UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO
INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Diseño e implementación de un dispositivo IoT para seguridad de niños

Autor

Pablo Juan Villalba Guzmán

Directores

Jorge Navarro Ortiz
Sandra Sendra Compte

Escuela Técnica Superior de Ingenierías Informática
y de Telecomunicación

—

Granada, septiembre de 2018

Diseño e implementación de un dispositivo IoT para seguridad de niños

Autor

Pablo Juan Villalba Guzmán

Directores

Jorge Navarro Ortiz
Sandra Sendra Compte

Diseño e implementación de un dispositivo IoT para seguridad de niños

Pablo Juan Villalba Guzmán

Palabras clave: IoT, microcontrolador, Android, Arduino, buzzer, geolocalización, plugin, microprograma, Wi-Fi, potencia, coordenadas, seguridad, niños, dispositivo, conexión

Resumen

En este trabajo se muestra el diseño hardware y la posterior implementación software de un dispositivo IoT destinado a controlar la distancia entre padre/madre e hijo/a en todo momento. Para ello se crea una conexión Wi-Fi entre el móvil de uno de los padres/madres y el dispositivo del menor, y se mide constantemente la potencia de la señal Wi-Fi recibida. En función de esta potencia se establecen varios escenarios diferentes, desde el caso en que la potencia recibida es máxima hasta el caso en que no existe conexión entre ambos. Es en este caso cuando será fundamental el papel del dispositivo IoT y del servidor al que enviará sus datos para que los padres/madres sepan, mediante una aplicación desarrollada para móviles Android, dónde se encuentra su hijo/a.

IoT device for children safety design and implementation

Pablo Juan Villalba Guzmán

Keywords: IoT, microcrontoller, Android, Arduino, buzzer, Geolocation, plugin, microprogram, Wi-Fi, coordinates, power, security, children, device, connection

Abstract

Along this project, it is shown the hardware design and subsequent IoT device software implementation, set aside for controlling the distance between parent and children in any time. To make this possible, a Wi-Fi connection is created among one of parent's smartphones and the child's device, and the reception signal of Wi-Fi power is constantly monitored. Depending on this result, several different scenarios are established, from the case in which received power is the maximum one until the case in which there is no connection between both devices. This last case is the one in which the roles that IoT device and the server receiver of its data perform, will be essential to inform parents about where their child is, by a mobile app developed with Android OS.

Yo, **Pablo Juan Villalba Guzmán,** alumno de la titulación Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Pablo Juan Villalba Guzmán

Granada a 7 de septiembre de 2018.

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

Dª. **Sandra Sendra Compte**, Profesora del Área de Ingeniería Telemática del Departamento Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Diseño e implementación de un dispositivo IoT para seguridad de niños***, ha sido realizado bajo su supervisión por **Pablo Juan Villalba Guzmán**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 7 de septiembre de 2018.

Los directores:

Jorge Navarro Ortiz **Sandra Sendra Compte**

Agradecimientos

A mis padres y hermana por apoyarme en todo momento y creer siempre en mí, por estar siempre ahí en los momentos buenos y más aún en los no tan buenos, por quererme tanto y por haber hecho posible que hoy esté donde estoy.

A mi pareja por animarme día a día a luchar por mis objetivos, por todos sus consejos, por su ayuda y por la motivación incondicional.

A mis amigos y compañeros de facultad por todos los momentos compartidos y el apoyo mutuo año tras año.

A Jorge por haberme guiado durante todo este tiempo, por contestar a mis innumerables correos fuese a la hora que fuese, por la actitud que ha mantenido siempre conmigo y por hacer que todo este trabajo haya sido más llevadero.

Contenido

1. Introducción

Muy frecuentemente, y más en épocas de vacaciones, nos encontramos con noticias [1] que cuentan historias de padres que pierden de vista a sus hijos. Esto provoca un gran susto y momentos de angustia a unos padres que no saben dónde se encuentra su hijo y muchas veces ni siquiera por dónde empezar a buscar.

Hay encuestas [2] que aseguran que el 65% de las madres con niños de una edad comprendida entre 2 y 9 años se han enfrentado alguna vez a esta situación. Esto suele ocurrir en lugares como centros comerciales, la playa, parques, aeropuertos, conciertos, etc. Siempre en lugares donde hay grandes aglomeraciones de gente y es fácil perderlos de vista en un momento.

En este trabajo se va a hacer uso de un dispositivo IoT para intentar evitar este tipo de situaciones. El término IoT se refiere a la interconexión digital de objetos cotidianos que constan de identificadores únicos y que tienen la capacidad de transferir datos a través de Internet. Este concepto fue propuesto originalmente por Kevin Ashton en 1999, y representa la próxima revolución de Internet, ya que supone un avance enorme en su capacidad para recopilar, analizar y distribuir datos que se pueden convertir en información, en conocimiento y, en última instancia, en sabiduría.

Alrededor de 2005, según Cisco [3], había tantos dispositivos como habitantes en el planeta conectados a Internet. En la actualidad, se estima que esta cantidad aumente en torno a tres y cuatro por persona, y se prevé que se duplique durante los tres próximos años. Es muy difícil predecir la evolución de este concepto debido a la novedad y el rápido crecimiento del sector, provocado por las grandes inversiones que éste recibe.

1.1 Motivación y objetivos

La motivación principal que impulsa este trabajo nace de la oportunidad de solucionar esta problemática de la vida cotidiana, aplicando muchas de las técnicas y conocimientos adquiridos durante el grado.

Como siguiente punto que motiva la realización de este proyecto, se encuentra la alta viabilidad de construcción del dispositivo final, así como la facilidad de uso junto a otros dispositivos y tecnologías que forman parte del día a día y que constituyen parte fundamental del escenario de uso. Éstos son: el dispositivo IoT que se va a diseñar durante este trabajo, que consta principalmente de un microcontrolador ESP8266 y un teléfono móvil que cuenta con el Sistema Operativo Android.

Actualmente ya existen algunos dispositivos que realizan esta misma función, pero utilizan tecnologías diferentes y normalmente son mucho más caros, por lo que otra de las motivaciones es conseguir la misma finalidad, siendo la solución propuesta mucho más flexible para incorporar nuevas funcionalidades y consiguiendo un precio mucho más económico.

Para llevar a cabo su realización se plantean una serie de objetivos:

- Implementar un programa capaz de avisar a los padres de que su hijo se está alejando.
- Implementar una alarma acústica.
- Realizar un diseño hardware funcional para el dispositivo.
- Conseguir datos de geolocalización.
- Desarrollar una aplicación que ayude a los padres a encontrar a su hijo si se pierde.
- Objetivos secundarios: que se utilicen móviles con sistema operativo Android y que el dispositivo de seguridad diseñado sea sencillo, barato y reprogramable.

1.2 Estructura de la memoria

El presente trabajo se estructura en 7 capítulos más un anexo final. A lo largo de ellos se podrá ver el proceso seguido para la elaboración de este trabajo final de grado.

Estado del arte

El capítulo 2 habla de una serie de trabajos o dispositivos que guardan una estrecha relación con este proyecto, puesto que todos ellos proponen soluciones distintas al problema que se ha comentado anteriormente.

Planificación y organización

El capítulo 3 describe la organización y la planificación inicial para realizar este trabajo, mostrando de manera orientativa el tiempo de que se espera dedicar para las diferentes tareas.

Escenario de trabajo

El capítulo 4 realiza una descripción del software y del hardware utilizado durante el desarrollo de este trabajo final de grado. En él se detallan sus principales características y se explican las funcionalidades de todos los programas utilizados, concretando para qué se han utilizado en este trabajo.

Diseño y desarrollo

El capítulo 5 recoge la parte central del trabajo. Por una parte, describe el diseño del dispositivo y, por otra parte, explica mediante varios diagramas el esquema de funcionamiento del proyecto en conjunto, con sus diferentes situaciones previstas. También explica el comportamiento del microcontrolador y de la aplicación para móvil por separado, profundizando en algunas partes o funciones de sus respectivos programas.

Pruebas y resultados

En el capítulo 6 se realiza una recopilación de las pruebas realizadas que muestran cómo funciona el dispositivo y que cumple con los requisitos iniciales. Al igual que en el capítulo anterior, consta de varios apartados. En el primero de ellos se centra en el funcionamiento del microcontrolador, para continuar con el funcionamiento de la aplicación para móvil, y por último muestra el funcionamiento del *plugin* creado para la página web del servidor.

Presupuesto

El capítulo 7 recoge la valoración económica estimada del proyecto, considerando todos los materiales y recursos empleados para su realización, entre los que se incluyen materiales hardware, software y horas de trabajo por parte del alumno y de los tutores.

Conclusiones y líneas futuras

El capítulo 8 habla sobre las muchas ideas y propuestas de mejora que se han ido ocurriendo a lo largo del desarrollo de este trabajo, y que son susceptibles de realizarse en una segunda versión o trabajo de continuación de este dispositivo. Finalmente se incluyen también la conclusión y una reflexión personal acerca del trabajo realizado.

Referencias

Tras el capítulo 8 de la memoria se muestran la bibliografía que ha servido de apoyo a la realización del presente trabajo.

Anexos

Por último, se incluye anexo a esta memoria el código desarrollado en Java con Android Studio para la aplicación móvil, el código desarrollado en Arduino para la aplicación del dispositivo de seguridad y el código del *plugin* de ThingSpeak en JavaScript.

2. Estado del arte

Como ya se mencionó anteriormente, existe un gran número de aplicaciones y dispositivos diseñados para que los padres puedan saber dónde se encuentran sus hijos. Las funcionalidades que estas aplicaciones ofrecen van desde establecer una zona segura hasta conectar la cámara del móvil de los menores para ver dónde o con quién están, pasando por controlar la velocidad a la que se mueven en cada momento.

La mayoría de estas aplicaciones y dispositivos basan su funcionamiento en sistemas GPS, que pueden estar incluidos en los móviles de los niños, en pequeñas pulseras o incluso en implantes subcutáneos, aunque éstos últimos han creado una gran controversia.

Otras aplicaciones desempeñan la misma función utilizando sistemas Bluetooth, y otras incluso añaden tecnología GSM para dotar al dispositivo de la posibilidad de realizar y recibir llamadas. Para conocer algunos ejemplos, a continuación, se muestra una lista que contiene algunos de los más conocidos:

- Cerberus [4]: se trata de una aplicación que solo está disponible para móviles con sistema operativo Android. Ofrece la posibilidad de ubicar los dispositivos en un mapa, iniciar una alarma, guardar una copia de seguridad, tomar fotografías del presunto ladrón y bloquear o borrar datos, entre otras posibilidades. Su intención inicial era ofrecer una solución antirrobo para todo tipo de dispositivos con este sistema operativo, pero como actualmente los niños cada vez poseen su primer teléfono móvil a edades más tempranas, también sirve para localizarlos a ellos.
- FiLIP [5]: se dan a conocer como 'El primer *smartwatch* para tu hijo'. La aplicación para móvil está disponible tanto en el Apple App Store como en el Google Play Store. El producto se vende exclusivamente en tiendas Movistar y está disponible por un precio de 135€. La aplicación ofrece multitud de funciones: recibir y realizar llamadas, localizar al menor combinando las tecnologías GPS, GSM y Wi-Fi, botón para iniciar un modo de emergencia, programar hasta 5 zonas seguras,

enviar mensajes de texto al dispositivo de manera unidireccional, ajustar la frecuencia de actualización de la ubicación, etc.

- Localizador Sherlog [6]: es un servicio ofrecido por una compañía que se dedica a gestionar flotas, pero también tiene este producto destinado a la localización de niños. Consigue la ubicación cada dos segundos mediante una doble triangulación GPS/GLONASS. Como principales características incluye notificaciones cuando el dispositivo entre o salga de zonas seguras, resistencia al agua y gran robustez anti golpes, y además asegura ser el dispositivo más pequeño y discreto del mercado.
- My Buddy Tag [7]: se trata de un dispositivo con unas funcionalidades más sencillas. Sus principales características son notificaciones por email cuando los menores se salen del rango del Bluetooth, alertas automáticas cuando el dispositivo permanece más de 5 segundos sumergido en el agua, y la más llamativa, su batería dura hasta un año entero. El dispositivo cuesta 39.99$, pero como su batería no se puede recargar, una vez que se le agota hay que comprarlo nuevamente para seguir disfrutando de sus funciones.

La lista podría ser mucho más larga, pero en general todos los dispositivos y aplicaciones que intentan mantener la seguridad de los menores mediante tecnologías inalámbricas tienen unas características parecidas y no es necesario detallar todos y cada uno de ellos para hacerse una idea de su funcionamiento. Lo que hace destacar el actual proyecto frente a todos los demás es que gracias a la antena Wi-Fi que lleva incorporada el microcontrolador ESP8266 que se incluye en el dispositivo, se pueden realizar casi todas las funciones que ofrecen los anteriores dispositivos mostrados, pero con unos componentes mucho más económicos.

3. Planificación y organización

Durante el presente capítulo se expone el método de planificación temporal que se ha seguido para el desarrollo del presente trabajo, así como las herramientas que se han empleado para la gestión y organización del mismo.

Las tareas se han agrupado en cinco bloques diferenciados según su función o importancia.

En primer lugar, se ha definido el bloque 'Fase inicial'. En éste, se han incluido las tareas que, inicialmente, dieron comienzo al desarrollo del proyecto. Entre estas tareas se encuentra la única que se ha catalogado como bloqueante, que corresponde a la adquisición del software necesario para trabajar en el posterior desarrollo.

El siguiente bloque de tareas que se llevó a cabo se engloba en el denominado 'Diseño y desarrollo inicial'. Dicho bloque se inició una vez que se tuvo una idea bien definida de la temática e iniciales pautas del proyecto y que se dispuso de todo el material hardware necesario. Se ha considerado este segundo bloque de tareas, 'Diseño hardware dispositivo de seguridad', como no bloqueante debido a que el diseño del dispositivo inicial pudo haber ido sufriendo cambios conforme se hacían pruebas, lo que podría haber resultado en el desarrollo de diferentes microprogramas para testeo.

El tercer bloque de tareas se ha denominado 'Desarrollo final'. Este bloque es el que se ha prolongado durante más tiempo debido a su dificultad y tiempo para elaborar todo lo que en él se ha englobado. Entre las tareas que se han definido en este bloque, se encuentran, en primer lugar, el desarrollo de una variedad de posibles versiones del programa final en el cual se han intentado reunir todas las funcionalidades desarrolladas en el bloque anterior, y, posteriormente, las versiones definitivas del código de la aplicación para móvil en Android, del programa para el dispositivo de seguridad y del *plugin* para el servidor.

En el cuarto bloque, Redacción', se ha englobado la tarea genérica de la realización de la presente memoria, que guía todo el desarrollo seguido para la elaboración del presente proyecto.

Para finalizar, se ha definido un quinto y último bloque que engloba todas las tareas que se han realizado para llevar a cabo el control y la gestión de la planificación de este trabajo, denominándose 'Tareas de planificación y organización'. Entre las actividades realizadas en relación con este apartado, se recogen tanto las relativas al tiempo dedicado a reuniones con los tutores, tanto presenciales como de forma remota, como a las que han supuesto el establecimiento de las demás tareas y su consecuente inclusión y constante actualización en la herramienta de gestión de proyectos, Trello y en el diagrama que recoge tanto este bloque como todos los anteriores.

El diagrama mencionado, realizado siguiendo la metodología del diagrama de Gantt, se muestra a continuación.

Diagrama de Gant correspondiente a la preparación, desarrollo y redacción del trabajo **Diseño e implementación de un dispositivo de seguridad para niños.**

Por Pablo Juan Villalba Guzmán

			Enero			Febrero				Marzo				Abril				Mayo					Junio				Julio					Agosto				Septiembre
Fase	Tarea principal	Tarea derivada	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	S31	S32	S33	S34
I	***Fase inicial***																																			
		Reuniones puesta en común de idea principal del proyecto																																		
		Adquisición hardware necesario																																		
		Preparación software necesario																																		
		Investigación sobre el tema del proyecto																																		
II	***Diseño y desarrollo inicial***																																			
		Diseño hardware dispositivo de seguridad																																		
		Aprendizaje IDE Android Studio																																		
		Desarrollo pequeños programas para ESP8266																																		
III	***Desarrollo final***																																			
		Desarrollo de un programa final para ESP8266																																		
		Desarrollo de una app final																																		
IV	***Redacción***																																			
		Redacción de la memoria del proyecto																																		
V	***Tareas de planificación y organización***																																			
		Reuniones/Comunicaciones presenciales y remotas con los tutores																																		
		Actualización herramientas de gestión de proyectos																																		

	Bloqueante
	No Bloqueante

Gráfica 1. Diagrama de Gantt

Como software de apoyo en la organización del trabajo se ha utilizado la herramienta Trello:

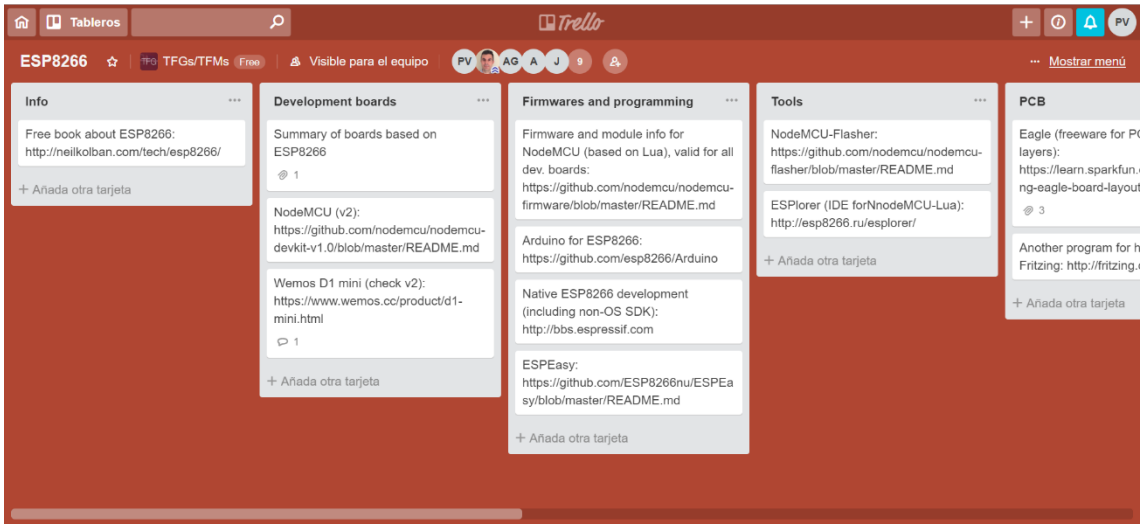

Imagen 1. Interfaz de la herramienta Trello

Tal y como se puede ver en la imagen anterior esta herramienta permite hacerse una idea, de un solo golpe de vista, del estado actual del proyecto. Tanto el alumno como los tutores tenían acceso a este software donde se pueden compartir dudas, información, notificar de los avances, marcar fechas orientativas para terminar determinadas tareas, etc.

4. Escenario de trabajo

En este apartado se va a realizar una breve descripción de los dispositivos hardware utilizados, así como de las distintas herramientas software que han sido necesarias para la realización del presente trabajo.

4.1 Hardware

4.1.1 Microcontrolador

Imagen 2. Microcontrolador ESP8266 Wemos D1 Mini Pro

Se trata del microcontrolador ESP8266 Wemos d1 mini pro, y en él se almacenará y ejecutará el programa que se ha desarrollado en este proyecto. Un microcontrolador es un circuito integrado que en su interior contiene una CPU, unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. En el caso del ESP8266 Wemos d1 mini pro, consta de 11 pines digitales de entrada/salida, 1 entrada analógica, conexión USB-TO-UART, 16M bytes y un tamaño muy reducido, además de un chip integrado con conexión Wi-Fi.

4.1.2 Buzzer

Imagen 3. Buzzer YL-44

Para hacer sonar la alarma del dispositivo se precisa de un pequeño *buzzer* (o zumbador). Este *buzzer* es activo ya que incorpora un oscilador simple lo que, a diferencia de los *buzzer* pasivos, permite liberar un poco de carga al procesador.

Además de estos dos dispositivos hardware, también han sido necesarios para el montaje del dispositivo de seguridad un cable USB-MicroUSB, una *protoboard* de pequeño tamaño y algunos cables.

4.1.3 Ordenador personal

Imagen 4. Ordenador personal Lenovo Yoga 700

Para llevar a cabo todo el desarrollo se ha utilizado este Lenovo Yoga 700. Es un PC que cuenta con una pantalla de 14 pulgadas, un procesador Intel Core i7 de 6ª generación, 8GB de memoria RAM y sistema operativo Windows 10.

4.1.4 Móvil personal 1

Imagen 5. Smartphone Xiaomi RedMi Note 5 Pro

Todo el proceso de test y la posterior fase de recopilación de pruebas se han realizado en este Xiaomi RedMi Note 5 Pro. Este móvil dispone de 4GB de RAM, 64GB de memoria y un procesador Snapdragon 636 a 1.8GHz, con el sistema operativo Android 8.1.0.

4.1.5 Móvil personal 2

Imagen 6. Smartphone Google Nexus 5

Para la fase de recopilación de pruebas son necesarios dos móviles. El primero de ellos será el móvil personal 1, y el segundo es este Nexus 5 que consta de 16GB de memoria, 2GB de RAM, un procesador Qualcomm Snapdragon de 4 núcleos y un sistema operativo Android 6.0.1.

4.2 Software

4.2.1 Arduino IDE

Imagen 7. Logotipo Arduino

Arduino IDE es el entorno de desarrollo integrado de Arduino y se usa principalmente para el desarrollo de programas para cualquier placa de Arduino y, en general, para cualquier microcontrolador soportado. Se ha utilizado la versión 1.8.5 de esta herramienta de software libre. Este software se ha utilizado para el desarrollo de todos los *sketches* o programas del ESP8266.

4.2.2 Android Studio IDE

Imagen 8. Logotipo Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Está basado en IntelliJ IDEA y consta de una gran cantidad de herramientas para desarrolladores. Es un entorno unificado donde se pueden desarrollar aplicaciones para multitud de dispositivos Android tales como móviles, televisores, coches, wearables, etc. En la realización de este proyecto se ha utilizado la versión 2.3.3, y en él se han desarrollado todas las aplicaciones para móvil.

4.2.3 Servidor ThingSpeak

Imagen 9. Logotipo ThingSpeak

Según sus desarrolladores, "*ThingSpeak es una aplicación de Internet de las cosas de código abierto y API para almacenar y recuperar datos de cosas que usan el protocolo HTTP a través de Internet o a través de una red de área local*". En la realización de este proyecto ThingSpeak ha desarrollado dos funciones. La primera de ellas es la de servidor con base de datos, puesto que los datos recogidos por el microcontrolador se envían aquí y esos mismos datos son recogidos de aquí por la aplicación para móvil. La segunda función es la de dar soporte a la aplicación móvil, ya que se ha desarrollado un *plugin* para esta herramienta en el que se puede ver la última ubicación del menor en un mapa de Google Maps.

4.2.4 Fritzing

Imagen 10. Logotipo Fritzing

Fritzing es una herramienta de software libre que ofrece la posibilidad de documentar prototipos. En su interfaz gráfica se puede montar sobre una *protoboard* los componentes que se necesiten para cualquier proyecto de manera intuitiva. Tiene una gran cantidad de componentes, y puedes importar o crear los que no estén en su lista predeterminada. En la realización de este proyecto se ha utilizado la versión 0.9.3b.

4.2.5 Trello

Imagen 11. Logotipo Trello

Trello es un software de administración de proyectos que actualmente pertenece a la empresa de software Atlassian. Permite tener una visión general del estado del proyecto gracias a su formato de presentación. Consiste en una serie de tableros organizados espacialmente donde se pueden crear listas y, dentro de estas listas, se pueden añadir tarjetas que recojan brevemente la información relacionada con alguna tarea. En este proyecto se ha usado como herramienta de ayuda a la organización.

5. Diseño y desarrollo

En este capítulo se va a describir paso a paso el funcionamiento lógico y el diseño de la aplicación para móvil, del dispositivo de seguridad y del *plugin* creado para ThingSpeak.

5.1 Diseño y funcionamiento del dispositivo de seguridad

En este apartado se realiza, en primer lugar, una descripción física del dispositivo de seguridad y, posteriormente, se describe el funcionamiento del programa que se ejecutará en el microcontrolador.

5.1.1 Diseño del prototipo

El dispositivo de seguridad está formado, en un primer prototipo, por una *protoboard* de tamaño pequeño, el microcontrolador ESP8266 Wemos d1 mini pro, un *buzzer* activo y algunos cables para realizar las conexiones entre dichos elementos.

Se ha realizado un diseño de este dispositivo con el software Fritzing, quedando como se muestra en la siguiente imagen:

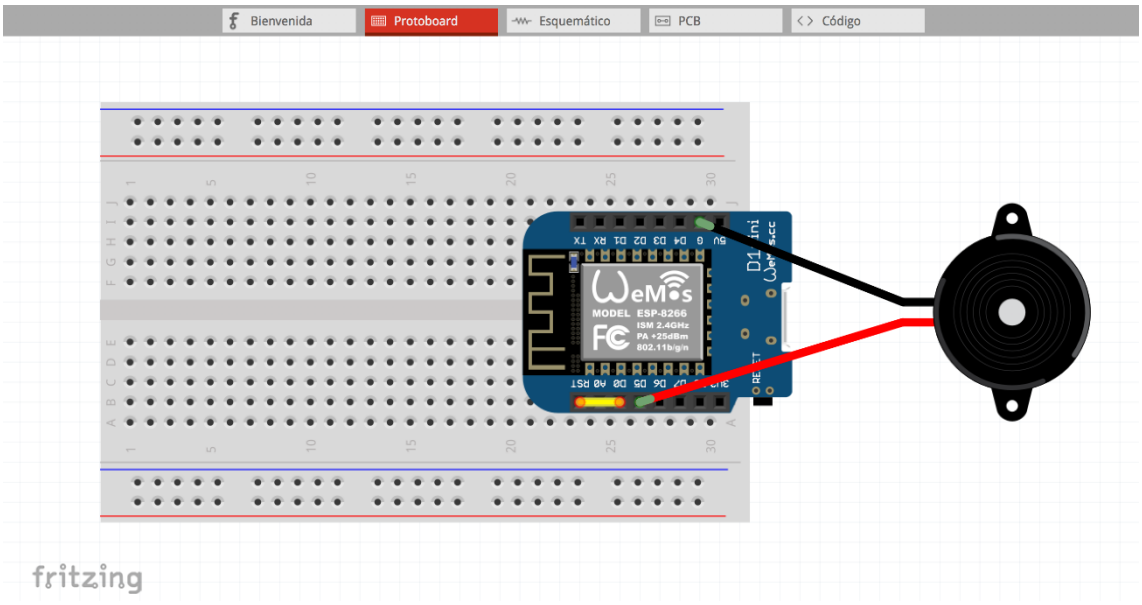

Imagen 12. Diseño protoboard con Fritzing del prototipo del dispositivo hardware microcontrolador + buzzer

En esta captura se pueden ver con detalle todas las conexiones que tiene el dispositivo de seguridad. Por una parte, se puede ver que el *buzzer* tiene ambos pines conectados al microcontrolador. Ambas conexiones están destinadas a la alimentación

del mismo. Mediante el cable negro uno de sus pines se encuentra unido a la tierra del microcontrolador y con el cable rojo se une el otro al pin D5. La elección de dicho pin ha sido propia, ya que se podría haber utilizado algún otro pin de entrada/salida digital. Por otra parte, vemos que el cable amarillo une los pines D0 y RST del microcontrolador. Esta conexión tiene como función permitir que el microcontrolador pueda usar el modo *deep sleep* que, como veremos más adelante, es fundamental para disminuir el consumo del dispositivo.

La siguiente tabla contiene la nomenclatura de los pines del ESP8266 Wemos d1 mini pro:

Pin	Function	ESP8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.3V input	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
GND	Ground	GND
5V	5V	–
3V3	3.3V	3.3V
RST	Reset	RST

Tabla 1. Nomenclatura relativa a los pines del ESP8266 Wemos D1 Mini Pro

Existe una diferencia entre el diseño realizado en la herramienta Fritzing y el diseño hardware real del dispositivo. Esta diferencia es que el *buzzer* del diseño en Fritzing es un *buzzer* pasivo que consta de solo dos pines. En el diseño real se ha utilizado un *buzzer* activo que consta de tres pines, del que ya se comentaron sus ventajas anteriormente.

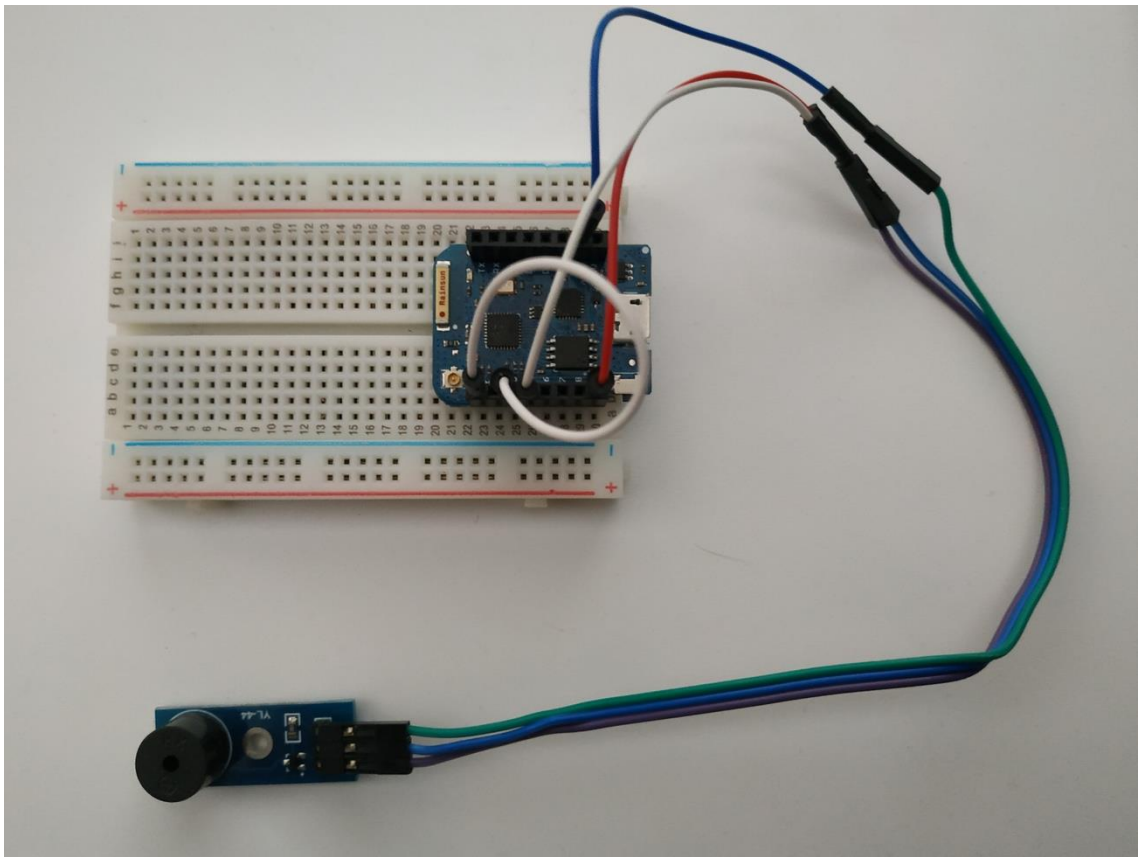

Imagen 13. Prototipo del dispositivo de seguridad

Como se puede ver en la anterior imagen, el ESP8266 Wemos d1 mini pro cuenta con un conector USB-TO-UART que facilita la programación y además permite suministrar el voltaje necesario.

5.1.2 Lógica del dispositivo

El programa que contiene el dispositivo de seguridad se ha desarrollado en el lenguaje de programación propio de Arduino. Para explicar su funcionamiento se ha creado el siguiente diagrama utilizando la herramienta Draw.io [9]:

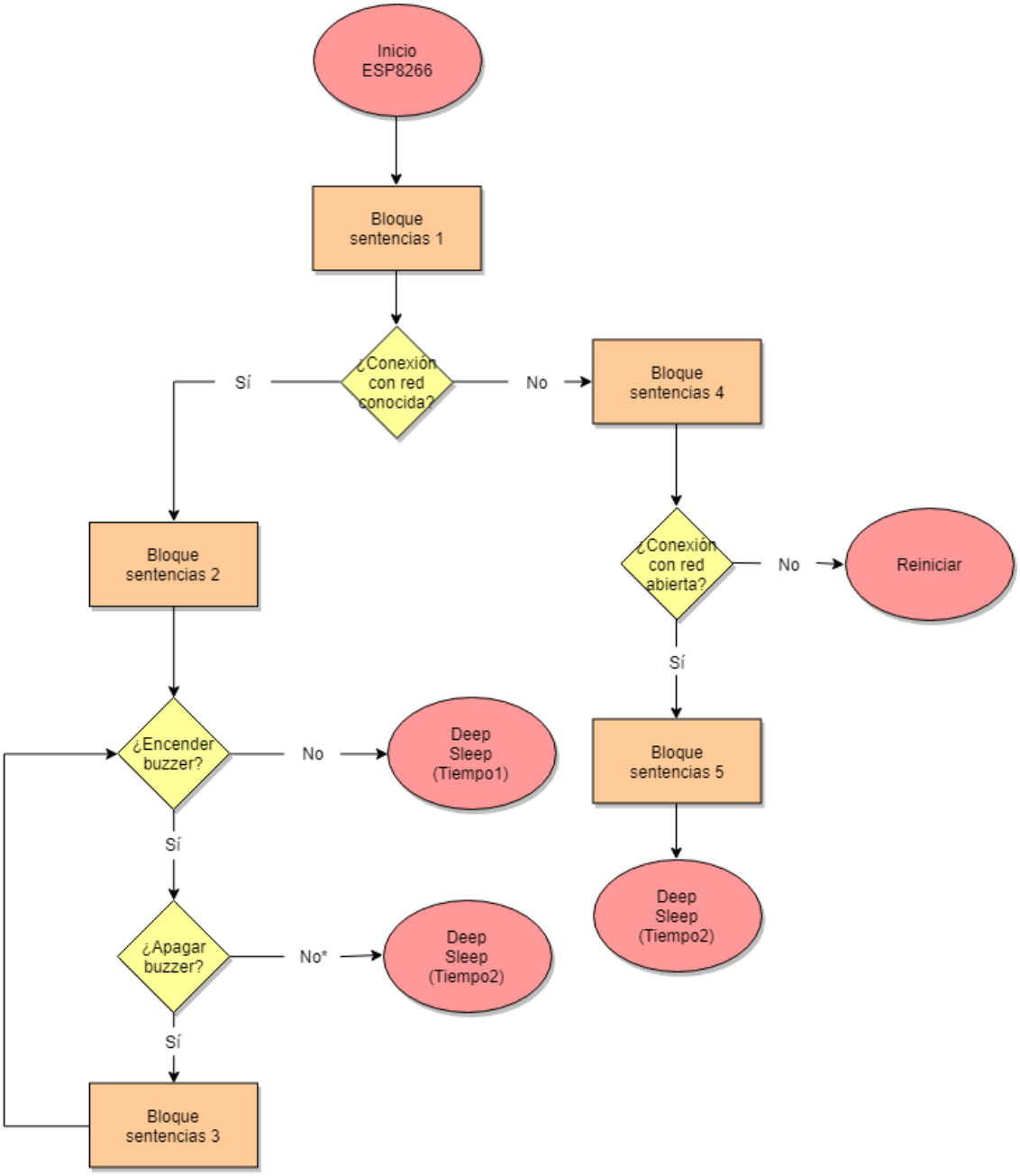

Imagen 14. Diagrama de bloques - Representación de la lógica de funcionamiento del programa en el microcontrolador

Como se puede ver en el diagrama anterior el programa está diseñado para que, sea cual sea la rama de flujo que se ejecute, el microcontrolador siempre se inicie al principio y entre en modo *deep sleep* o se reinicie al terminar de ejecutar las sentencias correspondientes.

Para conseguir este comportamiento todo el programa se ha desarrollado en la parte del setup() que contiene todo programa de Arduino:

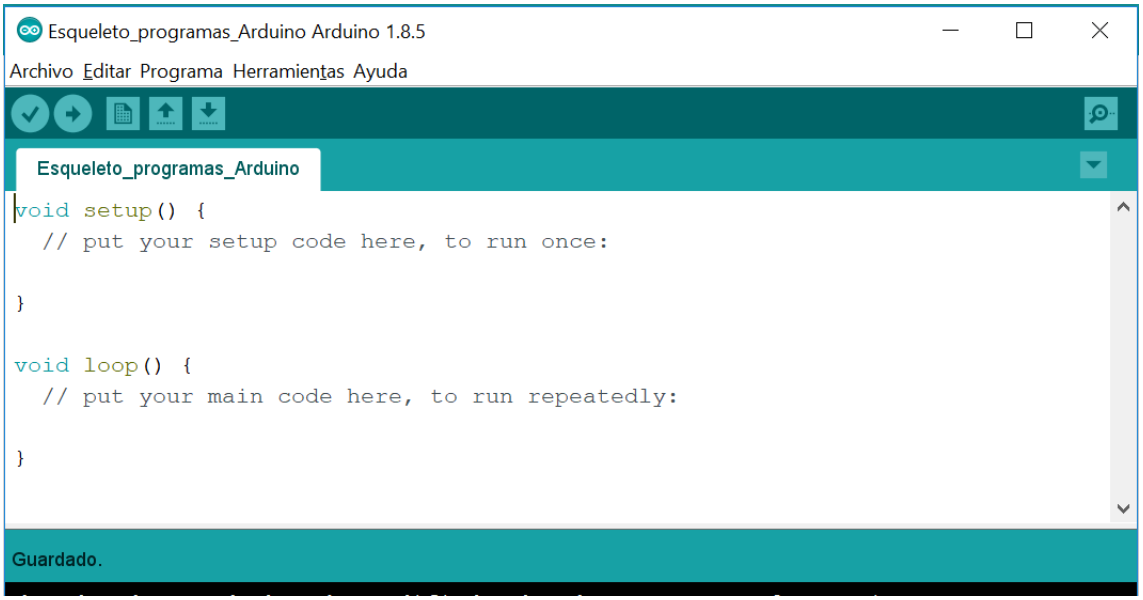

Imagen 15. Interfaz Arduino - Esqueleto del programa

El objetivo de este comportamiento es que el microcontrolador no esté encendido todo el tiempo. Usando el modo *deep sleep* del microcontrolador el consumo medio es de alrededor de 1 mAh cuando duerme, mientras que el consumo medio cuando está encendido es de alrededor de 70mAh [9]. Esto es así porque, cuando el microcontrolador está durmiendo, solo permanece activo su reloj de tiempo real (RTC, *Real Time Clock*). Gracias a este comportamiento y a otros que se comentarán más adelante, se ha conseguido que el consumo energético del dispositivo sea mucho menor y que por lo tanto pueda permanecer más tiempo funcionando.

Continuando con la explicación del diagrama de bloques (ver imagen 14), siempre que el microcontrolador se enciende realiza el llamado 'Bloque sentencias 1'. En este bloque de sentencias lo que se hace es definir todas las variables globales que se van a necesitar durante el resto del programa y recoger datos de todas las redes Wi-Fi que el microcontrolador sea capaz de detectar. Con estos datos posteriormente se podrá conseguir la ubicación del microcontrolador gracias a la *Geolocation API* de Google. Una vez que se han recogido estos datos, se intenta conectar a la red Wi-Fi creada en el móvil complementario de este proyecto. Para esta conexión se definen como parámetros fijos la dirección IP asignada al microcontrolador, la subred y la pasarela por defecto porque así también se ahorra en el consumo del microcontrolador [10]. Llegado a este punto, el programa puede tomar un flujo u otro en función de si se ha conseguido conectar a la red Wi-Fi conocida o no.

Si el microcontrolador consigue conectarse a dicha red Wi-Fi, se ejecuta la parte del programa correspondiente al llamado 'Bloque sentencias 2' del diagrama de bloques (ver imagen 14). En este bloque se reúnen una serie de datos que se han considerado importantes, se envían al servidor ThingSpeak, y se crea un servidor que permitirá activar el *buzzer* del dispositivo de seguridad.

Los datos que se recogen en este bloque de sentencias son la potencia recibida de la red Wi-Fi por parte del microcontrolador, la latitud, la longitud y la precisión de estas dos medidas relativas a la ubicación del microcontrolador. A continuación, se describe cómo se pueden obtener estos datos.

La potencia de la señal se puede medir fácilmente usando el comando 'WiFi.RSSI()' perteneciente a la librería 'ESP8266WiFi.h'.

La obtención de los datos relativos a la ubicación del microcontrolador son algo más complejos. Para ello se necesitan los datos que se obtuvieron en el 'Bloque sentencias 1' de las redes Wi-Fi detectadas por el dispositivo, la librería 'WifiLocation.h'[11] y una *Google API Key* de la *Geolocation API* de Google. En la petición que se realiza a la API se envían la dirección MAC del AP, la potencia recibida y el canal utilizado. Esta petición debe tener formato JSON y debe contener dos o más objetos. Como respuesta a esta petición, Google devuelve, también en formato JSON, la localización mediante una pareja latitud/longitud y su precisión en metros. En el siguiente par de imágenes se puede ver un ejemplo de los datos a enviar y de la respuesta recibida:


```
{
        "wifiAccessPoints": [
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-58,"channel":11},
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-85,"channel":11},
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-82,"channel":1},
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-89,"channel":6},
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-86,"channel":13},
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-89,"channel":4},
                {"macAddress":"XX:XX:XX:XX:XX:XX","signalStrength":-42,"channel":5}
   ]
}
```


Imagen 16. JSon Ejemplo envío objetos de geolocalización en Geolocation API

```
{
  "location": {
    "lat": 51.0,
    "lng": -0.1
  },
  "accuracy": 1200.4
}
```


Imagen 17. JSon Ejemplo recepción objetos de geolocalización en Geolocation API

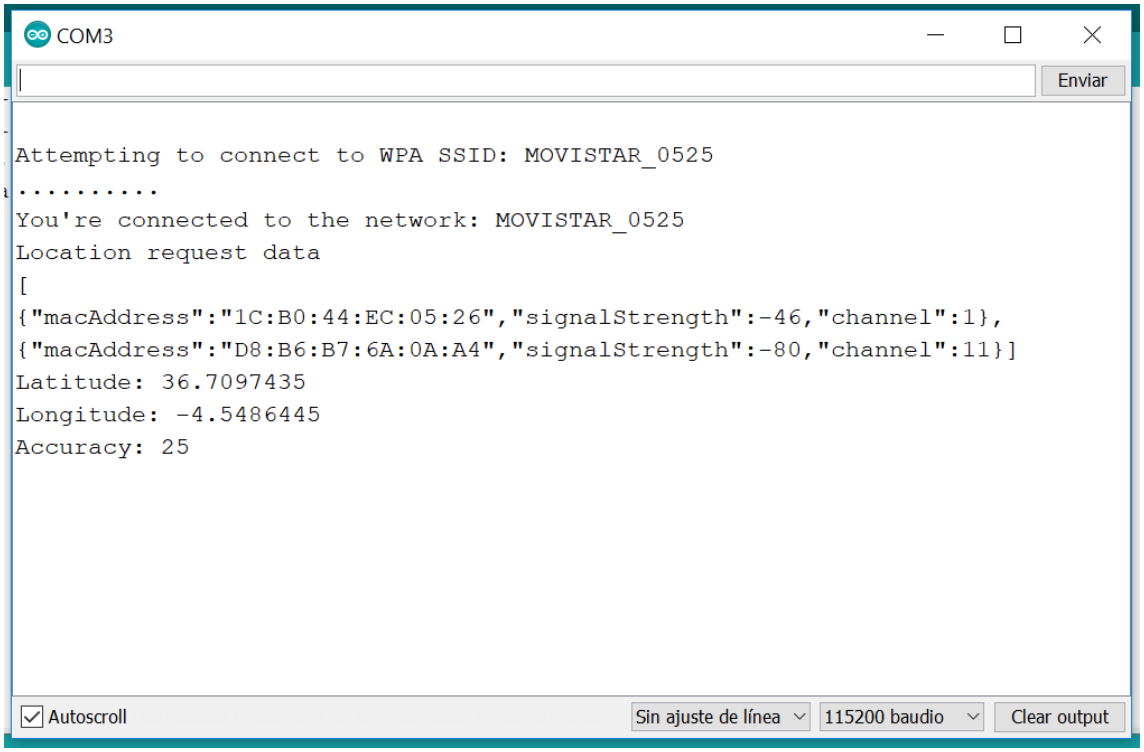

Imagen 18. Puerto Serie - Respuesta prueba Geolocation API

El siguiente paso es enviar todos estos datos al servidor ThingSpeak. Es necesario tener una cuenta en ThingSpeak, crear un canal donde almacenar los datos que se deseen y crear tantos campos como sean necesarios. Para este proyecto se ha creado un canal con cuatro campos para poder almacenar los datos de potencia recibida, latitud, longitud y precisión, respectivamente. Para almacenar estos datos se realiza una petición POST al servidor. En el cuerpo se concatenan todos los datos que se quieren enviar, identificándolos mediante etiquetas del tipo 'field1=X&field2=Y', y en la cabecera

se incluye la API Key relativa a la cuenta de ThingSpeak y al canal donde se quieren almacenar esos datos.

Lo último que se hace en este 'Bloque sentencias 2' es iniciar un servidor donde se puede encender o apagar el *buzzer* del dispositivo de seguridad. Este servidor estará esperando clientes en el puerto 9999 de la dirección IP X.X.X.123 durante 10 segundos. Como se dijo antes, se hizo una configuración por la cual la dirección IP del dispositivo será siempre la misma.

Si en ese tiempo no se conecta ningún cliente, el dispositivo entrará en modo *deep sleep* durante 20 segundos. Por el contrario, si se conecta un cliente y envía la cadena 'BUZZER=ON', el *buzzer* comenzará a sonar.

Una vez que está encendido el *buzzer*, se espera durante 10 segundos hasta recibir la petición por parte del cliente de apagar el *buzzer*. Si se recibe dicho mensaje, se vuelve al estado anterior en el que se espera un cliente que encienda el *buzzer*, con su correspondiente contador reiniciado. Si por el contrario no se recibe la cadena esperada durante los 10 segundos, directamente el dispositivo entrará en modo *deep sleep* para que pasados 40 segundos se reinicie el microcontrolador.

Para una mayor comodidad, se incluye aquí una referencia a la imagen (ver imagen 14) que contiene el diagrama con el esquema de funcionamiento del programa del microcontrolador.

Volviendo a la parte superior del diagrama, ahora se va a explicar el comportamiento del programa en el caso en que el dispositivo no pueda conectarse a la red Wi-Fi conocida. El objetivo de este flujo del programa es parecido al comportamiento del 'Bloque sentencias2', es decir, se reúnen una serie de datos para enviarlos al servidor.

En este caso se ejecuta el 'Bloque sentencias 4'. Para reunir los datos y posteriormente enviarlos al servidor, es fundamental estar conectado a alguna red y eso es lo que se hace en este bloque de sentencias. Como se ha dicho ya antes, el dispositivo en esta situación no se encuentra conectado a la red Wi-Fi conocida, por lo que lo primero que se realiza en este bloque es escanear todas las redes que pueden ser detectadas por el microcontrolador.

Para hacer esto es necesario de nuevo utilizar la librería 'ESP8266WiFi.h'. Su función 'WiFi.scanNetworks()' devuelve el número de redes que detecta el microcontrolador. Mediante un bucle que recorre todas estas redes, se comprueba el tipo de seguridad que tiene cada una de ellas y se almacenan en un *array* todas aquellas cuyo cifrado es ninguno. Para saber el tipo de cifrado que tiene una red se utiliza otra función, perteneciente a la misma librería, llamada función 'WiFi.encryptionType()'. Una vez se termina este primer bucle, se entra en otro cuya función es ordenar las redes que no están cifradas según la potencia recibida, con el objetivo de conocer de qué red le llega más potencia al dispositivo de seguridad.

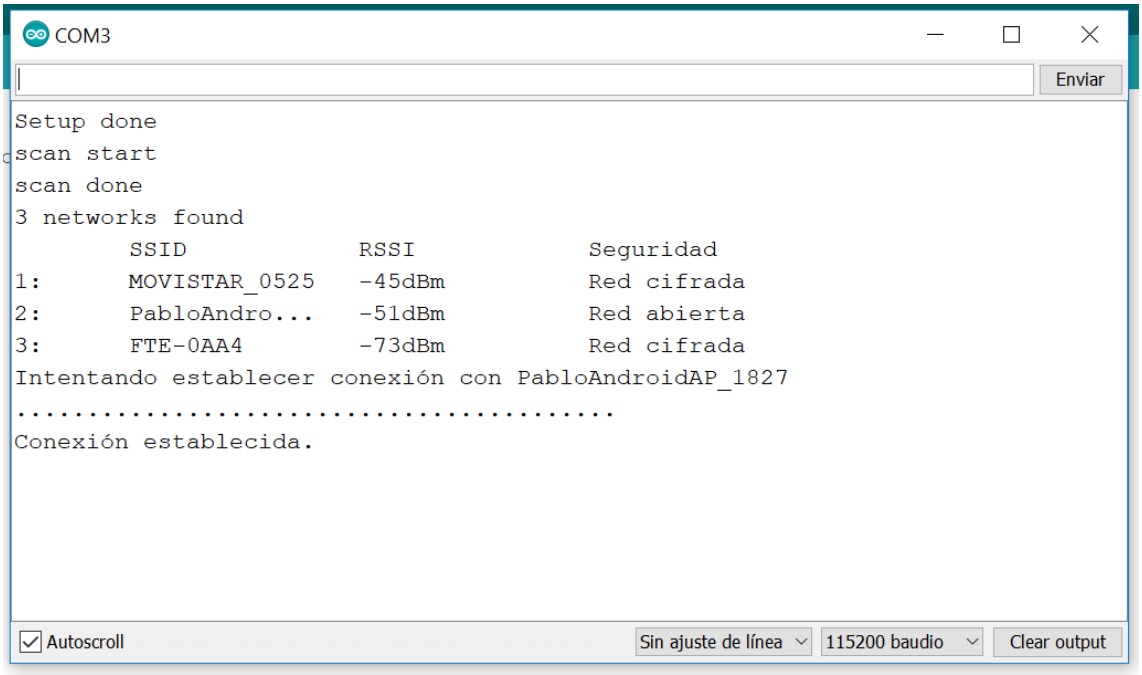

Imagen 19. Puerto serie - Ejemplo redes ordenadas según la potencia recibida

Si después de realizar el anterior bloque de sentencias el dispositivo no ha encontrado ninguna red Wi-Fi que reúna las condiciones, debe reiniciarse inmediatamente y comenzar de nuevo a ejecutar todo el programa. Esta vez no se utiliza el modo *deep sleep* porque se entiende que el portador del dispositivo no estará en un rango cercano al móvil encargado de crear la red Wi-Fi conocida, y además tampoco está enviando al servidor su ubicación, por lo que es muy importante que encuentre cuanto antes una red Wi-Fi a la que poder conectarse.

Por el contrario, si el dispositivo ha encontrado una red que reúna las condiciones, se conectará a ésta y comenzará el llamado 'Bloque sentencias 5'. La función de este bloque es terminar de reunir los datos y enviarlos al servidor de ThingSpeak.

Al igual que se explicó en el bloque de sentencias 2, en este momento también se necesitan los datos que se recogieron en el bloque de sentencias 1 sobre las redes Wi-Fi detectadas por el dispositivo. El procedimiento para conseguir los datos de latitud, longitud y precisión, y para su posterior envío al servidor ThingSpeak, es el mismo que se describió anteriormente.

En esta ocasión, el dato correspondiente a la potencia de la señal Wi-Fi recibida por el microcontrolador no se considera importante porque no será la red conocida. En cambio, los datos correspondientes a la ubicación del dispositivo son ahora muy importantes, ya que si el dispositivo no se encuentra conectado a la red Wi-Fi conocida puede que sea porque no se encuentre en su rango. Por esta razón, cuando al final del programa entra en modo *deep sleep* lo hace durante 20 segundos, que no durante 40 segundos como en el caso descrito anteriormente, para poder reunir la mayor cantidad de ubicaciones posible.

5.2 Funcionamiento de la aplicación móvil

En este apartado se describe en primer lugar la apariencia de la aplicación, y a continuación se explica su funcionamiento. Como ya se dijo anteriormente, esta aplicación está desarrollada para móviles que tengan sistema operativo Android. En concreto, la versión del sistema operativo debe ser igual o mayor a la versión Android 4.0.3 (*IceCreamSandwich*), lo cual incluye aproximadamente al 100% de dispositivos Android.

5.2.1 Interfaz de la aplicación

La aplicación consta de dos pantallas, que se conocen como actividades o '*activities'.* La actividad principal se llama 'mainActivity' y se corresponde con la siguiente imagen:

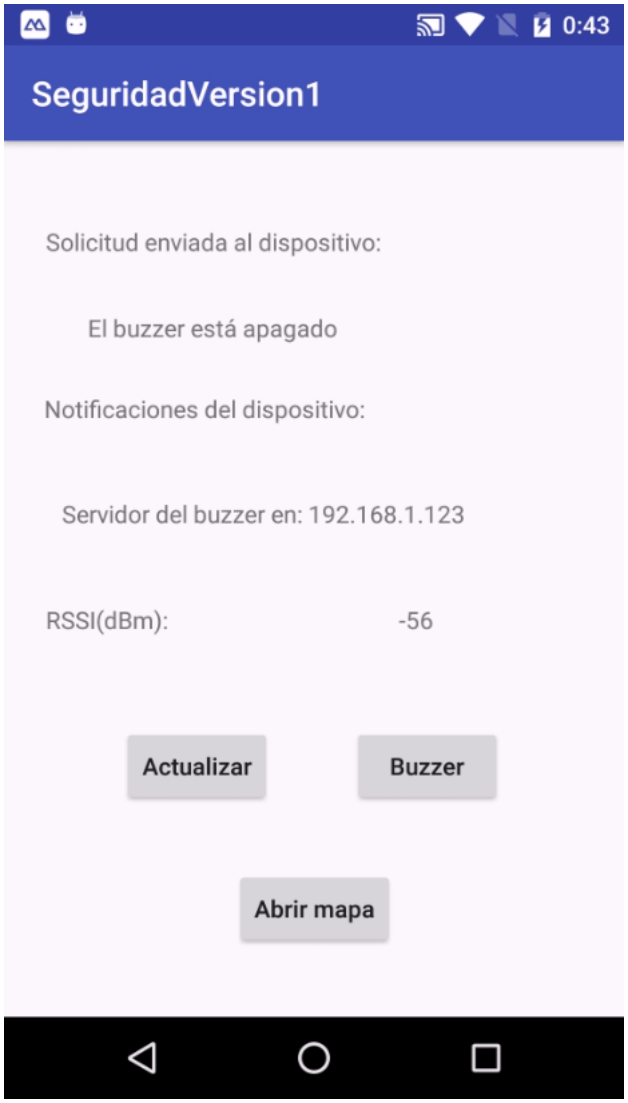

Imagen 20. Interfaz aplicación Android – Pantalla inicial

Como se puede ver, contiene una serie de campos de texto donde se va mostrando información relacionada con la última petición realizada al servidor del *buzzer*, el estado del *buzzer*, la dirección IP del servidor y el último dato de potencia recibida por el dispositivo de seguridad.

La primera pantalla de la aplicación consta además de tres botones con sus correspondientes funciones. Se ha buscado una frase o nombre descriptivo para poner en dichos botones de modo que se reconozca intuitivamente cuál es la función de cada uno.

La otra actividad de la que se compone la aplicación se llama 'mapActivity' y se corresponde con la siguiente imagen:

Imagen 21. Interfaz aplicación Android – Pantalla mapas

La ubicación mostrada en este mapa dependerá inicialmente de los datos de latitud y longitud que se hayan cargado por última vez en la aplicación.

5.2.2 Lógica de la aplicación

El desarrollo de la aplicación para móvil se ha realizado en el lenguaje de programación Java, haciendo uso del entorno de desarrollo integrado Android Studio IDE comentado anteriormente.

Al iniciar la aplicación lo primero que se hace es establecer la dirección IP que tendrá el servidor de *buzzer* del dispositivo de seguridad. Esto dependerá del móvil en que se ejecute la aplicación, ya que los nuevos móviles además de tener la interfaz 'wlan0' que tienen todos, tienen una interfaz llamada 'softap0' que es utilizada cuando se comparten datos.

Esta nueva interfaz 'softap0' es precisamente la que el móvil utilizará para compartir datos al dispositivo de seguridad. La dirección IP del servidor será siempre, porque así se ha decidido, igual a la dirección IP de esta interfaz salvo en su último número, que en el caso del servidor será 123.

Una vez que se ha establecido la dirección IP del servidor, el siguiente paso es obtener los últimos datos que se han almacenado en el servidor ThingSpeak. Para ello se crea una tarea asíncrona en la que se realiza una petición GET a la API del servidor. En esta petición se incluye la información relacionada con el canal del que se desean obtener los datos, la cantidad de datos que se quieren obtener, y la clave o 'API Key' que previamente se debe haber generado en el servidor.

El siguiente paso es realizar una comprobación sobre los datos que se han obtenido en el paso anterior. En concreto, el dato en el que se basa esta aplicación es en el valor de la potencia recibida por el dispositivo de seguridad, y las correspondientes fecha y hora en que este valor fue almacenado en el servidor.

Se realizan dos comprobaciones fundamentalmente en la aplicación.

La primera de ellas es comprobar el tiempo que ha pasado entre los dos últimos datos subidos al servidor. Se ha establecido un tiempo máximo en el que en el peor de los casos el dispositivo de seguridad haya podido actualizar los datos en el servidor. Si la diferencia de tiempo entre las dos últimas publicaciones no supera al tiempo máximo establecido, comienza la segunda comprobación. Si por el contrario la diferencia de tiempos entre publicaciones supera dicho límite, se lanza una notificación y una alarma para que el usuario de la aplicación sea consciente.

La segunda comprobación que se realiza es comparar el valor de la potencia recibida por el dispositivo con un límite que se ha establecido. Si dicho valor es menor que el límite significa que el menor se está alejando o que se ha metido detrás de algún muro u otro elemento que disminuye la potencia que recibe el dispositivo, lo cual también puede significar que haya desaparecido del campo de visión del usuario de la aplicación. Por esta razón, si se da esta situación se activa una alarma y se lanza una notificación diferente a la notificación comentada antes.

Si esta segunda comprobación tampoco se cumple, la aplicación continúa con su comportamiento normal, que consiste en actualizar periódicamente los datos que se van actualizando en el servidor.

Una vez explicado el funcionamiento habitual de la aplicación, a continuación, se va a explicar la función que cumple cada uno de los elementos de la aplicación.

Como se ha comentado antes, existen dos tipos de notificaciones. Cada una de ellas muestra un mensaje diferente, cada uno adecuado a la condición por la que ha sido lanzada dicha notificación. Cuando se pulsa la notificación se abre la actividad principal de la aplicación.

La alarma es una alarma personalizada que se ha creado para que sea fácilmente reconocible. Consiste en un *array* de tiempos que se corresponden con el tiempo que permanece activado el vibrador del móvil y el tiempo que permanece desactivado, alternativamente. Dicho *array* reproduce la secuencia de pulsos y silencios que se corresponde con la señal de SOS en código morse.

Por otra parte, como se describió en el apartado anterior, la actividad principal de la aplicación cuenta con tres botones. Cada uno de ellos ejecuta una acción distinta:

- 'Actualizar datos': este botón, como su propio nombre indica, sirve para actualizar a voluntad los valores que hay almacenados en el servidor.
- 'Buzzer': este botón envía un mensaje al servidor de *buzzer* que está implementado en el dispositivo. Este hará que el *buzzer* se encienda o se apague cada vez que se reciba el mensaje que se envía mediante este botón.
- 'Abrir mapa': tal como se puede suponer, este botón abre la actividad 'mapActivity' de la aplicación.

Por último, se ha desarrollado un marcador personalizado para la actividad 'mapActivity'. Este marcador se muestra en el mapa con un círculo sombreado y con contorno azul oscuro a su alrededor. El radio de este círculo en metros se corresponde con el dato de la precisión que se obtuvo en el dispositivo de seguridad a través de la Geolocation API de Google.

5.3 Funcionamiento del *plugin* de ThingSpeak

En este último apartado del capítulo se describen la apariencia que tiene y la función que realiza el *plugin* desarrollado para ThingSpeak.

Aprovechando una de las funcionalidades que ofrece el servidor que se ha elegido para este proyecto, se ha desarrollado un *plugin* que sirve de apoyo a la aplicación móvil. Para acceder a este servicio hay que iniciar sesión en ThingSpeak, ir al desplegable 'Apps' y seleccionar 'Plugins'.

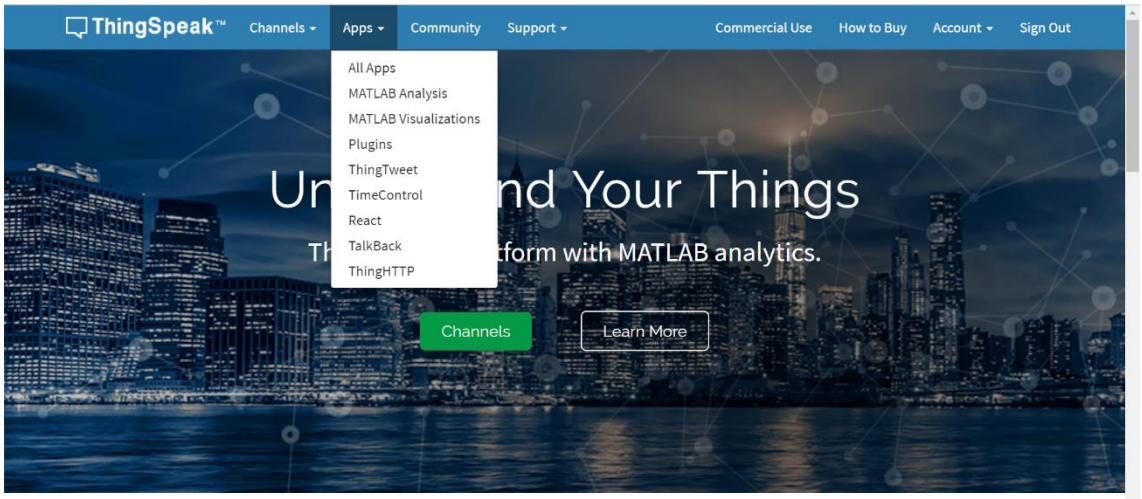

Imagen 22. Interfaz inicio ThingSpeak

Una vez hecho esto aparece en la página web una lista con todos los *plugins* que se han creado para el usuario con el que se ha iniciado sesión. El siguiente paso es seleccionar el *plugin*, que para el caso de este proyecto es 'Coordenadas ESP8266 en Google Maps'.

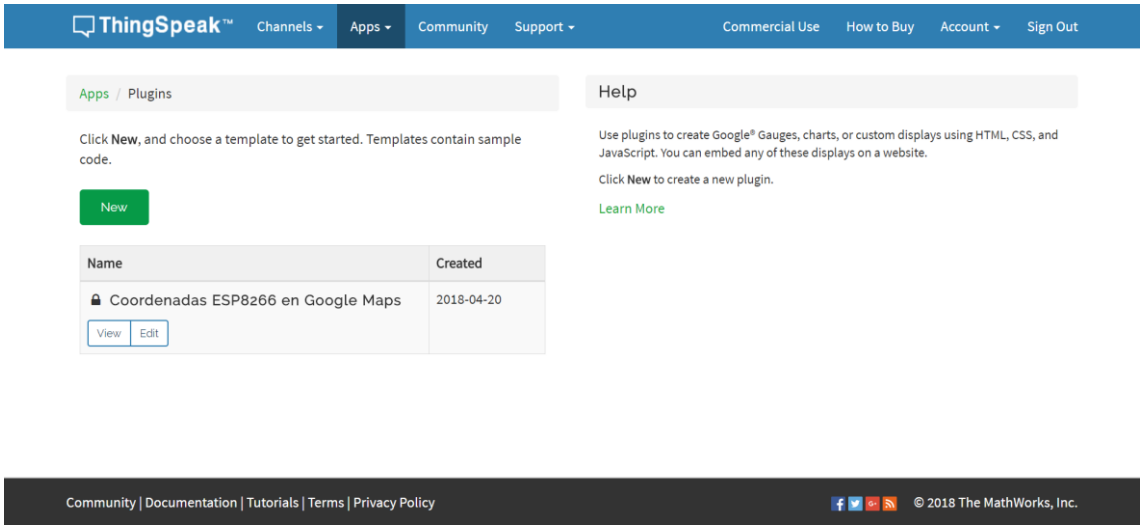

Imagen 23. ThingSpeak - Selección del plugin creado

En el código desarrollado en JavaScript correspondiente a este *plugin* se realiza una descarga de los datos relativos a las últimas coordenadas almacenadas en el servidor, se crea un marcador con esas coordenadas descargadas, se ajusta el zoom en el mapa para la zona alrededor del marcador creado y se muestra a pantalla completa ese mapa donde se marca mediante un icono la posición.

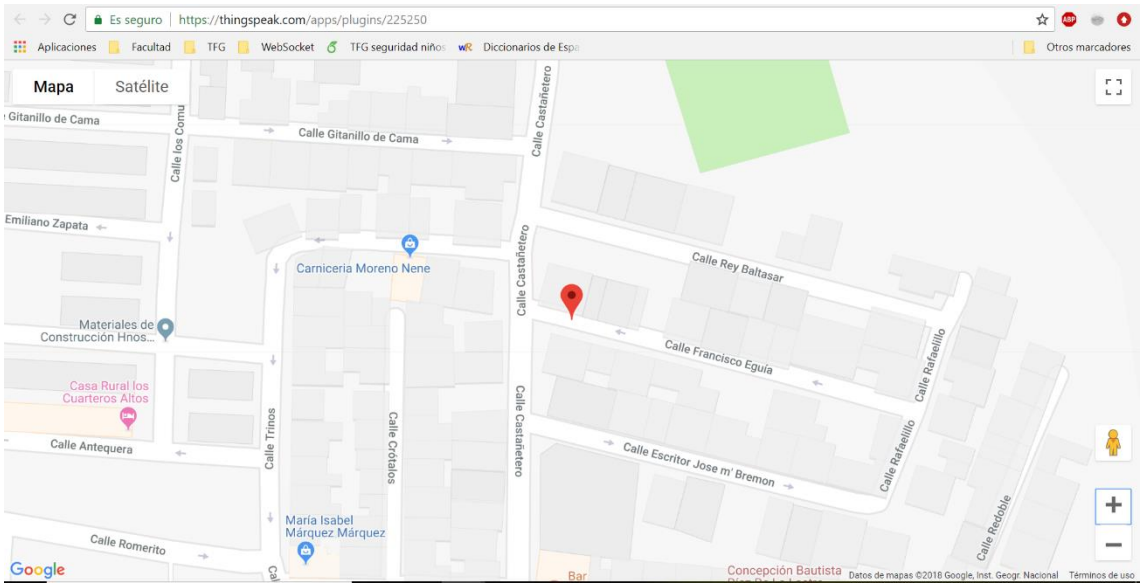

Imagen 24. Ejemplo prueba de geolocalización real en interfaz de ThingSpeak

6. Pruebas y resultados

A lo largo de este capítulo se presentan una serie de pruebas que se han realizado con todos los programas que intervienen en el presente trabajo final de grado. Al inicio del capítulo se incluyen pruebas unitarias que demuestran el funcionamiento y refuerzan la explicación de las tareas que realizan dichos programas. Posteriormente, se plantean y demuestran una serie de escenarios que podrían darse durante el funcionamiento habitual del dispositivo.

6.1 Prueba 1 – Conexión a una red establecida previamente

La primera prueba que se incluye es, por una parte, fundamental para el funcionamiento del dispositivo y, por otra parte, la más básica que se puede realizar. Se trata de comprobar que el dispositivo es capaz de conectarse a una red Wi-Fi. Para probar su funcionamiento se ha incluido una captura de pantalla del puerto serial del Arduino IDE:

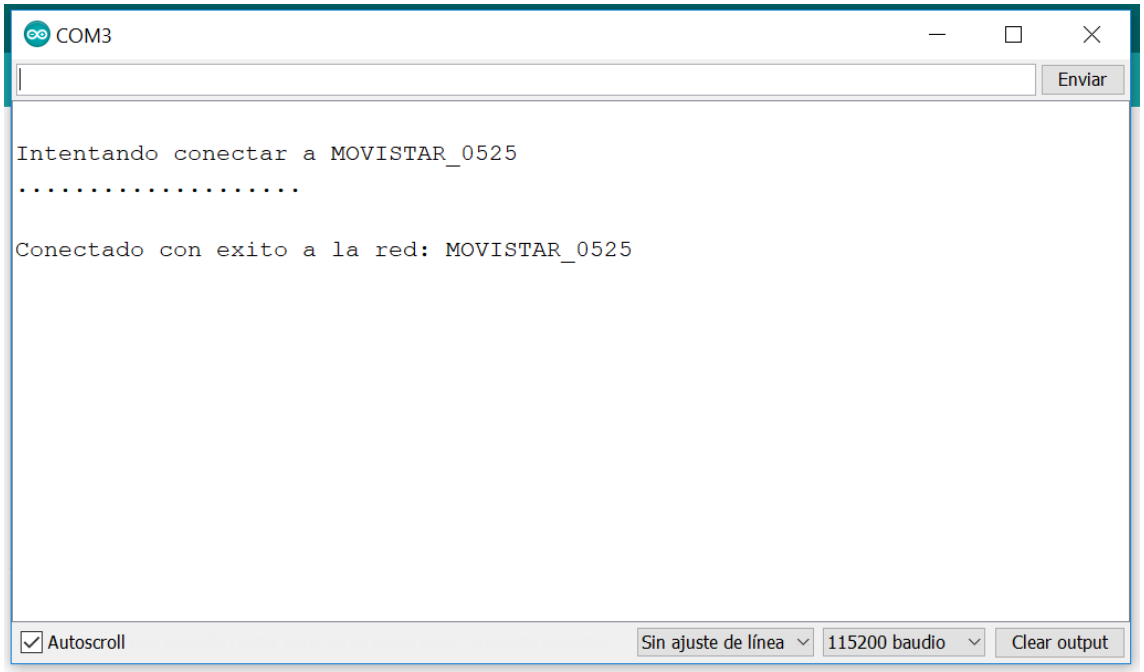

Imagen 25. Prueba 1 - Conexión a una red establecida previamente

6.2 Prueba 2 – Funcionamiento temporizador conexión fallida

A continuación, se muestra el caso contrario, cuando el dispositivo no ha podido conectarse a una red Wi-Fi en concreto. Esto puede suceder por varios motivos como por ejemplo que la red no exista, que la contraseña sea errónea, que la potencia sea muy baja, etc. Para este caso, la conexión no se ha realizado porque se ha sobrepasado el tiempo de espera máximo establecido para que se produzca esa conexión. En concreto, el tiempo de espera ha sido de 10 segundos. Se puede comprobar contando los puntos que aparecen en la imagen, correspondiéndose cada uno de ellos con el paso de 500 milisegundos.

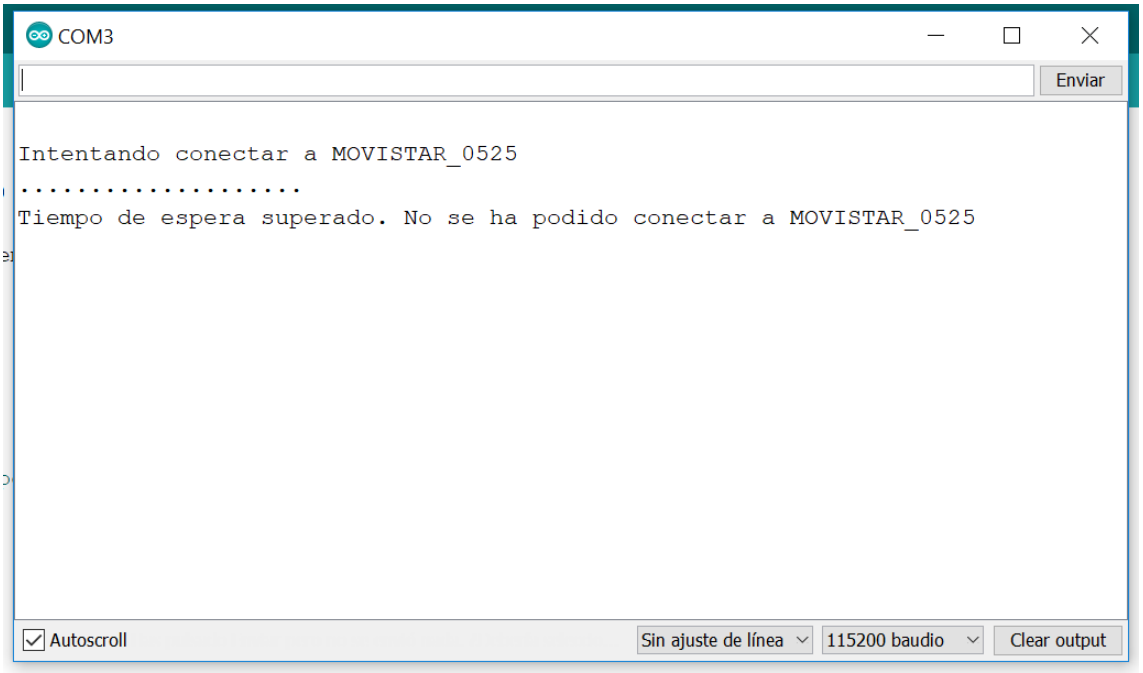

Imagen 26. Prueba 2 - Funcionamiento temporizador cuando se produce una conexión fallida

6.3 Prueba 3 – Configuración de direccionamiento IP

Para el correcto funcionamiento del dispositivo es necesario que el dispositivo siempre tenga una dirección IP fija. Para ello hay que realizar una configuración en la red Wi-Fi. En el desarrollo del programa para el dispositivo de seguridad se ha hecho uso de la función 'WiFi.config(param1, param2, param3)' para este fin.

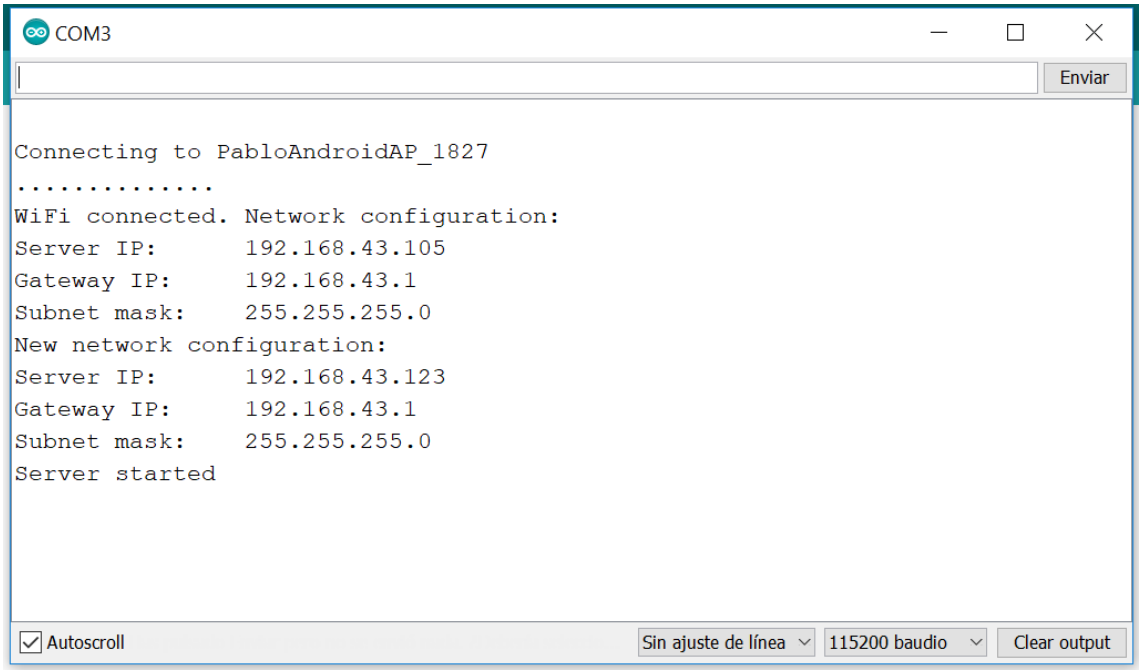

Imagen 27. Prueba 3 - Configuración del direccionamiento IP

6.4 Prueba 4 – Medida de la potencia recibida

Otra función básica que debe realizar el dispositivo es medir la potencia recibida de la red Wi-Fi a la que se encuentre conectado. Para ello se hace uso de la función 'WiFi.RSSI()' perteneciente a la librería 'ESP8266WiFi.h':

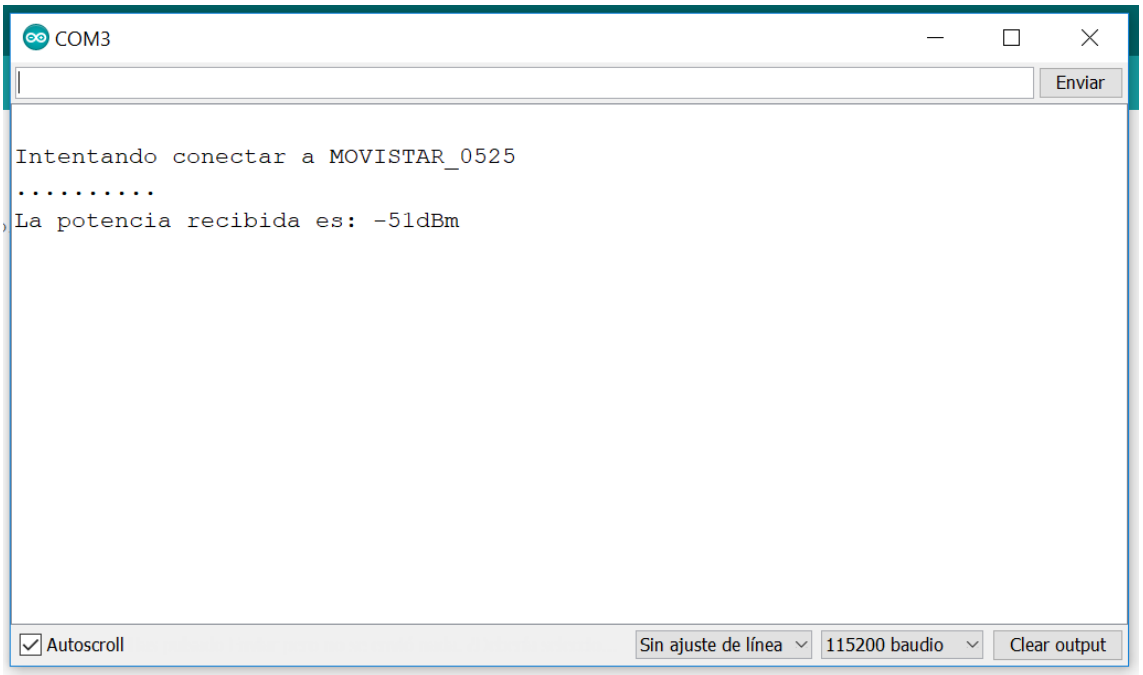

Imagen 28. Prueba 4 - Medida potencia recibida

6.5 Prueba 5 – Escaneo de redes Wi-Fi

Entre las posibilidades que se han contemplado, es posible que el dispositivo de seguridad no sea capaz de conectarse a una red Wi-Fi conocida. En este caso, como ya se dijo en el capítulo anterior, el dispositivo debe realizar una búsqueda y selección de otra red alternativa a la que conectarse. Para esto, el primer paso que hay que seguir es comprobar qué redes es capaz de detectar el microcontrolador.

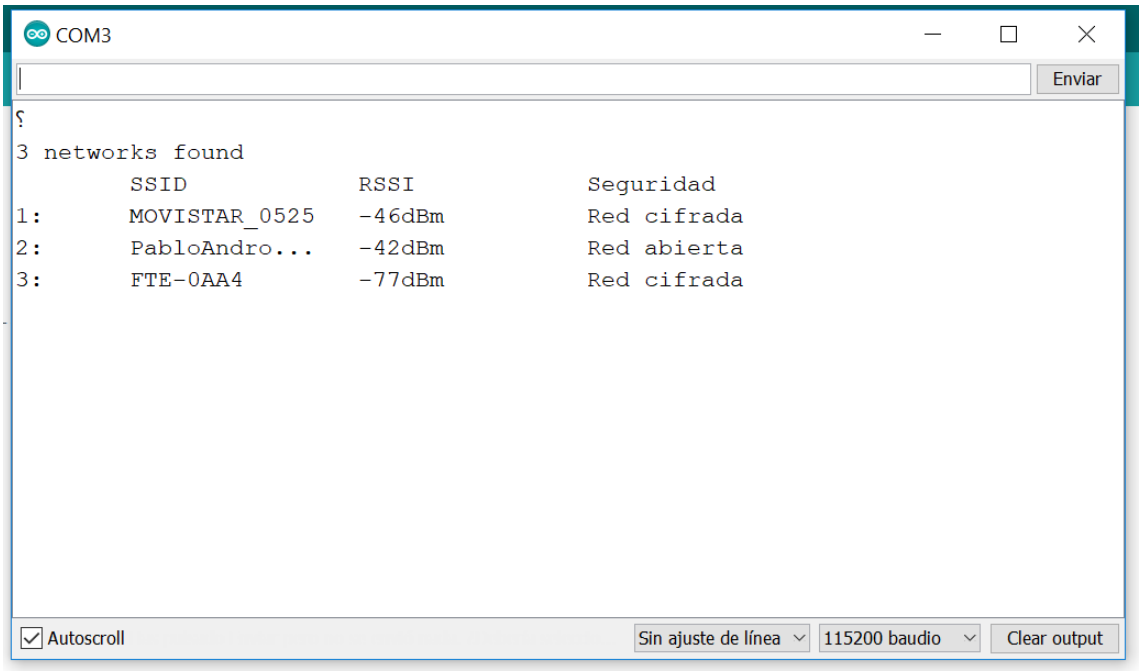

Imagen 29. Prueba 5 - Escaneo de redes Wi-Fi

El microprograma a partir del cual se ha sacado la captura de pantalla anterior también clasificaba las redes encontradas en función del cifrado que estas tenían. Para ello se utilizaba la función 'WiFi.encryptionType()' dentro de un bucle que recorría todas las redes encontradas. Entre los distintos valores que puede devolver esa función, si devolvía 'ENC_TYPE_NONE', la red se marcaba como 'red abierta', y en caso contrario, como 'red cifrada'.

6.6 Prueba 6 – Funcionamiento del modo *deep sleep*

Otra de las funciones que se han utilizado en el dispositivo de seguridad ha sido la función 'DeepSleep(param1)'. Esta función recibe como parámetro un número que se corresponde con la cantidad de microsegundos que el dispositivo va a permanecer en modo *deep sleep* antes de reiniciarse.

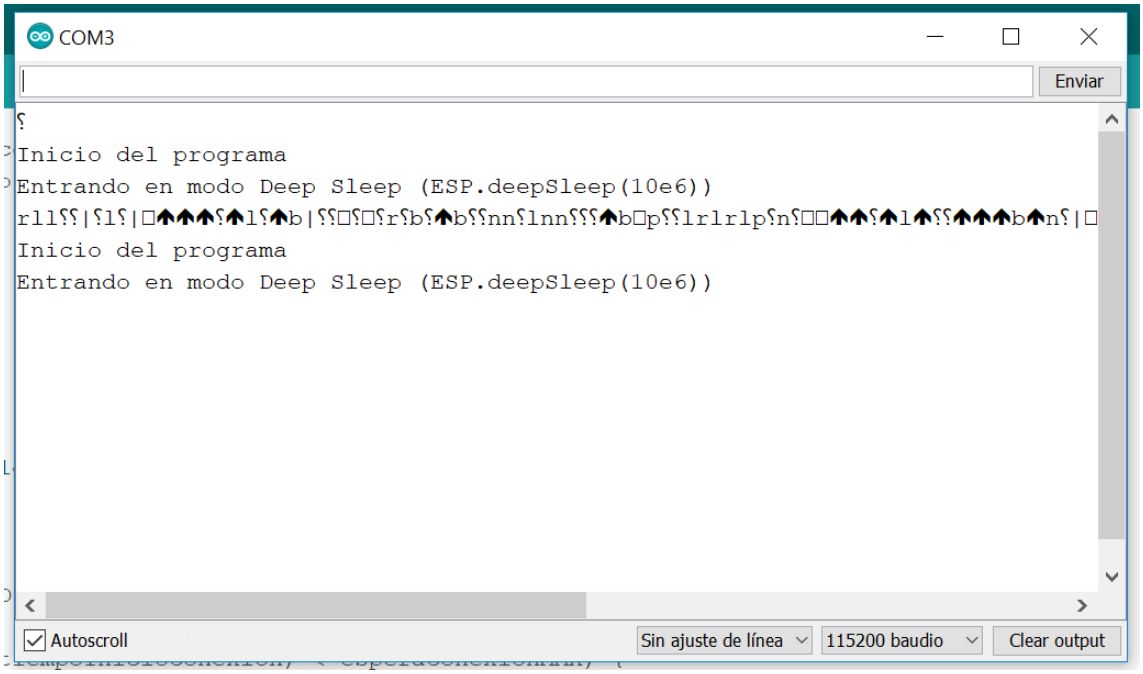

Imagen 30. Prueba 6 - Funcionamiento modo deep sleep

6.7 Prueba 7 – Petición a ThingSpeak

A partir de aquí las pruebas que se incluyen son algo más complejas. La siguiente prueba que se va a realizar es enviar al servidor una serie de datos. A modo de ejemplo, se envían al servidor los valores 20 y -20. En la siguiente imagen se muestra la petición que se envía desde el dispositivo de seguridad al servidor ThingSpeak.

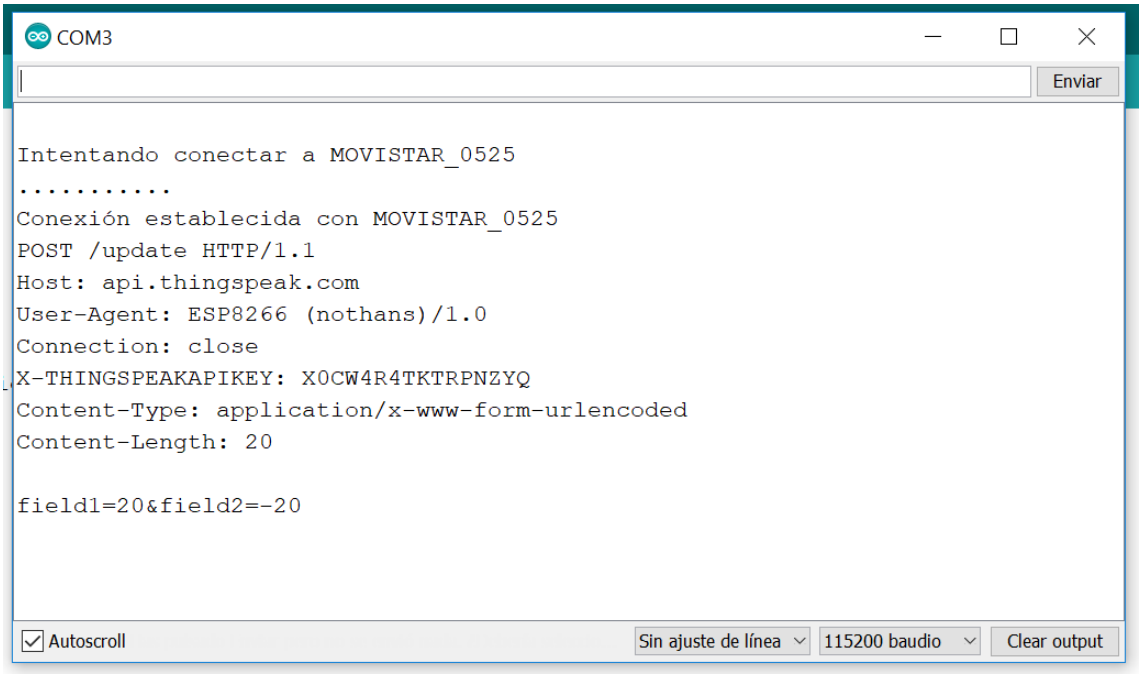

Imagen 31. Prueba 7 - Petición a ThingSpeak

Como consecuencia de esta petición, en el servidor ThingSpeak se han almacenado los datos en los respectivos campos. En la siguiente imagen se muestra una captura de un canal que se ha creado específicamente para esta prueba, de manera que no haya otros datos que puedan entorpecer lo que se quiere demostrar.

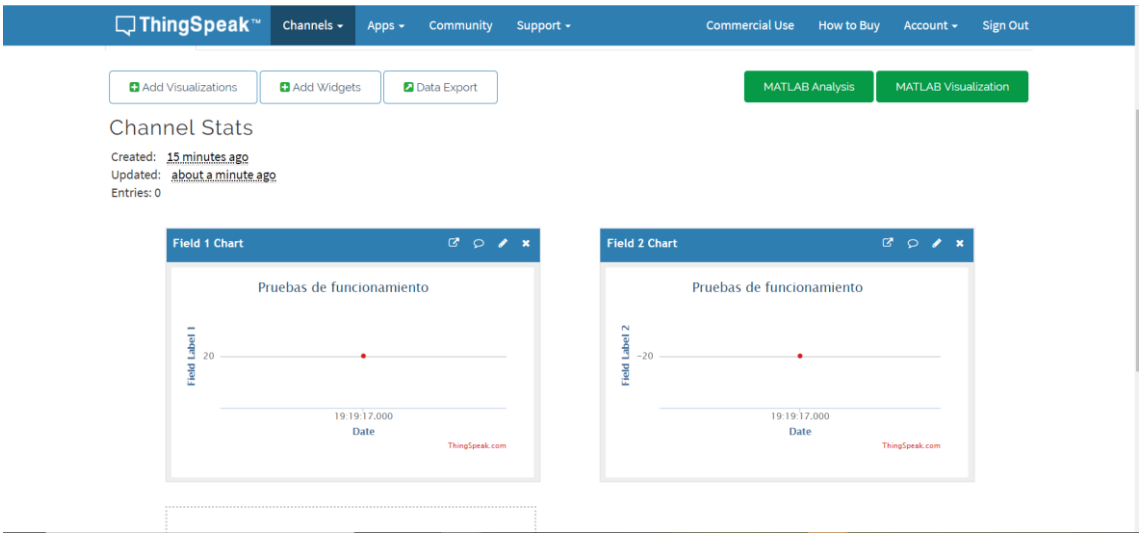

Imagen 32. Interfaz ThingSpeak - Comprobación petición OK

6.8 Prueba 8 – Petición a Geolocation API

A continuación, se incluye otra prueba algo más compleja que las anteriores. En esta ocasión lo que se demuestra es la obtención de las coordenadas a través de la API de Geolocalización de Google. En las siguientes imágenes se muestra la petición que el microcontrolador realiza a la API y la respuesta que este devuelve:

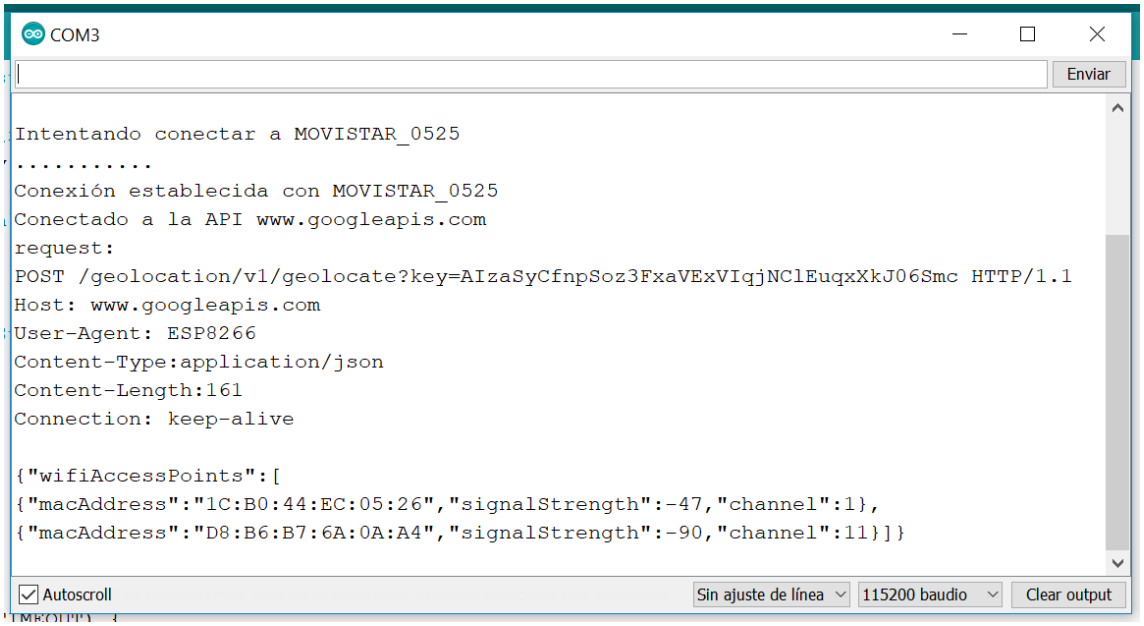

Imagen 33. Prueba 8 - Petición a Geolocation API

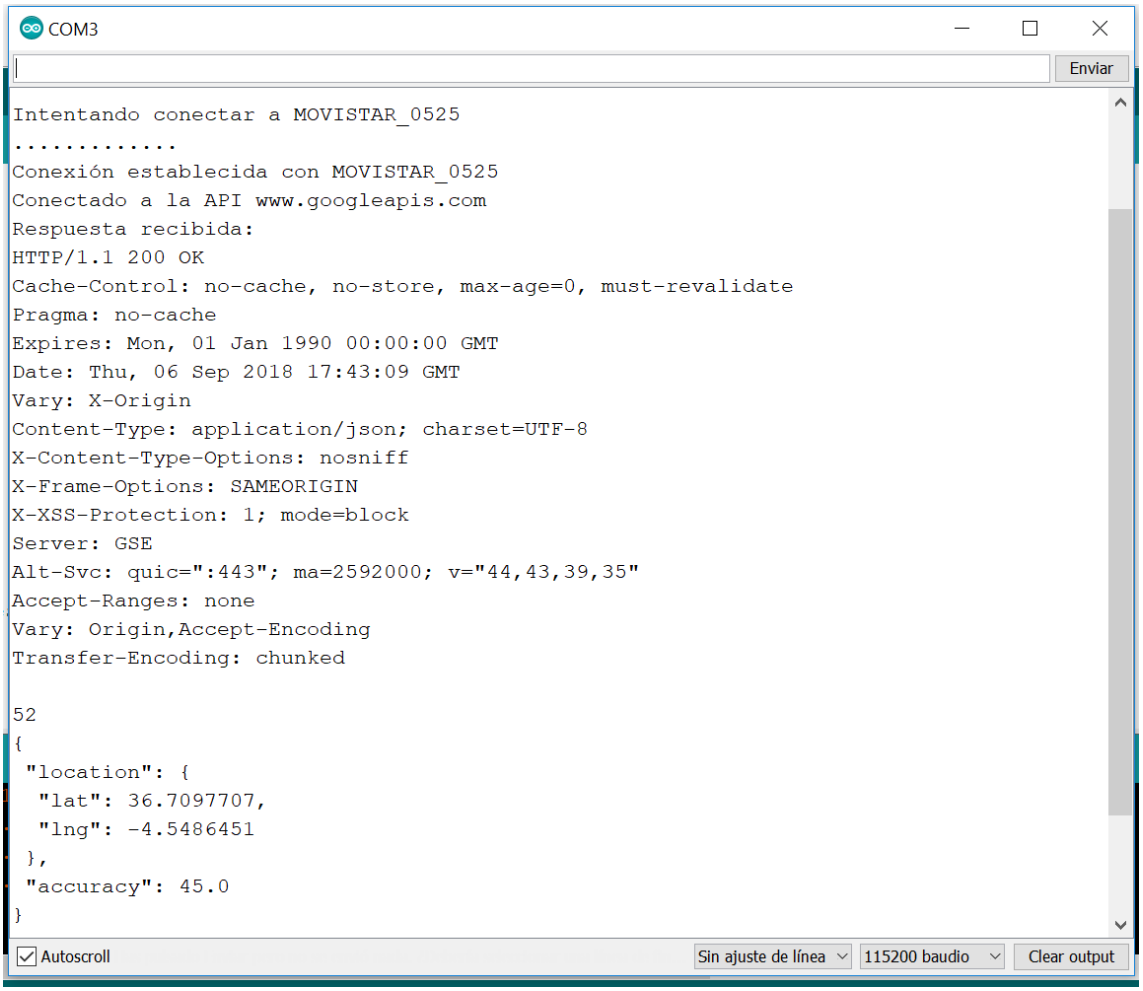

Imagen 34. Prueba 8 - Respuesta Geolocation API

Para poder utilizar la *Geolocation API* de Google previamente se creó una cuenta de Google y un proyecto. Durante el tiempo en que se ha estado desarrollando el presente trabajo final de grado han cambiado las políticas de facturación de *Google Cloud Platform* y ha sido obligatorio establecer unos datos de facturación asociados al proyecto, aunque gracias a que se hace un regalo a los nuevos usuarios de 300$ para consumir usando la plataforma durante los 12 primeros meses, no se ha tenido que realizar ningún desembolso económico. En la siguiente imagen se ve el nombre del proyecto creado para tal fin, junto con una serie de datos sobre peticiones realizadas:

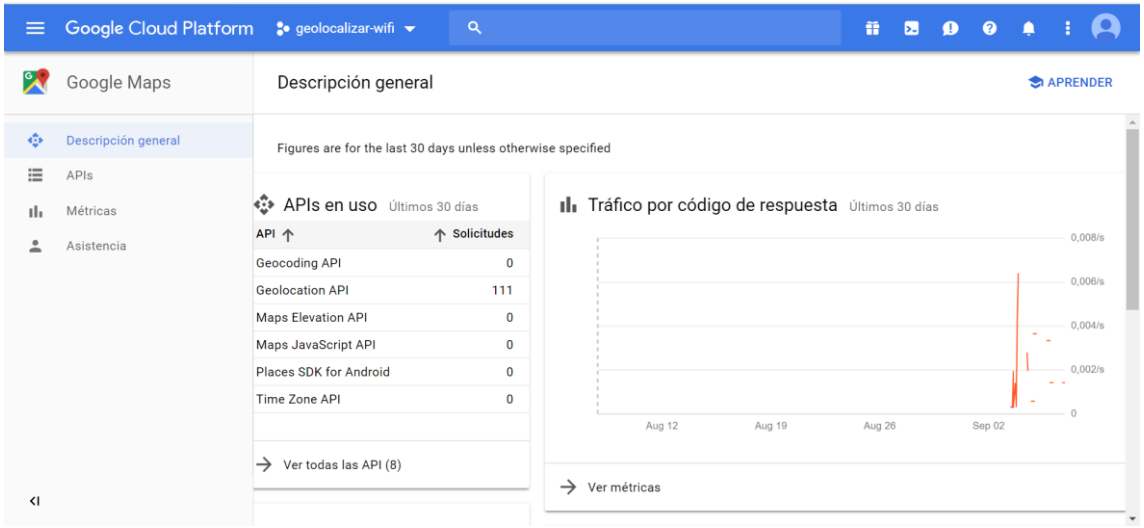

Imagen 35. Interfaz Google Cloud Platform

6.9 Prueba 9 – Lanzamiento de notificaciones en la app de Android

La siguiente prueba que se incluye está relacionada con la aplicación implementada para móviles con sistema operativo Android. En ella se lanzan los dos tipos de notificaciones que se han creado para la aplicación, y se pueden ver en las siguientes imágenes:

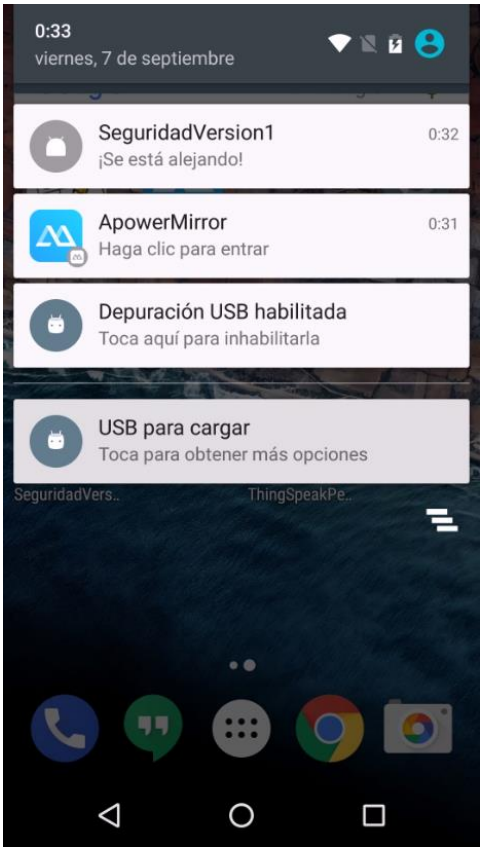

Imagen 36. Prueba 9 - Lanzamiento de notificaciones (1) en la app de Android

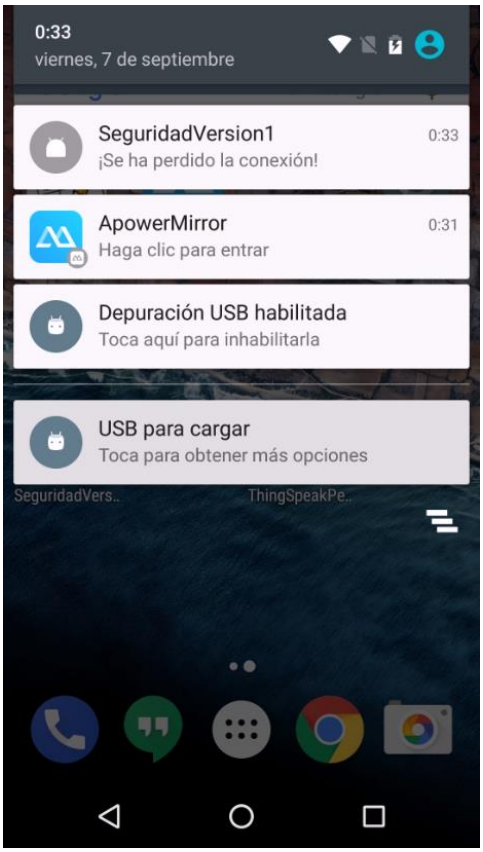

Imagen 37. Prueba 9 - Lanzamiento de notificaciones (2) en la app de Android

En esta última parte del capítulo se recogen una serie de pruebas más elaboradas que engloban las tareas descritas y mostradas anteriormente.

6.10 Prueba 10 – Búsqueda de redes alternativas abiertas

Como primera prueba elaborada se incluye la supuesta situación en la que el dispositivo de seguridad no ha sido capaz de conectarse a una red Wi-Fi conocida y que por consiguiente realiza una búsqueda y filtrado de las redes Wi-Fi que se encuentran en su rango. Puede verse el comportamiento del dispositivo de seguridad en la siguiente imagen:

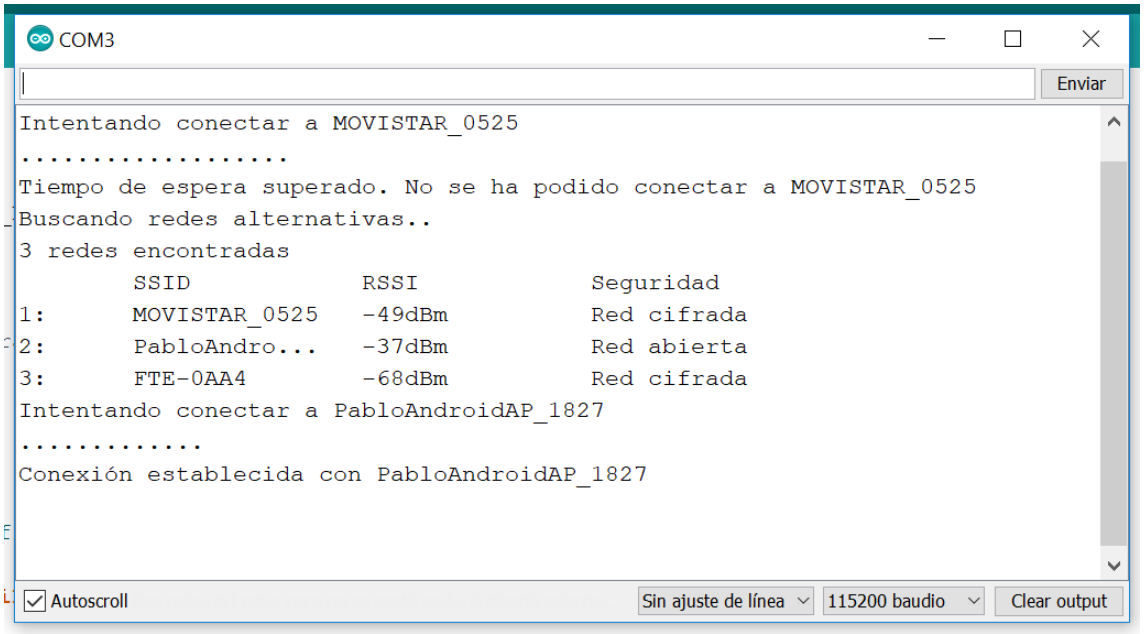

Imagen 38. Prueba 10 - Búsqueda redes alternativas abiertas

6.11 Prueba 11 – Transición de estados en el dispositivo de seguridad cuando el buzzer recibe peticiones

La siguiente prueba elaborada muestra los distintos estados por los que pasa el dispositivo de seguridad cuando su servidor de *buzzer* recibe peticiones. En primer lugar, se muestra la casuística de que el servidor no reciba ninguna petición. En ese caso cuando se agota el tiempo de espera entra en modo *deep sleep* para después comenzar de nuevo.

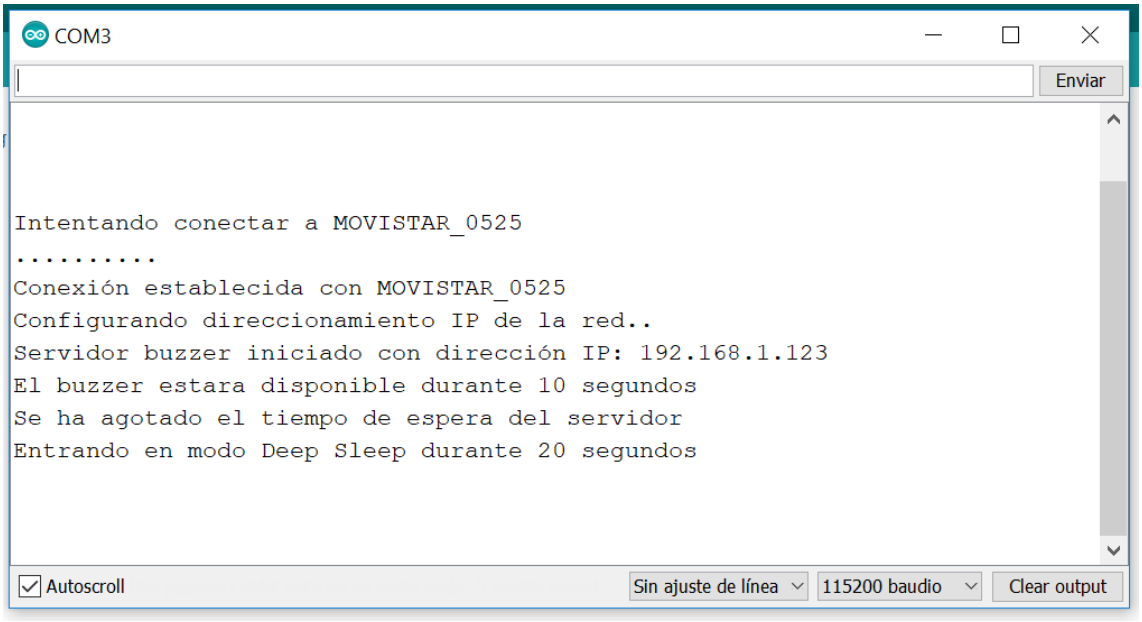

Imagen 39. Prueba 11 – Transición de estados en el dispositivo de seguridad cuando el buzzer recibe peticiones

6.12 Prueba 12 – Recepción de petición de encendido en el servidor de buzzer sin recepción de petición de apagado. Entrada en modo deep sleep

La siguiente casuística que se incluye es cuando el servidor de *buzzer* recibe solamente una petición para encenderlo, pero no recibe la correspondiente petición para apagarlo. En ese caso, el *timer* del servidor se agota y el dispositivo entra en modo *deep sleep*.

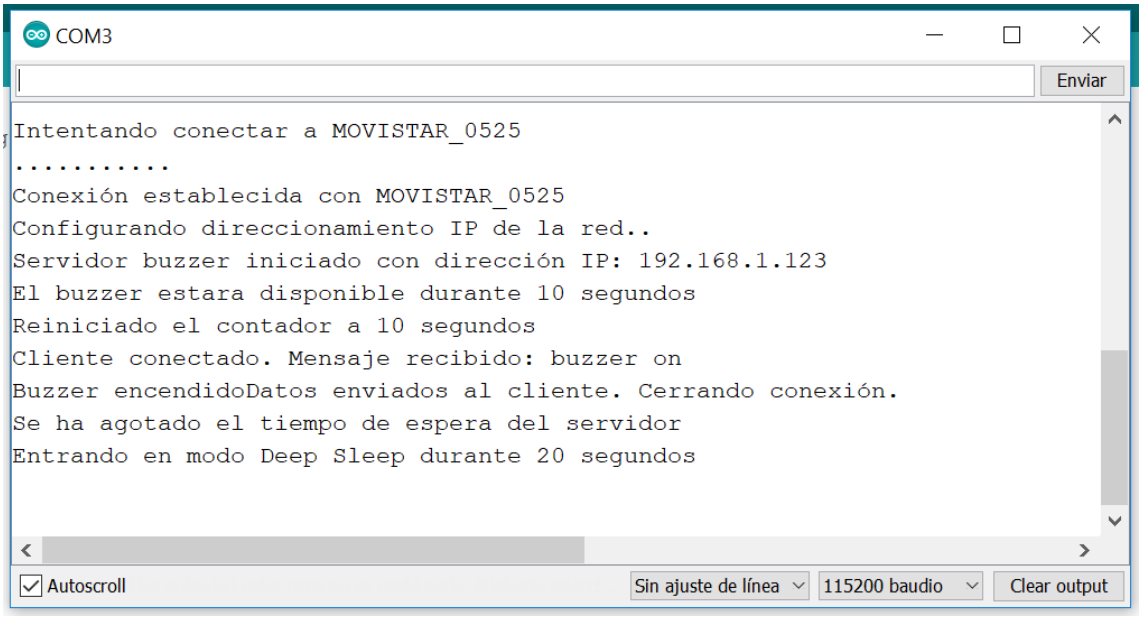

Imagen 40. Prueba 12 – Recepción de petición de encendido en el servidor de buzzer sin recepción de petición de apagado. Entrada en modo deep sleep

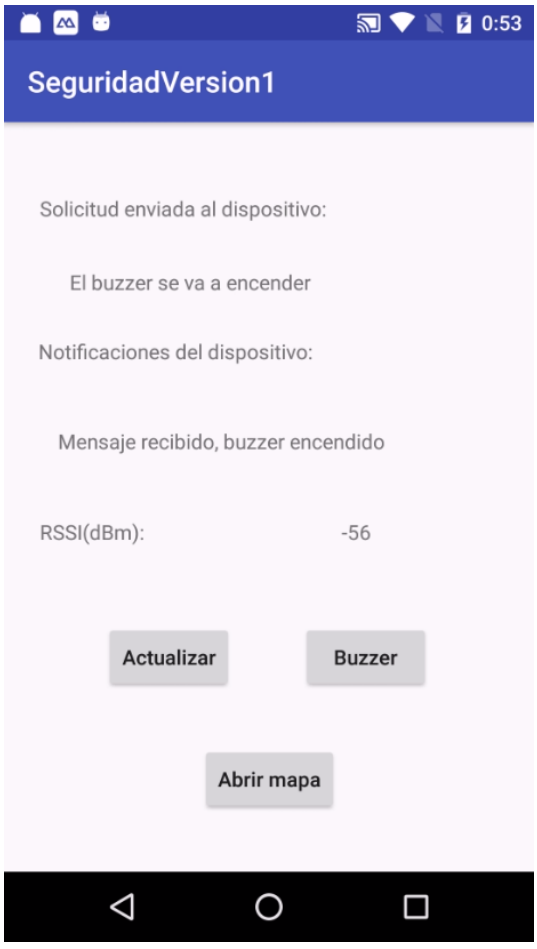

Imagen 41.Prueba 12 – Recepción de petición de encendido en el servidor de buzzer sin recepción de petición de apagado. Vista desde interfaz de la aplicación en Android

6.13 Prueba 13 – Recepción en servidor de buzzer de petición de encendido y apagado

La última casuística que queda por incluir es el caso en que el servidor de buzzer recibe dos peticiones, una para encender el buzzer y otra para apagarlo. En ese caso, los mensajes que muestra el dispositivo de seguridad son los siguientes:

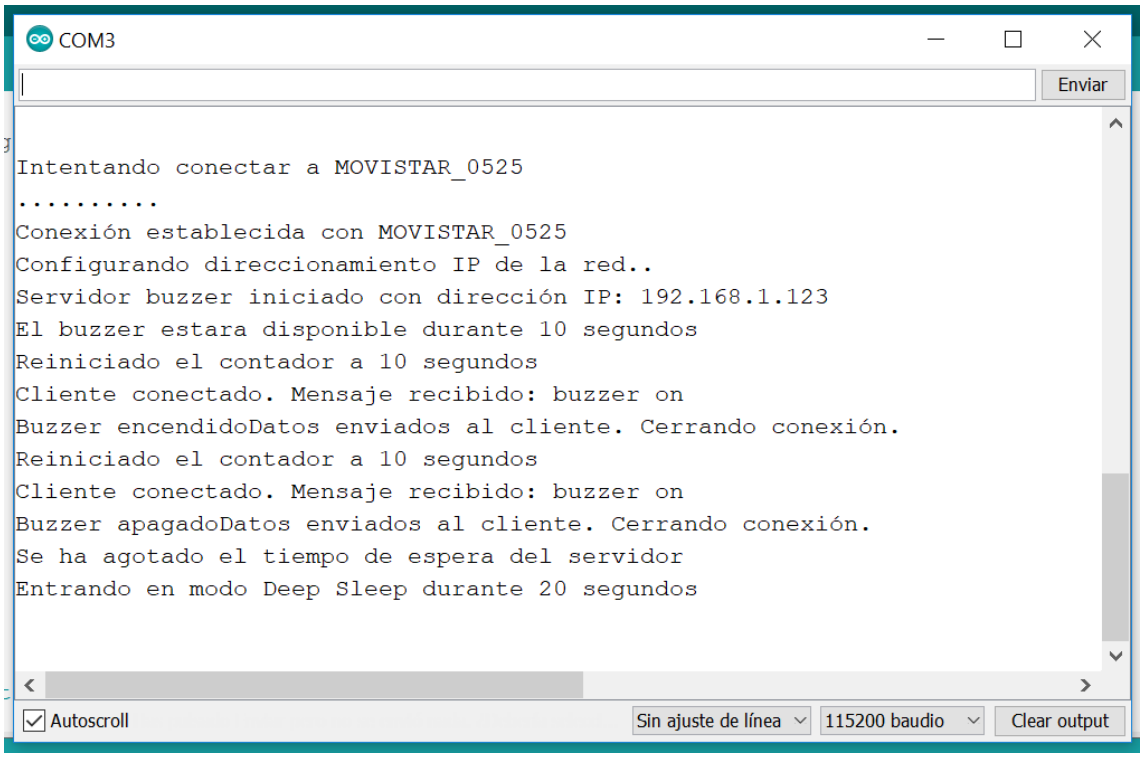

Imagen 42. Prueba 13 – Recepción en servidor de buzzer de petición de encendido y apagado

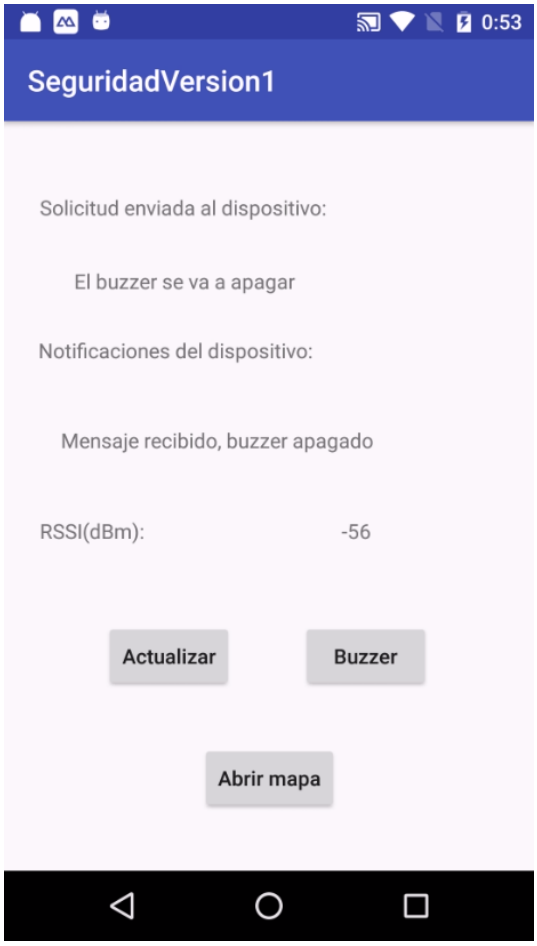

Imagen 43. Prueba 13 – Recepción en servidor de buzzer de petición de encendido y apagado. Vista desde la interfaz de la aplicación en Android

6.14 Prueba 14 – Proceso completo desde realización de petición a la API de Geolocation hasta la representación de las coordenadas de localización

Por último, se muestra todo el proceso seguido desde que el dispositivo de seguridad realiza una petición a la API de Geolocalización de Google hasta que estas coordenadas son representadas tanto en la aplicación para móvil como en el *plugin* de ThingSpeak.

Como previamente se ha demostrado con más detalle el funcionamiento de algunas de las tareas que se realizan en este proceso, en la siguiente imagen se muestra menos información que en las capturas de pantalla del microcontrolador anteriores. En

concreto se muestra la información enviada a la API y los datos contenidos en su respuesta.

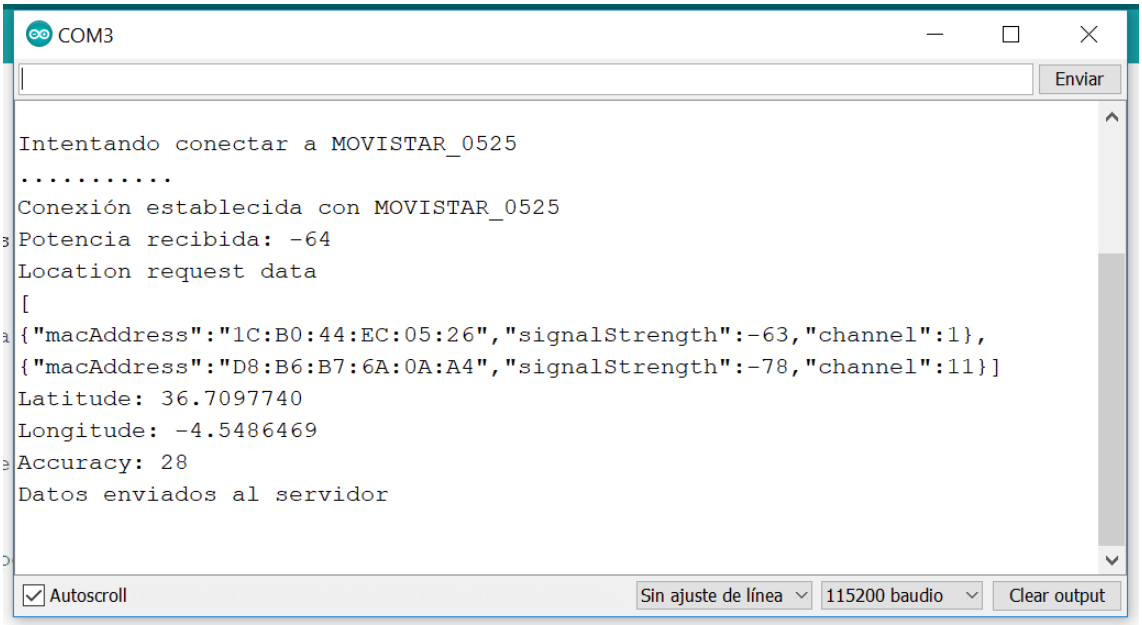

Imagen 44. Prueba 14 - Subida de los datos de localización al servidor

Tal y como puede verse en la anterior imagen, en este momento ya se encuentran los datos de la ubicación subidos en el servidor y por lo tanto accesibles para los demás programas. En la siguiente imagen se ve cómo estos valores han sido actualizados y representados en las gráficas de la página web del servidor ThingSpeak.

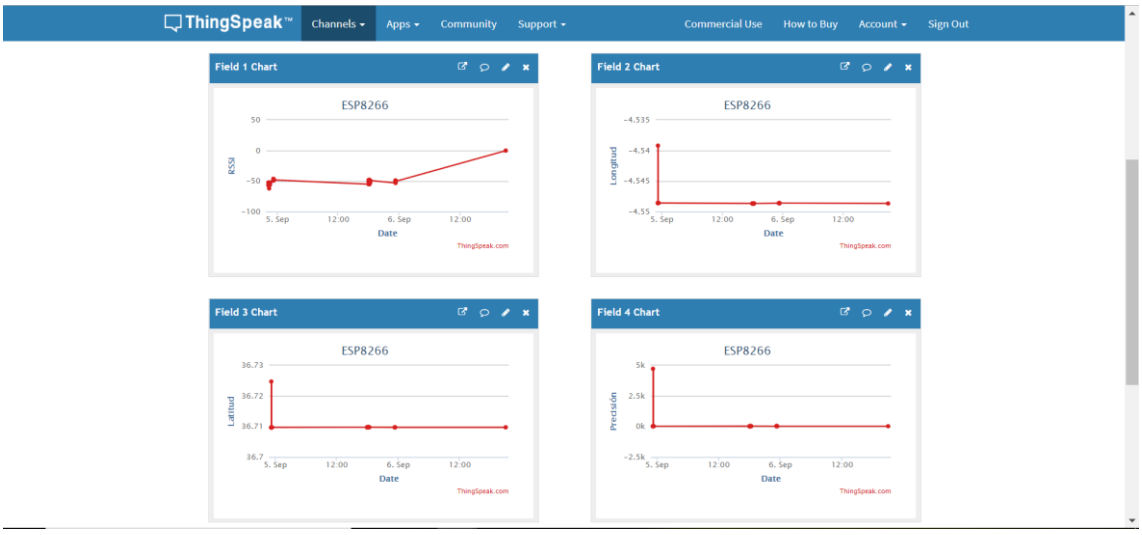

Imagen 45. Interfaz ThingSpeak - Comprobación datos subidos al servidor

En las imágenes que se incluyen a continuación se muestra el momento en que tanto el plugin de ThingSpeak como el 'mapActivity' de la aplicación de móvil han actualizado los datos de posición y muestran en sus respectivos mapas la ubicación mediante un marcador.

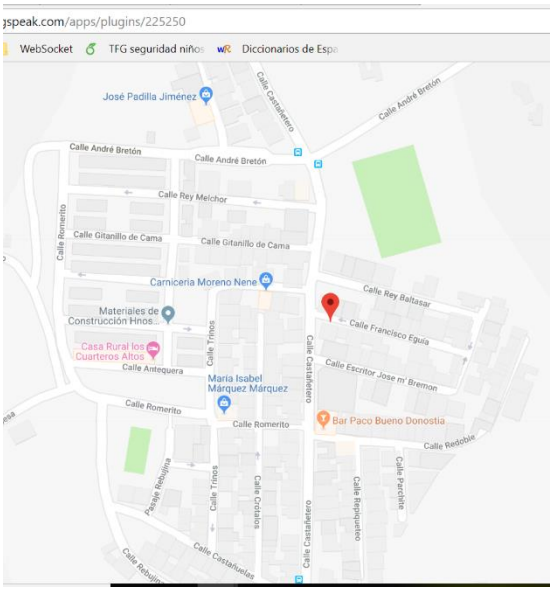

Imagen 46. Interfaz ThingSpeak - Comprobación de la actualización de la posición

Imagen 47. Interfaz App en Android - Comprobación localización dentro de un rango

7. Presupuesto

En este capítulo se especifican los costes asociados a la elaboración del trabajo final de grado, así como los costes relacionados con la implementación del dispositivo.

Los costes se encuentran desglosados en varios grupos, correspondientes a la mano de obra, el material necesario para el desarrollo y el material necesario para la implementación del dispositivo.

En primer lugar, se muestra la tabla que especifica los costes asociados a la mano de obra para unas determinadas horas de trabajo. La mano de obra se divide en horas del ingeniero y horas del supervisor:

Mano de obra	**Cantidad [horas]**	**Coste unitario [€/h]**	**Coste total [€]**
Ingeniero	300	20	6000
Supervisor	30	50	1500
Subtotal			**7500**

Tabla 2. Presupuesto - Subtotal Mano de obra

A continuación, se muestran en forma de tabla los costes asociados a la adquisición del material necesario para el desarrollo del proyecto, así como para la implementación del dispositivo:

Material para la elaboración del estudio	**Concepto**	**Cantidad**	**Coste total [€]**
Lenovo Yoga 700 [12]	Ordenador personal	1	224,78*
Documentación (valor aprox.)	Libros, publicaciones, etc.	-	0,00
Subtotal			**224,78**

Tabla 3. Presupuesto - Subtotal Material para la elaboración del estudio

Material para la implementación del dispositivo	Concepto	Cantidad	Coste unitario	Coste total [€]
ESP8266 Wemos d1 Mini Pro [14]	Microcontrolador	1	3,58	3,58
Buzzer [13]	-	1	0,41	0,41
Xiaomi RedMi Note 5 Pro [16]	Móvil personal 1	1	209	52,25*
Nexus 5 [17]	Móvil personal 2	1	179,99	45,00*
Otros (valor aprox.) [15]	*Protoboard* de tamaño pequeño, cables, etc.	1	2,85	2,85
Subtotal				**104,09**

Tabla 4. Presupuesto - Subtotal Material para la implementación del dispositivo

(*) Indica el precio correspondiente a su uso durante los meses que ha durado la realización del proyecto. Suponiendo que dichos dispositivos tienen una vida útil de tres años, su uso durante nueve meses significaría un 25% de su precio total.

Por último, en la siguiente tabla se recoge el presupuesto final, que comprende los subtotales obtenidos en las tablas anteriores:

Tipo de coste	Coste total [€]
Mano de obra	7500
Material necesario para la elaboración del estudio	224,78
Material necesario para la implementación del dispositivo	104,09
TOTAL	**7828,87**

Tabla 5. Presupuesto TOTAL

8. Conclusiones

Tras la realización del presente trabajo final de grado se puede decir que se han cumplido satisfactoriamente todos los objetivos que se marcaron inicialmente. Además de estos objetivos, a la par que se iban consiguiendo avances en el proyecto iban surgiendo nuevas posibilidades y mejoras. Muchas de estas nuevas ideas se han implementado, pero hay otras que no, ya sea porque se alejaban de la idea principal del proyecto o porque conllevarían mucho tiempo más de desarrollo del que se disponía.

8.1. Líneas de trabajo futuras

El estado actual del diseño hardware realizado para el dispositivo de seguridad es solo un primer prototipo. En este aspecto se puede mejorar mucho porque existen microcontroladores ESP8266 de tamaño mucho más reducido y que pueden ejecutar el mismo programa que se ha desarrollado en este proyecto. Por poner un ejemplo, el microcontrolador ESP8266 modelo ESP-02 tiene unas dimensiones de 14.2mm x 14.2mm. Con este tamaño tan reducido se podría pensar en el hecho de incluir un nuevo prototipo de dispositivo de seguridad dentro de una pulsera que sea cómoda y discreta.

Otro aspecto candidato a mejorar es la reducción del consumo del dispositivo. Aunque en la actual propuesta ya se contemplan métodos para que éste sea menor, si se pudiera reducir aún más aumentaría la duración de cada pila o recarga.

En relación a la definición de variables tales como el tiempo máximo de espera para conectarse a una red, el tiempo que debe permanecer el dispositivo en modo *deep sleep*, el tiempo que debe permanecer activo el servidor de *buzzer*, etc., podría implementarse una forma de que sea el propio usuario quien establezca estos valores.

Con respecto a la aplicación para móvil, podría mejorarse o rediseñarse la interfaz para que sea algo más bonita y también podría implementarse alguna mejora en la funcionalidad del mapa. Por ejemplo, en lugar de mostrar únicamente la última ubicación almacenada en el servidor se podría mostrar las últimas cinco o las últimas diez posiciones. De esta forma los usuarios de la aplicación podrían hacerse una idea del recorrido que está siguiendo el menor y a partir de ahí, hacerse una idea de cuál puede ser el destino de ese recorrido.

8.2. Valoración personal

Durante la realización del presente trabajo he tenido que enfrentarme a innumerables dudas que me han hecho investigar en muy diversas fuentes durante horas y horas hasta encontrar una solución adecuada al problema al que me enfrentaba. Esta tarea me ha hecho aprender a buscar mejor y en mejores fuentes, y es algo que voy a necesitar durante toda mi vida académica y profesional.

Por otra parte, el tema principal del trabajo ha sido una muy buena elección por la finalidad que tiene y me parece que puede llegar a ser una solución genial y accesible para todo el mundo. Me gustaría añadir que aunque durante la redacción de la presente memoria siempre se ha considerado que el portador del dispositivo de seguridad va a ser un menor de edad, este dispositivo podría utilizarse también con personas mayores afectadas por algún tipo de demencia senil.

Teniendo en cuenta todo lo dicho anteriormente se puede decir que la experiencia vivida trabajando en este proyecto ha sido muy gratificante tanto por lo aprendido como por los resultados obtenidos.

Referencias

Referencias - Introducción

[1] Artículo periodístico: *Un 25% de los españoles pierde a sus hijos durante las vacaciones*, fuente: *Periódico El Mundo*, autor: *N/D*. Disponible en recurso online: http://www.elmundo.es/viajes/el-baul/2018/07/18/5b4dfce9e2704e1d5f8b459f.html [Último Acceso 25 de julio de 2018].

[2] Artículo periodístico: *Los lugares donde es más frecuente perder a tu hijo*, fuente*: Periódico ABC*, autor: *N/D*. Disponible en recurso online: https://www.abc.es/familia-padres-hijos/20150619/abci-perder-201506181233.html [Último Acceso 26 de julio de 2018].

[3] Evans, D.; *Internet of Things, La próxima evolución de Internet lo está cambiando todo, abril 2011*. Publicado en Informe Técnico, fuente: *Cisco IBSG®*. Disponible recurso online: https://www.cisco.com/c/dam/global/es_es/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf [Último Acceso 13 de junio de 2018].

Referencias – Estado del Arte

[4] Página web oficial *Cerberus App, Solución antirrobo para Android*. Disponible en: https://www.cerberusapp.com/persona [Último Acceso 9 de agosto de 2018].

[5] Página web oficial *FiLIP*. Disponible en: http://www.myfilip.com/es/ [Último Acceso 9 de agosto de 2018].

[6] Página web oficial Localizador Sherlog, Localización de niños por GPS. Disponible en: https://www.localizadorsherlog.es/productos/localizacion-de-ninos/ [Último Acceso 10 de agosto de 2018].

[7] Página web oficial My Buddy Tag. Disponible en: https://mybuddytag.com/ [Último Acceso 10 de agosto de 2010].

Referencias – Diseño y desarrollo

[8] Herramienta de diseño *Draw.io*. Disponible en: https://www.draw.io/ [Último Acceso 1 de septiembre de 2018].

[9] De la Casa, J.*; Wemos D1 mini consumo en modo Deep Sleep, 30 de agosto de 2016*. Recurso disponible en: http://jdelacasa.es/2016/08/30/wemos-d1-mini-consumo-en-modo-deepsleep/ [Último Acceso 2 de septiembre de 2018].

[10] Bakke, O.; *Reducing WiFi power consumption on ESP8266, part 3*, mayo de 2017. Disponible en: https://www.bakke.online/index.php/2017/05/22/reducing-wifi-power-consumption-on-esp8266-part-3/ [Último Acceso 20 de agosto de 2018].

[11] Librería WifiLocation, *Google GeoLocation API wrapper for Arduino MKR1000, ESP8266 and ESP32*, fuente: *GitHub*, autor: *Germán Martín*. Disponible en: https://github.com/gmag11/WifiLocation [Último Acceso 7 de junio de 2018].

Referencias – Presupuesto

[12] Presupuesto *Lenovo Yoga 700*, fuente: página oficial de Amazon. Disponible en: https://www.amazon.es/Lenovo-Yoga-700-14ISK-Portátil-i7-6500U/dp/B0182CXXWY/ref=sr_1_1?ie=UTF8&qid=1511116492&sr=8-1&keywords=lenovo+yoga+700 [Último Acceso 3 de septiembre de 2018]

[13] Presupuesto *High Quality Passive Buzzer Module for Arduino*, fuente: página oficial de AliExpress. Disponible en: https://es.aliexpress.com/store/product/1pcs-lot-High-Quality-Passive-Buzzer-Module-for-Arduino/ [Último Acceso 3 de septiembre de 2018].

[14] Presupuesto *WEMOS D1 Mini Pro16 m Bytes conector de antena externa NodeMCU ESP8266 ESP-8266EX CP2104 desarrollo WIFI Micro USB*, fuente: página oficial de AliExpress. Disponible en: https://es.aliexpress.com/store/product/Wemos-D1-Mini-Pro-NodeMcu-16M-Bytes-Antenna-Connector-Wifi-Internet-of-Things-Development-Board-Module/ [Último Acceso 3 de septiembre de 2018].

[15] Presupuesto protoboard y conectores *400 puntos solderless Prototype Junta cubierta electrónica prueba + 65 unids Breadboard línea Alambres cable,* fuente: página oficial de AliExpress. Disponible en: https://es.aliexpress.com/store/product/Newest-Top-Selling-400-Points-Solderless-Prototype-Board-Electronic-Deck-Test-Board-65pcs-Breadboard-Tie-Line/ [Último Acceso 3 de septiembre de 2018].

[16] Presupuesto y características *Smartphone RedMi Note 5*, fuente: página oficial tienda online de Xiaomi®. Disponible en: https://buy.mi.com/es/buy/product/redmi-note-5 [Último Acceso 3 de septiembre de 2018].

[17] Presupuesto *Smartphone Google Nexus 5*, fuente: página oficial de Amazon. Disponible en: https://www.amazon.es/Google-Nexus-Smartphone-pantalla-Quad-Core/ [Último Acceso 3 de septiembre de 2018].

Anexos

Código del microcontrolador en Arduino

```
#include <ESP8266WiFi.h>
#include <WifiLocation.h>

//Variables para la conexión a una red conocida.
//const char* ssid = "PabloAndroidAP_1827";
const char* ssid = "MOVISTAR_0525";
const char* password = "kHWdHsf4wUsPDuh89aE7"; //Clave correcta
//const char* password = "123abc"; //Clave inventada

//Variables para controlar el tiempo en los intentos de conexión a
alguna red
unsigned long esperaConexionMAX = 10 * 1000; //10 segundos
unsigned long tiempoInicioConexion;

//Variables para configurar la dirección IP del microcontrolador
IPAddress serverIP, gateIP, subnetIP;

//Variables para la función buscarRedAbierta
int redesAbiertas[20];//Como maximo 20 redes no encriptadas
const int MAX_SSID_LENGTH = 32;
char redAbierta[MAX_SSID_LENGTH];

//Variables relacionadas con la obtención de coordenadas
const char* googleApiKey = "AIzaSyCfnpSoz3FxaVExVIqjNClEuqxXkJ06Smc";
WifiLocation location(googleApiKey);
location_t loc;

//Variables relacionadas con ThingSpeak
WiFiClient client;
const int channelID = 457068;
String writeAPIKey = "B21SF5URFGHGEUBC"; // write API key for your
ThingSpeak Channel
const char* ThingSpeakServer = "api.thingspeak.com";
long rssi;

//Variables relacionadas con el buzzer
WiFiServer server(9999);
unsigned long tiempoServidorBuzzerMAX = 10*1000;//10 segundos de
servidor activo
unsigned long tiempoServidorActivo;
const int buzzPin = D5;//la patilla donde pongo el buzzer
int frequency = 1000;
boolean sonando = false;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println();

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,password);
//  WiFi.begin(ssid);

  Serial.println("Intentando conectar a " + String(ssid));
  tiempoInicioConexion = millis();
```

```
  while (WiFi.status() != WL_CONNECTED && (millis() -
tiempoInicioConexion) < esperaConexionMAX) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();


  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Conexión establecida con " + String(ssid));
    configurarRed();
    medirPotencia();
    obtenerCoordenadas();
    enviarDatos();
    Serial.println("Datos enviados al servidor");
    buzzerServer();
    Serial.println("Entrando en modo Deep Sleep durante 20 segundos");
    ESP.deepSleep(40e6);
  } else {
    Serial.println("Tiempo de espera superado. No se ha podido
conectar a " + String(ssid));
    Serial.println("Buscando redes alternativas..");
    buscarRedAbierta();
    if (WiFi.status() == WL_CONNECTED) {
      obtenerCoordenadas();
      enviarCoordenadas();
      Serial.println("Entrando en modo Deep Sleep durante 30
segundos");
      ESP.deepSleep(20e6);
    } else {
      Serial.println();
      Serial.println("No se ha podido establecer conexión. Reiniciando
el dispositivo..");
      //ESP.reset() y ESP.restart() no pueden usarse porque la placa
ESP8266 tiene un 'hardware bug'
      ESP.deepSleep(20); //deepSleep funciona en microsegundos 1 s =
1.000.000 us
    }
  }
  Serial.println();
}

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

/*
 * El objetivo de esta función es medir la potencia y almacenarla en
una variable global
*/
void medirPotencia() {
  rssi = WiFi.RSSI();
  Serial.print("Potencia recibida: ");
  Serial.println(rssi);
}

/*
 * El objetivo de esta función es subir al servidor solo los datos de
ubicación (latitud, longitud y precisión)
*/
void enviarCoordenadas() {
```

```
  if (client.connect(ThingSpeakServer, 80)) {
    String body = "field2=";
           body += String(loc.lon, 7);
           body += "&field3=";
           body += String(loc.lat, 7);
           body += "&field4=";
           body += String(loc.accuracy);

    client.println("POST /update HTTP/1.1");
    client.println("Host: api.thingspeak.com");
    client.println("User-Agent: ESP8266 (nothans)/1.0");
    client.println("Connection: close");
    client.println("X-THINGSPEAKAPIKEY: " + writeAPIKey);
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Content-Length: " + String(body.length()));
    client.println("");
    client.print(body);
  }
}//FIN enviarCoordenadas

/*
 * El objetivo de esta función es subir al servidor todos los datos
(rssi, latitud, longitud y precisión)
*/
void enviarDatos() {
  if (client.connect(ThingSpeakServer, 80)) {
    String body = "field1=";
           body += String(rssi);
           body += "&field2=";
           body += String(loc.lon, 7);
           body += "&field3=";
           body += String(loc.lat, 7);
           body += "&field4=";
           body += String(loc.accuracy);

    client.println("POST /update HTTP/1.1");
    client.println("Host: api.thingspeak.com");
    client.println("User-Agent: ESP8266 (nothans)/1.0");
    client.println("Connection: close");
    client.println("X-THINGSPEAKAPIKEY: " + writeAPIKey);
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Content-Length: " + String(body.length()));
    client.println("");
    client.print(body);
  }

  client.stop();
}//FIN enviarDatos

/*
 * Función para obtener las coordenadas de la ubicación del
dispositivo
*/
void obtenerCoordenadas() {
//  location_t loc = location.getGeoFromWiFi();
  loc = location.getGeoFromWiFi();

  Serial.println("Location request data");
  Serial.println(location.getSurroundingWiFiJson());
  Serial.println("Latitude: " + String(loc.lat, 7));
  Serial.println("Longitude: " + String(loc.lon, 7));
```

```
  Serial.println("Accuracy: " + String(loc.accuracy));
}//FIN obtenerCoordenadas

/*
 * Esta función modifica la dirección IP recibida por el router. Es
necesario establecer gateway y subnetMask porque la función
 * WiFi.config() no funciona pasando solamente la dirección IP y hay
que enviar esos tres parámetros.
*/
void configurarRed() {
//  Serial.print("La dirección IP actual es: ");
//  Serial.println(WiFi.localIP());
  Serial.println("Configurando direccionamiento IP de la red..");
  serverIP = WiFi.localIP();
  serverIP[3] = 123;
  gateIP = WiFi.gatewayIP();
  subnetIP = WiFi.subnetMask();

  WiFi.config(serverIP, gateIP, subnetIP);
  delay(50);
//  Serial.print("Ahora la dirección IP es: ");
//  Serial.println(WiFi.localIP());
}//FIN configurarRed

/*
 * Esta función escanea todas las redes detectadas, almacena las que
no tienen cifrado y las compara según la potencia recibida
 * para quedarse con la que tenga una mayor potencia. Después se
intenta conectar a esa red.
 *
 * Variables globales:
 * tiempoInicioConexion, esperaConexionMAX, redesAbiertas[],
MAX_SSID_LENGTH, redAbierta[]
*/
void buscarRedAbierta() {
  delay(50);
  int n = WiFi.scanNetworks();
  delay(50);
  int aux1 = 0; //Indice para redesAbiertas[]
  int numRedAbierta;

  if (n == 0)
    Serial.println("No se han detectado redes");
  else
  {
    Serial.print(n);
    Serial.println(" redes encontradas");
    Serial.println("\tSSID\t\tRSSI\t\tSeguridad");
    for (int i = 0; i < n; ++i) {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(":\t");
      if(WiFi.SSID(i).length() > 13){
        Serial.print(WiFi.SSID(i).substring(0,10));
        Serial.print("...\t");
      } else {
        Serial.print(WiFi.SSID(i));
        Serial.print("\t");
      }
      Serial.print(WiFi.RSSI(i));
      Serial.print("dBm\t\t");
```

```
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?"Red
abierta":"Red cifrada");
      delay(10);

      if(WiFi.encryptionType(i) == ENC_TYPE_NONE) {
        redesAbiertas[aux1] = i;
        aux1++;
      }
    }

    if(aux1 == 0){
      Serial.println("No se ha encontrado ninguna red wifi abierta");
    } else if(aux1 == 1){
      Serial.println("Intentando conectar a " +
String(WiFi.SSID(redesAbiertas[0])));
      numRedAbierta = redesAbiertas[0];
    } else {
      int redMayorPotencia = redesAbiertas[0];
      for (int j=1; j<aux1; j++) {
        if(WiFi.RSSI(redesAbiertas[j]) > WiFi.RSSI(redMayorPotencia))
{
          redMayorPotencia = redesAbiertas[j];
        }
      }
      numRedAbierta = redMayorPotencia;
      Serial.println("Intentando conectar a " +
String(WiFi.SSID(numRedAbierta)));
    }
    strncpy(redAbierta, WiFi.SSID(numRedAbierta).c_str(),
MAX_SSID_LENGTH);
  }
  if(aux1>0){
    tiempoInicioConexion = millis();
    WiFi.begin(redAbierta);
    while (WiFi.status() != WL_CONNECTED && (millis() -
tiempoInicioConexion) < esperaConexionMAX) {
      delay(500);
      Serial.print(".");
    }
    Serial.println();

    if(WiFi.status() == WL_CONNECTED){
      Serial.print("Conexión establecida con ");
      Serial.println(redAbierta);
    } else
      Serial.println("Tiempo de espera superado. No se ha podido
conectar a la red alternativa.");
  }
}// FIN buscarRedAbierta()

/*
 * En esta función se inicia el servidor para hacer funcionar el
buzzer. Está activo durante 10 segundos. Si durante ese tiempo
 * recibe un cliente, el contador se reinicia. Si pasan esos 10
segundos y no recibe ningún cliente, se cierra.
*/
void buzzerServer() {
  pinMode(buzzPin, OUTPUT);

  boolean sonando = false;
  String leido = "";
```

```
  server.begin();
  delay(50);

  Serial.print("Servidor buzzer iniciado con dirección IP: ");
  Serial.println(WiFi.localIP());
  Serial.println("El buzzer estara disponible durante 10 segundos");
  tiempoServidorActivo = millis();
  while(millis() - tiempoServidorActivo < tiempoServidorBuzzerMAX){
    WiFiClient clientBuzzer = server.available();
    if(clientBuzzer){
      tiempoServidorActivo = millis();
      Serial.println("Reiniciado el contador a 10 segundos");

      Serial.print("Cliente conectado. ");

      if(clientBuzzer.connected()) {
        delay(50);
        while(clientBuzzer.available()){
          leido = clientBuzzer.readStringUntil('\n'); //el mensaje
enviado desde la app es "buzzer on\n"
        }

        Serial.print("Mensaje recibido: ");
        Serial.println(leido);
      }
      clientBuzzer.print("Mensaje recibido");
      leido.toLowerCase();
      if(leido.indexOf("on") != -1) {
//        Serial.print("Ha entrado a encender/apagar el buzzer");
        if (!sonando) {
          tone(buzzPin, frequency);
          sonando = true;
          clientBuzzer.print(", buzzer encendido");
          Serial.print("Buzzer encendido");
        }
        else {
          noTone(buzzPin);
          clientBuzzer.print(", buzzer apagado");
          Serial.print("Buzzer apagado");
          sonando = false;
        }

      }

      delay(1);
      clientBuzzer.stop();
      Serial.println("Datos enviados al cliente. Cerrando conexión.");
    }
    delay(100);
  }
  Serial.println("Se ha agotado el tiempo de espera del servidor");
}
```

Código plugin ThingSpeak

```
<!DOCTYPE html>
<html>
 <head>
  <title>ESP8266, Thingspeak and Google Map</title>
  <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
  <meta charset="utf-8">
  <style>
   #map {
    height: 100%;
   }
   html, body {
    height: 100%;
    margin: 0;
    padding: 0;
   }
  </style>
  <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDDrnW7W1uYvjE2DbPnaU
0a5lRvxG_Bp8M"></script>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script>
                var map;
                var x = 0;
                var y = 0;
                var mapaMostrado = false;
                var contador = 0;
                var marker;
```

```
function loadmaps(){

$.getJSON("https://api.thingspeak.com/channels/457068/fields/3/last.json?api_key=
W6B6AAHXE053N3BX", function(result){
    var m = result;
                x = Number(m.field3);
                });
$.getJSON("https://api.thingspeak.com/channels/457068/fields/2/last.json?api_key=
W6B6AAHXE053N3BX", function(result){
    var m = result;
    y = Number(m.field2);
                }).done(function() {
                        marker.setPosition(new google.maps.LatLng(x, y));
                        map.panTo(new google.maps.LatLng(x, y));
                });
}
        window.setInterval(function(){
                loadmaps();
}, 3*1000);
        function initialize() {
                var mapOptions = {
                 zoom: 15,
                 //center: {lat: latitud, lng: longitud}
                 center: {lat: x, lng: y}
                };
                map = new google.maps.Map(document.getElementById('map'),
mapOptions);
                marker = new google.maps.Marker({
                 position: {lat: x, lng: y},
                 map: map
```

```
                    });
                    var infowindow = new google.maps.InfoWindow({
                      content: '<p>Marker Location:' + marker.getPosition() + '</p>'
                    });
                    google.maps.event.addListener(marker, 'click', function() {
                      infowindow.open(map, marker);
                    });
                }
                google.maps.event.addDomListener(window, 'load', initialize);
    </script>
  </head>
  <body>
    <div id="map"></div>
  </body>
</html>
```

Código de la aplicación Java en Android

AndroidManifest-xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.pablovillalba.seguridadversion1">

    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required
to use
         Google Maps Android API v2, but you must specify either
coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!--
             The API key for Google Maps-based APIs is defined as a
string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key
used to sign the APK.
             You need a different API key for each encryption key,
including the release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets
in src/debug/ and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps"></activity>
    </application>

</manifest>
```

MainActivity.java

```
package com.example.pablovillalba.seguridadversion1;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.app.NotificationCompat;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.Inet4Address;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.Socket;
import java.net.SocketException;
import java.net.URL;
import java.net.UnknownHostException;
import java.util.Enumeration;
import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity {

    TextView estado, mensaje, mostrarRSSI;
    Button buzzer, botonRSSI, mapa;
    Boolean buzzerON = false;
    String serverIP;
    int rssi;
    double lat, lng;
    int prec;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        estado = (TextView)findViewById(R.id.estadoBuzzer);
        mensaje = (TextView)findViewById(R.id.mensajeServidor);
        mostrarRSSI = (TextView)findViewById(R.id.mostrarRssi);

        ajustarDireccionIPServidor();

        buzzer = (Button)findViewById(R.id.button);
```

```
        buzzer.setOnClickListener(buzzerOnClickListenner);

        botonRSSI = (Button)findViewById(R.id.button2);
        botonRSSI.setOnClickListener(RSSIOnClickListenner);

        mapa = (Button)findViewById(R.id.button3);
        mapa.setOnClickListener(mapaOnClickListenner);

        estado.setText("El buzzer está apagado");

        final Handler handler = new Handler();
        Timer timer = new Timer();

        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable() {
                    public void run() {
                        try {
                            ConectarURL conexion = new ConectarURL();
                            conexion.execute();
                        } catch (Exception e) {
                            Log.e("error", e.getMessage());
                        }
                    }
                });
            }
        };


        timer.schedule(task, 0, 10000);
    }

    View.OnClickListener mapaOnClickListenner = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ConectarURL obtenerCoordenadas = new ConectarURL();
            obtenerCoordenadas.execute();

            Intent intent = new Intent(MainActivity.this,
MapsActivity.class);

            intent.putExtra("latitud", lat);
            intent.putExtra("longitud", lng);
            intent.putExtra("precision", prec);

            startActivity(intent);
        }
    };

    View.OnClickListener RSSIOnClickListenner = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {

            ConectarURL actualizarRSSI = new ConectarURL();
            actualizarRSSI.execute();

            notificationCall1();
        }
```

```
    };

    View.OnClickListener buzzerOnClickListenner = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            notificationCall2();

            //el rango 192.168.43.X es el que asignan las redes Wi-Fi
creadas por móviles Android
            MyClientTask myClientTask = new MyClientTask(serverIP,
9999);
            myClientTask.execute();
            mensaje.setText("");
            String aux = "";
            if(buzzerON) {
                buzzerON = false;
                aux = "El buzzer se va a apagar";
            }
            else {
                buzzerON = true;
                aux = "El buzzer se va a encender";
            }
            estado.setText(aux);
        }
    };

    public class MyClientTask extends AsyncTask<Void, Void, Void>{

        String dstAddress;
        int dstPort;
        String response = "";

        MyClientTask(String a, int b){
            dstAddress = a;
            dstPort = b;
        }

        @Override
        protected Void doInBackground(Void... params) {

            Socket socket;

            try {
                socket = new Socket(dstAddress, dstPort);

                DataOutputStream dataOut = new
DataOutputStream(socket.getOutputStream());
                dataOut.writeBytes("buzzer on");


                BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                response = input.readLine();
            }catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }
```

```
        @Override
        protected void onPostExecute(Void result) {
            mensaje.setText(response);
            super.onPostExecute(result);
        }
    }

    public class ConectarURL extends AsyncTask<Void, Void, Void> {

        String sUrl =
"https://api.thingspeak.com/channels/457068/feeds.json?api_key=W6B6AAH
XE053N3BX&results=1";
        URL url;
        HttpURLConnection urlConnection;
        JSONObject jsonObject1, jsonObject2;

        ConectarURL() {

        }

        @Override
        protected Void doInBackground(Void... params) {

            try {
                url = new URL(sUrl);
                urlConnection = (HttpURLConnection)
url.openConnection();
                BufferedReader in = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                StringBuilder builder = new StringBuilder();
                String line;
                while ((line = in.readLine()) != null) {
                    builder.append(line);
                }

                jsonObject1 = new JSONObject(builder.toString());

            } catch (IOException e) {
                e.printStackTrace();
            } catch (JSONException e) {
                e.printStackTrace();
            } finally {
                urlConnection.disconnect();
            }

            return null;
        }

        @Override
        protected void onPostExecute(Void result) {
            try {
                if(jsonObject1!=null) {
                    jsonObject2 = new
JSONObject(jsonObject1.getString("feeds").substring(1,
jsonObject1.getString("feeds").length() - 1));
                    Toast toast1 =
Toast.makeText(getApplicationContext(), "Datos actualizados",
Toast.LENGTH_SHORT);
```

```
                    rssi =
Integer.parseInt(jsonObject2.getString("field1"));

mostrarRSSI.setText(jsonObject2.getString("field1"));
                    toast1.show();

                    lat =
Double.parseDouble(jsonObject2.getString("field3"));
                    lng =
Double.parseDouble(jsonObject2.getString("field2"));
                    prec =
Integer.parseInt(jsonObject2.getString("field4"));
                }

            } catch (JSONException e) {
                e.printStackTrace();
            }
            super.onPostExecute(result);
        }
    }

    public void ajustarDireccionIPServidor() {
        NetworkInterface interfaz = null;
        boolean encontrada = false;
        String aux = "";

        try {
            for (Enumeration<NetworkInterface> en =
NetworkInterface.getNetworkInterfaces(); en.hasMoreElements();) {
                NetworkInterface intf = en.nextElement();
                if("softap0".equals(intf.getName())){
                    interfaz = intf;
                    encontrada = true;
                }
            }
            if(encontrada) {
                for (Enumeration<InetAddress> enumIpAddr =
interfaz.getInetAddresses(); enumIpAddr.hasMoreElements();) {
                    InetAddress inetAddress =
enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress() &&
inetAddress instanceof Inet4Address) {
                        serverIP = inetAddress.getHostAddress();
                        String[] splitted = serverIP.split("\\.");
                        splitted[3] = "123";
                        serverIP =
splitted[0]+'.'+splitted[1]+'.'+splitted[2]+'.'+splitted[3];
                        aux += "Servidor del buzzer en: " + serverIP;
                    }
                }
            } else {
                interfaz = NetworkInterface.getByName("wlan0");
                for (Enumeration<InetAddress> enumIpAddr =
interfaz.getInetAddresses(); enumIpAddr.hasMoreElements();) {
                    InetAddress inetAddress =
enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress() &&
inetAddress instanceof Inet4Address) {
                        serverIP = inetAddress.getHostAddress();
                        String[] splitted = serverIP.split("\\.");
                        splitted[3] = "123";
```

```
                        serverIP =
splitted[0]+'.'+splitted[1]+'.'+splitted[2]+'.'+splitted[3];
                        aux += "Servidor del buzzer en: " + serverIP;
                    }
                }
            }
            mensaje.setText(aux);
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }


    public void notificationCall1() {

        Intent notificationIntent = new Intent(this,
MainActivity.class);

        notificationIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent pendingIntent = PendingIntent.getActivity(this,
0, notificationIntent, 0);


        NotificationCompat.Builder notificationBuilder =
(NotificationCompat.Builder) new NotificationCompat.Builder(this)
                .setDefaults(NotificationCompat.DEFAULT_ALL)
                .setSmallIcon(R.mipmap.ic_launcher)

.setContentTitle(this.getResources().getString(R.string.app_name))
                .setContentText("¡Se está alejando!")
                .setContentIntent(pendingIntent)
                .setAutoCancel(true);//este hace que una vez que se
toque la notificacion, esta desaparezca

        NotificationManager notificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(1, notificationBuilder.build());

    }

    public void notificationCall2() {

        Intent notificationIntent = new Intent(this,
MainActivity.class);

        notificationIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent pendingIntent = PendingIntent.getActivity(this,
0, notificationIntent, 0);


        NotificationCompat.Builder notificationBuilder =
(NotificationCompat.Builder) new NotificationCompat.Builder(this)
                .setDefaults(NotificationCompat.DEFAULT_ALL)
                .setSmallIcon(R.mipmap.ic_launcher)

.setContentTitle(this.getResources().getString(R.string.app_name))
                .setContentText("¡Se ha perdido la conexión!")
                .setContentIntent(pendingIntent)
                .setAutoCancel(true);//este hace que una vez que se
```

```
toque la notificacion, esta desaparezca

        NotificationManager notificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(1, notificationBuilder.build());

    }
}
```

activity_mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.pablovillalba.seguridadversion1.MainActivit
y">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Buzzer"
        android:textAllCaps="false"
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="8dp"
        app:layout_constraintVertical_bias="0.744"
        app:layout_constraintHorizontal_bias="0.757" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="8dp"
        android:text="Solicitud enviada al dispositivo:"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.087"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintHorizontal_bias="0.06" />

    <TextView
        android:id="@+id/estadoBuzzer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.204"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.192" />
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="141dp"
    android:text="Notificaciones del dispositivo:"
    app:layout_constraintHorizontal_bias="0.098"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    android:layout_marginBottom="8dp"
    app:layout_constraintVertical_bias="0.017" />

<TextView
    android:id="@+id/mensajeServidor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.234"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.42" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="RSSI(dBm): "
    app:layout_constraintTop_toTopOf="parent"
    android:layout_marginTop="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    android:layout_marginBottom="8dp"
    android:layout_marginRight="8dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginLeft="8dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintHorizontal_bias="0.061"
    app:layout_constraintVertical_bias="0.55" />

<TextView
    android:id="@+id/mostrarRssi"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.683"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.55" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:text="Actualizar"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.235"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.744" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Abrir mapa"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="8dp"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="8dp"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintVertical_bias="0.93" />

</android.support.constraint.ConstraintLayout>
```

MapsActivity.java

```
package com.example.pablovillalba.seguridadversion1;

import android.content.Intent;
import android.graphics.Color;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.CircleOptions;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback {

    double lat, lng;
    int prec;
    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map
is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

        Intent intent1 = getIntent();
        lat = intent1.getDoubleExtra("latitud", 0);
        lng = intent1.getDoubleExtra("longitud", 0);
        prec = intent1.getIntExtra("precision", 20);
    }


    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        LatLng ubicacion = new LatLng(lat,lng);
        CircleOptions circulo = new CircleOptions()
                .center(ubicacion)
                .radius(prec)
                .strokeColor(Color.parseColor("#0D47A1"))
                .strokeWidth(4)
                .fillColor(Color.argb(32, 33, 150, 243));

        mMap.addCircle(circulo);
        mMap.addMarker(new
MarkerOptions().position(ubicacion).title("Últimas coordenadas
recibidas"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(ubicacion));

        float zoomLevel = 18;
```

```
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(ubicacion,
zoomLevel));
    }
}
```

google_maps_api.xml

```
<resources>
    <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">
        AIzaSyAWmzwen_-pFNb4kDec8lixCVgjOKw_mPQ
    </string>
</resources>
```