Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE GRADO

TITULO DEL TFG: Internet of Things (IoT) implementación con nodos Zolertia RE-Mote

TITULACION: Grado en Ingeniería de Sistemas de Telecomunicación

AUTOR: Sergi Molinero Baixas

DIRECTOR: José Polo

Fecha: 1 de Febrero de 2018

Resumen

Hoy en día estamos rodeados de dispositivos electrónicos que nos facilitan el trabajo y nos ayudan en las tareas cotidianas de las personas. Estos aparatos son cada vez más sofisticados, aportando una mayor información y control de las cosas. Para ello se creó el llamado Internet de las Cosas (IoT por sus siglas en inglés) que consiste en poder conectar millones de dispositivos a Internet. La clave para que todos estos dispositivos puedan conectarse a Internet y poder interactuar entre ellos, es el uso de protocolos y estándares nuevos, que nos permitan interactuar con los usuarios.

En este proyecto se van a testear y probar los sensores RE-Mote de la marca Zolertia. Para ello se van a implementar sobre el estándar IEEE 802.15.4, y el protocolo abierto 6LoWPAN, que es ideal para equipos con ancho de banda y recursos limitados. Estos sensores consisten en una placa hardware inalámbrica, conectada a una batería recargable. Para programarlos, se utilizará Contiki OS, un sistema operativo para dispositivos IoT.

Estos sensores crearán una red WSN (Wireless Sensor Network) y se comunicaran entre ellos con el protocolo RPL (Routing Protocol for Low-Power and Lossy Networks). Asimismo, se comunicaran a Internet a través de un BR (Border Router o Router Frontera). Hay tres tipos de motas: la mota conectada al BR que actuara de Master y recibirá toda la información de la red, motas que harán de enrutador de paquetes y a la vez enviaran información al master, y las motas finales que solamente enviaran la información a las motas vecinas.

Para desarrollar el router frontera que separa la red PAN (Personal Area Network) de Internet, se ha utilizado una Raspberry Pi. Para ello se ha configurado la solución CETIC 6LBR[1]. A parte se ha creado un proceso en PYTHON[2] para que los datos recibidos sean almacenados en el servidor de UBIDOTS[3] que nos permite almacenar datos para después poder ser visualizados desde cualquier parte del mundo.

En el proyecto se prueba que: a) estos sensores son realmente eficientes para un consumo autónomo, b) una conexión inalámbrica sin pérdida de paquetes c) una fácil implementación d) fácil de gestionar y visualizar.

ÍNDICE

1. INTERNET DE LAS COSAS (IoT)

1.1 Introducción

Hoy en día el Internet de las cosas, se refiere a la interconexión digital de objetos cotidianos que sean inteligentes para que se puedan conectar con Internet y poder así facilitar el día a día de las personas. Cada vez más los aparatos electrónicos se conectan a Internet para enviar información de su estado y poder ser gestionados remotamente.

Figura 1. Internet of Things

1.2 Tecnologías de comunicación

IoT utiliza diferentes tecnologías de comunicación, desde comunicaciones inalámbricas conocidas como el protocolo IEEE802.15.4, Wifi, Bluetooth, redes móviles o también por cable Ethernet. La ventaja de estos dispositivos es que tienen una escalabilidad muy amplia para adaptarse a los diferentes y variables equipos de comunicación. En la Figura 2 se muestra una comparativa, del alcance y velocidad de transmisión, para varios protocolos inalámbricos.
En este proyecto utilizamos la arquitectura basada en el estándar IEEE 802.15.4, que nos permite una comunicación de corto/medio alcance y una baja transferencia de datos. Pudiendo así obtener consumos de energía muy bajos para sistemas autónomos.

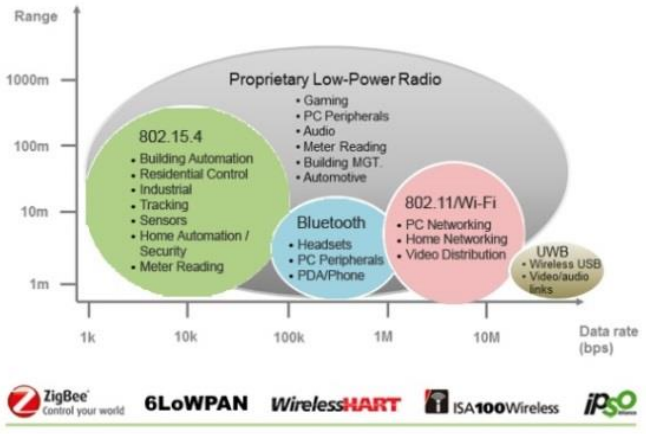

Figura 2. Tecnologías de comunicación según velocidad y distancia

1.3 WSN Red de sensores

Una WSN (Wireless Sensor Network) o red de sensores inalámbricos, es una red de ordenadores de tamaño muy pequeño (nodos o motas), equipados con sensores, que colaboran en una tarea común. Estos nodos forman una red inalámbrica y se conectan a través de un border router a otra red exterior. Los nodos son sensores autónomos especialmente distribuidos para monitorizar condiciones físicas o ambientales, como temperatura, sonido, presión, etc. Las redes más modernas son bidireccionales, y también permiten el control de la actividad del sensor.

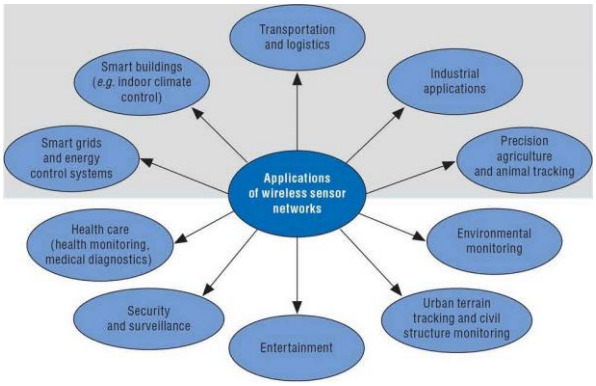

Figura 3. Aplicaciones típicas red WSN

Una de las cosas que caracterizan a estas redes es su facilidad de despliegue y el hecho de ser auto-configurables, pudiendo convertirse en todo momento en emisor, receptor y poder enviar datos entre nodos sin visión directa. Otra característica importante de estos nodos es su gestión eficiente de la energía, que les permite obtener alta tasa de autonomía. Estos sensores tienen los atributos de ser “pequeños”, “baratos” y “autónomos”.

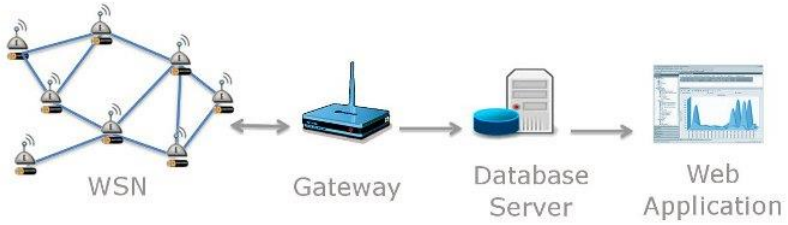

Figura 4. Ejemplo aplicación red WSN

Las redes WSN están típicamente organizadas en tres tipos de topologías de red. En primer lugar la topología de estrella, en la que cada nodo se conecta directamente al Gateway. En segundo lugar, la topología en árbol, en la que cada nodo se conecta a un nodo de mayor jerarquía en el árbol y después al Gateway. Y en tercer lugar, encontramos las redes tipo malla, la principal característica de esta topología es que los nodos se pueden conectar a múltiples nodos en el sistema y pasar los datos por el camino disponible de mayor confiabilidad.

En la red encontraremos un FDD “dispositivo de funcionalidad completa” que puede funcionar como coordinador de una red de área personal o como un nodo normal.

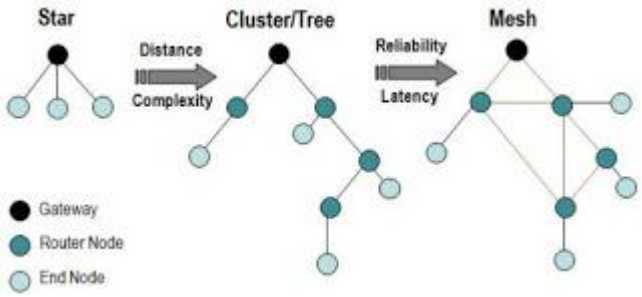

Figura 5. Topologías de Red WSN

1.4 Arquitectura y Protocolos

La arquitectura usada en nuestro proyecto se resume en la Figura 6. Consiste en una red de nodos tipo árbol con un nodo master conectado al EDGE Router, que será el encargado de hacer la conversión de protocolos IPv6 a IPv4. Este nodo master nos permitirá hacer la función de enrutador frontera entre nuestra red WSN con Internet a través del protocolo IP.

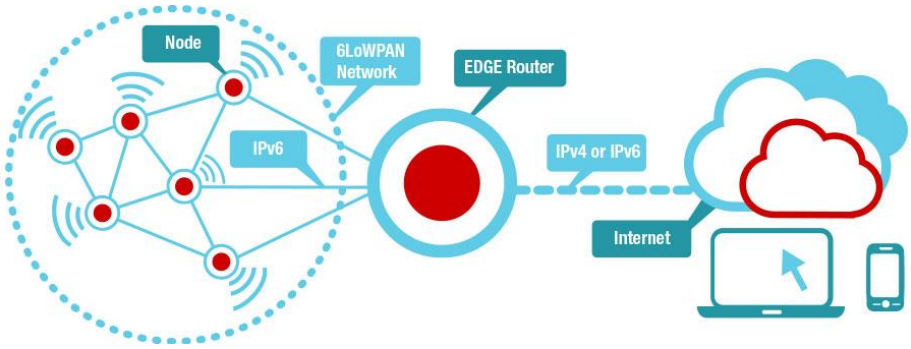

Figura 6. Arquitectura de una red IEEE 802.15.4

1.4.1 IEEE 802.15.4

El IEEE 802.15.4, es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos.
La principal ventaja de IEEE 802.15.4, es la adecuación de uso para tiempo real por medio de slots de tiempo garantizado y evasión de colisiones por CSMA/CA y soporte integrado a las comunicaciones seguras.

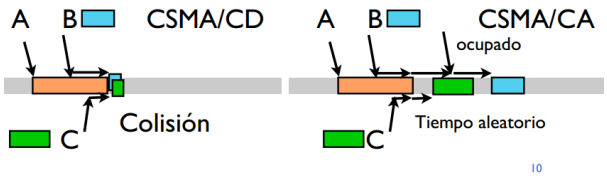

Figura 7. CSMA/CD vs CSMA/CA

Debido a que los equipos inalámbricos son de bajo consumo y nos interesa tener el menor consumo posible, CSMA/CA es más óptimo que CSMA/CD. Con CSMA/CA mantendremos nuestra radio apagada y cuando necesitemos transmitir, encenderemos la radio y veremos si el canal está ocupado. Si el canal no está ocupado, nos pondremos a transmitir. De ese modo, sólo estaremos transmitiendo el tiempo que dure el mensaje.
En la figura 8, se observan las bandas de transmisión que utiliza el IEEE 802.15.4, que son las mismas Bandas que las frecuencias ISM.

PHY (MHz)	Frequency Band (MHz)	Coverage	Data Rate (Kbps)	Number of Channels	RX Sensitivity	Modulation
868/915	868-868.6 902-928	ISM Europe ISM America	20 40	1 10	- 92 dBm - 92 dBm	BPSK BPSK
868/915 (optional)	868-868.6 902-928	ISM Europe ISM America	250 250	1 10	- 85 dBm - 85 dBm	ASK ASK
868/915 (optional)	868-868.6 902-928	ISM Europe ISM America	100 250	1 10	- 85 dBm - 85 dBm	O-QPSK O-QPSK
2450	2400-2483.5	ISM Worldwide	250	16	- 85 dBm	O-QPSK

Figura 8. Bandas de frecuencias ISM[4]

1.4.2 IPv6

Debido al gran número de dispositivos conectados a la red, nos hemos visto obligados a utilizar el protocolo IPv6, para poder conectarlos todos a Internet. El protocolo IPv4, está formado por una dirección IP de 32bits que en IPv6 tiene un tamaño muy superior 128 bits, pudiendo así evitar que estas direcciones se agoten en un futuro.

	Internet Protocol version 4 (IPv4)	Internet Protocol version 6 (IPv6)
Deployed	1981	1999
Address Size	32-bit number	128-bit number
Address Format	Dotted Decimal Notation: 192.149.252.76	Hexadecimal Notation: 3FFE:F200:0234:AB00: 0123:4567:8901:ABCD
Prefix Notation	192.149.0.0/24	3FFE:F200:0234::/48
Number of Addresses	2^{32} = ~4,294,967,296	2^{128} = ~340,282,366, 920,938,463,463,374, 607,431,768,211,456

Figura 9. Diferencias entre IPv4 e IPv6

Por qué IPv6 y no IPv4?

1. El protocolo de Internet necesita una conexión a nivel mundial. Para garantizar eso, IPv4 se ha quedado limitado y por eso debemos optar para la transición a IPv6.
2. Escalabilidad. IPv6 ofrece un esquema de direcciones altamente escalable. Proporciona 2e128 lo que se traduce en cerca de 670 mil billones de direcciones por milímetro cuadrado de la superficie de la Tierra.
3. Eliminar el termino NAT. El mecanismo NAT nos permite intercambiar paquetes entro dos redes que asignan mutuamente direcciones incompatibles. Debido a los límites del espacio de direcciones IPv4, la Internet actual tuvo que adoptar el truco para enfrentar su expansión no planificada. Permite que varios usuarios y dispositivos compartan la misma dirección IP pública.
4. Fuertes habilitadores de seguridad. IPv6 proporciona conectividad de extremo a extremo, con un mecanismo de enrutamiento más distribuido.
5. Pilas pequeñas disponibles. IPv6 ha desarrollado una versión comprimida 6LoWPAN. Es un mecanismo simple y eficiente para acortar el tamaño de la dirección IPv6 para dispositivos restringidos.
6. Permitir la extensión de Internet a la red de cosas. Gracias a su gran espacio de direcciones, IPv6 permite la extensión de Internet a cualquier dispositivo y servicio.
7. Movilidad. IPv6 proporciona características y soluciones sólidas para admitir la movilidad de los nodos finales, así como la movilidad de los nodos de enrutamiento de la red.
8. Dirección de autoconfiguración. IPv6 proporciona un mecanismo de autoconfiguración de direcciones (mecanismo sin estado). Los nodos pueden definir sus direcciones de manera muy autónoma. Esto permite reducir drásticamente el esfuerzo y el costo de la configuración.
9. Totalmente compatible con Internet. IPv6 es totalmente compatible con Internet. En otras palabras, es posible usar una red global para desarrollar su propia red de cosas inteligentes o para interconectar sus propias cosas inteligentes con el resto del mundo.

1.4.3 6LoWPAN

El 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks), es un protocolo diseñado para permitir la transmisión de paquetes IP en redes basadas en el estándar IEEE 802.15.4. Asimismo, está diseñado para aplicaciones con un flujo de datos muy bajo y unos recursos muy limitados.

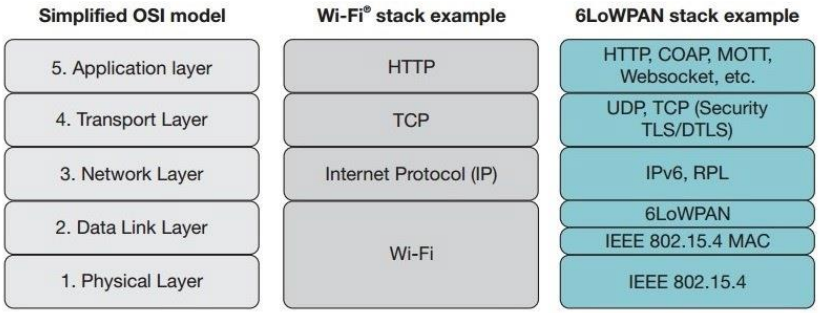

Figura 10. Modelo capa OSI 6LoWPAN

Las principales aportaciones de la capa de adaptación, que permiten la transmisión de paquetes IP en las redes inalámbricas de bajo consumo son:

- Fragmentación de los paquetes: cuando el MTU (Maxium Transfer Unit), de valor 1280 bytes, es inferior al tamaño del paquete, este se divide en varios segmentos más pequeños que son enviados de forma individual y son reconstruidos en el siguiente nodo o en el destino.
- Compresión de las cabeceras: Las cabeceras IP incluyen una compresión, obviando información duplicada, datos que ya contiene el nodo o contenido presente entre otras capas, evitando así la redundancia.
- Generación automática de direcciones IP: incorpora la denominada Stateless auto-configuration, que genera de forma automática la dirección IP de cada miembro de una red 6LoWPAN.
- Avance simultaneo de los niveles de conexión y de red: una vez que se ha decidido el siguiente salto a nivel de la capa de red o IP, la capa de adaptación busca y decide cual es el siguiente nodo a nivel de la capa de conexión (link) al que se tiene que dirigir los datos en su camino hacia el destino final marcado por el IP.

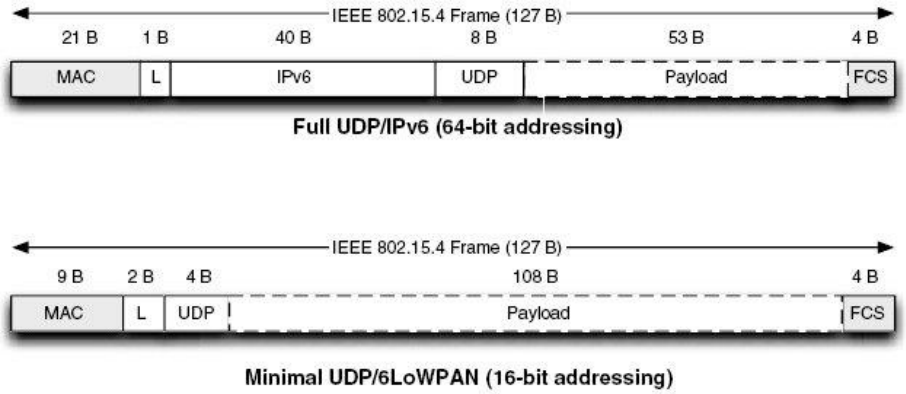

Figura 11. Compresión de la dirección en 6LoWPAN

1.4.4 RPL (Routing protocol for Low power and Lossy Networks)

El RPL, es un protocolo de enrutamiento para redes de sensores inalámbricos de baja potencia y con pérdidas. Es un protocolo basado en vector de

distancias. El router informa a sus vecinos de los cambios en la topología periódicamente.

Existen tres tipos de mensajes:

- DIO: Contiene información que permite a un nodo descubrir una instancia RPL, aprender parámetros de configuración y seleccionar padres DODAG.
- DIS: Solicita a DODAG información sobre el nodo RPL.
- DAO: Se utiliza para propagar la información de destino a lo largo del DODAG. DODAG (Destination-Oriented DAG): Todos los nodos apuntan al nodo Root.

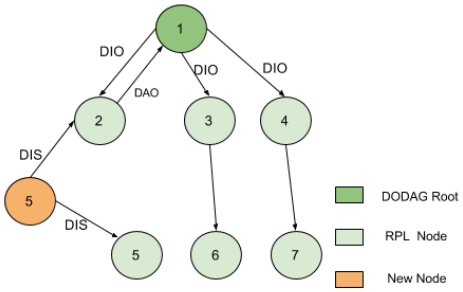

Figura 12. Proceso de paquetes al añadir un mote

1.4.5 UDP/TCP

Los UDP/TCP, son protocolos de nivel de transporte fundamentales para la transmisión de datos.
La diferencia principal de estos protocolos es que el TCP, garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. En cambio el protocolo UDP permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros y tampoco se sabe si han llegado correctamente.
En una aplicación como la medición de la temperatura, que no es una medición critica, en la que podemos perder algún paquete, utilizaremos el protocolo UDP. Principalmente, usaremos UDP porque al no llevar confirmación ni control de flujo, podremos ahorrar muchos paquetes de control y reenvíos de paquetes perdidos, evitando así, un consumo innecesario.

1.4.6 CoAP

El CoAP, es un protocolo de la capa de aplicación de Internet, que está diseñado para dispositivos con recursos restringidos. En concreto, permite que dispositivos con pocos recursos se puedan comunicar con cualquier nodo de Internet. Es ideal para el uso en aplicaciones para IoT.

1.4.7 MQTT

Mosquito o MQTT (Message Queue Telemetry Transport)[5], es un protocolo usado para la comunicación machine-to-machine (M2M) en el IoT. Es un protocolo orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos.

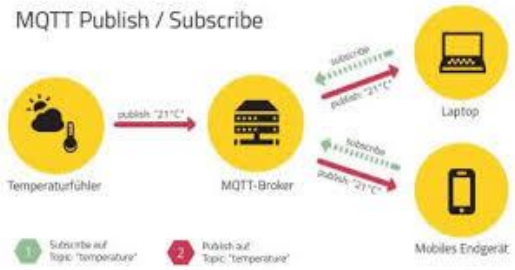

Figura 13. Ejemplo aplicación MQTT

2. DISEÑO E IMPLEMENTACIÓN

Esta sección, consiste en montar una red de cuatro sensores RE-Mote, que de forma autónoma e inalámbrica, nos van a proporcionar información de temperatura en cualquier lugar del mundo a través de Internet.
Debido a que actualmente el protocolo IPv6, no está implementado en Internet en su totalidad, nos vemos obligados a conectar un router frontera, que nos permita la comunicación entre ambos protocolos. Para ello, conectaremos un RE-Mote directamente por USB al router frontera, que utilizaremos de interfaz.

2.1 Plataformas hardware

Actualmente existen diferentes plataformas hardware muy variadas, desde electrodomésticos con sus propias placas hardware, hasta plataformas como Arduino, de código abierto. En este proyecto, utilizaremos para crear la red de sensores, la mota de la marca Zolertia el modelo RE-Mote.

2.2 Sistemas operativos para IoT

Hoy en día, debido a la gran variedad y demanda, existen diferentes sistemas operativos para el IoT.

- **Google Brillo:** Sistema operativo de google, basado en Android y específico para IoT. Para la capa de comunicaciones integra el lenguaje Weave.

- **Tizen:** Sistema operativo móvil basado en Linux a través de la plataforma Linux de Samsung.

- **Zephyr:** Sistema operativo de la organización Linux.

- **Riot:** Es un sistema operativo basado en Linux. Es de código abierto y desarrollado por la Universidad Abierta de Berlín, INRIA y la Universidad de Ciencias aplicada de Hamburgo. Utiliza el estándar de programación en C o C++, se puede desarrollar bajo Linux o Mac OS, ofrece Robustez y flexibilidad, permitiendo la máxima eficiencia energética y es un sistema operativo en tiempo real debido a su ultra-baja latencia de interrupción.

- **Contiki:** Es un sistema operativo de código abierto desarrollado por un equipo mundial de desarrolladores con contribuciones de Atmel, Cisco, ETH, Redwire LLC, SAP, Thingsquare y muchos otros, dirigidos por Adam Dunkels de Thingsquare. Contiki está desarrollado para uso en un número de pequeños sistemas, pasando desde ordenadores de 8-bit a sistemas integrados sobre microcontroladores, incluyendo nodos de redes de sensores. El sistema operativo Contiki soporta diferentes arquitecturas y plataformas hardware. Dispone de muchos ejemplos para

poder empezar a utilizar la plataforma de manera fácil y cómoda. Cuenta con un modo de simulación llamado Cooja en el que se pueden hacer simulaciones sin la necesidad de hardware.

2.2.1 Selección del sistema operativo

Nuestra plataforma Zolertia RE-Mote, actualmente soporta los sistemas operativos de Contiki, RIOT, and OpenWSN.
Para ejecutar el proyecto, hemos seleccionado Contiki, por disponer de un emulador llamado Cooja, que nos permite simular en tiempo real y validar la implementación de la aplicación.
Por otra parte el fabricante dispone de una plataforma Github[6], en la que ha desarrollado proyectos con ejemplos de Contiki, para desarrollar de una manera más fácil nuestro proyecto.

2.3 Arquitectura del sistema

En la Figura 14 podemos observar la arquitectura del sistema. Los sensores se encuentran dentro del rango de cobertura de la red inalámbrica 6LoWPAN. Esta red se conecta a través del border router con un sensor master. El enrutador frontera nos convierte las tramas 6LoWPAN en paquetes IPv4/IPv6. El servidor almacena la información y la envía al servidor de UBIDOTS que se encuentra en Internet a través de IPv4.

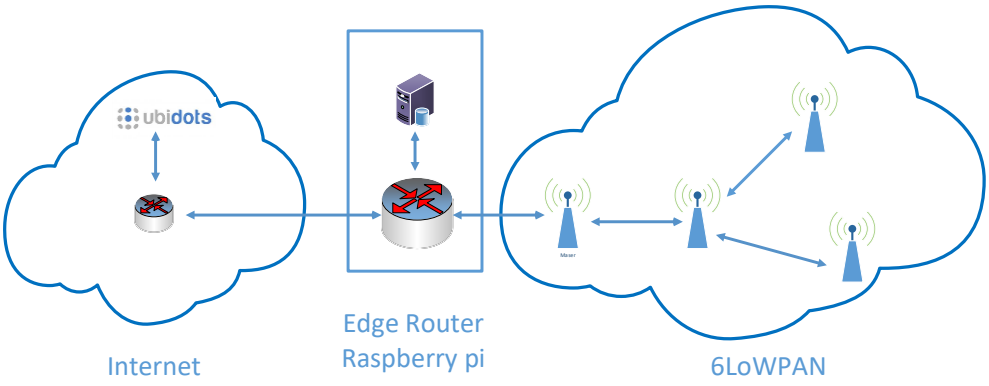

Figura 14. Arquitectura del Sistema

2.4 Arquitectura hardware

En la Figura 15 se muestra la arquitectura hardware del sistema. El sensor de Zolertia RE-Mote, incorpora en su interior una batería de litio recargable, para que pueda ser recargada con una placa solar o por alimentación a través del puerto USB.
El enrutador frontera utiliza un Re-Mote conectado a su puerto USB, que sirve de interfaz de radio IEEE 802.15.4. El enrutador se conecta a Internet a través de la tarjeta Ethernet o vía WIFI.

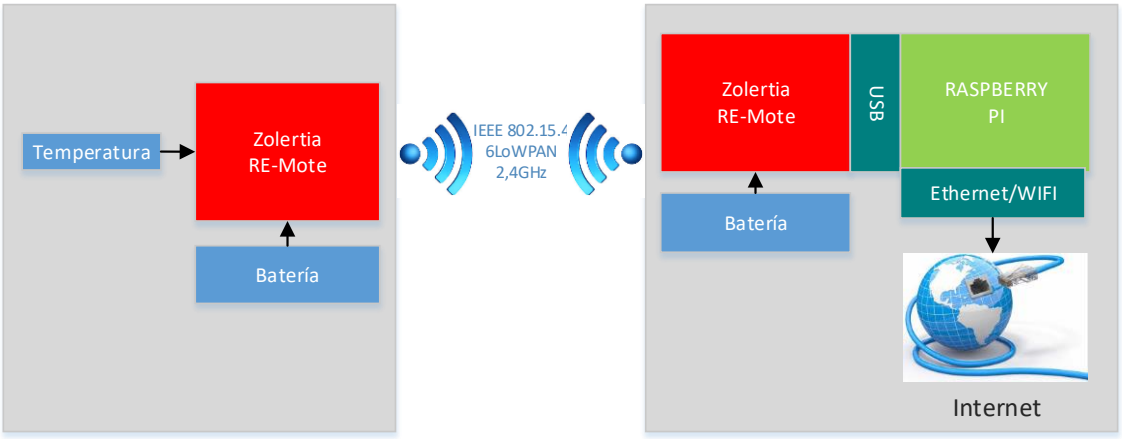

Figura 15. Arquitectura hardware

2.4.1 Zolertia RE-Mote

El módulo Zolertia RE-Mote, está basado en el CC2538 ARM Cortex-M3 de 32Mhz con una memoria flash de 512kb y una memoria flash de 32kb. Además cuenta con una entrada microSD sobre SPI y una batería integrada. También nos permite trabajar sobre un amplio margen entre 2 y 26 VDC.

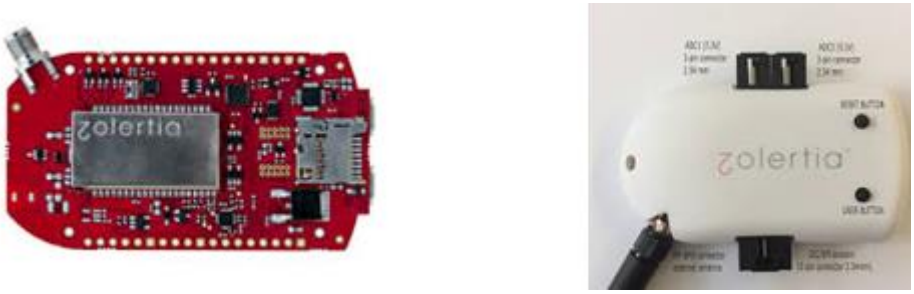

Figura 16. Zolertia RE-Mote

2.4.2 Raspberry Pi Model 3 B

Para realizar el enrutador frontera hemos seleccionado una Raspberry Pi 3[7], que realiza las funciones de enrutador frontera con Internet y almacenamiento de datos.
Cualquier computador nos hubiera servido, pero la Raspberry Pi es ideal, porque tiene un coste y consumo muy reducido y nos permite instalar el sistema operativo Linux.

Figura 17. Raspberry Pi 3 Model B

2.4.3 Sensor de temperatura

Zolertia Re-Mote tiene la opción de conectar sensores analógicos o digitales a través del puerto I2C[8].
En este caso hemos decidido utilizar un sensor analógico muy común, el LM35.
El LM35 es un sensor con una precisión calibrada de 1ºC y su rango abarca de -55ºC hasta 150ºC. Suficiente para la medición de temperatura meteorológica.
Es ideal ja que tiene un consumo de 60 µA.

Conexionado del sensor LM35
Debido a que el sensor opera entre 4v hasta 30v. Sólo podremos conectar el sensor en la entrada ADC3 que opera a 5.1V. La salida del ADC1 nos da una tensión de 3.3V.

Figura 18. Conexionado del sensor LM35

2.5 Enrutador frontera (BR Border Router)

Como hemos comentado antes, el router frontera, nos permite la interconexión entre redes 6LoWPAN a redes IPv4. Actualmente, tenemos varias soluciones para poder implementar el border router.

Una solución posible es comprar el módulo de Zolertia “ORION ROUTER”.

- Módulo Plug and play
- Conectividad perfecta para redes 6LoWPAN e IPv4/IPv6
- Interfaces web del estado y de configuración
- Operación de banda dual 2.4GHz o 868/915MHz

- Ethernet 10BASE-T con POE (Power Over Ethernet)
- Alimentación por USB o POE
- Precio de 189.95 EUR

Figura 19. ORION ROUTER de Zolertia

Por otro lado, también se puede utilizar un ordenador que haga la función de Border Router, con una implementación basada en Contiki y desarrollada por CETIC. CETIC 6LBR conecta dispositivos 6LoWPAN a Internet y es el que se encarga de gestionar el tráfico desde y hacia IPv6/IPv4 a interfaces 802.15.4. Es una plataforma de código abierto e ideal para correr en plataformas como Raspberry Pi, Econotag o la BEAGLEBONE.

Finalmente en este proyecto se ha decidido utilizar el módulo de Raspberry Pi para reducir costes, debido a que éste es más económico que el router ORION.

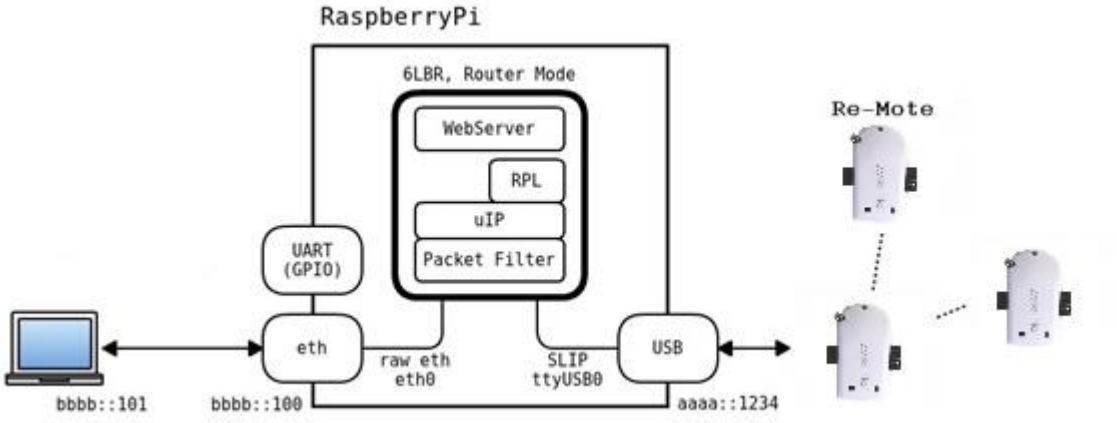

Figura 20. Conexionado de la Raspberry Pi

Como se observa en la Figura 20, el modulo RE-Mote se conecta a través del USB de la Raspberry Pi. El puerto eth se utiliza para la conexión a Internet, en IPv4 o IPv6.

2.5.1 Modos de operación del Border Router

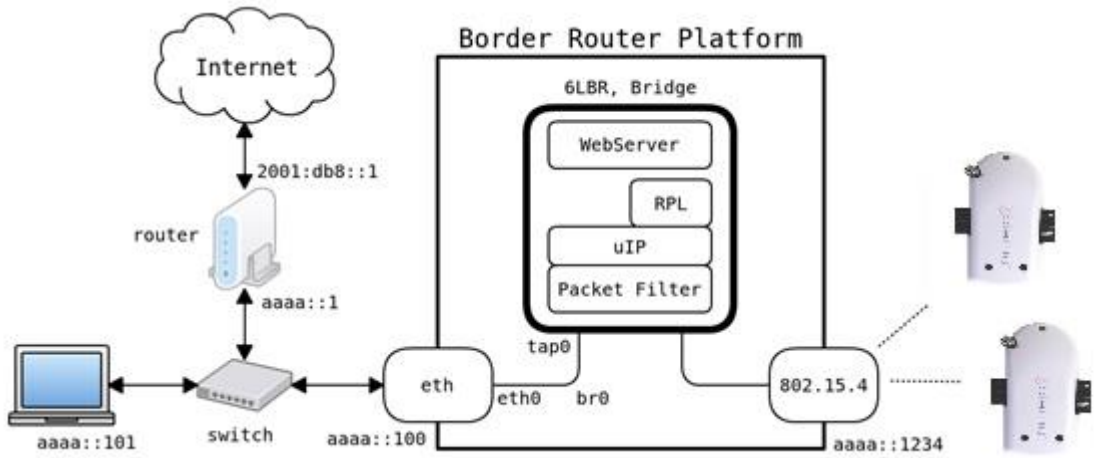

Figura 21. Border router modo Bridge

El BR puede funcionar principalmente en modo Puente o en Modo router. La diferencia principal, es que el modo router nos separa las redes de IPv6 a 6LoWPAN en diferentes subredes.

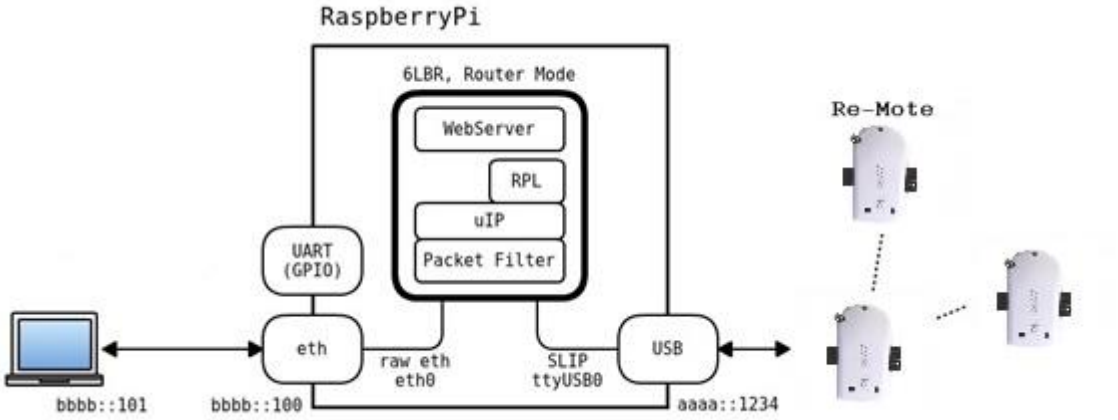

Figura 22. Border router modo Router

En el modo puente, el BR actúa como un puente inteligente que permite interconectar una red basada en IPv6 estándar con una malla WSN basada en RPL. El puente, está actuando como un proxy NDP en el lado Ethernet y está utilizando parámetros NDP para configurar la malla WSN.

Para el presente proyecto se ha decidido utilizar el modo Router, para separar la red de Internet con la red WSN.

2.5.2 Instalación y configuración

El 6LBR se configura sobre el sistema operativo GNU/Linux, en este caso Raspbian basado en Debian.

Una vez instalado[9] el sistema operativo, deberemos conectar las Raspberry Pi a través del puerto Ethernet o Wifi, a la red de Internet.
Seguidamente descargaremos las librerías libncurses y bridge-utils.

```
apt-get install libncurses5-dev bridge-utils
```

A continuación, descargaremos de la plataforma Git-hub la última versión de 6LBR.

```
git clone https://github.com/cetic/6lbr
cd 6lbr
git submodule update --init --recursive
cd examples/6lbr
git checkout develop-xxxxx #optional, if you plan to use a develop snapshot</pre>
```

En caso de no tener git instalado:

```
Sudo apt-get update
Sudo apt-get install git
```

Una vez descargado, compilaremos e instalaremos 6LBR.

```
make all
make plugins
make tools
make install
make plugins-install
update-rc.d 6lbr defaults
```

Una vez instalado, deberemos configurar nuestro 6LBR con el archivo de configuración /etc/6lbr/6lbr.conf

```
MODE=ROUTER
RAW_ETH=0
BRIDGE=1
CREATE_BRIDGE=0
DEV_BRIDGE=br0
DEV_TAP=tap0
DEV_ETH=eth0
RAW_ETH_FCS=0
DEV_RADIO=/dev/ttyUSB0
BAUDRATE=115200
NODE_CONFIG=/etc/6lbr/node_config.conf
```

En este archivo de configuración indicamos principalmente que queremos que nuestro BR haga la función de router y configuramos la velocidad y el puerto USB donde va conectado nuestro nodo.
Por otra parte tendremos que configurar nuestras interfaces de forma manual, para ello entraremos en el archivo de configuración de las interfaces de la Raspberry.

```
sudo nano /etc/network/interfaces
```

Una vez dentro, editamos el archivo de la siguiente manera:

```
auto br0
iface br0 inet dhcp
bridge_ports eth0
bridge_stp off
up echo 0 > /sys/devices/virtual/net/br0/bridge/multicast_snooping
post-up ip link set br0 address `ip link show eth0 | grep ether | awk '{print
$2}'`

auto lo
iface lo inet loopback

iface eth0 inet static
address 0.0.0.0
```

Una vez realizados los canvios, es necesario reiniciar el servicio.

```
service 6lbr restart
```

Finalmente, cuando se ha completado, nuestro BR ya está en funcionamiento. Para comprobar que funciona correctamente entraremos en el navegador de la Raspberry Pi y pondremos la IP de nuestro BR en el navegador.

```
http://[bbbb::100]
```

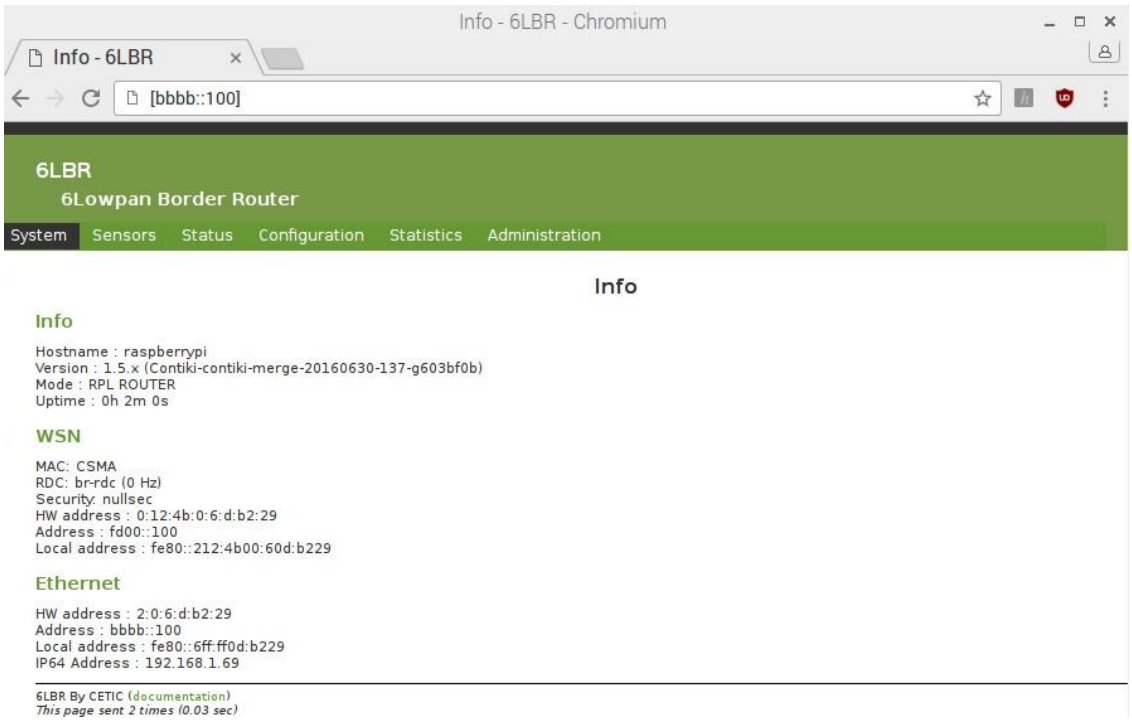

Figura 23. Configurador web del BR

Como podemos observar en la Figura 23, vemos por una parte nuestra red WSN y por otra parte la red IPv6/IPv4.

2.6 MOTA Zolertia RE-MOTE

En la red de nodos, encontraremos dos tipos de nodos diferentes para programar.
El nodo receptor/master, que va conectado directamente al BR y los nodos cliente/repetidores, que se encargan de enviar los datos y a la vez de enrutar los datos hacia el BR.
Los nodos clientes/repetidores leen los datos a través de la entrada analógica y estos las envían a través de UDP y RPL a la Mota que tiene la ruta mejor hacia el BR cada cierto periodo de tiempo.
Para ello, deberemos programar en el cliente cada cuándo queremos enviar el mensaje y hacia quién va dirigido. Una vez programado el nodo enviará periódicamente una trama con la IP de la mota origen y el valor de temperatura.

Ejemplo de nuestra aplicación.

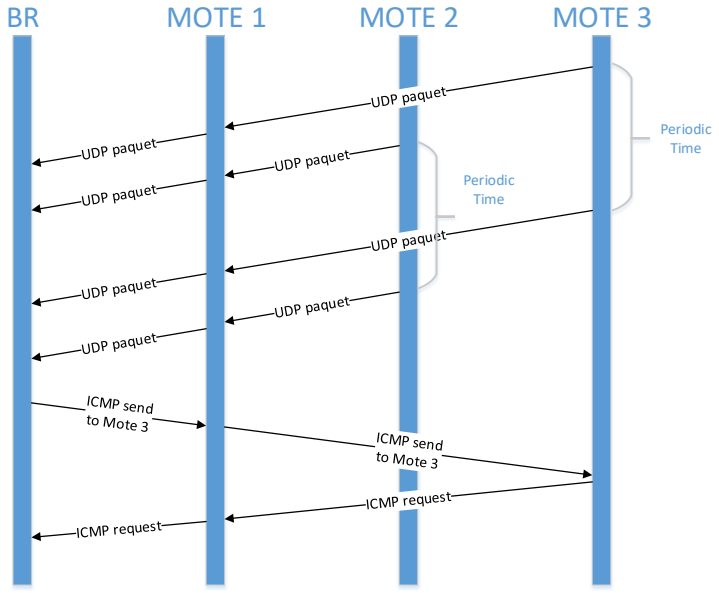

Figura 24. Envío de paquetes

En la Figura 24 observamos que la MOTA 2 y 3 tienen que pasar por la MOTA1 para poder llegar al BR, debido a que no se ven en su radio de cobertura.

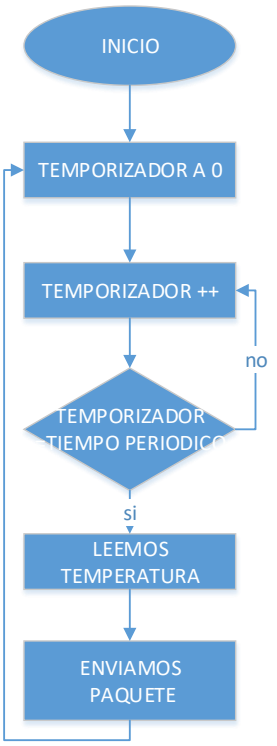

Figura 25. Ciclo de funcionamiento de la mota

2.6.1 Ajustes de parámetros de los sensores

Los sensores están programados para enviar la información cada cierto periodo de tiempo. Podemos indicarle a dichos sensores, cada cuando enviaremos en el #define PERIOD, por ejemplo si ponemos #define PERIOD 60 enviaremos un paquete cada 60 segundos aproximadamente.
Otro parámetro que podemos ajustar es la IP de cada una de las motas. Las motas por defecto ya se programan con una IP con el default prefix seguido de la mac de la mota.
Para poder enviar la información al router deberemos indicarle la IP del mismo. Para ello entraremos vía web al router y en el apartado system dentro la red WSN nos muestra la IP de nuestra mota frontera y en el archivo de configuración de la mota modificaremos en el set_global_address definiremos la IP de la siguiente manera:

uip_ip6addr (&ipaddr, UIP_DS6_DEFAULT_PREFIX, 0, 0, 0, 0x0250, 0xc2ff, 0xfea8, 0xcd1a);

3. PRUEBAS Y RESULTADOS

3.1 Simulación de la red WSN en Cooja

Como se ha comentado anteriormente Contiki dispone de un módulo para la simulación de las Motas. En el módulo podemos simular nuestra red antes de montarla físicamente.
El simulador nos permite ver en tiempo real o modificar la velocidad de simulación para poder ver en detalle, los envíos de paquetes entre motas. A la vez podemos ver la dirección en que se envían los paquetes y el contenido de cada paquete.
El simulador no incluye las motas utilizadas en el proyecto pero se pueden utilizar las motas Z1, que son del mismo fabricante y son muy parecidas.
En la Figura 26 observamos la simulación de un master o, en nuestro caso, el BR que recibe los datos de las tres motas clientes. Los clientes 2 y 3 no están en el radio de cobertura del BR y para llegar al BR necesitan pasar por la mota 1 y enrutar hacia el BR.

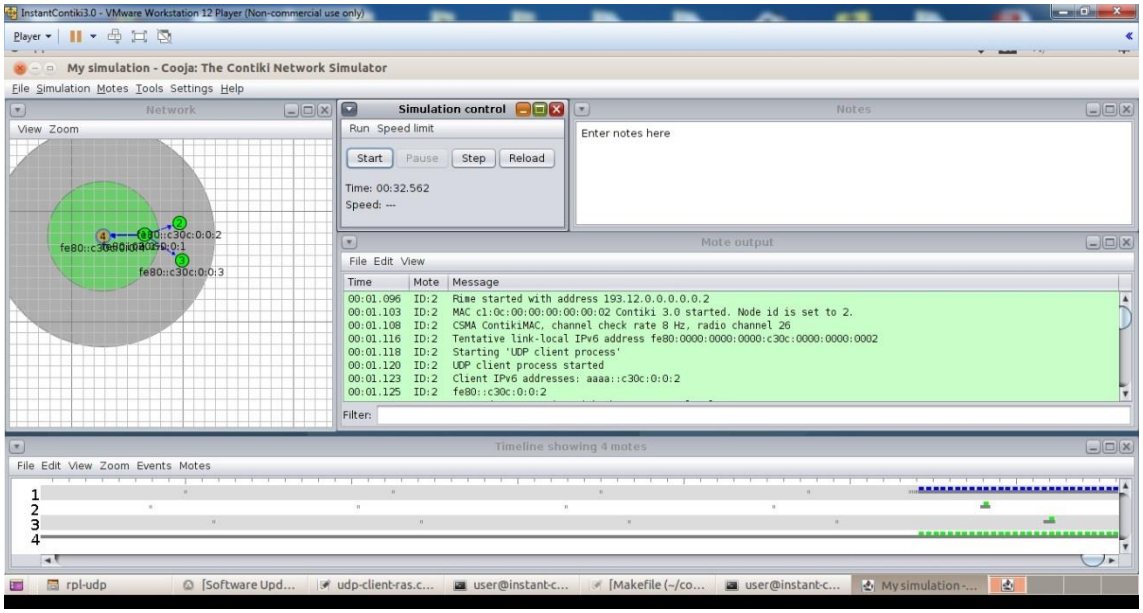

Figura 26. Simulación de las motas en Cooja

Una vez tenemos diseñado el escenario, vamos a montarlos y a realizar las pruebas y comprobaciones.

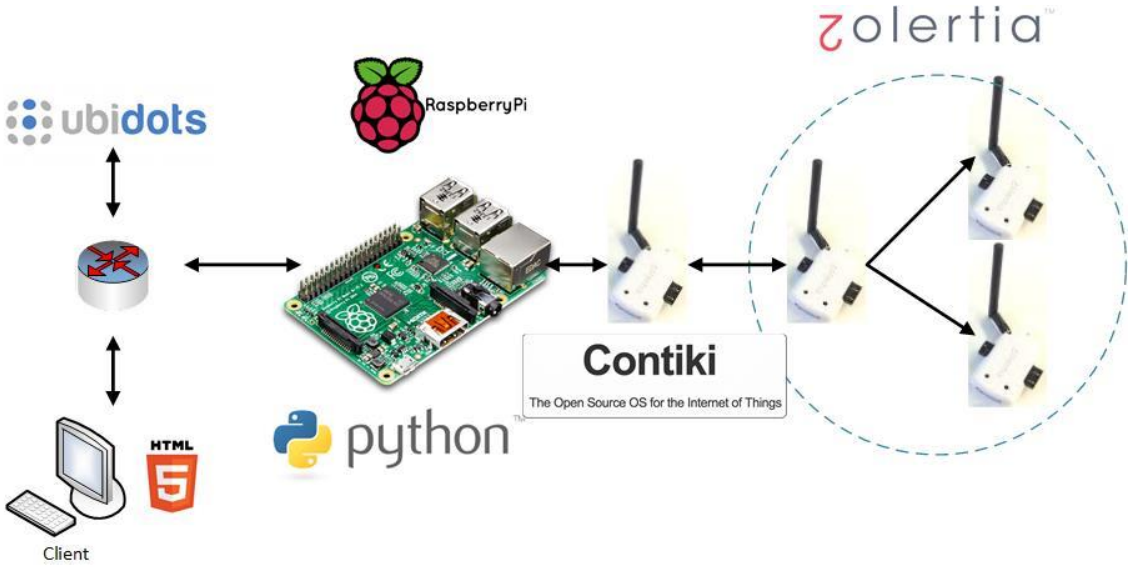

Figura 27. Escenario final del proyecto

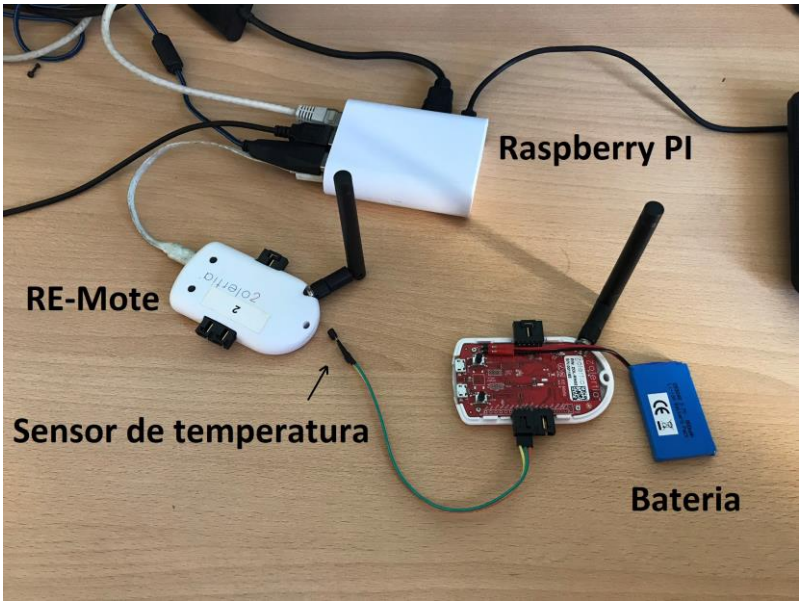

Figura 28. Escenario final

Cuando disponemos del escenario montado, entramos al explorador de la Raspberry Pi para comprobar que las motas están conectadas al Master. En el explorador introducimos la IP del router y en la parte de sensores comprobamos los nodos conectados directamente al BR.

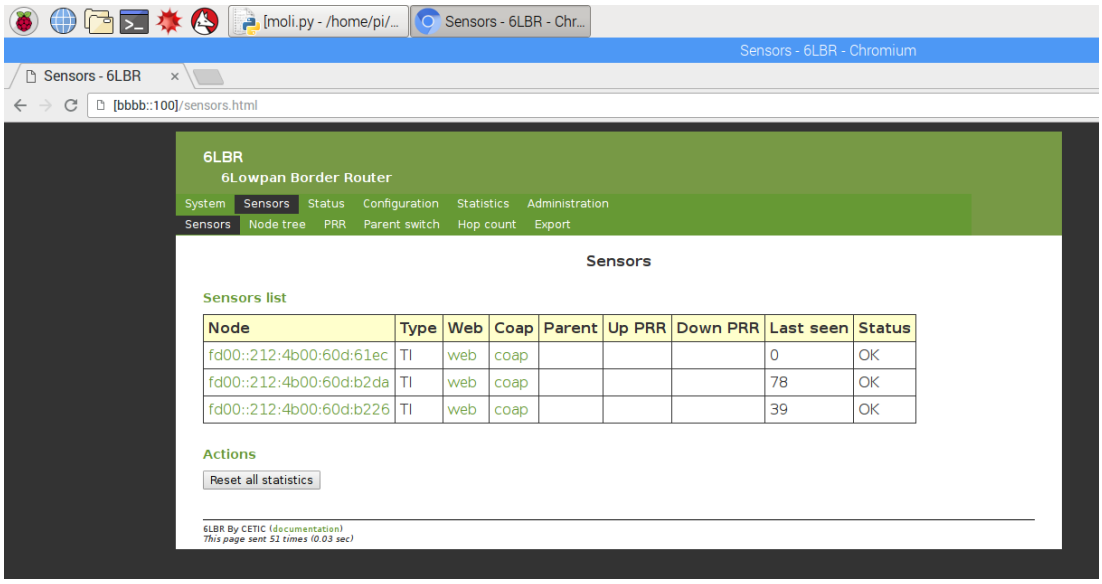

Figura 29. Visualización web de los clientes conectados

En la Figura 29 podemos comprobar como tenemos 3 motas conectadas a la red, su dirección IP y la última vez que ha enviado un paquete.

3.2 Monitorización del cliente

Para poder llevar un seguimiento del cliente y ver los parámetros que envía o los parámetros que deseamos, podemos conectarnos a él directamente a través del puerto USB. Para ello, creamos una conexión serie y que nos permite recibir datos interesantes.

En nuestro caso, la mota nos envía la información de temperatura a través de un valor que nos devuelve el conversor ADC.

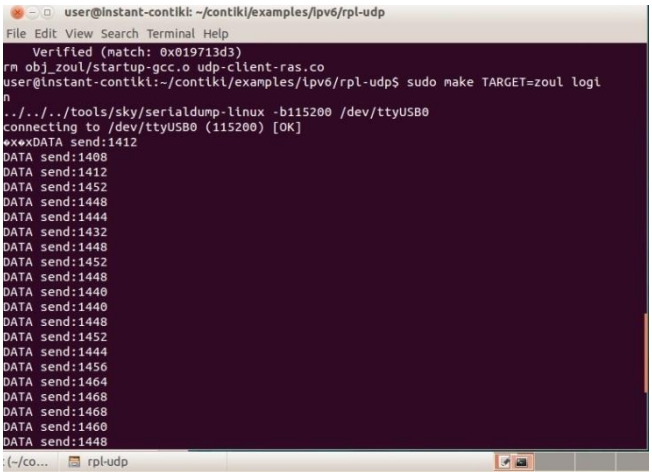

Figura 30. Monitorización del cliente a través de puerto USB

3.3 Capturas de paquetes con Wireshark

Otra opción que tenemos para comprobar que nuestro sistema funciona bien y que recibimos correctamente los paquetes es comprobarlo con Wireshark[10]. En la Figura 30 comprobamos los paquetes UDP que los clientes envían al master.

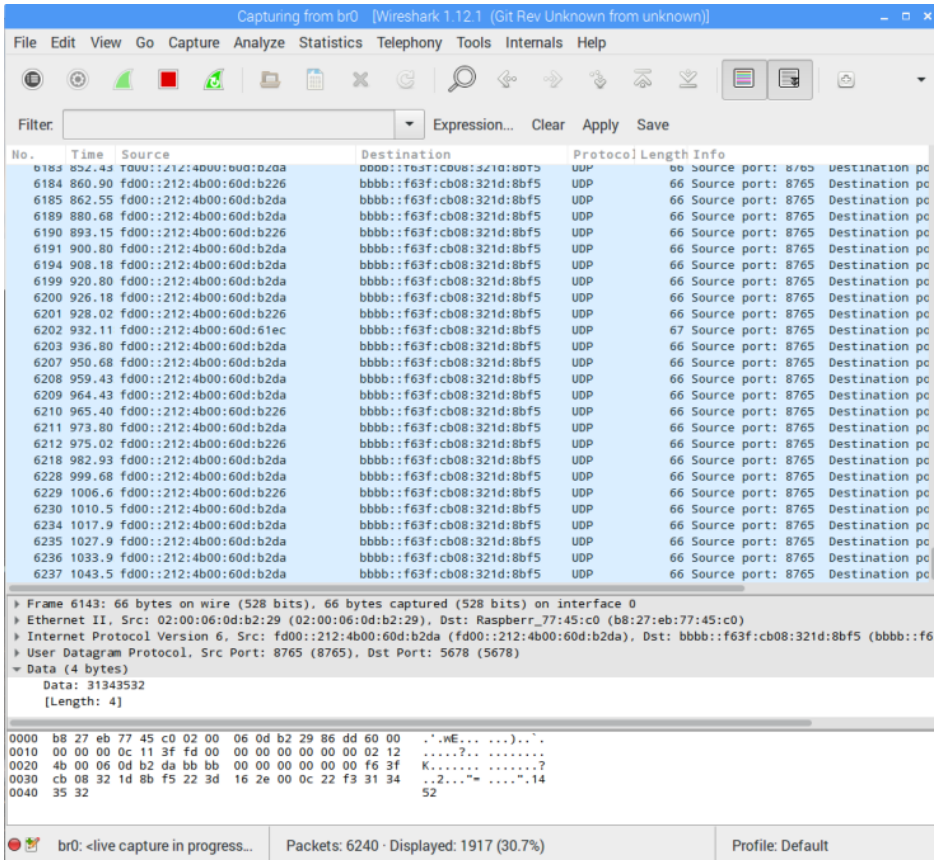

Figura 31. Captura de paquetes en Wireshark

3.4 Envío de datos a Ubidots a través de Python

Figura 32. Logo de Ubidots

Ubidots es una solución para las IOT, nos facilita el almacenamiento y la visualización de los datos obtenidos de los sensores. Para ello deberemos estar conectados al servidor ubicado en Internet.
Los datos a Ubidots, se envían a través de la conexión de Internet que tiene la Raspberry Pi y con un software programado en Python que permite enviar de forma automática los datos al servidor Ubidots.
En caso de perder la comunicación con Internet los datos son almacenados paralelamente en un documento de texto en la Raspberry Pi.

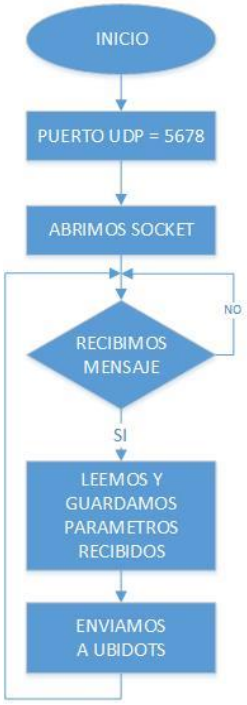

Figura 33. Proceso de envío de datos a Ubidots

3.5 Configuración de Python

Para poder recibir la información y luego procesarla para enviarla a Ubidots, necesitaremos abrir un socket en Python. Para ello deberemos indicarle la IP del BR y el puerto que queremos escuchar por defecto 5678.

```
s.bind(('bbbb::212:4b00:60d:b229', port))
```

Una vez configurados los valores de conexión, realizamos la conexión con Ubidots importando las librerías.
Cuando ya estén registrados en la página web de Ubidots, ésta nos proporcionara un código personal o token que utilizaremos para poder identificarnos.

```
api=ApiClient(token='XJI6MMGM9JlWbfPRg5JY2zVKSruDgn')
```

Posteriormente, al confirmar que se ha realizado la conexión, tendremos que definir el número de variables y el código que nos proporciona Ubidots.

```
my_variable3=api.get_variable('588ce90876254261e442079c')
my_variable2=api.get_variable('58b687ac7625421d5092c543')
my_variable1=api.get_variable('58ce66d8762542736913f03e')
```

Finalmente, una vez definidos estos parámetros, ya podemos arrancar a través de consola de Linux el servidor a Ubidots.

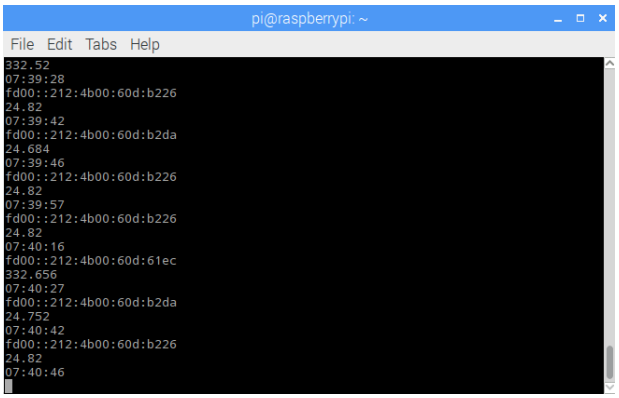

Figura 34. Consola de Python

En la Figura 34 se muestra la información de los sensores a través de la web Ubidots. Podemos observar la IP de la mota, el valor de temperatura y la hora de envío.

Cuando se arranca Python podemos ir directamente a la web de Ubidots para ver en tiempo real la información de nuestras motas.

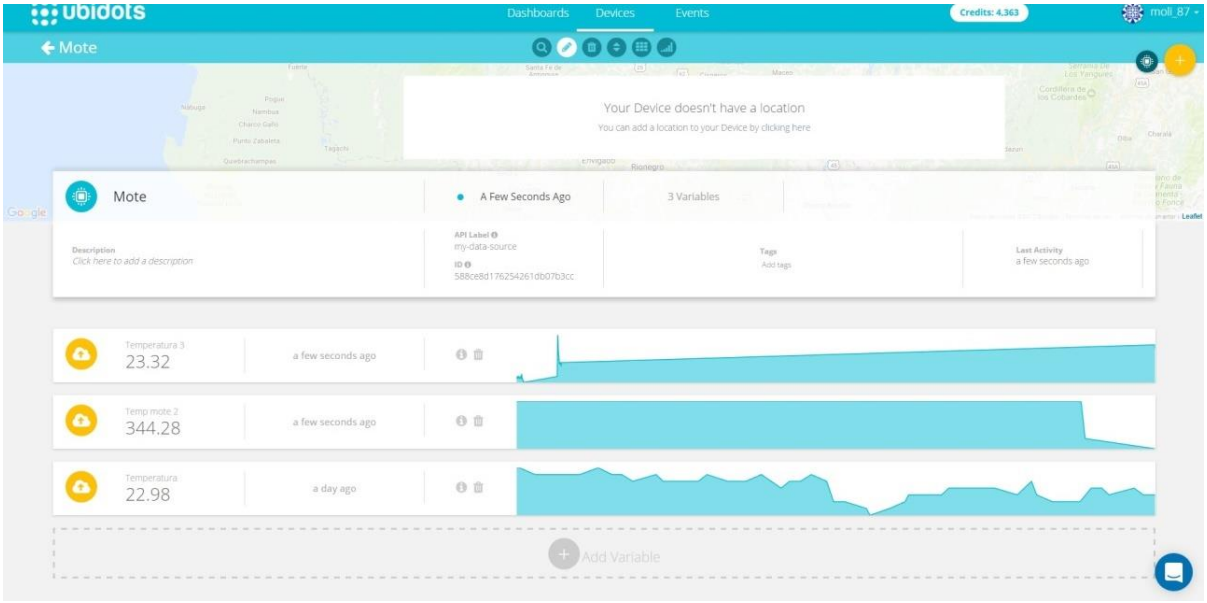

Figura 35. Página web de Ubidots

Si entramos dentro de cada sensor podemos visualizar la gráfica en tiempo real, hacer una media y visualizar los máximos y mínimos.

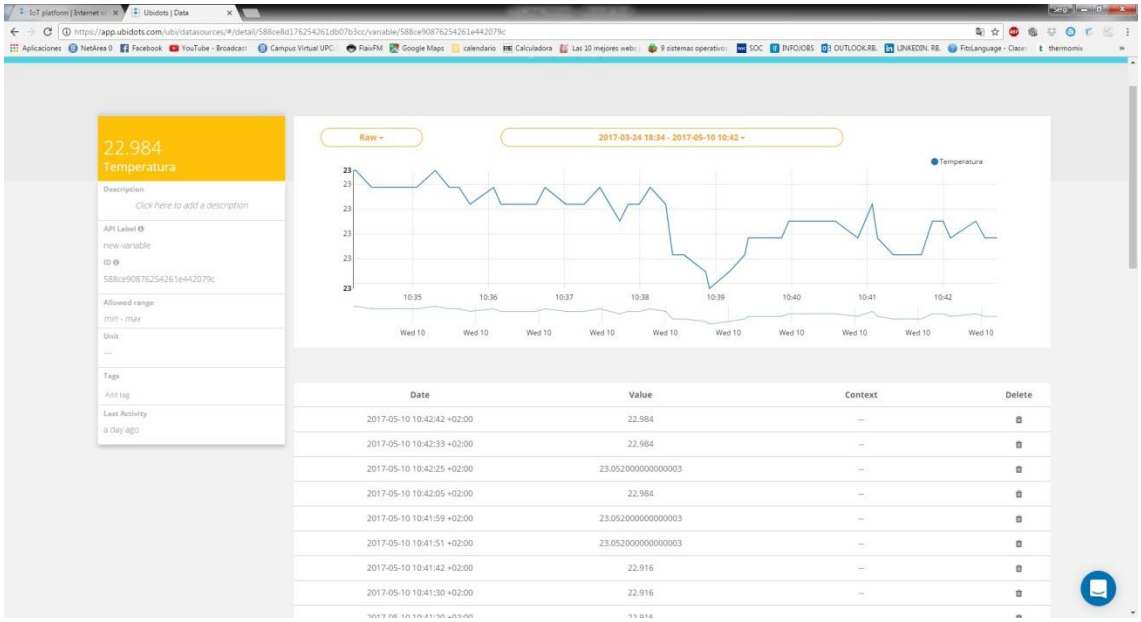

Figura 36. Grafica de temperatura en Ubidots

3.6 Consumo de energía

Como hemos podido observar anteriormente, una de las ventajas más importantes de estas motas es su bajo consumo, pudiendo ser autoabastecidos con energías renovables.
Para evaluar el consumo de energía, introducimos una resistencia Shunt entre la fuente de alimentación y la mota, como se muestra en la Figura 37.
Como tenemos un consumo muy pequeño, la resistencia de la mota es muy elevada. Poniendo una resistencia de 10 Ohms, la mota no sufrirá una caída de tensión que le pueda afectar.

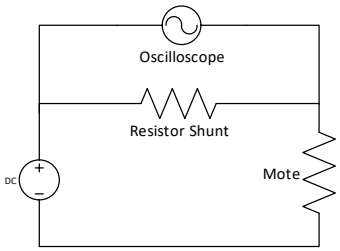

Figura 37. Esquema para medir el consumo

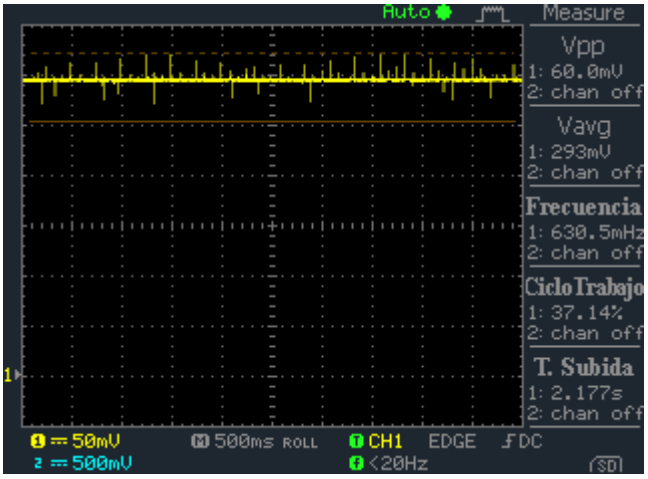

Figura 38. Caída de tensión resistencia Shunt con RDC desactivado

En la Figura 38 podemos observar la caída de tensión que tiene la resistencia Shunt. Aplicando la ley de Ohm, podemos calcular el consumo en mA.

$$I = \frac{V}{R} = \frac{293mV}{10\Omega} = 29{,}3mA$$

Como podemos observar es un consumo muy pequeño. Pero aun así, es un consumo muy elevado para poder utilizarlo ininterrumpidamente durante días.
La medida anterior, se ha hecho con el modo que viene en Contiki por defecto. Éste, es el modo RDC activado. Y lo que hace, es utilizar la radio todo el día encendida, incluso cuando la mota no transmite. Para mejorar el consumo desactivaremos este modo, de manera que la mota sólo transmite cuando tiene que modificar la red o enviar información.

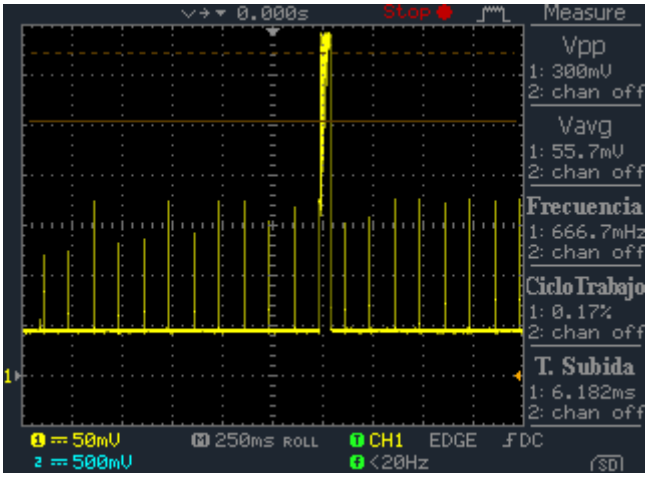

Figura 39. Caída de tensión de la resistencia Shunt con RDC activado

Cálculo del consumo en mA con RDC activado:

$$I = \frac{V}{R} = \frac{55.7mV}{10\Omega} = 5.57mA$$

Como podemos observar en la Figura 39, haciendo el cálculo de la intensidad que consume la mota, se puede ver como ha disminuido considerablemente respecto al RDC desactivado. El pico que se observa en medio de la gráfica, es del momento en el que se enciende el transmisor para enviar información.

3.7 Distancia máxima entre motas

Para calcular la distancia máxima entre motas, hemos salido a la calle y hemos comprobado cuando se pierde la conexión.
Esta prueba, la hemos realizado situándonos en una zona sin edificios, que nos permite tener una visibilidad directa entre motas. Una vez configuradas las motas, a través del ordenador, hemos realizado pings continuados, a la mota esclava, alejándonos progresivamente, hasta perder la conexión entre ellos. Cuando se ha perdido la comunicación, a través de una aplicación móvil “MyGPSCoordinates” para iPhone, hemos identificado las coordenadas GPS del master y del esclavo.

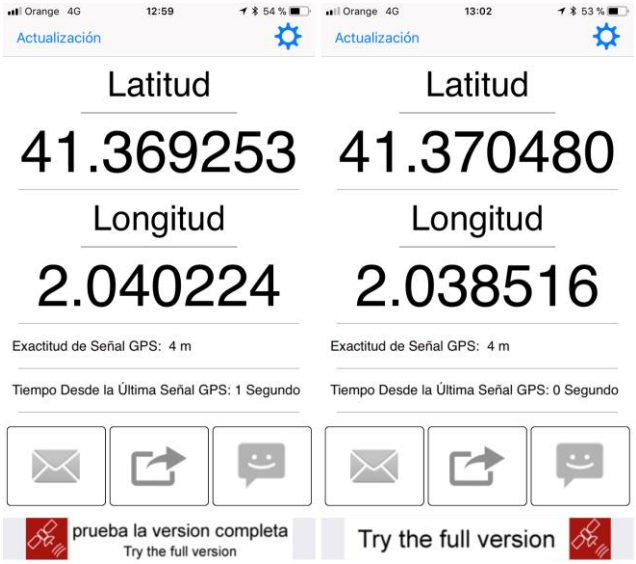

Figura 40. Coordenadas GPS de las motas

Con las coordenadas GPS, y con la finalidad de identificar la distancia entre motas, podemos utilizar calculadoras de distancia GPS, accesibles en Internet, o realizar el cálculo desde Google Earth[11].

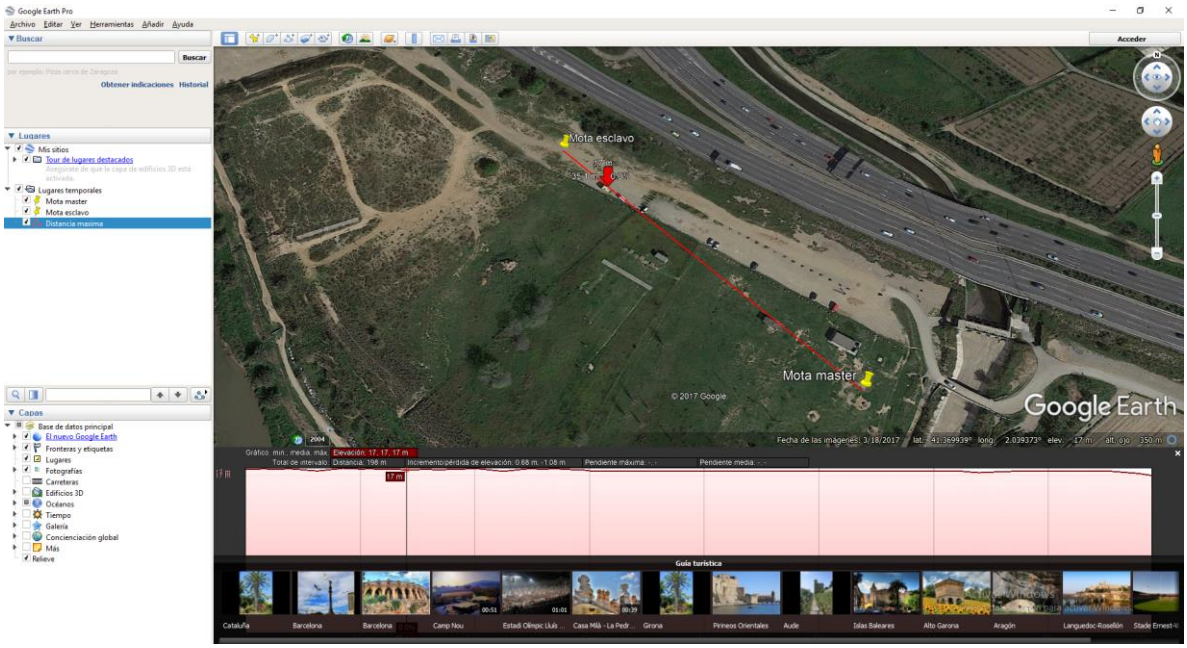

Figura 41. Google Earth cálculo de la distancia entre motas.

Una vez puesta las coordenadas, observamos que la distancia máxima entre motas es un poco menos de 200m.
Cuando se realiza la medición mientras nos alejamos progresivamente, podemos ver como la mota pierde la conexión al momento. Sin embargo, al volvernos a acercar recuperamos la conexión con el master automáticamente.
Esto nos permite conectar motas a la red automáticamente sin configurar nada, sólo necesitamos estar dentro del alcance de la red y ellas automáticamente se enlazan con la red WSN.

4. CONCLUSIONES

Una vez más, podemos ver como IoT nos ofrece un mundo muy amplio de posibilidades, para poder hacer que nuestro día a día sea mucho más fácil.

Probablemente, en un futuro no muy lejano, podremos observar en nuestro entorno, mayor presencia de sensores y dispositivos. Éstos, a través de centros de supervisión o directamente mediante teléfonos móviles nos permitirán visualizar y gestionar todo tipo de objetos remotamente.

Asimismo, podemos observar, que en el mercado, ya existe una gran variedad de plataformas hardware. Éstas, pueden ser, placas y protocolos propios de empresas privadas o plataformas de open source (como el caso de la marca Zolertia).

En concreto, Zolertia Re-Mote, la placa objeto de estudio en este proyecto, es una placa hardware de tamaño relativamente pequeño y código abierto, que podemos adaptar a muchas necesidades. Supone una gran ventaja el hecho de que Zolertia Re-Mote sea una placa de código abierto, nos permite encontrar muchas aplicaciones que hacen más sencilla su programación.

Por otro lado, Zolertia Re-Mote, abarca desde la creación de aplicaciones muy básicas hasta el desarrollo de aplicaciones de gran complejidad. La creación de una red de sensores de temperatura, representa un ejemplo de aplicación básica. Mientras que, por ejemplo, el desarrollo de una red de cultivo que combina factores como el riego en función de la humedad de la tierra, con la finalidad de potenciar el ahorro de agua en periodos de lluvia, mostraría un uso complejo de esta aplicación.

Destacamos como una de las principales ventajas de este sensor, la comunicación inalámbrica y su bajo consumo, el cual, nos permite crear redes de sensores autosuficientes con energías renovables sin la necesidad de ningún tipo de cableado. Otro elemento positivo a destacar, es que, gracias a protocolos como RPL podemos añadir o quitar dispositivos a la red, sin la necesidad de hacer ninguna modificación y de forma automática.

Su elevado coste (100€ aproximadamente por mota) hace que este dispositivo este orientado para aplicaciones profesionales y no tanto para aplicaciones domésticas.

En definitiva, tras haber montado con éxito una red multisensensor inalámbrica, estudiado su funcionamiento, y valorado todos los elementos y características de la placa Zolertia Re-Mote, podemos afirmar que el objetivo de este proyecto se ha alcanzado satisfactoriamente.

5. FUTUROS PROYECTOS

- Comprobar la velocidad de transmisión máxima de las motas, teniendo en cuenta el número de motas.
- Comprobar la velocidad máxima de nodos en función el número de nodos de una red.
- Diseñar un sistema de autoconsumo (con energía renovable) para cada mota.
- Programación con un sistema operativo diferente a Contiki, por ejemplo Riot.

6. REFERENCIAS

[1]Border Router CETIC 6LBR disponible en : https://github.com/cetic/6lbr/wiki

[2]Lenguaje de programación Python disponible en : https://www.python.org/

[3]Aplicación UBIDOTS para Internet of things disponible en : https://ubidots.com/

[4]Banda de frecuencias ISM (Industrial Scientific and Medical), más información en : https://es.wikipedia.org/wiki/Banda_ISM

[5]Protocolo MQTT, más información: https://mosquitto.org/

[6]Plataforma github de Zolertia https://github.com/Zolertia/Resources/wiki/RE-Mote)

[7]Más información sobre Raspberry Pi 3: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[8]Puerto de comunicaciones I2C más información: https://es.wikipedia.org/wiki/I%C2%B2C

[9]Instalación del sistema operativo Raspbian: https://www.raspberrypi.org/downloads/noobs/

[10]Instalación de Wireshark: https://www.enlinux.org/instalar-wireshark-en-gnulinux-debian-ubuntu-server-ubuntu-desktop/

[11]Software google Earth para Windows: https://earth.google.es/