Universidad Zaragoza
1542

Trabajo Fin de Grado

Teacch-Fi: Multilector por RFID de pictogramas
Teacch-Fi: Pictograms RFID multireader

Autor/es

Millán Carrascosa, Alberto

Director/es

Torres Moreno, Enrique F.

Escuela de Ingeniería y Arquitectura
2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Alberto Millán Carrascosa,

con nº de DNI 73085390S en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Teacch-Fi: Multilector por RFID de pictogramas

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 24 de Noviembre de 2017

Fdo: Alberto Millán Carrascosa

RESUMEN

En las últimas décadas hemos podido ver como el precio de la tecnología se ha abaratado hasta el punto de que cualquier persona a día de hoy puede hacerse por poco dinero con dispositivos hardware capaces de competir con los mejores ordenadores de hace un par de décadas. A su vez, estos dispositivos han sufrido una reducción de su tamaño hasta el punto de que pueden ser colocados en cualquier objeto. Con ello surgió el concepto de internet de las cosas o IoT.

No obstante muchos de estos dispositivos comparten un problema común. No disponen de periféricos de entrada y/o salida (como teclados o pantallas) que les permitan ser configurados de forma sencilla. A estos dispositivos se les denomina descabezados o headless. Este hecho hace que sea crucial disponer elementos que nos permitan controlar de forma lo más automática y cómoda posible dichos dispositivos de tal forma que su configuración y utilización sean sencillos para el usuario final.

Este proyecto, consiste en un ejemplo de como es posible combinar varias tecnologías de comunicación para cumplir con esta tarea de una forma transparente y efectiva. Para ello, se ha elaborado un dispositivo aquí denominado TeacchFi que permite la lectura de distintas tarjetas con el propósito de transmitirlas a otros dispositivos.

Sumario

1Introducción....4
1.1Contexto....4
1.2Dispositivo TeacchFi....5
1.3Escenarios....6
1.4Objetivos....8
2Requisitos....10
2.1Requisitos funcionales:....10
2.2Requisitos no funcionales:....10
3Diseño....11
3.1Dispositivo Hardware....12
3.1.1Elección del microcontrolador....12
3.1.1.1NodeMCU....12
3.1.1.2 ESP8266-12E [4]:....13
3.1.2 RFID-RC522 [6]:....13
3.1.2.1SPI (Serial Peripheral Interface):....14
3.2Elementos software....15
3.2.1Entorno de desarrollo integrado (IDE) Arduino:....15
3.2.1.1 OTA [8]:....16
3.2.1.2SPIFFS:....16
3.3Comunicación....16
3.3.1Linea serie:....16
3.3.2WiFi:....17
3.3.2.1 Configuración:....17
3.3.3Persistent WiFiManager [9]:....17
3.3.4Descubrimiento....18
3.3.4.1 SSDP [10]:....18
3.3.4.2 DNS-SD [11]:....18
3.3.5 Escucha de clientes: protocolo MQTT [12]:....18
4Implementación....19
4.1Estructura del dispositivo....19
4.1.1Conexionado del hardware....19
4.1.2Librerias de “Zeroconf”....20
4.1.3Persistent WiFi Manager....20
4.1.4OTA (Over The Air)....21
4.1.5Lectura de tarjetas....22
4.2Comportamiento general del dispositivo....23
4.3Distribución de la memoria flash....25
4.4Aplicación Android....26
4.4.1Inicio de la aplicación....26
4.4.2Asociación de tarjetas....26
4.4.3Creación de una tarjeta....27
4.4.4Lectura de la tarjeta....27
5Pruebas....27
5.1Pruebas de conectividad....27
5.1.1Funcionamiento de DNS-SD y SSDP....27
5.1.2MQTT....28
5.2Pruebas con usuarios....28
6Conclusiones....29
6.1Tiempo dedicado....29

6.1.1Reuniones.....29
6.1.2Estudio y Diseño.....30
6.1.3Implementación y Evaluación.....30
6.1.4Memoria.....30
6.2Librerías.....30
6.3Licencias.....30
6.4Opiniones personales.....31
7Anexos.....31
7.1Enlaces.....31

1 Introducción

1.1 Contexto

Las nuevas tecnologías están llegando a cada parte de la sociedad, y están demostrando ser de gran ayuda a la hora de romper un gran número de barreras sociales. Para personas con Trastorno del Espectro Autista (en adelante TEA) resulta complicado comunicarse, ya que su condición neurológica les afecta en como sienten e interactúan con el mundo que les rodea.

La metodología TEACCH (Treatment and Education of Autistic and Related Communication Handicapped Children) consiste en una forma de enseñanza especial creada para tratar con personas con TEA, la cual se basa en la estructuración completa de las actividades realizadas por el alumno. Esta metodología utiliza de forma periódica y sistemática los pictogramas como forma de comunicación, haciendo que el alumno se sienta cómodo al usarlos para expresarse con sus educadores.

En la imagen siguiente se muestra un aula típica de la metodología Teacch. Como se aprecia en esta imagen, el ordenador no siempre tiene por qué estar cerca del área de trabajo, de ahí la importancia de contar con una tecnología inalámbrica que permita comunicar el tablero con el ordenador.

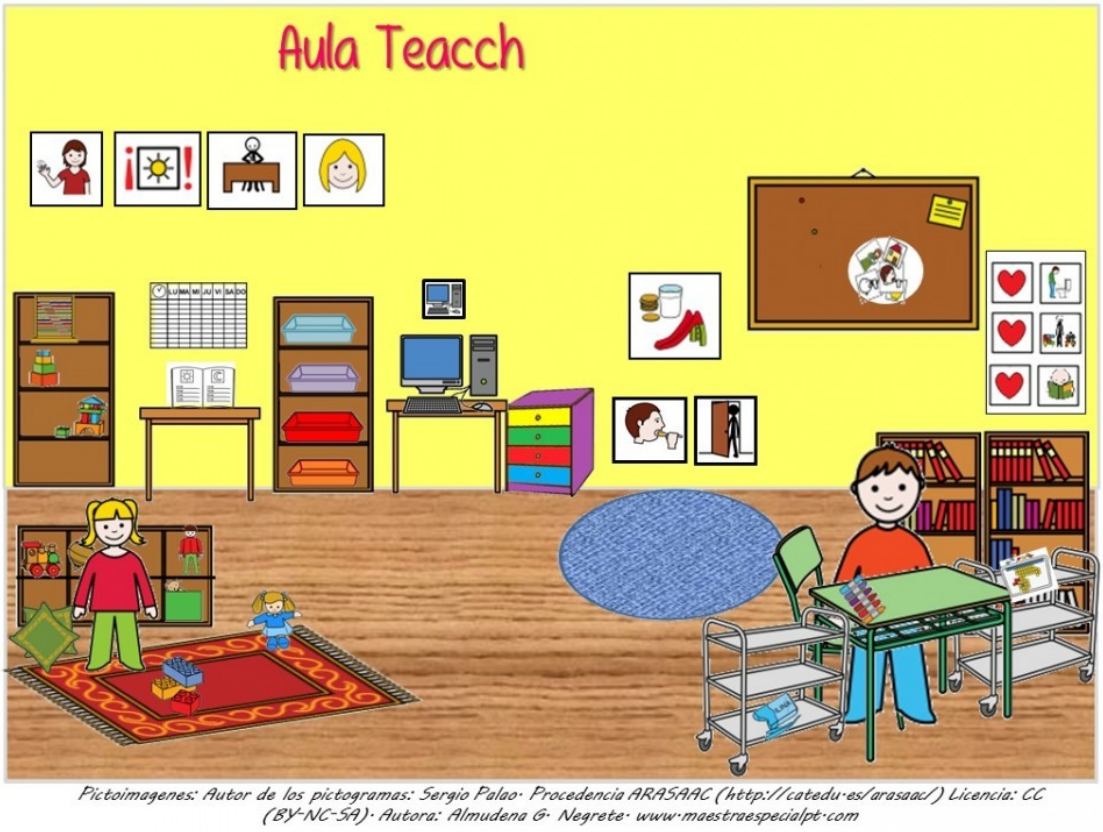

Ilustración 1: Ejemplo de una clase descrita en la metodología Teacch. La autoría de la foto se puede ver en la propia imagen.

La idea principal de este proyecto es dotar de mayor versatilidad a una de esas tecnologías. Agregándole mecanismos de comunicación y autoconfiguración que faciliten su utilización, además de la posibilidad de comunicarse inalámbricamente.

La herramienta en cuestión se trata del TeacchFi, un dispositivo multi-lector de tarjetas RFID que permite comunicarse con otras herramientas software para su posterior procesado de las mismas, bien mostrándolas y/o leyéndolas.

Esta herramienta se ha ido desarrollando junto al colegio publico de educación especial Alborada de Zaragoza dentro de un proyecto de colaboración con el grupo de investigación de arquitectura de computadores de Zaragoza (gaZ).

El proyecto fue realizado durante la primavera del 2017. Se evaluaron distintos diseños, tamaños de fichas, formatos de dispositivo. Durante el proceso se implemento un prototipo de electrónica cuyo componente principal se trata del micro esp8266. Se realizó esta elección dado que se trata de una pequeña placa de bajo coste que cuenta con conectividad WiFi.

A lo largo de la presente memoria se abordarán los detalles más relevantes sobre los elementos clave en la elaboración del firmware dedicado a la comunicación mediante WiFi.

1.2 Dispositivo TeacchFi

El dispositivo creado (en adelante TeacchFi) se trata de un tablero portátil capaz de leer mediante tecnología RFID distintas tarjetas identificadas de forma unívoca mediante un ID interno. Cada una de estas tarjetas cuenta con un pequeño pictograma que representa una porción de información (inicialmente palabras) que ayuda al usuario a comunicarse con sus educadores.
Para facilitar su uso, el dispositivo ha de ser capaz de comunicarse con distintos dispositivos de mayor complejidad con independencia de su naturaleza (tales como ordenador, táblet, móvil, etc) o sistema operativo. Para ello ha de utilizar protocolos estándar que permitan abstraerse, en la medida de lo posible, del dispositivo utilizado para interpretar la información generada.

El comportamiento de este dispositivo ha de permitir que el usuario final pueda colocar tarjetas sobre el tablero y estas sean interpretadas por uno o más dispositivo externos que estén conectados de forma inalámbrica al aparato.

A continuación se muestra dos imágenes donde se aprecia tanto el aparato como algunas de sus posibles formas de operar a nivel de red.

Ilustración 2: En este prototipo se muestra un ejemplo de lectura de tarjetas, las cuales son transmitidas a un ordenador.

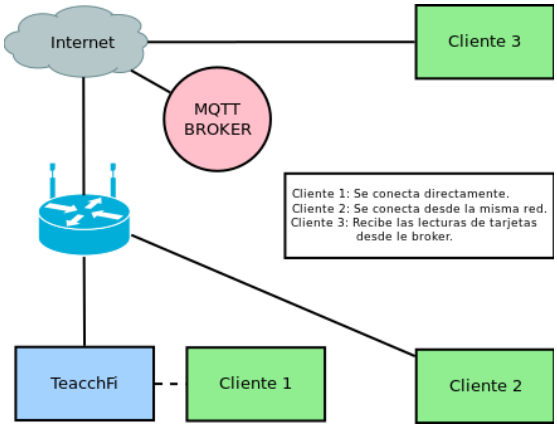

Ilustración 3: Diagrama de red con 3 de los posibles escenarios. Directamente conectado, en la misma red y MQTT.

En la primera de las imágenes, se puede ver el TeacchFi en marcha. En este caso, se encuentra en plena comunicación con el ordenador, transmitiendo cada tarjeta leída así como su número de bahía. En la pantalla del ordenador se puede apreciar la interpretación de las mismas.

En la segunda, se puede observar sobre un mismo esquema, algunos de los posibles escenarios en los que se puede encontrar operando en TeacchFi, donde hay un cliente directamente conectado al aparato, otro conectado por TCP desde la misma red local y un tercero accediendo mediante un broker intermediario de MQTT.

Para llegar a esta situación es necesario primero enfrentarse a varios problemas ¿Cómo puede conocer el dispositivo el SSID y contraseña de la WiFi a la que se conecta? ¿Cómo sabe el ordena que hay un dispositivo al que conectarse? ¿Qué métodos existen para poder configurar este dispositivo? Esta y otras cuestiones son las que se abordan a lo largo de esta memoria.

Se debe contar con que el usuario encargado de administrar el dispositivo se trata de profesorado de un colegio. Por tanto el dispositivo cuenta con unas dimensiones adecuadas para su transporte entre aulas así como de cierta autonomía mediante el uso de pilas.
Además dado que el usuario final es profesorado y no necesariamente informático, el TeacchFi cuenta con mecanismos que lo hacen fácil de configurar.

1.3 Escenarios

Este proyecto pese a haberse realizado en un entorno de laboratorio bajo condiciones optimas, ha sido pensando para un entorno escolar, donde la configuración de la WiFi puede no ser la más óptima para el funcionamiento de TeacchFi. Bajo un entorno escolar, es fácil encontrarse en situaciones donde la configuración de la infraestructura de red esté pensada para que cada dispositivo opere de forma aislada, impidiendo que

los mecanismos de autodescubrimiento aquí implementados no funcionen como esperaríamos. O es más, pueden darse escenarios donde no exista infraestructura de red a la que conectarse. Es por ello que es necesario plantear distintas situaciones posibles con sus respectivas soluciones.

Escenario 1: Primer encendido del aparato.

Antecedente: Una vez el TeacchFi ha sido adquirido / construido, este no cuenta con ninguna configuración base que le permita operar con la red final, por tanto se requiere de algún mecanismo que permita realizar una configuración inicial antes de comenzar a ser utilizado.

Solución: En el caso de que al encender el TeacchFi este detecta que no es capaz de conectarse a ninguna red, existen varias soluciones posibles. La primera de ellas consiste en crear su propia infraestructura de red, en la cual coloca un portal cautivo donde levantar una pequeña página de configuración donde poder introducir todos los posibles parámetros necesarios. Este portal cautivo se lleva a cabo mediante una librería externa denominada Persistent WiFi Manager.

La segunda solución, consistiría en disponer de tarjetas especiales RFID. Dado que estas tarjetas cuentan con un mínimo de capacidad de almacenamiento, es posible grabar en ellas una serie de parámetros de configuración tales como por ejemplo SSID, contraseña y dirección del servidor MQTT. Estos parámetros serían cargados en el dispositivo si por ejemplo se dejase colocada dicha tarjeta durante el arranque del dispositivo. El protocolo de configuración que permite esta configuración se trata de NDEF.

La tercera opción se trataría de utilizar el protocolo WPS, el cual permite que durante un periodo corto de tiempo tanto AP como TeacchFi puedan comunicarse e intercambiar credenciales.

Escenario 2: No hay conexión de red.

Antecedente: Es posible que se de la situación en la que no contemos con una infraestructura de red o bien no contemos con la posibilidad de acceder a ella. Por tanto resulta necesario poder construir una ad-hoc que nos permita operar con el TeacchFi.

Solución: Existen 3 posibles soluciones a este problema.
La primera consistiría en utilizar directamente la conexión USB del dispositivo, lo cual permitiría disponer de una única conexión pero muy fiable y fácil de utilizar.

La segunda consistiría en que el propio TeacchFi crease dicha infraestructura, esta opción cuenta con la ventaja de que serviría tanto para el caso 1 como para este. No obstante, tiene como inconveniente el no poder tener de acceso a otras redes donde podría situarse el broker MQTT.

La tercera opción consistiría en que fuese el cliente en que crease la red ad-hoc. En el caso de un teléfono móvil podría consistir una opción denominada Tethering que permitiría compartir la conexión de red del terminal, facilitando de esta manera el acceso a otras redes.

Escenario 3: Publicación en la red LAN

Antecedente: Una vez conectado con éxito a la red local, el resto de posibles clientes tienen que saber en que dirección de la red se encuentra el TeacchFi para poder comunicarse con él.

Solución: Para llevar a cabo esta tarea existen varios mecanismos y posibles soluciones. En el caso de la red local, existen 2 especialmente importantes: SSDP y DNS-SD. Ambos protocolos operan a nivel de red local dentro del dominio de broadcast. Estos dos protocolos son perfectos para operar en situaciones en las cuales no se posee internet, como por ejemplo cuando se está usando la WiFi del TeacchFi como infraestructura de red.

Escenario 4: Red LAN aislada (con internet).

Antecedente: Dado que la red a la que se tenga que conectar el TeacchFi puede ser la de un colegio, es posible que esta red posea mecanismos de aislamiento por cuestiones de seguridad, esto puede impedir que el TeacchFi no sea capaz de publicarse a sus clientes, ya que estos pese a estar en la misma red, no comparten dominio de broadcast.

Solución: Para solucionar este nuevo problema, existe MQTT, el cual además permite que el TeacchFi se pueda comunicar con los clientes mediante el uso de un broker externo. Este mecanismo exige que tanto el TeacchFi como cada uno de los posibles clientes conozcan previamente dicho broker.

1.4 Objetivos

El objetivo de este proyecto es el de estudiar e implementar distintos mecanismos de comunicación y autoconfiguración para poder hacer que un dispositivo inalámbrico que no dispone de periféricos de entrada o salida (el TeacchFi) pueda ser configurado y utilizado bajo distintas tipos de infraestructura de redes. Además el TeacchFi ha de ser capaz de leer y transmitir las distintas tarjetas RFID que se usen con el.

Para la elaboración de este proyecto se han tenido en cuenta los siguientes puntos:

- Que se pueda comunicar con teléfonos móviles tablets y ordenadores:

Dado que el principal objetivo del dispositivo a elaborar es el de leer tarjetas, el primero de los objetivos a cumplir es poder comunicarse con otros dispositivos. Para ello existen distintas tecnologías disponibles a tener en cuenta. Comunicación serie por cable, Bluetooth o WiFi. Posteriormente se explicara las ventajas y desventajas de cada una y cuales han sido las adoptadas.

- Que disponga de mecanismos para cambiar la configuración inicial:

Dado el que el TeacchFi se trata de un dispositivo que no cuenta con periféricos de tipo monitor o teclado (headless) su configuración puede resultar complicada, sobre todo la inicial en el caso de la WiFi. Por tanto el dispositivo ha de contar con los mecanismos necesarios para poder introducir de forma sencilla una configuración inicial.

- Que disponga de mecanismos de descubrimiento:

Dado que en una red IP los distintos hosts no tienen por qué conocerse, ha de existir algún mecanismo que sin contar con información o con información limitada, permita a los usuarios del dispositivo poder localizar a este dentro de la red. Estos mecanismos tienen que ser conocidos por ambos y en algunos casos han de permitir poder saltar entre redes.

- Que pueda comunicarse con otros dispositivos:

El dispositivo ha de ser capaz de transmitir la información generada al resto de dispositivo que se encuentren pareados con este, bien estando en la misma red o bien estando en redes separadas.

- Múltiples clientes:

El dispositivo ha de permitir la conexión simultanea de más de un cliente a la vez, de tal manera que toda la información generada sea recibida por todos los clientes de forma lo más simultanea posible.

- Ha de usar software libre:

Ya que es un dispositivo orientado a la enseñanza, se espera que dicho dispositivo pertenezca al ámbito del software libre, de esta manera cualquier persona que lo desee pueda utilizar, modificar y distribuir el trabajo aquí realizado.

En la próxima sección se introducirán los distintos requisitos planteados para este proyecto, los cuales surgen de los objetivos planteados en este punto.

Durante la sección de diseño se irá exponiendo como se han llevado a cabo cada uno de esos requisitos y se presentarán las tecnologías finalmente adoptadas.

Este punto se verá ampliado en el de implementación donde se explicará algunos de los detalles más relevantes del desarrollo del TeacchFi.

Finalmente a lo largo de las pruebas y conclusión se irá mencionando como se han ido cumpliendo los distintos objetivos planteados al inicio de esta memoria.

2 Requisitos

En este apartado se detallará una lista con todos los requisitos que se pretende cumplir en este proyecto, así como descripción de cada uno de ellos.

Tras lo expuesto en los puntos anteriores, se ha decidido que el TeacchFi ha de cumplir con la siguiente lista de requisitos para poder realizar su función.

2.1 Requisitos funcionales:

- RF1- Ha de poder leer tarjetas en las diferentes bahías.
  Consiste en poder determinar que tarjeta ha sido colocada en que bahía y determinar cuando ha sido retirada de ella.
- RF2- Ha de poder comunicar las tarjetas a otros dispositivos.
  Una vez una tarjeta ha sido identificada, ha de poder enviarse mediante red usando la tecnología apropiada en cada caso.
- RF3- Ha de permitir conexiones de varios usuarios.
  Estas conexiones han de poderse gestionar de forma simultanea, de tal forma que todos los usuario reciban la información independientemente de que mecanismos usen para conectarse al aparato.
- RF4- Ha de ser fácil de configurar como para que un usuario con conocimientos básicos sobre el dispositivo sea capaz de utilizarlo.
- RF5- Que se publique dentro de la red.
  Una vez el TeacchFi se ha conectado a la red, ha de poder ser localizado y alcanzado por el resto de dispositivos.

2.2 Requisitos no funcionales:

- RNF1- Ser compatible con Arduino.
  Dado el controlador que se va a emplear, es importante poder reutilizar buena parte del material que ya hay disponible en la red. En su mayor parte para la plataforma de Arduino. Además se desea que el dispositivo quede para el dominio del software libre, lo cual es compatible con las librerías que se suelen manejar en el entorno de Arduino.
- RNF2- Ha de poder conectarse con otras redes.
  Si se da el caso de poder contar con una infraestructura de red inalámbrica

(WiFi), ha de poder comunicar con otros dispositivos mediante esta tecnología con independencia de si se encuentran conectados a la misma red local o no.

- RNF3- Ha de poder comunicarse vía serie.
  De no poder contar con una comunicación inalámbrica, ha de poder seguir usándose el dispositivo mediante su conexión USB. No obstante se espera que el principal uso de este medio sea principalmente para desarrollo y depuración.
- RNF4- Ha de poder publicitarse en la red para permitir su descubrimiento. Dado que puede no recibir una dirección de red fija o conocida, ha de poder localizarse el dispositivo fácilmente dentro de la red.
- RNF5- Ha de ser reconfigurable.
  Una vez el dispositivo cuente con una configuración inicial, debe de existir algún mecanismo que permita reconfigurar el dispositivo de tal forma que le permita actualizar alguno de sus parámetros.

3 Diseño

En este capítulo se detallará algunos de los aspectos más importantes relacionados con las decisiones tomadas para su diseño, tales como: El protocolo de comunicación diseñado para transmitir las tarjetas, el proceso de descubrimiento del dispositivo, formas de configurarlo, errores contemplados y sus soluciones.

Se ha descartado el uso de lectores ópticos como por ejemplo una cámara por varios motivos. El primero de ellos es porque el proceso de identificación de la tarjeta puede entorpecer la experiencia del usuario. Pedirle a un niño con TEA o incluso con algún tipo de discapacidad motora, que sostenga la tarjeta sin ocluir o bloquear la visión del pictograma, puede no ser viable.

El segundo de estos motivos, es que requiere de una solución más costosa, tanto en hardware como en recursos. Añadir una cámara puede hacer subir el precio sensiblemente al dispositivo. Por otro lado, procesar una imagen requiere de mayor capacidad de computo que otras soluciones. No obstante quizá el problema principal sea las complejidades que aporta este tipo de tecnología, ya que incluir una cámara obliga a tener que estar haciendo re-calibraciones que dependan del entorno de luz con el que se disponga. Problemas adicionales para el usuario final con el que no cuentan otras tecnologías como RFID.

Por ello se ha optado por usar tecnología RFID. Esta tecnología permite la fácil lectura de la tarjeta añadiendo una única pegatina a la parte inferior de la tarjeta. A efectos del usuario, esta pegatina es imperceptible, ya que al estar por debajo, no entorpece su experiencia de uso. Cuenta con una desventaja importante, hay que dar previamente de alta cada tarjeta, además de que hay que incluir el coste de dichas pegatinas al coste final. No obstante, este proceso de alta de tarjetas es ajeno al propósito de este proyecto,

ya que su misión no es interpretar las tarjetas, sino transmitir su existencia. El uso de la tecnología RFID hace mucho más sencillo el diseño del dispositivo a la vez de hacerlo más portable.

En este trabajo se ha valorado tanto la posibilidad de utilizar como tecnología de comunicación inalámbrica de medio alcance, tanto bluetooth como WiFi. Ambas dos resultan tecnologías de sobra conocidas y perfectamente validas para este proyecto, no obstante, se ha decidido finalmente utilizar WiFi por los siguientes motivos;

WiFi al tratarse de una tecnología ampliamente extendida y disponible en la gran mayoría de dispositivos portables de la actualidad, resulta una muy buena opción.

3.1 Dispositivo Hardware

Como se ha comentado anteriormente, el dispositivo se compone de dos elementos principales. Los lectores RFID y el NodeMCU.

Estos dos elementos deben poder comunicarse para poder transmitir la información de las tarjetas a un dispositivo externo. Esta comunicación debe llevarse a cabo de manera que un mismo controlador pueda comunicarse con varios lectores RFID.

3.1.1 Elección del microcontrolador

En el amplio mercado de los micro-controladores, hay una gran cantidad de dispositivos que podrían encajar en este proyecto. No obstante, como pieza principal, se ha optado por utilizar el NodeMCU, una pequeña placa de bajo coste que cuenta con WiFi incorporado además de comunicación por USB. Se ha decidido utilizar este en concreto porque es ampliamente soportado por las librerías de Arduino así como tener un coste muy bajo (menos de 3€).

Además hay otra pieza clave que compone la solución creada es el RC522, una pequeña placa RFID que permite la fácil lectura y escritura de tarjetas NFC a frecuencias de 13.56MHz. Esta placa también cuenta con soporte para Arduino así como un bajo coste, lo cual la hace ideal para este proyecto.

3.1.1.1 NodeMCU

NodeMCU [3] consiste en una placa de desarrollo IoT basada en el firmware eLua que incorpora como parte principal un SoC llamado ESP8266-12E. Además de esta pieza, cuenta con otras partes importantes como son: Un regulador de corriente por si se desea alimentar mediante los pines de Vin y GND, una conexión micro-usb que sirve tanto para alimentación como para conexión serie con el ordenador. En este aspecto hay por lo menos 2 versiones según que fabricante se elija, una con el chip "CH340" y otra con el "CP1202", ambos son igual de validos para reprogramar el dispositivo vía usb.

Además cuenta con 30 pines repartidos en dos hileras con distintas funcionalidades, como por ejemplo GPIOs (11 pines), ADC, SPI, etc. Con esta cantidad de pines y sin añadir ningún hardware adicional, resulta posible incluir hasta un total de 7 lectores RFID.

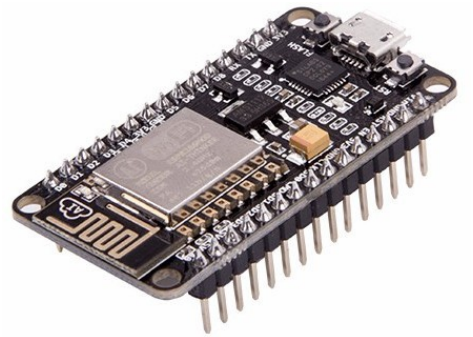

Ilustración 4: Placa integrada con chip de comunicación SoC ESP8266. La versión de la placa que se puede ver en la imagen corresponde a la versión con programador CP1202

3.1.1.2 ESP8266-12E [4]:

Este SoC se trata de la pieza más importante del NodeMCU. Inicialmente solo estaba pensada para ser programado usando como lenguaje de programación el lenguaje Lua, pero gracias a un "port" [5] se puede usar también el IDE de Arduino para desarrollar para este SoC. Este SoC cuenta con dos puntos fuertes. El primero de ellos es que es capaz de dotar de conexión wifi de 2.4GHz a nuestros proyectos además de tener un precio muy asequible. También cuenta con un procesador Tensilica L106 de 32-bit (de la empresa Espressif) y una memoria flash de 4MB, con lo que no será necesario añadir ningún componente de control adicional a nuestro proyecto.

3.1.2 RFID-RC522 [6]:

Se trata de un lector RFID de bajo coste (menos de 3€), alimentado con 3.3v y con capacidad para leer tarjetas NFC de 13,56 MHz. Cuenta con un chip Philips MFRC522. Cuenta tanto con comunicación I2C como SPI. No obstante solo está implementado SPI para arduino, lo cual hace que resulte muy sencillo el controlar varios lectores simultaneamente.

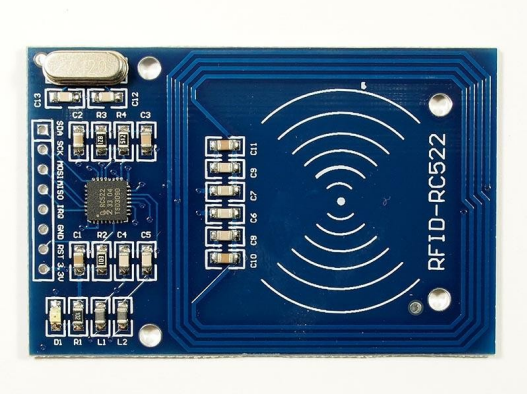

Ilustración 5: Placa lectora de tarjetas RFID para Arduino con chip MFRC522

3.1.2.1 SPI (Serial Peripheral Interface):

Se trata de la tecnología de comunicación que utilizan los lectores RFID. Esta comunicación es síncrona, full duplex. Cuenta con 3 líneas compartidas y una dedicada por cada lector. Estas lineas compartidas reducen el número de lineas necesarias, pero impiden que pueda usarse más de un lector de forma simultánea, lo cual limita los ciclos de lecturas de las tarjetas.

En la siguiente imagen se puede apreciar como es una configuración típica de varios dispositivos conectados a un mismo master.

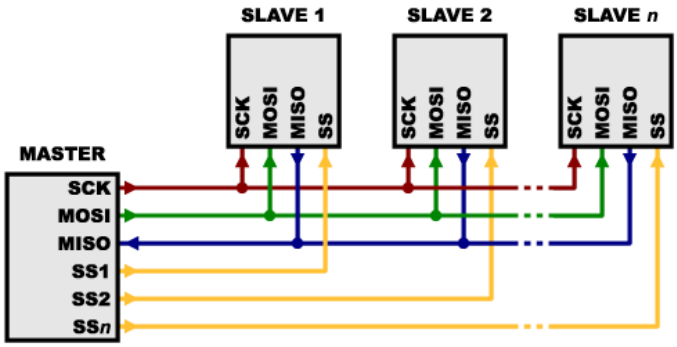

Ilustración 6: Esquema típico de conexión entre varios esclavos y un maestro

Tal y como está diseñada la librería en la que se basa este proyecto, es necesario añadir una linea de comunicación extra. Esta linea se trata de una linea reset que pueden utilizar los dispositivos esclavos para notificar al maestro si existe alguna tarea que requiera del bus, en este caso, la lectura de una tarjeta.

Dado que el ESP8266-E cuenta con hasta 11 GPIOs, se puede contar con un total de hasta 7 lectores funcionando simultáneamente sin la necesidad de añadir ningún tipo de hardware adicional. Una posible solución para poder liberar o decrementar el número de pins consumidos por los distintos dispositivos, sería el añadir un decodificador. Esto

sería posible ya que solo es posible utilizar un lector simultáneamente. De esta manera, podríamos controlar hasta 7 dispositivos utilizando tan solo 3 pines.

Con todo ello, queda afirmar que el barrido de las distintas bahías se realizará de forma secuencial, por lo que el tiempo de lectura de una tarjeta hasta que se comunica con los clientes dependerá directamente del número de bahías.

3.2 Elementos software

Además de distintas piezas de hardware utilizadas, se han utilizado varias piezas de software y protocolos. La totalidad de ellas pertenecientes al ámbito del software libre. Entre los elementos más importantes se encuentran los siguientes.

3.2.1 Entorno de desarrollo integrado (IDE) Arduino:

Se trata de un entorno de desarrollo multi-plataforma ampliamente conocido con una gran comunidad de usuarios que respalda su uso y que aportan constantemente nuevas librerías al proyecto.

Este entorno cuenta con la ventaja de que desde un mismo entorno es posible escribir el código, así como compilarlo, cargarlo en la flash del controlador y visualizar su ejecución mediante su consola de la línea serie.

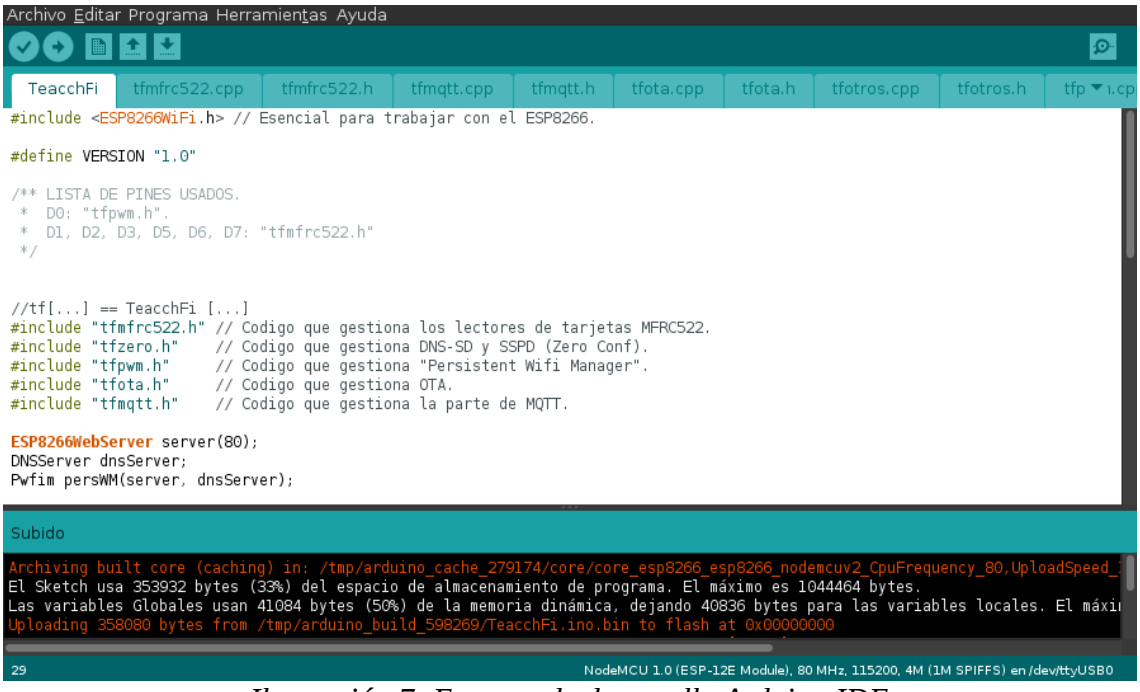

Ilustración 7: Entorno de desarrollo Arduino IDE

Como principal desventaja a su uso, es que no está pensado para depurar código haciendo ejecución paso a paso. Eso dificulta su utilización especialmente en proyectos

grandes, donde el uso de mensajes por consola se vuelve algo complejo para poder determinar donde puede encontrarse los fallos en el diseño del código.

A continuación se mostrará las librerias que se han utilizado así como las decisiones que han llevado a considerarlas necesarias para este proyecto.

3.2.1.1 OTA [8]:

Over The Air. Se trata de un sistema de actualización de software mediante comunicación inalámbrica. En este proyecto, se ha utilizado especialmente en la fase de desarrollo como forma de actualizar el firmware del ESP8266. Este método de carga del código binario presenta la importante ventaja de no necesitar cables así como la mayoer velocidad de esta frente al uso de la linea serie. De forma interna, esta librería hace uso de la librería de mDNS para comunicar el puerto y la IP al IDE de Arduino.

3.2.1.2 SPIFFS:

SPI Flash File System. Se trata de una librería que permite utilizar parte de la memoria flash como si fuese un sistema de ficheros persistentes, donde poder almacenar las paginas web o algunos parámetros. Es importante mencionar que el NodeMCU solo cuenta con 4MB de memoria para todo el dispositivo, por lo que este sistema de ficheros cuenta con unas limitaciones de espacio y cantidad de ficheros a tener en cuenta. No obstante dado que los únicos ficheros a almacenar son las páginas web del portal cautivo y las configuraciones de la WiFi, resulta más que suficiente la capacidad disponible para tal propósito.

3.3 Comunicación

En este apartado se hará un repaso a las dos vías de comunicación con las que contará el TeacchFi. Ambas resultarán indispensables para el proyecto, pero sin lugar a dudas la vía de comunicación inalámbrica es la más importante, por tanto es la que más dedicación se le ha dado.

3.3.1 Linea serie:

Es la forma principal de comunicación entre el nodeMCU y el IDE de Arduino. Gracias a su conexión por USB, se podrán cargar los “sketch”.

Pese a no ser el objetivo principal del proyecto, es indispensable poder contar con una linea serie de comunicación que permita tanto el desarrollo como la depuración del código. Es por ello que el NodeMCU se trata de una pieza clave, ya que este cuenta con un puerto microUSB que facilita las tareas de alimentado de la placa, volcado del firmware, al igual que comunicación para poder hacer el debugging.

No obstante en el tema del volcado del firmware no resulta del todo práctico ya que resulta mucho más lento que la otra alternativa ya mencionada en puntos anteriores (OTA).

3.3.2 WiFi:

Como se ha mencionado en los casos de uso existen diferentes escenarios a los que hay que dar solución. Para ello es necesario combinar distintas librerías que permitan solucionar los siguientes problemas.

3.3.2.1 Configuración:

Pese a existir 3 posibles soluciones para la configuración inicial de la WiFi mencionadas en el escenario 1 del punto 1.3, solo se ha llegado a implementar la del portal cautivo. Los motivos que han llevado a descartar las otras 2 son:

Para poder realizar la solución de NDEF, es necesario contar con un lector que permita escribir una tarjeta con la información necesaria. Pese a que a día de hoy existen dispositivos móviles capaces de realizar tal acción, es cada vez más frecuente que esta funcionalidad no esté presente en dichos dispositivos, por lo cual no parece una buena opción a la hora de implementarlo.
La opción de WPS, cuenta con el principal problema de que no permite realizar configuraciones específicas, por tanto hay parámetros, tales como la URL del broker MQTT, que no pueden ser introducidos por este mecanismo, haciendo insuficiente su uso. Además dado que el entorno en el que se espera utilizar el TeacchFi se trata de un entorno escolar, es posible que en muchos casos ni siquiera se tenga acceso al AP, algo necesario para poder iniciar el proceso de configuración ya que suele ser habitual que se requiera de pulsar un botón en el AP. No obstante la librería de Persistent WiFi Manager cuenta con dicha funcionalidad de serie por si llegase a ser útil.

Una vez el dispositivo se ponga en marcha, este desconoce por completo cual es la configuración de la red a la cual tiene que conectarse para dar servicio. Por tanto es necesario que se encienda inicialmente como AP para poder permitir así su configuración. Esto es posible gracias a la librería de *Persistent WiFi Manager* [9] que se encarga de gestionar el portal de configuración. Una vez se configuren todos los parámetros, el dispositivo se encontrará en condiciones de acceder a una red compartida.

3.3.3 Persistent WiFiManager [9]:

Esta librería esta pensada para facilitar el acceso y configuración de las credenciales para poder acceder a una wifi. Además de esto, cuenta con la posibilidad de añadir

paginas web que serán servidas durante la ejecución del programa. Todo ello se logra con la inclusión de varias librerías más como por ejemplo la de SPIFFS.

3.3.4 Descubrimiento

Una vez que el dispositivo se encuentra dentro de una red, es necesario que el resto de dispositivos dentro de esta red puedan conocer su presencia. En la realidad existen principalmente dos estandares comunmente utilizados. Estos dos estandares se tratan de SSDP y de DNS-SD.

Tanto SSDP como DNS-SD son dos protocolos que entran dentro de la categoría de "ZeroConf".

Dado que ambos protocolos son compatibles y pueden convivir dentro de la misma red, se ha optado por incluir ambos protocolos en su diseño, de esta manera se espera podre llegar a un mayor número de posibles clientes.

3.3.4.1 SSDP [10]:

Es un protocolo de descubrimiento que utiliza notificaciones de tipo HHTP. En este caso se utilizará para publicitar el TeacchFi dentro de la red local. De esta manera se comunicará a todos los potenciales clientes en que ip y puerto se encuentra el dispositivo. Este protocolo es especialmente útil para clientes con sistemas Windows.

3.3.4.2 DNS-SD [11]:

Al igual que el protocolo SSDP, este también vale para el descubrimiento del dispositivo en la red. Este protocolo hace uso de la librería de mDNS ya mencionada previamente en OTA. Se considera que resulta más sencillo de implementar por utilizar DNS en lugar de HTTP.

3.3.5 Escucha de clientes: protocolo MQTT [12]:

Dado que se desea poder atender a más de un cliente de forma simultanea, parece poco razonable que sea el TeacchFi quien se conecte a otros dispositivos. En su lugar serán los propios clientes quienes se conecten al TeacchFi. Esto hace mucho más fácil la tarea de servir tarjetas a los clientes, ya que es el propio TeacchFi quien gestiona esa lista de clientes. No obstante el comportamiento mediante MQTT es distinto.

MQTT funciona mediante un intercambio de mensajes de tipo clave valor, donde todos son clientes (incluido el propio TeacchFi). Previamente los distintos dispositivos han de subscribirse a un canal del broker. A la parte del mensaje de tipo clave se la denomina "topic". Este topic puede estar a su vez dividido en jerarquías. Estas jerarquías se separan mediante el carácter "/", de tal forma que una jerarquía podría ser "TeacchFi1/1". Esta división permitiría utilizar el mismo broker para más de un aparato.

4 Implementación

En este apartado se explicarán algunos de los detalles concretos de como se ha realizado las tareas marcadas durante las etapas anteriores. Esta implementación esta relacionada tanto con el dispositivo hardware, como la parte de firmware, como una herramienta Android especialmente creada para poder probar el dispositivo.

4.1 Estructura del dispositivo

Tal como se ha hecho en algunos de los apartados anteriores, se puede dividir el dispositivo en 2 partes.

4.1.1 Conexionado del hardware

El TeacchFi cuenta con dos elementos principales conectados por un bus SPI. Estos elementos como ya se ha mencionado son el nodeMCU y los lectores RFID-RC522.

En el siguiente diagrama se puede apreciar como se realiza este conexionado.

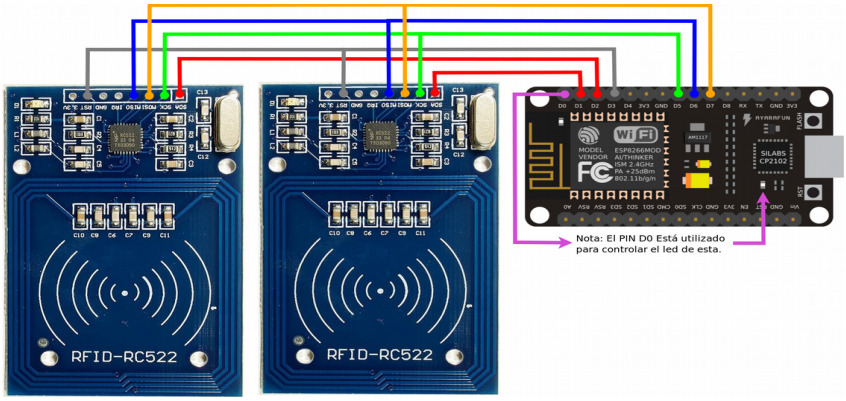

Ilustración 8: Conexionado de la placa con los lectores

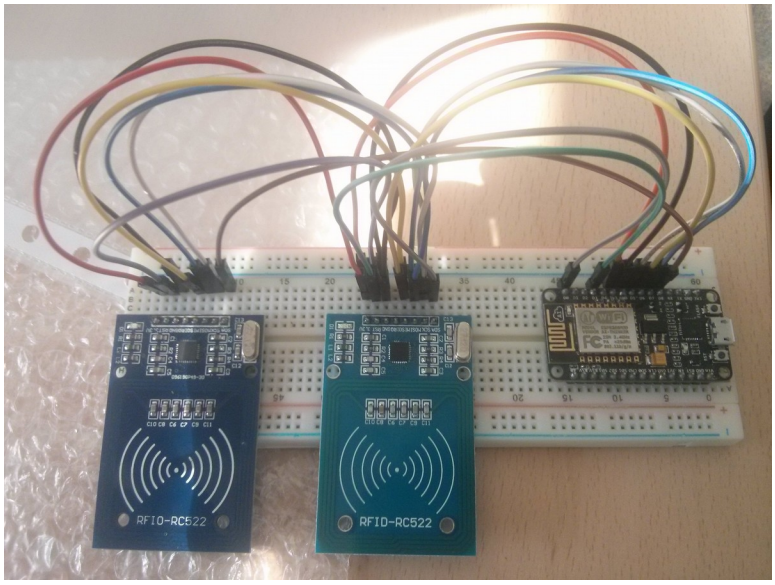

Ilustración 9: Prototipo real con las conexiones.

Como se aprecia en el esquema, cada uno de los lectores necesita de 5 conexiones, de las cuales 4 son compartidas. Estas conexiones son: SS (SDA según se muestra en la placa de los RFID), SCK, MOSI, MISO, RST.

Ilustración 10: En esta imagen se puede apreciar otro prototipo. En este caso se trata de un dispositivo con 7 bahías montadas sobre una PCB.

4.1.2 Librerias de “Zeroconf”

Los protocolos de mDNS y SSDP se implementan gracias a dos librerías existentes para Arduino. Su configuración y utilización resulta sencilla ya que unicamente hace falta inicializar al principio del código. Hay que destacar que la librería de OTA hace uso de mDNS para que el IDE de Arduino sea capaz de localizar al dispositivo. No obstante se ha optado por añadir además de ese mensaje, otro propio definido para identificar al TeacchFi.

Con estos dos protocolos podemos garantizar que al menos dentro de la red local cualquier software que se realice para complementar esta herramienta va a tener cierta facilidad para buscar este dispositivo.

4.1.3 Persistent WiFi Manager

Mediante esta librería se permite dotar al dispositivo de un medio para poder configurar de forma sencilla todos los parámetros necesarios para su funcionamiento, tales como SSID de la red WiFi a la que conectarse, contraseña así como otros parámetros concernientes a MQTT tales como: Dirección del servidor, puerto, contraseña y nombre de usuario.

El problema del resto de alternativas reside en que no permiten mantener el portal de configuración disponible a lo largo de toda la ejecución. Como contrapunto importante, esta librería no permite la configuración de un direccionamiento estático, por lo que estaremos sujetos a un direccionamiento dinámico por DHCP.

Esta librería cuenta con integración con el sistema de ficheros SPIFFS [14] el cual permite almacenar tanto las páginas web de configuración como cualquier otro parámetro que se necesite conservar entre ejecuciones del dispositivo.

Esta librería requiere de una activación constante a cada iteración del bucle principal, lo cual puede repercutir significativamente en el periodo de ejecución de cada iteración.

A continuación se muestra una captura de pantalla donde se aprecian algunas de estas configuraciones.

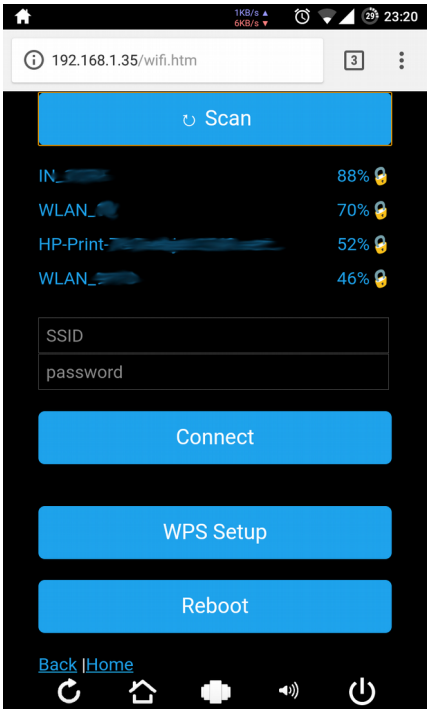

Ilustración 11: Portal de configuración de PWM desde un navegador en teléfono Android

Por último, con el propósito de dotar de cierta retroalimentación, se activará un led en la propia placa cuando el dispositivo esté en modo AP (sin conexión WiFi).

4.1.4 OTA (Over The Air)

Esta librería se ha utilizado a lo largo de todo el proceso de implementación para facilitar la actualización de las distintas versiones del códigos. Además, puede resultar útil a la hora de actualizar el dispositivo una vez esté terminado.

No obstante, añade algunas limitaciones importantes que hay que tener en consideración.

La primera de ellas es el amplio consumo de espacio necesario. Dado que es necesario que el dispositivo descargue en su memoria interna todo el binario que remplazará al existente, es necesario contar con al menos el 50% del espacio disponible para poder utilizarlo.

Otra desventaja es que al igual que la librería de Persistent WiFi Manager, requiere de una activación constante en cada iteración del bucle, lo cual añade un retardo. Este es menos considerable, pero relevante de igual forma.

Por todo ello, tal vez sería tener en buena consideración en no incluir esta librería en las etapas finales del desarrollo, con el objetivo de liberar al dispositivo de esta carga innecesaria.

4.1.5 Lectura de tarjetas

Se trata de un proceso en el cual se ha de realizar un barrido secuencial de cada una de las bahías de tarjetas disponibles, en el cual una vez se detecte un cambio en cualquiera de ellas, ha de almacenarse en un buffer de tarjetas a la espera de ser enviado a todos los clientes. Este proceso cuenta con algunas dificultades técnicas que ha habido que solventar.

La primera de ellas se trata de que la librería cuenta con un sistema de “desactivación” de la tarjeta hasta que esta sea retirada para evitar la lectura reiterada de una tarjeta, no obstante no cuenta con ningún mecanismo de poder identificar si la tarjeta ha sido retirada de la bahía, por lo que no es posible utilizar este mecanismo si se desea poder contar con esa información, por lo que ha sido necesario implementar un array con el estado de cada bahía a cada iteración del “loop”.

Otro problema, es que los lectores empleados no reconocen siempre la presencia de las tarjetas cuando estas siguen colocadas sobre el lector, por lo que en muchos casos producen la falsa apariencia de que una tarjeta ha sido retirada.
Para solucionar este problema, se ha añadido un contador de veces que ha fallado la lectura en una bahía determinada, de tal forma que si se llega a un umbral determinado de fallos consecutivos, se considera que la tarjeta ha sido realmente retirada. Este comportamiento se puede ver en el siguiente diagrama.

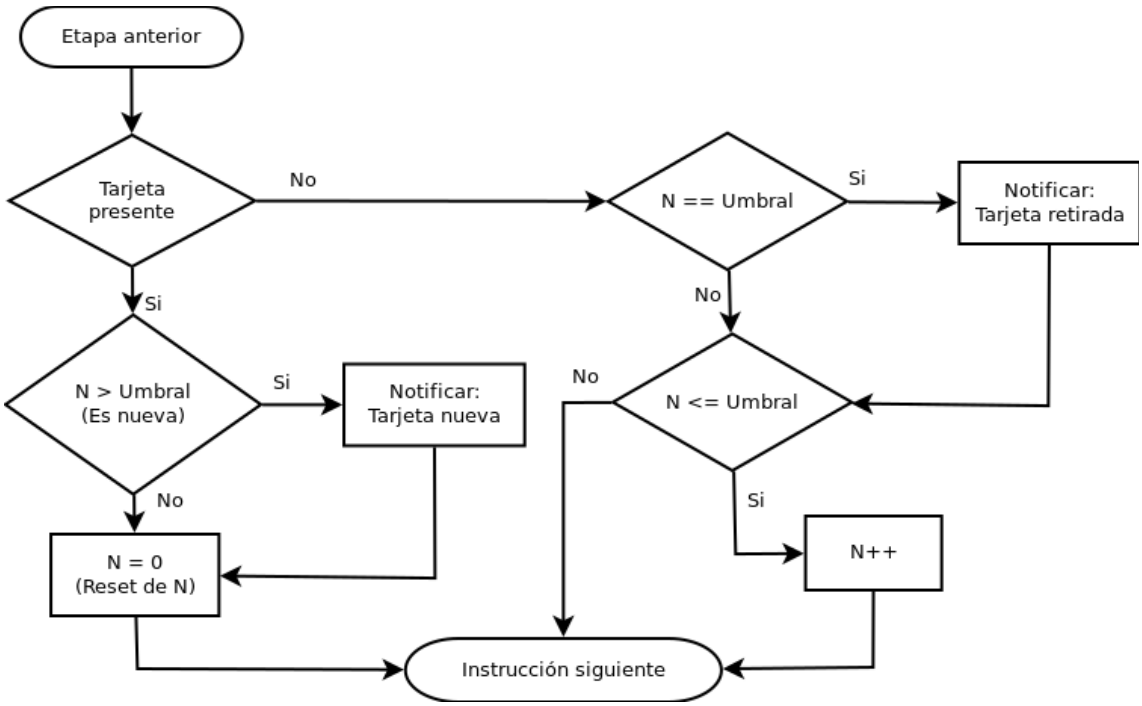

Ilustración 12: Proceso de control de errores en la lectura de tarjetas

Para la transmisión de las tarjetas leídas, se ha utilizado una sencilla concatenación entre el número de la bahía que identifica la tarjeta seguido del ID leido en hexadecimal. Ambos separados por el carácter ‘;’. Un ejemplo de tarjeta leída podría ser: “1;968f3413”. Donde “1” es la bahía y “0x968f3413” el identificador de la tarjeta.

4.2 Comportamiento general del dispositivo

El ESP8266-E cuenta con un procesador que solo permite una ejecución de vez, por lo que resulta importante que ninguna de las librerías implementadas sea bloqueante. Esto puede obviarse en el caso especial del inicio del dispositivo, donde es posible que alguna librería mantenga el dispositivo en espera durante un cierto tiempo. Tal es el caso de la librería de PWM, la cual permanece a la espera hasta que consiga conectar o hasta que se cumpla un timeout, momento en el cual el TeachFi pasa a comportarse como si fuese un AP.

En el siguiente diagrama de flujo puede verse la secuencia de ejecución de este dispositivo.

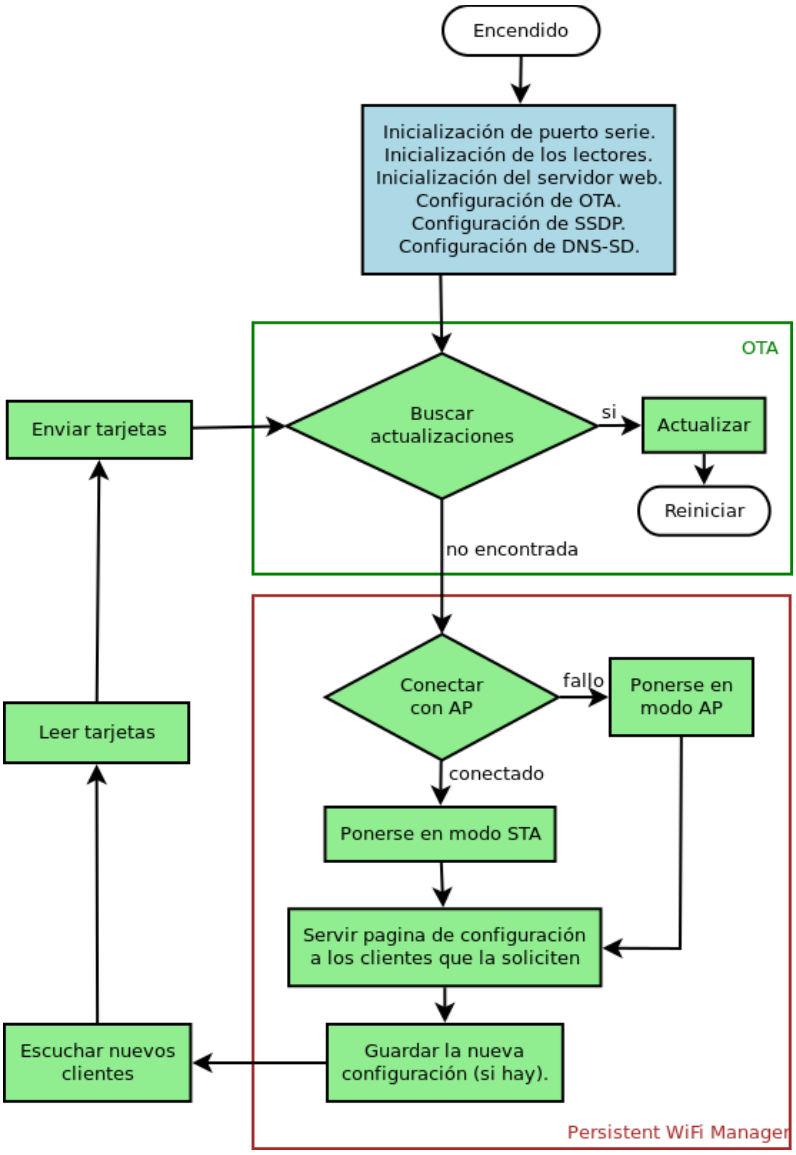

Ilustración 13: Diagrama explicativo de la secuencia de ejecución del TeachFi.

Como se puede apreciar en el diagrama, la caja azul corresponde a la parte de inicialización (o setup) del dispositivo, mientras que las restantes cajas verdes corresponden a la parte del bucle (o loop).

El proceso de funcionamiento resulta sencillo: Setup, búsqueda de actualizaciones, gestión de la WiFi, busqueda de clientes, lectura de tarjetas, envío de tarjetas y vuelta al comienzo del loop (actualizaciones).

4.3 Distribución de la memoria flash

Dado que dentro de la misma memoria flash han de convivir distintas tecnologías, es necesario hacer una mención a como se reparte las distintas zonas de memoria dentro del dispositivo.

```
|--------------|-------|---------------|--|--|--|--|--|
^              ^       ^               ^     ^
Sketch    OTA update   File system   EEPROM  WiFi config (SDK)
```

Ilustración 14: Distribución de la memoria flash del dispositivo.
Fuente: http://esp8266.github.io/Arduino/versions/2.0.0/doc/filesystem.html

En la siguiente tabla se detalla en profundidad como se reparte la memoria. No obstante esta tabla hace referencia a un ESP8266 con 512KB de memoria, por lo cual las cifras no se corresponden con los 4MB de memoria que posee el nodeMCU (ESP8266-E).

SPI Flash ROM Layout (with OTA upgrades)

This is for ESP IoT SDK version 0.8 and above, supporting OTA upgrades.

Address	Size	Name	Description	
00000h	4k	boot.bin	Bootloader	
01000h	64k	app.v6.flash1.bin	User application, slot 1	Aplicación en uso
11000h	180k	app.v6.irom0text1.bin	SDK libraries, slot 1	
3E000h	8k	master_device_key.bin	OTA device key	
40000h	4k		Unused	
41000h	64k	app.v6.flash1.bin	User application, slot 2	Espacio para OTA
51000h	180k	app.v6.irom0text1.bin	SDK libraries, slot 2	
7E000h	8k	blank.bin	Filled with FFh. May be WiFi configuration.	

Ilustración 15: División de la memoria en una flash de 512KB
Fuente: https://github.com/esp8266/esp8266-wiki/wiki/Memory-Map

Es importante destacar que la versión con OTA cuenta con un lanzador al comienzo de la memoria para gestionar las actualizaciones de firmware cargadas por este procedimiento.

4.4 Aplicación Android

Con el proposito de poder probar el descubrimiento del TeacchFi en la red local así como de poder probar la lectura de tarjetas, se ha creado una aplicación Android capaz llevar a cabo esta tarea.

A continuación se muestran varias capturas del funcionamiento de la aplicación.

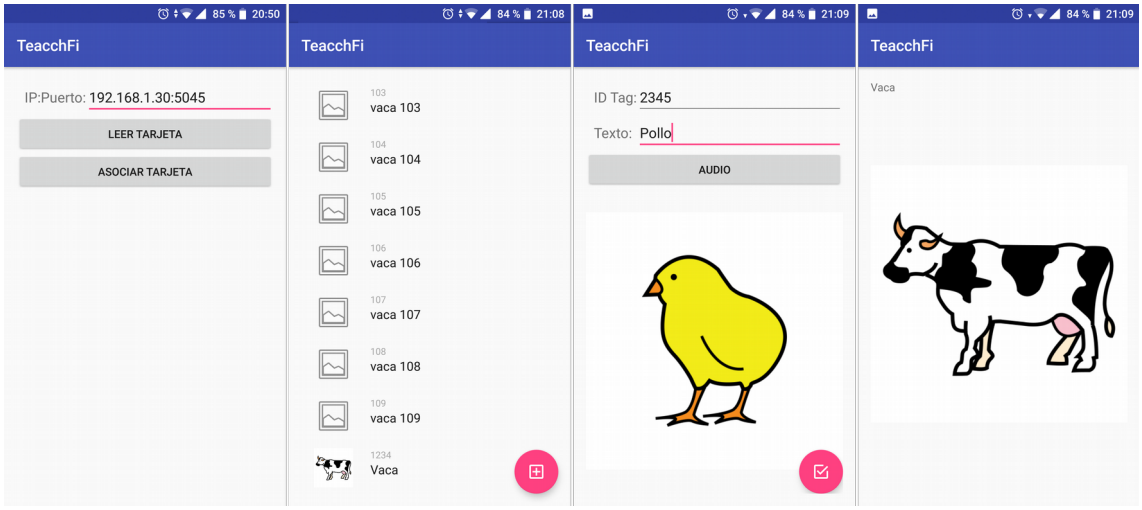

Ilustración 16: Varias capturas de la aplicación para el testeo del TeacchFi.

En estas imagenes se puede apreciar varias de las pantallas de las que dispone la aplicación.

A continuación se describe el comportamiento esperado en cada una de las pantallas mostradas. Este comportamiento sería (de izquierda a derecha):

4.4.1 Inicio de la aplicación

Al abrirse esta aplicación presenta una interfaz gráfica muy simple en la cual se muestran un par de botones y una caja donde se puede ver la IP/puerto del dispositivo con el que se desea asociar.

Inicialmente se muestra una IP fija predeterminada, pero en el momento en el que se descubre un anuncio con un identificador que corresponde al TeacchFi, esta ip es actualizada.

4.4.2 Asociación de tarjetas

En la segunda imagen, se muestra una lista con todas las tarjetas introducidas en el sistema. Estas tarjetas se componen de: una imagen, un identificador, un texto y un audio (opcional). Inicialmente solo hay unas cuantas tarjetas de prueba, pero tras pulsar el botón circular de la esquina inferior, es posible añadir nuevas tarjetas.

4.4.3 Creación de una tarjeta

En esta pantalla, se muestra los cuatro elementos de los cuales se compone una tarjeta dentro de la aplicación. Cabe destacar que el identificador no es necesario introducirlo a mano, sino que este es reconocido por la aplicación en el momento en el que una tarjeta es introducida en alguna de las bahías del TeacchFi.

4.4.4 Lectura de la tarjeta

Finalmente en la cuarta imagen se puede apreciar el comportamiento mostrado cuando la tarjeta correspondiente a un id conocido es depositada en una de las bahías del TeacchFi. Como se aprecia, tanto el texto como la imagenes son mostrados. Además de estos dos elementos, si la tarjeta posee un audio asociado, este es reproducido; en caso contrario un sintentizador de voz es el encargado de leer el texto asociado a la tarjeta.

5 Pruebas

A lo largo del transcurso de la implementación, se han ido realizando distintas pruebas con el propósito de verificar que se ha logrado conseguir los objetivos cumplidos en las especificaciones. El conjunto de pruebas y herramientas utilizadas para la verificación del funcionamiento del aparato se detallarán a continuación.

5.1 Pruebas de conectividad

La prueba inicial ha consistido en verificar que realmente el dispositivo se conecta a la WiFi. Para ello se ha verificado en la web de configuración del router que aparece un dispositivo nuevo cuando se conecta el TeacchFi.

5.1.1 Funcionamiento de DNS-SD y SSDP

Para este caso, se han empleado dos herramienta móvil de Android. Una de ellas es la creada en este proyecto con el propósito concreto de probar el TeacchFi y la otra herramienta en cuestión se llama PingTools Pro [15] (es de pago, pero tiene una versión gratuita). Esta aplicación entre otras cosas permite identificar los dispositivos de la red que se encuentran publicandose en esta. Como se puede apreciar en las siguientes capturas, ambos servicios funcionan perfectamente.

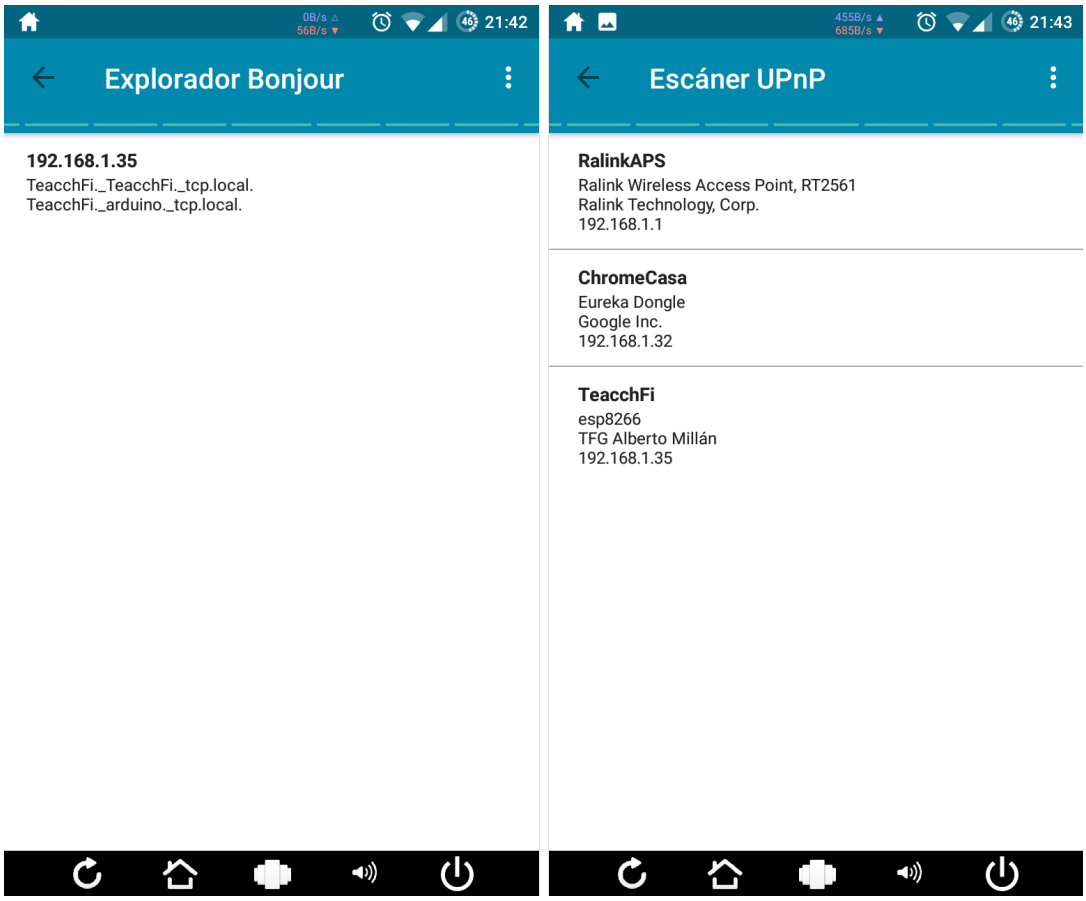

Ilustración 17:
A la izquierda: Descubrimiento del TeacchFi mediante DNS-SD
A la derecha: Descubrimiento del TeacchFi mediante SSDP (Ademas de otros dispositivos).

Hay que mencionar que tanto DNS-SD como SSDP pueden recibir nombres distintos según la herramienta con la que se consulten. En este caso concreto, “Bonjour” hace referencia a la implementación de Apple y UPnP (Universal Plug & Play) hace referencia a al protocolo que utiliza SSDP.

5.1.2 MQTT

Para las pruebas de MQTT se ha utilizado el servicio cloud denominado CloudMqtt [16]. Este servicio resulta muy práctico ya que no requiere de la instalación de ningún software adicional. No obstante si se deseasen hacer pruebas en la propia red, existe una implementación denominada Mosquitto [17] que puede ser de gran ayuda. Este servidor cuenta con la ventaja de que es compatible con RaspberryPi

5.2 Pruebas con usuarios

Además de las pruebas realizadas a cada uno de los componentes de la aplicación, también se han llevado a cabo pruebas con usuarios. En la siguiente imagen se puede apreciar a uno de estos usuarios operando con el dispositivo.

Ilustración 18: Usuario del TeacchFi realizando pruebas con las tarjetas.

6 Conclusiones

Finalmente se expondrán algunas detalles finales relacionados con la elaboración de este proyecto.

6.1 Tiempo dedicado

Para la elaboración de este proyecto se ha invertido un total aproximado de 450h. Este proyecto se comenzó a comienzos del año 2017 (febrero) y se concluye con la realización de esta memoria en septiembre de 2018. Estas horas se han ido invirtiendo en las siguientes tareas.

6.1.1 Reuniones

Estas abarcan tanto reuniones con el tutor del proyecto como reuniones con el Colegio Público de Educación Especial Alborada, el cual se le ha ido haciendo participe de cada uno de los avances realizados.

Las reuniones con el tutor fueron llevadas a cabo con una periodicidad aproximada de 2 al mes durante los primeros meses de desarrollo de la herramienta y participe en un total de 3 reuniones con el colegio Alborada.

Una vez el grueso del trabajo se encontró finalizado estas reuniones bajaron a una media de una cada 2 meses.

6.1.2 Estudio y Diseño

Estas fases fueron realizadas al comienzo del proyecto, donde junto con el tutor se realizaron una serie de pruebas iniciales para entender como podían operar los distintos componentes utilizados. Su duración se extendió por un periodo de unos 2 meses.

6.1.3 Implementación y Evaluación

Estas dos etapas fueron realizadas de forma simultanea, haciendo participe al usuario de cada uno de los avances realizados en los distintos prototipos creados. Su duración se extendió por un periodo de unos 2 meses.

6.1.4 Memoria

Durante el mes de Junio del 2017 empece a trabajar en una empresa del sector informático realizando labores de ingeniero de sistemas, lo cual hizo que al disponer de menos horas al día la redacción de la memoria se viese retrasada hasta ser finalizada en septiembre del año siguiente a su comienzo.

6.2 Librerías

Uno de los principales problemas encontrados ha sido en de encontrar librerías que se adecuen a este proyecto, en algunos casos estas estaban escasa o nulamente documentadas lo que ha conducido a que en no pocas ocasiones se tenga que recurrir al propio código fuente de estas para entender a fondo su comportamiento.

En otros casos, las librerías no se comportan de forma esperada o directamente resulta incompatibles con otras utilizadas dentro del mismo proyecto. Este fue el caso entre las librerías que controlan los RFID y el WiFiManager, que por algún motivo hacían que no se asignase bien la IP al estar corriendo las dos a la vez.

Un punto fuerte a favor las librerías utilizadas es que cuentan con un gran apoyo de la comunidad, con foros de ayudas donde poder preguntar y compartir información sin ningún tipo de coste. Esto hace que proyectos como este sean posibles de llevar a cabo sin más coste que el invertido en el propio hardware.

6.3 Licencias

Un aspecto que se ha tenido en cuenta a la hora de realizar este trabajo es que todo el material empleado esté licenciado bajo licencias abiertas o libres que permitan el acceso completo de las herramientas. En ese sentido, unicamente se ha utilizado la herramienta de Ping Tools durante la verificación que no cuenta con este tipo de licencias. No obstante, como alternativa, existe por ejemplo “Avahi” para linux.

6.4 Opiniones personales

Estoy muy contento de haber podido realizar este trabajo ya que me ha permitido aprender muchas cosas nuevas que desconocía sobre el mundo de los microcontroladores y en especial sobre el ESP8266. Debo de dar las gracias al Colegio Público de Educación Especial Alborada [18] por haberme abierto los ojos frente a una problemática que desconocía por completo como es la docencia de personas con TEA, así como alabar su trabajo en ese campo.

Finalmente me gustaría agradecer el esfuerzo dedicado por Enrique Torres al permitirme realizar con el este proyecto bajo su tutela.

7 Anexos

7.1 Enlaces

[1]: SoyVisual: www.soyvisual.org

[2]: Articulo del CDC (Centro para el Control y Prevención de Enfermedades) sobre el TEA: https://www.cdc.gov/ncbddd/spanish/autism/facts.html

[3]: NodeMcu: http://nodemcu.com/index_en.html

[4]: ESP8266. Foro oficial dedicado a este chip: http://www.esp8266.com/

[5]: Port del ESP8266 al IDE de arduino: http://www.esp8266.com/viewtopic.php?f=31&t=2150

[6]: Datasheet del RFID-RC522: http://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

[7]: Web oficial de Arduino: https://www.arduino.cc

[8]: Repositorio OTA: https://github.com/esp8266/Arduino/tree/master/doc/ota_updates

[9]: Persistent WiFiManager: http://ryandowning.net/PersWiFiManager/

[10]: Librería SSDP para el ESP8266: https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266SSDP

[11]: Librería mDNS para el ESP8266: https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266mDNS

[12]: Librería MQTT para el ESP8266: https://pubsubclient.knolleary.net/

[13]: LabLUA, creadores de uno de los lenguajes disponibles para el ESP8266: http://www.lua.inf.puc-rio.br/

[14]: Librerias SPIFFS:
http://esp8266.github.io/Arduino/versions/2.0.0/doc/filesystem.html

[15]: Aplicación Ping Tools Pro para Android:
https://play.google.com/store/apps/details?id=ua.com.streamsoft.pingtoolspro
Y su versión gratuita: https://play.google.com/store/apps/details?
id=ua.com.streamsoft.pingtools

[16]: CloudMqtt. Este servicio cuenta con una versión gratuita que permite la conexión de hasta 10 dispositivos de forma simultanea: https://www.cloudmqtt.com/

[17]: Mosquitto. Implementación Open Source de un servidor MQTT:
https://mosquitto.org/

[18]: Colegio Público de Educación Especial Alborada (Zaragoza):
http://cpeealborada.blogspot.com.es/