde está la tapa, permitiendo así minimizar el espacio que ocupa.

Cuenta con unas pequeñas ranuras en la parte que lo une con el módulo principal que permiten que se retire de manera rápida y sencilla, haciendo así que se facilite el cambio de batería.

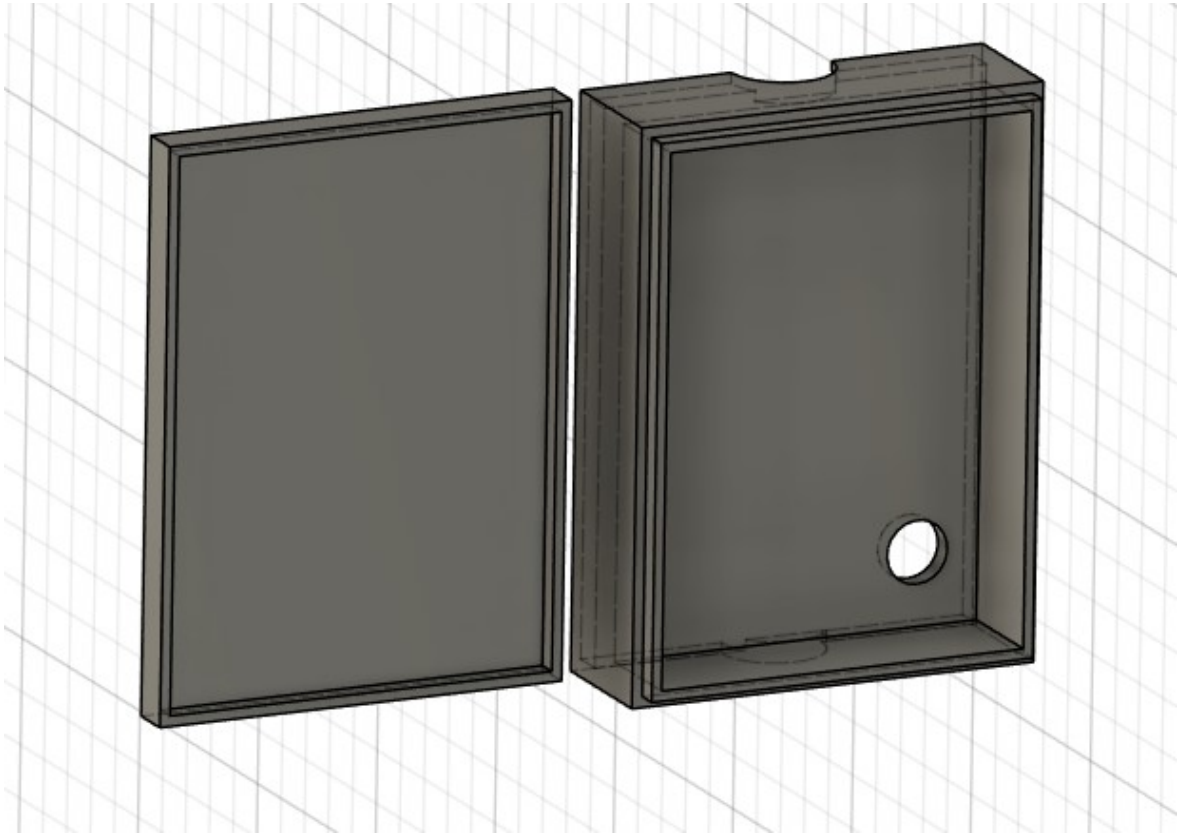

Figura 29: Pieza módulo de la batería 3D

Con estas tres piezas unidas se obtiene un prototipo que cumple en gran medida con las especificaciones requeridas, siendo de un tamaño reducido para minimizar problemas a la hora de utilizarlo, con un módulo de batería intercambiable que permite mantener cargado el dispositivo, una iluminación que no entorpezca la toma de la imagen y una cámara centrada y a una distancia lo suficientemente grande para que se vean los dígitos y lo suficientemente pequeña para que la calidad no de problemas.