# CAL: Reservas e inventario en bibliotecas sin contacto

*CAL: Contactless booking and inventory for libraries*


UNIVERSIDAD COMPLUTENSE
MADRID

Trabajo de Fin de Grado
Curso 2021-2022

Autora
Marina Jiménez Garrido

Directores
José Luis Vázquez-Poletti
Juan Carlos Fabero Jiménez

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

# Agradecimientos

Primero quiero darle las gracias a mi familia y en especial a mis padres. Gracias por alentar mi amor por la lectura, por darme la oportunidad de estudiar la carrera que quería, por apoyarme y por vuestra paciencia durante tantos años. No habría llegado hasta aquí sin vosotros.

A mis mejores amigas Andre y Cris por estar a mi lado, apoyarme, celebrar mis logros conmigo, tener siempre palabras de ánimo cuando las necesitaba e ilusionaros tanto como yo con mi idea sobre libros para este TFG. Espero poder enseñaros cómo funciona en persona pronto.

A Leila y Vic por estar siempre ahí, por todas las risas y por los planes (pasados y futuros).

A ASCII, porque un día allí encontré una segunda familia, aprendí mucho, me llevo muy buenos recuerdos de entonces y por todos los años de secretaria que me ayudaron más a mí que a la asociación.

A mis directores de TFG, Juan Carlos Fabero Jiménez y José Luis Vázquez-Poletti, por aceptar mi idea inicial para ayudarme con los libros de mis estanterías, por vuestra idea de llevarla un paso más allá y por vuestra ayuda desde el principio. Ha sido increíble poder realizar este proyecto que venía queriendo hacer desde hace años.

A todas las personas que se alegraron por mí cuando les dije que mi TFG iba sobre libros. Y a Zero que se alegró el doble cuando supo que tenía nombres basados en Doctor Who.

Y aunque probablemente nunca lo sabrán, a Georgia Byng y Elisabetta Gnone porque sin sus libros yo no sería la misma persona hoy y nunca habría llegado a querer hacer un TFG así.

# Índice

# Índice de ilustraciones

# Índice de tablas

## Resumen

CAL es un sistema de reservas e inventario para bibliotecas sin contacto. Se compone de múltiples dispositivos colocados en los estantes de cada estantería de toda la biblioteca que se comunican con un módulo instalado en el ordenador principal de la biblioteca. Cada libro debe tener una etiqueta RFID y estar registrado en el programa de la biblioteca. Cada dispositivo contiene un microprocesador ESP32, un módulo WiFi, una antena RFID y un programa de control. Periódicamente cada dispositivo lee las etiquetas RFID que tiene a su alcance y las envía al módulo instalado en el ordenador principal de la biblioteca. Los dispositivos se registran incluyendo su localización de estante y estantería en este módulo. El módulo de control actualiza con esta información una base de datos de todos los libros que se encuentran en la biblioteca. De esta forma se mantiene el inventario real de los libros que hay en la biblioteca y su localización real. El sistema de reservas de la biblioteca se conecta a un servicio web instalado en este módulo para poder consultar en cualquier momento si determinado libro existe y dónde se encuentra.



### Palabras clave

Inventario, biblioteca, RFID, ESP32, IoT, Arduino, ISO180006C, IND9011, Visual Studio Code, PlatformIO

## Abstract

CAL is a system for contactless booking and inventory for libraries. It consists of multiple devices placed on the shelves of each bookshelf throughout the library that communicate with a module installed in the main computer of the library. Every book must have a RFID tag and it must be registered in the library program. Each device contains an ESP32 microprocessor, a WiFi module, a RFID antenna and a control program. Each device reads the RFID tags at their reach periodically and they send them to the installed module on the main computer of the library. The devices are registered in this module including their location on the shelf and bookshelf. The control module updates a data base with all the books of the library with this information. This way the actual inventory of the books of the library and their actual location is maintained. The booking system of the library connects to a web service installed in this module to be able to consult if certain book exists and its location at any moment.



### Keywords

Booking, library, RFID, ESP32, IoT, Arduino, ISO180006C, IND9011, Visual Studio Code, PlatformIO

# 1. Introducción

El nombre de CAL [1] viene de Charlotte Abigail Lux de la serie Doctor Who (2005). En el octavo episodio de la cuarta temporada (Silence in the Library / Silencio en la biblioteca) y su segunda parte en el siguiente episodio (Forest of the Dead / El bosque de los muertos) el Doctor y su compañera Donna viajan a la Biblioteca [2] en el siglo 51, un planeta que es la mayor biblioteca del universo. Fue creada por Felman Lux para salvar la mente de su hija Charlotte que, tras una terrible enfermedad, habría fallecido muy joven. Felman trasplantó la consciencia de Charlotte al disco duro de la Biblioteca y creó al Doctor Moon para que mantuviese la Biblioteca y cuidase de la consciencia de Charlotte. La Biblioteca contenía todos los libros que se habían escrito y sirvieron como entretenimiento para Charlotte, que adoraba leer más que nada. Fue así como comenzó CAL: para proteger la identidad de Charlotte, la familia hizo que los ordenadores de la Biblioteca se refiriesen a ella como CAL para definir el corazón de su sistema.

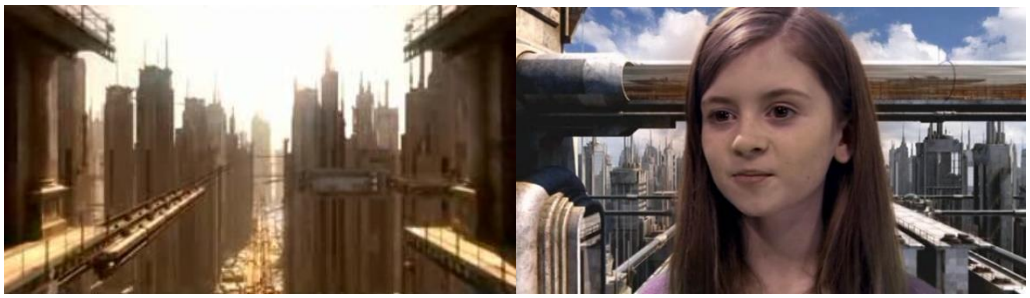

*Ilustración 1 La Biblioteca*[1]

*Ilustración 2 CAL - Charlotte Abigail Lux*[2]

*Ilustración 3 La Biblioteca*[3]

[1] Fuente: https://tardis.fandom.com/wiki/The_Library

[2] Fuente: https://tardis.fandom.com/wiki/Charlotte_Lux

[3] Fuente: https://es.fanpop.com/clubs/doctor-who/images/1325676/title/4x08-silence-library-promo-pics-photo

El planeta tenía una sola luna, una inteligencia artificial conocida como Doctor Moon [3] debido a que se encargaba de controlar que no hubiera ningún virus y estaba diseñado para monitorizar los sistemas de CAL y su bienestar espiritual.

*Ilustración 4 Charlotte y el Doctor Moon*[4]

Para realizar el inventario automático de libros se usará un microcontrolador IoT [4] y una antena RFID. La comunicación con CAL la realizaba sin cables el Doctor Moon. Por este motivo al conjunto microcontrolador y antena se le llama Moon porque se comunica sin cables con Charlotte.

Como en la Biblioteca de Doctor Who, en CAL el corazón del sistema será Charlotte. Charlotte será quien se comunique directamente con cada Moon, quien le mande los comandos que pida el bibliotecario. Desde fuera el bibliotecario utilizará el servicio web que estará en Charlotte, no sabrán de su existencia directamente.

## 1.1. Motivación

La idea de este trabajo viene de una idea para mi biblioteca personal. Mi pasatiempo favorito es leer y tengo 1879 libros en mi colección personal de mi habitación. Hace unos años me surgió la necesidad de poder encontrar un libro en mi colección sin necesidad de ir buscando por zonas. Tengo las estanterías organizadas de varias formas pero es inevitable que a veces no encuentre los libros que busco. Tengo varias estanterías con dos o incluso tres filas de libros y para ver los de la segunda y tercera fila tengo que apartar los libros de la primera y segunda fila y requiere tiempo. Se me ocurrió entonces la idea de un dispositivo que yo pudiera ir pasando delante de las estanterías y que encontrase el libro que estaba buscando.

[4] Fuente: *https://www.express.co.uk/showbiz/tv-radio/1278324/doctor-who-writer-steven-moffat-future-doctor-secret-david-tennant-latest-news-bbc*


*Ilustración 5 Primera fila de mis estanterías*

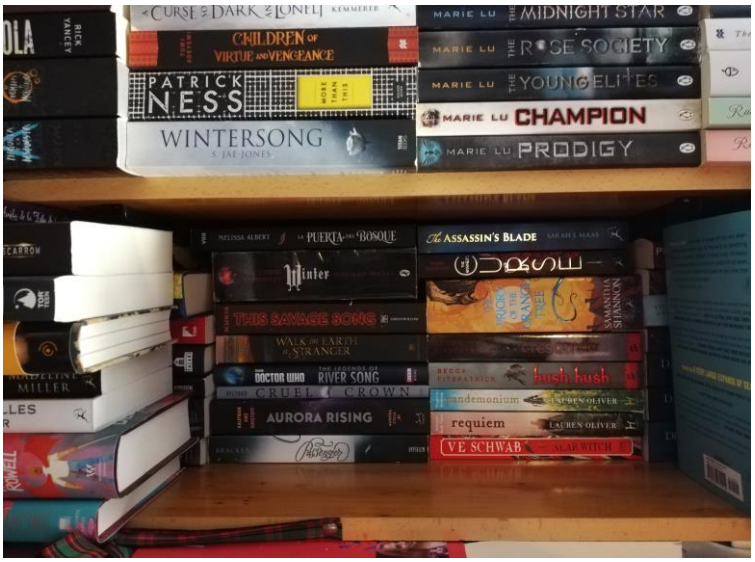


*Ilustración 6 Segunda fila de mis estanterías*

Le presenté esta idea a mis ahora directores del TFG y ellos me ayudaron a llevarlo un paso más allá y crear algo para bibliotecas más grandes. Así nació la idea de este TFG llamada CAL, un sistema de reservas e inventario para bibliotecas sin contacto. Un sistema con varios dispositivos situados a cierta distancia en las estanterías que, utilizando tecnología RFID, se encargarán de leer las etiquetas que tienen los libros cercanos. Con esto será posible hacer un inventario real de la biblioteca y encontrar libros aunque estén en la estantería incorrecta.

## 1.2. Objetivos

El objetivo es crear un dispositivo que pueda realizar el inventario automático en una biblioteca. El dispositivo debe ser pequeño tamaño y autónomo no necesitando para su funcionamiento ningún ordenador o hardware adicional.

Periódicamente y de forma independiente realizará la lectura de todas las etiquetas RFID de los libros que tenga a su alrededor y las enviará a una aplicación centralizada. Para ello, he utilizado un dispositivo IoT modelo ESP32 con una antena RFID y la aplicación que gestiona la lectura y la comunicación con la aplicación centralizada. La aplicación centralizada a su vez será consultada por el sistema de reservas propio de la biblioteca.

# 2. Introduction

The name for CAL [1] comes from Charlotte Abigail Lux from the TV show Doctor Who (2005). In the eighth episode of the fourth season (Silence in the library) and its second part in the next episode (Forest of the Dead) the Doctor and his companion Donna travel to the Library [2] in the 51st century, a planet-sized library that is the biggest library of the universe. It was created by Felman Lux to save the mind of his daughter Charlotte who, after a terrible disease, would've died very young. Felman transplanted Charlotte's conscience to the hard drive of the Library and he created Doctor Moon to maintain the Library and to take care of Charlotte's conscience. The Library contained every book ever written to keep Charlotte entertained, who loved reading more than anything. That's how CAL started: to protect Charlotte's identity, the family had the Library computers refer to her as CAL to define the heart of their system.

The planet had a single moon, an artificial intelligence known as Doctor Moon because it was in charge of controlling that there were no viruses and it was designed to monitor CAL systems and her spiritual well-being.

An IoT [4] microcontroller and an RFID antenna will be used to carry out the automatic inventory of books. The communication between Doctor Moon and Charlotte was wireless. For this reason the microcontroller and antenna assembly is called Moon because it communicates wirelessly with Charlotte.

Just as in the Library in Doctor Who, in CAL the heart of the system will be Charlotte. Charlotte will be the one who communicates directly with every Moon, the one who will send them the commands that the librarian asks for. From the outside the librarian will use the web service that will be in Charlotte, but the librarian won't know of Charlotte's existence.

## 2.1. Motivation

The idea for this project comes from an idea for my own personal library. My favourite hobby is reading and I own 1879 books in my personal collection in my bedroom. A few years ago I had the need to be able to find a book in my collection without having to search by areas. I organize my bookshelves in different methods but it's inevitable that I won't find the books I'm searching for sometimes. I have several bookshelves with two or even three rows of books and if I want to see the books on the second or third row I need to put away the books of the first and second row and that takes time. I came up with the idea of a device that I could pass in front of the shelves and find the book I was looking for.

I presented this idea to my now directors of this project and they helped me take it one step ahead and create something for bigger libraries. That's how the idea for this project called CAL was born, a system of contactless booking and inventory for libraries. A system with several devices placed at a distance in the bookshelves that, using RFID technology, will be in charge of reading the tags of the nearest books. Thanks to this it will be possible to make a real inventory of the library and to find the books even if they're in the wrong shelf.

## 2.2. Objectives

The objective is to create a device that can make the automatic inventory in a library. The device must be small and self-contained so that it won't need any computer or additional hardware to its functioning.

Periodically and independently, it will read all the RFID tags of the books around it and it will send them to a centralized application. To do this, I've used an ESP32 model IoT device with an RFID antenna and the application that that manages the reading and communication with the centralized application. The centralized application will be consulted by the library's own reservation system.

# 3. Estado del arte

En este proyecto se usa un microprocesador para controlar la antena RFID y las comunicaciones. Utiliza dos diodos LED, uno de encendido y otro de lectura. Las etiquetas se leen mediante una antena RFID que va conectada al microcontrolador mediante una interfaz RS-232. Las comunicaciones de Moon con Charlotte se hacen mediante WiFi.

## 3.1. Microprocesador

Existen multitud de tipos de microcontroladores utilizados en IoT [4]. El rango va desde microcontroladores de 8 bits y unas decenas de bytes de memoria hasta microcontroladores de 32 bits y varios megabytes de memoria. Para facilitar su uso se encuentran integrados en placas de desarrollo que contienen un regulador de tensión, conector USB, algún LED, etc…

Una de las placas más utilizadas es Arduino en una de sus muchas variantes. Esta placa contiene los elementos básicos y se le pueden añadir otras placas con WiFi, Ethernet, Bluetooth, etc…

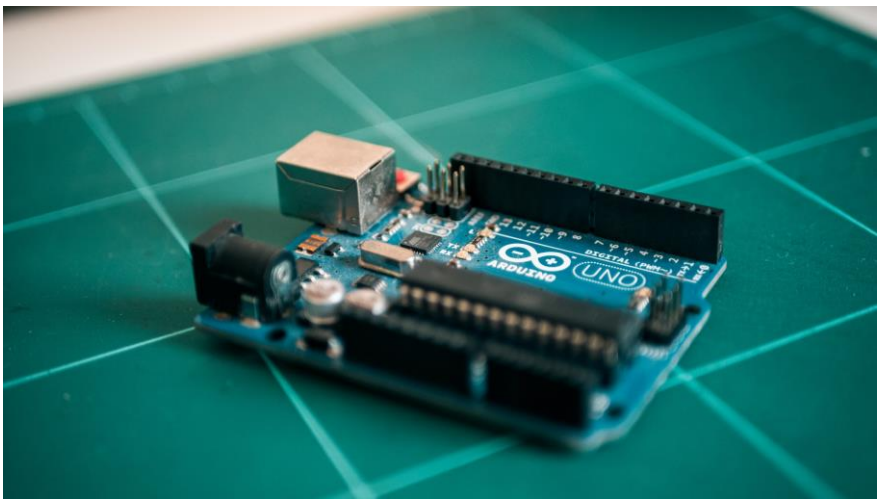


*Ilustración 7 Arduino*

Otra familia ampliamente utilizada es la basada en los microcontroladores ESP8266 y ESP32. El ESP8266 es un chip de bajo costo WiFi con un stack TCP/IP completo y un microcontrolador, fabricado por Espressif, una empresa afincada en Shanghái, China. ESP32 es la denominación de una familia de chips SoC de bajo costo y consumo de energía, con tecnología WiFi y Bluetooth de modo dual integrada.

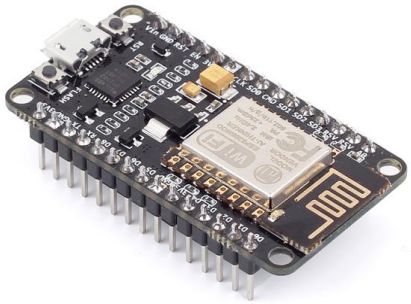

*Ilustración 8 Microcontrolador con ESP8266*

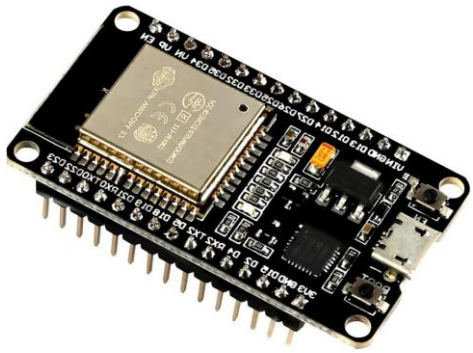

*Ilustración 9 Microcontrolador con ESP32*

## 3.2. LED

También se van a utilizar dos LED, uno para indicar que está encendido y otro cuando está leyendo etiquetas. El led de encendido parpadeará cuando haya errores. Un LED es un diodo emisor de luz. Un diodo se forma por la unión de dos materiales semiconductores con dopados distintos. Los diodos tienen polaridad lo que significa que solo dejan pasar la corriente en un sentido así que hay que conectar correctamente la tensión al dispositivo.

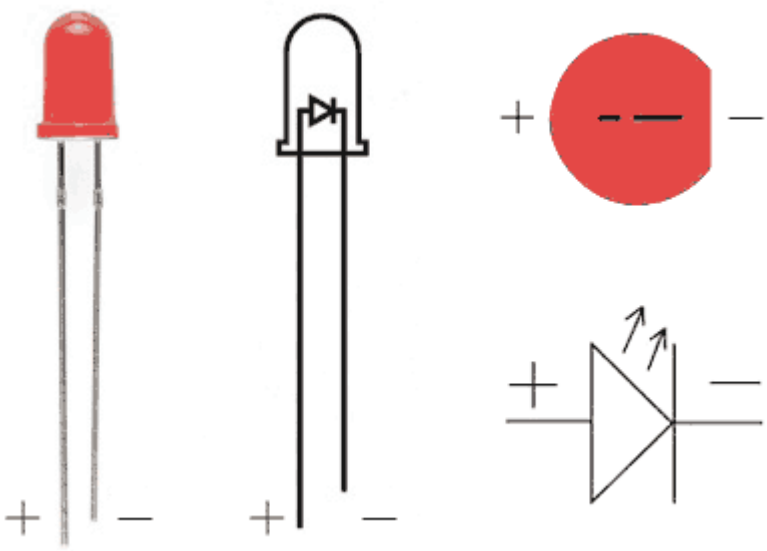

*Ilustración 10 Esquema de un LED*

## 3.3. Tecnología RFID

RFID [5] (identificación por radiofrecuencia, del inglés Radio Frequency Indentification) es una forma de comunicación inalámbrica entre un lector y un emisor. Los emisores son dispositivos denominados etiquetas, tarjetas o transpondedores RFID (un transpondedor es un dispositivo que emite y recibe). La transmisión de datos se realiza mediante ondas de radio, lo que tiene la ventaja de que no requiere visión directa entre emisor y receptor.

Su funcionamiento consiste en que el lector envía una señal continua dentro de un radio de alcance y al encontrar un emisor, éste le envía la información que tiene almacenada sobre el objeto en el que esté (dependiendo de cómo esté programado).

Las etiquetas RFID [6] contienen chips con la información de identificación de la etiqueta y también tienen antenas para permitir la transmisión de esa información almacenada en el chip. El chip contiene memoria interna con una capacidad concreta dependiendo del modelo. Existen varios tipos de memoria:

- Sólo lectura: Al fabricar la etiqueta se le asigna un código de identificación único que no se puede cambiar.
- De lectura y escritura: Se puede modificar la información de identificación.
- Anticolisión: se trata de etiquetas especiales que permiten que un lector identifique varias al mismo tiempo (habitualmente las etiquetas deben entrar una a una en la zona de cobertura del lector).

El lector de RFID [7] contiene una antena que transmite señales periódicamente para ver si hay alguna etiqueta cercana. Cuando capta una señal de una etiqueta (la cual contiene la información de identificación de esta), extrae la información y se la pasa al subsistema de procesamiento de datos.

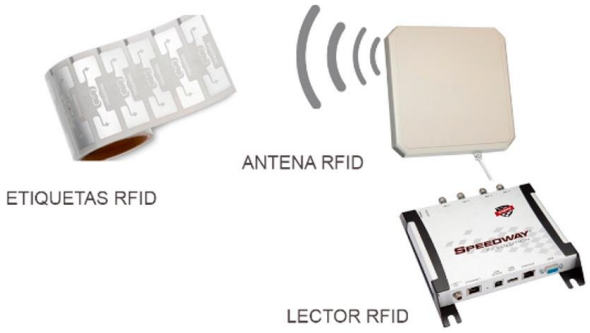


*Ilustración 11 Sistema RFID*

Hay dos tipos de etiquetas: las etiquetas pasivas y las etiquetas activas. Las etiquetas pasivas no necesitan alimentación eléctrica interna y se activan cuando un lector cercano les suministra energía. Las etiquetas activas requieren alimentación.


*Ilustración 12 Etiquetas RFID pasivas*

Al no tener alimentación interna, las etiquetas pasivas necesitan recibir la señal de un lector para inducir una corriente eléctrica pequeña para su funcionamiento y así generar y transmitir una respuesta. La mayoría de etiquetas RFID son pasivas porque son más baratas de fabricar y no necesitan batería. Además, se pueden montar en diversos soportes, como una pegatina, una tarjeta o un sobre de papel. Ejemplos de uso de etiquetas RFID pasivas son las ya populares tarjetas de pago sin contacto, que la mayoría de bancos expiden para sus clientes. Las etiquetas

pasivas tienen una distancia de lectura de entre 10 milímetros y 6 metros dependiendo de la antena de la etiqueta y de la potencia y frecuencia del lector.

Las etiquetas activas tienen su propia alimentación que utilizan para dar corriente a sus circuitos integrados y propagar su señal al lector. Gracias a esto son más fiables y eficientes en entornos complicados y a distancias mayores. Aunque suelen ser más grandes, más caras y su vida útil suele ser más corta que las etiquetas pasivas. Las tecnologías RFID activas están básicamente circunscritas a dos campos, la lectura de etiquetas a gran velocidad (etiquetas desplazándose a velocidades de hasta 150 km/hora) y la localización por triangulación radioeléctrica.

Existen varias diferencias entre las etiquetas RFID y los códigos de barras tradicionales. Las principales ventajas de los códigos de barras son su estandarización en multitud de sectores, su sencillez de uso y su nivel de precisión. Las desventajas son que la lectura requiere de la intervención humana y contacto directo del lector con el código lo que retarda el proceso, la capacidad de almacenamiento es muy limitada, no se pueden modificar una vez han sido impresos y, aunque los códigos pueden ser impresos con sistemas sencillos, se pueden deteriorar más fácilmente dificultando su lectura.

La mayor ventaja de la tecnología RFID frente a los códigos de barras es su versatilidad. La capacidad de almacenamiento de datos de las etiquetas RFID es mucho mayor, la lectura es mucho más rápida ya que existe la posibilidad de leer varias etiquetas a la vez, no necesitan contacto visual directo (se pueden leer a través de otros materiales) y la información de las etiquetas RFID puede reescribirse cuando sea necesario. También es más seguro que los códigos de barras porque es más difícil de copiar o duplicar. Como desventaja, es menos económico que los códigos de barras al implicar un desarrollo tecnológico mayor.

Los sistemas RFID se clasifican dependiendo del rango de frecuencias que usan. Existen cuatro tipos de sistemas: de frecuencia baja (LF: 125 o 134.2 kHz) con un rango de lectura de menos de 10cm; de alta frecuencia (HF: 13.56 MHz) con un rango de lectura entre 10cm y 1m; de frecuencia ultra elevada (UHF: 868 a 956 MHz) que puede alcanzar más de 12m con transpondedores pasivos y hasta 100m con transpondedores activos; y de microondas (2.45 GHz) que puede alcanzar hasta 3m con transpondedores pasivos y hasta 300m con transpondedores activos.


*Ilustración 13 Antena RFID*

## 3.4. Comunicaciones

La comunicación entre Moon y CAL se puede realizar mediante cable Ethernet o WiFi. Para usar cable Ethernet habría que añadir un módulo adicional que aumentaría el precio y el tamaño de Moon. Se utilizará comunicación WiFi porque ya está integrada en Moon.

### 3.4.1. WiFi

El WiFi [8] es una tecnología que permite la interconexión inalámbrica de dispositivos electrónicos. Los dispositivos habilitados con WiFi pueden conectarse entre sí o a Internet a través de un punto de acceso de red inalámbrica.

Para garantizar la seguridad de las redes WiFi existen varios protocolos de cifrado de datos que se encargan de codificar la información transmitida en la red para proteger su confidencialidad. Algunos de ellos son:

- WEP: cifra los datos de su red para que solo el destinatario pueda acceder a ellos. Utiliza una “clave” para codificar los datos antes de enviarlos. No es muy recomendable porque se puede conseguir sacar la clave y tener acceso a los datos debido a vulnerabilidades intrínsecas del protocolo.
- WPA: tiene mejoras sobre WEP como la generación dinámica de la clave de acceso.
- WPA2: es una mejora de WPA que utiliza el algoritmo de cifrado AES (*Advanced Encryption Standard*).
- WPA3: utiliza cifrado de 128 bits en modo WPA3-Personal (192 bits en WPA3-Enterprise) y confidencialidad de reenvío. El estándar WPA3 también reemplaza el intercambio de claves pre-compartidas con la autenticación simultánea de iguales, lo que resulta en un intercambio inicial de claves más seguro en modo personal. Es el protocolo más seguro hasta ahora.

Existen varios dispositivos WiFi:

- Los puntos de acceso generan una red WiFi a la que se pueden conectar otros dispositivos.
- Los repetidores inalámbricos son dispositivos que se usan para extender la cobertura de una red inalámbrica. Se conectan a una red con una señal débil y crean una señal más fuerte para que se puedan conectar equipos que estén a su alcance.
- Los encaminadores inalámbricos son dispositivos diseñados para redes pequeñas. Se usa para equipos que no tienen WiFi, se conectan vía cable al encaminador y el encaminador actúa de cliente WiFi.

### 3.4.2. RS-232

RS-232 [9] (Recommended Standard 232, en español: "Estándar Recomendado 232") es una interfaz que designa una norma para el intercambio de datos binarios serie entre un DTE (Equipo Terminal de Datos) y un DCE (Equipo de Comunicación de Datos). A veces se quiere conectar otro tipo de equipos como ordenadores. En el caso de interconexión entre los mismos, la conexión será de un DTE con otro DTE. Para estos casos se utiliza una conexión entre los otros dos DTE sin usar módem llamado módem nulo. El RS-232 consiste en un conector de 25 pines y también hay de 9 pines.

*Ilustración 14 Conector DB9 para RS-232*

Las UART o U(S)ART (Transmisor y Receptor Asíncrono Universal) se diseñaron para convertir las señales que maneja la CPU y transmitirlas al exterior. Las UART deben resolver problemas tales como la conversión de tensiones internas del DCE con respecto al DTE, gobernar las señales de control, y realizar la transformación desde el bus de datos de señales en paralelo a serie y viceversa. Cuando se necesita conectar un microcontrolador (con señales típicamente entre 3,3 y 5 V) con un puerto RS-232 estándar, se utiliza un driver de línea, típicamente un MAX232 o compatible que permite obtener la señal bipolar (típicamente alrededor de +/- 6V) requerida por el estándar.

La comunicación serie asíncrona es un tipo de comunicación serie en la que las interfaces de los extremos no están continuamente sincronizados por una señal de reloj común. En lugar de

una señal de sincronización común, los flujos de datos contienen información de sincronización en la forma de señales de inicio y parada, antes y después de cada unidad de transmisión, respectivamente. La señal de inicio prepara el receptor para la llegada de datos y la señal de parada reinicia su estado para permitir desencadenar una nueva secuencia.

Antes de que funcione la señalización, se deben configurar los parámetros de señalización del emisor y receptor:

- Operación full o half-duplex.
- El número de bits por carácter, actualmente son casi siempre caracteres de 8 bits, pero históricamente algunos transmisores han utilizado un código de carácter de 5 bits, de 6 bits o de 7 bits.
- La velocidad o bits por segundos de la línea.
- Cuándo se usa o no una paridad.
- Paridad par o impar, si se usa.
- El número de bits de parada.

# 4. Requisitos

## 4.1. Requisitos funcionales

**RF1 Moon deberá tener una configuración persistente**

La configuración habrá que guardarla en memoria no volátil para que no se pierda en caso de un fallo de alimentación o un reinicio. En la configuración persistente se guardará al menos el nombre y la contraseña de la red WiFi, el intervalo entre lecturas y otra información adicional que se considere pertinente.

**RF2 Moon tendrá dos estados: configuración y ejecución**

En el estado de configuración se podrán modificar los parámetros de Moon. El estado normal de funcionamiento será el de ejecución.

**RF3 Primera ejecución**

La primera vez que se inicia Moon no tiene datos en la memoria persistente y se encontrará en modo configuración.

**RF4 Reinicio y configuración**

Cuando se inicia Moon una vez que esté configurado pasará a modo ejecución. Otra forma de cambiar la configuración es presionar el pulsador de Moon durante al menos 10 segundos.

**RF5 Modo configuración**

El instalador podrá configurar o modificar a Moon.

**RF6 Modo ejecución**

En modo ejecución Moon quedará a la espera de recibir un comando de Charlotte o ejecutar la siguiente lectura de etiquetas.

**RF7 Ejecución de comandos**

Moon ejecutará los comandos que reciba de Charlotte.

**RF8 Comandos que soporta Moon**

Los comandos soportados serán:

- Ejecución de un barrido manual
- Modificación del tiempo entre lecturas

**RF9 Ejecución de una lectura**

Moon deberá leer etiquetas RFID.

**RF10 Envío de lecturas a Charlotte**

Cuando Moon realiza una lectura le enviará a Charlotte el identificador de Moon y los tags de todas las etiquetas leídas.

**RF11 Dar de alta Moon en Charlotte**

Tras la configuración hay que dar de alta cada Moon en Charlotte y definir su identificador y su localización.

**RF12 Barrido automático**

Cada Moon realizará un barrido automáticamente con el intervalo de tiempo definido.

**RF13 Barrido manual**

El bibliotecario podrá ejecutar un barrido en un Moon.

**RF14 Servicio web para el módulo de reservas**

Charlotte tendrá un servicio web para que el sistema de reservas de la biblioteca consulte su base de datos con el inventario y verifique que hay copias de determinado libro y su localización real.

**RF15 Base de datos**

Charlotte contiene una base de datos con el inventario actualizado de la biblioteca. Se actualizará automáticamente con los datos recibidos de los módulos Moon y se consultará por el servicio web.

## 4.2. Requisitos no funcionales

**RNF1 Tipo etiquetas RFID**

Moon debe ser capaz de leer etiquetas RFID tipo UHF EPC Gen2 –ISO180006C

**RNF2 Lectura múltiple**

Debe ser capaz de leer 50 señales RFID simultáneamente.

**RNF3 Alcance**

El alcance mínimo de lectura será al menos de un metro.

**RNF4 Botón configuración**

Tendrá un botón para configurar.

**RNF5 LED encendido**

Tendrá un LED para indicar que está encendido.

**RNF6 LED lectura**

Tendrá un LED que se encenderá mientras realiza una lectura.

# 5. Arquitectura del sistema

CAL se compone del módulo Charlotte que se ejecuta en el sistema de reservas de la biblioteca y múltiples dispositivos Moon distribuidos por las estanterías. La comunicación entre el sistema de reservas de la biblioteca y Charlotte se realiza mediante un servicio web y la comunicación entre Charlotte y los dispositivos Moon se realiza mediante WiFi. Cada dispositivo Moon contiene una antena RFID que se utiliza para leer las etiquetas de todos los libros a su alcance.

El esquema de la arquitectura se muestra en la siguiente figura:

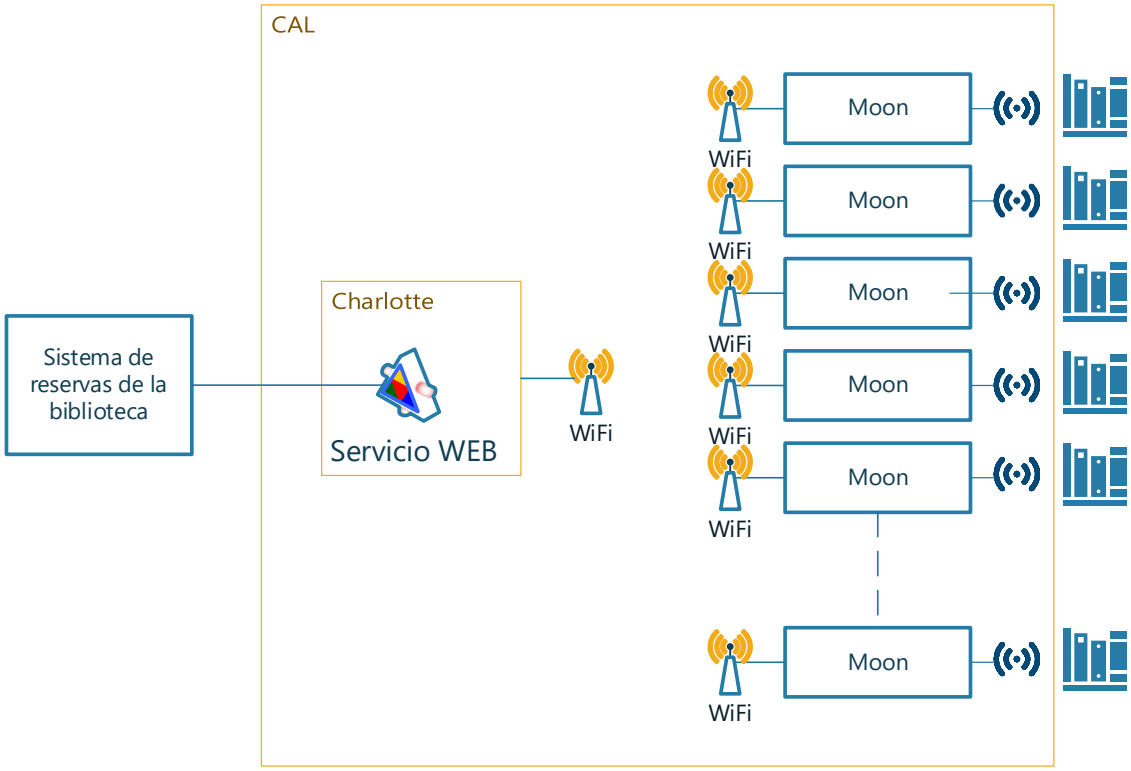


*Ilustración 15 Diagrama de arquitectura*

En la siguiente figura se muestra un esquema más detallado de la arquitectura:

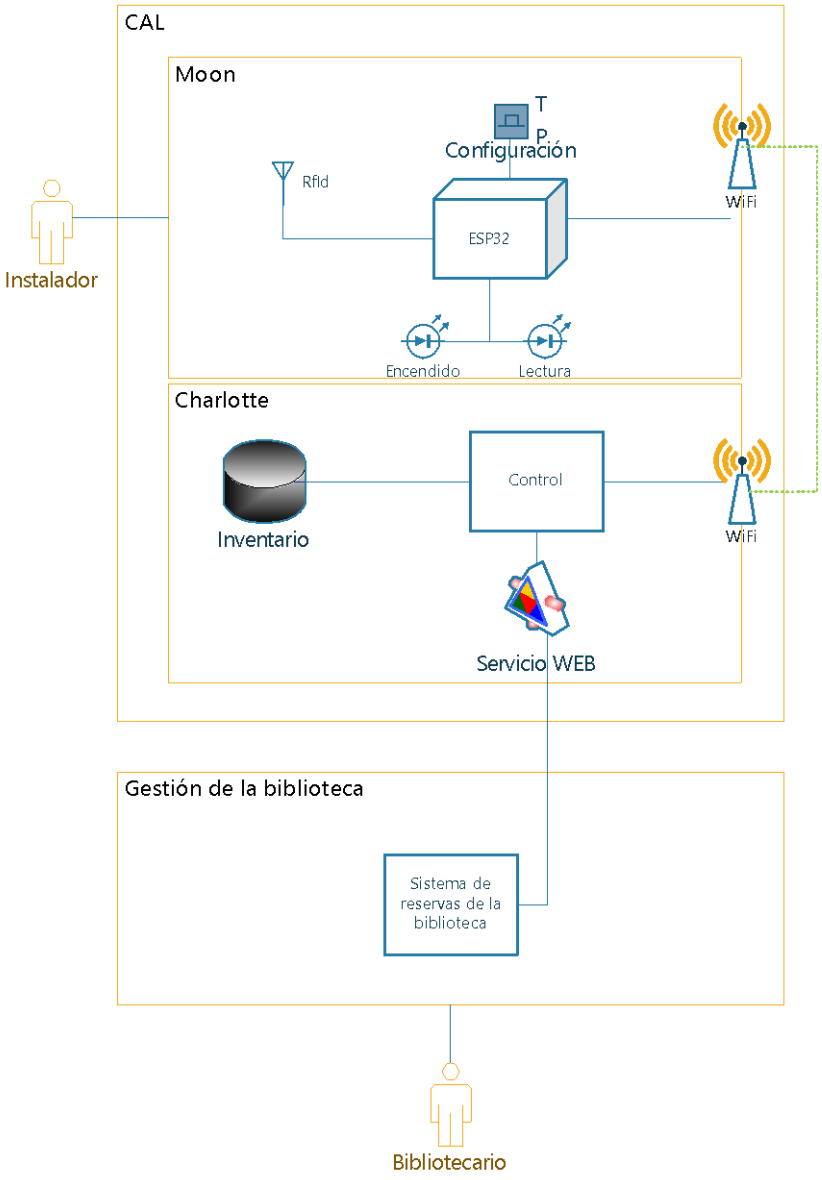


*Ilustración 16 Diagrama de arquitectura detallado*

Moon está controlado por un microprocesador ESP32 que tiene conectados un LED de encendido (que parpadea si hay errores), otro que se enciende durante la lectura de etiquetas, un pulsador de configuración, un módulo WiFi y una antena RFID externa conectada mediante RS-232. Al pulsar el botón de configuración, el instalador configura los valores de Moon mediante una conexión WiFi al dispositivo. Moon se conecta a Charlotte mediante WiFi para enviarle los resultados de la lectura de etiquetas. Charlotte contiene la base de datos con el inventario actualizado en tiempo real de todos los libros de la biblioteca y su localización física. Se comunica con Moon a través de WiFi para mandarle algún comando y recibir las lecturas. Implementa un servicio web que será utilizado por el sistema de gestión de reservas de la biblioteca. El sistema de gestión de reservas de la biblioteca se conecta al servicio web de Charlotte cuando necesita la localización física de algún libro.

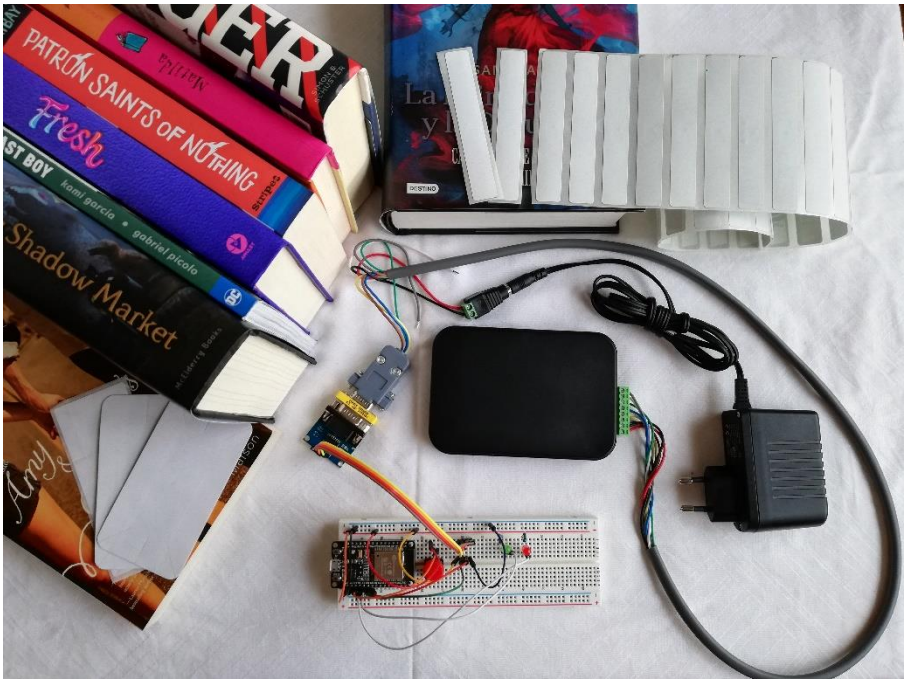


*Ilustración 17 Prototipo Moon y etiquetas*

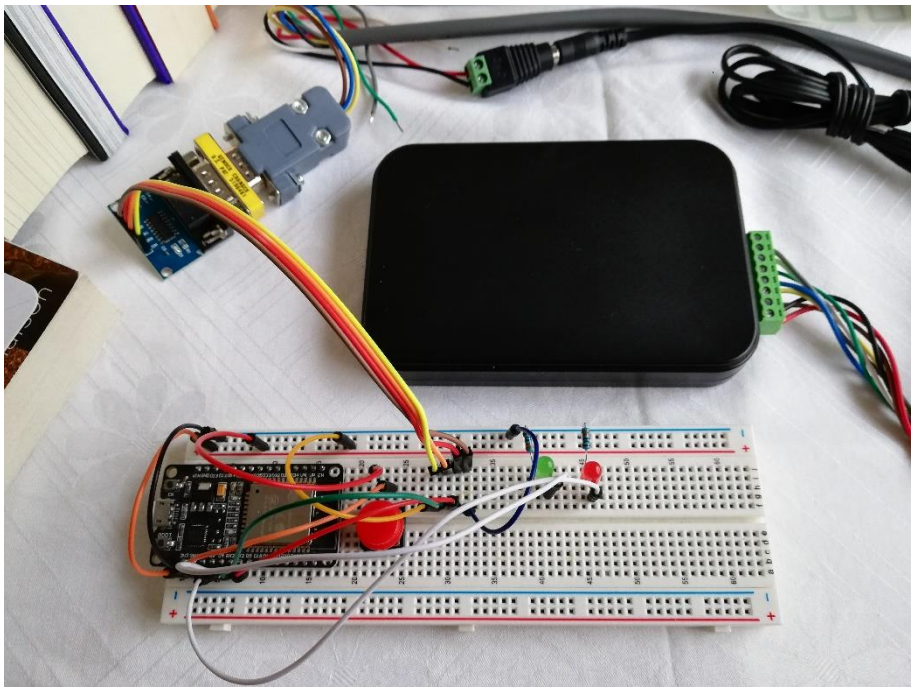

*Ilustración 18 Prototipo Moon*

## 5.1. Etiquetas

Los libros deben tener una etiqueta RFID tipo UHF EPC Gen2 –ISO180006C para que puedan ser identificados por Moon.


*Ilustración 19 Etiquetas RFID UHF EPC Gen2 –ISO180006C*

## 5.2. Antena

La antena utilizada es el modelo IND9011 [10].

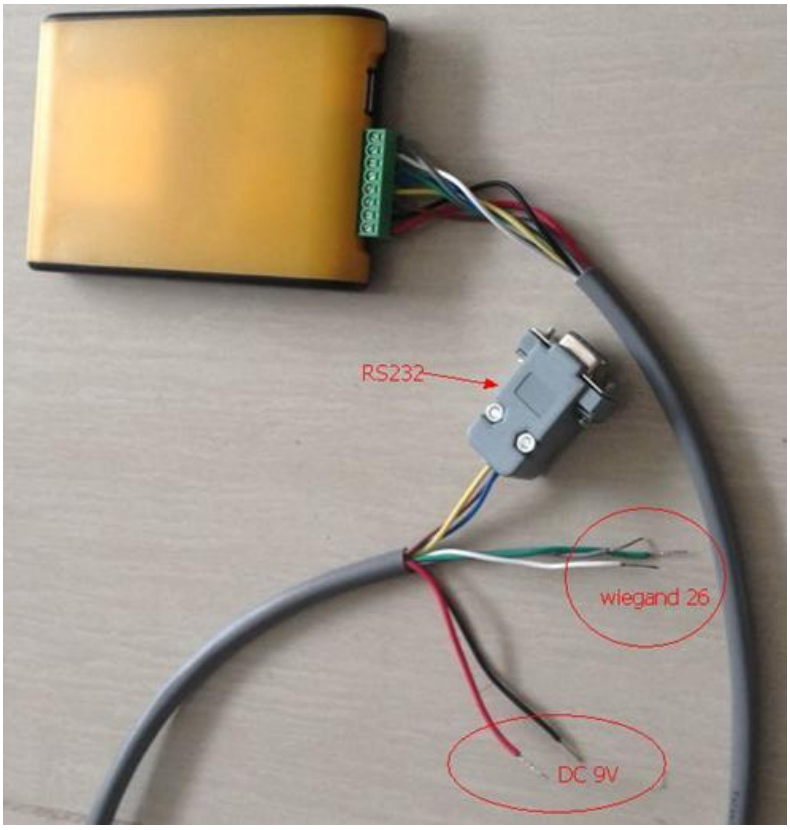


*Ilustración 20 Antena RFID IND9011*

La conexión con Moon se realiza mediante el puerto RS-232. La antena se ha comprado a un proveedor minorista, de entre las que se han encontrado esta antena era la mejor en relación calidad precio. Esta antena tiene un alcance máximo aproximado de un metro sin obstáculos. Al utilizarlo para la lectura de libros el alcance se reduce por lo que no es la antena más idónea para este proyecto. Para una instalación real habría que buscar una antena con mayor alcance.

## 5.3. Microcontrolador

No se selecciona una placa Arduino porque las placas más sencillas no disponen de WiFi.

Se selecciona el ESP32 [11] en lugar del ESP8266 porque tiene mayor conectividad ya que dispone de Bluetooth, permite ampliar la funcionalidad en un futuro y la diferencia de precio con el ESP8266 es muy pequeña.

Sus características principales traducidas del manual de referencia del fabricante [12] son:

- Espacio de direcciones
  - Espacio de direcciones de 4 GB (32 bits) para el bus de datos y el bus de instrucciones
  - Espacio de direcciones de memoria integrada de 1296 KB
  - Espacio de direcciones de memoria externa de 19704 KB
  - Espacio de direcciones entrada/salida de 512 KB
  - Se puede acceder a algunas regiones de la memoria integrada y externa mediante el bus de datos o el bus de instrucciones
  - Espacio de direcciones DMA de 328 KB
- Memoria integrada.
  - ROM interna de 448 KB
  - SRAM interna de 520 KB
- Memoria externa
  - La memoria SPI fuera del chip se puede asignar al espacio de direcciones disponible como memoria externa. Partes de la memoria integrada se pueden utilizar como caché transparente para esta memoria externa.
  - Admite hasta 16 MB de Flash SPI sin chip.
  - Admite hasta 8 MB de SPI SRAM sin chip.
- Periféricos
  - 41 periféricos
- DMA
  - 13 módulos son capaces de operar con DMA

# 6. Casos de uso

A continuación se detallan los actores y las tablas de los casos de uso en las que se muestra la secuencia de pasos necesaria para realizar la tarea especificada así como las posibles variantes que puedan darse.

## 6.1. Actores

De la lista de requisitos se obtiene la siguiente lista de actores:

- Bibliotecario/a. Accede a CAL a través de un servicio web para consultar el inventario, realizar una lectura manual o configurar CAL.
- Moon. El software de control y el lector RFID.
- Servicio web. Sirve de interfaz entre CAL y el bibliotecario.
- Charlotte. Se comunica con Moon y contiene el servicio web.
- Instalador. Configura el módulo Moon en modo fábrica.

## 6.2. CU01 Modo configuración

*Tabla 1 CU01 Modo configuración*

| Caso de uso | CU01 Modo configuración |
|---|---|
| Descripción | Moon creará una red WiFi en modo punto de acceso con un nombre y contraseña fijos. Un instalador tendrá que configurar a Moon con los datos de la red WiFi y el identificador de Moon en un servidor web HTTP que haya creado Moon. Si hay errores en los datos introducidos Moon enviará una página con los errores. Si están bien Moon modificará la configuración y pasará a modo ejecución.<br><br>Si ya existía una configuración previa, se iniciará un temporizador de dos minutos para que el instalador envíe los nuevos datos. Si el tiempo expira y no los envía, desconectará la red WiFi, el servidor web y terminará el modo configuración. |
| Requisito relacionado | RF1, RF2, RF3, RF4, RF5 |
| Actores | Instalador, Moon |

| | |
|---|---|
| Precondición | Pulsar el botón de Moon durante 10 segundos o apagar y encender Moon |
| Postcondición | Se ha modificado la configuración |
| Escenario básico | 1. Moon crea una red WiFi en modo punto de acceso con un nombre y contraseña fijos<br>2. Moon creará un servidor web HTTP<br>3. El instalador se conecta a la red WiFi<br>4. El instalador abre un navegador web<br>5. El instalador introduce la URL de Moon<br>6. Moon envía al navegador la página de configuración<br>7. El instalador introduce los nuevos valores y los envía<br>8. Moon verifica los valores<br>9. Moon modifica la configuración<br>10. Moon termina el modo configuración<br>11. Moon cierra el servidor web<br>12. Moon cierra la red WiFi<br>13. Moon pasa a modo ejecución |
| Escenario alternativo | 2. Si existía una configuración previa Moon inicia un temporizador de dos minutos, cuando expire Moon pasará a modo ejecución. Se reiniciará cada vez que reciba datos del instalador.<br>6. Si ya había valores en la memoria persistente Moon los envía en la página web<br>8. Si hay errores Moon le enviará una página con los errores encontrados hasta que reciba los datos sin errores |
| Observaciones | |

## 6.3. CU02 Modo ejecución

*Tabla 2 CU02 Modo ejecución*

| | |
|---|---|
| Caso de uso | CU02 Modo ejecución |

| | |
|---|---|
| Descripción | Moon leerá la configuración, se conectará a la red WiFi, creará un servidor de comandos para recibir comandos de Charlotte y quedará a la espera de esos comandos o de ejecutar la siguiente lectura de etiquetas. |
| Requisito relacionado | RF6, RF7, RF12, RF13 |
| Actores | Moon, Charlotte |
| Precondición | Que Moon esté configurado |
| Postcondición | Moon se queda a la espera de un comando de Charlotte o de ejecutar la siguiente lectura de etiquetas |
| Escenario básico | 1. Moon lee la configuración de la memoria EEPROM<br>2. Moon se conecta a la red WiFi<br>3. Moon crea un nombre de dispositivo y lo registra en la red<br>4. Moon inicia, si no está iniciado, un servidor de comandos en un puerto predefinido para recibir comandos de Charlotte<br>5. Moon queda a la espera de:<br>a. Recibir un comando de Charlotte<br>i. Moon va al caso de uso CU04<br>b. Ejecutar la siguiente lectura de etiquetas<br>i. Moon va al caso de uso CU05<br>c. La pulsación del botón de configuración durante 10 segundos<br>i. Moon va al caso de uso CU01<br>6. Moon va al paso 4 |
| Escenario alternativo | 5. Moon conectará la red WiFi si no está conectada |
| Observaciones | |

## 6.4. CU03 Ejecutar un comando

*Tabla 3 CU03 Ejecutar un comando*

| | |
|---|---|
| Caso de uso | CU03 Ejecutar un comando |
| Descripción | Moon ejecutará los comandos que reciba de Charlotte. |
| Requisito relacionado | RF7, RF8 |
| Actores | Moon, Charlotte |
| Precondición | CU02 Modo ejecución |
| Postcondición | Se ha ejecutado el comando |
| Escenario básico | 1. Charlotte se conecta al servidor de comandos de Moon y le manda un comando<br>2. Moon lee el comando<br>3. Moon notifica la aceptación del comando a Charlotte<br>4. Moon ejecuta el comando:<br>a. Lectura de etiquetas<br>b. Modificar el intervalo de lecturas |
| Escenario alternativo | 3. Si Moon no reconoce el comando le responde con un error |
| Observaciones | |

## 6.5. CU04 Lectura de etiquetas

*Tabla 4 CU04 Lectura de etiquetas*

| | |
|---|---|
| Caso de uso | CU04 Lectura de etiquetas |
| Descripción | Moon ejecutará una lectura de etiquetas |
| Requisito relacionado | RF9, RF10, RF13, RF14 |
| Actores | Moon, Charlotte |
| Precondición | CU02: Modo ejecución |

| | |
|---|---|
| Postcondición | Se envía la lectura de etiquetas a Charlotte |
| Escenario básico | 1. Moon enciende el LED de lectura<br>2. Moon inicia la conexión con el lector RFID<br>3. Moon le envía un comando al lector para el barrido de etiquetas<br>4. El lector ejecuta la lectura<br>5. Moon guarda los datos en la memoria<br>6. Moon termina la conexión con el lector RFID<br>7. Moon apaga el LED de lectura<br>8. Moon se conecta a Charlotte<br>9. Moon le envía la lectura a Charlotte<br>10. Moon cierra la comunicación con Charlotte |
| Escenario alternativo | 3. Si no puede conectar con el lector el LED de encendido parpadea varias veces.<br>8. Moon conectará la red WiFi si no está conectada |
| Observaciones | La lectura contendrá el identificador de Moon y los tags de todas las etiquetas leídas. |

## 6.6. CU05 Dar de alta Moon

*Tabla 5 CU05 Dar de alta Moon*

| | |
|---|---|
| Caso de uso | CU05 Dar de alta Moon |
| Descripción | Tras la configuración hay que dar de alta cada Moon en Charlotte y definir su localización |
| Requisito relacionado | RF11 |
| Actores | Charlotte, Bibliotecario |
| Precondición | CU01: Modo configuración |
| Postcondición | Se ha dado de alta a cada Moon en Charlotte |
| Escenario básico | 1. El bibliotecario selecciona “dar de alta Moon” |

| | |
|---|---|
| | 2. El bibliotecario introduce el identificador y la localización de Moon<br>3. Cuando estén los datos:<br>a. Si hay que dar de alta más Moon pulsa “añadir”<br>b. Si no hay más Moon pulsa “terminar” |
| Escenario alternativo | 2. Hay errores en los datos y Charlotte muestra un mensaje |
| Observaciones | |

## 6.7. CU06 Hacer una reserva

*Tabla 6 CU06 Hacer una reserva*

| | |
|---|---|
| Caso de uso | CU06 Hacer una reserva |
| Descripción | El sistema de reservas de la biblioteca comprobará mediante la llamada a un servicio web de CAL si el libro está físicamente en su lugar o en otra estantería y el número de copias sin necesidad de desplazamiento |
| Requisito relacionado | RF14, RF15 |
| Actores | Bibliotecario, Servicio web |
| Precondición | La base de datos está actualizada |
| Postcondición | Se ha reservado un libro |
| Escenario básico | 1. El sistema de reservas de la biblioteca llama al servicio web con la referencia del libro a buscar<br>2. El servicio web consulta la base de datos<br>3. El servicio web devuelve la ubicación real del libro y el número de copias disponibles |
| Escenario alternativo | 3. Si no quedan copias disponibles el servicio web devuelve 0 copias disponibles |
| Observaciones | |

## 6.8. Matriz de relaciones casos de uso – requisitos

Cada requisito funcional debe estar implementado al menos por un caso de uso. En la matriz de relaciones entre los casos de uso y los requisitos se puede comprobar que se cumple.

En las filas se muestran los casos de uso y en las columnas se muestran los requisitos funcionales. En la intersección de los casos de uso y los requisitos se muestra una flecha que indica qué caso de uso implementa cada requisito funcional. Se observa que se cumple que cada requisito está implementado al menos por un caso de uso.

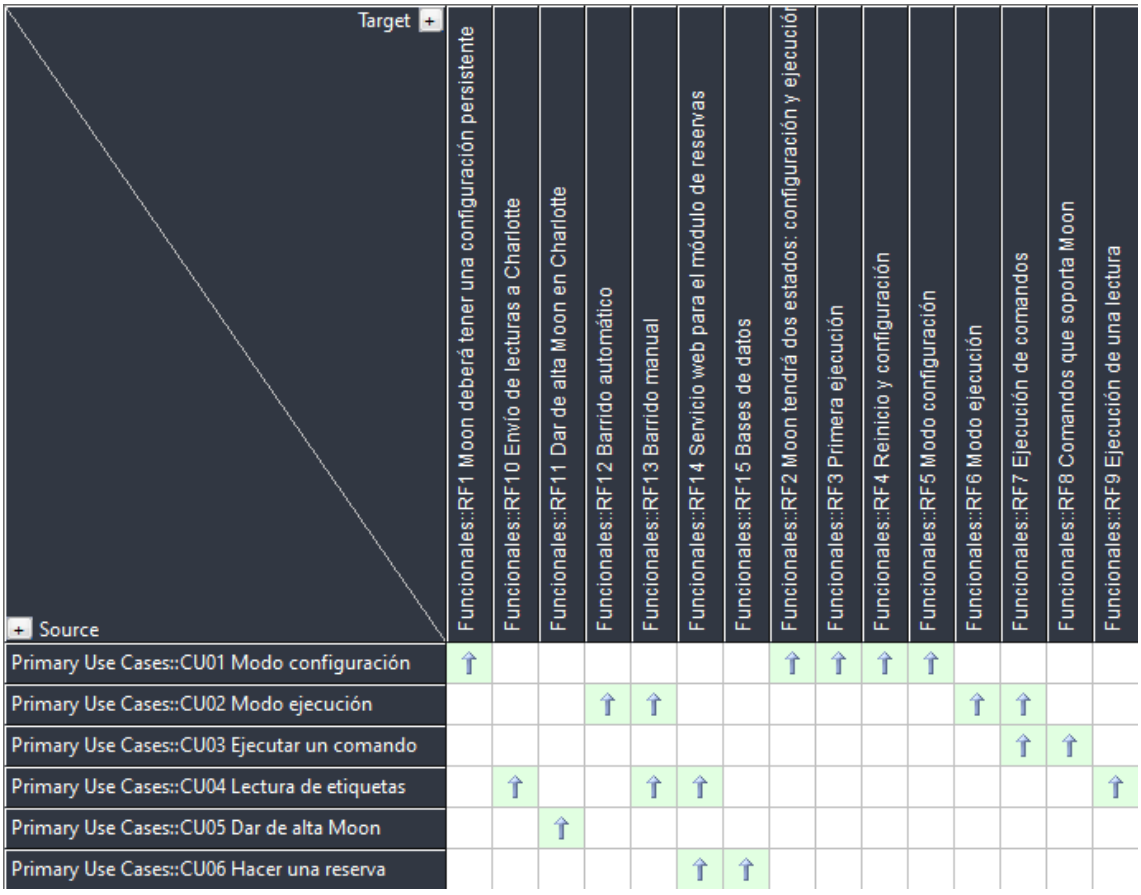


*Ilustración 21 Matriz de relación casos de uso - requisitos*

## 6.9. Representación gráfica de los casos de uso

La representación gráfica de los casos de uso muestra las opciones del sistema que puede realizar cada actor.

El siguiente diagrama son los casos de uso:

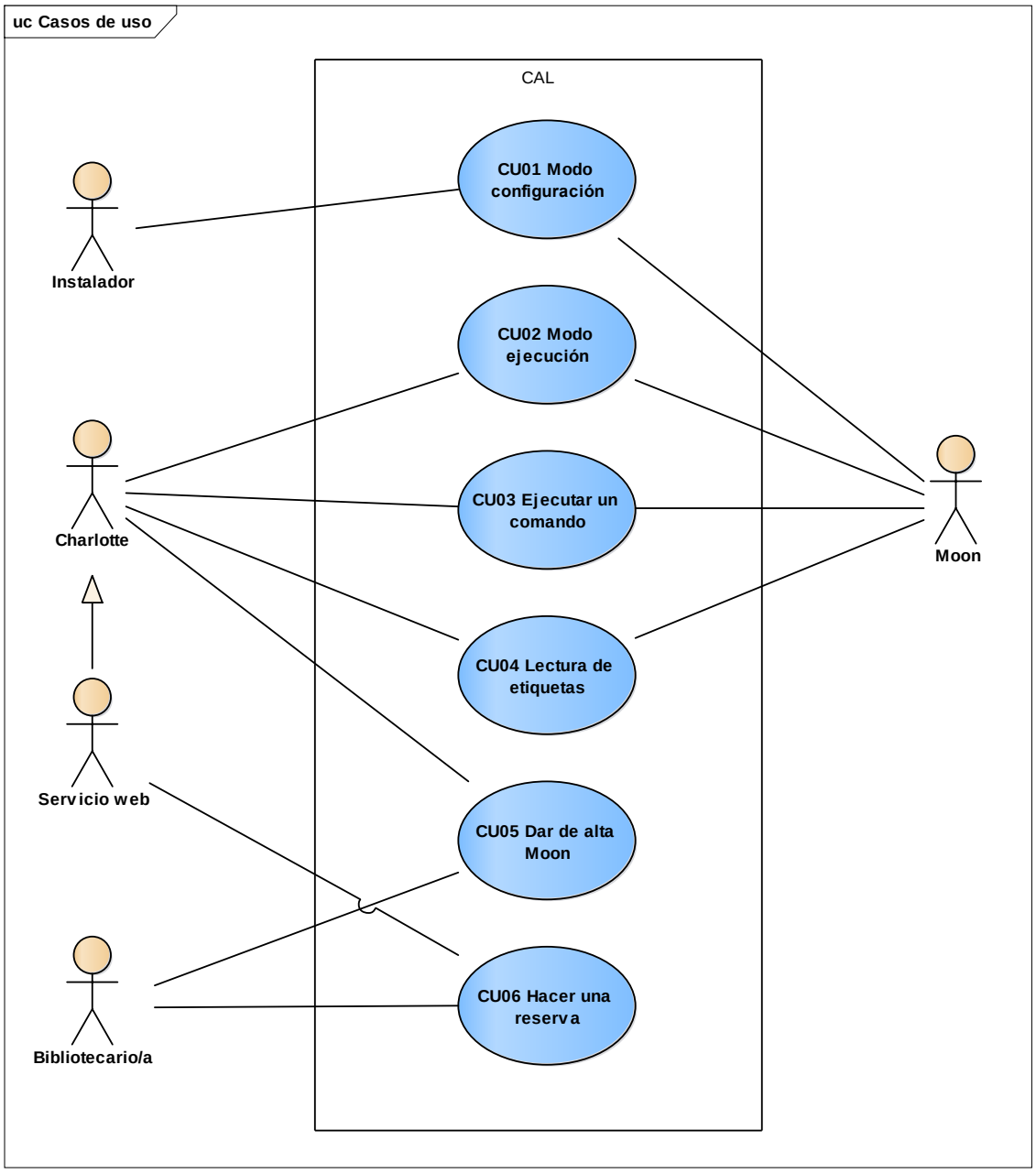


*Ilustración 22 Casos de uso*

El siguiente diagrama son los casos de uso con los requisitos funcionales que implementan:

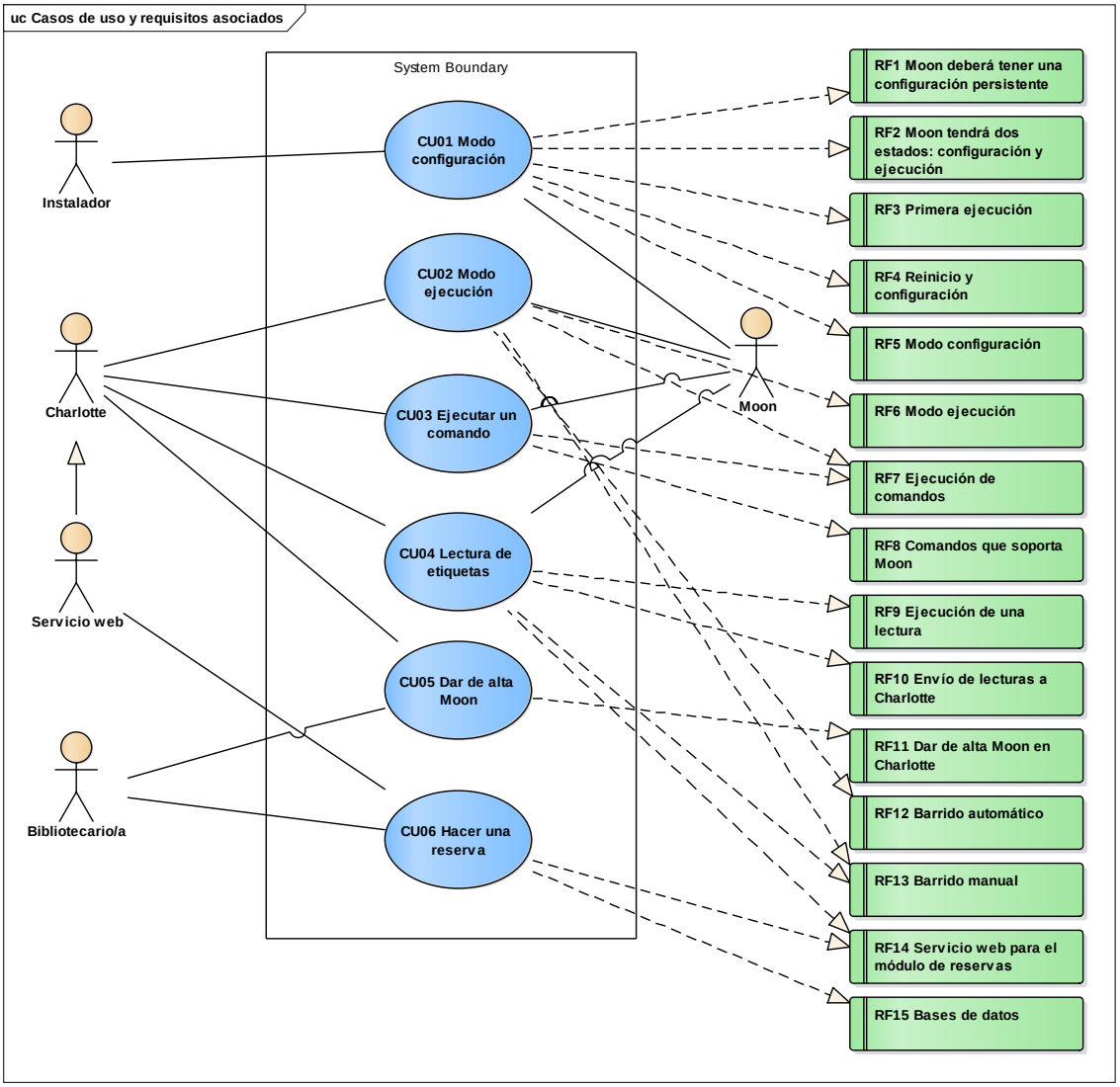


*Ilustración 23 Casos de uso y requisitos asociados*

# 7. Desarrollo/Experimentación/Validación

## 7.1. Entorno de desarrollo

Para los microcontroladores tipo Arduino [13] existe un entorno de desarrollo con su mismo nombre. Este entorno de desarrollo es muy sencillo, sólo tiene una ventana y no sugiere las instrucciones más comunes según vas escribiendo código. Está pensado para pequeños desarrollos.

Se va a utilizar Visual Studio Code [14] que es un entorno multiplataforma gratuito y al que se le puede añadir diferentes plugin para trabajar con diferentes lenguajes. Lo primero que hay que hacer es instalar Visual Studio Code. Para trabajar con microcontroladores tipo Arduino hay que instalar el plugin [15] en Visual Studio Code. Para instalarlo hay que abrir VSCode Extension Manager, buscar la extensión PlatformIO IDE e instalarla.

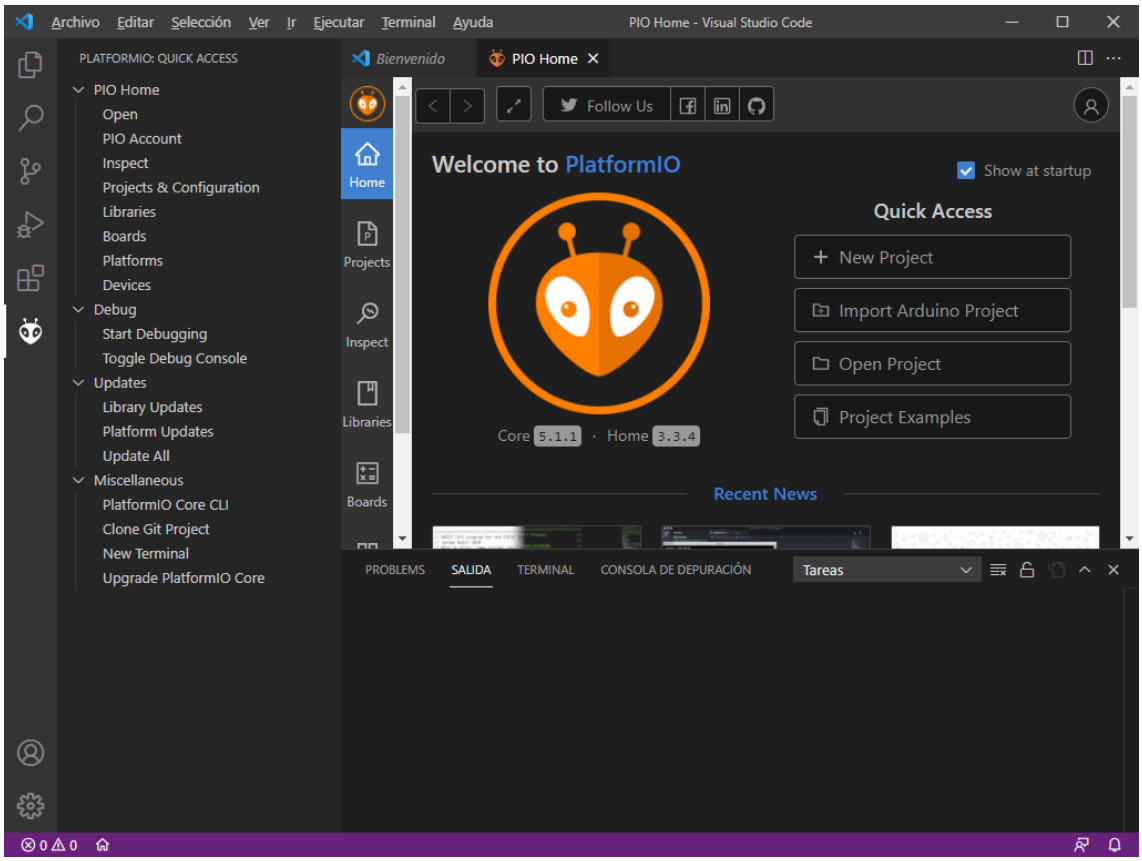


*Ilustración 24 Visual Studio Code con PlatformIO*

Es necesario instalar el IDE Arduino para que se instale el driver USB a puerto serie que se utilizará para trabajar entre el PC y el microcontrolador. Para verificar que está instalado hay que abrir el administrador de dispositivos y en puertos COM y LPT debe aparecer un driver de comunicación serie.

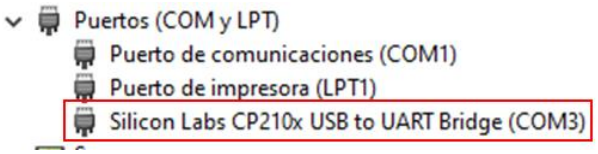


*Ilustración 25 Driver PC microcontrolador*

Para trabajar con el ESP32 en PlatformIO hay que seleccionar Devices, Boards y buscar ESP32 y el modelo concreto, en este caso el modelo DOIT, e instalarlo.

Para la creación del servicio web se ha utilizado el entorno de desarrollo Visual Studio [16]. El lenguaje utilizado es C# que tiene una sintaxis similar a Java. Para la base de datos de Charlotte se ha utilizado SQLServer [17].

## 7.2. Microprocesador IoT ESP32

El microprocesador utilizado está montado sobre una placa de desarrollo para facilitar su uso en desarrollos a pequeña escala. La disposición y tipo de pines son las siguientes[5]:

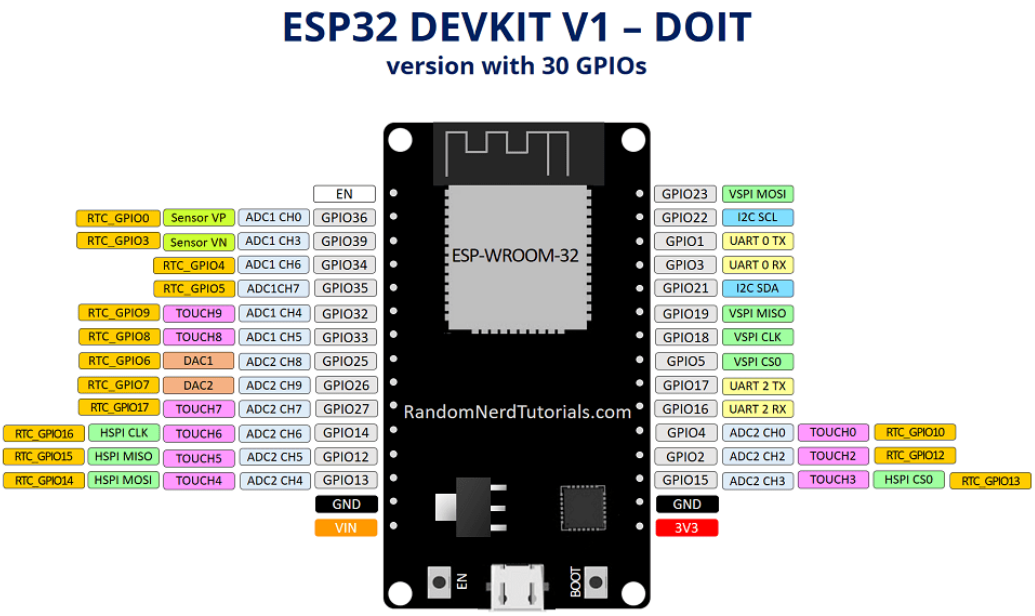


*Ilustración 26 ESP32 Asignación de pines*

Aunque tiene muchos pines que realizan diferentes funciones existen una serie de restricciones en algunos de ellos. Las restricciones se definen en la hoja de datos [11].

La función de los pines se muestra en la siguiente tabla [11]:

[5] Fuente: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

*Tabla 7 Pines del ESP32*

| Nombre | No. | Tipo | Función |
|---|---|---|---|
| SENSOR_VP | 4 | I | GPIO36, ADC1_CH0, RTC_GPIO0 |
| SENSOR_VN | 5 | I | GPIO39, ADC1_CH3, RTC_GPIO3 |
| IO34 | 6 | I | GPIO34, ADC1_CH6, RTC_GPIO4 |
| IO35 | 7 | I | GPIO35, ADC1_CH7, RTC_GPIO5 |
| IO32 | 8 | I/O | GPIO32, XTAL_32K_P (32.768 kHz entrada del oscilador de cristal), ADC1_CH4,<br>TOUCH9, RTC_GPIO9 |
| IO33 | 9 | I/O | GPIO33, XTAL_32K_N (32.768 kHz salida del oscilador de cristal), ADC1_CH5,<br>TOUCH8, RTC_GPIO8 |
| IO25 | 10 | I/O | GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0 |
| IO26 | 11 | I/O | GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1 |
| IO27 | 12 | I/O | GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV |
| IO14 | 13 | I/O | GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK,<br>SD_CLK, EMAC_TXD2 |
| IO12 | 14 | I/O | GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2,<br>SD_DATA2, EMAC_TXD3 |
| GND | 15 | P | Masa |

| | | | |
|---|---|---|---|
| IO13 | 16 | I/O | GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3,<br>SD_DATA3, EMAC_RX_ER |
| SHD/SD2* | 17 | I/O | GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD |
| SWP/SD3* | 18 | I/O | GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD |
| SCS/CMD* | 19 | I/O | GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS |
| SCK/CLK* | 20 | I/O | GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS |
| SDO/SD0* | 21 | I/O | GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS |
| SDI/SD1* | 22 | I/O | GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS |
| IO15 | 23 | I/O | GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD,<br>SD_CMD, EMAC_RXD3 |
| IO2 | 24 | I/O | GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0,<br>SD_DATA0 |
| IO0 | 25 | I/O | GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK |
| IO4 | 26 | I/O | GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1,<br>SD_DATA1, EMAC_TX_ER |
| IO16 | 27 | I/O | GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT |
| IO17 | 28 | I/O | GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180 |
| IO5 | 29 | I/O | GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK |

| | | | |
|---|---|---|---|
| IO18 | 30 | I/O | GPIO18, VSPICLK, HS1_DATA7 |
| IO19 | 31 | I/O | GPIO19, VSPIQ, U0CTS, EMAC_TXD0 |
| NC | 32 | - | - |
| IO21 | 33 | I/O | GPIO21, VSPIHD, EMAC_TX_EN |
| RXD0 | 34 | I/O | GPIO3, U0RXD, CLK_OUT2 |
| TXD0 | 35 | I/O | GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2 |
| IO22 | 36 | I/O | GPIO22, VSPIWP, U0RTS, EMAC_TXD1 |
| IO23 | 37 | I/O | GPIO23, VSPID, HS1_STROBE |
| GND | 38 | P | Masa |

Los pines utilizados en Moon son:

*Tabla 8 Pines utilizados del ESP32*

| Pin (GPIO) | Uso |
|---|---|
| 2 | LED rojo indicador de lectura de etiquetas |
| 4 | LED verde indicador de encendido |
| 15 | Botón de configuración |
| 16 | RX2, Recepción de antena RFID |
| 17 | TX2, Transmisión de antena RFID |

## 7.3. Librerías externas

Existen múltiples librerías para el ESP32 y siempre que sea posible se utilizará una librería existente en lugar de crear una nueva. Las librerías utilizadas en este TFG son:

*Tabla 9 Librerías externas*

| Nombre de librería | Autor | Utilizada en |
|---|---|---|
| AsyncTCP-esphome [18] | Hristo Gochkov | ESPAsyncWebServer-esphome |
| ESPAsyncTCP-esphome [19] | Hristo Gochkov | ESPAsyncWebServer-esphome |
| ESPAsyncWebServer-esphome [20] | ESPHome Team | Clase WiFiWebServer |
| Streaming [21] | Mikal Hart | Varias clases |

## 7.4. Módulos software de Moon

La clase Moon contiene las instancias de los elementos del hardware, las de comunicaciones con Charlotte y la lectura de etiquetas.

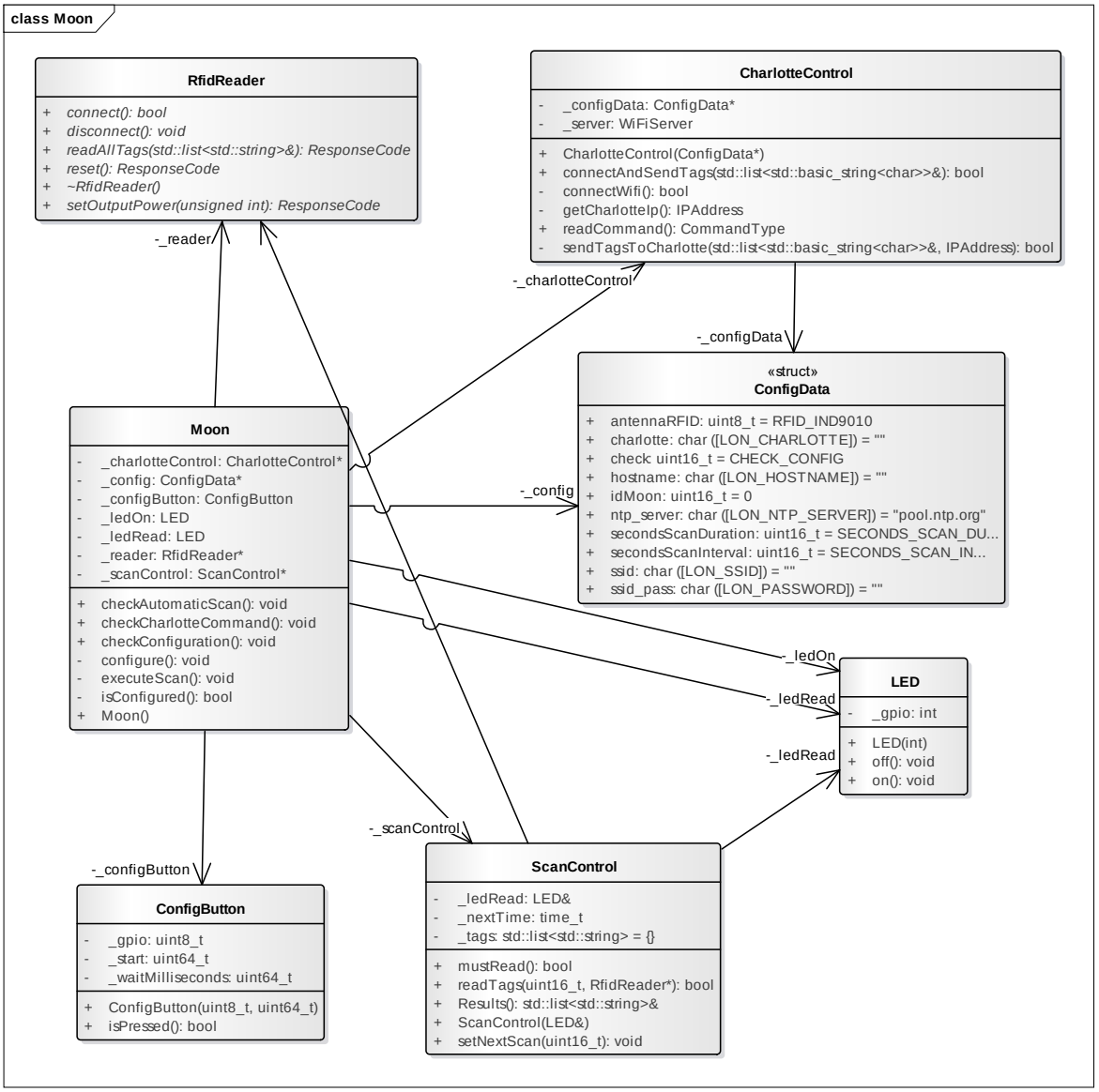


*Ilustración 27 Diagrama de clases de Moon*

### 7.4.1. Configuración persistente

El ESP32 dispone de una memoria EEPROM cuyo contenido no se pierde al apagarlo o reiniciarlo.

El diagrama de clases es:

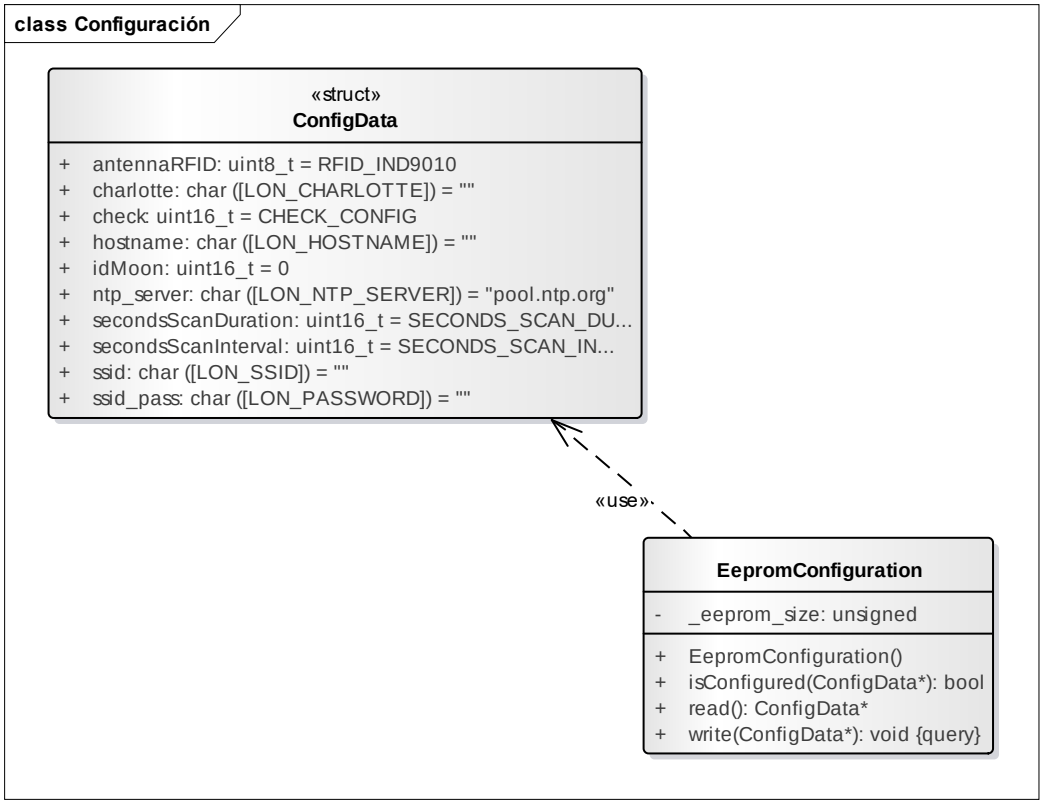


*Ilustración 28 Diagrama de clases de la configuración persistente*

La estructura ConfigData contiene la representación en memoria de todas las variables que se guardan en la memoria EEPROM del microprocesador ESP32. La clase EepromConfiguration es la que implementa los métodos para leer y guardar la estructura ConfigData en esta memoria.

### 7.4.2. Configuración de Moon

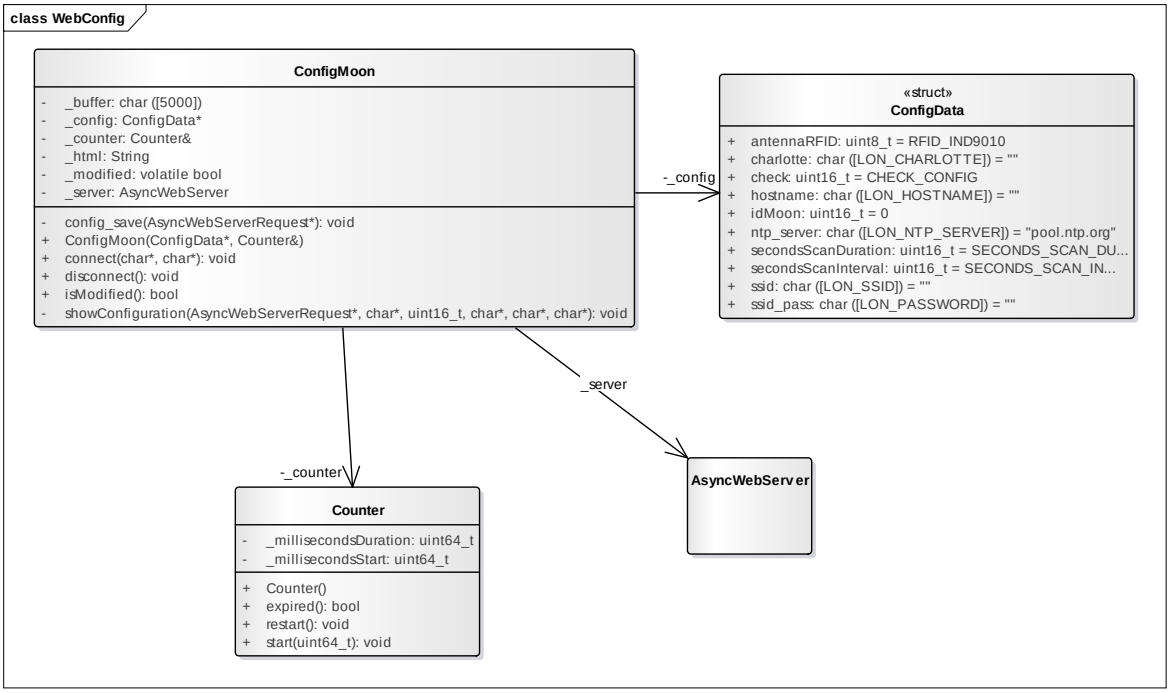


*Ilustración 29 Diagrama de clases de la configuración de Moon*

Para poder utilizar Moon debe configurarse. Los parámetros que necesitan son su identificador, las credenciales del WiFi al que debe conectarse y el nombre del equipo donde está instalado Charlotte.

La primera vez que se enciende Moon crea una red WiFi con nombre “MoonServer” y contraseña “12345678”. El instalador debe conectar un navegador a este WiFi e ir a la página http://192.168.4.1. En el navegador aparecerá la página de configuración de Moon donde deberán rellenarse los datos de identificación del dispositivo, nombre del WiFi al que se debe conectar y su contraseña.

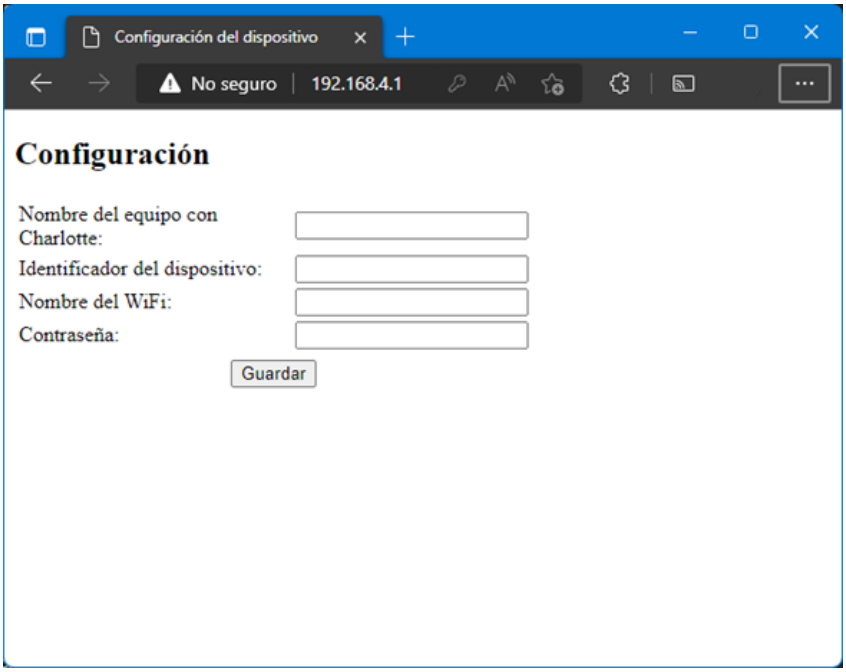


*Ilustración 30 Página de configuración de Moon*

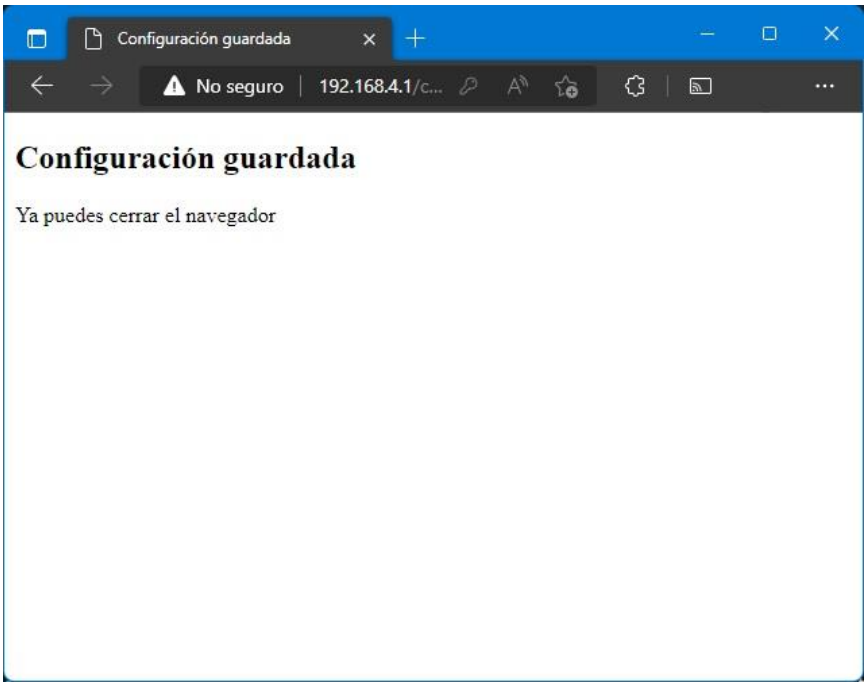

*Ilustración 31 Confirmación guardada correctamente*

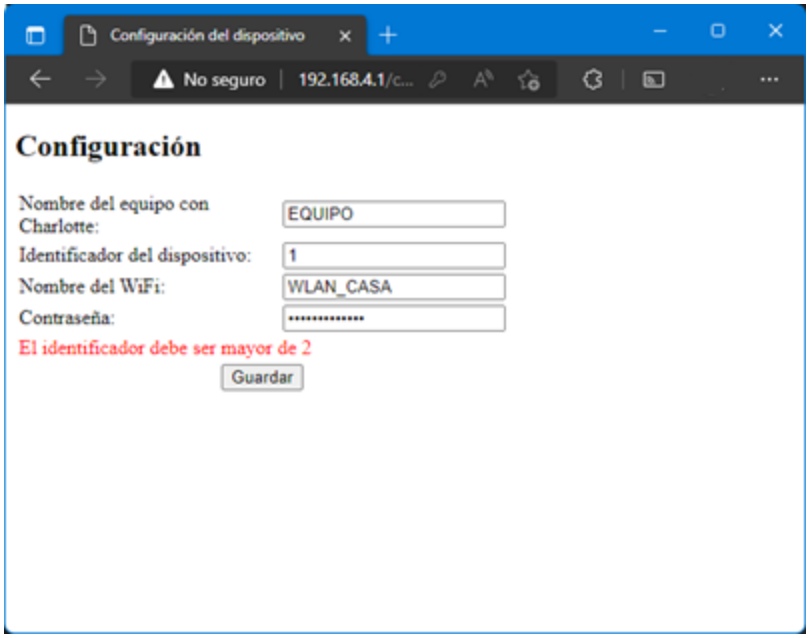


*Ilustración 32 Configuración con errores*

Una vez se introducen los datos Moon comprueba que sean válidos. Si hay errores en los campos mostrará un mensaje (si el nombre del equipo con Charlotte está vacío, si el identificador es menor que dos, si el nombre del WiFi está vacío o si la longitud de la contraseña es menor de 8 caracteres). Una vez se introduzcan correctamente, Moon los guardará en la configuración para poder acceder a ellos en caso de que se reinicie. Para poder cambiar la configuración hay que pulsar el botón de Moon durante al menos diez segundos. El motivo es evitar que entre en modo configuración por una pulsación involuntaria. Al pulsar el botón para cambiar la configuración se mostrará de nuevo la página web de configuración con los datos actuales para modificarlos. Al estar configurado si entra en modo configuración y no hay interacción del instalador vuelve a modo ejecución a los dos minutos. Si se quedara en modo configuración no realizaría ninguna lectura automática de etiquetas.

### 7.4.3. Lectura de etiquetas

La comunicación entre Moon y la antena RFID se realiza mediante el protocolo serie. La librería de clases de Arduino tiene una clase llamada HardwareSerial que implementa el protocolo serie. El controlador ESP32 dispone de dos puertos serie controlados mediante hardware. El puerto 0 se usa para programar el controlador ESP32, en este caso se ha elegido el puerto 2.

Para facilitar el cambio de antena por otro modelo se ha creado una clase base abstracta llamada RfidReader con los métodos necesarios para el control de la antena y lectura de las etiquetas. Para usar una antena hay que derivar una clase de esta e implementar sus métodos.

La clase que realiza el control de la antena es la RfidReader9011 que implementa la clase RfidReader y tiene una instancia de HardwareSerial para realizar la comunicación con la antena. Para abstraer al resto del código el uso de la clase RfidReader9011 sólo debe utilizarse la clase RfidReader. Para crear la instancia de RfidReader9011 u otra que se pueda crear posteriormente se ha usado el patrón Factory. La clase RfidReaderFactory tiene el método create que selecciona qué clase hay que utilizar según la configuración y devuelve una instancia de la clase base RfidReader. De este modo se aísla el resto de código de la clase concreta utilizada para el manejo de una antena RFID.

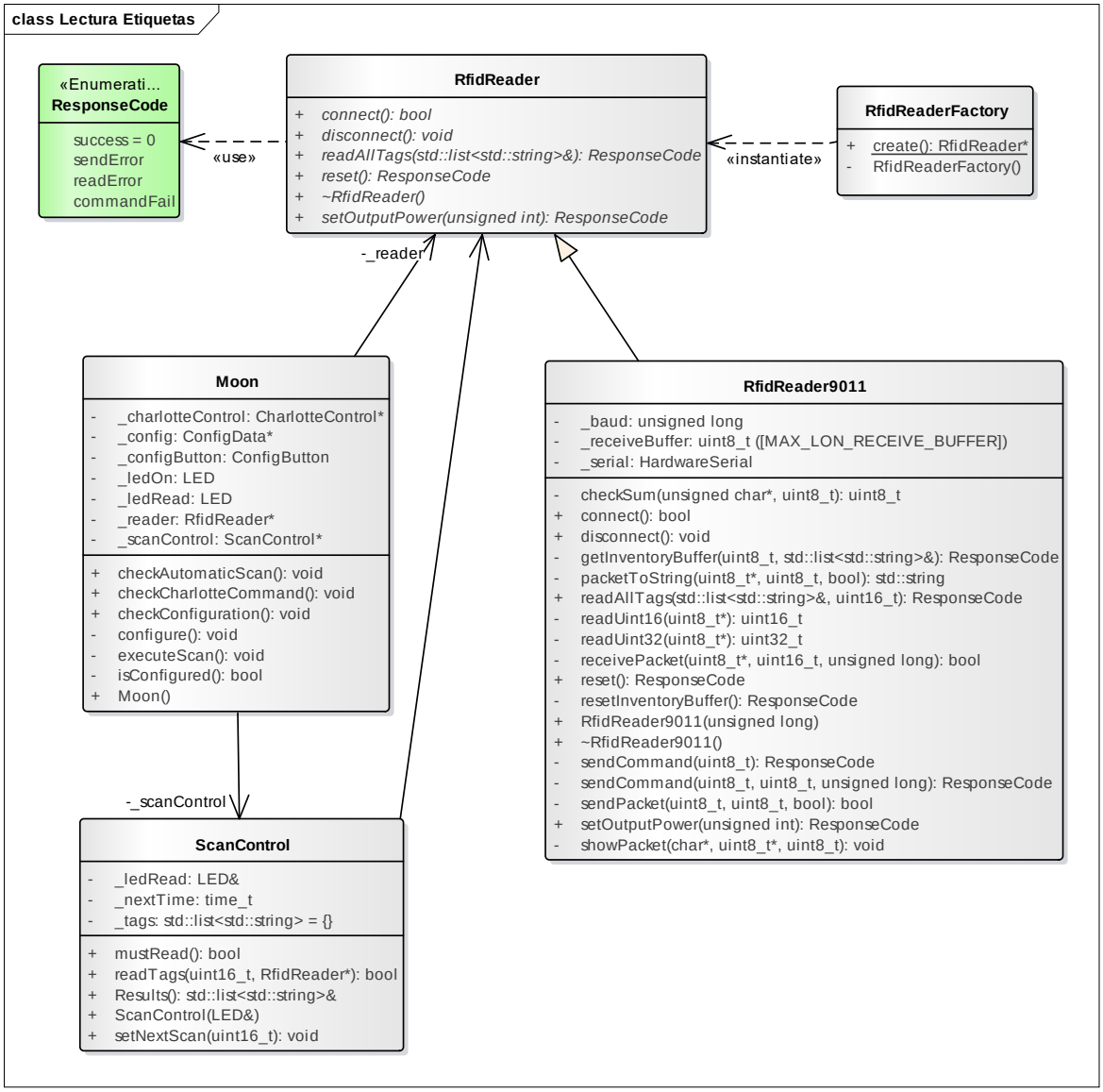


*Ilustración 33 Diagrama de clases de la lectura de etiquetas*

Los métodos definidos en la clase RfidReader son:

*Tabla 10 Métodos de la clase RfidReader*

| Método | Descripción | Parámetros | Respuesta |
|---|---|---|---|
| connect | Conecta el lector RFID | Ninguno | Booleano que indica si ha podido conectar |
| reset | Reinicia el lector RFID | Ninguno | Instancia de ResponseCode con el código de la respuesta |
| setOutputPower | Configura la potencia de salida del lector | Unsigned int con el valor de la potencia entre 0 y 100. Siendo 0 la potencia mínima y 100 la potencia máxima | Instancia de ResponseCode con el código de la respuesta |
| readAllTags | Lee todas las etiquetas RFID | Parámetro de salida con la lista de etiquetas leídas | Instancia de ResponseCode con el código de la respuesta |
| disconnect | Desconecta el lector RFID | Ninguno | Ninguna |

La clase RfidReader9011 realiza la comunicación con la antena RFID utilizando los comandos vistos en los puntos anteriores. El fabricante no proporciona ningún código de ejemplo en C++ que pueda utilizarse en este proyecto. Hay unos ejemplos escritos en C# que se han utilizado como guía para realizar el código de esta clase. El método sendCommand crea el paquete de datos y se lo envía al lector. El método receivePacket lee la respuesta del lector, la guarda en una variable de la clase, decodifica el código de error y lo devuelve. Tiene un parámetro que indica el tiempo máximo que esperará una respuesta de la antena. El método checksum calcula la suma de verificación del paquete. El código de esta función lo proporciona el fabricante en su documentación. El método readAllTags utiliza los métodos descritos para enviar el comando, leer la respuesta, las etiquetas leídas las devuelve en un parámetro de salida.

La clase Moon periódicamente consulta a la clase ScanControl para comprobar si hay que hacer una lectura automática. En ese caso, llama al método executeScan que lee las etiquetas a

través de ScanControl::readTags que a su vez se comunica con la clase RfidReader para realizar la lectura. Una vez realizada la lectura llama al método CharlotteControl::connectAndSendTags para enviar los resultados a Charlotte. En el caso de que haya errores durante la lectura o el envío el LED de encendido parpadeará diez veces.

#### *7.4.3.1. Antena IND9011*

El fabricante de la antena [10] ha proporcionado la documentación necesaria para el manejo de la antena incluyendo los comandos, su estructura y el protocolo utilizado. En el manual del fabricante *IND9010_Protocol_User's_Guide_V2.38_en.pdf*, incluido en la entrega del proyecto, se describe el protocolo y los comandos necesarios para programar el lector RFID.

##### *7.4.3.1.1. Protocolo de comunicaciones*

La interfaz física es compatible con las especificaciones de RS-232 [9].

1 bit de inicio, 8 bits de datos, 1 bit de stop, sin paridad.

La velocidad puede seleccionarse entre 38400bps y 115200bps. El valor por defecto es 115200bps.

##### *7.4.3.1.2. Definición del paquete de datos*

*Tabla 11 Definición del paquete de datos. Antena IND9011*

<table>
<tr><th>Cabecera</th><th>Longitud</th><th>Dirección</th><th>Comando</th><th>Datos</th><th>Comprobación</th></tr>
<tr><td>0xA0</td><td>1 Byte</td><td>1 Byte</td><td>1 Byte</td><td>N Bytes</td><td>1 Byte</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td rowspan="3">Descripción del parámetro</td><td colspan="2">Cabecera</td><td colspan="3">Cabecera del paquete, todos los paquetes empiezan con 0xA0.</td></tr>
<tr><td colspan="2">Longitud</td><td colspan="3">Longitud del paquete de bytes. Empieza desde el tercer byte. Los bytes de la cabecera y la longitud no están incluidos.</td></tr>
<tr><td colspan="2">Dirección</td><td colspan="3">La dirección del lector para la conexión RS-485. Las direcciones comunes son 0～254(0xFE), 255(0xFF) es la dirección pública.</td></tr>
</table>

| | | |
|---|---|---|
| | Comando | Byte de comando. |
| | Datos | Parámetros del comando. |
| | Comprobación | Suma de comprobación. Incluye todos los bytes excepto el suyo propio. |

La antena responde con un paquete de datos que contendrá información del comando utilizado o un código de error. La estructura del paquete de datos de la respuesta se muestra en la tabla siguiente:

*Tabla 12 Definición de la respuesta. Antena IND9011*

| Cabecera | Longitud | Dirección | Datos | Comprobación |
|---|---|---|---|---|
| 0xA0 | 1 Byte | 1 Byte | N Bytes | 1 Byte |
| Descripción del parámetro | | Cabecera | Cabecera del paquete, todos los paquetes empiezan con 0xA0. | |
| | | Longitud | Longitud del paquete de bytes. Empieza desde el tercer byte. Los bytes de la cabecera y la longitud no están incluidos. | |
| | | Dirección | La dirección del lector. | |
| | | Datos | Datos del lector. | |
| | | Comprobación | Suma de comprobación. Incluye todos los bytes excepto el suyo propio. | |

#### *7.4.3.1.3. Lista de comandos utilizados*

Sólo se usa una parte de los comandos que implementa la antena IND9011. La tabla de los comandos utilizados se muestra a continuación.

*Tabla 13 Lista de comandos utilizados. Antena IND9011*

| ID | Código | Nombre | Descripción |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Reader Control Commands | | | |
| 1 | 0x70 | cmd_reset | Reinicia el lector. |
| 7 | 0x76 | cmd_set_output_power | Define la potencia de salida RF. |
| 8 | 0x77 | cmd_get_output_power | Obtiene la potencia de salida RF. |
| 18000-6C Commands | | | |
| 23 | 0x80 | cmd_inventory | Realiza el inventario de etiquetas EPC C1G2 y lo almacena. |
| Buffer control Commands | | | |
| 41 | 0x90 | cmd_get_inventory_buffer | Obtiene y borra los datos almacenados. |
| 42 | 0x91 | cmd_get_and_reset_inventory_ buffer | Obtiene los datos almacenados sin borrarlos. |
| 43 | 0x92 | cmd_get_inventory_buffer_tag_ count | Pregunta cuántas etiquetas están almacenadas. |
| 44 | 0x93 | cmd_reset_inventory_buffer | Limpia el almacenamiento. |

#### 7.4.3.1.3.1. cmd_reset

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Comprobación |
|---|---|---|---|---|
| 0xA0 | 0x03 | | 0x70 | |

◆**Éxito:** No hay datos de respuesta, el lector se resetea y reinicia. El zumbador emite un pitido.

◆Fallo:

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | CódigoError | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x70 | | |
| | | | | | |
| Descripción del parámetro | CódigoError | Código de error. | | | |

7.4.3.1.3.2. cmd_set_output_power

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Potencia | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x76 | | |
| | | | | | |
| Descripción del parámetro | Potencia | Potencia RF, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. | | | |

O:

| Cabecera | Longitud | Dirección | Comando | Pot1 | Pot2 | Pot3 | Pot4 | Comprobación |
|---|---|---|---|---|---|---|---|---|
| 0xA0 | 0x07 | | 0x76 | | | | | |
| | | | | | | | | |
| Descripción del parámetro | Pot1 | Potencia de salida de la antena 1, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. | | | | | | |
| | Pot2 | Potencia de salida de la antena 2, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. | | | | | | |
| | Pot3 | Potencia de salida de la antena 3, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. | | | | | | |

| | | |
|---|---|---|
| | Pot4 | Potencia de salida de la antena 4, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. |

◆Éxito:

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | CódigoError | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x76 | CommandSuccess | |

◆Fallo:

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | CódigoError | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x76 | | |
| | | | | | |
| Descripción del parámetro | CódigoError | Código de error. | | | |

El valor de la potencia se guarda en la flash interna para que no se pierda al apagarlo.

#### 7.4.3.1.3.3. cmd_get_output_power

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Comprobación |
|---|---|---|---|---|
| 0xA0 | 0x03 | | 0x77 | |

Si todas las antenas tienen la misma potencia de salida, entonces el paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | Potencia | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x77 | | |
| | | | | | |

| | | |
|---|---|---|
| Descripción del parámetro | Potencia | Potencia de salida actual. |

Si no el paquete de respuesta es:

| Cabecera | Longitud | Dirección | Comando | Pot1 | Pot2 | Pot3 | Pot4 | Comprobación |
|---|---|---|---|---|---|---|---|---|
| 0xA0 | 0x07 | | 0x77 | | | | | |

| | | |
|---|---|---|
| Descripción del parámetro | Pot1 | Potencia de salida de la antena 1, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. |
| | Pot2 | Potencia de salida de la antena 2, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. |
| | Pot3 | Potencia de salida de la antena 3, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. |
| | Pot4 | Potencia de salida de la antena 4, rango de 0 a 33(0x00 – 0x21), la unidad es dBm. |

#### 7.4.3.1.3.4. cmd_inventory

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Repetición | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x80 | | |

| | | |
|---|---|---|
| Descripción del parámetro | Repetición | Tiempo de repetición del inventario.<br><br>Cuando Repetición es 255, minimiza la duración del inventario. Por ejemplo, si solo hay una o dos etiquetas, la duración del inventario podría ser 30-50 ms, esta función permite a la aplicación el cambio rápido de antena en equipos con varias antenas. |

Cuando el lector recibe este comando, inicia el inventario de las etiquetas EPC GEN, los datos de las etiquetas se almacenan en la memoria interna.

Atención:

★ Cuando se establece el parámetro Repetir en 255 (0xFF), el algoritmo anticolisión se optimiza para aplicaciones con una pequeña cantidad de etiquetas, lo que proporciona una mejor eficiencia y menor tiempo de respuesta.

◆Éxito:

Paquete de respuesta:

<table>
<tr><th>Cabecera</th><th>Longitud</th><th>Dirección</th><th>Comando</th><th>IDAntena</th><th>Cont Etiquetas</th><th>Velocidad Lectura</th><th>TotalLeído</th><th>Comprobación</th></tr>
<tr><td>0xA0</td><td>0x0C</td><td></td><td>0x80</td><td></td><td>2 Bytes</td><td>2Bytes</td><td>4Bytes</td><td></td></tr>
<tr><td colspan="9"></td></tr>
<tr><td rowspan="4">Descripción del parámetro</td><td>IDAntena</td><td colspan="7">Identificador de antena usado.</td></tr>
<tr><td>ContEtiquetas</td><td colspan="7">Número de etiquetas identificadas. Las etiquetas se diferencian por el EPC, las etiquetas con la misma EPC son consideradas como una. Si el almacenamiento del lector tiene datos, el contador de etiquetas se incrementa.</td></tr>
<tr><td>VelocidadLectura</td><td colspan="7">Velocidad de identificación de etiquetas (etiquetas/segundo). Incluye las etiquetas con la misma EPC.</td></tr>
<tr><td>TotalLeído</td><td colspan="7">Número total de etiquetas leídas. Las etiquetas con la misma EPC se incluyen en el total.</td></tr>
</table>

◆Fallo:

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | CódigoError | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x80 | | |

| | | |
|---|---|---|
| | | |
| Descripción del parámetro | CódigoError | Código de error |

7.4.3.1.3.5. cmd_get_inventory_buffer

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Comprobación |
|---|---|---|---|---|
| 0xA0 | 0x03 | | 0x90 | |

◆Éxito:

Paquete de respuesta: Este comando puede tener múltiples paquetes de respuesta, la cantidad de paquetes de respuesta equivale a la cantidad de etiquetas que se han guardado.

| Comando | Cont Etiquetas | Longitud Datos | Datos | RSSI | FREC | Frec Antena | Cont Inventario | Comprobación |
|---|---|---|---|---|---|---|---|---|
| 0x90 | 2 Bytes | | N bytes | | | | | |

| | |
|---|---|
| ContEtiquetas | Número de etiquetas guardadas. 16bits. |
| LongitudDatos | Longitud de datos útiles de una etiqueta. (PC+CRC+EPC), en bytes. |
| Datos | Datos útiles de una etiqueta.<br>PC (2 bytes) + EPC (bytes) + CRC (2 bytes) |
| RSSI | El RSSI de la etiqueta. |
| FrecAntena | Los 6 bits más significativos contienen la frecuencia; los 2 menos significativos contienen el ID de la antena. |

| | |
|---|---|
| ContInventario | Número de veces que ha identificado la etiqueta. Si el valor es 0xFF, significa que la ha identificado 255 o más veces. |

Atención:

★Los datos del almacenamiento no se perderán después de la ejecución de este comando.

★Otros comandos 18000-6C pueden borrar el almacenamiento.

◆Fallo:

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | CódigoError | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x90 | | |
| | | | | | |
| Descripción del parámetro | CódigoError | Código de error. | | | |

#### 7.4.3.1.3.6. cmd_get_and_reset_inventory_buffer

Ver el comando cmd_get_inventory_buffer.

Tras la ejecución de este comando, el almacenamiento se vacía.

#### 7.4.3.1.3.7. cmd_get_inventory_buffer_tag_count

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Comprobación |
|---|---|---|---|---|
| 0xA0 | 0x03 | | 0x92 | |

◆Éxito:

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | Cont | Comprobación |
|---|---|---|---|---|---|

| | | | | Etiquetas | |
|---|---|---|---|---|---|
| 0xA0 | 0x05 | | 0x92 | 2 Bytes | |
| | | | | | |
| Descripción del parámetro | ContEtiquetas | Cuántas etiquetas hay guardadas | | | |

#### 7.4.3.1.3.8. cmd_reset_inventory_buffer

Paquete del host:

| Cabecera | Longitud | Dirección | Comando | Comprobación |
|---|---|---|---|---|
| 0xA0 | 0x03 | | 0x93 | |

Paquete de respuesta:

| Cabecera | Longitud | Dirección | Comando | CódigoError | Comprobación |
|---|---|---|---|---|---|
| 0xA0 | 0x04 | | 0x93 | CommandSuccess | |

#### 7.4.3.1.3.9. Códigos de error

| ID | Code | Nombre | Descripción |
|---|---|---|---|
| 1 | 0x10 | CommandSuccess | Comando ejecutado con éxito. |
| 2 | 0x11 | command_fail | Fallo del comando. |
| 3 | 0x20 | mcu_reset_error | Error de reinicio de CPU. |
| 4 | 0x21 | cw_on_error | Error de encendido. |
| 5 | 0x22 | antenna_missing_error | Falta la antena. |
| 6 | 0x23 | write_flash_error | Error al escribir en la memoria flash. |
| 7 | 0x24 | read_flash_error | Error al leer en la memoria flash. |

| | | | |
|---|---|---|---|
| 8 | 0x25 | set_output_power_error | Error al definir la potencia de salida. |
| 9 | 0x31 | tag_inventory_error | Error al hacer el inventario. |
| 10 | 0x32 | tag_read_error | Error al leer. |
| 11 | 0x33 | tag_write_error | Error al escribir. |
| 12 | 0x34 | tag_lock_error | Error al bloquear. |
| 13 | 0x35 | tag_kill_error | Error al terminar el proceso. |
| 14 | 0x36 | no_tag_error | No hay etiqueta. |
| 15 | 0x37 | inventory_ok_but_access_fail | La etiqueta se ha inventariado pero falla el acceso. |
| 16 | 0x38 | buffer_is_empty_error | El almacenamiento está vacío. |
| 17 | 0x40 | access_or_password_error | Error de acceso o error en la contraseña. |
| 18 | 0x41 | parameter_invalid | Parámetro inválido. |
| 19 | 0x42 | parameter_invalid_wordCnt_too_long | WordCnt es muy largo. |
| 20 | 0x43 | parameter_invalid_membank_out_of_range | MemBank fuera de rango. |
| 21 | 0x44 | parameter_invalid_lock_region_out_of_range | Región de bloqueo fuera de rango. |
| 22 | 0x45 | parameter_invalid_lock_action_out_of_range | LockType fuera de rango. |
| 23 | 0x46 | parameter_reader_address_invalid | Dirección del lector inválida. |
| 24 | 0x47 | parameter_invalid_AntennaID_out_of_range | AntennaID fuera de rango. |
| 25 | 0x48 | parameter_invalid_output_power_out_of_range | Potencia de salida fuera de rango. |
| 26 | 0x49 | parameter_invalid_frequency_region_out_of_range | Región de frecuencia fuera de rango. |

| | | | |
|---|---|---|---|
| 27 | 0x4A | parameter_invalid_baudrate_out_of_range | Velocidad en baudios fuera de rango. |
| 28 | 0x4B | parameter_beeper_mode_out_of_range | Valor de zumbador fuera de rango. |
| 29 | 0x4C | parameter_epc_match_len_too_long | Coincidencia de EPC muy larga. |
| 30 | 0x4D | parameter_epc_match_len_error | Error en la longitud EPC. |
| 31 | 0x4E | parameter_invalid_epc_match_mode | Modo de coincidencia EPC inválido. |
| 32 | 0x4F | parameter_invalid_frequency_range | Rango de la frecuencia inválido. |
| 33 | 0x50 | fail_to_get_RN16_from_tag | Fallo al recibir RN16 de la etiqueta. |
| 34 | 0x51 | parameter_invalid_drm_mode | Modo DRM inválido. |
| 35 | 0x52 | pll_lock_fail | No se puede bloquear PLL. |
| 36 | 0x53 | rf_chip_fail_to_response | El chip RF no responde. |
| 37 | 0x54 | fail_to_achieve_desired_output_power | No puede alcanzar la potencia definida. |
| 38 | 0x55 | copyright_authentication_fail | No se puede verificar el copyright del firmware. |
| 39 | 0x56 | spectrum_regulation_error | Error de la regulación del espectro. |
| 40 | 0x57 | output_power_too_low | Potencia de salida muy baja. |

### 7.4.4. Comunicaciones con Charlotte

Para que Charlotte pueda enviar comandos a Moon necesita saber su nombre. El nombre estará compuesto de “Moon_identificador.local” donde identificador es el número de identificador que se asigna en la instalación.

Las comunicaciones de Moon a Charlotte y de Charlotte a Moon se realizan con la clase CharlotteControl. Moon periódicamente llama al método readCommand de CharlotteControl. El método readCommand conecta el WiFi si no está, después inicia el servidor TCP si no está iniciado y si finalmente Charlotte ha enviado algún comando lo ejecuta. Tras la ejecución devuelve una instancia de CommandType a Moon indicando si hay error o el tipo de comando ejecutado. Si el comando fue para modificar el intervalo de lecturas Moon cambia la próxima lectura mediante la clase ScanControl para definir el nuevo intervalo.

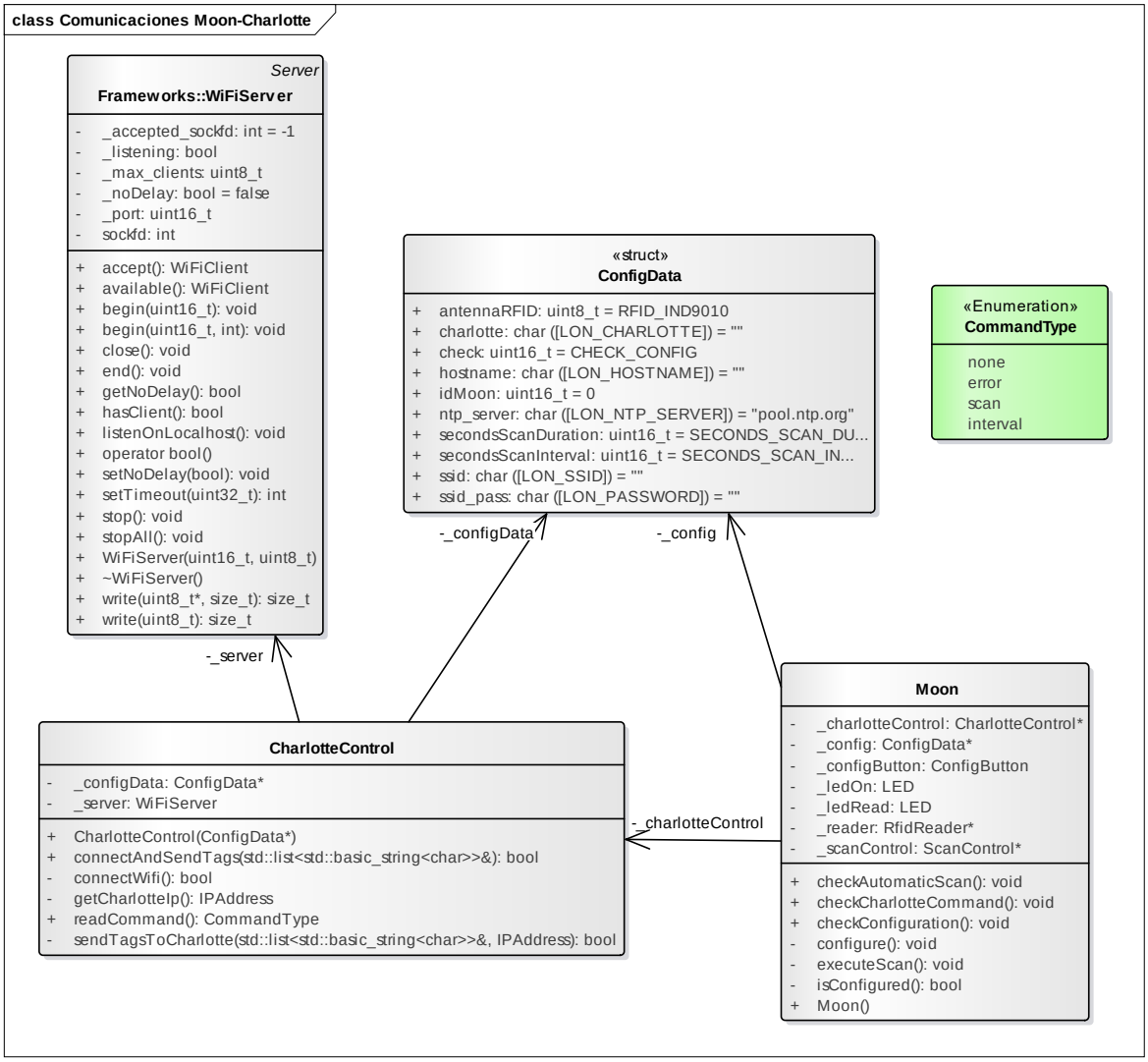


*Ilustración 34 Diagrama de clases de comunicaciones Moon-Charlotte*

## 7.5. Módulos software de Charlotte

Charlotte se ejecuta en un ordenador y se comunicará con múltiples Moon. Charlotte se ha desarrollado en C# que tiene una sintaxis muy parecida a Java. Charlotte tendrá un nombre para que pueda ser encontrado por los dispositivos Moon.

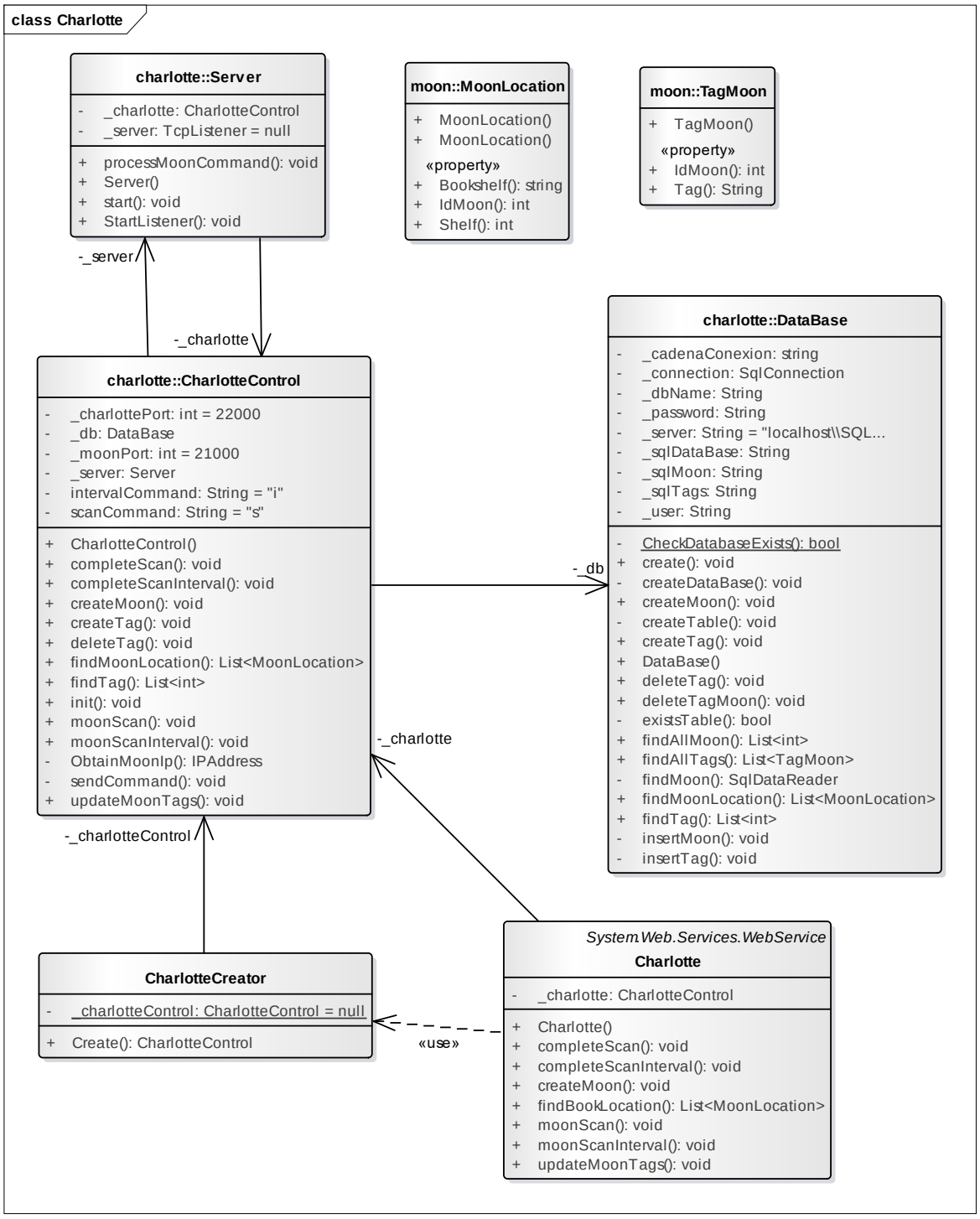


*Ilustración 35 Diagrama de clases de Charlotte*

### 7.5.1. Base de datos

La base de datos de Charlotte contendrá los datos de todos los Moon instalados con su id y localización y las etiquetas RFID de todos los libros de la biblioteca. La actualización de las etiquetas se hará cada vez que se reciban datos de un dispositivo Moon. La estructura se muestra en la siguiente figura:

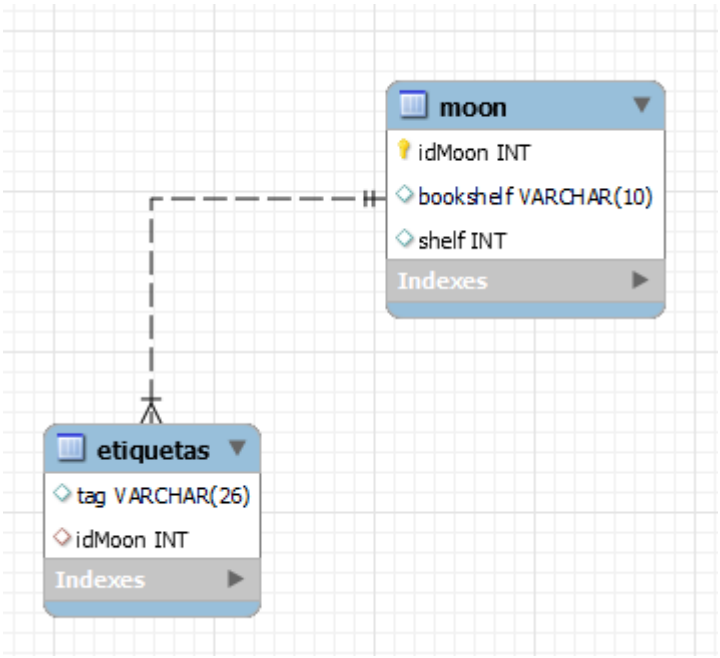


*Ilustración 36 Base de datos de Charlotte*

La tabla de etiquetas puede tener repetida alguna etiqueta asociada a diferentes idMoon. Esto ocurrirá cuando haya un libro al alcance de más de un dispositivo Moon. Cuando el bibliotecario realice una consulta de una de estas etiquetas obtendrá la referencia de varios Moon y esto le ayudará a localizar más fácilmente el libro ya que se encontrará ubicado entre estos Moon.

El control de la base de datos se implementa en la clase DataBase. Esta clase se comunica con SQLServer y contiene las consultas SQL y los métodos para el acceso a la base de datos.

### 7.5.2. Servicio web

En Charlotte existe un servicio web para que el sistema de reservas de la biblioteca consulte la localización física de los libros, dé de alta un Moon, ejecute una lectura manual en uno o todos los dispositivos Moon y cambie el intervalo entre lecturas automáticas de uno o todos.

La lista de métodos del servicio web es:

*Tabla 14 Métodos del servicio web*

| Nombre | Descripción | Parámetros |
|---|---|---|
| findBookLocation | Devuelve una lista de localizaciones de una etiqueta de un libro | El identificador del libro a buscar |

| | | |
|---|---|---|
| createMoon | Da de alta un Moon en Charlotte | La localización del dispositivo Moon |
| moonScan | Ejecuta una lectura manual en un dispositivo Moon concreto | La localización del dispositivo Moon |
| completeScan | Ejecuta una lectura manual en todos los dispositivos Moon | Ninguno |
| moonScanInterval | Cambia el intervalo de lecturas automáticas de un dispositivo Moon | La localización del dispositivo Moon y el intervalo de lecturas |
| completeScanInterval | Cambia el intervalo de lecturas automáticas de todos los dispositivos Moon | El intervalo de lecturas |

El servicio web se implementa en la clase Charlotte. Los métodos de la clase Charlotte usan una instancia de la clase CharlotteControl para ejecutar la operación solicitada. El constructor de la clase Charlotte utiliza la clase CharlotteCreator para obtener la instancia de CharlotteControl. Esta clase implementa el patrón Singleton, si no existe la instancia de CharlotteControl la crea y si existe la devuelve.

### 7.5.3. Comunicaciones con Moon

Las comunicaciones de Charlotte a Moon se realizan con la clase CharlotteControl. Las comunicaciones de Moon a Charlotte se realizan con la clase Server. La clase CharlotteControl tiene el método sendCommand que realiza el envío de un comando a Moon. Este método implementa un cliente TCP. Primero obtiene la dirección IP de Moon a partir de su nombre. Después se conecta y le envía el comando y los datos correspondientes. Al terminar cierra la conexión.

La clase Server se ejecuta en un hilo independiente e implementa un servidor TCP que soporta múltiples clientes en paralelo. Esta clase recibe los comandos de Moon. Cuando se conecta un cliente Moon se crea un hilo de ejecución donde se procesa y ejecuta el comando.

## 7.6. Ejecución de un comando

La lista de comandos de Moon a Charlotte es:

| Comando | Descripción | Parámetros |
|---|---|---|
| TAGS | Moon envía la lista de etiquetas leídas | Identificador de Moon, número de etiquetas y lista de etiquetas |

El comando se envía en modo texto seguido de un carácter de nueva línea. Los parámetros también se envían en modo texto añadiendo un carácter de nueva línea al final.

La lista de comandos de Charlotte a Moon es:

| Comando | Descripción | Parámetros |
|---|---|---|
| s | Ejecutar lectura manual en Moon | Identificador de Moon |
| i | Cambia el intervalo y la duración de la lectura automática de etiquetas en Moon | Identificador de Moon, intervalo y duración |

El comando y los parámetros se envían en modo texto seguido de un carácter de nueva línea.

## 7.7. Programa de prueba del servicio web

Se ha desarrollado un programa de prueba con una interfaz gráfica para probar los métodos del servicio web simulando lo que haría el programa de la biblioteca.

La interfaz se muestra a continuación:

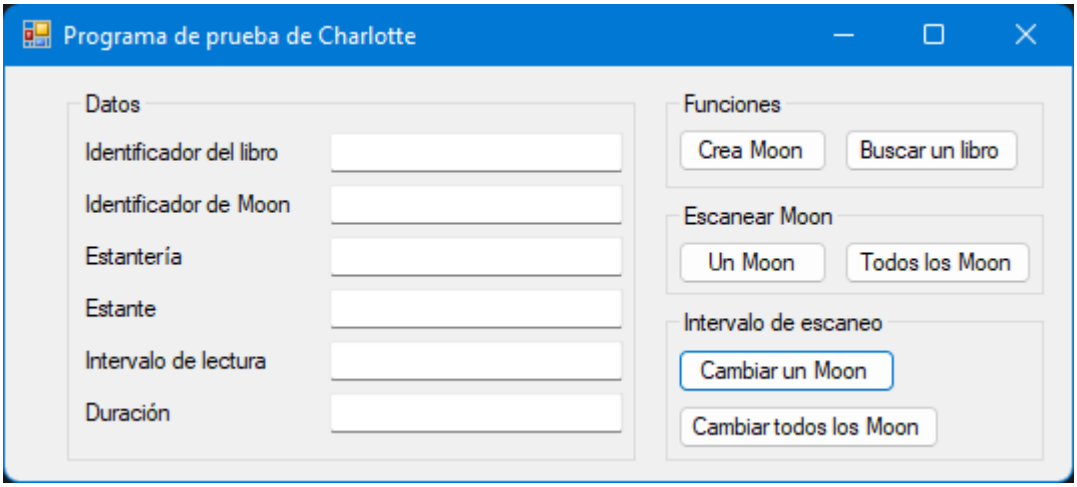


*Ilustración 37 Programa de prueba del servicio web*

En la parte izquierda se escriben los datos necesarios para ejecutar el método correspondiente del servicio web pulsando uno de los botones de la derecha.

Al pulsar el botón de Buscar un libro, si existe, aparecerá una ventana emergente con la localización del libro. Si no existe mostrará un mensaje indicándolo. El resto de botones no muestran ninguna indicación visual de su ejecución.

# 8. Circuito impreso

Uno de los objetivos de este proyecto es utilizarlo en mi propia biblioteca, por este motivo he diseñado una placa de circuito impreso de Moon.

Como se mostraba anteriormente el prototipo inicial de Moon es:

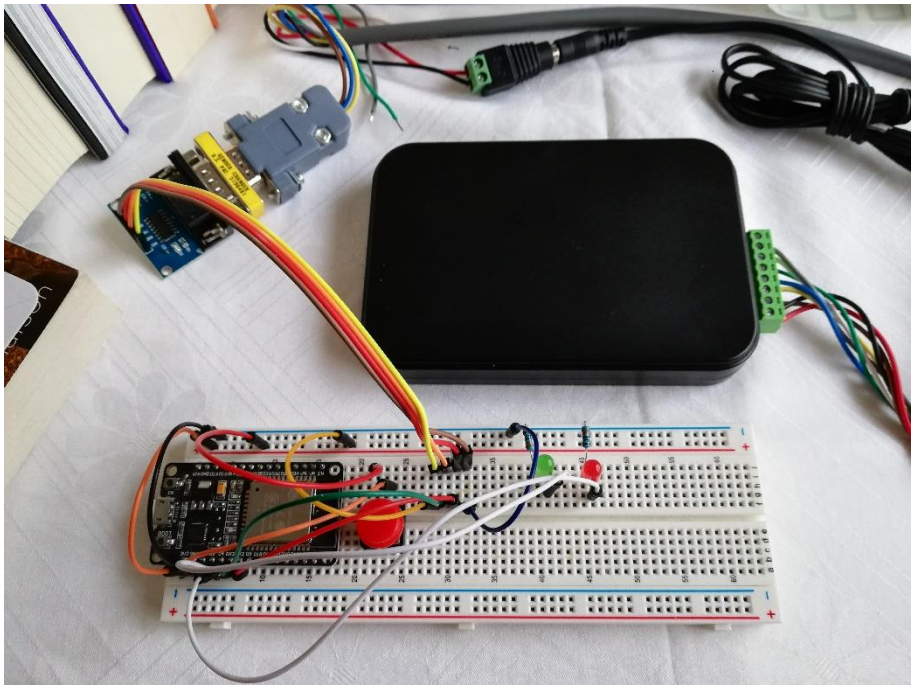

*Ilustración 38 Prototipo inicial de Moon*

Para el diseño del circuito impreso he utilizado la página web de EasyEDA [18]. Hay que crear un usuario para realizar el diseño. No hace falta descargarse nada, se hace todo online.

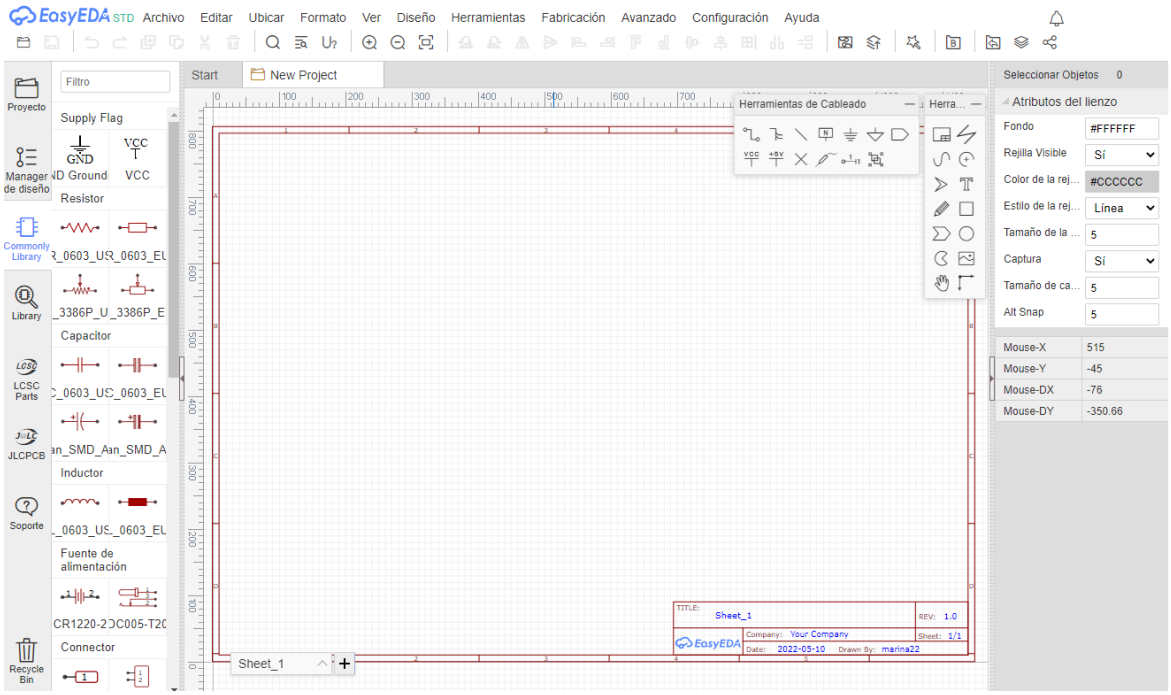


*Ilustración 39 EasyEDA*

Su uso es muy sencillo y se puede usar sin experiencia previa. Consiste en arrastrar y soltar componentes y unirlos con líneas o con puertos etiquetados. Tiene opciones para verificar que no hay errores en el diseño y librerías de componentes de los propios fabricantes y de los usuarios.

## 8.1. Esquema

Para el esquema de Moon se ha utilizado un microcontrolador ESP32, LED, condensadores, un circuito integrado para convertir los niveles de tensión del ESP32 a RS-232. Está basado en el esquema de un adaptador RS-232 de lasselukkari[6]. La conversión de los niveles de señal entre el microcontrolador y la antena RFID se realiza con el circuito MAX3232 [23]. El esquema final de Moon se muestra en la siguiente figura:

[6] Fuente: GitHub - lasselukkari/SerialChiller: A wireless RS232 adapter for the Internet of old things.

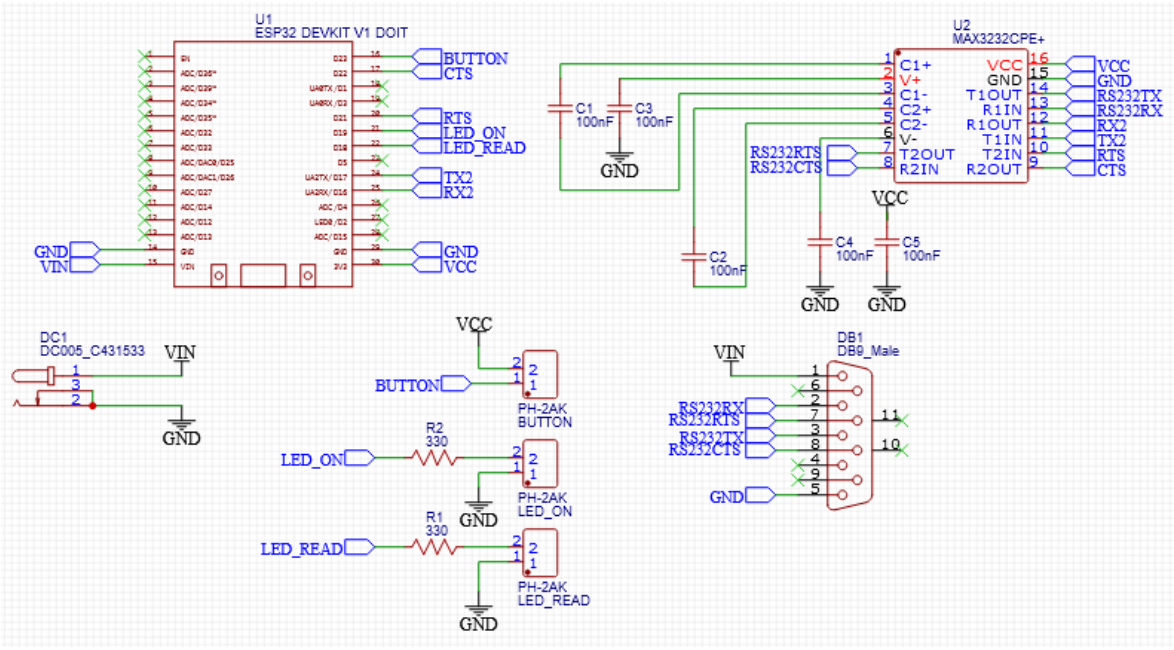


*Ilustración 40 Esquema de Moon*

En la siguiente figura se muestra el diseño dentro del entorno:

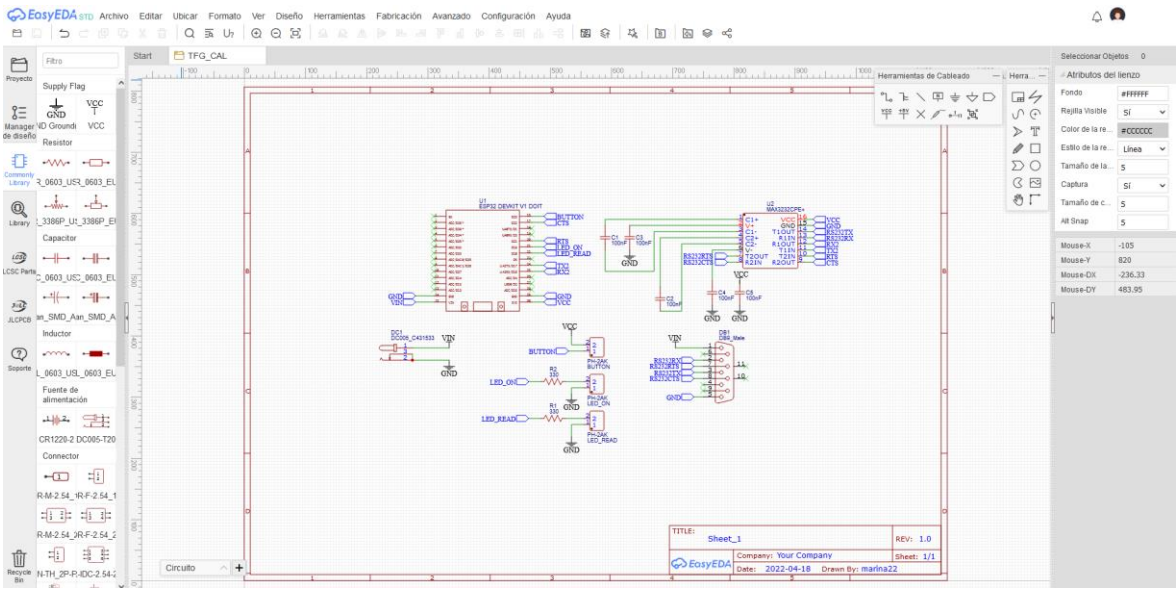


*Ilustración 41 Diseño de Moon en EasyEDA*

## 8.2. PCB

Una vez que el esquema está terminado se genera el PCB en una nueva pestaña con una vista del circuito impreso final. En la siguiente figura se muestra el PCB generado por el entorno sin ordenar los componentes y sin las pistas de cobre:

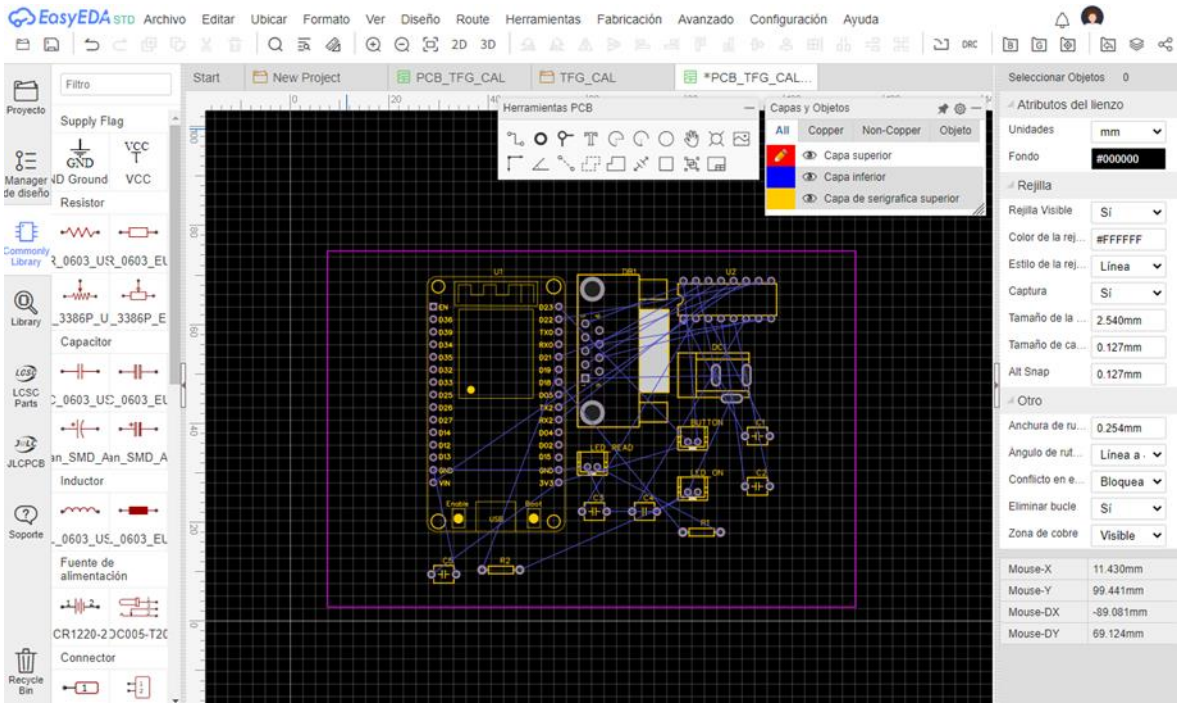


*Ilustración 42 PCB de Moon generado por EasyEDA*

En este paso se colocan los componentes, se giran y añaden los textos. Una vez colocados los componentes hay que generar las pistas de cobre con la opción AutoRoute. Para que funcione hay que descargar y ejecutar EasyEDA Router. En la siguiente figura se muestra el diseño del PCB terminado:

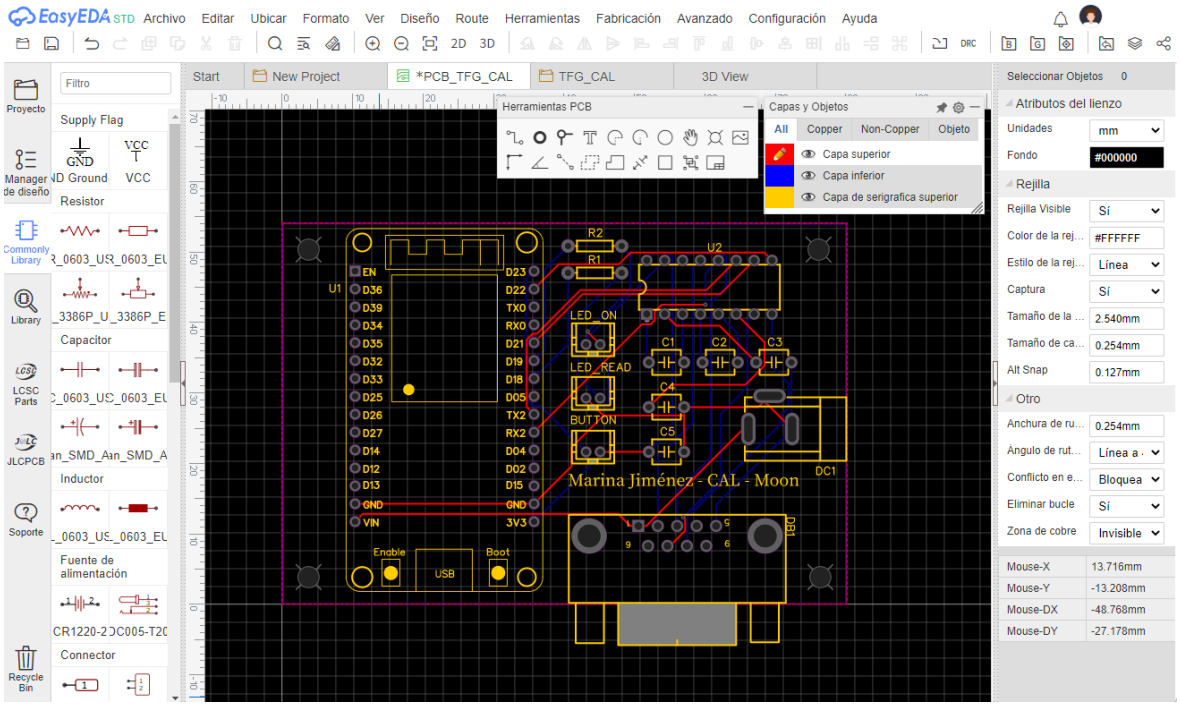


*Ilustración 43 Diseño PCB de Moon terminado*

Hay una opción que permite colocar una capa de cobre en la parte superior e inferior del PCB. En las siguientes figuras se muestra el PCB terminado con la capa de cobre:

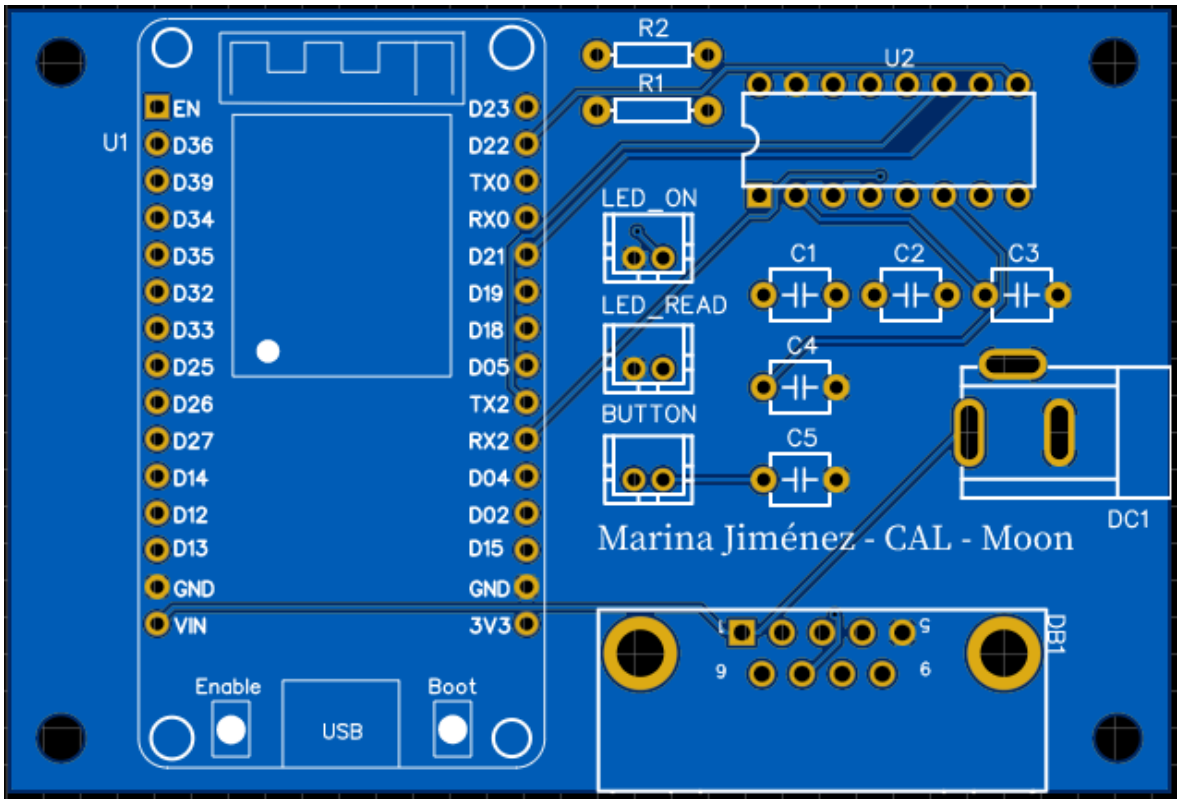


*Ilustración 44 Vista de la cara superior del PCB terminado*

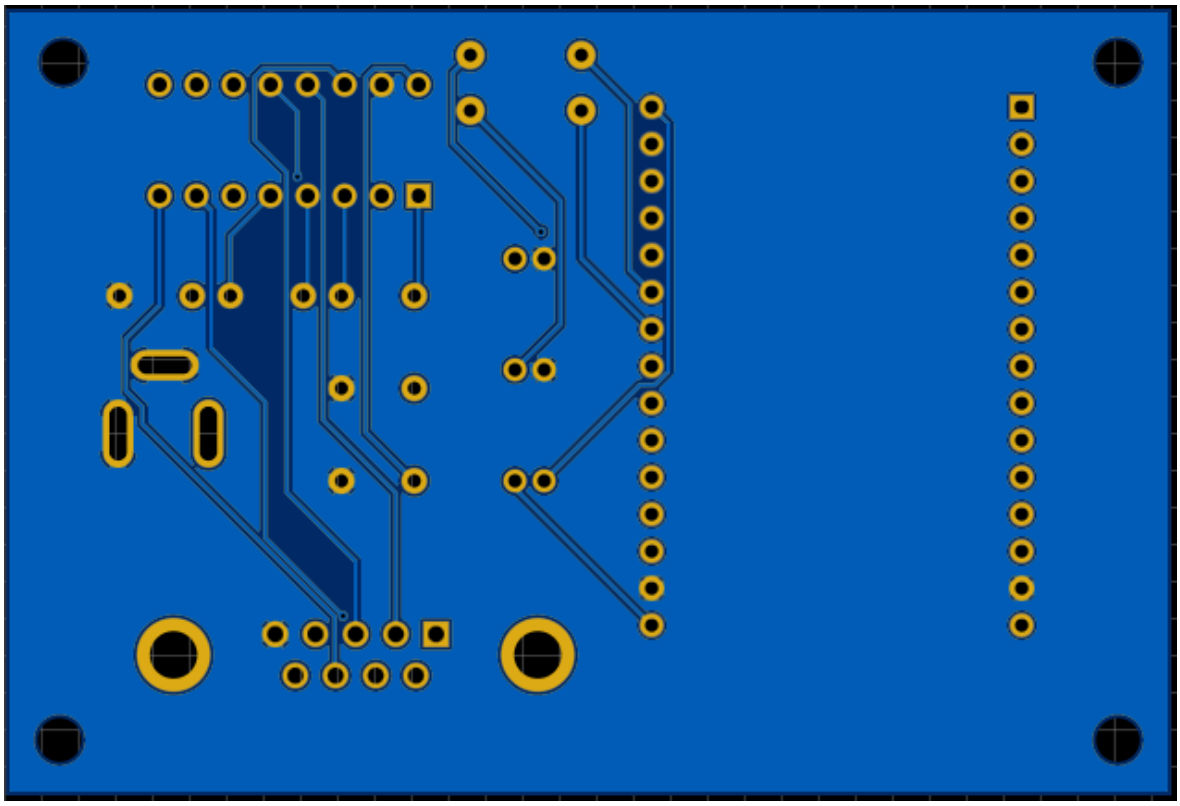

*Ilustración 45 Vista de la cara inferior del PCB terminado*

Tiene una opción para mostrar una vista en 3D con el diseño final.

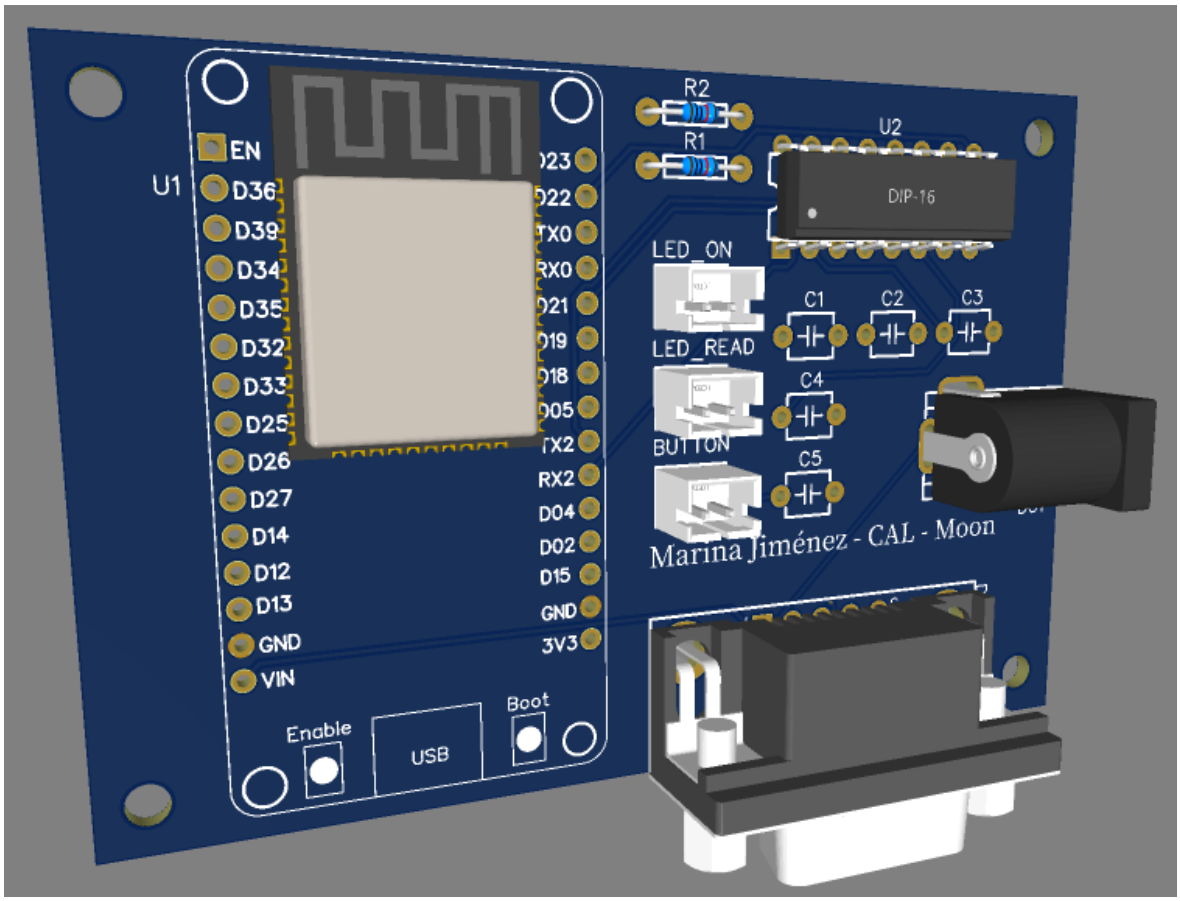


*Ilustración 46 Vista 3D del PCB terminado*

Los condensadores de la ilustración anterior no se muestran porque el tipo de condensador elegido no tiene vista 3D.

## 8.3. Fabricación

Para encargar la fabricación del circuito impreso es necesario generar un fichero Gerber a partir del PCB. En la propia página de EasyEDA se puede generar y enviar el fichero Gerber para su fabricación y compra.

## 8.4. Producto final

El pedido mínimo es de cinco placas de circuito impreso como se muestra en la figura:

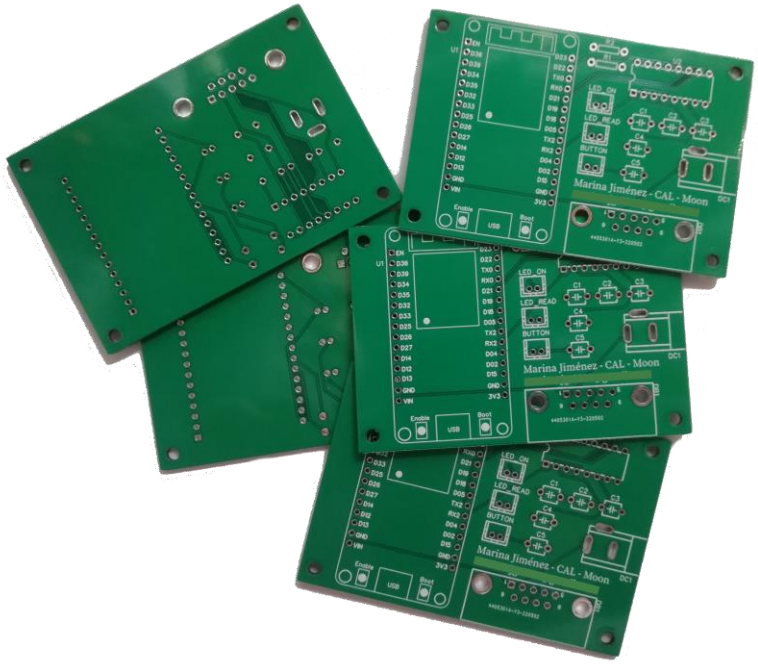


*Ilustración 47 Placas de circuito impreso de Moon*

La lista de componentes utilizados según el esquema es:

*Tabla 15 Lista de componentes*

| Nombre | Elemento | Modelo | Cantidad |
|---|---|---|---|
| PH-2AK | BUTTON,LED_ON,LED_READ | CONN-TH_PH-2A | 3 |
| 100nF | C1,C2,C3,C4,C5 | CAP-TH_L4.0-W3.5-P5.00-D0.5 | 5 |
| DB9_Male | DB1 | DSUB-TH_DMR-9P | 1 |
| DC005_C431533 | DC1 | DC-IN-TH_DC005 | 1 |
| 330 | R1,R2 | R_AXIAL-0.3 | 2 |
| ESP32 DEVKIT V1 DOIT | U1 | DOIT-ESP32-DEVKIT-V1 | 1 |
| MAX3232CPE+ | U2 | DIP-16_L19.8-W6.5-P2.54-LS7.6-BL | 1 |

Los componentes antes de montar la placa se muestran a continuación:

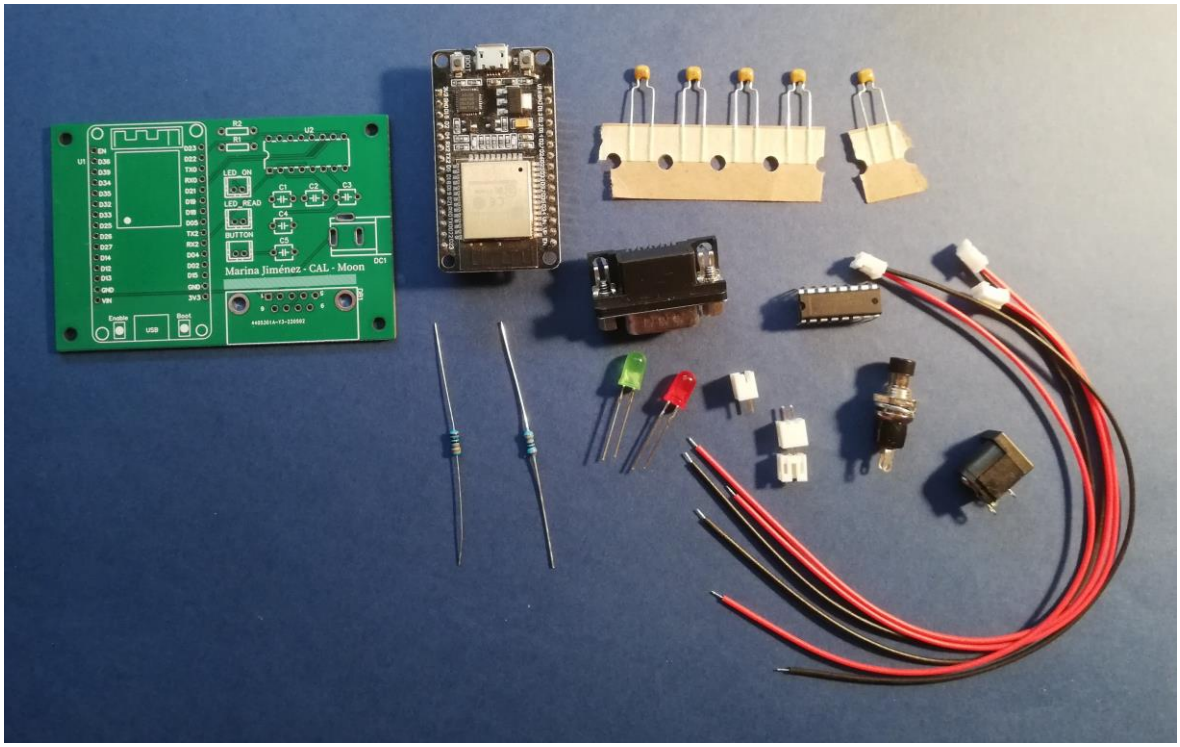


*Ilustración 48 Componentes*

La placa final montada:

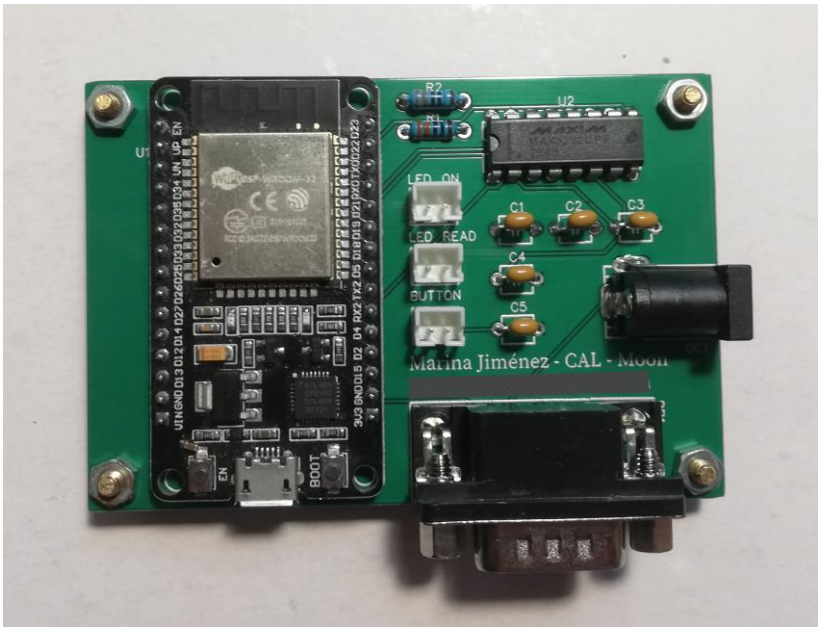


*Ilustración 49 Vista de la placa*

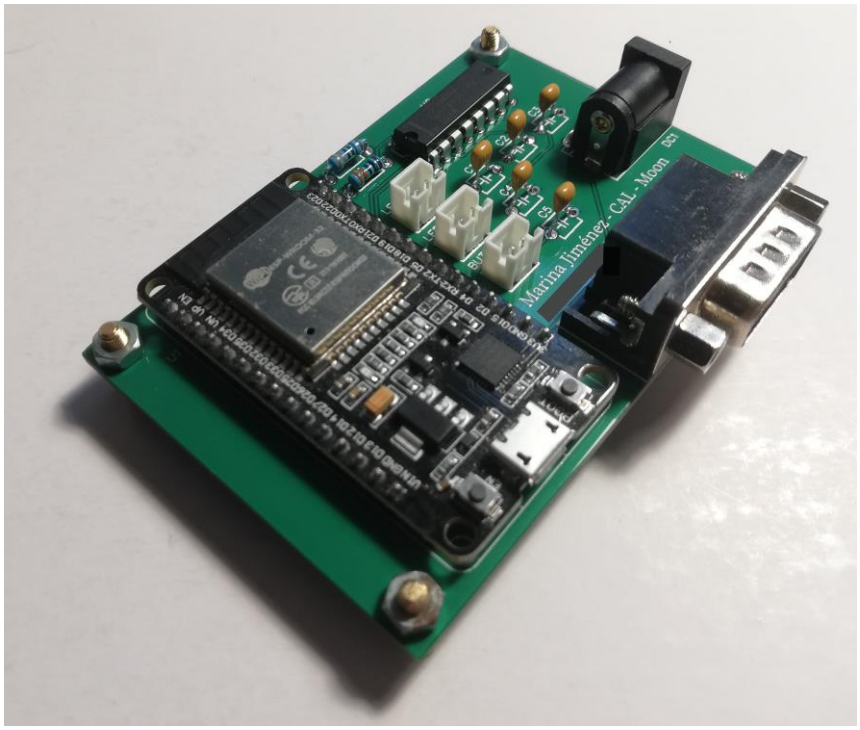


*Ilustración 50 Otra vista de la placa*

El dispositivo Moon terminado se muestra en la figura:


*Ilustración 51 Dispositivo Moon terminado*

# 9. Conclusiones y trabajo futuro

El proyecto nace de mi idea inicial para organizar mi biblioteca personal y evoluciona a un proyecto mayor para utilizar en bibliotecas grandes. El proyecto CAL se compone de varios dispositivos Moon que detectan las etiquetas de los libros de la biblioteca y envían esta información al ordenador central que tendrá instalado Charlotte donde hay un servicio web con el que interacciona el programa de la biblioteca.

Siguiendo las directrices de Ingeniería del Software se escribieron unos requisitos a partir de los cuales se generaron unos casos de uso que permitieron la escritura del código que posteriormente se probó con el prototipo de Moon. Más adelante se diseñó el circuito y la placa de circuito impreso que se mandó a fabricar y se montó para obtener el dispositivo Moon final.

Con las pruebas realizadas se ha verificado el correcto funcionamiento de CAL y esto facilitará el trabajo de un bibliotecario porque el sistema automáticamente detecta qué libros están en la biblioteca y dónde se encuentran independientemente de que estén o no en la estantería correcta. Cuando un usuario retire un libro de la estantería, en la siguiente lectura Moon enviará la lista de los libros actuales a Charlotte y se dará de baja en la base de datos. En el caso de que un usuario coloque el libro en la estantería se hará el proceso inverso. Con este proyecto se ha logrado crear un sistema de reservas e inventario en bibliotecas sin contacto.

La siguiente lista es una compilación de ideas para trabajo futuro:

- La creación de un plugin para las plataformas más usadas en las bibliotecas para que traduzcan sus peticiones a lo que espera recibir Charlotte.
- Detectar que un libro no está en su estantería.
- El bibliotecario puede ver en tiempo real el movimiento de libros.
- Al consultar un libro el bibliotecario puede saber cuándo lo han cogido de la estantería.
- Si no se reciben datos de un módulo Moon durante cierto tiempo se intenta conectar y configurar y si falla se muestra un aviso.
- Cuando estén todos los Moon instalados hay que definir los vecinos de cada uno en la base de datos.
- Barrido inteligente
    - Cada módulo Moon realizará un barrido y comprobará si ha habido cambios. Si hay cambios enviará solo dichos cambios.
- Lista inteligente. Los libros que estén en el límite del alcance de un Moon pueden ser detectados en un barrido y no en el siguiente. Este libro estará dentro del alcance de un Moon vecino (saldrá en la base de datos en dos Moon diferentes cuando ambos lo lean).

También evita dar de alta un libro de la biblioteca que tenga un usuario cercano en el momento de hacer el barrido. Para evitar dar de alta y baja repetitivamente un libro en un Moon se plantea:

  - Cada módulo Moon dará de alta un libro si lo encuentra en tres barridos sucesivos.
  - Cada módulo Moon dará de baja un libro si no lo encuentra en tres barridos sucesivos.

- Adaptar a Charlotte para otros sistemas operativos.
- Que un Moon pueda gestionar varias antenas

# 10. Conclusions and future work

The project is born from my initial idea to organize my personal library and it evolves into a larger project to use in big libraries. The CAL project is made up of several Moon devices that detect the tags of the library's books and send this information to the central computer that will have Charlotte installed, where there is a web service with which the library program interacts.

Following the guidelines of Software Engineering, some requirements were written from which some use cases were generated that allowed the writing of the code that was later tested with the Moon prototype. Later on the circuit and the printed circuit board were designed and ordered to be manufactured and assembled to obtain the final Moon device.

With the tests carried out, the correct functioning of CAL has been verified and this will facilitate the work of a librarian because the system automatically detects which books are in the library and where they are, regardless of whether or not they are on the correct shelf. When a user removes a book from the shelf, at the next reading Moon will send the list of current books to Charlotte and will be removed from the database. In the event that a user places the book on the shelf, the reverse process will be done. With this project, it has been possible to create a contactless booking and inventory for libraries.

The next list is a compilation of ideas for future work:

- The creation of a plugin for the most used platforms in the libraries so that they translate their petitions to what Charlotte expects to receive.
- Detecting when a book isn’t in its shelf.
- The librarian can see the movements of the books in real time.
- When checking on a book the librarian will be able to know when it was taken from the shelf.
- If it’s been a while since receiving data from a Moon module, it tries to connect and configure and if it fails a warning is displayed.
- When all the Moon are installed, the neighbours of each Moon have to be defined on the data base.
- Intelligent sweep
    - Each module will do a sweep and it will check is there was any changes. If there are changes it will only send those changes.
- Intelligent list. The books that are at the limit of the range of a Moon can be detected in a sweep and not in the next one. This book will be within reach of a neighbour Moon (it will show up in two different Moon in the data base when they both read it). It also avoids

registering a book from the library that a nearby user has in the moment of the sweep. To avoid repeatedly registering and deregistering a book in a Moon, it is proposed:

    - Each Moon module will register a book if they find it in three consecutive sweeps.
    - Each Moon module will deregister a book if they can't find it in three consecutive sweeps.

- Adapt Charlotte for other operating systems.
- A Moon can manage several antennas.

# 11. Bibliografía

[1] «Charlotte Lux | Tardis | Fandom,» [En línea]. Available: ttps://tardis.fandom.com/wiki/Charlotte_Lux.

[2] «The Library | Tardis | Fandom,» [En línea]. Available: https://tardis.fandom.com/wiki/The_Libra

[3] «Doctor Moon | Tardis | Fandom,» [En línea]. Available: https://tardis.fandom.com/wiki/Doctor_Moon.

[4] «Internet of Things (IoT) Products & Solutions - Cisco,» [En línea]. Available: https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html.

[5] «Bionix Technologies,» [En línea]. Available: https://bionixtechnologies.com/tecnologia-rfid/.

[6] «RFID: Soluciones y Etiquetas RFID, Industria 4.0, Trazabilidad (dipolerfid.es),» [En línea]. Availab https://www.dipolerfid.es/.

[7] «Lector RFID,» [En línea].

[8] «WiFi Alliance,» [En línea]. Available: https://www.wi-fi.org/.

[9] «Interface Circuits for TIA/EIA-232-F (Rev. A),» [En línea]. Available: https://www.ti.com/lit/an/slla037a/slla037a.pdf.

[10] «0-80cm short read range cheap 860Mhz-960Mhz uhf rfid usb reader writer - UHF RFID Reader - Innod, UHF RFID Reader ,RFID sports Timing system with software,UHF RFID Tag ,UHF RFID Antenna (innod-rfid.net),» [En línea]. Available: http://www.innod-rfid.net/productshow.php?cid=6&id=98.

[11] «esp32-wroom-32_datasheet_en.pdf (espressif.com),» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[12] «esp32_technical_reference_manual_en.pdf (espressif.com),» [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en

[13] «Arduino - Home,» [En línea]. Available: https://www.arduino.cc/.

[14] «Visual Studio Code - Code Editing. Redefined,» [En línea]. Available: https://code.visualstudio.co

[15] «A professional collaborative platform for embedded development · PlatformIO,» [En línea]. Available: https://platformio.org/.

[16] «Visual Studio: IDE y Editor de código para desarrolladores de software y Teams (microsoft.com) [En línea]. Available: https://visualstudio.microsoft.com/es/.

[17] «Plataforma de datos de Microsoft | Microsoft,» [En línea]. Available: https://www.microsoft.com/es-es/sql-server/.

[18] «GitHub - OttoWinter/AsyncTCP: Async TCP Library for ESP32,» [En línea]. Available: https://github.com/OttoWinter/AsyncTCP?utm_source=platformio&utm_medium=piohome.

[19] «GitHub - OttoWinter/ESPAsyncTCP: Async TCP Library for ESP8266,» [En línea]. Available: https://github.com/OttoWinter/ESPAsyncTCP?utm_source=platformio&utm_medium=piohome.

[20] «GitHub - esphome/ESPAsyncWebServer: Async Web Server for ESP8266 and ESP32,» [En línea]. Available: https://github.com/esphome/ESPAsyncWebServer.git?utm_source=platformio&utm_medium=piohom

[21] «Streaming | Arduiniana,» [En línea]. Available: http://arduiniana.org/libraries/streaming/.

[22] «EasyEDA Simulador de circuitos y diseño de circuitos impresos online,» [En línea]. Available: https://easyeda.com/.

[23] «MAX3232 3-V to 5.5-V Multichannel RS-232 Line Driver and Receiver,» [En línea]. Available: https://www.ti.com/lit/ds/symlink/max3232.pdf.

# ANEXO. Manual del administrador

CAL se podrá utilizar en las bibliotecas que dispongan de etiquetas RFID UHF EPC Gen2 – ISO180006C en todos los libros.

Una vez realizada la instalación los dispositivos Moon registrarán automáticamente todos los libros de la biblioteca en la base de datos de Charlotte y CAL estará listo para su funcionamiento.

## Instalación de Charlotte

Charlotte se instalará en un ordenador con sistema operativo Microsoft Windows, puede ser el de la propia biblioteca u otro que esté conectado a la misma red. Para instalarlo se ejecutará el programa Charlotte.msi y se seguirán sus instrucciones.

Habrá que configurar o modificar el sistema de la biblioteca para que conecte con el servicio web de Charlotte en el momento de reservar, consultar un libro, hacer una lectura manual o modificar el tiempo entre lecturas automáticas de los dispositivos Moon.

### Servicio web

El servicio web estará disponible en la dirección Charlotte.local:22308/WebService1.asmx. El archivo wsdl estará accesible en la URL <IP del ordenador>:22308/WebService1.asmx?wsdl. Dentro del archivo wsdl se describe todo el servicio web, las operaciones que tiene disponibles y las estructuras de datos. Mediante este archivo se creará o configurará el cliente de servicio web de la biblioteca.

La lista de métodos del servicio web es:

*Tabla 16 Métodos del servicio web*

| Nombre | Descripción | Parámetros |
|---|---|---|
| findBookLocation | Devuelve una lista de localizaciones de una etiqueta de un libro | El identificador del libro a buscar |
| createMoon | Da de alta un Moon en Charlotte | La localización del dispositivo Moon |
| moonScan | Ejecuta una lectura manual en un dispositivo Moon concreto | La localización del dispositivo Moon |

| | | |
|---|---|---|
| completeScan | Ejecuta una lectura manual en todos los dispositivos Moon | Ninguno |
| moonScanInterval | Cambia el intervalo de lecturas automáticas de un dispositivo Moon | La localización del dispositivo Moon |
| completeScanInterval | Cambia el intervalo de lecturas automáticas de todos los dispositivos Moon | El intervalo de lecturas |

### Alta de dispositivos Moon

El bibliotecario seleccionará dar de alta Moon en el programa de la biblioteca e introducirá los datos de identificación y localización de cada Moon instalado. El programa enviará los datos al servicio web que los dará de alta en la base de datos de Charlotte.

### Instalación de dispositivos Moon

Para alimentar Moon se necesita un adaptador de 9V de corriente continua y 1,5A. Es necesario conocer las credenciales de la red WiFi de la biblioteca. Cada Moon tiene un identificador numérico diferente. El nombre de cada Moon en la red será "Moon_identificador.local" donde identificador es el número asignado a cada Moon.

El instalador tendrá que instalar cada Moon en los estantes a cierta distancia unos de otros para que puedan leer todas las etiquetas de los libros entre todos. La separación entre los Moon dependerá del tipo de antena RFID utilizada. Con la antena actual la separación será inferior a un metro. Una vez instalados tendrá que configurarlos. Los parámetros que necesitan son el nombre del ordenador donde está Charlotte, su identificador y las credenciales del WiFi al que debe conectarse.

La primera vez que se enciende Moon crea una red WiFi con nombre "MoonServer" y contraseña "12345678". El instalador debe conectar un navegador a este WiFi e ir a la página http://192.168.4.1. En el navegador aparecerá la página de configuración de Moon donde deberán rellenarse los datos de identificación del dispositivo, nombre del WiFi al que se debe conectar y su contraseña.

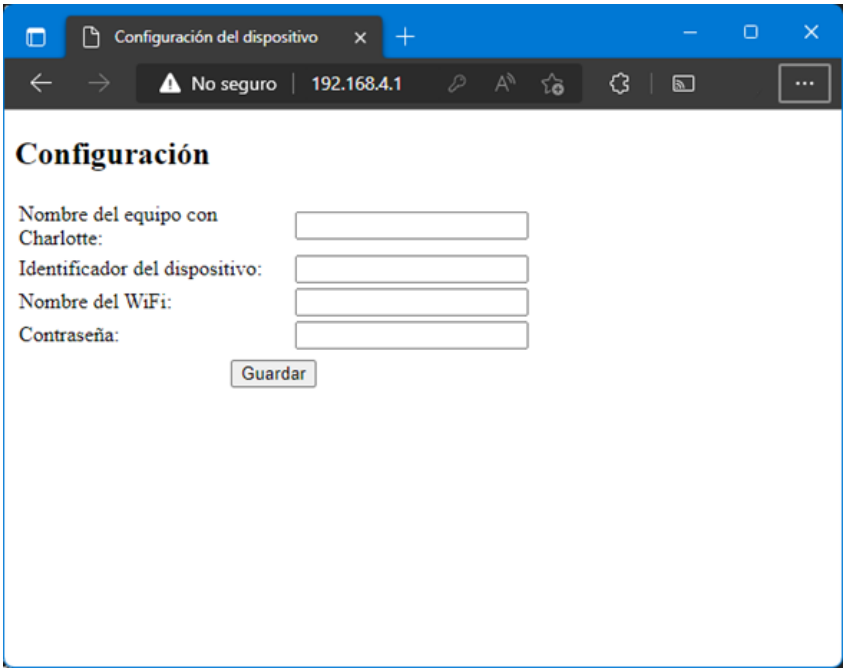


*Ilustración 52 Página de configuración de Moon*

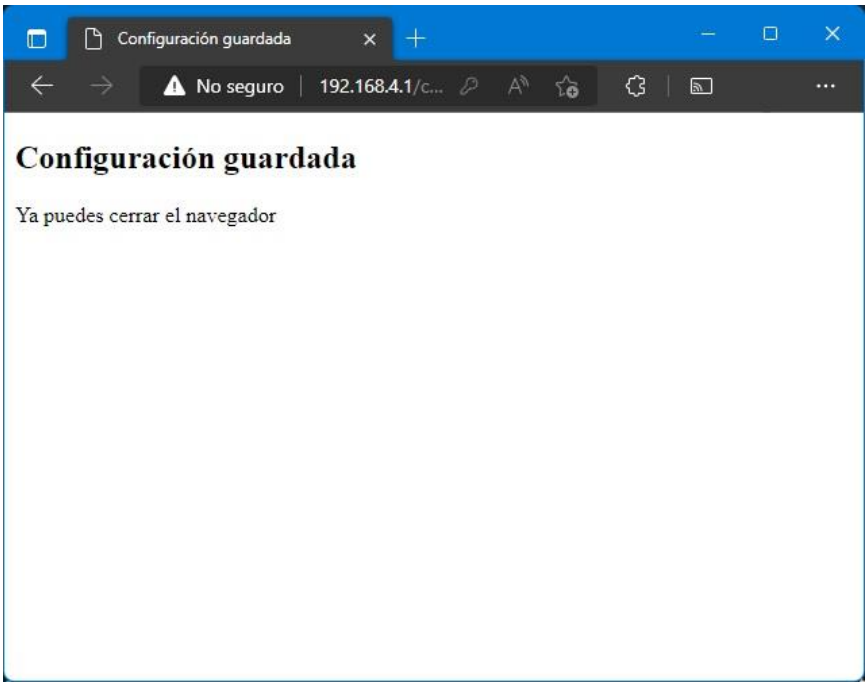


*Ilustración 53 Confirmación guardada correctamente*

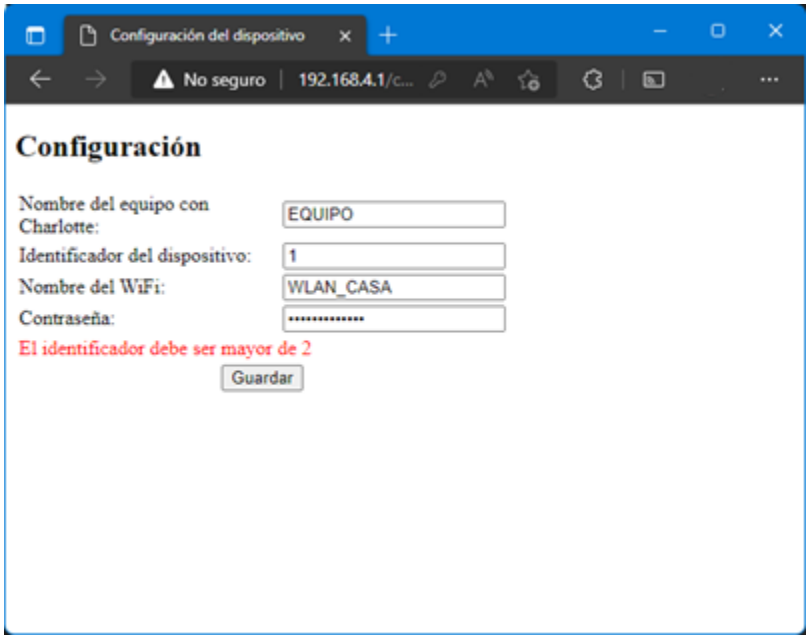


*Ilustración 54 Configuración con errores*

Una vez se introducen los datos Moon comprueba que sean válidos. Si hay errores en los campos mostrará un mensaje (si el nombre del equipo con Charlotte está vacío, si el identificador es menor que dos, si el nombre del WiFi tiene está vacío o si la longitud de la contraseña es menor de 8 caracteres). Una vez se introduzcan correctamente, Moon los guardará en la configuración para poder acceder a ellos en caso de que se reinicie. Para poder cambiar la configuración hay que pulsar el botón de Moon durante al menos diez segundos. Al pulsar el botón para cambiar la configuración se mostrará de nuevo la página web de configuración con los datos actuales para modificarlos. Al estar configurado si entra en modo configuración y no hay interacción del instalador vuelve a modo ejecución a los dos minutos. Si se quedara en modo configuración no realizaría ninguna lectura automática de etiquetas.