UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

Plataforma inalámbrica para la web física

GRADO EN INGENIERÍA DE
TECNOLOGÍAS DE TELECOMUNICACIÓN

JOSÉ ANTONIO YÉBENES GÁLVEZ
MÁLAGA, 2018

Universidad de Málaga
Escuela Técnica Superior de Ingeniería de Telecomunicación

Plataforma inalámbrica para la web física

Autor
José Antonio Yébenes Gálvez

Tutores
Ignacio Herrero Reder y José Manuel Cano García

Departamento: Tecnología Electrónica (DTE)

Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación

Palabras clave: IOT, web física, web, red, sensores, inalámbrica, internet, servidor

Resumen:

En este trabajo de fin de grado se implementa una red inalámbrica en la banda de 868 MHz utilizando microcontroladores CC1350, que adicionalmente pueden difundir balizas compatibles con Bluetooth Low Energy. Esto permite la interacción con dispositivos móviles usando tecnología web, un paradigma que se conoce como web física.

Este proyecto no sólo ha implicado el desarrollo de la red inalámbrica, sino que también se ha implementado un servicio web desde donde es posible monitorizar y gestionar la red. El servicio web consta de un Back-end creado sobre NodeJS y un Front-end para el que ha utilizado AngularJS.

Con el objetivo de demostrar el funcionamiento de la plataforma se ha realizado una aplicación con dos nodos diferentes: uno que envía paquetes de la web física y otro que simularía el comportamiento de un aparcamiento de bicicletas públicas.

Universidad de Málaga
Escuela Técnica Superior de Ingeniería de Telecomunicación

Wireless framework for physical web

Author
José Antonio Yébenes Gálvez

Supervisors
Ignacio Herrero Reder y José Manuel Cano García

Department: Tecnología Electrónica (DTE)

Degree: Grado en Ingeniería de Tecnologías de Telecomunicación

Keywords: IOT, physical web, web, network, sensors, wireless, physical web, internet

Abstract:

In this final degree project, several CC1350 microcontroller are used to create a wireless network operating in the 868 MHz band. These microcontrollers can additionally broadcast Bluetooth Low Energy advertisement packets, thus allowing the interaction with mobile hand-held devices usign web technology. This communication paradigm is known as the physical web.

In addition to the deployment of the wireless network, a web service has been built in order to monitor and manage it. The web service consists of a back-end module based on NodeJS and a front-end application developed with AngularjS.

In order to verifty the performance of the platform, a test application with two different nodes has been created. One of the nodes sends packets from the physical web and the other one is responsible for simulating a public bicycle parking.

A mi familia,
por facilitarme cumplir mis objetivos.
A Lucía,
por ayudarme a no perder el rumbo.

Acrónimos

API	*Application Programming Interface*
APL	*Application*
BLE	*Bluetooth Low Energy*
CCS	*Code Composer Studio*
CM0	*Cortex M0*
CM3	*Cortex M3*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
DFE	*Directed Frame Exchange*
ETSIT	Escuela Técnica Superior de Ingeniería de Telecomunicación
H2M	*Human-to-Machine*
HP	*Hewlett-Packard*
ICall	*Indirect Call Framework*
IEEE	*Institute of Electrical and Electronics Engineers*
IOT	Internet de las cosas o *Internet Of Things*
ISM	*Industrial, Scientific and Medical*
LBT	*Listen Before Talk*
M2M	*Machine-to-Machine*
MAC	*Medium Access Control*
MPU	Microprocesador

MCU	Microcontrolador
MT	*Managment and Test*
MVC	Modelo-Vista-Controllador
NPI	*Network Processor Interface*
NWK	*Network*
PAN	red de área personal o *Personal Area Network*
PHY	*Physical*
QR	*Quick Response*
REST	*Representational State Transfer*
RF	Radiofrecuencia
RTOS	Sistema operativo en tiempo real o *Real-Time Operating System*
TFG	Trabajo Fin de grado
UART	*Universal Asynchronous Receiver/Trasnsmitter*
uBle	*Micro BLE Stack*
UMA	Universidad de Málaga
URL	*Uniform Resource Locator*
Wavenis-OSA	*Wavenis Open Standard Alliance*
WSN	*Wireless Sensor Networks*

Índice

Acrónimos XI

I Introducción 1

1 Objetivos 3
- 1.1 Aplicación de ejemplo . . . 3
  - 1.1.1 Nodo genérico . . . 4
  - 1.1.2 Nodo aparcamiento . . . 4

2 Descripción general 5
- 2.1 Esquema General . . . 5
- 2.2 Nodo . . . 6
- 2.3 Concentrador . . . 6
- 2.4 Servidor . . . 6

3 Estado del arte 7
- 3.1 Web física . . . 7
- 3.2 Redes inalámbricas e internet de las cosas . . . 9
  - 3.2.1 Visión general de las soluciones existentes . . . 9

II Desarrollo del proyecto 13

4 Red inalámbrica 15
- 4.1 Introducción . . . 16
- 4.2 Nodo . . . 16
  - 4.2.1 Introducción . . . 16
  - 4.2.2 Arquitectura hardware . . . 16
  - 4.2.3 Arquitectura de la aplicación . . . 17
  - 4.2.4 TI-RTOS . . . 18
  - 4.2.5 Función de inicio . . . 19

4.2.6 Tarea principal . . . 19
4.2.7 Funciones de gestión del Stack . . . 20
4.2.8 Funciones de acceso a la web física . . . 25
4.2.9 Mensajes OTA . . . 25
4.3 Concentrador . . . 26
4.3.1 Introducción . . . 26
4.3.2 Diagrama de bloques y modelo de la interfaz . . . 26
4.3.3 Descripción del directorio del proyecto . . . 27
4.3.4 Funcionamiento . . . 28
4.3.5 Hilo del concentrador . . . 29
4.3.6 Protocol Buffers . . . 30

5 Servidor **31**
5.1 Introducción . . . 31
5.2 Estructura del directorio . . . 31
5.3 Inicio del servidor . . . 33
5.4 Back-end . . . 34
5.4.1 Mensajes enviados entre el concentrador y el Back-end . . . 34
5.4.2 Mensajes enviados entre el Back-end y Front-end . . . 35
Front-end . . . 36

6 Pruebas **39**
6.1 Prueba en entorno real . . . 39
6.1.1 Introducción . . . 39
6.1.2 Metodología . . . 39
6.1.3 Resultados . . . 40
6.2 Prueba de consumo . . . 41
6.2.1 Introducción . . . 41
6.2.2 Medida del consumo . . . 41
6.2.3 Resultados . . . 42

III Conclusiones y lineas futuras **45**

Conclusiones **47**

Líneas futuras **49**

IV Apéndices **51**

A Estructuras de mensajes OTA **53**

B Herramientas utilizadas **55**

C Manual de uso **57**
C.1 Inicio de la red 57
C.1.1 Iniciar el servidor 57
C.1.2 Encender concentrador 58
C.1.3 Encender nodos 59
C.1.4 Permitir conexiones de los nodos 59
C.2 Administración de los nodos 60
C.2.1 Administración de nodo genérico 61
C.3 Mapa 62

Bibliografía **65**

Índice de figuras

1.1 Aparcamiento de bicicletas públicas de la ciudad de Málaga 4

2.1 Esquema general del proyecto 5

3.1 Logo de la web física 7
3.2 Ejemplos de comunicación cercana 8

4.1 Diagrama de bloques de Simplelink™CC13x0 17
4.2 Diagrama de bloques de la aplicación 18
4.3 Flujo de la tarea principal 19
4.4 Configuración como dispostivo único y como coprocesador 21
4.5 Aplicación ICall - Abstracción del protocolo 22
4.6 Ejemplo de comunicación usando el módulo ICALL 23
4.7 Arquitectura de software a alto nivel de las aplicaciones TI 15.4-Stack 2.1.0 Linux® 26
4.8 Estructura del directorio TI 15.4-Stack 2.1.0 Linux® 28
4.9 Flujo del hilo principal del concentrador 29

5.1 Estructura del directorio del servidor 32
5.2 Diagrama de flujo en el inicio del servidor 33
5.3 Diagrama de flujo en el inicio del servidor 36

6.1 Posición del concentrador y nodo durante la prueba en el punto de máximo alcance 40
6.2 Estadísticas de la red después de la prueba 40
6.3 Programa de LabView para captura rápida 42
6.4 Programa de LabView para captura en promedio 42
6.5 Consumo en mA del nodo a la máxima frecuencia de muestreo 43

A.1 Estructuras de los mensajes del fichero smsgs.h 54

C.1 Web de gestión de instancias en Amazon Web Services 57
C.2 Apariencia de la web cuando no está conectado el concentrador . . . 58

C.3 Apariencia de la web cuando ya está conectado el concentrador 59
C.4 Apariencia de la web cuando hay dos nodos conectados 60
C.5 Panel de administración de los nodos 60
C.6 Panel de administración de un nodo tipo genérico 61
C.7 Panel de administración de un tipo parking de bicicletas 62
C.8 Vista del mapa con las posiciones de los nodos 63

Índice de Tablas

4.1 Bandas permitidas en TI 15.4-Stack y las frecuencias de sus canales . 22
4.2 Formato de la trama *Eddystone-URL* 25

5.1 Identificadores de los mensajes enviados entre el concentador y el servidor 35
5.2 Tipos de mensajes enviados por WebSocket entre el Back-end y el Front-end 36

Parte I

Introducción

Capítulo 1

Objetivos

La web física es un término que describe la forma de interactuar con cualquier objeto usando la web. A partir este enfoque, es posible navegar y controlar objetos en el mundo a través de dispositivos móviles. Esto ofrece a los usuarios la forma de realizar sus tareas diarias utilizando los objetos de su entorno. Para utilizar este enfoque, lo primero es seleccionar la manera con la que el objeto se comunicará con el usuario, como podría ser códigos QR o etiquetas RFID. De los diferentes enfoques que tiene la web física, el que se va a tratar en este proyecto es el basado en proximidad inalámbrica.

Con el modelo que plantea la web física los objetos pueden necesitan un canal por donde enviar y recibir datos y el problema se agrava cuando los objetos están distribuidos y no tienen cerca un punto de acceso a internet o tienen que funcionar con baterías. En estos casos, buscar una solución de comunicación inalámbrica de gran alcance y bajo consumo es prioritario.

Con estos conceptos en mente, el presente trabajo de fin de grado va a abordar el concepto de web física, implementando una red inalámbrica que permita comunicarse con los objetos con el fin de obtener una red versátil.

1.1. Aplicación de ejemplo

Para comprobar el funcionamiento de la red, se realizará una aplicación de ejemplo que podría simular un sistema de gestión de una ciudad inteligente donde habría dos tipos de nodos uno para solo enviar información a los usuarios (nodo genérico) y otro para gestionar un aparcamiento de bicicletas públicas (nodo aparcamiento).

1.1.1. Nodo genérico

El nodo genérico es el nodo más simple de la plataforma, este permite enviar mensajes de la web física que serán definidos desde la web, además este nodo enviará periódicamente mensajes de seguimiento al servidor con información del estado del dispositivo, como por ejemplo temperatura y batería.

1.1.2. Nodo aparcamiento

Este nodo hereda todas las funciones del nodo genérico pero además simulará un uso que podría tener la web física. En este caso simulará un aparcamiento de bicicletas públicas de los que se pueden encontrar por muchas ciudades (véase figura 1.1). El dispositivo simulará los estados de los candados que mantiene a cada bicicleta anclada con un array de unos y ceros, donde el uno significará abierto y el cero significará cerrado.

Figura 1.1: Aparcamiento de bicicletas públicas de la ciudad de Málaga

La idea de funcionamiento sería simple: el usuario se aproxima a las bicicletas, y le llega una notificación de la web física con un enlace a una web donde poder elegir una bicicleta. Al seleccionar una bicicleta se enviaría un mensaje al nodo a través de la red inalámbrica con el comando de abrir el candado de la bicicleta seleccionado.

En este proyecto solamente se ha abordado la parte de poder enviar la información desde la web al nodo, dejando lo demás como línea futura de este Trabajo Fin de grado (TFG).

Capítulo 2

Descripción general

Contenido

2.1 Esquema General 5
2.2 Nodo 6
2.3 Concentrador 6
2.4 Servidor 6

2.1. Esquema General

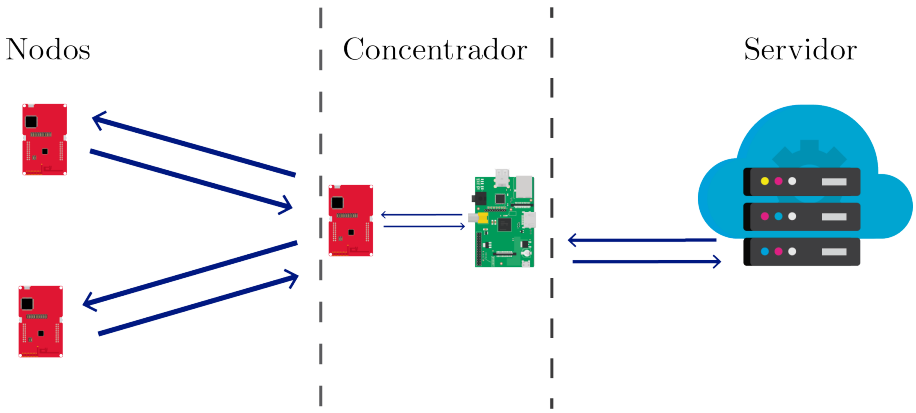


Figura 2.1: Esquema general del proyecto

En la figura 2.1 se observa el diagrama simplificado de este proyecto donde se están representados los diferentes elementos que componen la arquitectura.

Del diagrama se extrae que hay tres bloques distintos: nodo, concentrador y servidor. Cada uno de estos elementos tiene su propia linea de ejecución diferente de los demás. También se observa como están interconectados los diferentes dispositivos usando protocolos diferentes según sea la comunicación. En las siguientes secciones se repasa la función de cada uno de los dispositivos y como se comunican con los demás.

2.2. Nodo

El nodo es el extremo de la red, y es el encargado de comunicarle al usuario la *Uniform Resource Locator* (URL) tal y como define la web física. Este está basado en el microcontrolador CC1350 SimpleLink™de Texas Instruments que permite comunicación en dos bandas de frecuencia diferentes, en nuestro caso 868MHz y 2.4Ghz.

Con el uso de estas dos bandas de frecuencia, el nodo se comunicará con el usuario usando la banda de 2.4GHz y el protocolo bluetooth. Y con el resto de la red usando la banda de 868MHz y el protocolo TI 15.4 Stack.

2.3. Concentrador

El concentrador está compuesto por dos dispositivos diferentes, un dispositivo CC1350 que actúa como coprocesador de una RaspberryPi. Este bloque es el nodo central de la red TI 15.4 Stack, este se encarga de comunicarse con los nodos a 868MHz. Este dispositivo ejecuta un código precompilado, que implementa una capa 802.15.4e/g MAC/PHY y proporciona una interfaz basada en el protocolo *Managment and Test* (MT) que conecta el dispositivo con el host linux, en nuestro caso una Raspberry Pi.

2.4. Servidor

El servidor ofrece una interfaz donde es posible administrar la red y enviar comandos a los sensores. Está compuesto por un servidor NodeJS y una aplicación AngularJS que dan soporte al *Back-end* y *Fron-end*.

Capítulo 3

Estado del arte

Contenido

3.1 Web física **7**
3.2 Redes inalámbricas e internet de las cosas **9**
3.2.1 Visión general de las soluciones existentes 9

3.1. Web física

El primer concepto que hay que conocer para afrontar el proyecto es el de Web Física. La Web Física es un término que describe el proceso de presentar objetos cotidianos en internet[1]. Este enfoque ofrece a los usuarios la posibilidad de gestionar sus tareas diarias en el uso de objetos cotidianos. Los objetos comienzan a ser inteligentes y remotamente controlables. Este modelo permite a los usuarios móviles navegar y controlar los objetos físicos que rodean al dispositivo móvil.[2].

Figura 3.1: Logo de la web física

Podemos mencionar en este contexto a los conocidos códigos *Quick Response* (QR), los códigos QR son un código de barras en dos dimensiones, a menudo utilizados para mapear URL con objetos físicos [3].

Las etiquetas inalámbricas son uno de los enfoques más utilizados para el marcado de objetos físicos. Las etiquetas inalámbricas pueden soportar protocolos como *Bluetooth Low Energy* (BLE) y *WiFi*. Los protocolos mencionados son soportados por la mayoría de los teléfonos móviles modernos.

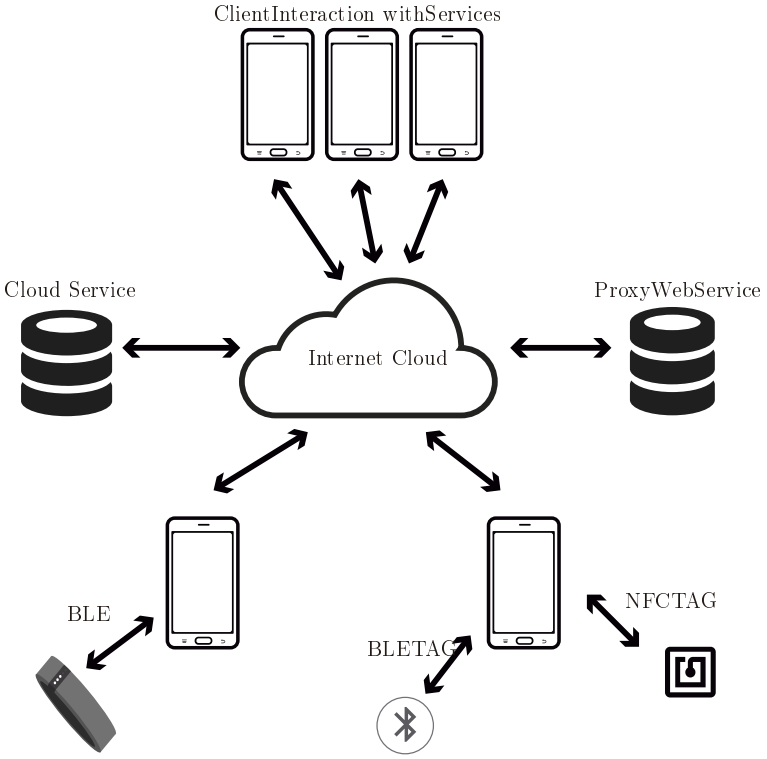


Figura 3.2: Ejemplos de comunicación cercana

En la Web Física, personas, lugares y objetos tienen páginas web que proveen información y mecanismos de interacción. Sin embargo, es la amplitud y la profundidad de la pila que rodea a la web, que hacen de esta una atractiva visión para la evolución del Internet de las cosas o *Internet Of Things* (IOT).[1]

Las páginas web son una fantástica tecnología para interacción *Human-to-Machine* (H2M), pero muchos de los casos de uso del IOT son interacciones *Machine-to-Machine* (M2M). Los formatos de datos usados por `Schema.org` y otros, permiten a los agentes de usuario y servicios en la nube analizar los datos para eventos, organizaciones, personas, lugares, productos y así sucesivamente, acutando sobre ellos de forma interactiva y proactiva. [1]

Uno de los primeros proyectos en fomentar esta idea fue *Hewlett-Packard* (HP)

con *Cooltown*, que usaba balizas infrarrojas para transmitir URLs. Más recientemente, BLE se puede comportar como una baliza de bajo consumo que puede emitir URL en paquetes periódicos (`www.uribeacon.org`). [1]

3.2. Redes inalámbricas e internet de las cosas

El futuro de internet tiene como meta integrar diferentes tecnologías de comunicación, cableadas e inalámbricas, con el objetivo de contribuir sustancialmente a mejorar el concepto de IOT [4]. Aunque hay muchas maneras de describir el IOT, podemos definirlo como una red con objetos interconectados con direcciones únicas, basadas en un protocolo estándar de comunicación.[5]

Los sensores de bajo precio han facilitado la proliferación de *Wireless Sensor Networks* (WSN) en muchos escenarios como monitorización medioambiental, agricultura, salud, y construcciones inteligentes. Las WSN están caracterizadas por una alta heterogeneidad porque están basadas en diferentes soluciones, propietarias y no propietarias. Este gran rango de soluciones está retrasando actualmente desarrollos a gran escala de estas tecnologías a fin de que se obtenga una gran red virtual de sensores que permita integrar todos las redes de sensores existentes.[6]

Las redes de sensores basadas en sistemas cerrados o propietarios son islas conectivamente hablando, con limitadas comunicaciones con el mundo exterior. Por lo general, es necesario usar *gateways* con conocimiento específico de la aplicación para exportar los datos de la WSN a otros dispositivos conectados a Internet. Además, no hay comuniación directa entre diferentes protocolos a menos que se implementen complejas conversiones específicas para la aplicación en los *gateways* o *proxies*.

3.2.1. Visión general de las soluciones existentes

En este apartado presentaremos una rápida visión general de las principales tecnologías usadas para WSN [7]. Analizaremos las soluciones que no están basadas en los protocolos de internet.

ZIGBEE

ZigBee es una tecnología de red inalámbrica desarrollada por la ZigBee Alliance para baja tasa de transmisión de datos y aplicaciones de corto alcance [8]. La pila de protocolos ZigBee está compuesta por 4 principales capas: la capa *Physical* (PHY), la capa *Medium Access Control* (MAC), la capa *Network* (NWK) y la capa

Application (APL). PHY y MAC de ZigBee están definidas por el estándar *Institute of Electrical and Electronics Engineers* (IEEE) 802.15.4, mientras el resto de la pila está definida por la especificación ZigBee.

Esta versión inicial del IEEE 802.15.4, en la que ZigBee está basado, funciona en la bandas de 868 MHz (Europa), 915 MHz (Norteamérica) y 2.4 GHz (global).

Una nueva especificación de ZigBee es RF4CE [9], que tiene una pila de red simplificada para topologías en estrellas solamente, ofreciendo una solución simple para el control remoto de electrónica de consumo.

Z-WAVE

Z-Wave es un protocolo inalámbrico desarrollado por ZenSys y promovida por la Z-Wave Alliance para automatizaciones residenciales y pequeños entornos comerciales. El principal objetivo es permitir transmisiones seguras de cortos mensajes desde una unidad de control hasta uno más nodos en la red [10]. Z-Wave esta organizado de acuerdo a un arquitectura cumpuesta por 5 capas principales: PHY, MAC, transferencia, enrutado y capas de aplicación.

Z-Wave opera principalmente en la banda de 900 MHz (868 MHz en Europa y 908 MHz en Estados Unidos) y 2.4 GHz. Z-Wave permite tasas de transmisión a 9.6 kb/s, 40 kb/s y 200 kb/s.

INSTEON

INSTEON [11] es una solución desarrollada por SmartLabs y promovida por la INSTEON Alliance. Una de las características distintivas de INSTEON es la forma de definir una topología de red compuesta de Radiofrecuencia (RF) y *power line links*. Los dispositivos pueden ser RF, *power-line*, o pueden soportar ambos tipos de comunicación.

INSTEON opera a 904 MHz como frecuencia central, con una tasa de datos bruta de 38.4 kb/s.

Los dispositivos INSTEON son duales, lo que significa que cualquiera de ellos pueden tener el rol de emisor, receptor o repetidor. La comunicación entre ambos dispositivos que no estén en el mismo rango se logra mediante un enfoque «multisalto» que usa los repetidores en un esquema de sincronización temporal.

WAVENIS

Wavenis es un protocolo inalámbrico desarrollado por Coronis System para el control y monitorización de aplicaciones en entornos exigentes, incluida la domótica y la automatización de edificios. Wavenis actualmente está promovida y gestionada por la *Wavenis Open Standard Alliance* (Wavenis-OSA). Está definido por la funcionalidad de las capas física, de enlace y de red [12]. Los servicios de Wavenis pueden ser accedidos desde capas superiores mediante una *Application Programming Interface* (API).

Wavenis opera principalmente en las bandas de 433 MHz, 868 MHz y 915 MHz, que son bandas reservadas para *Industrial, Scientific and Medical* (ISM) en Asia, Europa y Estados Unidos. Algunos productos también operan en la banda de 2.4 GHz. Las tasas de transmisión mínimas y máximas dadas por Wavenis son 4.8 kb/s y 100 kb/s, respectivamente, con 19.2 kb/s como valor típico.

IEEE 802.15.4

El estándar del IEEE 802.15.4, define las especificaciones de la capa física y la subcapa de acceso al medio para conectividad inalámbrica con baja tasa de datos y bajo consumo de energía. Una red 802.15.4 puede simplemente funcionar como una red en estrella con un único salto o, cuando las distancias son grandes, puede funcionar como una red en estrella multisalto. [13]

Los enlaces inalámbricos utilizando 802.15.4 pueden operar en tres bandas ISM. En total 27 canales son utilizables en 802.15.4, donde 16 canales en la banda de 2.4GHz, 10 canales en la banda de 915MHz y 1 canal en la banda de 868MHz.

Parte II

Desarrollo del proyecto

Capítulo 4

Red inalámbrica

Contenido

4.1 Introducción 16

4.2 Nodo 16

4.2.1 Introducción 16

4.2.2 Arquitectura hardware 16

4.2.3 Arquitectura de la aplicación 17

4.2.4 TI-RTOS 18

4.2.5 Función de inicio 19

4.2.6 Tarea principal 19

4.2.7 Funciones de gestión del Stack 20

4.2.8 Funciones de acceso a la web física 25

4.2.9 Mensajes OTA 25

4.3 Concentrador 26

4.3.1 Introducción 26

4.3.2 Diagrama de bloques y modelo de la interfaz 26

4.3.3 Descripción del directorio del proyecto 27

4.3.4 Funcionamiento 28

4.3.5 Hilo del concentrador 29

4.3.6 Protocol Buffers 30

4.1. Introducción

La red inalámbrica es la parte de este TFG que engloba al nodo y al concentrador (véase figura 2.1). En este capítulo se detallan las características y funcionamiento de cada una de las partes.

4.2. Nodo

4.2.1. Introducción

El nodo implementa la aplicación que le permite conectarse a la red creada por el concentrador. El sensor periódicamente envía reportes de datos en intervalos configurados por el concentrador y este responde con mensajes de rastreo.

4.2.2. Arquitectura hardware

El nodo está basado en la placa de desarrollo *SimpleLink™CC1350 wireless microcontroller (MCU) LaunchPad™* que combina una interfaz radio *Bluetooth® low energy* y otra de *Sub-1GHz* en un único chip.

En esta sección se describen los distintos módulos que contiene el microcontrolador CC1350.

ARM Cortex M0 (Núcleo radio)

El núcleo *Cortex M0* (CM0) en el CC1350 es responsable de la interfaz audio, y traduce complejas instrucciones del núcleo *Cortex M3* (CM3) en bits que son enviados a través del enlace radio. Para el protocolo TI 15.4-Stack, el CM0 implementa la capa PHY de la pila de protocolos.

El *firmware* del núcleo de radio no está destinado a ser usado o modificado por la aplicación del desarrollador.

ARM Cortex M3 (Núcleo del sistema)

El núcleo CM3 está diseñado para ejecutar la pila del protocolo inalámbrico desde la capa de enlace hasta la capa de aplicación de usuario. La capa de enlace actúa como interfaz del núcleo de radio cómo un módulo software llamado *RF driver*.

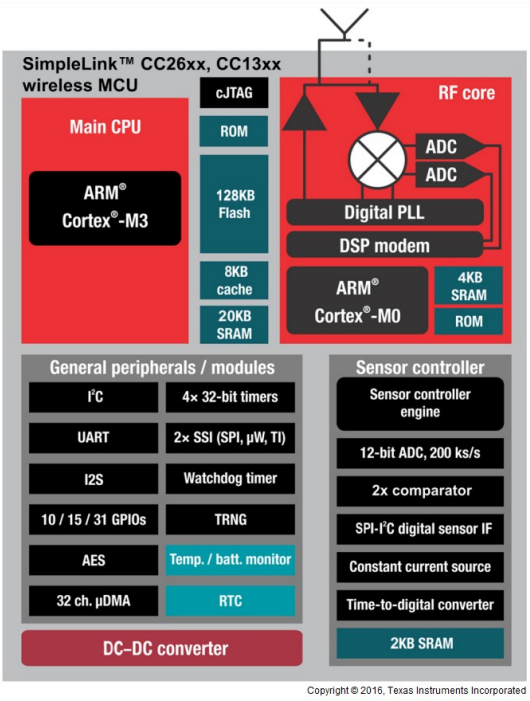


Figura 4.1: Diagrama de bloques de Simplelink™CC13x0

Flash, RAM y periféricos

El CC1350 contiene en el sistema 128KB de memoria flash programable, 20KB de SRAM, y un amplio rango de periféricos. La memora flash se divide en partes que se pueden borrar de 4KB. El CC1350 también contiene 8kB de caché SRAM que puede ser utilizada para extender la capacidad de la RAM o puede funcionar como una caché normal para incrementar el rendimiento de la aplicación. Otros periféricos incluidos son UART, I2C, I2S, AES, TRNG, temperatura y monitor de la batería.

4.2.3. Arquitectura de la aplicación

En la figura 4.2 se muestra el diagrama de bloques de la aplicación del nodo.

Aplicación: Este bloque contiene la lógica específica de la aplicación a desarrollar.

Controlador de lógica de enlace: Implementa varias funciones específicas del IEEE 802.15.4 o Wi-SUN (para una configuración *frequency-hopping*) para formación, conexión y reconexión de la red.

Inicio de TI-RTOS: Inicializa la aplicación.

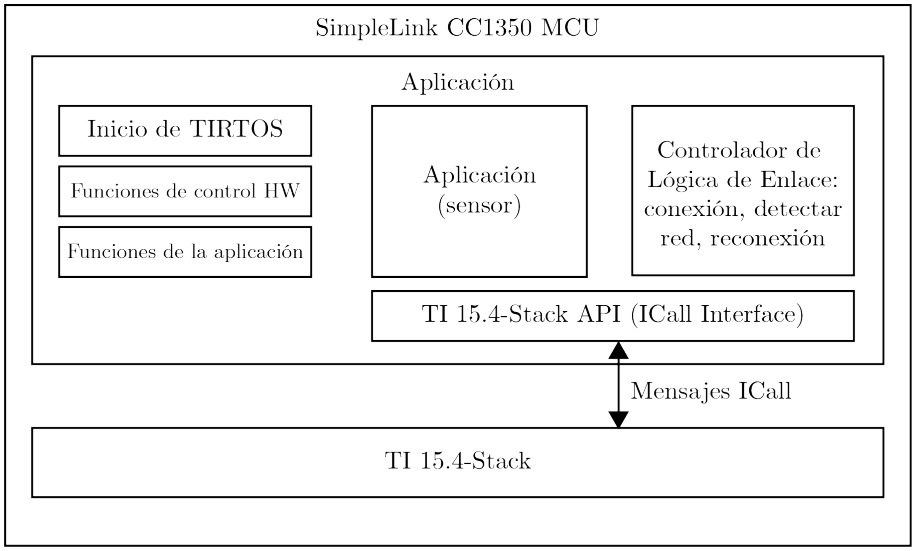


Figura 4.2: Diagrama de bloques de la aplicación

Funciones de control HW Provee varias utilidades para usar LCD, temporizadores, botones y más.

Funciones de la aplicación: Implementa funciones como guardado de datos, y provee una interfaz para gestionar pulsaciones botones o mostrar información esencial en un LCD.

TI 15.4-Stack API (API MAC Module): Este módulo proporciona una interfaz para gestión y los servicios de datos del 802.15.4 stack mediante el módulo *Indirect Call Framework* (ICall).

4.2.4. TI-RTOS

En este proyecto se ha incluido un Sistema operativo en tiempo real o *Real-Time Operating System* (RTOS) que permita la separación en tareas de los distintos procesos. Para facilitar la compatibilidad entre los distintos módulos se ha utilizado el sistema operativo TI-RTOS de Texas Instruments.

TI-RTOS es un entorno operativo para proyectos TI 15.4-Stack en dispositivos CC1350. El kernel TI-RTOS es una versión adaptada del kernel SYS/BIOS y funciona como un sistema operativo con controladores en tiempo real, con prioridades, multitarea y herramientas para la sincronización y planificación.

4.2.5. Función de inicio

La función *main()* dentro del archivo *main.c* es el punto de inicio de la ejecución de la aplicación. En este punto los componentes relaciones con la placa son inicializados. Las tareas se configuran en eta función, inicializando los parámetros necesarios como su prioridad y su tamaño en la pila. En el paso final, las interrupciones se habilitan y el planificador *SYS/BIOS* se inicia llamando a *BIOS_ start()*.

4.2.6. Tarea principal

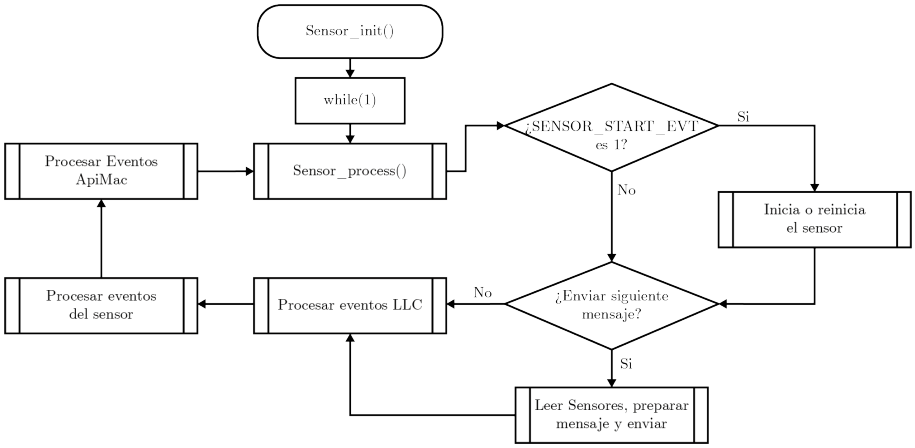


Figura 4.3: Flujo de la tarea principal

Después de que la tarea sea construida y el planificador *SYS/BIOS* se inicie, la función que se le pasa durante la construcción de la tarea es ejecutada cuando la tarea está lista. En esta función se inicializan las funciones de gestión de la energía y el módulo ICall se inicia con la función *ICall_ init()*. La dirección IEEE address (programada por TI) es obtenida desde la memoria flash. La tarea de la aplicación (Aplicación Sensor) es inicializada y ejecutada.

Sensor_ init() establece varios parámetros de configuración, así como:

- Inicializa las estructuras para los datos
- Inicializa el TI 15.4-Stack
- Configura la seguridad y el *Logical Link Controller*
- Registra las funciones de retorno MAC

Como se observa en la figura 4.3 después de ejecutar la función *Sensor_ init()* la tarea entra en un bucle donde se realizan los mismos procedimientos:

1. Conectar o reconectar el nodo a la red.
2. Enviar el mensaje periódico con los datos de sensores.
3. Procesar los eventos LLC que gestionan la capa de enlace.
4. Procesar los eventos específicos del sensor y enviar los mensajes de la web física en los intervalos definidos.
5. Procesar los eventos de la pila ApiMac.

4.2.7. Funciones de gestión del Stack

Introducción

El TI 15.4-Stack es una plataforma completa libre de derechos de autor para desarrollar aplicaciones que requieren una solución inalámbrica con topología en estrella, un extremado bajo consumo, largo alcance, fiable, robusto y seguro.

Este capítulo explica en detalle los diferentes modos de configuración de la red soportadas por el TI 15.4-Stack.

Elección de arquitectura

TI 15.4-Stack se puede utilizar con diferentes arquitecturas como las que se pueden observar en la figura 4.4: [14]

- Una configuración como dispositivo único la podemos observar en la figura 4.4 (izquierda). La aplicación y el protocolo son implementados en el CC13X0 como una solución *single-chip*. Esta configuración es la más simple y común cuando se usa el CC13X0 para nodos de la red. Esta configuración es la arquitectura más económica y de menor consumo.
- Una arquitectura como coprocesador se observa en la figura 4.4 (derecha). La pila de protocolos se ejecuta en el CC1350 mientras la aplicación es ejecuta en un Microprocesador (MPU) o Microcontrolador (MCU). La aplicación se comunica con el CC1350 usando el *Network Processor Interface* (NPI) sobre una comunicación serie *Universal Asynchronous Receiver/Trasnsmitter* (UART). Esta configuración es usada en aplicaciones que añadan comunicación inalámbrica de rango alcance o en un ordenador sin los requerimientos para implementar los complejos protocolos asociados a una red inalámbrica.

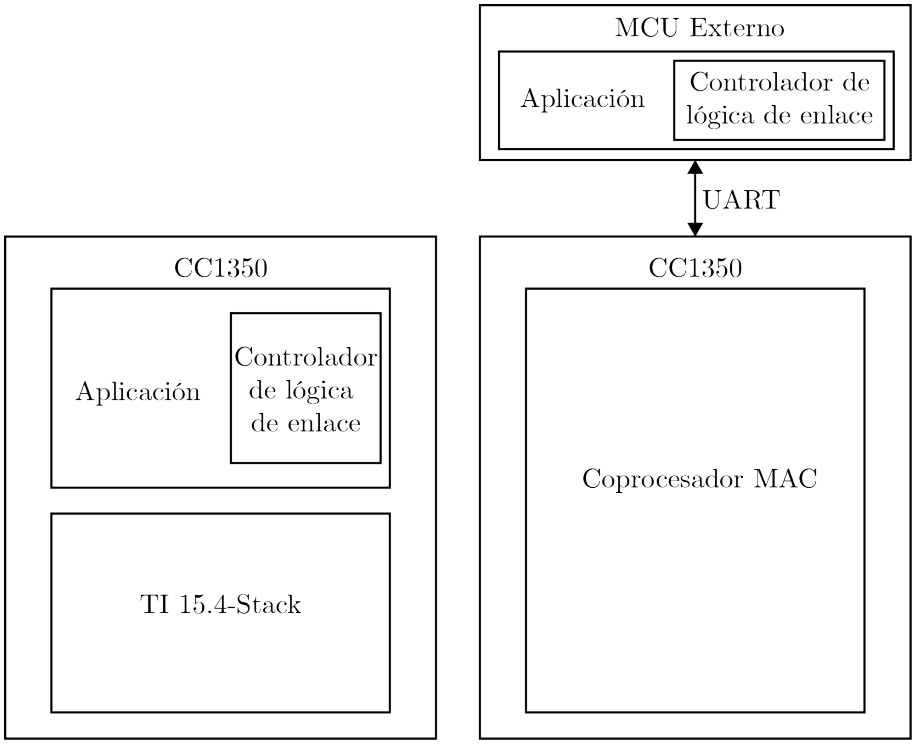


Figura 4.4: Configuración como dispostivo único y como coprocesador

Para el desarrollo de este TFG se utilizarán ambas arquitecturas, usando la arquitectura de dispositivo único para los nodos y la basada en coprocesador para el nodo central o concentrador.

Banda de frecuencias y tasa de transmisión

La elección de una banda y una tasa de transmisión puede elegirse configurando el apropiado atributo (PHY ID). Las opciones son explicadas en la tabla 4.1.

En el TFG se han utilizado las frecuencias de 2.4GHz y 868MHz por ser bandas de uso libre en España. Usando la banda de 868Mhz para la comunicación de largo alcanzo y la de 2.4GHz para enviar los mensajes de la web física.

Indirect Call Framework

ICall es un módulo que provee un mecanismo para que la aplicación se comunique con los servicios del TI 15.4-Stack, así como con ciertos servicios primitivos proporcionados por el RTOS. ICall permite que la aplicación y la pila del protocolo operen eficientemente, comunicandose y compartiendo recursos en un entorno unificado RTOS.

PHY ID	Tasa de datos	Frecuencia canal 0	Nº canales	Espacio canales
0	250 kbps	2405 MHz	16	5 MHz
1	50 kbps	902.2 MHz	129	200 kHz
30	50 kbps	863.125 MHz	34	200 kHz
128	50 kbps	403.3 MHz	7	200 kHz
129	5 kbps	902.2 MHz	129	200 kHz
130	5 kbps	403.3 MHz	7	200 kHz
131	5 kbps	863.125 MHz	34	200 kHz
132	200 kbps	902.4 MHz	64	400 kHz
133	200 kbps	863.225 MHz	17	400 kHz

Tabla 4.1: Bandas permitidas en TI 15.4-Stack y las frecuencias de sus canales

El componente central de la arquitectura ICall es el *dispatcher*, que facilita la comunicación entre la aplicación y las tareas del TI 15.4-Stack.

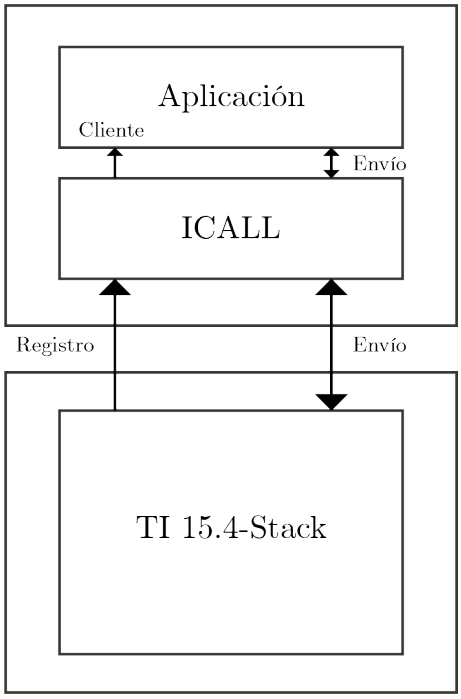


Figura 4.5: Aplicación ICall - Abstracción del protocolo

La figura 4.6 muestra un ejemplo de como un comando se envía desde la aplicación hasta el TI 15.4-Stack, con su correspondiente respuesta.

ICall_ init() inicializa la instancia del módulo ICall y la llamada *ICall_ createRemotetasks()* crea una tarea, con una función de entrada en una dirección conocida. Después de inicializar el ICall, la tarea de la aplicación se registra con el módulo ICall usando *ICall_ registerApp()*. Durante la ejecución de la tarea de la aplicación, esta envía un comando del protocolo como por ejemplo *ApiMac_ mlmeSetreqArray()*. El comando no se ejecuta en el hilo de la aplicación, en lugar, el comando es encapsulado en un mensaje ICall y es dirigido a la tarea del TI 15.4-Stack a través del módulo ICall. Mientras la aplicación espera al correspondiente mensaje. Cuando el TI 15.4-Stack finaliza la ejecución del comando, la respuesta es envíada a través del módulo ICall a la aplicación.

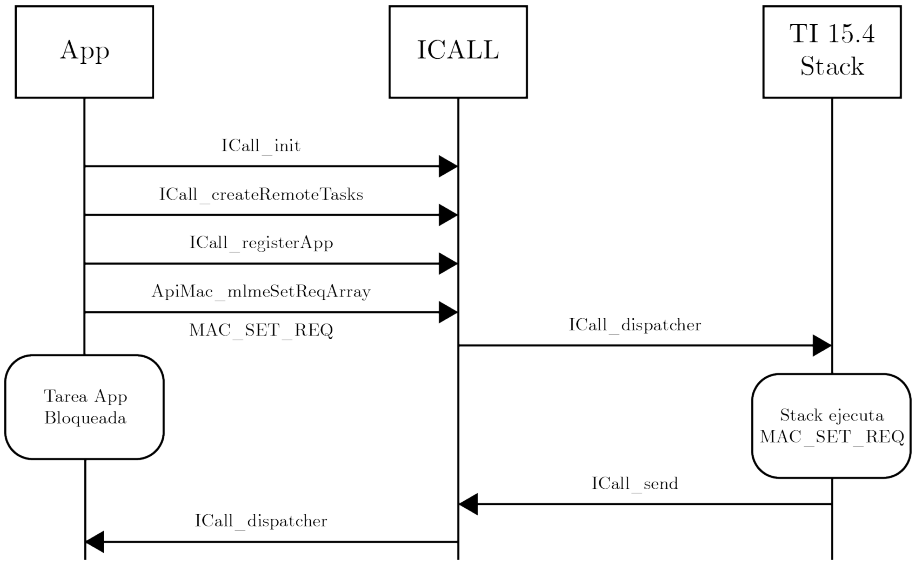


Figura 4.6: Ejemplo de comunicación usando el módulo ICALL

Modos de operación

Modo Beacon Las especificaciones IEEE 802.15.4 definen un modo de operación *beacon-enabled* donde el dispositivo coordinador de la red de área personal o *Personal Area Network* (PAN) transmite *beacons* para indicar su presencia y permite que otros dispositivos encuentren la PAN y se sincronicen. Los *beacons* proporcionan información sobre las especificaciones de la super-trama, que ayuda a los dispositivos con la intención de unirse a la red a sincronizarse y conocer los parámetros de la red antes de comenzar el proceso de unión. La super-trama está dividida en periodos activo e inactivos. Durante el perido activo, los dispositivos se comunican usando el procedimiento *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) excepto en la banda de 863MHz donde se usa el procedimiento *Listen Before Talk* (LBT) para el acceso al canal. Los periodo inactivos permiten a los dispositivos en

la red conservar energía.

Modo NonBeacon Las especificaciones IEEE 802.15.4 definen un modo de operación *non-beacon* donde el coordinador de la red no envía *beacons* periódicos. El modo *non-beacon* es un modo de operación asíncrono donde los dispositivos se comunican usando el mecanismo CSMA/CA.

Modo FrequencyHopping Las aplicaciones que son desarrolladas usando el TI 15.4-Stack pueden ser configuradas para operar en redes con saltos de frecuencia donde los dispositivos de la red cambian de frecuencia. Este modo de funcionamiento está basado en el modo *Directed Frame Exchange* (DFE) de las especificaciones de Wi-SUN FAN.

uBle

Micro BLE Stack (uBle) es una variante del paquete BLE-Stack para los dispositivos con conectividad Sub1-GHz y 2.4GHz, como el CC1350. Este paquete permite a las aplicaciones ser encontradas, escanear o actuar como monitor de conexión. El paquete uBle utiliza el *MultiMode RF Driver*. El *MultiMode RF Driver* permite a las aplicaciones usar ambos modos donde otro protocolo de comunicación es integrado junto al uBle.

El Micro BLE Stack tiene las siguientes restricciones y requisitos:

- Las opciones de diseño dependen de una parcial integración de ICall para guardar recursos del sistema. En el caso de uso del módulo ICall, se tiene que utilizar el sistema de gestión de pila del módulo ICall.
- No puede haber interacción humana-computador porque no existe separación entre el controlador y el host.
- Las opciones de privacidad no son soportadas.
- Para minimizar el consumo de memoria y eliminar redundantes cambios de contexto el uBle no utiliza diferentes tareas en TI-RTOS.
- Solo configuraciones utilizando el *MultiMode RF Driver* pueden ser usada con otros protocolos RF.

4.2.8. Funciones de acceso a la web física

La librería uBle permite a la aplicación enviar un paquete en modo *broadcast* a todos los usuarios que estén en el radio de acción del dispositivo. Este paquete de datos es el que utilizamos para notificar a los usuarios con la información de la web física.

Los paquetes bluetooth usan la trama *Eddystone-URL*, estas forman parte del núcleo de la web física. Una vez el usuario la decodifica la trama podrá acceder a la URL si tiene conexión a internet.

Especificaciones de la trama

Byte offset	Campo	Descripción
0	Tipo de trama	valor = 0x10
1	Potencia TX	Potencia de TX calibrara a 0 m
2	Prefijo de la URL	Prefijo de la web
3+	URL codificada	Longitud de 1-17 bytes

Tabla 4.2: Formato de la trama *Eddystone-URL*

Potencia TX La potencia de transmisión es la potencia recibida a 0 metros, en dBm, en el rango de valores de -100 dBm a +20dBm con una resolución de 1 dBm.

Prefijo de la URL El prefijo de la url define la expansión utilizada por la url, por ejemplo "http://www." o "https://" son codificadas por los bytes 0x00 o 0x03 respectivamente.

Sufijo de la URL El esquema de URL HTTP está definia por RFC 1738, por ejemplo "https://goo.gl/S6zT6P", y es usada para designar recursos accesibles usando HTTP.

4.2.9. Mensajes OTA

Los posibles mensajes entre el concentrador y el nodo, están definidos en el archivo *smsgs.h*. En el Apéndice A se puede encontrar un diagrama con las estructuras de estos mensajes.

Ambos, Nodo y Concentrador han de tener definidos las mismas estructuras de mensajes para una correcta comunicación.

4.3. Concentrador

4.3.1. Introducción

En este capítulo se describe la arquitectura y funcionamiento del concentrador (nodo central). Para este desarrollo se ha utilizado el proyecto de ejemplo que proporciona el fabricante llamado *TI 15.4-Stack Linux Gateway*.

La aplicación sobre Linux proporciona la funcionalidad de concentrador de la red, añadiendo una interfaz como servidor socket para comunicarse con la aplicación *Gateway*. Las aplicaciones del Concentrador y *Gateway* establecen un puente entre el protocolo IEEE 802.15.4 con el protocolo IP siendo una gran punto de comienzo para el IOT.

4.3.2. Diagrama de bloques y modelo de la interfaz

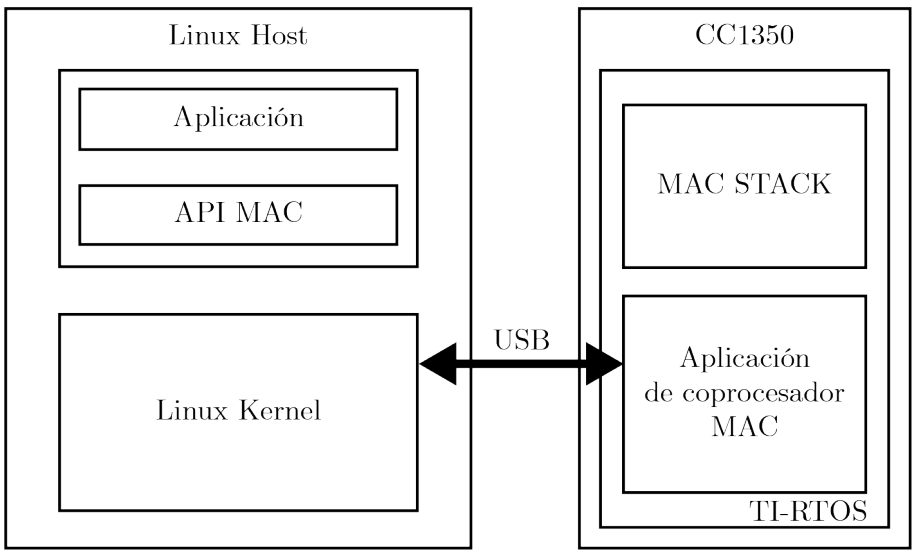


Figura 4.7: Arquitectura de software a alto nivel de las aplicaciones TI 15.4-Stack 2.1.0 Linux®

Esta sección describe la arquitectura de alto nivel basada en coprocesador, los componentes software, y la arquitectura general del sistema (veáse figura 4.7). El coprocesador es una entidad que implementa el estándar MAC IEEE 802.15.4e/g en un chip dedicado y provee una interfaz serie por la que un procesador externo controla y procesa las operaciones del coprocesador.

El concentrador se centra en una arquitectura escalable con una división perfecta donde el procesador ejecuta las capas sobre el IEEE 802.15.4e/g MAC/PHY.

En esta aplicación, el programa se ejecuta en una plataforma basada en Linux. Aunque los componentes de alto nivel , pueden ser conceptualmente aplicados a otras plataformas no basadas en Linux. Los componentes desarrollados serán descritos más adelante.

La interfaz entre el procesador y el coprocesador están definidas como capas lógicas que están separadas en esta arquitectura: una capa física (por ejemplo, USB o UART), una capa lógica de enlace, y la capa de presentación.
Componentes software:

- **Aplicación del coprocesador:** Es el programa ejecutandose en el dispositivo CC1350. Esta aplicación implementa una capa 802.15.4e/g MAC/PHY y proporciona una comunicación serie.
- **Kernel Linux:** El kernel Linux provee los controladores para la interfaz serie que está disponible en un puerto físico (por ejemplo, USB).
- **Aplicación TI 15.4-Stack:** Este módulo implementa la aplicación usando el protocolo 802.15.4e/g y la estructura del modelo MT.

4.3.3. Descripción del directorio del proyecto

La figura 4.8 muestra la estructura del directorio del proyecto del concentrador. A continuación se explica una descripción de alto nivel de cada carpeta:

cc13xx-sbl: Herramientas para la actualización para los dispositivos CC13x0.

collector: Aplicación de ejemplo que demuestra como iniciar una red, permitir la conexión de dispositivos y recoger datos desde dispositivos remotos.

docs En esta carpeta se encuentra la documentación generada por Doxygen.

npi_server2: Interfaz socket para comunicarse con el coprocesador.

google: Contiene un makefile para descargar e instalar el compilador de *Google protocol buffer*.

build.sh Compila el proyecto.

run.sh Ejecuta la aplicación.

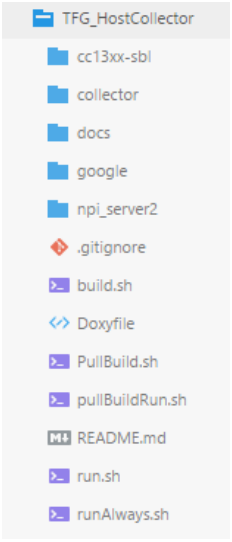


Figura 4.8: Estructura del directorio TI 15.4-Stack 2.1.0 Linux®

PullBuild.sh Descarga una nueva versión del código desde Github y compila el proyecto.

pullBuildRun.sh Descarga, compila y ejecuta el proyecto.

4.3.4. Funcionamiento

El proyecto comienza en la función *main()* en el fichero *linux_main.c*, donde se inicializan las diferentes interfaces, se lee el fichero de configuración y ejecuta la función *App_main()* del fichero *appsrv.c*.

La función *App_main()* se encarga de inicializar los dos hilos de ejecución principales que tiene el programa, *client-thread* y *collector-thread*. La tarea del cliente se encarga de conectarse al servidor y procesar la transmisión y recepción de datos por ese canal. Por otro lado la tarea del concentrador, se encarga de generar la red TI 15.4-Stack y procesar los mensajes enviados por este protocolo.

Hilo del cliente web

Este hilo mantiene la comunicación con el servidor, y espera recibir un mensaje de este. Cuando un mensaje es recibido la función *appsrv_handle_appClient_request()* es la encargada de procesar el mensaje y notificar a la red TI 15.4-Stack a través del hilo del concentrador.

4.3.5. Hilo del concentrador

La función *Collector_process()* del fichero *collector.c* contiene la lógica de este hilo. En la figura 4.9 se observa el diagrama de flujo de ejecución del hilo del concentrador, este hilo comienza inicializando las estructuras de datos y los distintos controladores de la red y pasa a un bucle infinito donde ejecuta una y otra vez las mismas tareas.

En el bucle se revisan los eventos para la inicialización de la red, el envío de mensajes de rastreo o de configuración. Además se procesan los mensajes de la capa de enlace, los eventos de la aplicación y finalmente los eventos de la Api MAC.

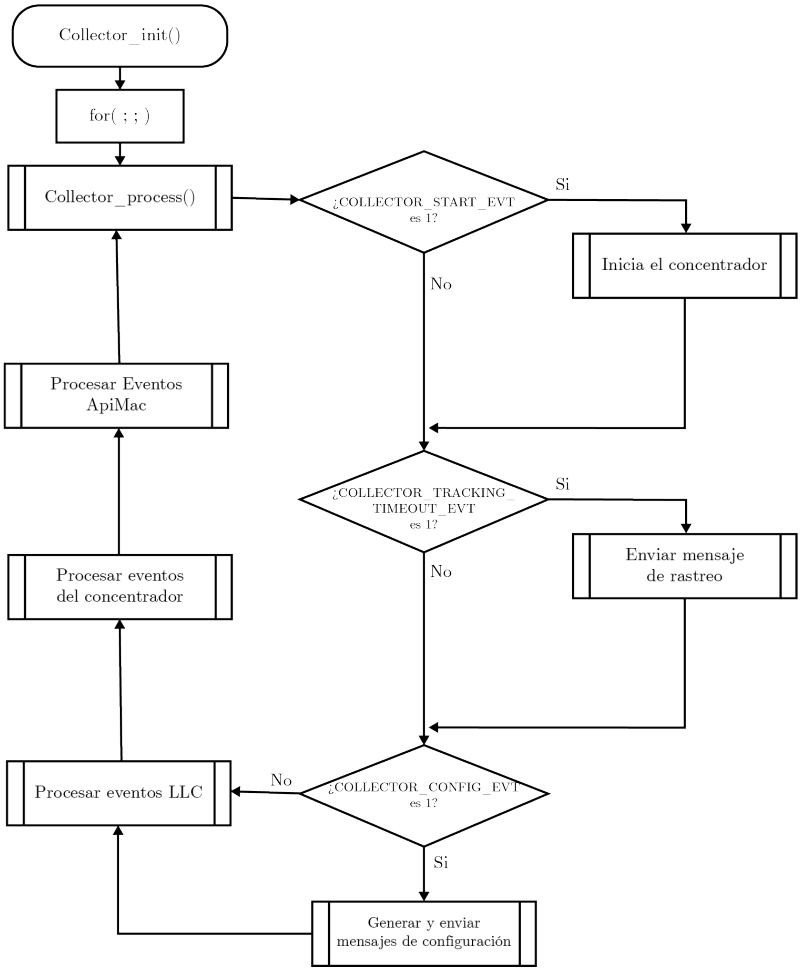


Figura 4.9: Flujo del hilo principal del concentrador

4.3.6. Protocol Buffers

Para facilitar el envío de datos entre el concentrador y el servidor, se ha utilizado *Protocol Buffers*.

Protocol Buffers es un mecanismo flexible, eficiente y automatizado para estructurar datos estructurados. Solo es necesario indicar como se estructuran los datos y al compilarse generan la implementación en multiples lenguajes de programación de los mecanismo para codificar y descodificar datos.

Funcionamiento

La estructura de los datos a codificar se definen en archivos .proto. Cada mensaje es una pequeña estructura que contiene una serie de parejas clave-valor. (Apéndice A)

Como se observa en el listado 4.1, el formato de los mensajes es simple y similar a la definición de variables en código C. Una vez definidos los mensajes, se ejecuta el compilador de *Protocol Buffers* para el lenguaje de tu aplicación, en nuestro caso C.

Listado 4.1: Ejemplo de estructura con Protocol Buffers

```
message Smsgs_configReqMsg
{
    /*! Command ID - 1 byte */
    required Smsgs_cmdIds cmdId = 1;
    /*! Reporting Interval */
    required uint32 reportingInterval = 2;
    /*! Polling Interval */
    required uint32 pollingInterval = 3;
    /*! Time now */
    required uint32 timeNow=4;
    /*! URL ble */
    required string url = 5;
}
```

Las estructuras de datos para nuestra aplicación se pueden observar en el Apéndice A. En este podemos observa que hemos creado un fichero con las mismas estructuras a smsgs.h para poder convertir los mensajes que nos llegan de los nodos en mensajes *Protocol Buffers* y así poder enviarlos al servidor.

Capítulo 5

Servidor

Contenido

5.1 Introducción 31

5.2 Estructura del directorio 31

5.3 Inicio del servidor 33

5.4 Back-end 34

5.4.1 Mensajes enviados entre el concentrador y el Back-end . . 34

5.4.2 Mensajes enviados entre el Back-end y Front-end 35

Front-end 36

5.1. Introducción

El servidor se puede dividir en dos partes *Front-end* y *Back-End* que son términos que se refieren a la separación entre una capa de presentación y una capa de acceso a datos respectivamente.

5.2. Estructura del directorio

El servidor está compuesto por el directorio que se observa en la figura 5.1 y a continuación se describe la función de cada directorio:

api: en este directorio se encuentran las definiciones de llamadas *Representational State Transfer* (REST) al servidor. Las llamadas REST se utilizan para poder acceder a los datos almacenados desde fuera del servidor.

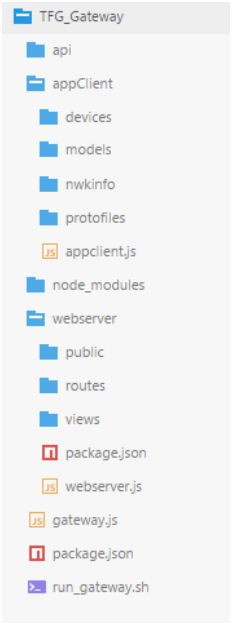


Figura 5.1: Estructura del directorio del servidor

appClient: Esta carpeta contiene el *Back-end* de la web:

- **devices:** se definen las funciones relacionadas con la gestión de los nodos.
- **models:** aquí se definen los modelos para guardarlos en la base de datos.
- **nwkinfo:** se definen las funciones relacionadas con la gestión del concentrador.
- **protofiles:** aquí se almacenan los ficheros .proto de *Protocol Buffers*.
- **appClient.js:** en este fichero se inicia el servidor que se comunica con el concentrador y se procesan los mensajes.

node_modules: librerías utilizadas en el proyecto.

webserver: en este directorio está contenida la lógica del *Front-end*:

- **public:** interfaz del cliente en AngularJS.
- **routes:** Definición de rutas.
- **views:** archivos html de las vistas.
- **webserver.js:** se inicia el cliente y se gestiona las peticiones a la web por parte del usuario.

gateway.js: inicia el *back-end* y el *front-end* y la comunicación entre ambas por *web-sockets*.

run_gateway.sh: *script* para iniciar el servidor.

5.3. Inicio del servidor

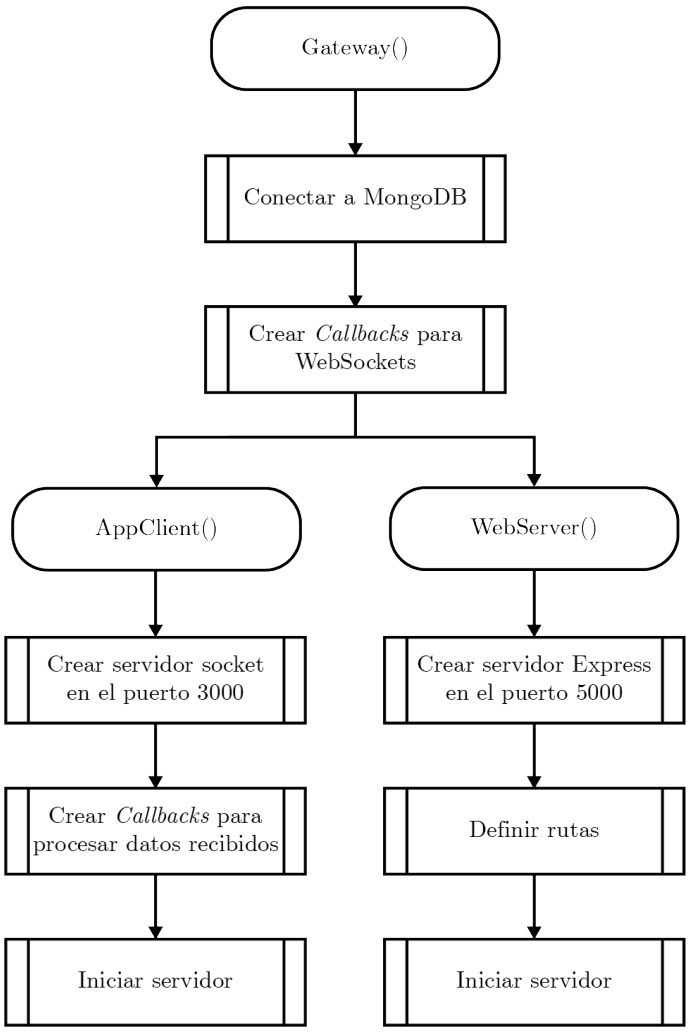


Figura 5.2: Diagrama de flujo en el inicio del servidor

La función *Gateway()* inicia todo el servidor, en esta se realiza la conexión a la base de datos MongoDB y se crean las funciones *Callbacks* para el intercambio de información entre el *Back-end* y *Front-end* usando *WebSockets*. La figura 5.2 refleja el proceso de inicio del servidor.

5.4. Back-end

El *Back-end* es el área que se dedica a la lógica, en este encontramos una interfaz que se encarga de comunicarse con el concentrador y otra que se encarga de comunicarse con el *Front-end* usando *Web-Sockets*.

Como se observa en la figura 5.2, la función *AppClient()* es la que implementa la creación del servidor en el puerto 3000 para que el concentrador pueda enviar datos al servidor.

Cuando la conexión entre el concentrador y el collector está establecida, el *Back-end* queda a la espera de recibir datos del concentrador o del *Front-end*. Todos los datos recibidos del concentrador se almacenan en la base de datos MongoDB para tener un registro de todos los datos de la red que pudieran ser útiles en un futuro y se envían al *Front-end*.

5.4.1. Mensajes enviados entre el concentrador y el Back-end

En la tabla 5.1 se pueden observar todos los posibles mensajes enviados entre el concentrador y el *Back-end*. Estos mensajes están definidos en los ficheros .proto tanto en el concentrador y en el *Back-end*.

Identificador de mensaje	**Descripción**
DEVICE_JOINED_IND	Indicación de que un nodo se ha conectado a la red
DEVICE_LEFT_IND	Indicación de que un nodo se ha desconectado de la red
NWK_INFO_IND	Indicación de que está disponible la información de la red
GET_NWK_INFO_REQ	Petición de información de la red
GET_NWK_INFO_RSP	Respuesta de información de la red
GET_NWK_INFO_CNF	Acuse de recibido de GET_NWK_ INFO_RSP
GET_DEVICE_ARRAY_REQ	Petición de la lista de dispositivos de la red
GET_DEVICE_ARRAY_CNF	Acuse de recibido de GET_DEVICE_ ARRAY_REQ
DEVICE_NOTACTIVE_UPDATE_IND	Indicación de que un dispositivo no está activo
DEVICE_DATA_RX_IND	Indicación de los datos recibidos de un dispositivo
COLLECTOR_STATE_CNG_IND	Indicación del estado del concentrador
SET_JOIN_PERMIT_REQ	Petición de permiso de conexión de nuevos dispositivos
SET_JOIN_PERMIT_CNF	Acuse de recibido de SET_JOIN_ PERMIT_REQ
TX_DATA_REQ	Petición de transmisión de datos
TX_DATA_CNF	Acuse de recibido de TX_DATA_REQ
GET_COLLECTOR_STATS_REQ	Petición de las estadísticas del concentrador
GET_COLLECTOR_STATS_RSP	Respuesta con las estadísticas del concentrador

Tabla 5.1: Identificadores de los mensajes enviados entre el concentador y el servidor

5.4.2. Mensajes enviados entre el Back-end y Front-end

Para que se pueda administrar la red y observar todos sus parámetros en el *Front-end* se han creado los mensajes que se observan en la tabla 5.2 que se envían a través de *WebSockets*.

Identificador	**Descripción**
sendConfig	Envío de la petición de configuración
sendToggle	Envío de conmutación de un led de la placa de un nodo
sendToggleBLE	Envío de conmutación del estado del BLE de un nodo
getDevArrayReq	Obtener la lista de dispositivos
getNwkInfoReq	Obtener la información de la red
setJoinPermitReq	Permitir la conexión de nuevos dispositivos
changeUrl	Cambiar la URL que envía un nodo
bikeSelectReq	Seleccionar una bicicleta en un nodo de tipo aparcamiento
collectorStatReq	Obtener las estadísticas del collector
nodeChangePosition	Cambiar las coordenadas de un nodo
collChangePosition	Cambiar las coordenadas del concentrador
permitJoinCnf	Respuesta del mensaje setJoinPermitReq
connDevInfoUpdate	Actualización de la información de un nodo
nwkUpdate	Actualización de la información de la red
getdevArrayRsp	Respuesta del mensaje getDevArrayReq

Tabla 5.2: Tipos de mensajes enviados por WebSocket entre el Back-end y el Front-end

Front-end

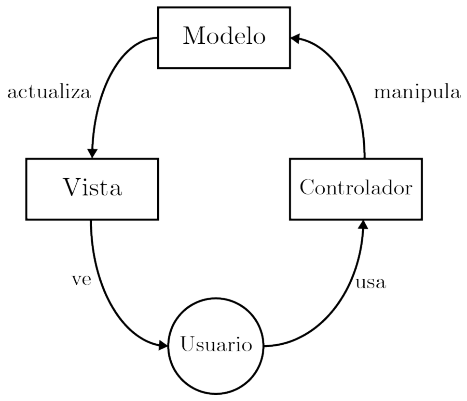


Figura 5.3: Diagrama de flujo en el inicio del servidor

El *Front-end* es la interfaz del servidor con el usuario. Para facilitar el control de las vistas se ha utilizado el *framework AngularJS*.

Angular es un *framework* Modelo-Vista-Controllador (MVC) para la construcción de aplicaciones de una única página del lado del cliente en HTML y JavaScript.

En la figura 5.3 se representa la interacción entre las diferentes partes de la arquitectura MVC:

Vistas: Será el código HTML y todo lo que presente los datos o información.

Controladores: Se encarga de toda la lógica de la aplicación.

Modelo: El modelo es la estructura que define un tipo de dato, y permite acceder a él en la base de datos.

Todas las funcionalidades del *Front-end* se pueden observar en el manual de uso (Apéndice C).

Capítulo 6

Pruebas

Contenido

6.1 Prueba en entorno real 39
6.1.1 Introducción 39
6.1.2 Metodología 39
6.1.3 Resultados 40
6.2 Prueba de consumo 41
6.2.1 Introducción 41
6.2.2 Medida del consumo 41
6.2.3 Resultados 42

6.1. Prueba en entorno real

6.1.1. Introducción

Esta prueba consiste en simular el funcionamiento de la red en un entorno lo más parecido a un caso de uso real. Para ello se ha instalado el concentrador en un punto fijo como se observa en la figura 6.1 (marcador central del círculo) y el nodo se ha ido cambiando de posición. Con esta prueba se ha comprobado el alcance de la red, el envío de los mensajes de la web física por Bluetooth y las reconexiones de los nodos a la red.

6.1.2. Metodología

Cómo se ha indicado anteriormente, el concentrador se ha ubicado en un punto fijo que estuviera elevado, por ello se ha situado en el balcón de la 2º planta de un

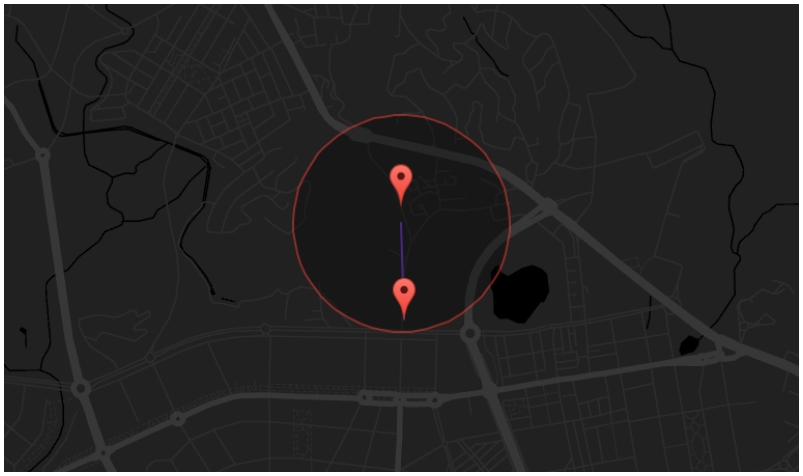


Figura 6.1: Posición del concentrador y nodo durante la prueba en el punto de máximo alcance

edificio. Y el nodo se ha ido cambiado de posición hasta llegar a la distancia máxima y que este perdiera la conexión con el concentrador, después de esto se ha vuelto a acercar para comprobar la reconexión a la red.

6.1.3. Resultados

Después de realizar las pruebas se ha estimado la distancia máxima a la que el nodo puede situarse para poder mantener la conexión que es de 400m, pero esta medida mejoraría cambiando la antena y utilizando un amplificador en transmisión lo que quedará como una línea futura de mejora de este proyecto.

En cuanto a la reconexión del nodo, ha sido todo un éxito. Después de repetir la desconexión y reconexión del nodo varías veces, en todas de ellas la ha realizado sin problemas. Y además no ha dejado de enviar los mensajes de la web física en ningún momento durante la prueba.

Finalmente en la figura 6.2 se observan las estadísticas que ha recogido la red

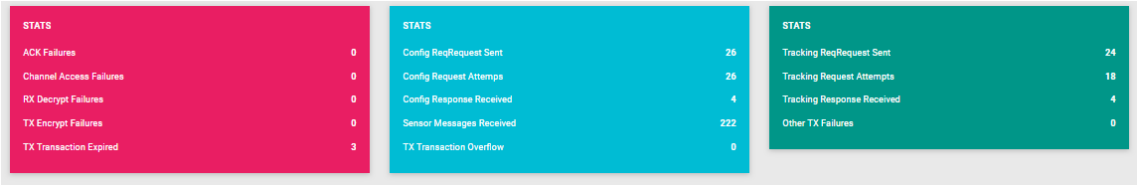


Figura 6.2: Estadísticas de la red después de la prueba

después de las pruebas, donde se tiene que el concentrador ha recibido 26 peticiones

del nodo para que le envíe la configuración (esta petición se realiza después de una conexión o reconexión) y 222 mensajes de tipo "Sensor"que son los que el nodo envía periódicamente con datos de temperatura, batería, estadísticas y algunos parámetros más.

6.2. Prueba de consumo

6.2.1. Introducción

Para una exitosa implementación del IOT uno de los requisitos más importantes es un consumo eficiente del sistema.

La comunicación inalámbrica entre los nodos comienza a ser un problema cuando la fuente de alimentación es limitada. Un sistema que pueda funcionar con una sola pila AAA durante años es lo ideal en el IOT [15]. Por ello, durante el desarrollo de este proyecto se ha utilizado como protocolo inalámbrico el IEEE 802.15.4, que destaca su bajo consumo.

6.2.2. Medida del consumo

Para realizar la medida de consumo se ha utilizado el multímetro digital Keysight 34411A, que proporciona 6 dígitos y medio de resolución o una velocidad de muestreo de 50000 muestras/s a una resolución inferior. [16]

Para el control del multímetro se ha utilizado dos programas de LabView, uno de ellos permite obtener datos a la máxima velocidad de muestreo aunque eso limite la resolución a 4 dígitos (figura 6.3) y el otro programa permite tomar datos promediados en intervalos de tiempo (figura 6.4).

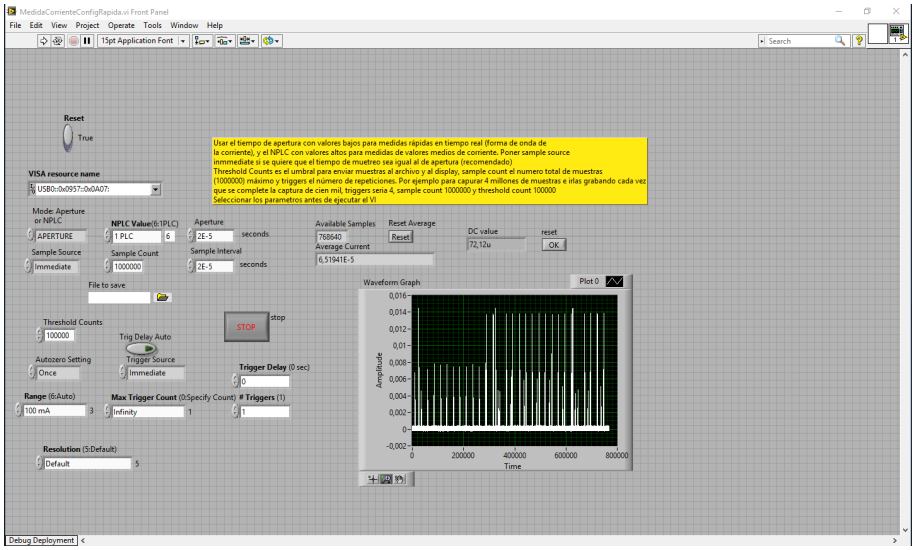


Figura 6.3: Programa de LabView para captura rápida

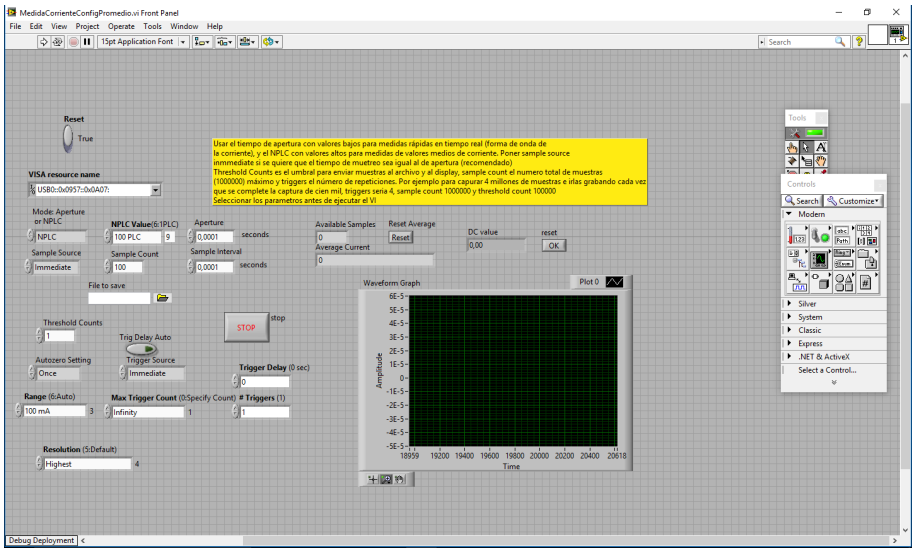


Figura 6.4: Programa de LabView para captura en promedio

6.2.3. Resultados

Aunque el objetivo de esta prueba es conocer el consumo medio de corriente del nodo. Antes de nada se necesita conocer que fondo de escala utilizar en el multímetro para la medida en promedio, para ello se hace uso del programa de captura a máxima frecuencia de muestreo, donde se obtiene la corriente instantánea máxima que consume el nodo, unos 15 mA (figura 6.5).

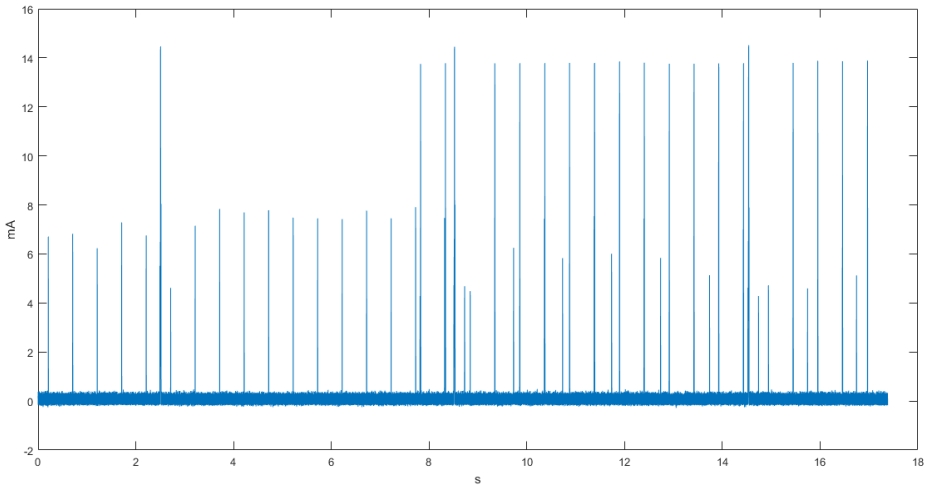


Figura 6.5: Consumo en mA del nodo a la máxima frecuencia de muestreo

Finalmente, utilizando el programa de medida promediada de LabView con un fondo de escala de 100mA, realizamos una medida de la corriente promedio durante 50 ciclos de subidas al servidor. Con esta configuración se obtiene que el consumo medio de un nodo es de 169.6 μA, lo que equivale a más de 5 meses de funcionamiento utilizando una pila de botón CR 2477.

Parte III

Conclusiones y lineas futuras

Conclusiones

Después del desarrollo del proyecto, es pertinente hacer una valoración final del mismo, respecto a los resultados obtenidos, las expectativas o el resultado de la experiencia acumulada.

Este TFG nace con el objetivo de ser un trabajo multidisciplinar que abarque gran parte de los conocimientos adquiridos durante una titulación generalista, como son las comunicaciones, telemática y electrónica. Desde este punto se hizo un análisis de diferentes tecnologías recientes y se seleccionaron dos como principales: el concepto de "Web física" y el microntrolador "CC1350". De esta forma surge el objetivo principal de este TFG que es la integración de estas dos tecnologías en una plataforma que pudiera utilizarse como una solución al problema de la interacción a través de la red.

En este TFG se ha conseguido con éxito la integración de ambas tecnologías en el marco de la web física desde la capa física hasta las capas más altas.

Durante el desarrollo de este TFG surgieron problemas debidos principalmente al poco tiempo que llevaba el microcontrolador en el mercado que dificultaba la búsqueda de información. Con el tiempo fue surgiendo más documentación que hizo cambiar la linea de diseño del proyecto hasta llegar a la que plasma este TFG.

Para este proyecto se ha utilizado Github que es un servicio para alojar proyectos utilizando el sistema de control de versiones GIT. Al inicio del proyecto se crearon cuatro repositorios diferentes donde se ha guardado la mayor parte de la información del TFG. Los repositorios son enumerados a continuación:

TFG_memoria (https://github.com/joseayebenes/TFG_Memoria) Almacena la información sobre esta memoria escrita en LaTeX.

TFG_HostCollector (https://github.com/joseayebenes/TFG_HostCollector) Este repositorio contiene el proyecto del concentrador o nodo central de la red.

TFG_Sensor (https://github.com/joseayebenes/TFG_Sensor) Contiene el código que ejecutan los nodos de la red.

TFG_Gateway (https://github.com/joseayebenes/TFG_Gateway) En este se guarda el proyecto del servidor.

Finalmente, se han visto concluidos satisfactoriamente los principales objetivos de este proyecto, que han permitido un mejor conocimiento de las tecnologías utilizadas y de la metodología de trabajo para llevar un proyecto desde el concepto hasta el prototipo.

Líneas futuras

Durante el desarrollo del TFG se ha observado un crecimiento en el uso de las tecnologías que implementa este proyecto, lo que vaticina un futuro donde el concepto de web física sea familiar a la población. De este crecimiento se extraen lineas futuras de desarrollo en este ámbito como son:

- Mejora del alcance analizando las diferentes alternativas que ofrece TI 15.4-Stack y añadiendo una antena diferente.
- Crear un nodo que emule una máquina expendedora, donde el usuario pueda realizar su compra a través de la web física.
- Mejora de la duración de la batería optimizando el código y añadiendo tecnología de recolección de energía.
- Creación de placas de circuito impreso para nodo y concentrador que permita reducir el tamaño.
- Desarrollo de una aplicación para teléfonos móviles que detecte los mensajes BLE de la web física y pueda filtrarlos según su tipo (publicidad, servicios, etc).

Parte IV

Apéndices

Apéndice A

Estructuras de mensajes OTA

Em la figura A.1 se observan los distintos tipos de mensajes que se envían sobre la red inalámbrica, estos mensajes están definidos en el fichero *smsgs.h*

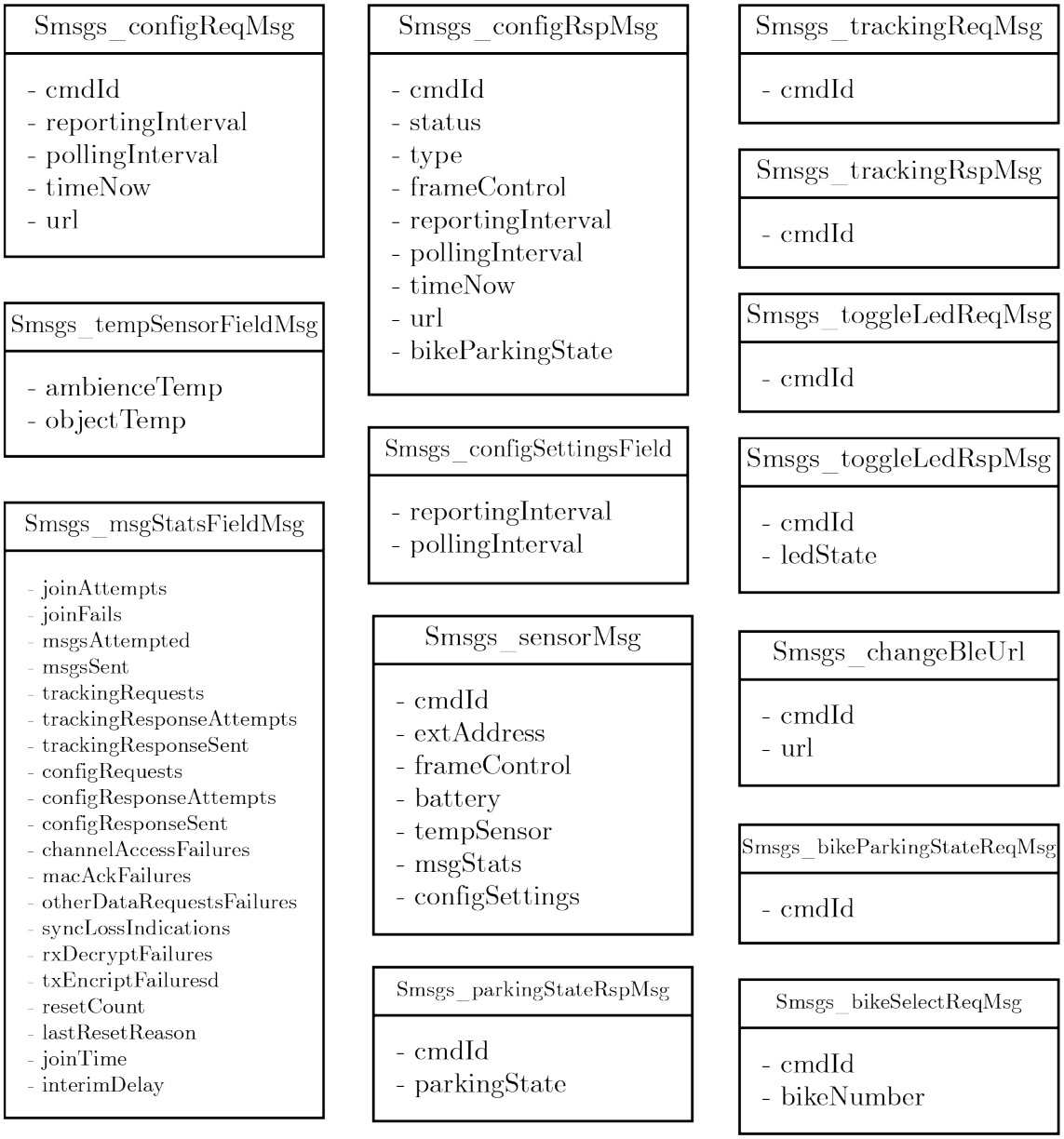
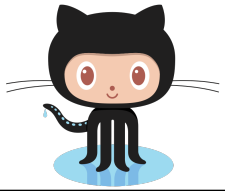
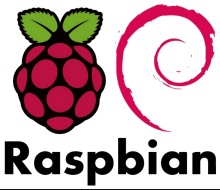


Figura A.1: Estructuras de los mensajes del fichero smsgs.h

Apéndice B

Herramientas utilizadas

	GitHub es una plataforma para alojar proyectos utilizando el sistema de control de versiones GIT.
	*Amazon Web Services* (AWS) es una colección de servicios web que conjunto forman una plataforma de computación en la nube.
	TeXstudio es un entorno de escritura intregrado para crear documentos LaTeX.
	LaTeX es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica.
	Adobe Illustrator es un editor de gráficos vectoriales y está destinado a la creación artística de dibujo y pintura para ilustración.

	Sublime text es un editor de texto y editor de código fuente.
	Node.js es un entrono de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome.
	AngularJS es un *framework* de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.
	Code Composer Studio es un entorno integrado de desarrollo que soporta los microcontroladores de Texas Instruments.
	MongoDB es una base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.
	Raspbian es un distribución del sistema operativo GNU/-Linux y basado en Debian Jessie para la Raspberry Pi.
	Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.

Apéndice C

Manual de uso

C.1. Inicio de la red

C.1.1. Iniciar el servidor

El primer paso es iniciar el servidor, ya que toda la red depende de este. Como se ha utilizado Amazon Web Services, desde su web iniciamos la instancia que corre en el inicio el servidor de la aplicación (véase figura C.1).

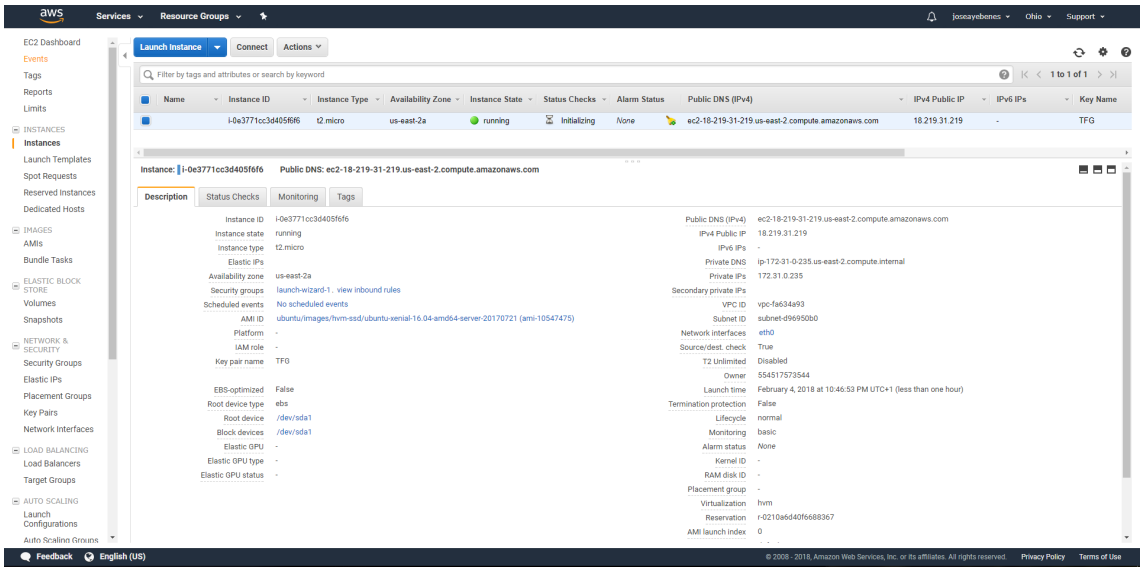


Figura C.1: Web de gestión de instancias en Amazon Web Services

Después de iniciar el servidor, es posible acceder a la web de gestión desde la dirección DNS de la instancia del servidor en el puerto 5000. Inicialmente el concen-

trador no está conectado por lo que nos aparece un mensaje indicándolo (ver figura C.2).

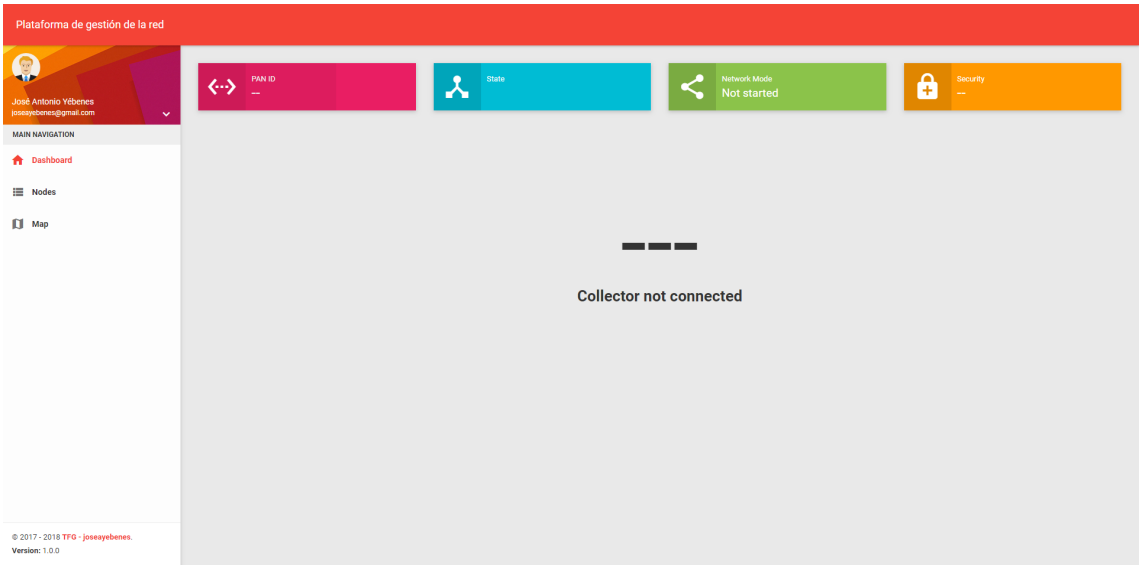


Figura C.2: Apariencia de la web cuando no está conectado el concentrador

C.1.2. Encender concentrador

Antes de enceder la RaspberryPi, conectamos el coprocesador por USB y damos conexión a Internet a la RaspberryPi. La RaspberryPi está configurada para ejecutar el programa del concentrador y conectarse al servidor, por lo que no es necesaria más interacción con el concentrador. Después de encenderse la RaspberryPi, la web habrá cambiado y ahora ya nos indicará algunos parámetros de la web (ver figura C.3).

En la parte superior de la web se observan los parámetros de la red como son su dirección PAN, el estado que indica si la red permite la conexión de los nodos, el tipo de comunicación y si está activada la seguridad.

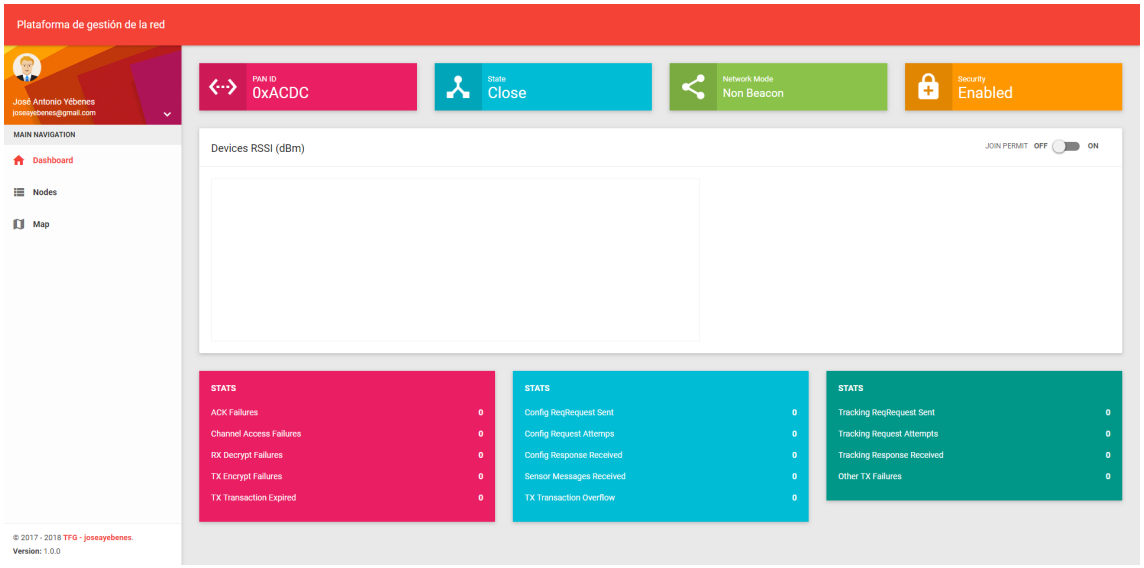


Figura C.3: Apariencia de la web cuando ya está conectado el concentrador

C.1.3. Encender nodos

Encendemos todos los nodos necesarios, en las pruebas realizadas hemos utilizado dos.

C.1.4. Permitir conexiones de los nodos

Para permitir la conexión de los nodos a la red, es necesario poner el botón "*JOIN PERMIT*" a "ON". Después de eso, los nodos que ya haya encendidos, se conectarán a la red y aparecerán en la web.

En la figura C.4 se observa la apariencia de la web cuando dos dispositivos está conectados. En la parte central se observa una gráfica con potencia de la señal de cada uno de los nodos en dBi, y en la parte inferior estadísticas generales de la red.

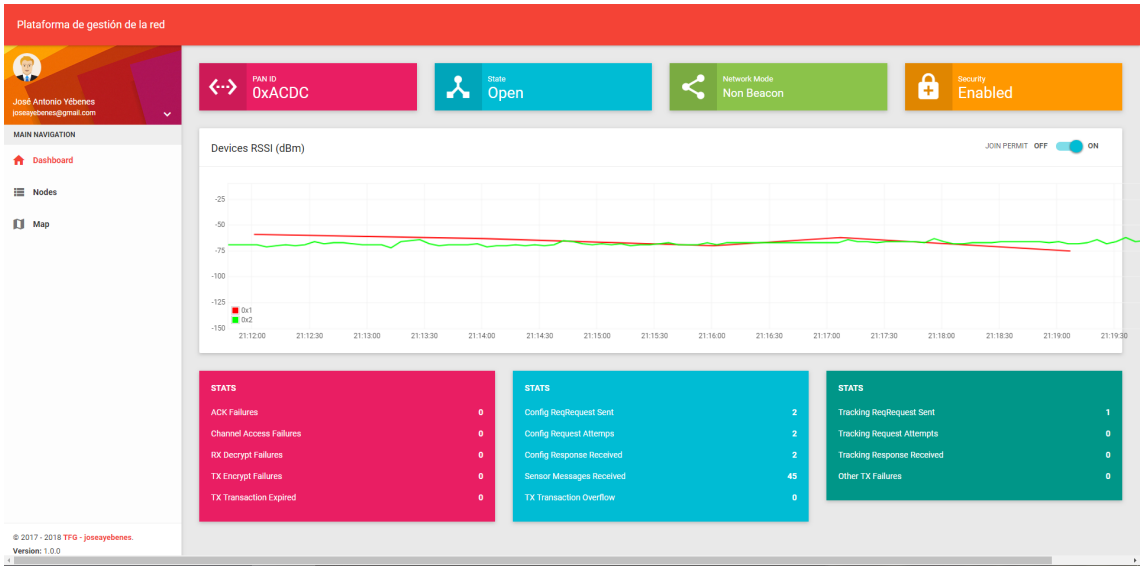


Figura C.4: Apariencia de la web cuando hay dos nodos conectados

C.2. Administración de los nodos

Si se hace click en la barra de navegación derecha en el botón "*Nodes*", se accede al panel de administración de los nodos, en el aparecen la lista de nodos conectados y la información relevante de cada uno, como se observa en la figura C.5.

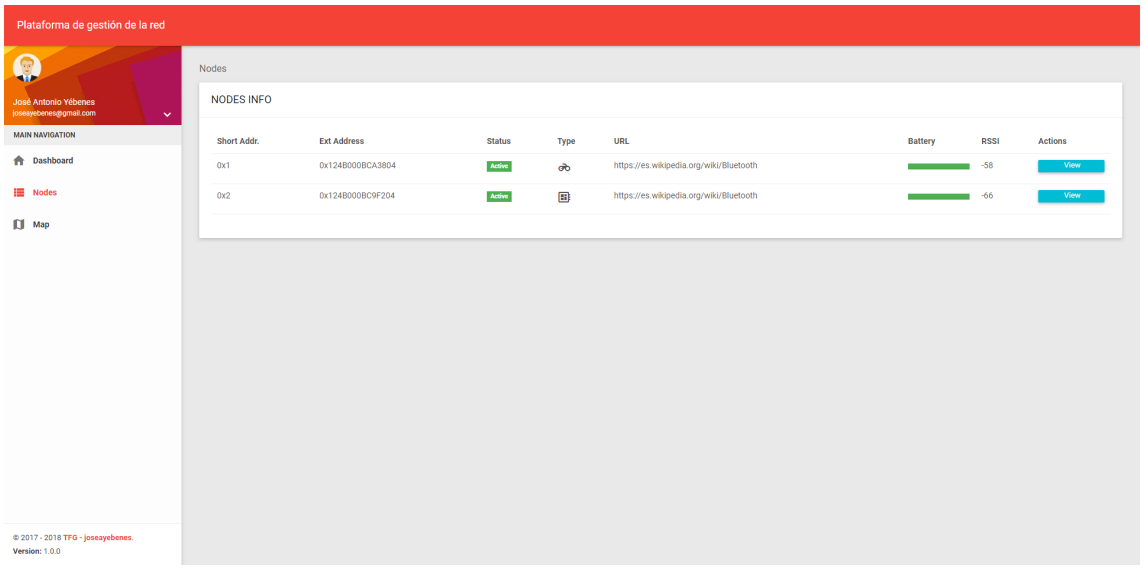


Figura C.5: Panel de administración de los nodos

Haciendo click en alguno de los botones "*View*", se accede a la administración del nodo correspondiente, como se aprecia en la tabla hay dos tipos de nodos, uno representado con una bicicleta y el otro con el símbolo de un microcontrolador, que representan un nodo destinado al control de un parking de bicicletas y un nodo genérico de información respectivamente. Cada tipo de nodo tiene una interfaz de administración diferente.

C.2.1. Administración de nodo genérico

En la figura C.6, se puede observar toda la información relevante sobre el nodo seleccionado.

En el cuadro de "*Physical Web - URL*" es posible modificar la URL que envía dicho nodo. Justo debajo del título aparece la URL que actualmente está enviando. En el cuadro se puede introducir cualquier URL de tipo HTTPs y al hacer click en el botón se enviará el cambio al nodo.

También es posible indicar cuales son las coordenadas del nodo desde el cuadro "*Position*".

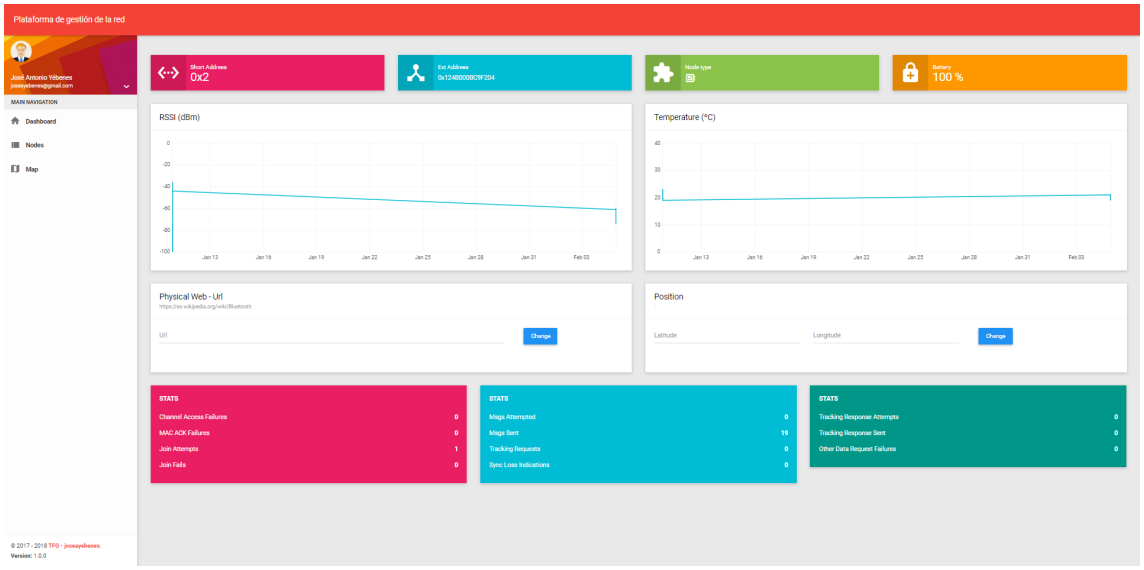


Figura C.6: Panel de administración de un nodo tipo genérico

Administración de un tipo parking de bicicletas

El panel de administración de este tipo de nodo se observa en la figura C.7. Esta web permite lo mismo que la web de un nodo genérico, pero además hay un cuadro llamado "*Parking*", en este cuadro se observaría el estado de cada anclaje del parkin, y si se hace click en cualquiera de ellos el estado del anclaje cambiaría de abierto a cerrado o vicerversa. Esta función está simulada en el nodo como un array donde se guardan los estados de los candados.

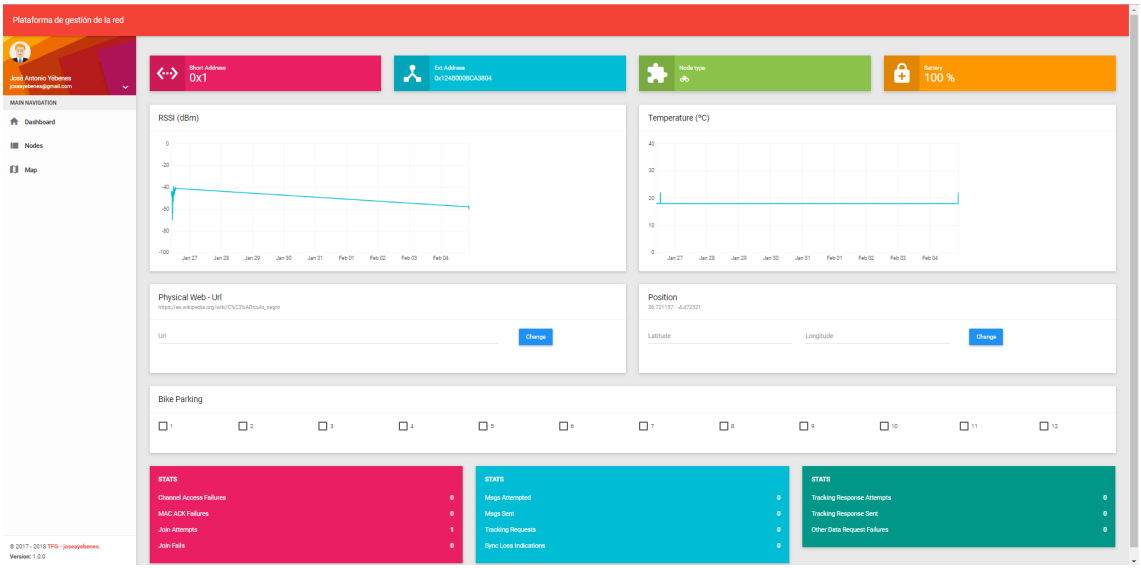


Figura C.7: Panel de administración de un tipo parking de bicicletas

C.3. Mapa

Desde la web también es posible observar donde está cada nodo en la red, para ello accediendo desde la barra lateral a la opción "*Map*" se observa un mapa como el que se puede observar en la figura C.8. En este mapa podemos observar las posiciones de los nodos que se han indicado en sus respectivos paneles de administración y además desde el formulario inferior es posible cambiar la posición del concentrador y su radio de alcance. Con esto es posible tener una idea de si alguno de los nodos se ha colocado demasiado cerca del borde del alcance del concentrador.

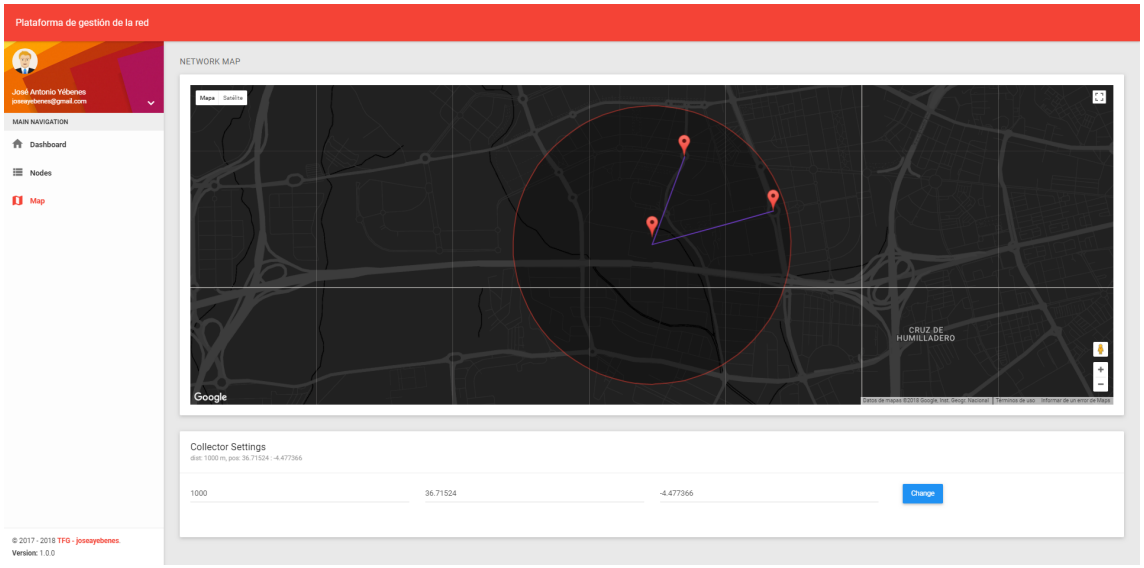


Figura C.8: Vista del mapa con las posiciones de los nodos

Bibliografía

[1] Roy Want, Bill N. Schilit, and Scott Jenson. Enabling the internet of things. *Computer*, 2015.

[2] Dmitry Namiot and Manfred Sneps-Sneppe. The physical web in smart cities. *Advances in Wireless and Optical Communications*, 2015.

[3] Yue Liu, Ju Yang, and Mingjun Liu. Recognition of QR code with mobile phones. In *Control and Decision Conference*.

[4] Gerd Kortuem, Fahim Kawsar, and Vasughi Sundramoorthy. Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, 2009.

[5] Luca Mainetti, Luigi Patrono, and Antonio Vilei. Evolution of wireless sensor networks towards the internet of things: A survey. *Telecommunications and Computer Networks*, 2011.

[6] Michele Zorzi, Alexander Gluhak, and Sebastian Lange. From today's intranet of things to a future internet of things: a wireless and mobility-related view. *IEEE Wireless Communicationss*, 2010.

[7] Carles Gomez and Josep Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 2010.

[8] Zigbee Alliance. Zigbee home automation public apllication profile. Technical report, Zigbee Alliance, 2007.

[9] Zigbee Alliance. The zigbee rf4ce standard. Technical report, Zigbee Alliance, 2009.

[10] Z-Wave. Z-wave protocol overview. Technical report, Z-Wave, 2007.

[11] Usama Mehboob, Qasim Zaib, and Chaudhry Usama. Survey of iot communication protocols techniques, applications, and issues. *xFlow Research Inc, Pakistan*, 2016.

[12] Ana Belén García Hernando, José Fernán Martínez Ortega, Juan Manuel López Navarro, Aggeliki Prayati, and Luis Redondo López. *Problem Solving for Wireless Sensor Networks.* Springer, 2008.

[13] Jianliang Zheng and Myung J Lee. A comprehensive performance study of ieee 802.15. 4. *Sensor network operations*, 4:218–237, 2006.

[14] Texas instruments. Ti 15.4-stack - simplelink cc13x0 ti 15.4-stack user's guide 2.04.00.00 documentation, enero 2018.

[15] Mahmoud Shuker Mahmoud and Auday AH Mohamad. A study of efficient power consumption wireless communication techniques/modules for internet of things (iot) applications. *Advances in Internet of Things*, 6(02):19, 2016.

[16] Keysight. *Keysight Technologies 34410A and 34411A Multimeters. Data sheet.*