POSICIONAMIENTO INDOOR CON 6LOWPAN
INDOOR POSITIONING WITH 6LOWPAN

TRABAJO FIN DE MÁSTER
CURSO 2019-2020

AUTOR
JOSE MARÍA SEGOVIANO GARCÍA

DIRECTORES
JOSE IGNACIO GÓMEZ PÉREZ, FRANCISCO D. IGUAL PEÑA

CONVOCATORIA: JUNIO 2020
CALIFICACIÓN: 9.5 (SOBRESALIENTE)

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

AGRADECIMIENTOS

Me gustaría agradecer el apoyo incondicional que he obtenido de todas las personas que me rodean.

En especial, a mi mujer Nuria por la comprensión y su incansable apoyo en los momentos más difíciles, sin ella no hubiera sido posible.

A mi familia por los ánimos, el interés y entender mi ausencia.

A mis compañeros de trabajo por aguantar el mono-tema durante este año.

Y por supuesto a los directores de este TFM, Fran y Nacho, por la confianza, las valiosas contribuciones y sus ánimos.

RESUMEN

Posicionamiento Indoor con 6LoWPAN

El posicionamiento *indoor* no es un tema nuevo dentro del mundo de las tecnologías de la información. Sin embargo, existen múltiples formas de comunicación que intentan aprovechar sus particulares bondades para potenciar esta funcionalidad dependiendo del ecosistema donde se quiera utilizar. Por ejemplo, no se requiere la misma precisión si queremos guiar a una persona con movilidad reducida, o si nuestro objetivo es descubrir patrones de movimiento de personas en un centro comercial. Además, el consumo de batería en los diferentes dispositivos tampoco se comportaría de igual forma ni requeriría la misma atención en función del escenario de aplicación.

Este trabajo se centra en un caso de uso muy habitual dentro del mundo industrial, que es el *tracking* de activos en una planta de manufactura.

La tecnología 6LoWPAN, junto con el protocolo RPL diseñado específicamente para redes de bajo consumo de energía, nos presenta una oportunidad muy interesante para el estudio de este caso de uso.

En este trabajo se realiza un estudio del funcionamiento del protocolo RPL que implementa el sistema operativo Contiki, con el fin de modificarlo para calcular las coordenadas de un nodo en función de sus nodos vecinos de los cuales se conocen sus coordenadas y su RSSI (*Received Signal Strentgh Indication*). Para ello, se utiliza el simulador Cooja como base para realizar las pruebas de la implementación y estudiar el comportamiento de la red en cuanto a los tiempos de convergencia de esta.

Palabras clave

Posicionamiento, Indoor, 6LoWPAN, RPL, Cooja, Contiki, MQTT, Thingsboard.

ABSTRACT

Indoor Positioning with 6LoWPAN

Indoor positioning is not a new issue in the world of information technology. However, there are many forms of communication that try to take advantage of its particular benefits to enhance this functionality depending on the ecosystem where it will be used. For example, the same accuracy is not required if we want to guide a person with reduced mobility, or if our objective is to discover movement patterns for people in a shopping centre. Besides, the battery consumption in the different devices would not be the same either.

This paper focuses on a very common case in the industrial world, which is the tracking assets in a manufacturing plant.

The 6LoWPAN technology together with the RPL protocol designed specifically for low energy consumption networks, give us a very interesting opportunity to study this use case.

In this project, a study of the functioning of the RPL protocol implemented by the Contiki operating system is carried out in order to modify it to calculate the coordinates of a node based on neighbors' nodes whose coordinates and RSSI (Received Signal Strentgh Indication) are known.

The Cooja simulator is used as a basis to test the implementation and study the behavior of the network in terms of convergence times.

Keywords

Positioning, Indoor, 6LoWPAN, RPL, Cooja, Contiki, MQTT, Thingsboard.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Motivación

El mundo industrial está experimentando desde hace algunos años una transformación digital tremendamente importante. Esta transformación se conoce como la Industria 4.0 y dentro de este concepto podemos encontrar una gran variedad de iniciativas y líneas de trabajo relacionadas con Internet de las Cosas (IoT).

Una de estas iniciativas es el seguimiento de activos y herramientas que se utilizan en una planta de manufactura con el fin de optimizar su uso y mantenimiento en las órdenes de trabajo asignadas a los operarios. En muchas ocasiones, las herramientas y activos tienen un valor económico muy alto y es necesario conocer su posición y su uso con el fin de reducir su mantenimiento y por consiguiente los costes asociados.

Actualmente, existen varias tecnologías que ofrecen posicionamiento *indoor* en un entorno industrial, cada una de ellas usando diferentes técnicas para el cálculo de la posición.

El hecho de no encontrar demasiados estudios concluyentes a la hora de utilizar 6LoWPAN y RPL como una de las tecnologías en posicionamiento indoor, y sin embargo, con una gran proyección en las redes de bajo consumo de energía, convierte a esta combinación (posicionamiento *indoor* sobre tecnologías de red de bajo consumo) en una línea de investigación con un potencial innegable.

1.2 Objetivos y plan de trabajo

El principal objetivo de este TFM es desarroollar un ejemplo práctico de posicionamiento de nodos-sensores en un entorno industrial con el fin de estudiar el comportamiento de 6LoWPAN y RPL en este caso de uso.

Para conseguir el objetivo principal, será necesario establecer objetivos o hitos intermedios que guíen en el desarrollo de la investigación:

En primer lugar, será necesario estudiar el funcionamiento del protocolo RPL dentro del estándar 6LoWPAN y conseguir recuperar el RSSI (*Received Signal Strentgh Indicator*).

En segundo lugar, se diseñará la forma menos intrusiva de modificar el protocolo RPL para conseguir que los diferentes nodos del sistema sean capaces de enviar la información relativa a la posición sin que afecte a nodos de la red que no necesiten tener este posicionamiento pero que sin embargo formen parte de la red desplegada y, por lo tanto, parte de las comunicaciones.

En tercer lugar, se estudiará un algoritmo de trilateración en función del RSSI, de las coordenadas de 3 nodos vecinos y de la pérdida de potencia recibida según se indica en diferentes estudios [1].

También será necesario diseñar una topología de red que se encargará de exponer la información de las coordenadas de los nodos en un dashboard donde un usuario podrá consultar en tiempo real el posicionamiento de estos. Un diseño conceptual de esta topología puede observarse en la figura 1.1.

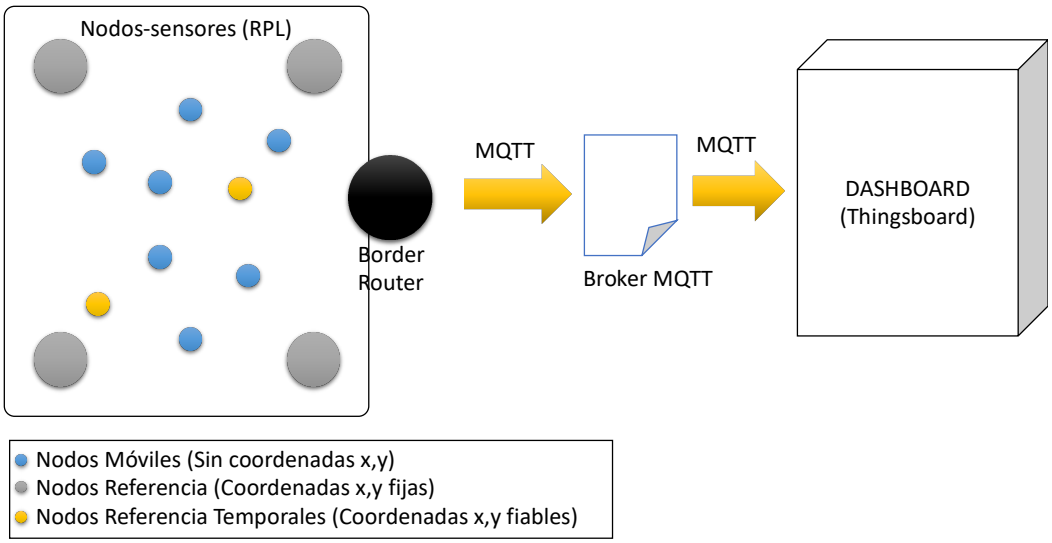


Figura 1.1: Topología de red

Para la exportación de los datos se utilizará un router de borde que se encargará de publicar en un broker MQTT[24] la posición de cada uno de los nodos-sensores. Desde el punto de vista de la visualización de los datos, se desarrollará un *dashboard* con la

información relativa al posicionamiento de los sensores, utilizando para ello la plataforma de IoT *Thingsboard,* que nos proporciona esta funcionalidad.

Para la construcción de la solución, se modificará el código del sistema operativo Contiki, embebiendo la información relativa al posicionamiento en el protocolo RPL, centrándonos de esta manera en la capa de red para el cálculo de posiciones y la convergencia de la red, sin la necesidad de utilizar un protocolo específico a nivel de aplicación

Se diseñarán pruebas con diferentes casuísticas de movimiento de los nodos que componen la red. Estas pruebas se realizarán con la herramienta de simulación Cooja que facilita el sistema operativo Contiki y que simula la comunicación entre los diferentes nodos de la red, facilitando los tiempos de convergencia de la red.

1.3 Organización de la memoria

Este primer capítulo de la memoria realiza una introducción y un enfoque conceptual del caso de uso y de los objetivos que se quieren alcanzar.

En un segundo capítulo, se desarrolla el estado del arte de la tecnología en referencia al posicionamiento en interiores.

El tercer capítulo está destinado al desarrollo de la solución implementada, desde su diseño hasta su implementación.

El diseño y ejecución de las pruebas se desarrolla en el capítulo cuatro de este documento.

Por último, se detallan las conclusiones y los siguientes pasos en el quinto y último capítulo.

Capítulo 2 - Estado del arte

Como se ha comentado anteriormente, existen diversos estudios referentes a las tecnologías usadas para el posicionamiento en interiores. En este capítulo se van a enumerar y valorar el estado del arte de algunas de ellas, realizando una comparativa de las ventajas y desventajas que aportan en los diferentes casos de uso.

Por otro lado, se realizará una descripción referente al seguimiento de activos en la industria y su valor añadido.

Como último punto, se explicará el uso de la tecnología 6LoWPAN en nuestros días con el fin de dar una visión del futuro de esta tecnología.

2.1 Métodos de localización *indoor*

La elección del sistema que mide la localización puede variar dependiendo del caso de uso y los objetivos que se quieren conseguir; algunos de ellos pueden ser la precisión, la seguridad, el tiempo de vida de las baterías, el tiempo real de la información o el coste de implementación. Los sistemas de localización se pueden clasificar en un árbol similar al que se puede observar en la figura 2.1, dependiendo del tipo de cálculo (*positioning*), la variable que se utiliza para el cálculo (*variable*), la técnica de medida de la distancia (*ranging*) y el dispositivo a utilizar (*device*).

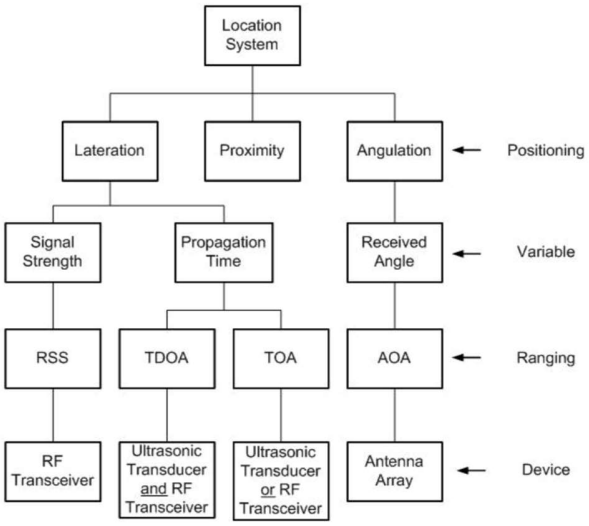


Figura 2.1: Clasificación de sistemas de localización [3].

La estimación por proximidad se considera un método de "grano grueso" y en ella no se calcula una coordenada exacta para el objeto. Sin embargo, estimación por "angulación" y "lateración" se considera de "grano fino" y sí calcula una localización exacta. La diferencia entre ambas es que "angulación" utiliza el ángulo entre el nodo objetivo y los nodos referencia, y "lateración" utiliza la distancia [3].

La técnica para medir la distancia se divide en cuatro tipos [3]:

- *AoA (Angle of Arrival)*. Las antenas miden el ángulo de llegada de la señal trasmitida. Este método puede conseguir una alta precisión en distancias pequeñas; sin embargo, es necesario un hardware especial que consume más energía que otros métodos más sencillos como RSSI. Además, la precisión se deteriora a medida que la distancia aumenta.
- *ToA (Time of Arrival)* o *ToF (Time of Flight)*. Mide el tiempo de llegada de la señal.
- *TDoA (Time Difference of Arrival)*. Es un método evolucionado de ToA que evita problemas de sincronización y paquetes perdidos.
- *RSS (Recived Signal Strengh)*. Mide la potencia de radio recibida en la comunicación. Es fácil de implementar y está muy extendido. Esta técnica será la utilizada en este TFM.

2.1.1 Triangulación

El método de triangulación se basa en fórmulas trigonométricas para el cálculo de las coordenadas (véase figura 2.2), donde se necesitan tres nodos referencia, un nodo objetivo y el ángulo de incidencia entre ellos.

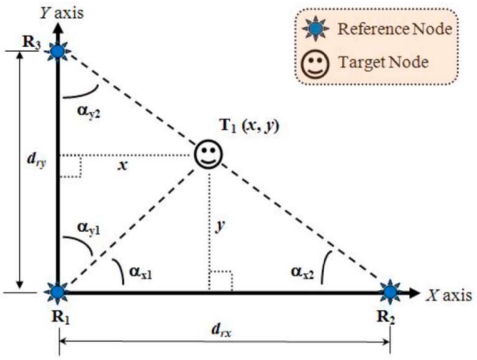


Figura 2.2: Método de triangulación [3].

2.1.2 Trilateración

A diferencia del método de triangulación, este método utiliza la distancia que existe entre el nodo que emite la señal y el nodo que la recibe para el cálculo de coordenadas.

El nodo objetivo, es decir, el nodo del cual queremos calcular las coordenadas, se encuentra en la intersección de las tres circunferencias cuyo radio es la distancia entre el nodo referencia y el nodo objetivo como se muestra en la figura 2.3.

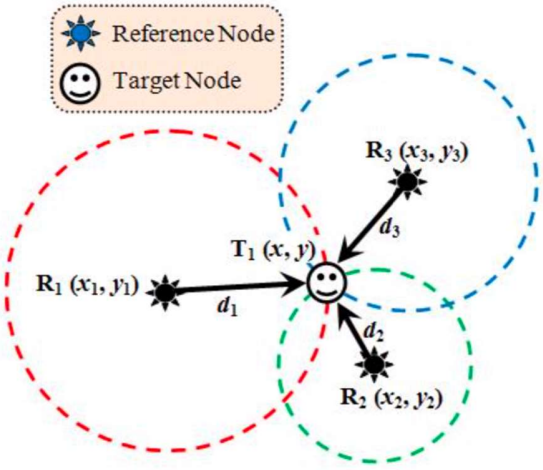


Figura 2.3: Método de Trilateración [3]

2.2 Tecnologías existentes

2.2.1 BLE (Bluetooth Low Energy)

Bluetooth es una tecnología de radiofrecuencia muy madura y extendida en el mercado~~, la cual, .~~ Dd~~e~~sde la versión 1.1 en 2002 ~~recibe~~ premite recibir el RSSI de otro nodo, con lo que resulta relativamente sencillo aplicar una fórmula para realizar trilateración como hemos vista anteriormente.

Las últimas versiones de Bluetooth, conocidas como BLE (Bluetooth Low Energy), pueden transmitir hasta 24Mbps y cubrir hasta 100m con un alto nivel de eficiencia energética[2]. La capa de enlace de BLE no soporta sincronización por tiempo, con lo que resulta imposible utilizar los métodos ToA o TDoA. El método más extendido para realizar posicionamiento indoor con BLE es el uso del RSSI por su simplicidad y porque los beacons (sensores) tienen un coste menor si no monta una antena con ángulo de llegada (AoA).

Antes de la llegada de Bluetooth Mesh en julio de 2017[7], las comunicaciones entre los beacons bluetooth se realizaban de manera unidireccional y en topología estrella[8]. Cuando aparece Bluetooth Mesh se abre la posibilidad de operar las comunicaciones con una red de tipo mesh que acelara la comunicación entre los nodos, potenciando de esta manera su uso como tecnología para el posicionamiento en interiores.

Hoy en día podemos encontrar múltiples productos en el mercado para realizar posicionamiento indoor con BLE. Incluso grandes empresas como Apple (iBeacon) o Google (EddyStone) apuestan por crear protocolos específicos principalmente basados en servicios de proximidad, pero extrapolables a ser utilizados como posicionamiento de interior de activos o personas.

Uno de los inconvenientes que tiene esta tecnología es su dificultad para atravesar obstáculos, lo que hace que sea necesario establecer cuidadosamente la localización de los nodos fijos que darán las posiciones de referencia a los nodos móviles.

2.2.2 UWB (Ultra Wide Band)

Esta tecnología inalámbrica utiliza pulsos cortos transmitidos en periodos de menos de un nanosegundo. Utiliza un rango de frecuencia entre 3.1 y 10.6Ghz [2] y un ciclo bajo de trabajo lo que hace que su consumo de energía sea bajo. Además, utiliza un tipo de señal completamente diferente a otras tecnologías lo que le hace inmune a las interferencias, siendo su principal atractivo, además de su capacidad de penetración en diferentes materiales[2] y su precisión de posicionamiento, en torno a los 30 cm.

Aunque fue creado para redes PAN (Personal Area Networks) ha tenido un desarrollo muy lento y esto hace que sea complicado encontrar dispositivos estándar que utilicen esta tecnología. Sin embargo, en el último año, las grandes compañías empiezan a incorporarlo en sus dispositivos. Por ejemplo, el iPhone 11 lo implementa, aunque todavía no lo usa. También Samsung y Sony apuestan por UWB a corto plazo[4].

2.2.3 Wifi

WiFi (IEEE 802.11 estándar) se utiliza comúnmente para dar servicio a internet a dispositivos privados, públicos y entornos comerciales. Existen multitud de estándares de red que han ido apareciendo a lo largo de los años y cada uno tiene unas características específicas. Por ejemplo, el estándar 802.11ah opera por debajo de 1Ghz lo que puede ser una opción interesante en IoT [10].

Al ser una tecnología muy extendida, existen una gran cantidad de dispositivos comerciales que la incorporan, tales como móviles, ordenadores portátiles, tablets, etc… lo que lo hace un gran candidato para el posicionamiento indoor.

Inicialmente, una red WiFi no está pensada para realizar posicionamiento, con lo que es necesario definir nuevos algoritmos de enrutamiento para optimizar el uso de los métodos conocidos como RSS, ToF y AoA [2].

En cuanto a la precisión, depende mucho de los puntos de acceso que se configuran en la red [9] y del ruido que pueda encontrar la señal, pero en algunos estudios se han llegado a conseguir 23 cm de precisión[11].

Es muy habitual encontrar sinergias entre esta tecnología y UWB, de tal manera que se utiliza UWB como base para la localización y una red mesh WiFi para las comunicaciones [2].

2.2.4 ZigBee

ZigBee opera bajo el estándar IEEE 802.15.4 al igual que BLE o 6LoWPAN y su uso está extendido en redes de tipo PAN (Personal Area Networks).

ZigBee utiliza un protocolo de comunicación propietario [11] lo que hace que tenga una evolución lenta. Sin embargo, existen integraciones con pilas IPv6 [12] utilizando 6LoWPAN y el protocolo RPL que pueden impulsar esta tecnología para diferentes casos de uso, entre ellos posicionamiento indoor.

En algunos estudios encontrados en referencia al posicionamiento indoor con ZigBee[13] se utiliza la métrica RSSI entre pares de nodos. Este valor se almacena en una base de datos que se utiliza para posteriormente utilizar un algoritmo específico para el cálculo de la posición del nodo objetivo.

2.2.5 RFID (Radio Frequency Identification Device)

RFID es una tecnología de radio frecuencia de corto alcance. Su funcionamiento se centra en tener un lector con capacidad para escuchar la frecuencia en la que emiten las etiquetas.

Existen dos tipos básicos de etiquetas:

- Etiqueta activa: Este tipo de etiqueta requiere una fuente de energía para funcionar y se podría utilizar para el posicionamiento indoor aunque la precisión no supera el metro de distancia.
- Etiqueta pasiva: No requiere una fuente de energía y su coste es muy bajo. Esto lo hace una opción muy atractiva para el tracking de activos en la industria. Su desventaja es su limite en el alcance, ya que su rango no supera los 2 metros.

Su uso se centra en la identificación de activos más que en el posicionamiento y seguimiento exhaustivo de los mismos [14].

2.2.6 Luz Visible (VLC)

Es una tecnología emergente enfocada a la transferencia de datos a alta velocidad utilizando diodos de luz para la emisión de luz visible entre 400 y 800 THz [2].

Respecto al posicionamiento indoor, los diodos (LED) actúan como beacons, siendo el sensor el que recibe la señal. Se utiliza la técnica AoA consiguiendo una precisión muy alta. Sin embargo, tiene una limitación importante, ya que no puede existir ningún elemento entre el sensor y el LED para conseguir esta precisión [2].

2.2.7 Señal Acústica

Esta tecnología puede utilizar el micrófono de un Smartphone para recoger la señal acústica que envían los nodos referencia.

Se utiliza la técnica ToF para la estimación de la distancia y según los estudios encontrados [2] se muestra una buena precisión, pero se genera contaminación acústica debido a la limitación de los micrófonos. Sin embargo, emitir sonidos por debajo de la banda acústica audible no sería suficiente para conseguir esta precisión. Además, es necesario un alto coste en infraestructura y energía, lo que hace que esta tecnología no este muy extendida.

2.3 Comparativa de tecnologías

Para poder situar cada una de las tecnologías citadas anteriormente en un contexto válido para su comparativa en el caso de uso que nos ocupa, se van a seleccionar diferentes características clave que influyen en la eficacia del posicionamiento indoor. En concreto nos fijamos en el alcance, la precisión, el consumo de batería, ventajas y desventajas.

	Alcance	Precisión	Consumo de batería	Ventajas	Desventajas
BLE	10m-80m	0,5m – 4m Depende de la distancia de los nodos referencia	Bajo	Bajo consumo de energía y muy extendido en el mercado.	Baja precisión debido a la exposición a interferencias.

WiFi	150m	0,3m – 2,2m Depende del número de APs	Moderado	Muy extendido y fácil de desplegar	Complejos algoritmos para conseguir precisión. Consumo de energía
UWB	10-150m	0,3m	Moderado	Alta precisión. Inmune a las interferencias.	Requiere hardware específico.
ZigBee	40m	0,51m	Bajo	Bajo consumo de energía	Protocolo propietario
RFID	200m Depende de las etiquetas		Bajo	Bajo consumo de energía Muy utilizado para control de activos	Baja precisión para la localización
VLC	1,4km	7cm – 40cm Depende de la configuración de los nodos referencia	Alto	Precisión muy alta	Alto consumo de batería Muy afectado por obstáculos
Señal acústica	2m	Alta	Moderado	Puede tener alta precisión dependiendo del hardware.	Afectado por la contaminación acústica
IEEE802.15.4 + 6LoWPAN	40-100m		Bajo	Bajo consumo de batería	Muy pocas investigaciones al respecto.

Tabla 1: Comparativa tecnologías para localización

2.4 Posicionamiento *indoor* de activos en la industria

El caso de uso pensado para este TFM es un tema muy atractivo desde el punto de vista de la industria. Ser capaces de posicionar y monitorizar activos en una planta de producción puede significar un ahorro de coste y tiempo muy importante para una compañía.

Podemos clasificar el posicionamiento *indoor* en dos tipos:

El primero de ellos es el **posicionamiento por presencia**, es decir, solo se desea averiguar si un activo ha llegado a una determinada zona. En este caso, no será necesario disponer del seguimiento del activo en tiempo real, sino simplemente avisar cuando el activo llega. Algunos casos de uso relativos a este tipo son los siguientes:

- Control de *stock* en un almacén logístico.
- Control de equipaje en un aeropuerto.
- Control de mercancías en grandes centros comerciales.
- Control de visitantes en un hospital.
- Control de equipamiento en hospitales y fábricas.
- Ocupación eficiente de espacio en oficinas.

El segundo tipo de posicionamiento es la **monitorización de los movimientos** que tiene un activo en una determinada área. En este caso, será necesario una actualización periódica de la posición del activo en el sistema, lo que se puede convertir en una necesidad de refresco de la posición que puede llegar a tiempo casi real. Este sería el marco de trabajo de este TFM. Algunos casos de uso que se pueden aplicar a este tipo son los siguientes:

- Gestión de un AGV[25] en una planta de producción.
- Guiado de personas con reducción de movilidad en aeropuertos, estaciones de tren o cualquier espacio interior.
- Patrones de comportamiento de personas en un centro comercial.
- Optimización del uso de herramientas en plantas de producción.
- Posicionamiento de materiales y herramientas en cadenas de montaje.

2.5 6LowPAN

6LoWPAN es un estándar de IPv6 sobre redes para dispositivos con restricciones de potencia (Low Power Wireless Personal Area Networks) que se encarga de adaptar los paquetes IPv6 a los diferentes protocolos que se sitúan en las capas superiores como

se ve en la figura 2.4 (básicamente, capas de transporte y aplicación) [18]. Este hecho supone una gran ventaja en IoT ya que nos permite diseñar una arquitectura con protocolos de aplicación ligeros sin preocuparnos de las comunicaciones entre los dispositivos de una misma red. La definición de este estándar lo podemos encontrar en el RFC6282.

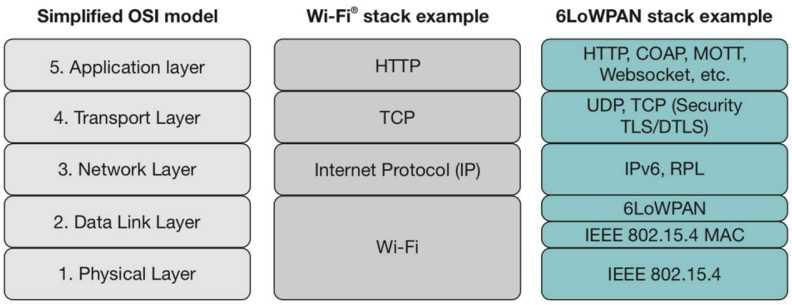


Figura 2.4: Modelo OSI, ejemplo de pila Wi-Fi y pila 6LoWPAN

Actualmente, 6LoWPAN está adaptándose a múltiples tipos de redes incluyendo Sub-1 GHz low-power RF, Bluetooth Smart, PLC (Power Line Control) y low-power Wi-Fi [18]. Debido a la transparencia que provee 6LoWPAN con la capa de aplicación, es muy sencillo el uso de protocolos conocidos, lo que significa que la curva de aprendizaje para su implementación es baja. Además, es una tecnología muy escalable por el uso de IPv6 y requiere un bajo coste de cómputo debido al enrutamiento directo de sus nodos.

Hoy en día, podemos encontrar que el uso más importante de 6LoWPAN es la implementación de Thread [19], protocolo usado para redes *mesh* que se está extendiendo ampliamente [21] gracias al apoyo de algunas grandes compañías (véase Figura 2.5), lo que promete una gran evolución en los próximos años.

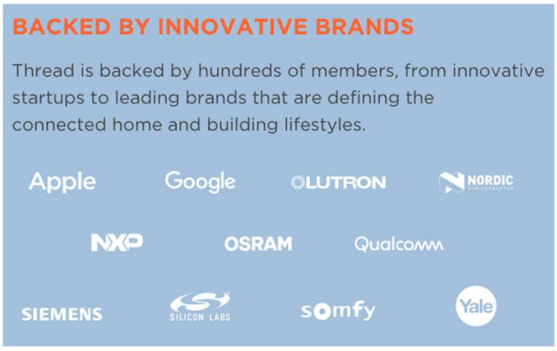


Figura 2.5: Apoyo de grandes compañías sobre Thread [20]

Capítulo 3 - Desarrollo de la solución

En este capítulo se detallan los diferentes pasos que se han seguido en la investigación para el desarrollo de la solución.

En primer lugar, se estudiará el funcionamiento del protocolo RPL para posteriormente diseñar la evolución necesaria a realizar para incluir el envío de las posiciones de los nodos vecinos.

Una vez hecho esto, se realizarán los diseños necesarios para la comunicación entre los nodos de la red de posicionamiento y el exterior donde se enviará la información para ser mostrada en un dashboard visible desde el exterior de la red.

Finalmente, se implementará la solución descrita para sus posteriores pruebas y estudio de resultados.

3.1 Estudio del protocolo RPL

3.1.1 Introducción al protocolo RPL

Dentro del ecosistema IoT, es habitual encontrar restricciones de hardware en los nodos que componen una red. También nos enfrentamos continuamente a redes donde existen tasas muy bajas de transferencia de datos y pérdidas de paquetes entre los nodos. El grupo de trabajo IETF ROLL ha definido requerimientos específicos para estas redes de baja potencia y con pérdidas (LLN), los cuales se detallan en los RFCs RFC5867, RFC5826, RFC5673 y RFC5548. RPL (*Routing Protocol for Low-power and Lossy Networks* (LLN)) fue diseñado específicamente para cubrir los requerimientos de este tipo de redes (LLN).

Es un protocolo de enrutamiento basado en vectores de distancia (distance-vectors) lo que significa que cada nodo de la red conoce la distancia (coste) para llegar a otro nodo, lo que permite el encaminamiento basado en restricciones.

Soporta comunicación multipunto-a-punto, punto-a-multipunto y punto-a-punto y opera sobre IPv6 [5].

Este protocolo se sitúa en la capa de red del modelo OSI (véase Figura 3.1) y organiza la topología de nodos como un Grafo Acíclico Dirigido (DAG) particionado en uno o varios DAGs orientados a destino (*destination-oriented DAG* o DODAG).

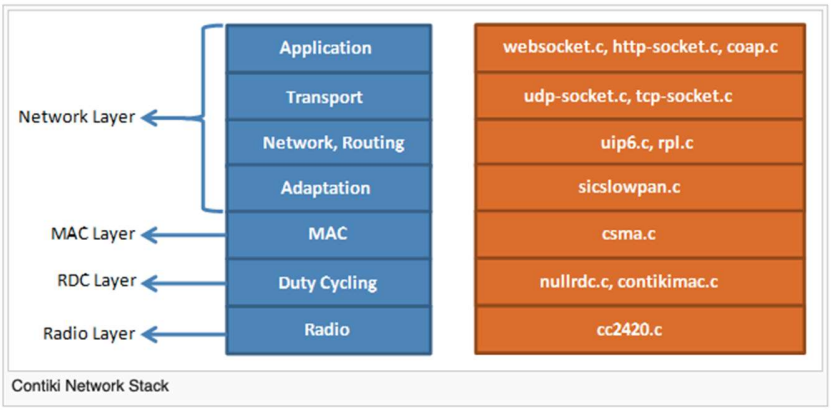


Figura 3.1: Modelo de capas en contiki

Todos los DAGs deben tener una raíz (*root*) que será el punto final de todas las rutas (concepto de DAG *root*). Una *Instancia de RPL* puede contener uno o más DODAGs (véase Figura 3.2).

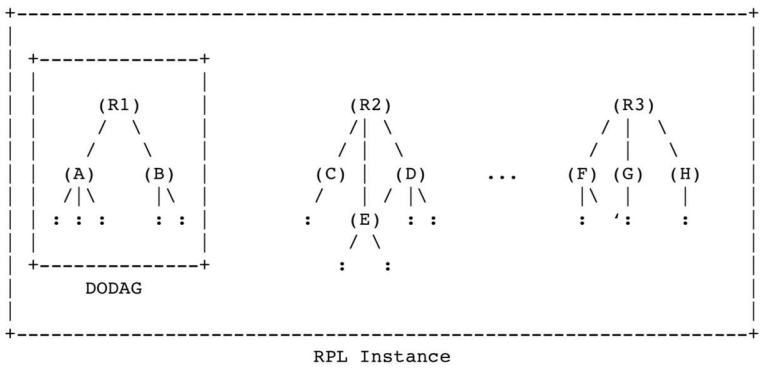


Figura 3.2: Instancia RPL con varios DODAGs

Cada nodo en un DODAG tiene asociado un parámetro "*Rank*" que indica su posición relativa a otros nodos en el mismo DODAG. Este Rank crece en sentido descendiente del DODAG y decrece en sentido ascendente, es decir, hacia el nodo raíz (DODAG Root).

El DODAG se construye en base a una función objetivo (*Objective Function*, OF) que implementa las métricas a utilizar para construir la ruta óptima y calcular el parámetro Rank. Es decir, esta función es la que establece la forma de elegir a los padres

en función de este coste y por lo tanto la encargada de formar el DODAG. Los DODAGs que forman parte de la misma instancia, comparten obligatoriamente la misma función objetivo.

3.1.2 Flujo de información en un DODAG

Un mensaje de control en RPL es un mensaje de tipo ICMPv6, lo que significa que se distribuye como aparece en la figura 3.3. El espacio reservado para el código que define el tipo de mensaje tiene los siguientes valores:

- **0x00 →DODAG Information Solicitation (DIS)**

  Este tipo de mensaje se usa para solicitar información de un DODAG a un nodo vecino, es decir, solicita un mensaje DIO.

- **0x01 → DODAG Information Object (DIO)**

  Transporta información de la instancia RPL del DODAG y permite al nodo conocer los parámetros de configuración para elegir un padre y mantener el DODAG.

- **0x02 → Destination Advertisement Object (DAO)**

  Se usa para propagar la información del DODAG hacia arriba, es decir, hacia el destino.

- **0x03 → DAO-ACK**

  Se usa para informar al emisor de un DAO que ha llegado el mensaje o bien de un posible error.

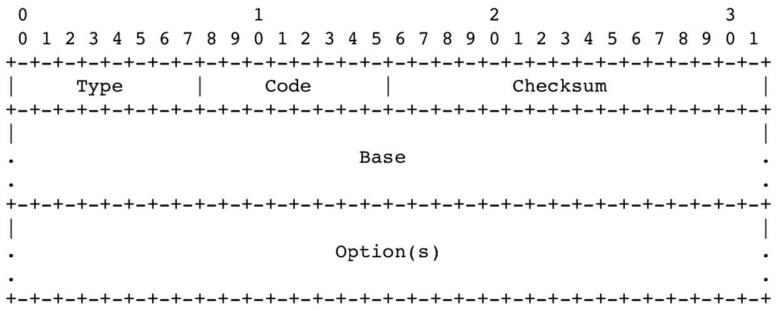


Figura 3.3: Mensaje ICMPv6 [6].

También es posible utilizar una capa de seguridad (véase Figura 3.4), dando lugar a códigos diferentes para los tipos de mensajes:

- 0x80 → Secure DODAG Information Solicitation (SDIS)
- 0x81 → Secure DODAG Information Object (SDIO)
- 0x82 → Secure Destination Advertisement Object (SDAO)
- 0x83 → SDAO-ACK
- 0x84 → Consitency check

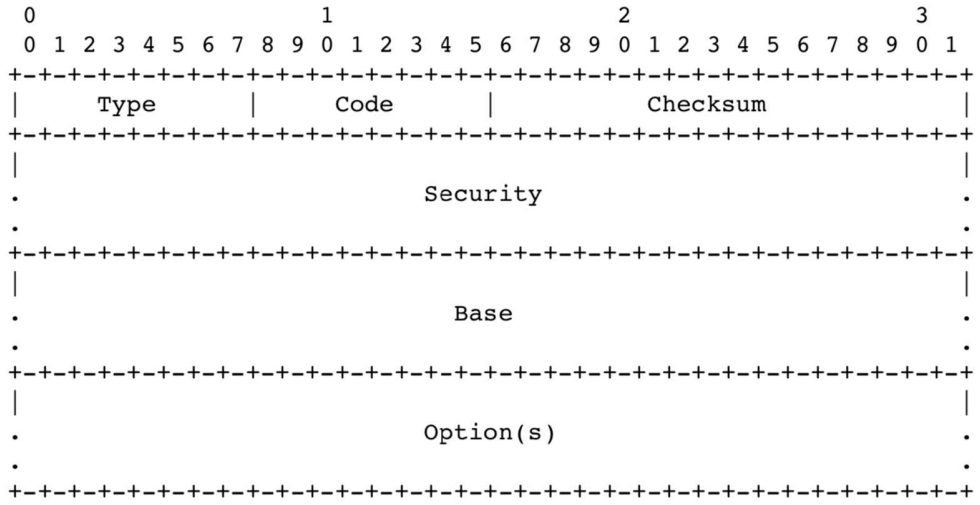


Figura 3.4: Mensaje ICMPv6 con capa de seguridad

El flujo de estos mensajes de control en el protocolo RPL siguen el orden que se muestra en la figura 3.5.

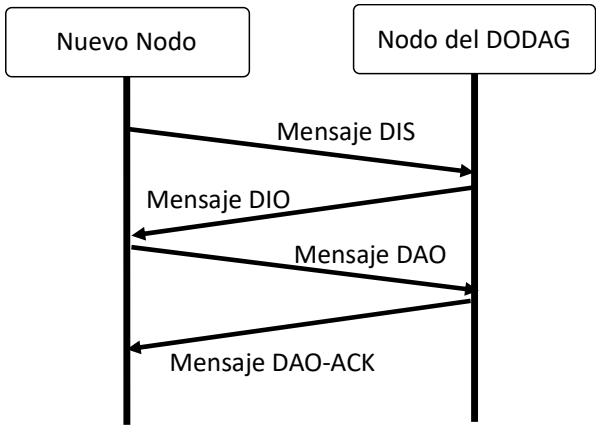


Figura 3.5: Flujo de mensajes de control

3.1.2.1 Formato de un mensaje DIO

El formato que sigue un mensaje DIO (véase Figura 3.6) contiene información referente al DODAG y todos los parámetros que ayudan al mantenimiento y la estabilidad del DODAG. A continuación, se detallan los parámetros que forman parte de este mensaje y su función dentro del protocolo RPL.

- **RPLInstanceID**: Campo de 8 bits que indica de qué instancia RPL es parte el DODAG.
- **Version Number**: Campo de 8 bits que identifica el número de versión del DODAG.
- **RANK**: Campo de 16 bits que informa el valor del parámetro Rank del nodo que envía el mensaje DIO.
- **Grounded** (G): Este flag indica si el DODAG tiene una aplicación final definida, es decir, si está conectado a una red externa o si es flotante. Si el flag está vacío el DODAG es flotante.
- **Mode of Operation** (MOP): Identifica el modo de operación de la instancia RPL. Todos los nodos que se unen al DODAG deben respetar este modo de operación para actuar como enrutadores, en caso contrario actuarán como nodos hoja. Los valores que puede tomar este campo son los siguientes:
  - 0: Sin rutas descendentes mantenidas por RPL
  - 1: Modo *non-storing*. Esto significa que los nodos no almacenan información de ruta descendente.
  - 2: Modo *storing* sin soporte *multicast*.
  - 3: Modo *storing* con soporte *multicast*.
- **DODAGPreference** (Prf): Indica, con un valor numérico positivo de 3 bits, como de preferible es el root comparado con otros roots de otros DODAGs.

- **Destination Advertisement Trigger Sequence Number** (DRSN): Valor indicado por el nodo que envía el DIO. Se usa como parte del procedimiento para mantener las rutas descendentes.
- **Flags y Reserved**: Campos no usados que se envían a cero por el emisor del mensaje DIO.
- **DODAGID**: 128 bits establecidos por el DODAG root. Es una dirección IPv6 que identifica de manera única un DODAG.
- **Options**: Un mensaje DIO puede transportar una serie de opciones que ayudan a la configuración y mantenimiento del DODAG. Este campo en particular es interesante porque nos permitirá enviar información extra entre los nodos, por ejemplo, información relativa a las coordenadas de los nodos, de manera no intrusiva.

RPL utiliza un temporizador *Trickle* para el envío de mensajes DIO entre los nodos, lo que significa que ese nodo transmite datos a menos que escuche datos de otras transmisiones que le sugieran que la transmisión es redundante. Existen varios parámetros para configurar el funcionamiento del temporizador Trickle:

- **Imin**: Interacciones mínimas de mensajes DIO.
- **Imax**: Interacciones máximas de mensajes DIO.
- **K**: Constante de redundancia de mensajes DIO. Si este valor es 0x00 significa que no se suprime ningún mensaje.

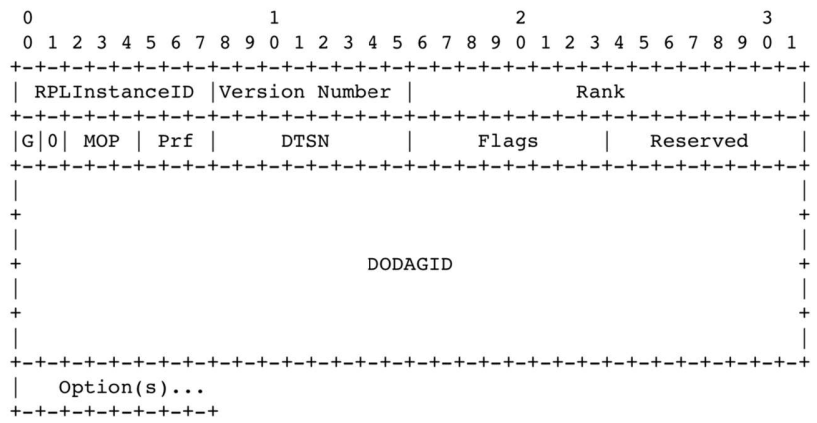


Figura 3.6: Formato de mensaje DIO

3.1.2.2 Formato de un mensaje DAO

Un mensaje DAO se utiliza para propagar la información hacia arriba a través del DODAG. En modo Storing, un mensaje DAO es de tipo Unicast, es decir, el mensaje va desde el hijo hasta el padre seleccionado; sin embargo, en modo Non-Storing el mensaje llegará hasta el DODAG *root*. Opcionalmente, el nodo receptor puede informar de manera explícita o bien por un error al nodo emisor del mensaje, mediante un mensaje de reconocimiento DAO-ACK.

El formato que sigue este tipo de mensaje DAO (véase Figura 3.7) contiene los siguientes parámetros:

- **RPLInstanceID**: Campo de 8 bits que indica de que instancia RPL es parte el DODAG.
- **K**: Indica que el emisor está esperando un mensaje de reconocimiento DAO-ACK.
- **D**: Indica que el campo DODAGID de 128 bits está presente en el mensaje.
- **Flags** y **Reserved**: Campos no usados que se envían a cero por el emisor del mensaje DIO.
- **DAOSequence**: Se incrementa cada vez un nodo envía un mensaje DAO y se recibe un DAO-ACK.
- **DODAGID**: 128 bits establecidos por el DODAG root. Es una dirección IPv6 que identifica de manera única un DODAG. Este campo está presente cuando una instancia local de RPL está en uso.
- **Options**: Igual que en un mensaje DIO, un mensaje DAO puede transportar una serie de opciones que ayudan a la configuración y mantenimiento del DODAG.

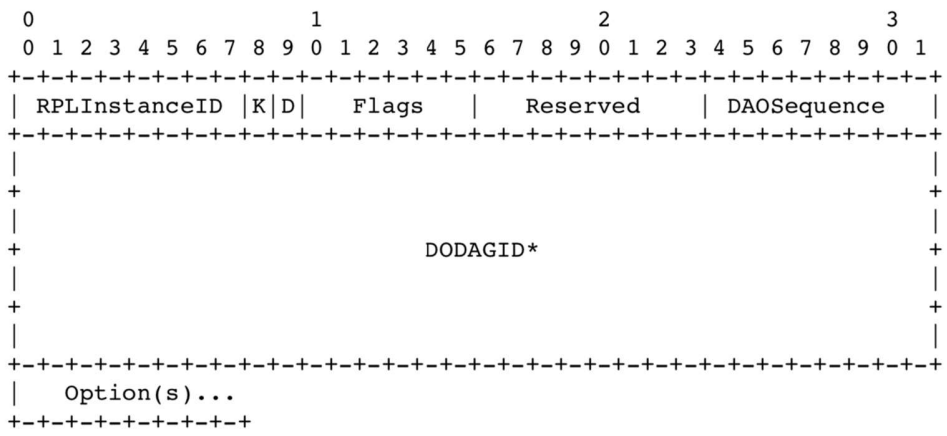


Figura 3.7: Formato de mensaje DAO

3.1.3 Función objetivo

La función objetivo en el protocolo RPL define cómo los nodos seleccionan y optimizan las rutas de una instancia RPL. También se encarga de calcular una métrica en un valor para el parámetro "Rank" que ayuda a la elección del padre dentro de un DODAG.

La función objetivo está diseñada para encontrar la tierra más cercana, es decir, para encontrar el root del DODAG. Esta función selecciona un padre preferente y un posible nodo *backup* sucesor. Todo el tráfico de subida en el grafo se dirige vía el padre preferido.

El cálculo del Rank del nodo (véase Figura 3.8) se realiza en base a los siguientes parámetros:

- Rank del padre preferido: *R(P)*.
- Factor máximo de Rank: Una implementación puede tener categorías que ayudan al cálculo del Rank. Estas categorías se convierten en un factor de multiplicaciónm *Rf (Rank factor)*.
- Máximos pasos de un Rank: *Sp (Step of Rank)*.
- Pasos que se habilitan para elegir un sucesor: *Sr (Strech of Rank)*.

```
R(N) = R(P) + rank_increase where:

rank_increase = (Rf*Sp + Sr) * MinHopRankIncrease
```

Figura 3.8: Cálculo del rank de un nodo [16].

3.2 Evolución del protocolo RPL para posicionamiento

Una vez analizado el funcionamiento del protocolo RPL, se procede a estudiar la implementación de este protocolo en el sistema operativo Contiki que se utilizará en la solución final para, de esta forma, identificar el flujo de llamadas de las funciones y poder así implementar las modificaciones necesarias para conseguir enviar las coordenadas de un nodo a otro.

Tal y como hemos observado, el mensaje DIO es un tipo de mensaje que un nodo emite para descubrir una instancia de RPL, conocer la configuración y mantener el DODAG. En nuestro caso, vamos a utilizar este mensaje para incluir la información relativa al posicionamiento del nodo, es decir, sus coordenadas y el tipo de nodo.

En este trabajo, se han definido, de forma lógica, diferentes tipos de nodos:

- **Nodo Referencia (NR)**: Este tipo de nodo se inicializa con unas coordenadas fijas en el sistema y es un nodo *fiable* a la hora de ayudar a calcular las coordenadas de otro nodo.
- **Nodo Móvil (NM)**: Este tipo de nodo hace referencia a un nodo que se desplaza dentro del sistema, lo que significa que inicialmente no tiene coordenadas asignadas.
- **Nodo Referencia Temporal (NRT)**: Un nodo de tipo NM se convierte a NRT cuando lleva un tiempo sin moverse. Idealmente, se puede disponer de nodos con acelerómetro para detectar este movimiento y alternar este parámetro. Si un nodo de tipo NRT se mueve, el valor del parámetro tipo cambia a NM.

3.2.1 Estimación por trilateración

Nuestra solución se basa en la técnica de trilateración por RSSI, ya que aprovecharemos el valor de este parámetro, el cual podemos obtener de la capa de enlace. Para poder llevar a cabo este método y calcular las coordenadas de un nodo objetivo, es necesario disponer de las coordenadas y las distancias de tres nodos referencia.

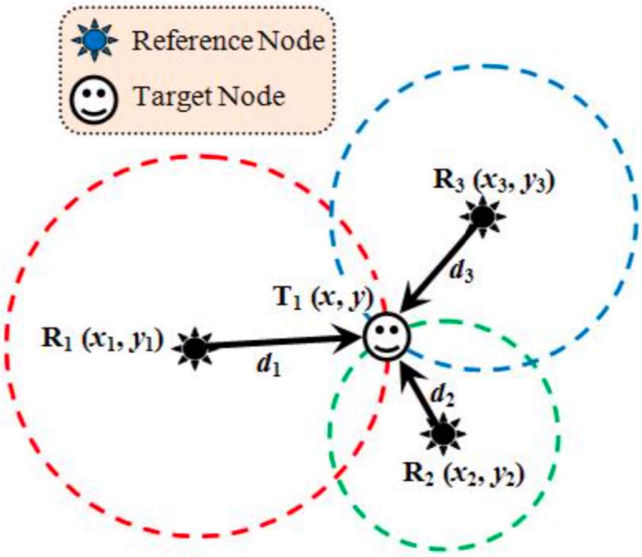


Figura 3.9: Estimación por trilateración [3]

En la figura 3.9 se representan los actores que forman parte en el método de cálculo de coordenadas por trilateración. El nodo objetivo (T_1) del cual queremos calcular las coordenadas (x,y), tres nodos referencia (R_1, R_2 y R_3) con sus coordenadas (x_i,y_i) y sus distancias (d_1, d_2 y d_3) respectivas.

Utilizando el teorema de Pitágoras obtenemos las siguientes expresiones:

$$d_1^2 = (x_1 - x)^2 + (y_1 - y)^2$$

$$d_2^2 = (x_2 - x)^2 + (y_2 - y)^2$$

$$d_3^2 = (x_3 - x)^2 + (y_3 - y)^2$$

Organizando las fórmulas para calcular las coordenadas del nodo objetivo obtenemos las siguientes ecuaciones [3]:

$$x = \frac{AY_{32} + BY_{13} + CY_{21}}{2(x_1Y_{32} + x_2Y_{13} + x_3Y_{21})}$$

$$y = \frac{AX_{32} + BX_{13} + CX_{21}}{2(y_1X_{32} + y_2X_{13} + y_3X_{21})}$$

Donde:

$$A = x_1^2 + y_1^2 - d_1^2$$

$$B = x_2^2 + y_2^2 - d_2^2$$

$$C = x_3^2 + y_3^2 - d_3^2$$

$$X_{32} = (x_3 - x_2)$$

$$X_{13} = (x_1 - x_3)$$

$$X_{21} = (x_2 - x_1)$$

$$Y_{32} = (y_3 - y_2)$$

$$Y_{13} = (y_1 - y_3)$$

$$Y_{21} = (y_2 - y_1)$$

Para poder llevar a cabo este algoritmo es necesario convertir el valor del campo RSSI a metros. Esta conversión no es trivial, ya que las ondas electromagnéticas presentan una relación inversamente proporcional entre la potencia recibida y la distancia al cuadrado como indica la siguiente expresión [3]:

$$P_r \propto \frac{1}{d^2}$$

La diferencia entre la potencia emitida y la potencia recibida se denomina *path loss*. Esta variable tiene un efecto en el cálculo de las coordenadas, ya que su comportamiento es exponencial según se muestra en la figura 3.10. Será necesario tener en cuenta este parámetro a la hora de transformar la potencia en distancia en un entorno real. En el caso de la solución de este TFM no se ha tenido en cuenta este parámetro ya que es un entorno simulado y la relación entre potencia y distancia en el simulador es lineal.

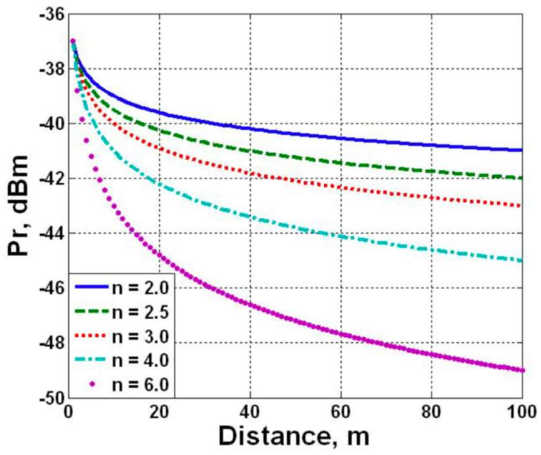


Figura 3.10: Efectos del "path loss"[3]

3.2.2 Información de un nodo

Un nodo, el cual llamaremos nodo propietario, debe almacenar la información de tres nodos vecinos de tipo referencia o referencia temporal para calcular su posición; para ello se ha diseñado una estructura que representa una tabla de posiciones (véase figura 3.11) de cuatro filas con las siguientes columnas:

- **X**: Representa la coordenada "x" del nodo propietario en la primera fila y las coordenadas x de los nodos referencia en las siguientes.
- **Y**: Representa la coordenada "y" del nodo propietario en la primera fila y las coordenadas y de los nodos referencia en las siguientes.
- ***rssi***: Representa el valor del parámetro RSSI de cada nodo referencia.
- ***last_update***: Representa la última vez que se actualizó la información de esa fila.
- ***type***: Representa el tipo de nodo que se almacena en esa fila (R, M, T).
- ***ipaddr***: Representa la IP del nodo que se almacena en esa fila. La primera fila llevará la IP del nodo propietario.

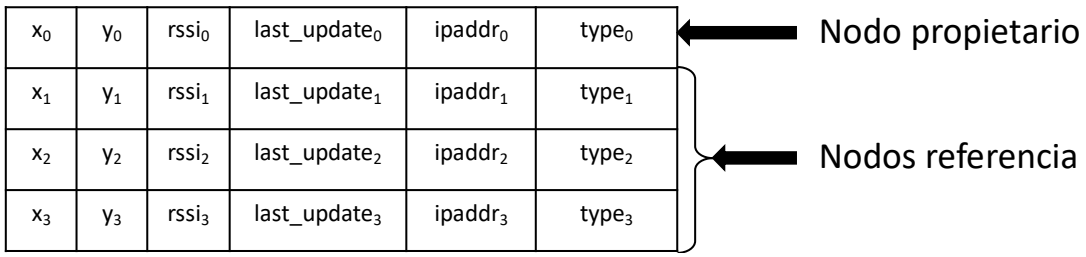


Figura 3.11: Tabla de posiciones

3.2.3 Funciones y pila de llamadas

Dentro del sistema operativo Contiki se identifican una serie de funciones que será necesario modificar y/o implementar para trabajar con la tabla anteriormente definida. Dentro de la pila de red de Contiki (véase Figura 3.1) se van a definir las nuevas funciones en el fichero `rpl.h`. Se definen en este fichero para que puedan estar disponibles en la capa de aplicación donde será necesario acceder a las coordenadas para publicarlas posteriormente. Las funciones definidas son las siguientes:

- *rpl_node_position_init*: Se encarga de inicializar la tabla que almacena todas las coordenadas necesarias para el cálculo de la posición.
- *rpl_set_node_position*: Actualiza la posición del nodo propietario, es decir, la primera fila de la tabla. Tiene como argumentos, las coordenadas y el tipo de nodo.
- *rpl_get_node_position*: Devuelve un puntero a la instancia de la tabla con todas las posiciones de los nodos.
- *rpl_print_positions*: Escribe en el puerto serie la tabla con todas las posiciones de los nodos. Se utiliza para debuguear el funcionamiento del sistema.
- *position_to_str*: Devuelve por referencia una cadena con los valores de la posición del nodo propietario, es decir, la primera fila de la tabla de posiciones. Cada campo viene separado por un pipeline "|".
- *position_to_JSON*: Devuelve por referencia una cadena con los valores de la posición del nodo propietario, es decir, la primera fila de la tabla de posiciones en formato JSON.

Según la interacción definida (véase figura 3.12) entre los diferentes módulos que existen en Contiki y que se encargan de las comunicaciones, se establece el envío de las coordenadas del nodo propietario en el broadcasting del mensaje DIO, es decir, en la función *dio_output*. En esta función se implementa el código para enviar las coordenadas y tipo de nodo del nodo propietario.

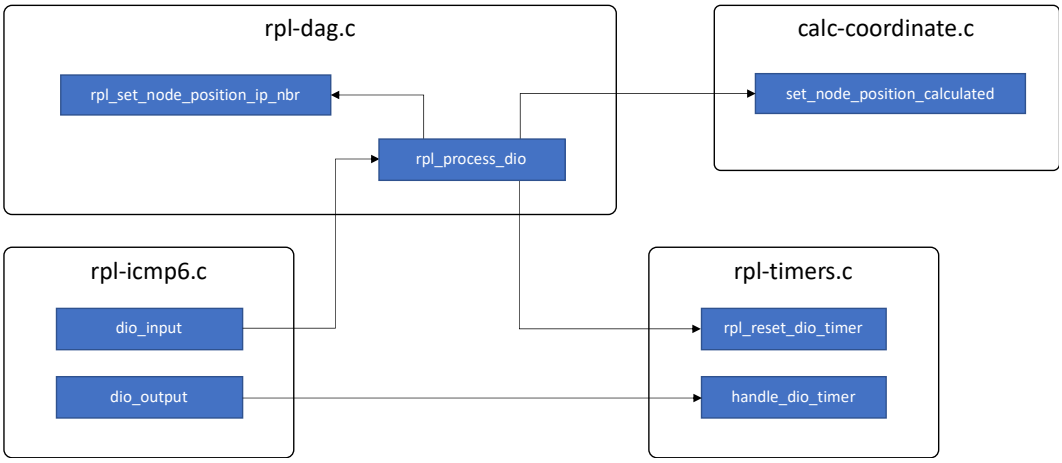


Figura 3.12: Flujo de llamadas para el cálculo de la posición

Es necesario crear una nueva opción dentro del mensaje DIO, que se añadirá al *payload* del mensaje. Esta opción se define en la interface `rpl-private.h` y se utiliza tanto en la función *dio_output* como en *dio_input*. También será necesario modificar la definición del objeto *rpl_dio_t* para almacenar la información relativa a las coordenadas del nodo que envía la información. Los campos creados son los siguientes:

- *x*: Valor de la coordenada x del nodo emisor. Tipo *uint16_t*.
- *y*: Valor de la coordenada y del nodo emisor. Tipo *uint16_t*.
- *rssi*: Valor del RSSI del nodo emisor. Tipo *int16_t*.
- *type*: Tipo de nodo del nodo emisor. Tipo *unsigned char*.

El flujo de las llamadas funciona de la siguiente manera:

1. Se envía un mensaje mediante la función *dio_output* en función del tiempo que maneja la función *handle_dio_timer*. Recordemos que el valor de este tiempo se puede configurar para que no se supriman mensajes. En nuestra solución se implementa el envío de las coordenadas y el tipo de nodo en las opciones del mensaje.

2. La función *dio_input* recibe el mensaje y prepara la instancia del objeto *rpl_dio_t*. En nuestra solución, se rellenan los campos creados específicamente para el cálculo de coordenadas y explicados anteriormente.
3. La función rpl_process_dio recoge la información que llega en el objeto rpl_dio_t y realiza las operaciones pertinentes para incluir el nodo en el DODAG. En nuestro caso, se sigue la siguiente lógica:
    a. Comprobamos que el nodo propietario no es de tipo referencia, ya que estos nodos disponen de información de posición fija.
    b. Si el tipo de nodo del nodo emisor es de tipo móvil limpiamos este nodo de la tabla de posiciones puesto que, si anteriormente era referencia temporal, no es fiable para el cálculo.
    c. Invocamos a la función *rpl_set_node_position_ip_nbr* que se encarga de gestionar los nodos activos dentro de la tabla de posiciones y que se utilizan para el cálculo de las coordenadas del nodo propietario. El funcionamiento de esta función se explica en el siguiente capítulo.
    d. Invocamos a la función *set_node_position_calculated* que se encargará de aplicar el algoritmo de trilateración explicado anteriormente si la tabla de posiciones tiene tres nodos válidos para el cálculo.

3.2.3.1 Gestión de la tabla de posiciones

La tabla de posiciones (véase Figura 3.11) es clave en nuestra solución, ya que permite a un nodo propietario conocer la posición de los nodos vecinos y de esta manera, calcular sus coordenadas. Además, guarda un registro del último momento en el que se actualizaron los datos; así pues, ayuda también a establecer un tiempo límite para el cual un nodo emisor no es fiable para el cálculo de la posición porque haya salido del rango de visión del nodo propietario.

El código que implementa esta gestión se encuentra en la función rpl_set_node_position_ip_nbr y recibe como argumento la información de posición de

un nodo emisor, es decir, *x, y, rssi, ipaddr y type*. Con esta información, se gestiona la tabla de posiciones estableciendo la siguiente lógica:

1. Limpiar referencias antiguas: En función del parámetro *LAST_UPDATE_LIMIT* se limpian las filas de la tabla cuya columna *last_update* supere el valor de este parámetro.
2. Si el nodo emisor se encuentra en la tabla se compara el rssi y si es distinto se actualiza la fila con la información recibida.
3. Si el nodo emisor no se encuentra en la tabla y el *rssi* es mayor, es decir, más fuerte ya que es un valor en negativo, se actualiza esa fila con la información recibida.

Esta función solo opera sobre los nodos referencia, es decir, no modifica ni consulta el nodo propietario.

3.3 Topología de red

Para diseñar la topología de red que utilizará nuestra solución, nos fijamos en la capa de aplicación de la pila de red del sistema operativo Contiki. Nuestro objetivo final es mostrar de manera actualizada la posición de los nodos en un mapa a escala sobre el que monitorizamos los nodos-sensores en la zona física. Para ello, vamos a necesitar diferentes funcionalidades en la capa de aplicación dependiendo del tipo de nodo y, por consiguiente, de su cometido en el sistema.

La topología de red (véase figura 3.13) está formada por un router de borde que dará salida al exterior a la información de nuestra red, un broker MQTT que tendrá dos funcionalidades. La primera, publicar las posiciones de los nodos referencia para que estos puedan inicializar el sistema con los valores correspondientes. Y la segunda, publicar las coordenadas de los nodos móviles que queremos monitorizar. La herramienta tunslip de Contiki nos proporciona comunicación entre el router de borde y el exterior, sin embargo, desde las motas no ha sido posible ir directamente a la red global de internet. Solamente para que la simulación sea completa y se pueda observar en un dashboard en internet el movimiento de los nodos, se ha optado por poner un node-red que reenvía la información entre el broker del host local y el broker de la

plataforma Thingsboard. En un entorno real, será necesario establecer la configuración de la red para que los nodos móviles publiquen directamente en el *broker* de la plataforma sin necesidad de utilizar node-red como proxy en las comunicaciones.

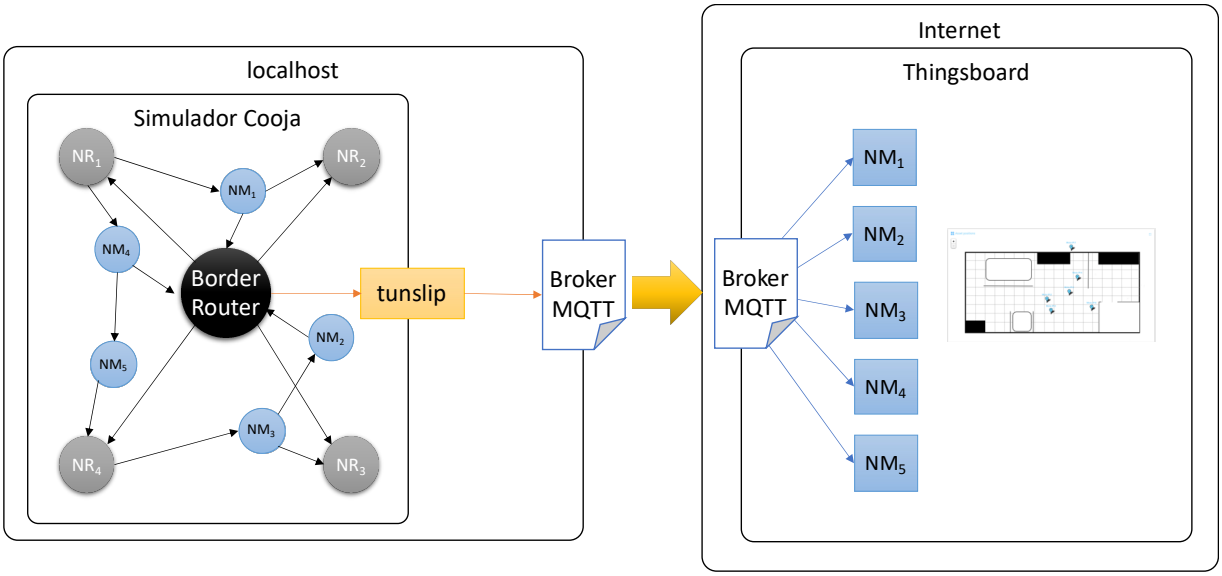


Figura 3.13: Topología ejemplo

Toda la topología referente a la red de nodos-sensores estará simulada en el simulador Cooja que proporciona Contiki. Para poder comunicar los nodos del simulador con el entorno real, será necesario utilizar la herramienta *tunslip* que crea un túnel entre el router de borde y el sistema físico (realmente, un enlace con cualquier red física externa).

Un ejemplo de topología con 4 nodos referencia y 5 nodos móviles se puede observar en la figura 3.13.

3.3.1 Tipo de nodo router de borde

Para este tipo de nodo se utilizará el ejemplo que viene desarrollado sobre el protocolo RPL en el sistema operativo Contiki. En este caso no hay que realizar ninguna modificación, ya que viene preparado para ser el nodo *root* del DODAG con el que trabaja el sistema.

3.3.2 Tipo de nodo referencia

Con el fin de poder establecer las coordenadas de este tipo de nodo, sin la necesidad de reconstruir el sistema completo con las coordenadas fijas. Este tipo de nodo está suscrito a una cola MQTT donde se publican las coordenadas del nodo en función de su IP en formato comprimido. El *topic* al que está suscrito este tipo de nodo tiene el formato "*/position/[IP]*" y el mensaje que se publica debe seguir el formato "*[x] | [y] | [tipo_de_nodo]*".

3.3.3 Tipo de nodo móvil

En la capa de aplicación, este nodo lleva implementada la funcionalidad que se encarga de publicar en el *broker* MQTT las coordenadas del nodo propietario, así como el tipo de nodo y la IP en formato comprimido. El *topic* donde se publica la información tiene el formato "*v1/devices/me/telemetry*" definido por el producto Thingsboard al definir un dispositivo. El mensaje debe llevar el formato JSON para que se publique correctamente y sigue el formato "*{IP:[valor_ip],x:[valor_x],y:[valor_y],type:[valor_tipo]}*".

3.4 Implementación de la solución

En primer lugar, es necesario clonar el repositorio oficial de Contiki, obteniendo la última versión del código sobre el que vamos a trabajar. La versión más actualizada de Contiki se llama Contiki-NG y supone un cambio importante en cuanto a estructura y arquitectura de software respecto a las anteriores versiones. Sin embargo, para este proyecto se ha optado por la versión anterior en su versión 3.x. Esta decisión se debe a que la simulación se debe realizar sobre motas con mayor capacidad de almacenamiento, lo que supone utilizar la plataforma wismote que posee hasta 128K de memoria ROM, pero que no existe en Contiki-NG.

Sobre una máquina virtual Linux versión 18.04 de 64 bits clonamos el repositorio con el siguiente comando: *sudo git clone "https://github.com/contiki-os/contiki/tree/master"*

La implementación realizada sobre esta versión de Contiki se ha subido a un repositorio GitHub el cual se puede descargar y utilizarlo en un nodo-sensor que soporte 6LoWPAN. La solución no es intrusiva ya que se ha diseñado ampliando las opciones que

ya incluyen los mensajes DIO, con lo que un nodo-sensor que no tenga esta versión de posicionamiento funcionará sin ningún problema con su cometido para el que fue pensado, pero sin ser posicionado en el mapa.

El código modificado para esta solución se encuentra en el repositorio de GitHub público https://github.com/jose-segoviano/TFM-CONTIKI-3. Para identificar fácilmente el código añadido que implementa esta solución se ha enmarcado dentro de comentarios con el texto "JSG – INI" y "JSG – FIN". Además, hay que tener en consideración un nuevo fichero con nombre `calc-coordinate.c` añadido para el cálculo de las coordenadas, el cual se encuentra en el directorio "*core/lib/*" de Contiki.

Aunque la solución no es intrusiva con respecto al protocolo original RPL, cabe destacar las funciones del código original que sí han sido modificados para dar solución al envío y recepción de los parámetros referentes al posicionamiento del nodo y que se han añadido a las opciones enviadas en el mensaje DIO.

En la función *dio_output* se han añadido las coordenadas y el tipo del nodo propietario que se envían ahora en el mensaje DIO (véase Figura 3.14).

```
// JSG - INI Send the position and the node type.
buffer[pos++] = RPL_OPTION_POSITION;
rpl_node_position_t *node_position = rpl_get_node_position();
set16(buffer, pos, node_position->x[0]); // con uint16
pos += 2; // con uint16
set16(buffer, pos, node_position->y[0]); // con uint16
pos += 2; // con uint16
buffer[pos++] = node_position->type[0];
// JSG - END
```

Figura 3.14: Envío de variables de posición en dio_output

En la función dio_input se recogen las variables de posicionamiento (véase figura 3.15) siempre que el mensaje tenga la opción activada en rpl-private.h.

```
// JSG - INI - new option to read from payload option
case RPL_OPTION_POSITION:
  dio.x = get16(buffer, i+1);
  dio.y = get16(buffer, i+3);
  dio.rssi = packetbuf_attr(PACKETBUF_ATTR_RSSI);
  dio.type = buffer[i+5]; // i+3 con uint8
  break;
  // JSG - END
```

Figura 3.15: Lectura de variables de posición en dio_input

En la función rpl_process_dio se recupera la información relativa a la posición que llega en el mensaje dio, se gestiona la tabla de posiciones y se calculan las coordenadas si se dispone de tres nodos referencia (véase Figura 3.16).

```
// JSG - INI
if (node_position.type[0] != RPL_NODE_POSITION_TYPE_REFERENCE){
  if (dio->type != RPL_NODE_POSITION_TYPE_MOBILE) {
    rpl_set_node_position_ip_nbr(dio->x, dio->y, dio->rssi, *from, dio->type);
    set_node_position_calculated(&node_position);
  }
}
rpl_print_positions("Dio");
// JSG - END
```

Figura 3.16: Lógica de cálculo de posiciones en rpl_process_dio en rpl-dag.c.

Capítulo 4 - Diseño de pruebas y resultados

El contenido de este capítulo está enfocado a realizar las pruebas necesarias sobre la solución implementada con el objetivo de encontrar los puntos de mejora y el valor añadido que puede tener una solución de este tipo. Para ello, utilizaremos la herramienta de simulación Cooja que proporciona Contiki.

Las pruebas se han separado en dos vías. La primera de ellas, identificar los tiempos de convergencia desde que un nodo móvil se incorpora al sistema y se le asignan coordenadas. La segunda, la monitorización de esta información en un *dashboard* en el exterior del sistema.

4.1 Herramienta de simulación – Cooja

Cooja es una herramienta destinada a la simulación de redes bajo el protocolo RPL sobre nodos implementados utilizando Contiki. Está desarrollada en Java y su código está disponible en el repositorio oficial de Contiki [23]. Inicialmente, Contiki provee de una imagen (Instant Contiki) que nos facilita una máquina virtual con un entorno de desarrollo completo. Sin embargo, es necesario recurrir a otras fuentes de información y foros que solucionan los problemas de configuración existentes a la hora de trabajar con Cooja de una manera óptima [17].

Las simulaciones que se van a realizar para este trabajo se han creado con la opción UDGM (Unit Disk Graph Model) (véase Figura 4.1) y con motas de tipo Wismote (véase Figura 4.2). En las pruebas definidas en los siguientes capítulos se explicará en más detalle el uso de las funcionalidades de esta herramienta.

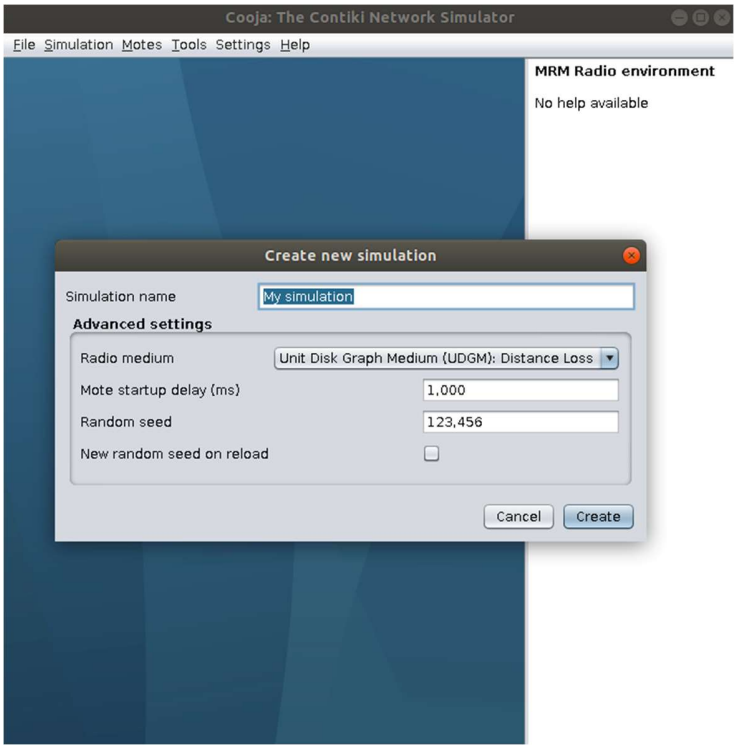


Figura 4.1: Venta de nueva simulación en Cooja

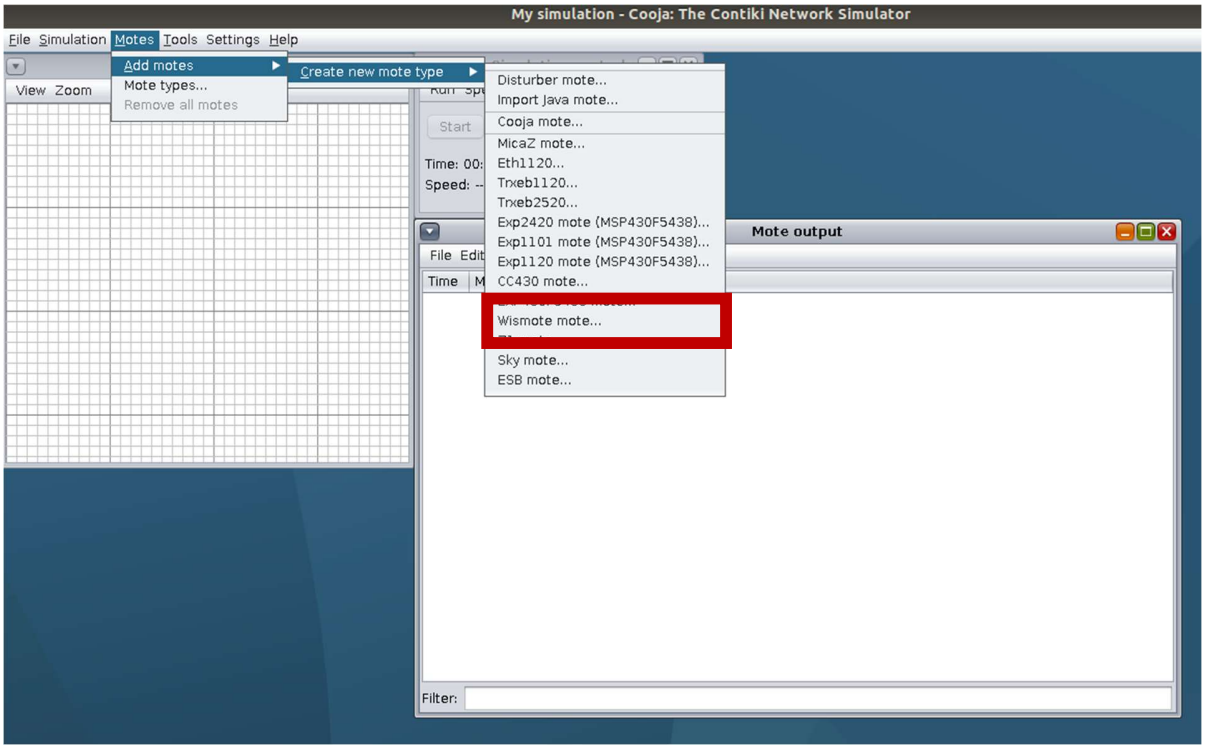


Figura 4.2: Añadir nodo de tipo Wismote a la simulación

4.2 Cálculo de distancia en función del RSSI

Como ya se ha explicado anteriormente, la transformación a realizar de potencia a distancia en un entorno real no es trivial debido al "*path loss*" detectado en las comunicaciones. Para nuestras pruebas, esta relación va a ser lineal, sin embargo, debemos encontrar los parámetros que definen nuestra recta para separar el cálculo y poder parametrizarlo en un futuro entorno real.

Para llevar a cabo esta tarea, se ha definido una prueba con dos motas en las que se muestra por consola el RSSI recibido y se relaciona con la distancia conocida. Se utiliza el plugin *mobility [26]* de Cooja el cual permite mover de manera automática una mota a una coordenada concreta. Se toman las siguientes premisas:

- Las coordenadas mostradas por el simulador corresponden a metros en un entorno físico.
- Los decimales de las coordenadas se descartan y se trabaja con números enteros.
- Una mota de tipo Wismote tiene un alcance de aproximadamente 35 m en el simulador bajo los parámetros de simulación utilizados.

El plugin Mobility de Cooja mueve las motas en la vista según un fichero donde cada fila debe seguir el siguiente formato:

- Columna 1: número de la mota que se va a posicionar. La primera es la cero.
- Columna 2: Tiempo en el que el simulador mueve la mota.
- Columna 3: Coordenada x donde se moverá la mota.
- Columna 4: Coordenada *y* donde se moverá la mota.

Para facilitar la generación de ficheros de prueba, se ha generado un pequeño script en Python (véase Figura 4.3) que genera el fichero de posiciones para cada escenario.

Partimos de una posición inicial en la que la mota 1 y la mota 2 no están dentro de alcance (véase Figura 4.4), pero una vez lanzada la simulación con el fichero de

coordenadas, la mota 2 se va acercando a la mota 1 escribiendo por pantalla el valor del RSSI recibido (véase Figura 4.5).

```
g=open("/home/chema/Simulaciones/positions3-motas.dat", "w")
y = 70.00
for i in range(80):
    y -= 1
    linea = "1 " + str(i*100) + " 70.00 " + str(y) + "\n"
    g.write(linea)
g.close()
```

Figura 4.3: Script para generar ficheros de pruebas con el plugin mobility.

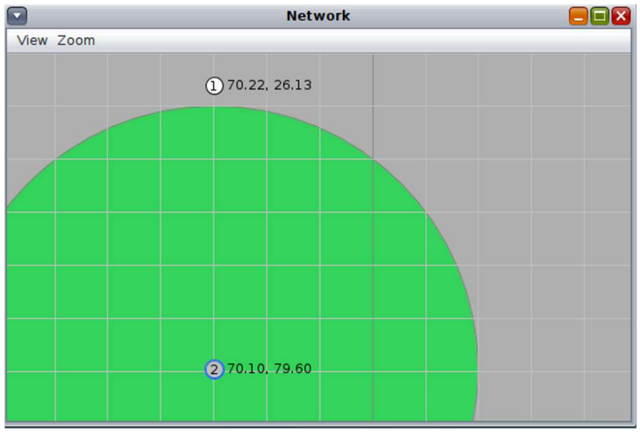


Figura 4.4: Situación inicial para recuperar datos del RSSI

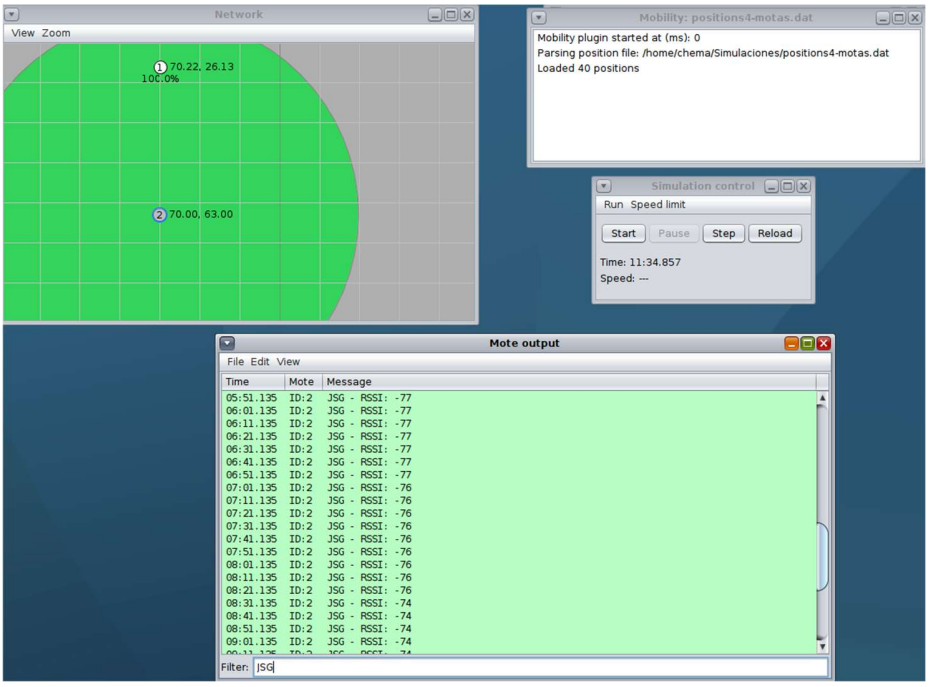


Figura 4.5: Ejecución de la simulación con el plugin mobility

Con la información obtenida, podemos cruzarlo con nuestro fichero de coordenadas y tiempos para obtener la relación entre el RSSI y su distancia. Si representamos estos datos en una distribución (véase Figura 4.6) podemos obtener la ecuación que nos transformará cualquier valor RSSI en una distancia:

$$d = -0{,}3928(RSSI) - 15{,}921$$

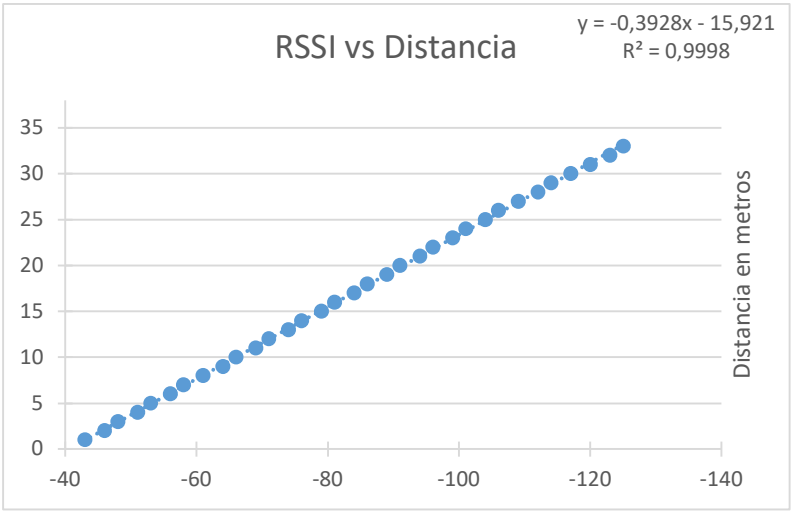


Figura 4.6: Distribución de RSSI frente a Distancia.

En un entorno real, esta ecuación es muy diferente, ya que se conoce que la relación es exponencial dependiendo del valor del *path loss*.

4.3 Pruebas de convergencia sin router de borde

El objetivo principal de estas pruebas es recuperar información relativa a los tiempos de convergencia del sistema para estudiar su posible viabilidad como sistema de posicionamiento *indoor*. Las pruebas realizadas son las siguientes:

1. Prueba 1. Tres nodos referencia y un nodo móvil.
2. Prueba 2. Tres nodos referencia y cinco nodos móviles.
3. Prueba 3. Tres nodos referencia y 10 nodos móviles.
4. Prueba 4. Tres nodos referencia y 20 nodos móviles.

4.3.1 Prueba 1. Tres nodos referencia y un nodo móvil

En este primer ejemplo se crea un entorno de simulación con los tres nodos de tipo referencia (véase Figura 4.7) en las coordenadas (1,1), (20,10) y (10,30). En nuestro caso, la mota 4 está fuera del alcance ya que es el nodo de tipo móvil sobre el que realizaremos las pruebas y estudiaremos su convergencia.

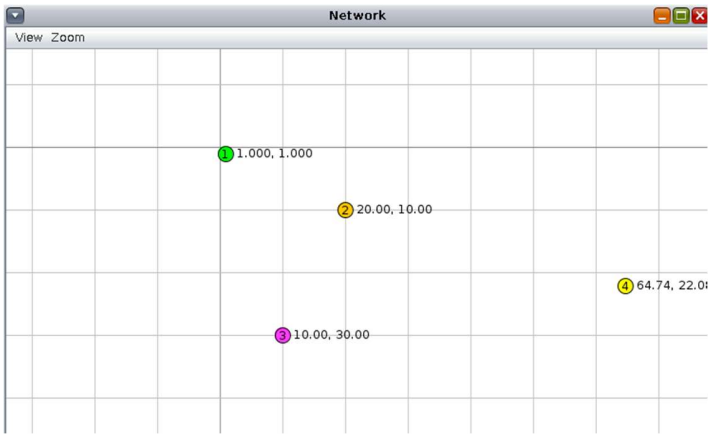


Figura 4.7: Entorno de simulación de la prueba 1

El código que se ejecuta en cada nodo es el mismo con una pequeña diferencia y es que cada uno de ellos debe tener las coordenadas que le pertenecen por posición en el mapa del simulador. Esto, también se puede hacer mediante la opción que da Cooja en cada una de las motas, donde se puede acceder a las variables del código (véase Figura 4.8). Para ello habría que preparar el código con esas variables específicas y poderlas modificar desde esta herramienta.

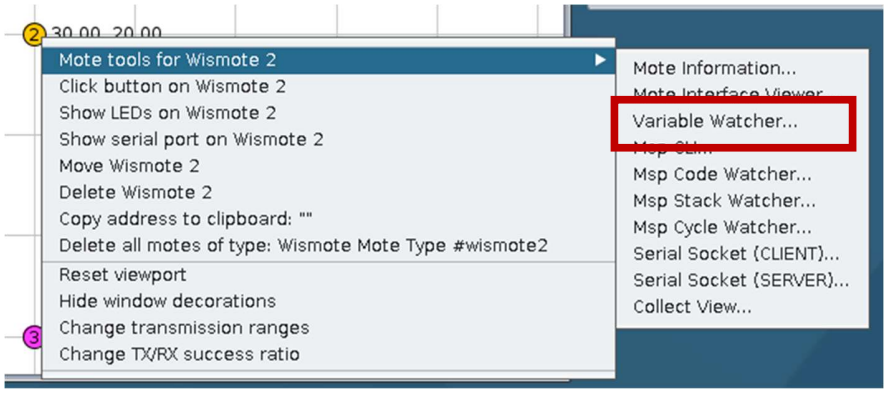


Figura 4.8: Variable Watcher para modificar variables del código de la mota

Durante la ejecución de la simulación movemos la mota 4 por diferentes puntos del mapa y esperamos a que la consola imprima las nuevas coordenadas. Después de realizar varios cambios, exportamos a un fichero de texto la información para analizarlo.

En la gráfica mostrada en la figura 4.9 se representa el tiempo que ha tardado el sistema en calcular las nuevas posiciones para cada movimiento hecho en el simulador.

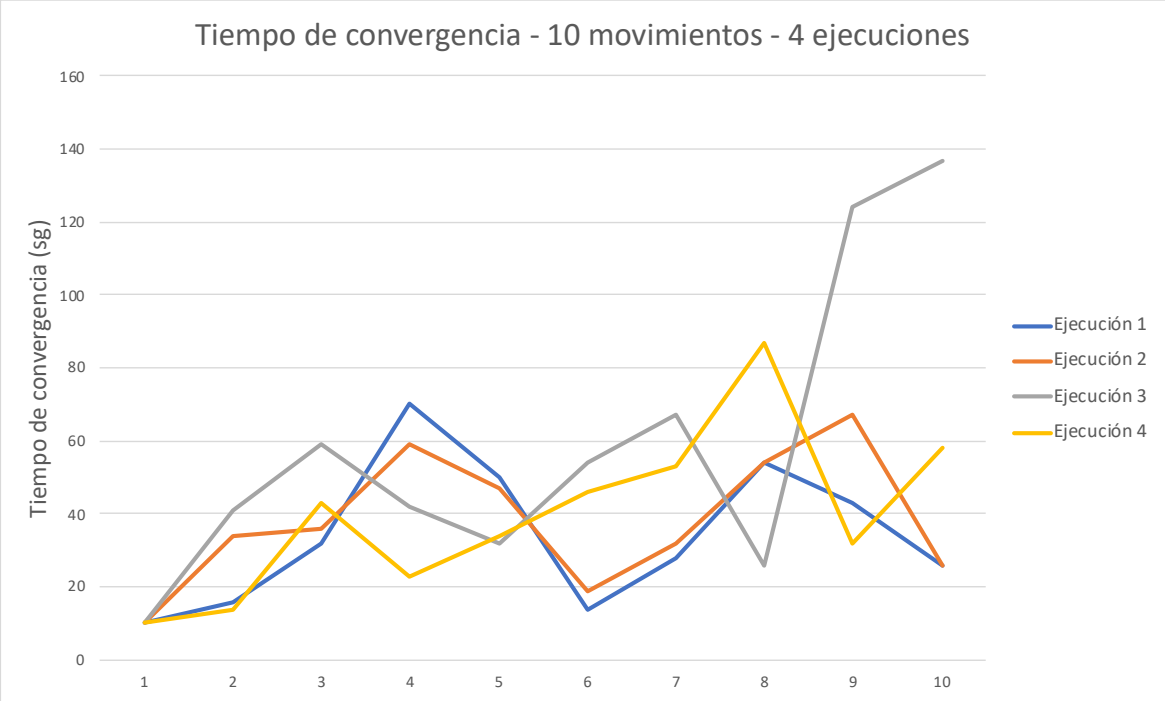


Figura 4.9: Tiempo de convergencia con un nodo móvil

Si observamos las gráficas que dibuja cada una de las ejecuciones lanzada, podemos observar que la convergencia no sigue ningún tipo de patrón al relacionarlo con el tiempo. Esto puede deberse a la propia herramienta de simulación, ya que se ha observado durante el desarrollo de este TFM que la herramienta muestra una alta discrepancia entre valores de pruebas similares. Esto nos hace pensar, que la herramienta Cooja puede ser muy útil para el desarrollo de los procesos y el testing de la funcionalidad, pero no es fiable para medir tiempos simulando un entorno real.

En cualquier caso, el sistema es capaz de calcular correctamente las coordenadas de los nodos móviles.

4.3.2 Prueba 2. Tres nodos referencia y cinco nodos móviles

Sobre la misma base de la prueba anterior, se crean 5 nodos móviles (véase Figura 4.10) que iremos moviendo aleatoriamente en el mapa del simulador. Para medir los tiempos tendremos en cuenta que todos los nodos móviles muestran las coordenadas correctas por la consola del simulador la cual exportaremos en un fichero de texto como en la prueba anterior, para analizar posteriormente.

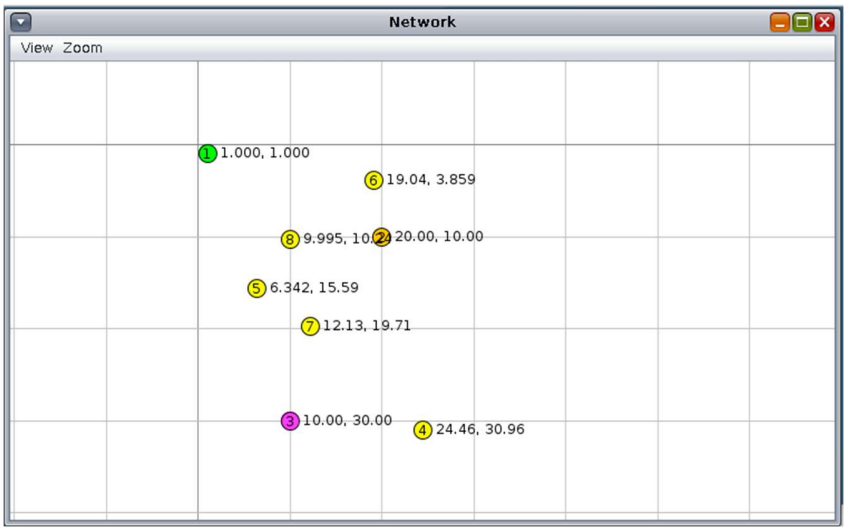


Figura 4.10: Entorno de pruebas con 5 nodos móviles

Los resultados obtenidos a nivel de tiempos son muy dispares y no tiene un valor añadido representar un gráfico con la convergencia del sistema ya que cada nodo devuelve unos valores de convergencia muy diferentes. En general, se observa que los nodos encuentran más rápidamente su posición debido a que algunos nodos de tipo móvil pasan a referencia debido a su estacionamiento. Esto ayuda a nodos que incluso están fuera de rango de los tres nodos referencia principales, a obtener una posición gracias a los nodos vecinos. Es decir, se observa que gracias al funcionamiento del protocolo RPL, un sistema con muchos nodos puede resultar muy útil a la hora de obtener el posicionamiento de todos los nodos que forman parte del sistema ya que se ayudan entre ellos.

Puede ocurrir el caso en el que un determinado nodo de tipo móvil se convierta en referencia por estacionamiento, pero en el momento de moverse de nuevo, existe un periodo de tiempo en el que el grafo tarda en converger. En ese tiempo, este nodo

está enviando la información errónea de que es un nodo referencia y hace que los nuevos nodos vecinos calculen mal su posición. Este problema en un entorno real se suprime debido a que el sensor puede llevar un acelerómetro que sea el que actualice la posición y el tipo de nodo, lo que hará que no se envíen coordenadas erróneas.

4.3.3 Prueba 3. Tres nodos referencia y 10 nodos móviles

En esta prueba, el objetivo es observar el tiempo que toma en conseguir la posición un nodo que está fuera del rango de los tres nodos referencia fijos, como es el caso de la mota 12 de la figura 4.11.

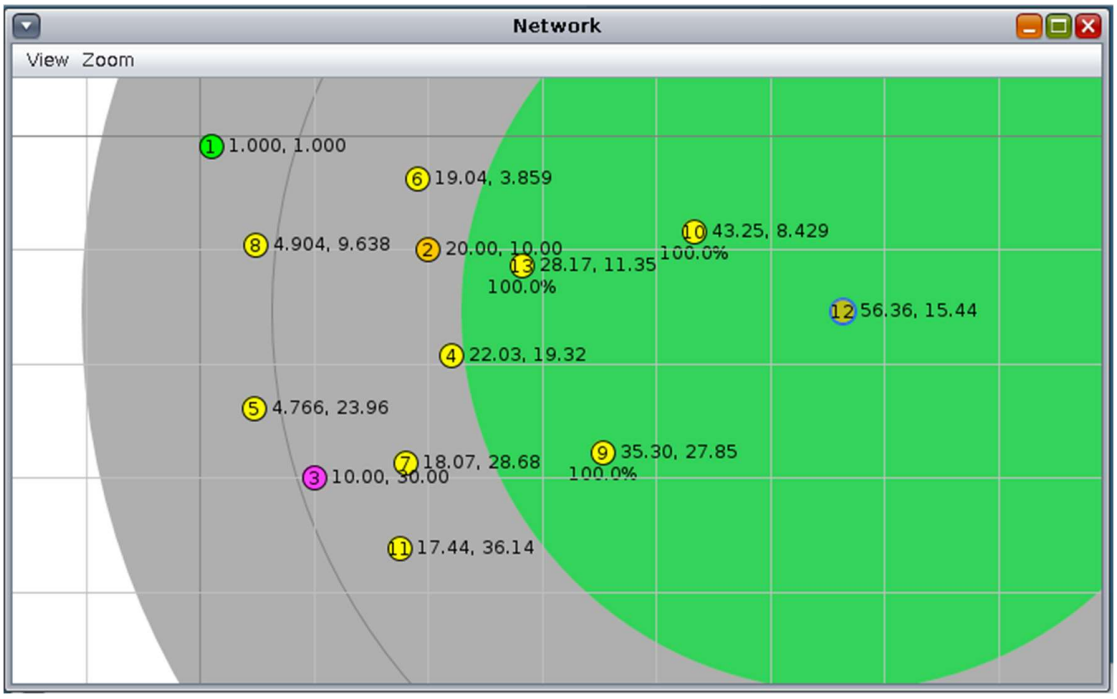


Figura 4.11: Distribución de nodos prueba 3.

Para que el nodo 12 consiga las coordenadas, es necesario que al menos tres de los nodos más próximos se conviertan en nodo referencia. El nodo 12 consigue las coordenadas 49 segundos después de comenzar la simulación (véase figura 4.12). Las motas en las que se apoya para conseguir sus coordenadas son las motas 10, 9 y 13, las cuales han tenido que esperar a obtener las coordenadas con el apoyo de otros nodos de tipo referencia temporal ya que tampoco estaban en el alcance de los nodos referencia fijos.

Este tiempo es relativo al tiempo asignado al parámetro que define cuándo un nodo de tipo móvil cambia a tipo referencia temporal. El valor de este parámetro en esta prueba es de 10 segundos después de conseguir las coordenadas. Si bajamos este valor a 2 segundos observamos que el tiempo mínimo de convergencia baja a 28 segundos (véase Figura 4.13).

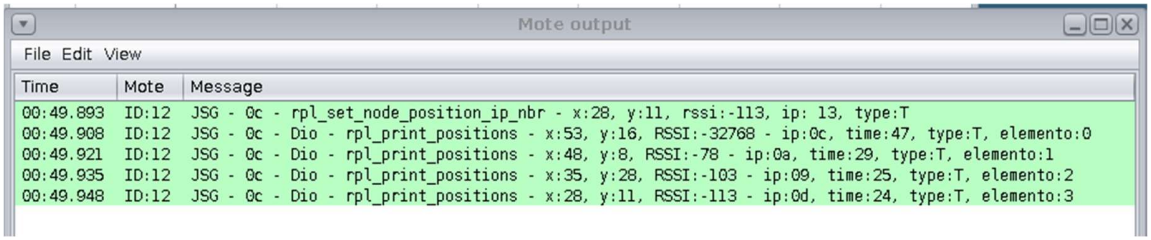


Figura 4.12: Consola del simulador Cooja para la mota 12 con límite en 10 segundos.

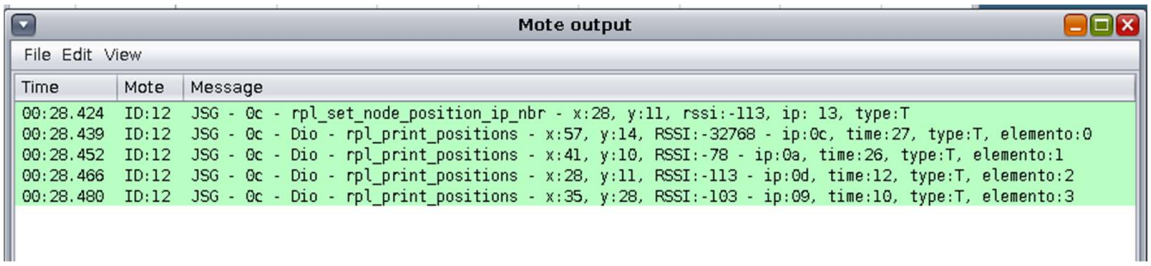


Figura 4.13: Consola del simulador Cooja par la mota 12 con límite en 2 segundos

Se observa también que la precisión disminuye debido a que el cálculo se realiza con motas de tipo referencia temporal, las cuales tienen unas coordenadas calculadas y por lo tanto sin una precisión 100% fiable. Es decir, cuantas más motas intermedias de tipo referencia temporal tengamos, menos fiable será la precisión en nuestro sistema.

4.3.4 Prueba 4. Tres nodos referencia y 20 nodos móviles

En la siguiente prueba se eleva el número de nodos móviles hasta 20 (véase Figura 4.14) para comprobar si el tiempo de convergencia aumenta a medida que usamos más nodos sin posición fija.

En los resultados que obtenemos de esta prueba, vemos que el tiempo de convergencia, utilizando un valor de 2 segundos para el parámetro que define el límite antes de convertirse en nodo referencia temporal, es de 26 segundos (Figura 4.15). Es decir, no se observa una relación entre el número de nodos en la red y el tiempo de convergencia.

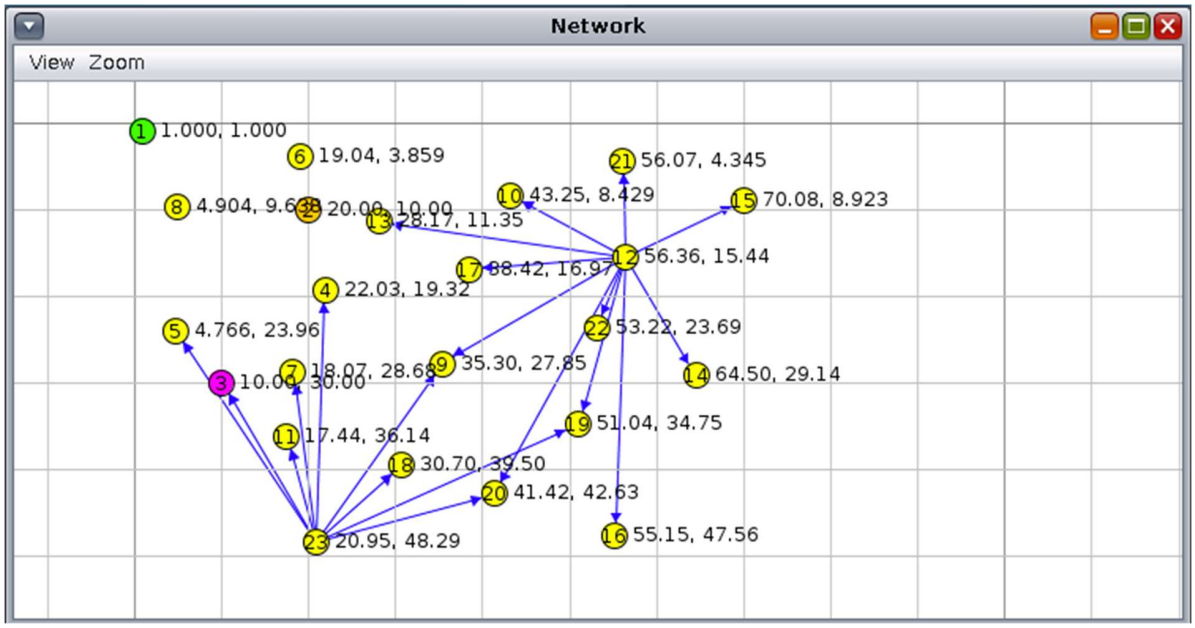


Figura 4.14: Simulación con tres nodos fijos y 20 móviles.

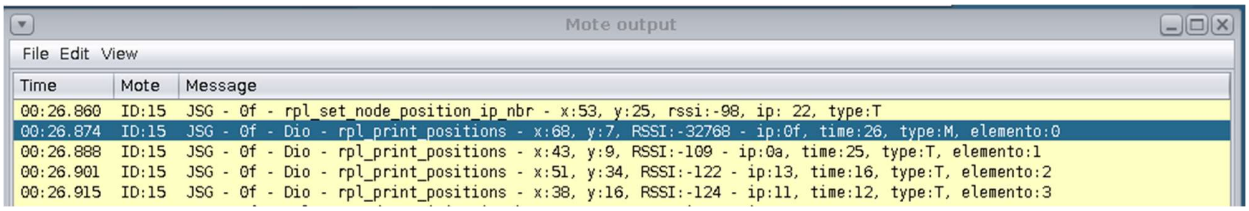


Figura 4.15: Consola del simulador Cooja para la mota 15.

4.4 Pruebas de convergencia con router de borde

El objetivo principal de estas pruebas es monitorizar en un mapa, fuera del simulador Cooja, el posicionamiento de los nodos y de esta manera poder demostrar el funcionamiento correcto del sistema en un entorno más cercano a la realidad. Para ello se utiliza la herramienta Thinsgboard en su modo Cloud.

Esta prueba se divide en dos partes. El objetivo en la primera de ellas es enviar las posiciones de nodos a un servidor UDP que se está ejecutando en el host local sobre el que se ha implementado la solución. En la segunda prueba, el objetivo es publicar la información en el broker MQTT que ofrece Thingsboard, además de representar los nodos en un mapa que represente un espacio físico real.

4.4.1 Prueba 1. Router de borde y servidor UDP en localhost

La prueba se prepara con los siguientes componentes en el sistema:

- *Mota Router de borde*: Representa el enlace con el exterior del simulador. En la figura 4.16 esta mota está representada en color verde.
- *Broker MQTT*: En localhost se levanta el servicio mosquitto [27] que se encargará de publicar las posiciones de los nodos de tipo referencia para su configuración.
- *Motas Referencia*: Se utilizan tres motas que tienen el papel de nodos referencia; es decir, es necesario indicarles su posición fija cuando se arranca el sistema. Las motas están suscritas al topic "/position/[num_mota]" en el broker MQTT donde se publicará su posición y su tipo de nodo con un mensaje con formato "x|y|tipo". En un entorno real, esta configuración nos ayuda a escalar el sistema de manera sencilla, ya que nos permite personalizar la topología de red en función de las necesidades del caso de uso. En la figura 4.16 estas motas son las representadas en color rosa.
- *Motas móviles*: Se utilizan 15 motas con el objetivo de examinar cómo se ayudan los nodos móviles cuando cambian a tipo referencia temporal. Para ello, se colocan las motas de manera estratégica en el simulador Cooja (véase Figura 4.16) para comprobar la forma de propagar el cálculo en función del tipo de nodo. En la figura 4.16 estas motas son las representadas en color naranja. La información de la posición de estas motas se envía a un servidor UDP.
- *Servidor UDP:* Se implementa en Python y escucha sobre el puerto 3000. Se ejecuta en el host local. En las motas móviles es necesario indicar la IP en formato IPv6 de este servidor.
- *Herramienta tunslip*. Con esta herramienta se establece un túnel entre el simulador Cooja y el localhost. Se encuentra en la carpeta tools de Cooja y es necesario crear el ejecutable con make.

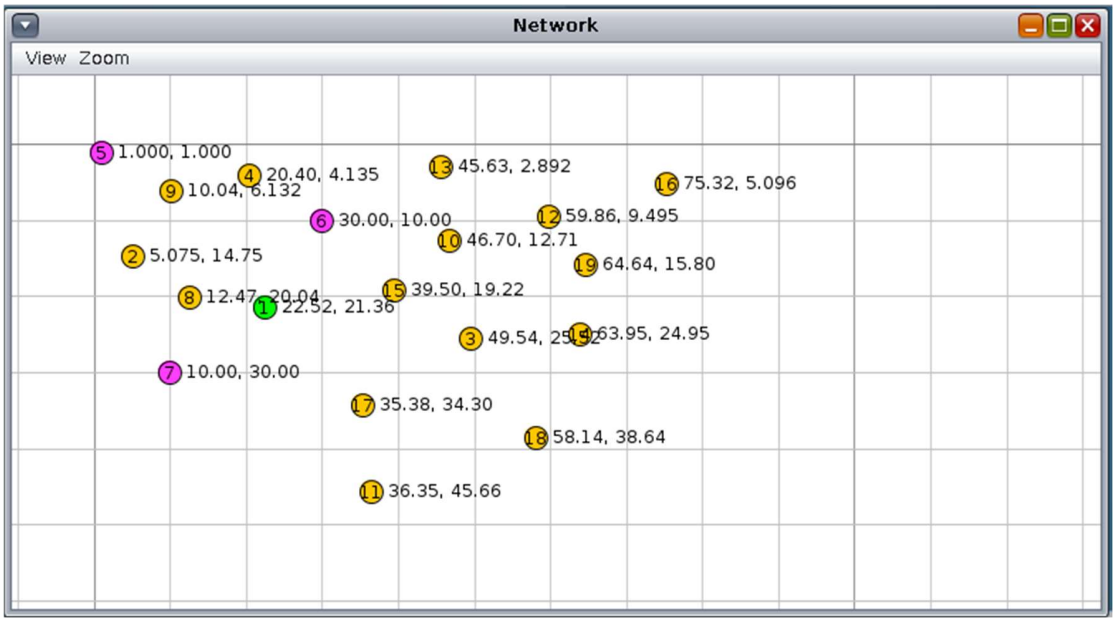


Figura 4.16: Topología con 15 nodos de tipo móvil.

Para ejecutar esta prueba se realizan los siguientes pasos:

1. En la mota del border router será necesario indicar que escuche por el puerto 60001, para que se comunique con la herramienta tunslip. Para ello en el botón derecho de la mota (véase Figura 4.17) elegimos la opción "Mote tolos for wismote 1/Serial Socket (SERVER)" y pinchamos en el botón "start" (véase Figura 4.18)

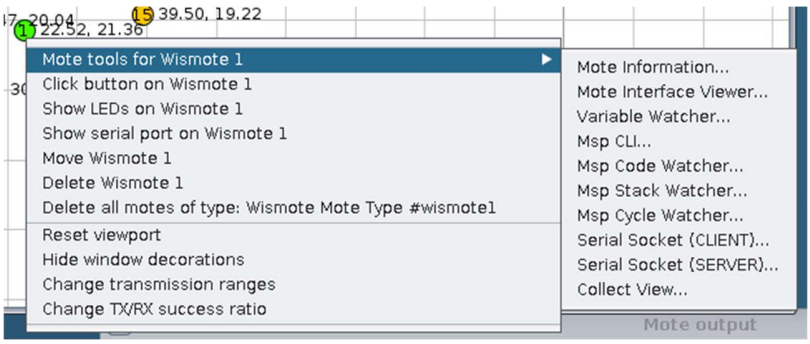


Figura 4.17: menú del botón derecho de la mota border router.

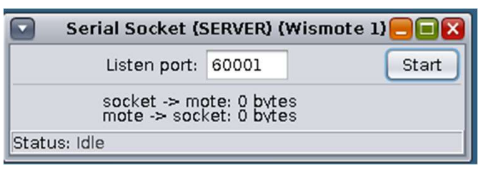


Figura 4.18: puerto de escucha del border router para comunicación con tunslip.

2. En un terminal del sistema operativo ejecutamos la aplicación tunslip con el siguiente comando:

   ```
   sudo ./tunslip6 -a localhost aaaa::1/64
   ```

   la opción -a indica el servidor que en nuestro caso es localhost.

   El siguiente parámetro es un rango de IPs con el prefijo que utiliza la red y que se puede configurar en el Makefile.

3. Levantamos el servidor UDP que se encuentra en la carpeta apps/udp-server-py mediante el comando:
   ```
   python3 udp-server.py
   ```
4. Lanzamos la simulación en Cooja y una vez que comprobamos (Figura 4.19) que las motas de tipo referencia se han conectado y suscrito a la cola MQTT, en un terminal del sistema operativo, publicamos las posiciones con los siguientes comandos:
   ```
   mosquitto_pub -h localhost -t "/position/05" -m "1|1|R"
   mosquitto_pub -h localhost -t "/position/06" -m "30|10|R"
   mosquitto_pub -h localhost -t "/position/07" -m "10|30|R"
   ```

Figura 4.19: Simulador Cooja con motas que se conectan a la cola MQTT.

5. A medida que las motas van obteniendo las posiciones se publican en el servidor UDP y se muestran por pantalla (véase Figura 4.20)

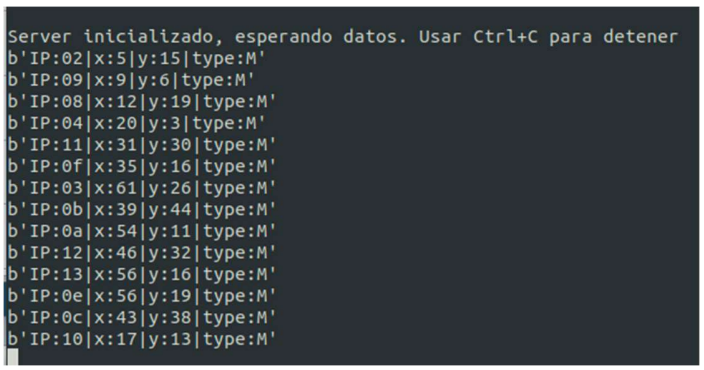


Figura 4.20: Coordenadas llegando al servidor UDP.

4.4.2 Prueba 2: Representación en mapa con Thingsboard

El objetivo de esta prueba es representar el movimiento de las motas que se crean en el simulador Cooja, en un mapa que representa un espacio físico, completando de esta manera un entorno similar a una prueba real. Para llevar a cabo esta prueba se utilizan los siguientes componentes:

- *Mota Router de borde*: Representa el enlace con el exterior del simulador. En la figura 4.21 esta mota está representada en color verde.
- *Node-red:* Se utiliza de proxy entre localhost y la plataforma Thingsboard. Este paso es posible suprimirlo en un entorno real haciendo que el router de borde publique directamente en el broker de Thingsboard, sin embargo, por falta de tiempo se ha realizado la configuración con node-red por ser más sencilla. La configuración del flujo de node-red utilizado se detalla en el apéndice.
- *Motas Referencia*: Se utilizan cuatro motas que tienen el papel de nodos referencia, es decir, es necesario indicarles su posición fija cuando se levanta el sistema. Su configuración se realiza exactamente igual que en la prueba 4.4.1.
- *Motas móviles*: Se utilizan seis motas que publicarán las coordenadas en una cola MQTT en formato JSON. En la figura 4.21 estas motas son las representadas en color naranja.
- *Broker MQTT*: En localhost se levanta el servicio mosquitto que actúa de broker donde las motas referencia se suscriben para conocer sus coordenadas y las motas móviles publican la información de su posición.
- *Herramienta tunslip*: Funciona de la misma manera que en la prueba 4.4.1.
- *Plataforma IoT Thingsboard*: Esta herramienta se utiliza para configurar un dashboard público donde se puede ver el movimiento de las motas en un mapa. La configuración de la plataforma y sus componentes se desarrolla en detalle en el apéndice. El dashboard es público en la url

https://demo.thingsboard.io:/dashboard/a77e8c20-b39c-11ea-a0ad-19f7b95d9654?publicId=bb84cec0-b39a-11ea-a0ad-19f7b95d9654

- *MongoDB*: Base de datos donde se almacenan las posiciones de los nodos.

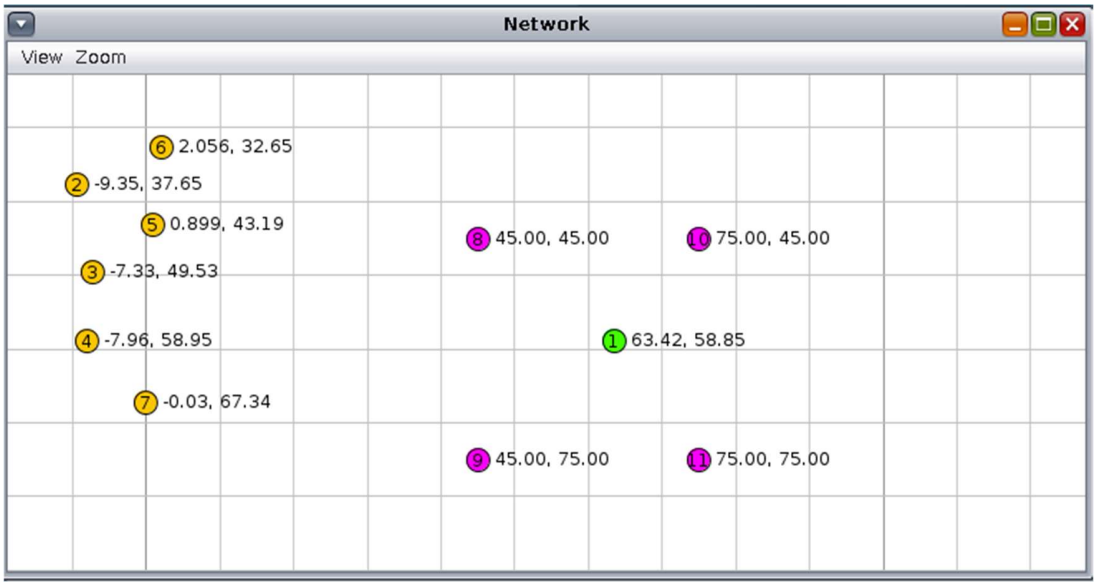


Figura 4.21: Topología prueba 2 con router de borde.

Al igual que en la prueba 4.4.1 es necesario realizar una serie de pasos para configurar el entorno y que el router de borde simulado en Cooja tenga salida al exterior del simulador. En este caso particular no utilizamos un servidor UDP y lo centralizamos todo el despliegue en el protocolo MQTT.

En esta prueba, los nodos de tipo móvil no se convierten a nodo de tipo referencia temporal, ya que el objetivo de la prueba es conocer las coordenadas de nodos en movimiento y si activamos esa opción, se pierde precisión en el cálculo de las coordenadas debido a la acumulación de error en el cálculo de las coordenadas. Por ejemplo, un nodo de tipo referencia temporal tiene unas coordenadas con un rango de error entre uno y dos puntos en la x e y. La nueva coordenada a calcular a partir de este dato, incrementa este rango exponencialmente en función del número de nodos de tipo referencia temporal en el que nos basamos, llegando a calcular posiciones muy lejos de la posición real.

La mejor manera de poder demostrar el funcionamiento de esta prueba es un video donde se pueda observar el movimiento de los nodos en el simulador y

posteriormente en el dashboard de la plataforma. El vídeo está publicado en la dirección URL https://github.com/jose-segoviano/TFM-CONTIKI-3/blob/master/doc/Video-demo-positioning.mp4.

En el vídeo se puede observar lo siguiente:

- Las motas referencia toman un pequeño tiempo en suscribirse al broker MQTT. Hasta ese momento no se pueden configurar con sus posiciones reales (vésae Figura 4.22).

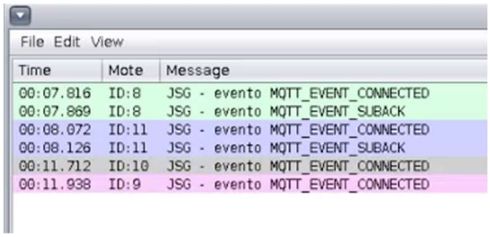


Figura 4.22: Suscripción MQTT de los nodos referencia.

- Una vez que la mota entra nueva en la red, la convergencia del sistema es relativamente rápida, ya que al ser un nodo nuevo los mensajes DIO se iteran con rapidez para reajustar el DODAG (véase Figuras 4.23, 4.24 y 4.25).

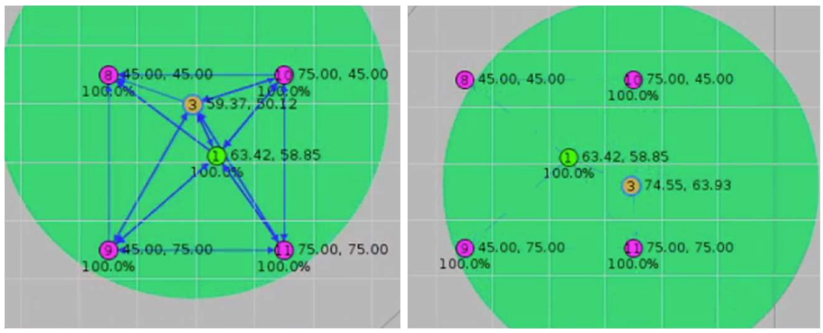


Figura 4.23: Nodo móvil 03 en Cooja en diferentes posiciones.

En la figura 4.26 se pueden observar todos los nodos dentro del DODAG. En la consola se observa la tabla de posicionamiento del nodo 04. Se puede observar que está a falta de las coordenadas de un nodo referencia para calcular sus propias coordenadas. En la figura 4.27 se observa el dashboard con los nodos posicionados excepto el 04.

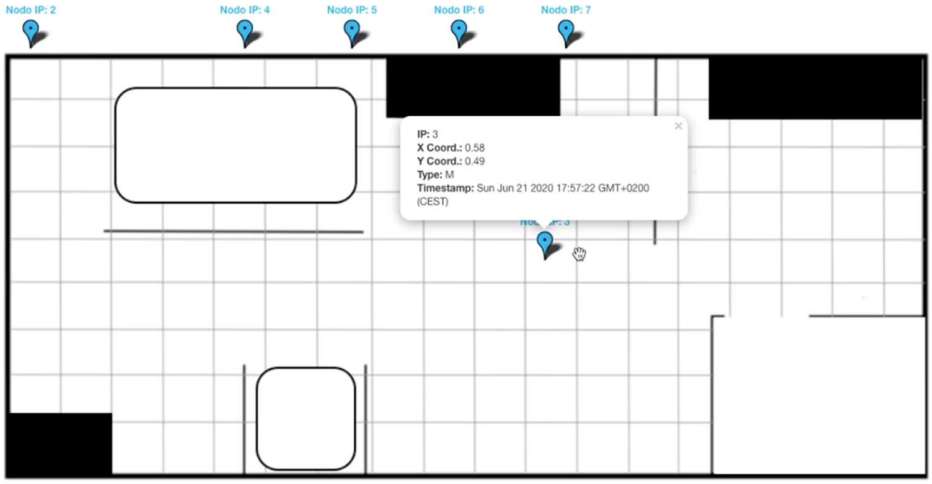


Figura 4.24: Representación del nodo 3 en Thingsboard.

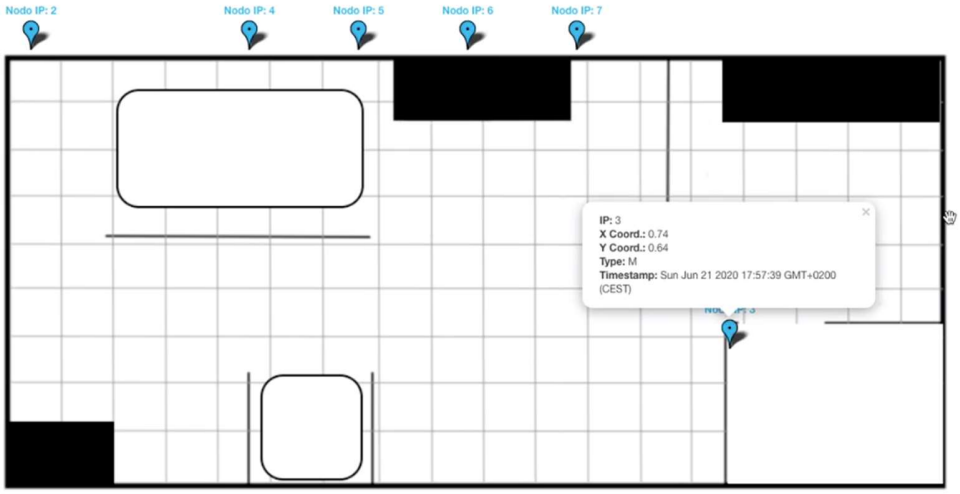


Figura 4.25: Representación del nodo 03 en diferentes posiciones.

- Cuando las motas se mueven por el simulador, la actualización en el dashboard depende más del tiempo de publicación de las posiciones en la cola MQTT que del tiempo de convergencia del grafo. Por ejemplo, se observa que la mota 6 ha cambiado de posiciones en el simulador (véase Figura 4.28), pero no se actualiza en el dashboard (véase Figura 4.29), esto es debido a que el software que publica en la cola MQTT utiliza una máquina de estados y en alguna ocasión pierde la conexión con el broker que tiene que reiniciarse. En este sentido, podría ser más eficiente el uso

de un servidor UDP o bien optimizar el software MQTT para que funcione de manera más estable.

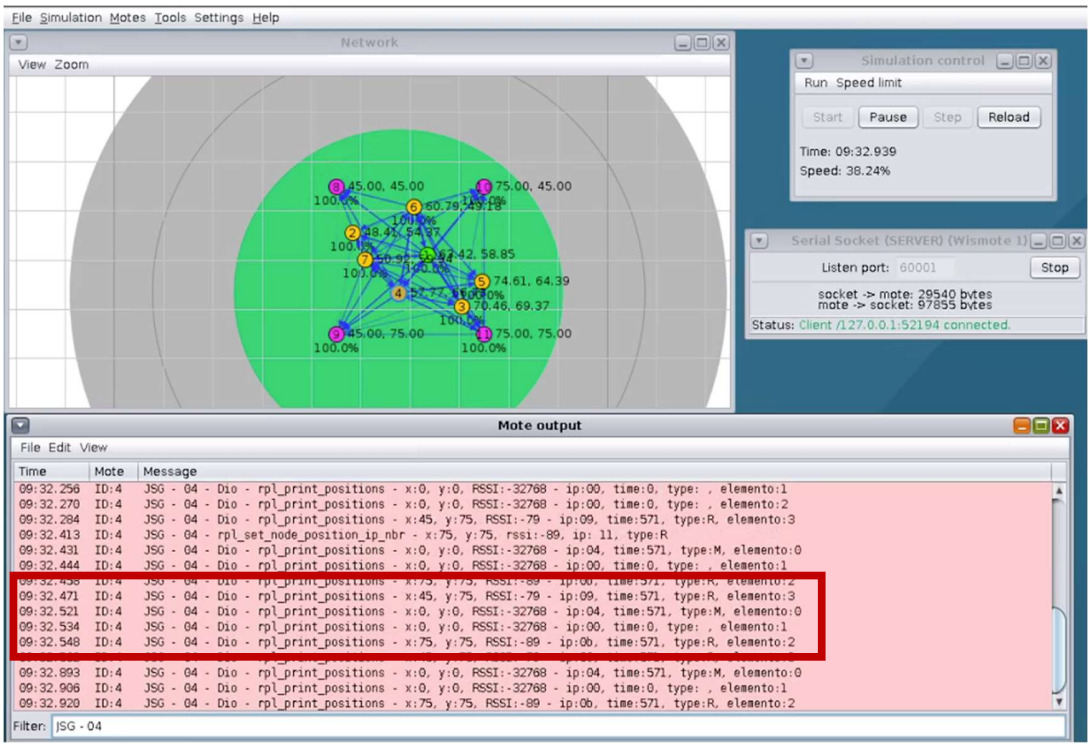


Figura 4.26: Simulador Cooja con todos los nodos móviles dentro del DODAG.

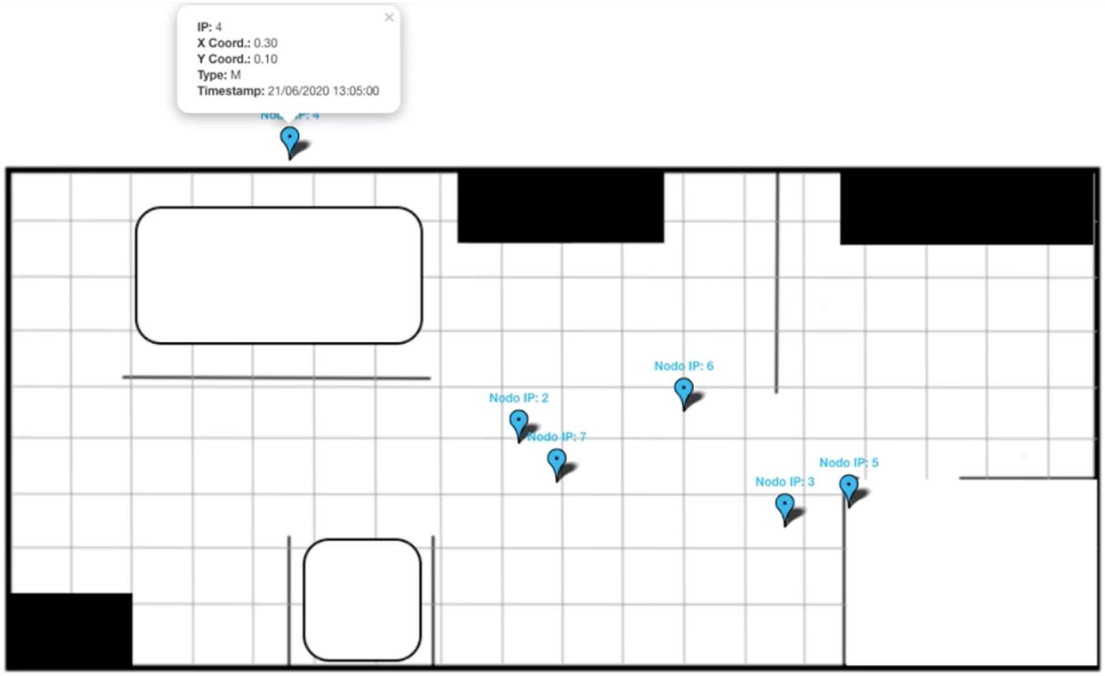


Figura 4.27: El nodo 04 está esperando las coordenadas en thingsboard.

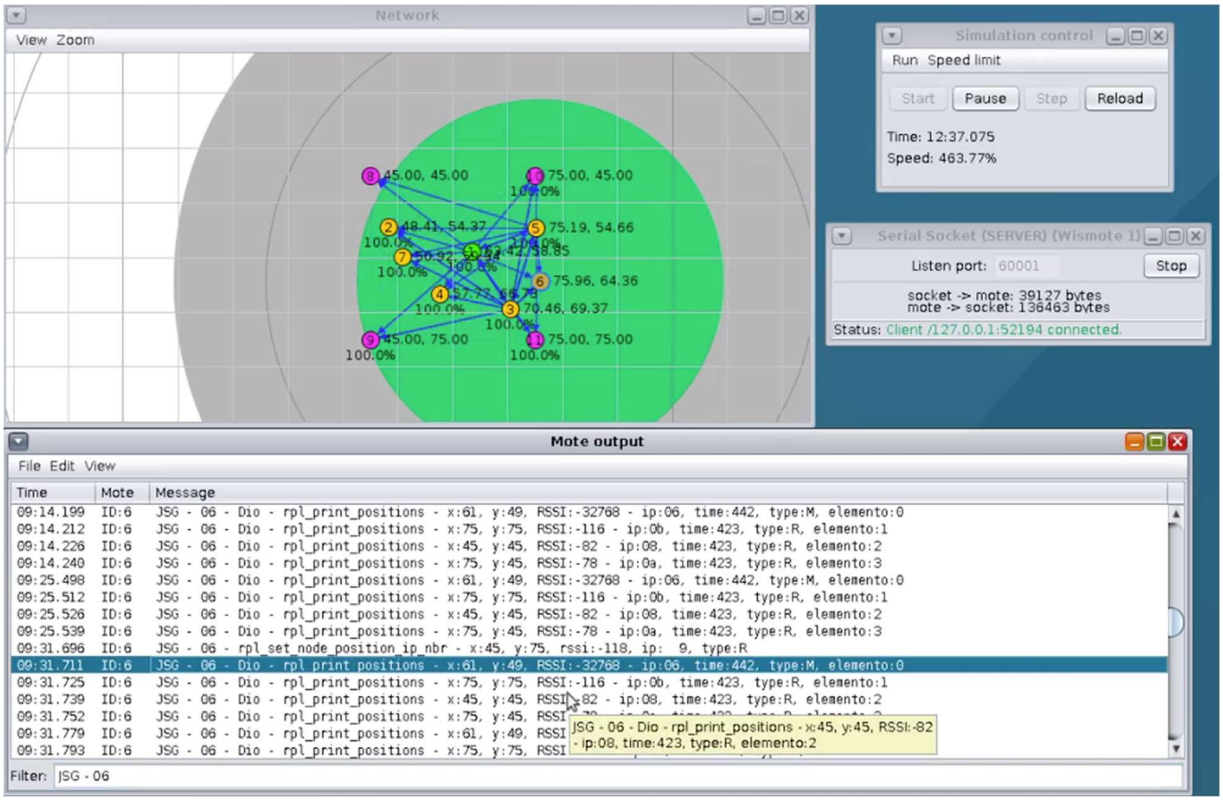


Figura 4.28: Simulador Cooja. Mota 06 con coordenadas.

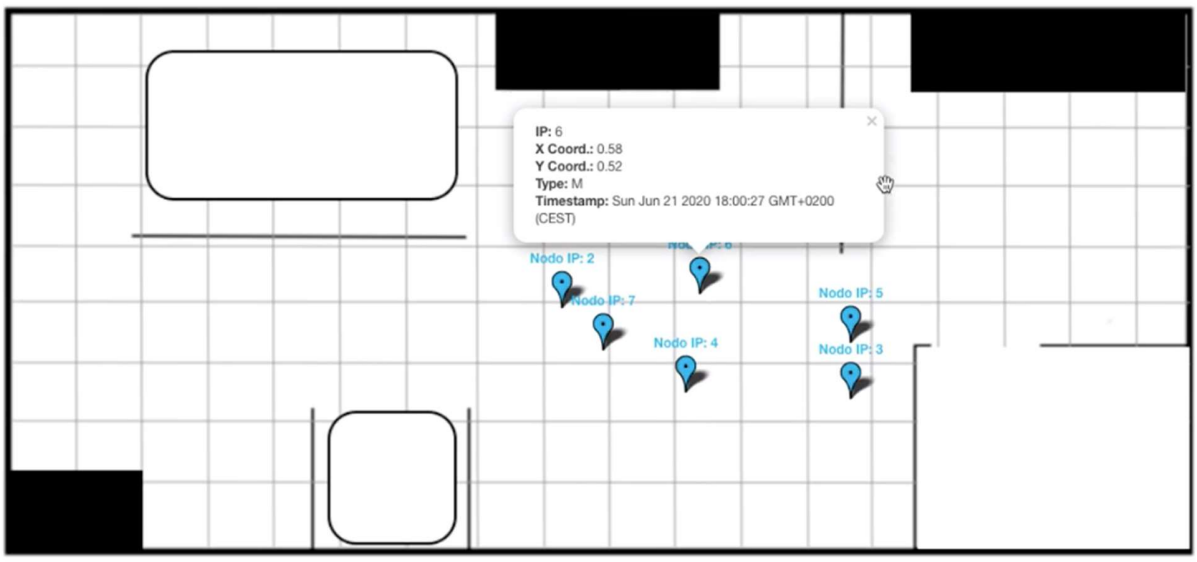


Figura 4.29: Dashboard en Thinsboard con el nodo 06 marcado.

Capítulo 5 - Conclusiones y siguientes pasos

5.1 Conclusiones

El desarrollo de este TFM ha estado marcado por una situación sin precedentes como ha sido la pandemia del Covid-19. Debido a esta peculiaridad, las pruebas se han enfocado desde un punto de vista menos práctico y más teórico, basándonos en un simulador que supone un punto de referencia muy bueno para posteriormente probarlo en sistemas reales.

El objetivo principal de conseguir un sistema capaz de posicionar nodos a través de la tecnología 6LoWPAN de manera no intrusiva con el protocolo RPL, se ha conseguido satisfactoriamente. Una red de nodos-sensores funcionaría de manera correcta mezclando nodos que calculan su posicionamiento y nodos sin posicionamiento, ya que se aprovechan los mensajes propios del protocolo RPL para la comunicación de la información relativa al posicionamiento.

Los tiempos de convergencia no pueden tomarse como concluyentes debido a que el simulador Cooja no siempre es estable y durante las pruebas hemos obtenidos tiempos muy dispares para pruebas configuradas exactamente igual.

La evolución del protocolo RPL para que pueda soportar posicionamiento con los nodos sensores que forman parte del sistema es muy satisfactoria y da lugar a pruebas mucho más exhaustivas donde poder medir de una manera fiable, tiempos de convergencia, precisión y consumo de baterías, parámetros que son decisivos a la hora de diseñar un sistema de posicionamiento *indoor*.

En cuanto al objetivo referente al diseño del algoritmo de trilateración para conseguir las coordenadas, no puede darse por definitivo ya que la simulación se ha realizado siempre sin ningún obstáculo entre los sensores, lo que significa que la señal RSSI que se obtiene es lineal a la distancia. Es sabido que esta situación no es real y será necesario ajustar el cálculo de conversión del RSSI en distancia, teniendo en cuenta el parámetro "path loss".

En cuanto a la topología final utilizando una cola MQTT como base de comunicación entre los nodos-sensores y el exterior, se ha observado que el protocolo

UDP funciona de una manera más eficiente, pero hay que tener presente que el simulador Cooja puede no ser fiable en cuanto a tiempos y por ese motivo no sería concluyente utilizar un protocolo u otro hasta realizar pruebas en un ecosistema real. En cualquier caso, es aconsejable utilizar un protocolo de aplicación ya existente como MQTT o CoAP antes que diseñar uno propio. MQTT es ideal para redes de tipo LoWPAN debido a su bajo consumo de datos de transferencia y su sencillez de uso.

5.2 Siguientes pasos

Para conseguir tener un sistema de posicionamiento *indoor* con 6LoWPAN fiable y con posibilidades de hacerse un hueco entre el resto de las tecnologías, es necesario realizar los siguientes pasos:

- Realizar pruebas reales y medir la eficiencia en cuanto a los tiempos de respuesta.
- Medir el consumo de batería de los nodos-sensores con una configuración de RPL donde los mensajes DIO no se optimizan con el tiempo.
- Medir la precisión en un entorno real teniendo en cuenta interferencias en las señales.
- Aplicar una capa de seguridad, cifrando los mensajes con DTLS para que un nodo intruso no pueda robar información del sistema.

Capítulo 6 - Introduction

Motivation

The industrial world has been undergoing a tremendous digital transformation for some years now. This transformation is known as Industry 4.0 and within this concept we can find a great variety of initiatives and lines of work related to the IoT.

One of these initiatives is the tracking of assets and tools used in a manufacturing plant in order to optimize their use and maintenance in the work orders assigned to operators. Frequently, tools and assets have a very high economic value and it is necessary to know their position and use in order to reduce their maintenance and consequently the associated costs.

Currently, there are several technologies that offer indoor positioning in an industrial environment, each using different techniques for position calculation.

The fact that I did not find many conclusive studies when using 6LoWPAN and RPL as one of the technologies in indoor positioning, and yet with a great projection in low energy consumption networks, made me decide to choose it as TFM in this master.

Objectives and working plan

The main objective of this TFM is to make a practical example of sensor node positioning in an industrial environment in order to study the behaviour of the use of 6LoWPAN and RPL in this case of use.

To achieve the main objective, it will be necessary to establish intermediate objectives or milestones that will help us with the study of the research:

Firstly, it will be necessary to study the functioning of the RPL protocol within the 6LoWPAN standard and to recover the RSSI (Received Signal Strentgh Indicator).

Secondly, the least intrusive way of modifying the RPL protocol will be designed to ensure that the different nodes of the system are able to send position information with no impact in network nodes that do not need to have this positioning but are nevertheless part of the deployed network and therefore part of the communications.

Thirdly, a trilateration algorithm will be studied as a function of the RSSI, the coordinates of 3 neighbouring nodes and the received power loss as indicated in different studies [1].

It will also be necessary to design a network topology that will expose the information of the coordinates of the nodes in a dashboard where a user will be able to consult in real time the positioning of these. A conceptual design of this topology can be seen in figure 1.1.

For the exposure of the data, an edge router will be used, which will be responsible for publishing the position of each of the sensor nodes in an mqtt broker.

For the dashboard with the information, the open platform of IoT Thingsboard will be used, which provides this functionality.

For the construction of the solution, the code of the Contiki operating system relative to the RPL protocol will be modified, focusing in this way on the network layer for the calculation of positions and the convergence of the network.

Tests will be designed with different movements of the nodes that compose the network. These tests will be carried out with the Cooja tool provided by the Contiki operating system that simulates the communication between the different nodes of the network, facilitating network convergence times.

Document structure

The first chapter of the report provides an introduction and a conceptual approach to the use case and the objectives to be achieved.

In a second chapter, the state of the art of technology in reference to indoor positioning is developed.

The third chapter is devoted to the development of the implemented solution, from its design to its implementation.

The design and execution of the tests are carried out in chapter four of this document.

Finally, the conclusions and next steps are detailed in the last chapter.

Chapter - Conclusions and future work

Conclusions

The development of this TFM has been marked by the unprecedented situation of the Covid-19 pandemic. Due to this peculiarity, the tests have been focused from a less practical and more theoretical point of view, based on a simulator that is a very good reference point for later testing in real systems.

The main objective of achieving a system capable of positioning nodes through 6LoWPAN technology in a non-intrusive way with the RPL protocol, has been satisfactorily achieved. A network of sensor nodes would work correctly by mixing nodes that calculate their positioning and nodes without positioning, since the messages of the RPL protocol are used to communicate information related to positioning.

The convergence times cannot be taken as conclusive due to the fact that the Cooja simulator is not always stable and during the tests we have obtained very different times for tests configured exactly the same.

The evolution of the RPL protocol so that it can support positioning with the sensor nodes that are part of the system is very satisfactory and has led to much more exhaustive tests where convergence times, precision and battery consumption can be measured reliably, parameters that are decisive when designing an indoor positioning system.

As for the objective regarding the design of the trilateration algorithm to obtain the coordinates, it cannot be considered definitive since the simulation has always been carried out without any obstacle between the sensors, which means that the RSSI signal obtained is linear to the distance. It is known that this situation is not real and it will be necessary to adjust the RSSI distance conversion calculation taking into account the "path loss" parameter.

As for the final topology using a MQTT queue as a communication base between the sensor nodes and the outside, it has been observed that the UDP protocol works in a more optimal way, but it must be taken into account that the Cooja simulator may not be reliable in terms of time and for that reason it would not be conclusive to use one

protocol or another until testing in a real ecosystem. In any case, it is advisable to use an existing application protocol such as MQTT or CoAP rather than designing one of your own. MQTT is ideal for LoWPAN type networks due to its low transfer data consumption and its simplicity of use.

Next steps

To achieve a reliable indoor positioning system with 6LoWPAN and with the possibility of making a gap between the other technologies, the following steps must be taken:

- Carry out real tests and measure the efficiency in terms of response times.
- Test the battery consumption of the sensor nodes with an RPL configuration where the DIO messages are not optimized within time.
- Measure the accuracy in a real environment taking into account signal interference.
- Apply a security layer, encrypting the messages with DTLS so that an sink node cannot steal information from the system.

BIBLIOGRAFÍA

[1] Goldoni, Emanuele & Savioli, Alberto & Risi, Marco & Gamba, Paolo. (2010). Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4. 71 - 77. 10.1109/EW.2010.5483396.

[2] F. Zafari, A. Gkelias and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2568-2599, thirdquarter 2019, doi: 10.1109/COMST.2019.2911558.

[3] Pu, Chuan Chin & Pu, Chuan-Hsian & Lee, Hoon-Jae. (2011). Indoor Location Tracking Using Received Signal Strength Indicator. 10.5772/10518.

[4] "Sony and Samsung resurrect ultra-wideband to improve location tracking" Available: https://www.engadget.com/2019-08-01-sony-and-samsung-resurrect-ultra-wideband-to-improve-location-tr.html [Último acceso: 06 06 2020]

[5] "RPL Basics" Available: https://anrg.usc.edu/contiki/index.php/RPL_UDP#RPL_Basics [Último acceso: 07 06 2020]

[6] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[7] "Mesh networking is blue" Available: https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/mesh/ [Último acceso: 09 06 2020]

[8] Lin, You-Wei & Lin, Chi-Yi. (2018). An Interactive Real-Time Locating System Based on Bluetooth Low-Energy Beacon Network †. Sensors. 18. 1637. 10.3390/s18051637.

[9] C. Yang and H. Shao, "WiFi-based indoor positioning," in IEEE Communications Magazine, vol. 53, no. 3, pp. 150-157, March 2015, doi: 10.1109/MCOM.2015.7060497.

[10] "IEEE 802.11ah" Available: https://en.wikipedia.org/wiki/IEEE_802.11ah [Último acceso: 09 06 2020]

[11] "Zigbee Alliance" Available: https://zigbeealliance.org/solution/zigbee/ [Último acceso: 10 06 2020]

[12] Borean, Claudio & Franceschinis, Mirko & Pastrone, Claudio & Spirito, Maurizio. (2013). On the performance of ZigBee Pro and ZigBee IP in IEEE 802.15.4 networks. 10.1109/WiMOB.2013.6673344.

[13] Uradzinski, M., Guo, H., Liu, X. et al. Advanced Indoor Positioning Using Zigbee Wireless Technology. Wireless Pers Commun 97, 6509–6518 (2017).

[14] "Indoor Localization with RFID" Available: https://www.infsoft.com/technology/positioning-technologies/rfid [Último acceso: 10 06 2020]

[15] "Indoor Positioning Using Li-Fi (VLC)" Available: https://www.infsoft.com/blog/indoor-navigation-indoor-positioning-and-location-based-services-using-vlc-visible-light-communication [Último acceso: 10 06 2020]

[16] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <https://www.rfc-editor.org/info/rfc6552>.

[17] Thomson, C., Romdhani, I., Al-Dubai, A., Qasem, M., Ghaleb, B., & Wadhaj, I. (2016). Cooja Simulator Manual. Edinburgh: Edinburgh Napier University

[18] Jonas Olsson (2014). 6LoWPAN demystified: Texas Instruments

[19] "What is Thread" Available: https://www.threadgroup.org/What-is-Thread [Último acceso: 16 06 2020]

[20] "Overview" Available: https://www.threadgroup.org/What-is-Thread/Overview [Último acceso: 16 06 2020]

[21] "Case Studies" Available: https://www.threadgroup.org/BUILT-FOR-IOT/Case-Studies [Último acceso: 16 06 2020]

[22] "Thingsboard Installations" Available: https://thingsboard.io/installations/ [Último acceso: 21 06 2020]

[23] "The official git repository for Contiki, the open source OS for the Internet of Things" Available: https://github.com/contiki-os/contiki [Último acceso 21 06 2020]

[24] "FAQ – Frequently Asked Questions" Available: http://mqtt.org/faq [último acceso 29 06 2020]

[25] "Automated guided vehicle" Available: https://en.wikipedia.org/wiki/Automated_guided_vehicle [Último acceso 29 06 2020]

[26] "Mobility of Nodes in Cooja" Available: https://anrg.usc.edu/contiki/index.php/Mobility_of_Nodes_in_Cooja [Último acceso 30 06 2020]

[27] "Download | Eclipse Mosquitto" Available: https://mosquitto.org/download/ [Último acceso 01 07 2020]

APÉNDICES

Apéndice A - Preparación del entorno de desarrollo

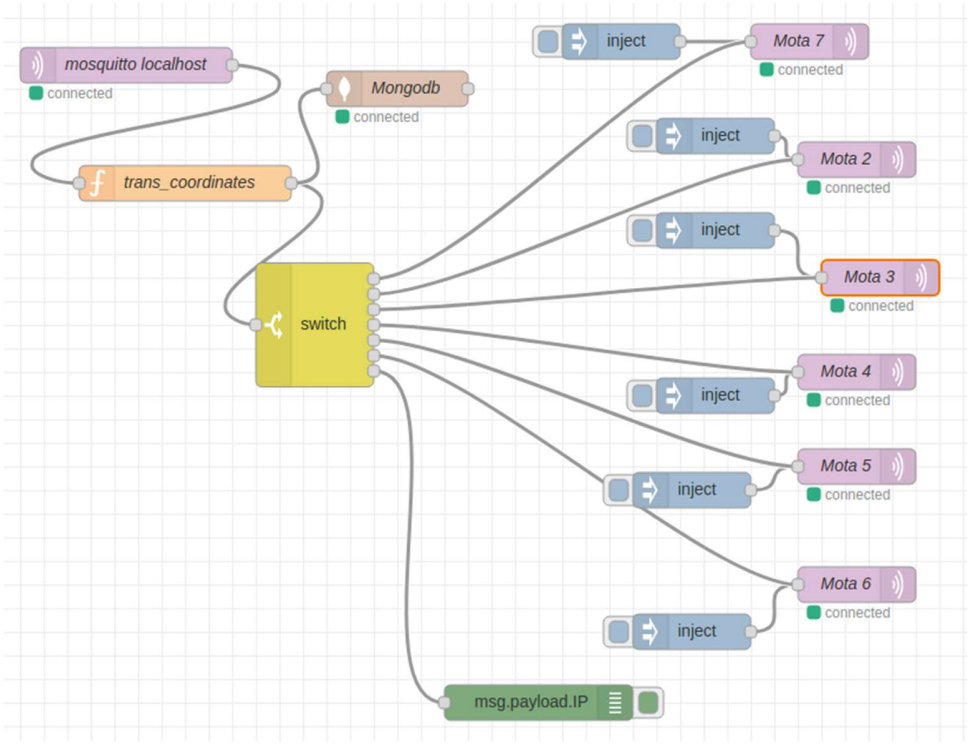
Configuración de Node-red

En este ejemplo se utiliza la versión 1.0.6 de node-red. El objetivo de usar este framework en esta prueba es que actúe de proxy entre el simulador Cooja y la plataforma de IoT cumpliendo dos funciones: la primera publicar las posiciones de cada mota con su correspondiente token de seguridad. La segunda, almacenar las posiciones en una base de datos mongodb junto con un campo timestamp que nos indicará el historial del nodo.

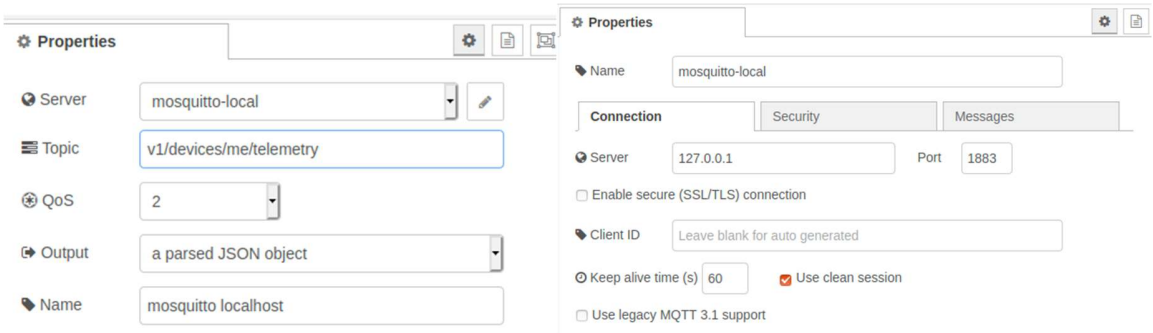
La configuración del flujo (véase Figura A-1) en node-red utiliza los siguientes componentes:

- Mosquitto localhost: Este nodo, de tipo *mqtt in*, se suscribe al topic "v1/devices/me/telemetry" en el broker mosquitto localhost (véase figura A-2).
- Trans_coordinate: Este nodo, de tipo *function*, se encarga de transformar las coordenadas para que el componente en la plataforma Thingsboard las sitúe correctamente. El rango de las coordenadas de la plataforma Thingsboard está entre 0 y 1, sin embargo, en el simulador Cooja se representan en proporción 1:1, así que fijamos el entorno en el simulador entre 0 y 100 y dividimos la coordenada entre 100. Este cálculo se puede hacer también en el código del nodo, pero el tipo de la coordenada es un entero para que ocupe menos. También se encarga de añadir el timestamp al dato (véase Figura A-3).
- Switch: Distribuye, en función de la IP, la información de la posición a cada dispositivo configurado en la plataforma Thingsboard. En caso de no conocer la IP escribe el payload en la consola de node-red.

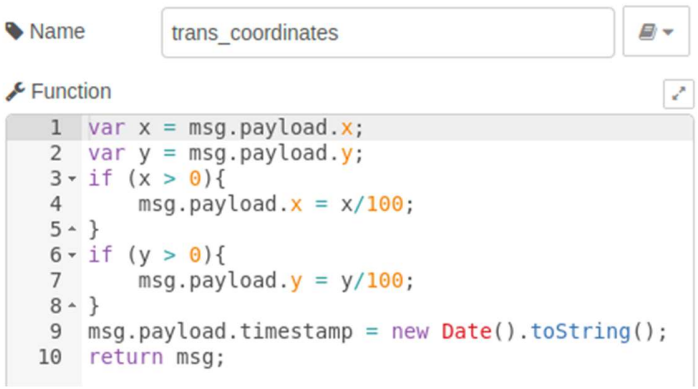
- Mongodb: Guarda la información de la posición en una colección de mongodb. La base de datos se llama *positions* y la colección *list_positions*.
- Mota 2-7: Cada nodo, tipo *mqtt out*, publica en el broker de Thingsboard la posición de una mota del simulador Cooja. El topic es el mismo para todos los nodos, ya que Thingsboard diferencia cada dispositivo a través del token único que asigna a cada dispositivo y que se indica en el campo *username* (véase Figura A-4).



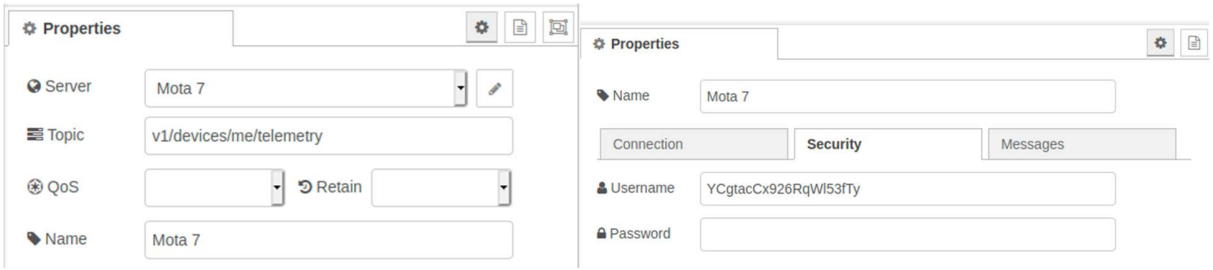
A-1: Configuración de node-red en la prueba 4.4.2



A-2: Configuración de mosquitto en node-red.



A-3: Transformación de coordenadas para Thingsboard.

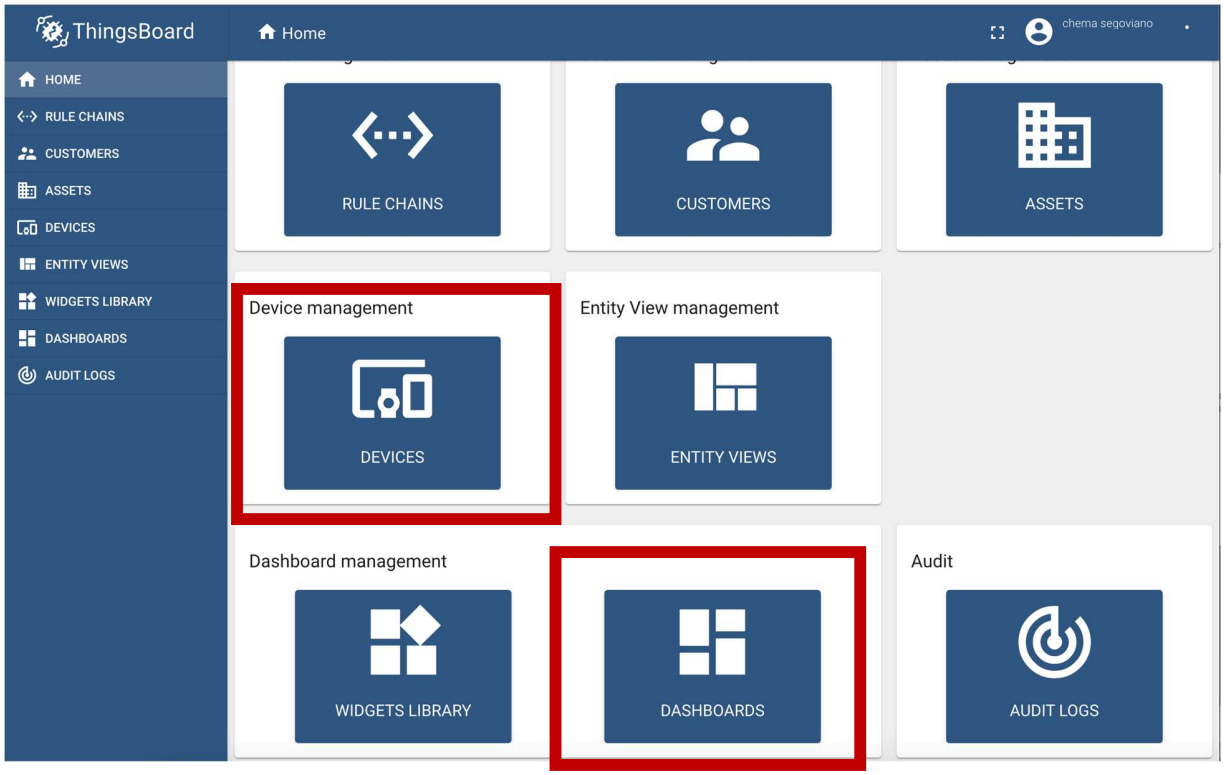


A-4: Configuración para publicar en MQTT de Thingsboard..

Configuración de Thingsboard

Thingsboard es una plataforma abierta de IoT que nos ofrece la posibilidad de descargar una versión *community* para trabajar *on-premise* de manera gratuita. Si deseamos utilizar la versión cloud, existe la posibilidad de utilizar una versión *trial* que tiene una duración de 30 días y limitación en algunas funcionalidades limitadas [22]. Para conseguir una prueba completa desde el simulador Cooja hasta un dashboard publicado en internet se ha escogido la versión cloud.

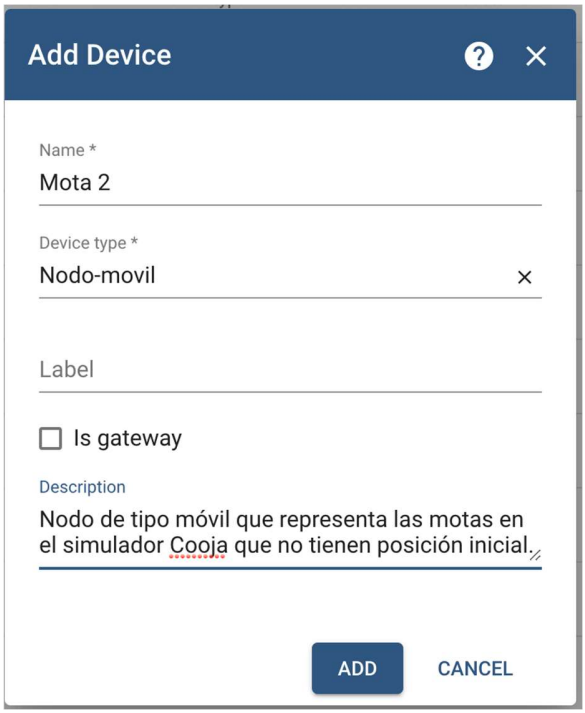
Dentro de la plataforma Thingsboard se utilizan dos utilidades de ésta. *Devices* y *Dashboards* (véase figura A-5). *Devices* representa los dispositivos que forman parte de nuestro sistema, en nuestro caso serán 6 nodos sensores de tipo móvil. *Dashboard* es una utilidad donde podemos crear un entorno personalizado con una serie de componentes que nos ofrece la plataforma. En nuestro caso, utilizamos el mapa.



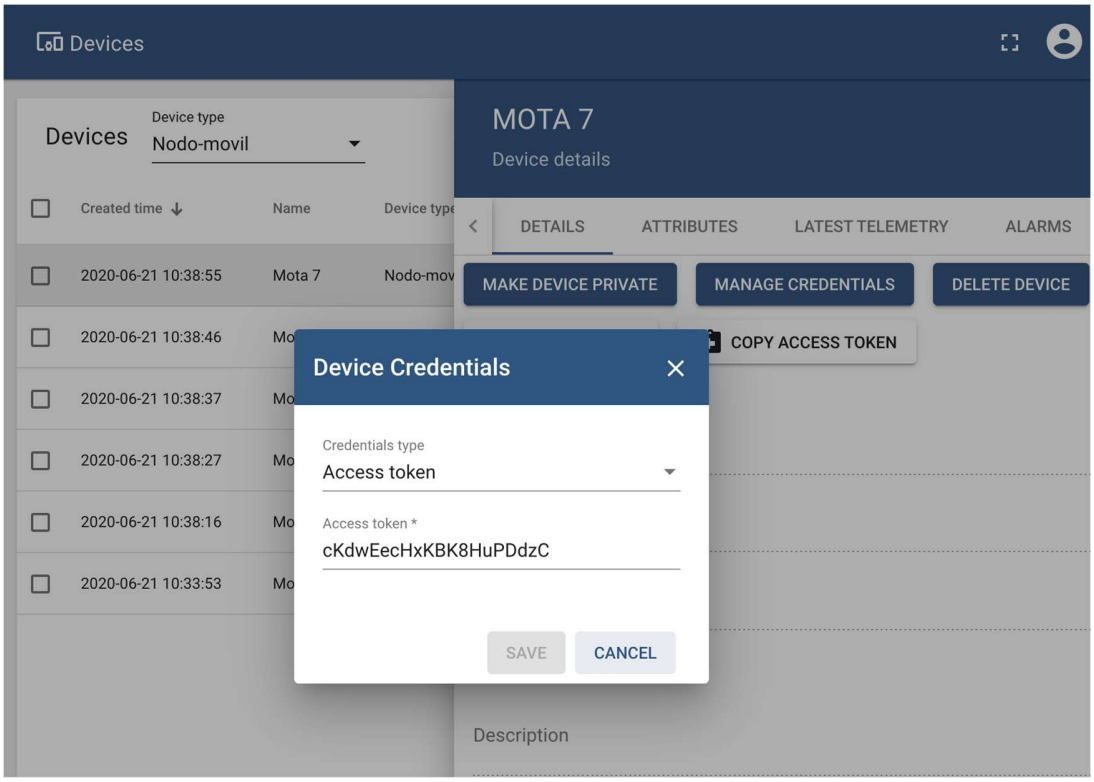
A-5: Panel principal de Thingsboard.

En la utilidad *Devices* creamos los seis nodos (véase figura A-6) que forman parte de nuestra prueba y que tienen cada uno un *token* que les identifica de manera univoca. El token que nos asigna la plataforma se puede modificar y también podría utilizarse un certificado que podría ir configurado en el nodo sensor. En nuestro caso, utilizamos un *token* (véase figura A-7) que hacemos coincidir con el campo username utilizado para cada mota en la configuración explicada en *node-red*.

En nuestro caso particular, para que el dashboard pueda consultarse desde el exterior sin credenciales, estos nodos deben declararse públicos.



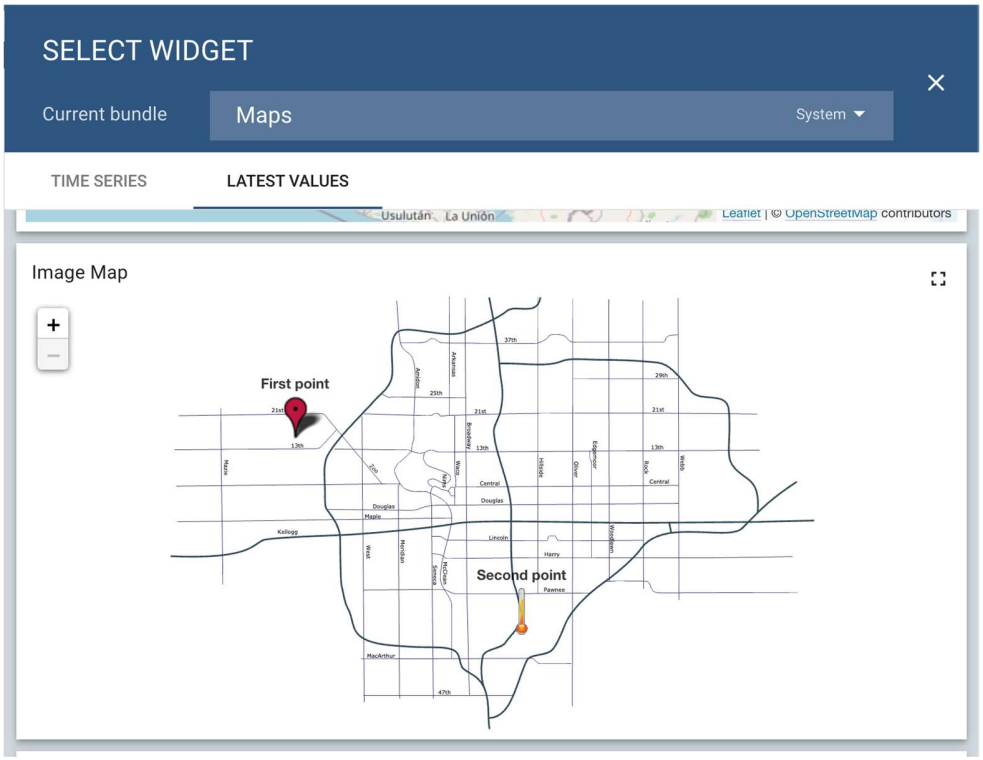
A-6: Creación de un nodo en Thingsboard.



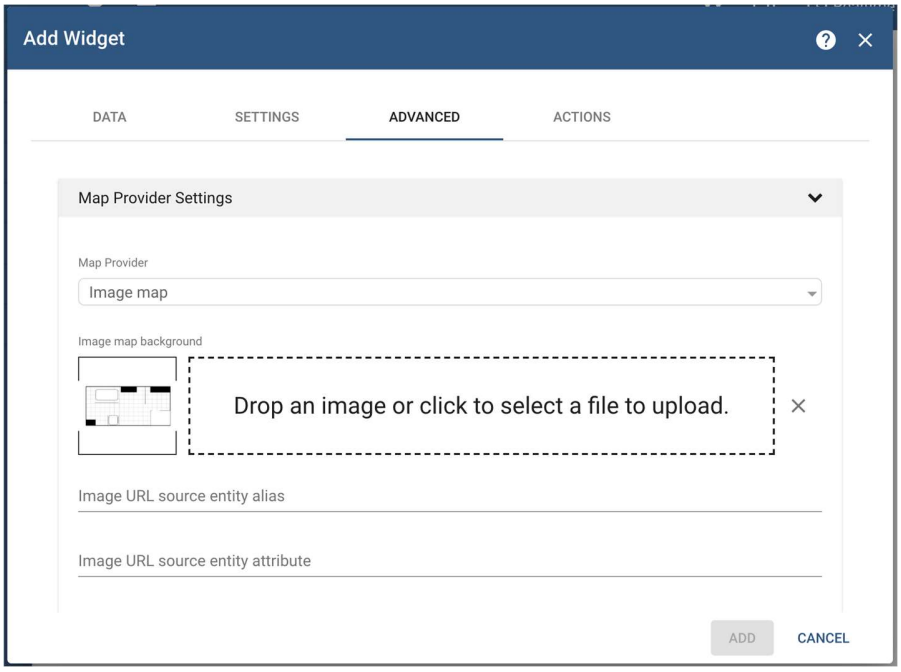
A-7: Configuración del token de un nodo en thingsboard.

Para configurar el dashboard seguimos los siguientes pasos:

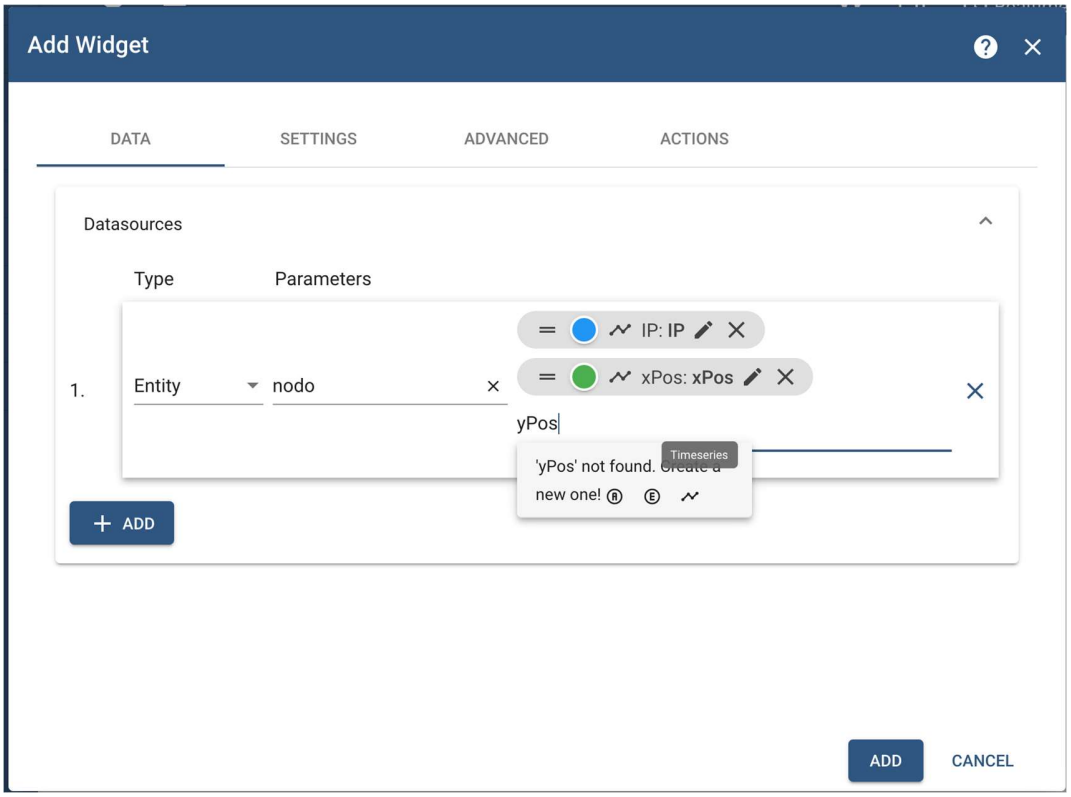
- Crear un dashboard en el panel de dashboard.
- Abrir el dashboard creado y añadir un widget de tipo *Maps* y dentro de esta opción, elegir *Image Map* (véase figura A-8).
- Añadir la imagen del mapa que utilizamos para la simulación en la pestaña ADVANCED del widget (véase Figura A-9).
- En la pestaña DATA haremos referencia a los dispositivos creados y a los campos que vienen definidos en formato JSON (véase figura A-10). Es importante que los campos se creen como *Timeseries*. Además, cada atributo debe corresponderse con su nombre en el mensaje JSON que se publica en la cola MQTT. Esta correspondencia se realiza entre los campos *key* y *label* de un atributo en la pestaña DATA (véase Figura A-11).
- En la pestana ADVANCED hay que configurar la información que mostrará el nodo cuando hagamos click sobre él (véase figura A-12).



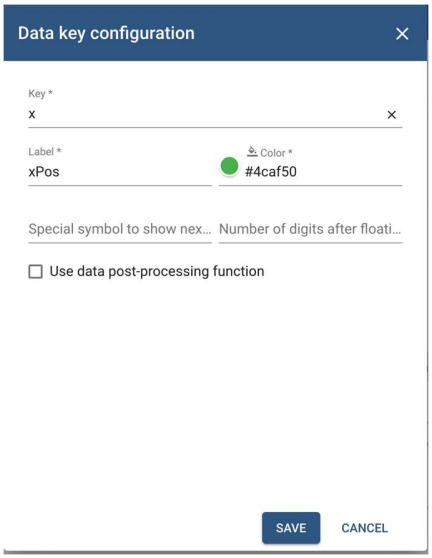
A-8: Selección del widget Maps - Image Map en Thingsboard.



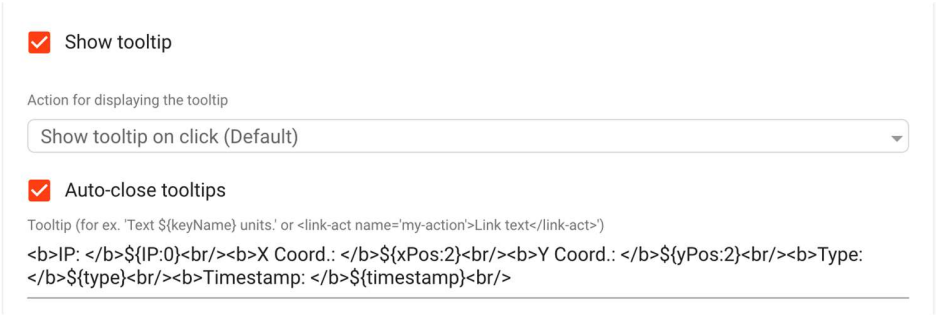
A-9: Pestaña ADVANCED del widget Maps en Thingsboard.



A-10: Añadir atributos a los dispositivos de un tipo.



A-11: Configuración de un atributo de un nodo.



A-12: Configuración del tooltip de un nodo en el mapa.