Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Estudio del módulo DWM1001 de Decawave para la medida de rangos en UWB

Autor: Pablo Amores Cuenca

Tutor: José Manuel Villadangos Carrizo

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Estudio del módulo DWM1001 de Decawave para la medida de rangos en UWB

Autor: Pablo Amores Cuenca

Tutor: José Manuel Villadangos Carrizo

TRIBUNAL:

Presidente: Juan Jesús García Domínguez

Vocal 1º: Ana Isabel De Andrés Rubio

Vocal 2º: José Manuel Villadangos Carrizo

FECHA: 30/06/2021

ÍNDICE

Índice de Figuras

Índice de Tablas

1. RESUMEN

Con este Trabajo de Fin de Grado se pretende estudiar una nueva forma de posicionamiento en tiempo real para interiores mediante la tecnología *Ultra Wide Band* (UWB). Se estudiará y se desarrollará el módulo DWM1001 creado por *Decawave,* y se comprobarán realizando distintas pruebas los beneficios e inconvenientes que tiene como pueden ser cobertura, error en la localización, frecuencia de adquisición, etc., para saber si tendrá un gran crecimiento en el futuro en distintas aplicaciones en Sistemas de Localización en Tiempo Real (RTLS).

2. ABSTRACT

This End of Grade Project aims to study a new form of real-time indoor positioning using Ultra Wide Band (UWB) technology. The DWM1001 module created by Decawave will be studied and developed, different tests are carried out to see the benefits and disadvantages, such as coverage, location error, acquisition frequency, etc., to see if it will have a great growth in the future in different applications in Real-Time Location Systems (RTLS).

3. RESUMEN EXTENDIDO

En los últimos años, debido a que los sistemas de posicionamiento se han convertido en esenciales en la mayoría de los dispositivos, surge la necesidad de estudiar nuevos sistemas para garantizar la localización en interiores. Actualmente con los principales métodos de posicionamiento como GPS es complicado posicionar en interiores ya que no dispone de suficiente cobertura. Por ello se han tenido que desarrollar nuevos sistemas de localización en tiempo real (RTLS) basados en Ultra Wide Band (UWB) o en español Banda Ultra Ancha. UWB es una tecnología cada vez más presente ya que además es capaz de utilizar bandas de frecuencia asignadas a otros servicios y funcionar con ellos sin causar interferencias perjudiciales.

Decawave ha trabajado en esto y ha creado un módulo UWB muy económico y de un reducido error lo que lo hace una solución bastante útil, con un software integrado sencillo capaz de realizar mediciones de ubicación bastante precisas con errores menores de 20 centímetros, llamado DW1000. A partir de este transceptor se desarrolla el módulo DWM1001 el cual es el que se estudiará y se tratará de desarrollar el software para añadir alguna funcionalidad adicional.

Primero se verán los conceptos teóricos que se necesitarán para el desarrollo del proyecto como la tecnología Ultra Wide Band (UWB), la descripción y funcionamiento del módulo DWM1001, la forma de interactuar con el módulo y de desarrollar nuevas funcionalidades de firmware y las normas y las limitaciones para el desarrollo e instalación de un sistema de localización en tiempo real.

Una vez estudiados los conceptos teóricos, se realizará una instalación del sistema en el sector Oeste y en la sala N21 de la segunda planta de la Escuela Politécnica Superior de la Universidad de Alcalá y se realizarán varias pruebas para comprobar el funcionamiento y los errores cometidos por un sistema de localización en tiempo real. Por último, se desarrollará alguna funcionalidad al firmware integrado de serie como es el de un detector de caídas y se estudiarán posibles aplicaciones en las que se puede aplicar estos sistemas.

Los módulos se pueden configurar como *anchors*, que serán las balizas con una posición fija, como *tags*, los cuales serán los módulos a posicionar y que se moverán por la red. Por último, se pueden configurar como *bridge* el cual se puede conectar a una *Raspberry Pi* y así conectar el sistema a la red. Para configurar las balizas se puede hacer a través de *Bluetooth* con cualquier dispositivo Android mediante la aplicación llamada *DRTLS Manager*, intercambiando mensajes conectando el PC a través de UART mediante comandos *Shell* o conectándose a la dirección IP de la *Raspberry Pi* que actúa como puerta de enlace donde se sitúa una página web con la que se puede ver la red de localización formada.

Un ejemplo del sistema formado es como el mostrado en la Figura 1.

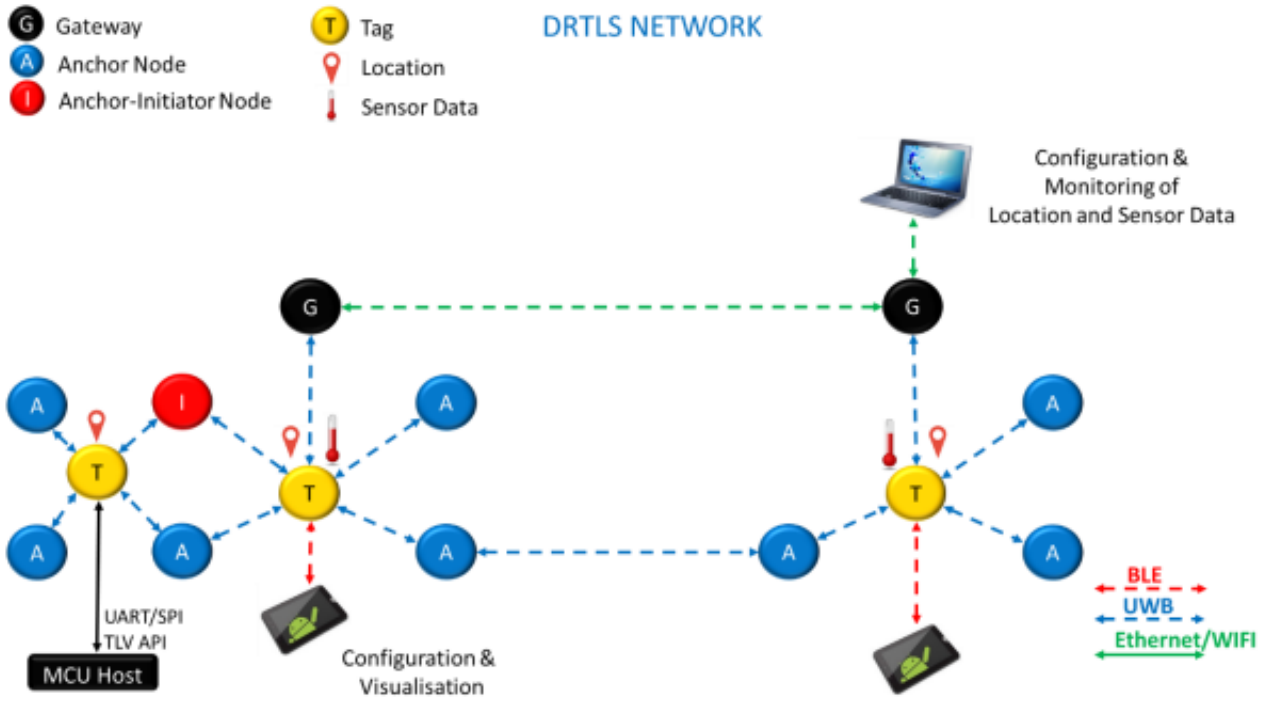


Figura 1. Ejemplo de red de localización a implementar [7]

4. PALABRAS CLAVE

Palabras clave: posicionamiento, anchor, tag, UWB, DWM1001.

5. DESARROLLO TEÓRICO

5.1. Introducción

En este capítulo se va a realizar una explicación de los conceptos teóricos necesarios para llevar a cabo el trabajo. Primero se explica en que consiste la tecnología utilizada para el funcionamiento del sistema que es el *Ultra Wide Band* (UWB) o en español Banda Ultra Ancha. Luego se explicará en que consiste y el funcionamiento del sistema utilizado para la realización del trabajo, el módulo y la tarjeta de desarrollo creada por *Decawave*, DWM1001. Más adelante se desarrollará la forma de interactuar con el módulo y de desarrollar nuevas funcionalidades de *firmware*. Por último, se introducirán los pasos, las normas y las limitaciones para el desarrollo e instalación de un sistema de localización en tiempo real.

5.2. Ultra Wide Band (UWB)

La tecnología *Ultra Wide Band* (UWB) es la utilizada por *Decawave* para que se comuniquen los dispositivos. UWB se define como aquella transmisión de radio que ocupa un ancho de banda mayor al 25 % de la frecuencia central (ecuaciones 1 y 2) o cuyo ancho de banda absoluto sea mayor de 500 MHz (ecuación 3). Hay dos diferencias que hace al UWB inconfundible respecto a los sistemas de banda estrecha y banda ancha. Uno de ellos es como su propio nombre indica el ancho de banda que utiliza un ancho de banda mucho mayor que cualquier otra tecnología. La otra gran característica que lo diferencia de los demás es que UWB transmite pulsos en banda base muy cortos, normalmente de duración menores de un nanosegundo, en vez de señales sinusoidales. Estos pulsos cortos hacen que esta tecnología sea muy prometedora para aplicaciones de baja potencia y posicionamientos precisos en interiores, en la Figura 2 se puede ver un ejemplo de pulso en el dominio del tiempo. En la Figura 3 se observa el espectro de frecuencias de varias tecnologías y se ve la gran diferencia de ancho de banda que existe entre ellas.

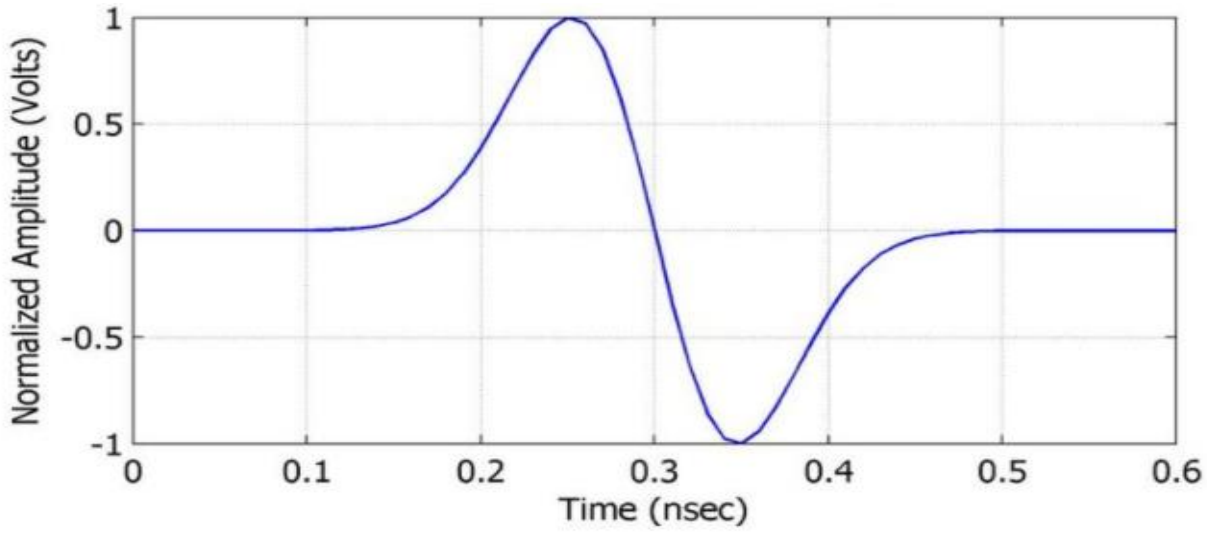


Figura 2. Pulso UWB en dominio del tiempo [20]

$$f_c = \frac{f_H + f_L}{2} \quad (1)$$

$$\frac{BW}{f_c} = \frac{f_H - f_L}{\frac{f_H + f_L}{2}} = 2\,\frac{f_H - f_L}{f_H + f_L} \geq 0.25 \quad (2)$$

$$BW = f_H - f_L \geq 500\ MHz \quad (3)$$

Siendo:

-f_H: frecuencia máxima a la que la densidad espectral de potencia de la transmisión UWB es de -10 dB con relación a la frecuencia de la transmisión UWB máxima.

-f_L: frecuencia inferior a la que la densidad espectral de potencia de la transmisión UWB es de -10 dB con relación a la frecuencia de la transmisión UWB máxima.

-f_c: frecuencia central del ancho de banda a -10 dB.

-BW: ancho de banda absoluto

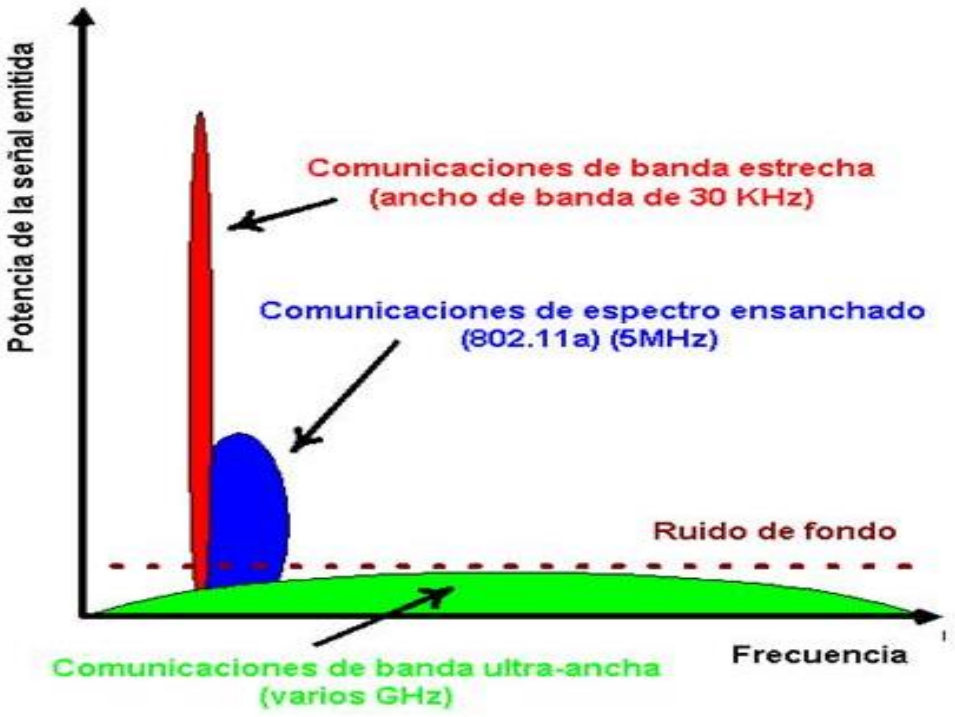


Figura 3. Espectros de frecuencias [1]

UWB se crea para mejorar características de otras tecnologías inalámbricas como aumentar la velocidad de transmisión, aumentar la autonomía de funcionamiento y aumentar la seguridad. Además, otras características de esta tecnología tan apreciada son su alta precisión y la ausencia de distorsión multi trayectoria.

Las características más destacadas por las que se utiliza esta tecnología son las siguientes:

- **Alta velocidad de datos**: El gran ancho de banda hace que los dispositivos UWB puedan alcanzar velocidades de datos altas.
- **Baja potencia**: La duración de la batería de los dispositivos será mayor debido a la baja potencia transmitida y a que el consumo es muy bajo debido a que la arquitectura es muy simple y a que transmite pulsos muy cortos constantemente en vez de transmitir ondas moduladas continuamente.
- **Comunicaciones más seguras**: Los dispositivos UWB puede coexistir con otros usuarios que trabajen en las mismas frecuencias. La baja emisión y la naturaleza impulsiva hacen que las comunicaciones sean más seguras. Las probabilidades de detección y de interferencias son muy bajas debido a que utiliza una energía muy baja por cada banda de frecuencia y por ello es una tecnología adecuada para aplicaciones de alta seguridad.
- **Comunicaciones de acceso múltiple**: Debido al gran ancho de banda, los sistemas UWB pueden dar acceso a varios dispositivos.
- **Efectos multitrayecto**: Al utilizar pulsos muy cortos la mayoría de las reflexiones de señal no se superponen al pulso original. Debido a esto las reflexiones multitrayecto como la presencia de sombras o reflejos de paredes y objetos, se pueden resolver y por tanto UWB es perfecto para aplicaciones de posicionamiento de alta resolución.

5.3. Descripción y Funcionamiento general del sistema DWM1001

El módulo DWM1001 está basado en el módulo transceptor UWB DW1000 IC de *Decawave*, en el *system on a chip* multiprotocolo, nRF52832 de *Nordic Semiconductor*, en el sensor de movimiento STM LIS2DH12TR y en un circuito regulador de energía, mostrados en la Figura 4. Además, cuenta con un *Firmware* de la pila de posicionamiento y redes (PANS) que permite crear Sistemas Precisos de Localización en Tiempo Real (RTLS) basados en Banda Ultra Ancha (UWB).

5.3.1. Transceptor DW1000

El transceptor UWB DW1000 es el elemento clave y es el encargado de implementar la red de *anchors* y realizar los intercambios TWR con los *tags* para que así cada *tag* calcule su propia ubicación. El DW1000 cuenta con un cristal de referencia de 38.4 MHz y es capaz de trabajar en seis bandas de frecuencia diferentes entre 3.5 GHz y 6.5 GHz con dos anchos de banda diferentes 500 MHz y 900 MHz y tres velocidades de datos diferentes 110 kbps, 850 kbps y 6.8 Mbps. Sin embargo, la implementación del DWM1001 limita todo esto a una sola configuración que tiene una frecuencia central de 6.5 GHz y un ancho de banda de 500 MHz.

Este transceptor tiene una memoria *Always-On* (AON) la cual se utiliza para mantener los datos de configuración, aunque el sistema se encuentre en estados de baja potencia, este uso de la memoria puede ser configurable por el usuario.

Los sensores de voltaje y temperatura incluidos en el módulo permiten al usuario *host* leer sus valores del pin VDDAON y la temperatura interna del DW1000.

5.3.2. Microprocesador Bluetooth nRF52832

El módulo de *Nordic* integra un chip inalámbrico de 2.4 GHz de potencia que proporciona la conectividad *Bluetooth* utilizada para la configuración del sistema y un microprocesador *ARM Cortex-M4* con memoria flash de 512 kB y una memoria RAM de 64 kB y que ejecuta el *firmware* y proporciona la funcionalidad de habilitación RTLS. El módulo DWM1001 puede configurarse para comportarse como *anchor*, un nodo fijo del sistema, *tag*, un nodo móvil que se moverá por el sistema y es el nodo el cual hay que localizar y como *bridge* o nodo puente para conectar el sistema a la red externa.

5.3.3. Sensor de movimiento STM LIS2DH12TR

Este sensor de movimiento es un acelerómetro de tres ejes de alto rendimiento y baja potencia. El sensor se puede configurar por el usuario con distintas escalas ±2g/±4g/±8g/±16g y es capaz de medir aceleraciones con datos de 1 Hz hasta 5.3 kHz. Su funcionamiento está garantizado para un rango de temperatura de -40 ºC a 85 ºC.

5.3.4. Regulador de tensión

Consiste en un regulador en modo conmutación. Es un convertidor de voltaje reductor ya que el voltaje de entrada puede estar en el rango de 2.8 V a 3.6 V y la salida es de 1.8 V el cual es el voltaje necesario para alimentar el módulo DW1000.

5.3.5. Firmware a bordo

Los módulos vienen precargados con el *firmware* integrado que les proporciona la funcionalidad de mediciones de rango bidireccional (TWR) y permite crear Sistemas Precisos de Localización en Tiempo Real (RTLS) basados en *Ultra Wide Band* (UWB).

La configuración del sistema se puede conseguir a través de *Bluetooth* utilizando la aplicación para *Android Decawave DRTLS Manager* o mediante una conexión UART o SPI a través de un *host* externo. El módulo se monta en una PCB y es llamado DWM1001-DEV.

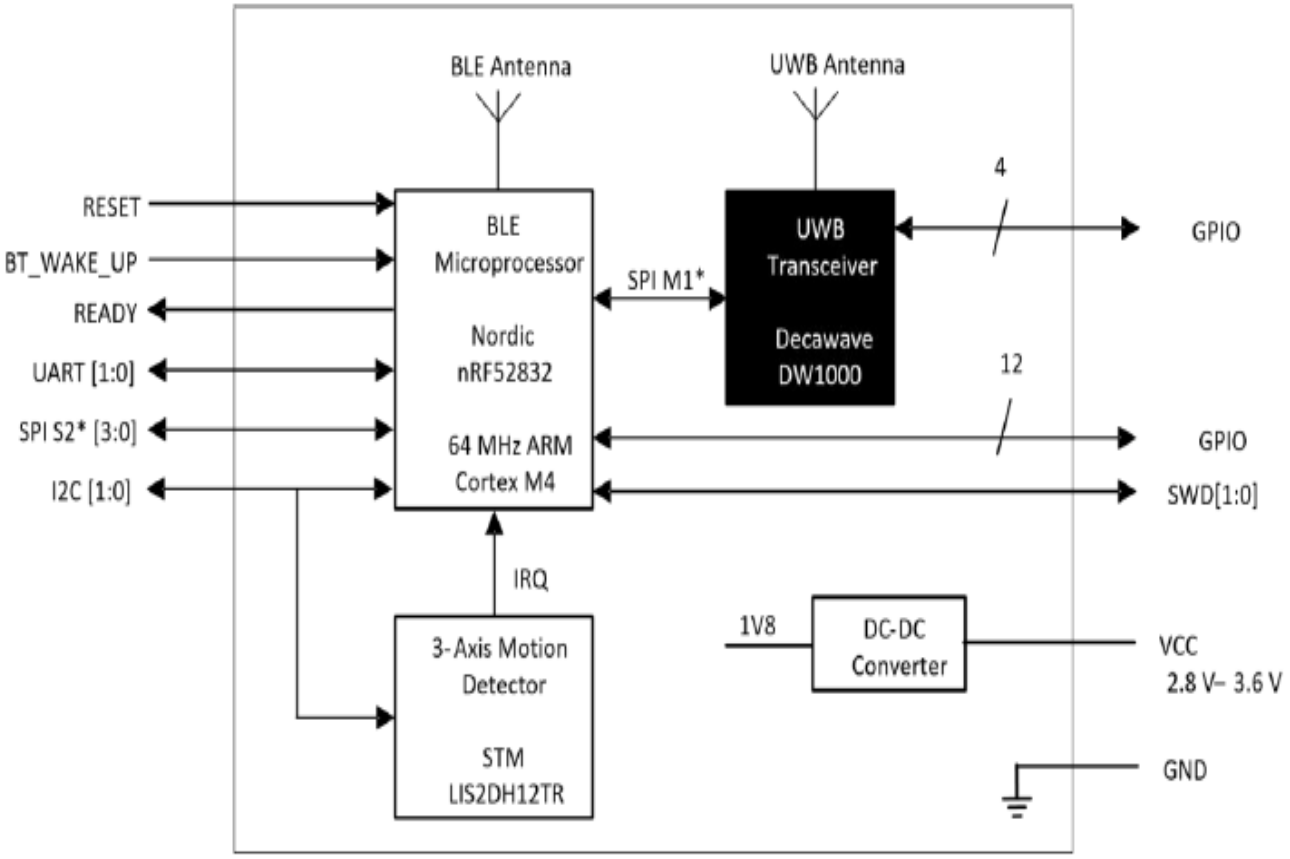


Figura 4. Diagrama de bloques del módulo DWM1001 [2]

Además de las características principales comentadas, el DWM1001-DEV tiene otras características como, por ejemplo: lleva incorporado J-link a bordo lo que permite depurar y flashear a través de USB, se puede acceder a todos los GPIOS a través de las cabeceras a bordo.

Si se suelda un encabezado de pines a la tarjeta de desarrollo es posible conectarla a una *Raspberry pi* y así poder interactuar con ella y formar una puerta de enlace entre la red y el sistema de localización. Cuenta con un circuito de carga de la batería y cuenta con una API de

software para la personalización de aplicaciones. Además, la PCB cuenta con dos botones, uno es el *Reset* que se utiliza para reiniciar el sistema, y el otro se encuentra conectado al pin BT_WAKE_UP y por tanto activa la función *Bluetooth* cuando un *tag* está en modo de bajo consumo.

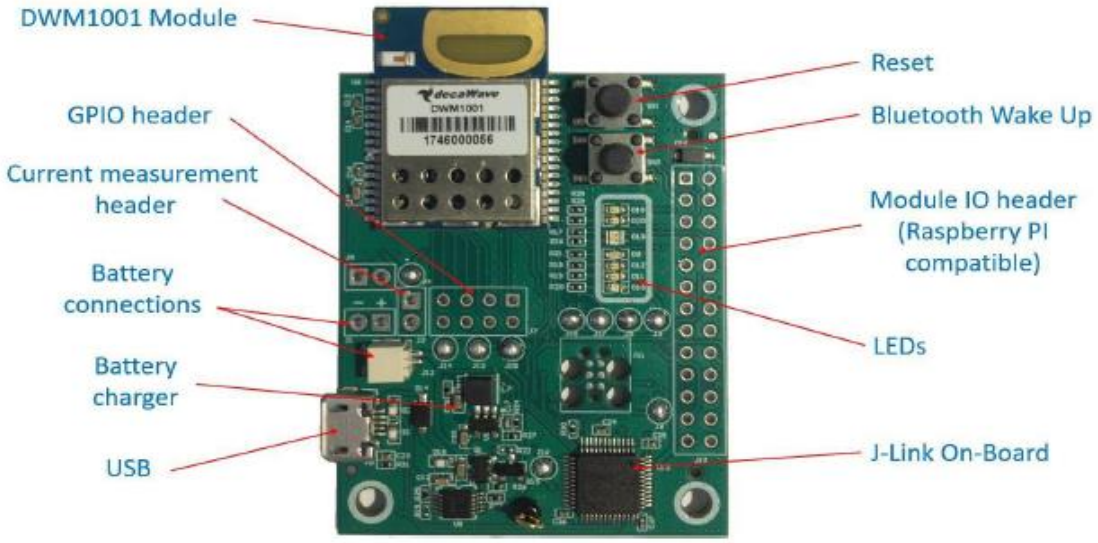


Figura 5. Componentes de Tarjeta de desarrollo DWM1001-DEV [5]

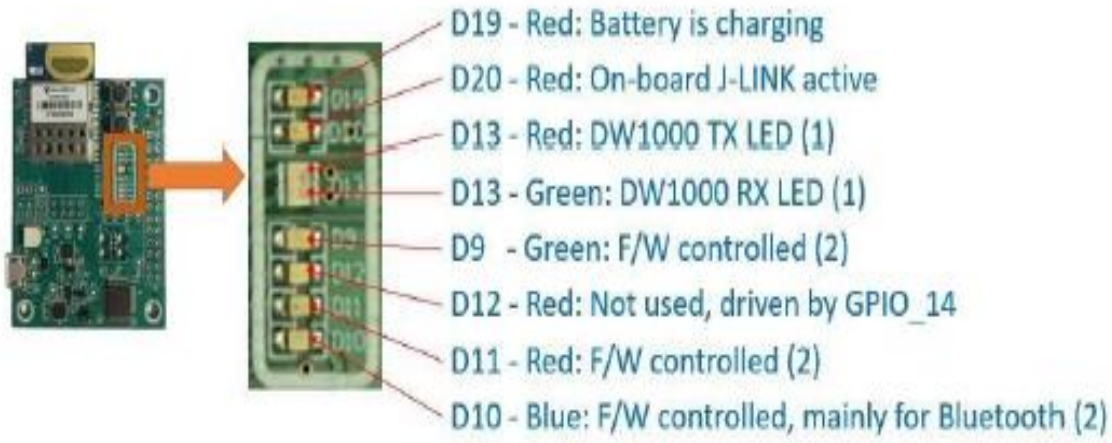


Figura 6. Leds de la tarjeta de desarrollo [5]

La tarjeta de desarrollo incorpora una serie de LED que serán útiles para indicar el funcionamiento de los distintos eventos del sistema. En la Figura 6 se muestran estos leds con su respectivo funcionamiento con el firmware precargado proporcionado por *Decawave*.

- **Led D19**: es de color rojo y se enciende cuando la batería o pila está cargando. El led se apagará si no hay fuente de alimentación USB o *Raspberry Pi* conectada a la vez que la pila o si la carga es completa.

- **Led D20:** es de color rojo y se enciende cuando el J-Link incorporado en la tarjeta está activo.
- **Led D13:** está compuesto por dos leds, el de la parte superior es de color rojo e indica cuando hay una transmisión a través del puerto UART. El de la parte inferior es de color verde y se enciende cuando se recibe un dato a través de la interfaz UART.
- **Led D12:** este led es de color rojo y no es utilizado en este *firmware*, está conectado a través de GPIO_14. Se puede utilizar por el usuario para encenderlo o apagarlo cuando suceda algún evento concreto.
- **Leds D9, D11 y D10:** son los que indican el estado del firmware y se ven explicados en la Tabla 1.

Tabla 1. Funcionamiento de los Leds [5]

D9 (VERDE)	D11 (ROJO)	D10 (AZUL)	Función	Descripción
Parpadeo rápido	Parpadeo rápido	Parpadeo rápido	Arranque	Los leds parpadean dos veces
Parpadeo rápido	Parpadeo rápido	Parpadeo rápido	Actualización de firmware en curso	Los leds alternan
Encendido	Encendido		Estado del modo	Modo UWB pasivo o apagado
Parpadeo rápido	Parpadeo rápido			No se detecta señal UWB durante más de 8 s en el modo *anchor*
Parpadeo lento	Parpadeo lento			No se detecta señal UWB durante más de 8 s en el modo *tag*
Encendido				Modo de baja potencia en *tag* activado
Apagado				Modo de baja potencia de *tag*: durmiendo
Encendido			Estado MAC	Dispositivo conectado
Parpadeo lento				Iniciador *anchor* conectado
Parpadeo rápido				Comunicación UWB detectada
Apagado				Baja potencia, no se detecta señal durante más de 6 s
		Encendido	Estado Bluetooth	Bluetooth conectado
		Apagado		Bluetooth desconectado
	Encendido		Estado de medición	UWB TX/RX activo
	Apagado			Inactivo

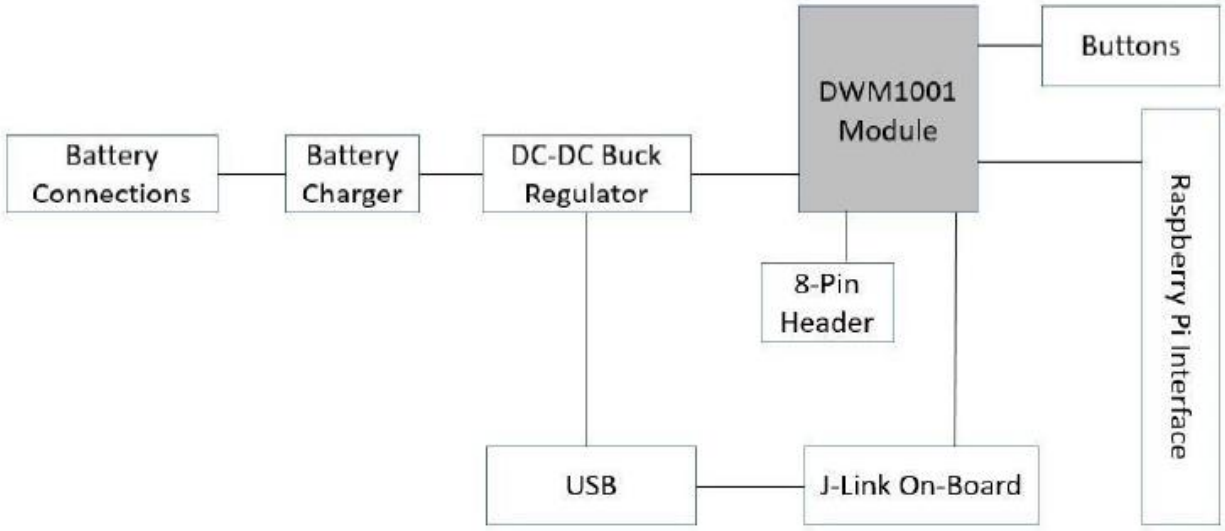


Figura 7. Diagrama de bloques de la tarjeta de desarrollo [5]

En la Figura 5 y Figura 7 se muestra la tarjeta de desarrollo del módulo DWM1001, en la primera se muestra una visión general de los componentes principales de la tarjeta. En la segunda figura se muestra el diagrama de bloques de la tarjeta.

La conexión USB tiene dos funciones, una alimentar a la tarjeta de desarrollo y la otra permite flashear y depurar el firmware del módulo DWM1001 a través del J-Link incorporado en la tarjeta de desarrollo.

Hay varias formas de alimentar la tarjeta de desarrollo que cumplen con el requisito de alimentación de voltaje entre 3.6V a 5.5V, a través de una fuente de alimentación por USB, a través de una pila o a través de una *Raspberry Pi* que alimenta el módulo y los demás dispositivos de la tarjeta. Además, es posible cargar la pila si se alimenta con una fuente de alimentación por USB o con la *Raspberry Pi*.

5.3.6. Conexionado de pines

La tarjeta de desarrollo cuenta con unas conexiones de pines donde la numeración se puede ver en la Figura 8. Son un total de 34 pines los cuales son compatibles con *Raspberry Pi* y así poder intercambiar datos con ella.

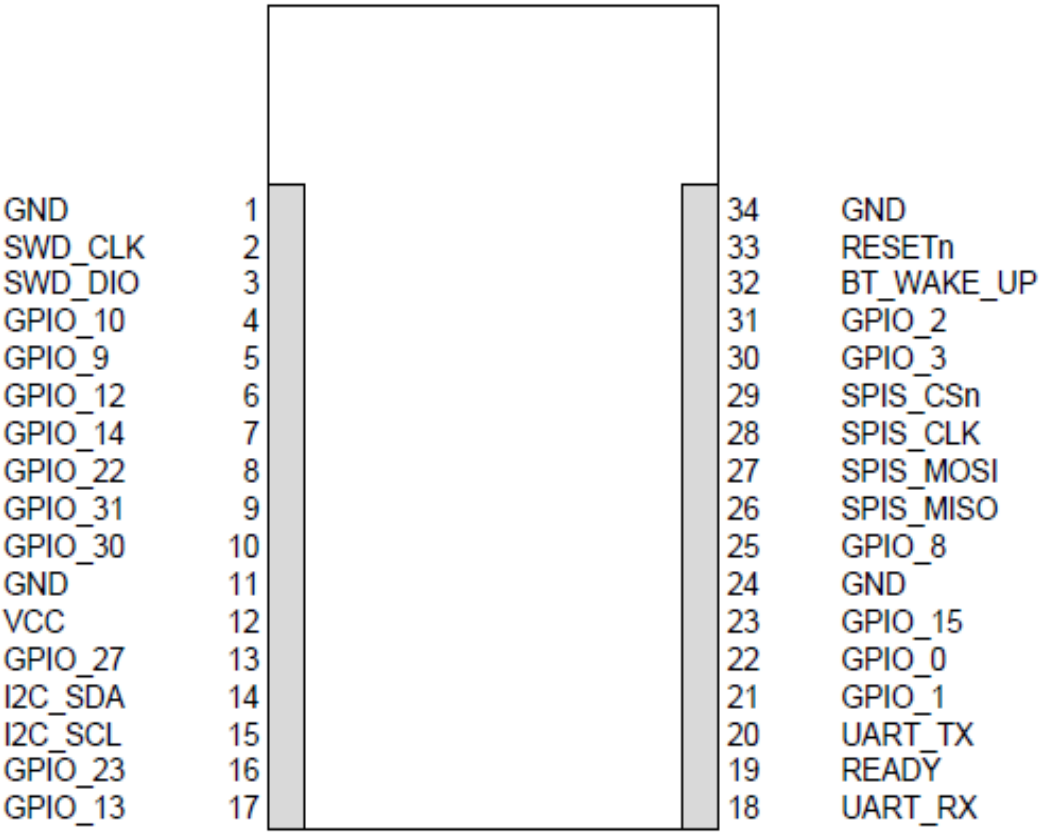


Figura 8. Conexionado de pines [6]

5.3.7. Consumo de energía de DWM1001

La Figura 9 muestra el consumo de energía de un módulo DWM1001 montado sobre una tarjeta de desarrollo DWM1001-DEV configurado como *tag* cuando es utilizado para hacer mediciones con el rango bidireccional (TWR) con 3 *anchor*s.

Primero se observa una fase llamada *Deep Sleep* donde solo está en funcionamiento el RTC y el acelerómetro y por tanto el consumo de corriente es muy bajo, de unos 12 µA. Cuando el *tag* "despierta", estado *WakeUp* consume 6 mA, luego pasa al estado de *idle* donde consume 13 mA para esperar a intercambiar información con los *anchors*. El *tag* trasmite las tramas a los *anchors* donde llega a un pico de consumo de 111 mA (82 mA de media), luego espera a que los *anchors* le respondan. El *tag* recibe respuesta de cada uno de los anchors, en este caso son 3 y por ello se observan tres picos donde se reciben las tramas, después de realizar la transmisión. Aquí es donde más consumo se observa alcanzando un pico máximo de 154 mA (134 mA de media). Por último, el *tag* se tomará un tiempo calculando su posición antes de volver al estado de *Deep Sleep*.

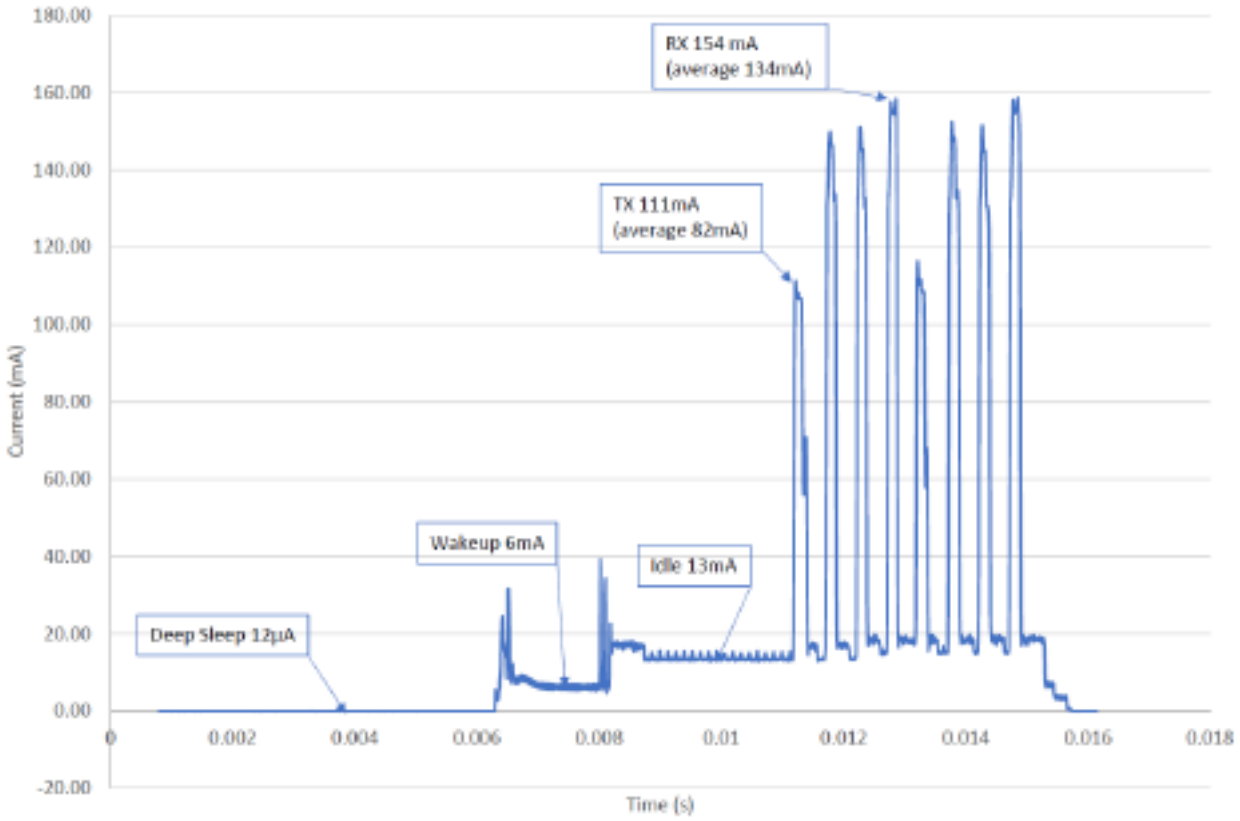


Figura 9. Consumo de energía del módulo DWM1001 [6]

5.3.8. Funcionamiento de la antena

En la Figura 10 se observan los patrones de radiación de la antena del módulo DWM1001. Esta antena se encuentra polarizada verticalmente y por tanto es recomendable situar las tarjetas de desarrollo en posición vertical para un mejor funcionamiento del sistema de posicionamiento. Por tanto, si se posicionan las balizas en una posición vertical se obtendrá una radiación, para el eje vertical Phi, omnidireccional de la ganancia (dBi) en función del ángulo (º), en el plano XZ observado en la Figura 11. En cambio, para el patrón polarizado horizontalmente como Theta se encuentra con valores nulos en ciertos ángulos pudiendo limitar el alcance y fallando en la exactitud de la posición.

Para los planos XY e YZ se ve en la Figura 12 como los patrones Phi y Theta no son omnidireccionales y por tanto en algún ángulo las medidas no pueden ser exactas y tendría un mayor error en el cálculo de la ubicación del *tag*.

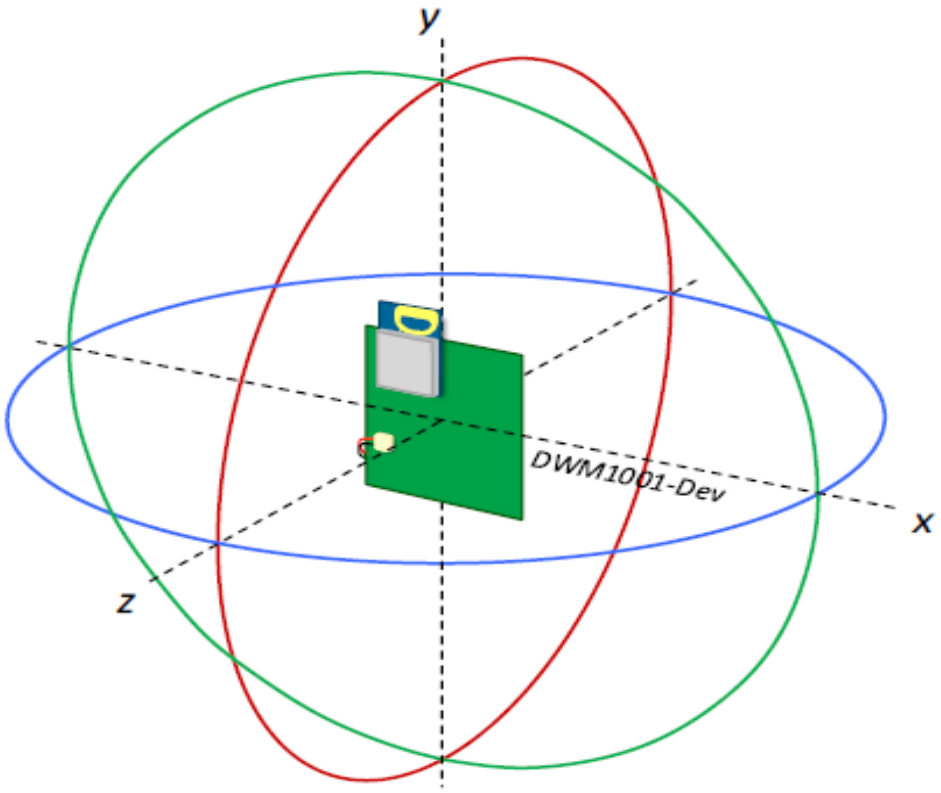


Figura 10. Patrones de radiación [6]

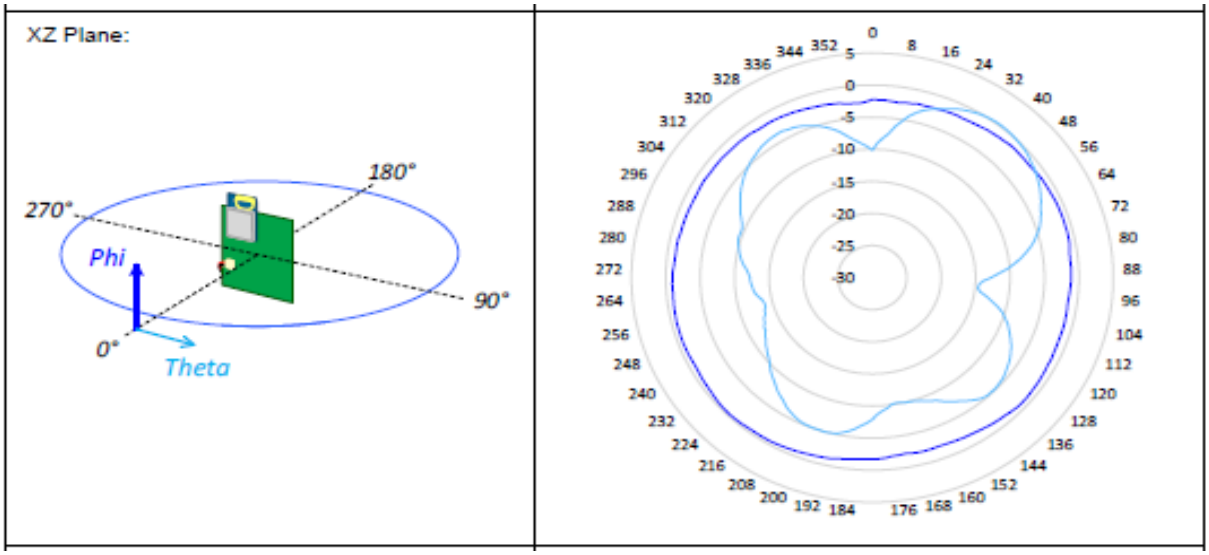


Figura 11. Radiación plano XZ [6]

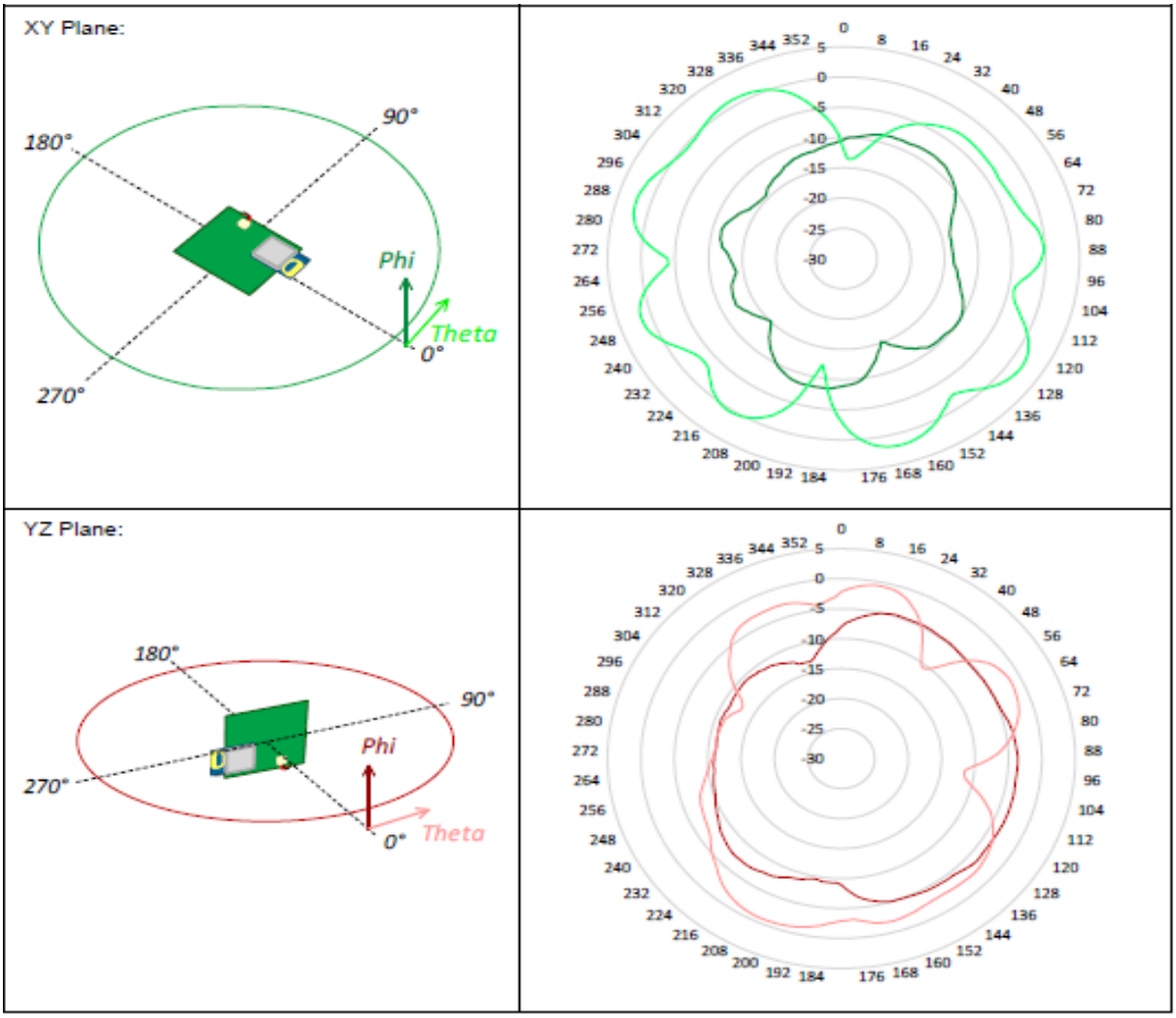


Figura 12. Radiación planos YZ y XY [6]

5.4. DWM1001 Real Time Location System (DRTLS)

El sistema de localización en tiempo real de *Decawave* consta de una tarjeta de desarrollo DWM1001-Dev que se pueden configurar como *anchor*s, *tag*s o *bridge nodes*. Para localizar un *tag* se utiliza un mínimo de tres *anchor*s y con el método de trilateración esférica es capaz de posicionar el *tag* en tres dimensiones a partir de la medida de rangos o distancias.

Los *anchors* que verá un *tag* para calcular su posición son como máximo cuatro con el *firmware* instalado de fábrica. Uno de los *anchors* se debe configurar como iniciador el cual iniciará y controlará la red y permitirá que otros *anchors* se unan a la red. Si se configura una puerta de enlace con un módulo DWM1001 y una *Raspberry Pi* se puede conectar el sistema a la red externa (LAN/WAN).

En Figura 13 se muestra un ejemplo de una red típica DRTLS con dos *tag*s y dos puertas de enlace que conectan el sistema a la red LAN/WAN.

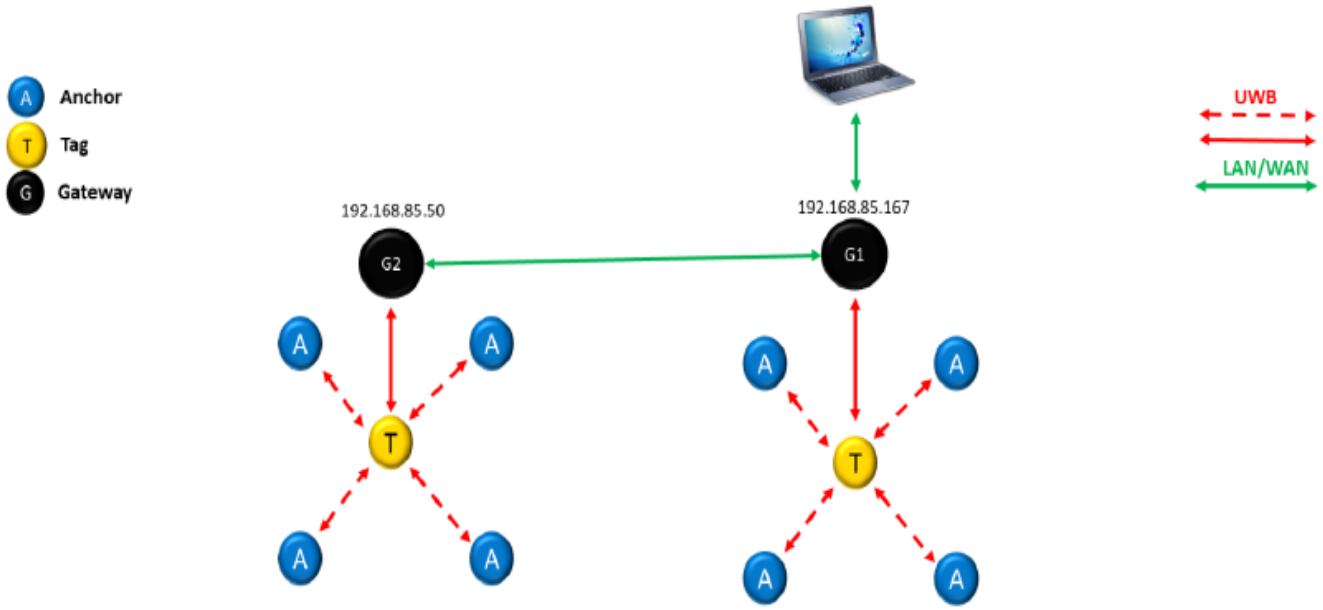


Figura 13. Ejemplo de red DRTLS [14]

5.4.1. Funcionamiento DRTLS

El DRTLS utiliza el acceso múltiple por división de tiempo, TDMA para acceder al canal. Este es un método de acceso al medio para una red compartida que permite a varios dispositivos compartir el mismo canal de frecuencia dividiendo la señal en varios intervalos de tiempo.

Los nodos utilizan una estructura de "supertrama" repetitiva de 100 ms de duración. El *anchor* configurado como iniciador es el coordinador de la red y controla la sincronización. La supertrama comienza con 30 ranuras o *slots* en las cuales los *anchors* envían el mensaje *Beacon* (BCN) o baliza, esto quiere decir que hasta 30 *anchors* pueden actuar en la misma área. Estos mensajes tienen el formato de trama de datos IEEE 802.15.4 y contienen información sobre el uso de la trama, como qué ranuras TWR están siendo utilizadas actualmente por los *tags* o los *anchors* o las coordenadas en la que está situada el *anchor*. Cada ranura BCN se corresponde con un número de identificación, el cual se le asignará a cada *anchor* para distinguirlos.

La supertrama continúa con dos ranuras de servicio (SVC) las cuales se utilizan para *Almanac*, es decir, se realiza la petición a red y se envía un mensaje por *anchor* en orden según el número de identificación que tengan. Este mensaje contiene la versión del hardware, la versión y el tamaño del *firmware*, las capacidades del nodo, lista de *tags* asignadas por cada *anchor* o solicitudes de unión a la red. Además, se realizan los intercambios de datos uplink/dowlink por IoT en los *anchors*.

La trama continúa mandando 15 ranuras bidireccionales, este método se le conoce como *Two-Way Ranging* (TWR) o Rango Bidireccional. Este método es utilizado para que el *tag* y el *anchor* hagan intercambios de información bidireccionales y calcular las distancias a partir de la medida

de los retardos de ida y vuelta. Además, en estas ranuras se envían los datos *uplink/dowlink* de IoT en el *tag*.

Más tarde, se envían 30 ranuras en las cuales los nodos que actúan como *bridge* mandan mensajes *beacon* los cuales dan información a los *tags* si existe algún dato *dowlink* para ellos. Por último, hay un tiempo de inactividad que completa los 100 ms de la supertrama. En la Figura 14 se pude ver un ejemplo de las supertramas.

El *tag* interactúa con hasta cuatro *anchors* y una vez sabe la distancia del *tag* a cada *anchor* calcula su ubicación. Los *tag*s escuchan inicialmente los BCN y SVC y así aprenderán sobre la tipología de la red y seleccionarán los cuatro *anchors* más cercanos con los que interactuar.

La posición del *tag* se envía a través de UWB, mediante la puerta de enlace a los clientes MQTT, por *Bluetooth* a la aplicación *Decawave DRTLS Manager* o a través de UART cuando un *tag* esté conectado a un PC.

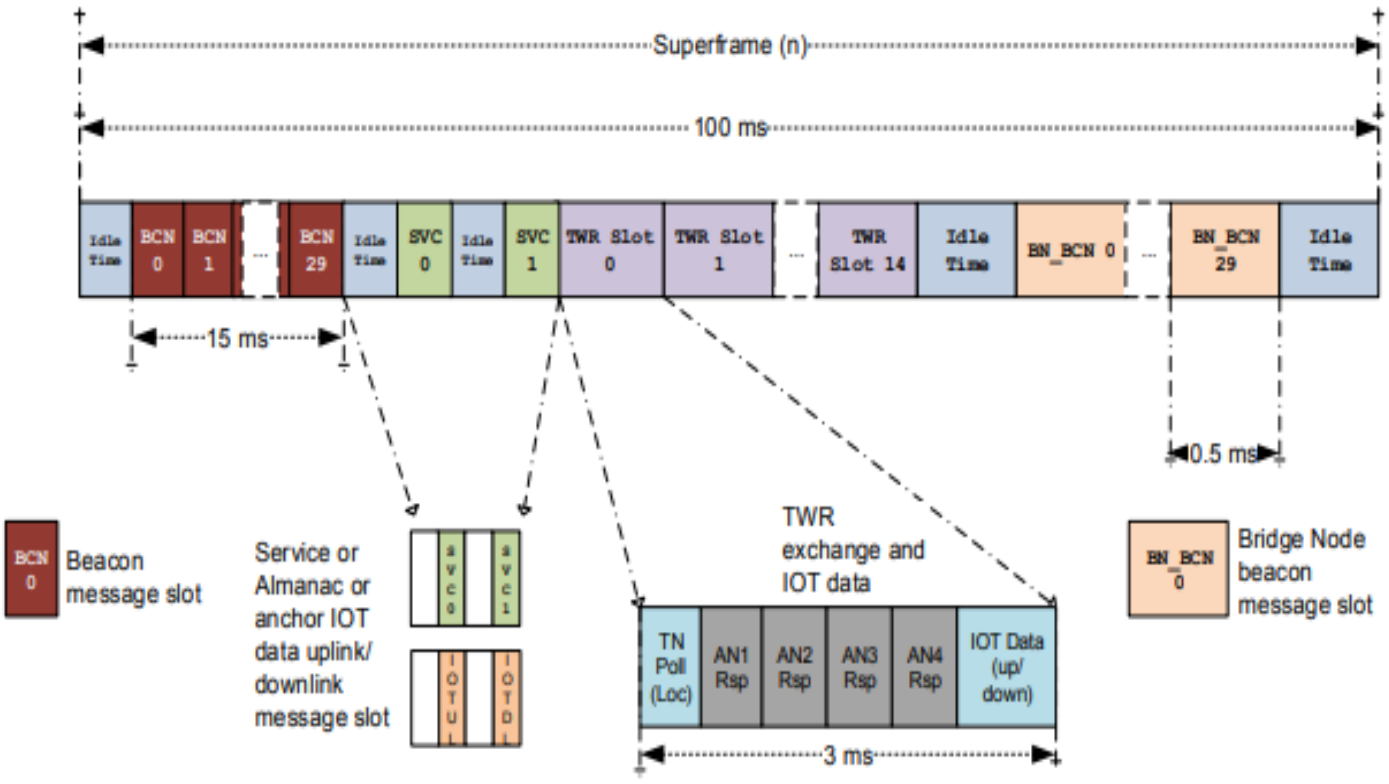


Figura 14. Estructura de la supertrama [7]

5.4.2. Registro y funcionamiento de un *anchor* en el sistema

Para que un *anchor* se pueda registrar en una red de localización, se deben cumplir las siguientes condiciones:

- El mensaje *beacon* debe indicar que hay algún número de identificación de *anchor* libre para ocupar, es decir el número de ranuras BCN ocupadas en la supertrama debe de ser menor de 30.
- El nivel de reloj de los *anchors* es de rango inferior a 127. Los *anchors* que pueden oír los mensajes *beacon* del iniciador están en el nivel 1, los siguientes en el nivel 2 y así sucesivamente.
- Todos los *anchors* en el rango han confirmado que el número de identificación solicitado por el *anchor* que desea unirse está libre y no se producen colisiones con otros dispositivos.
- El *firmware* debe de ser compatible, la versión debe coincidir con la versión del iniciador.

El proceso de registro de un *anchor* en la red es el siguiente:

Primero, el *anchor* que desea unirse escucha los mensajes BCN y SVC y comprueba que la versión de *hardware*, *firmware* y el tamaño son compatibles con los demás nodos. Si no es compatible, se inicia un proceso de actualización de *firmware*.

La actualización del *anchor iniciador* se puede hacer por depuración (SWD) o por *Bluetooth*. Una vez el iniciador está actualizado, los demás *anchors* ya pueden ser actualizados por el iniciador pudiéndolo realizar por aire (OTA) mediante UWB, para ello esta actualización debe de estar activada de forma predeterminada. Si no es así, se debería ir actualizando todos los *anchors* de uno en uno por los métodos nombrados anteriormente.

Una vez compatibles, el nuevo *anchor* continúa escuchando mensajes *beacon* y almacena todos los números de identificación de los *anchors* que puede oír. Después de oír cada uno de los *anchors* tres veces mínimo y comprueba que hay un número de identificación libre para poder unirse, comienza a buscar una ranura de servicio (SVC) libre para enviar el mensaje de petición de unión a la red con un número de identificación libre solicitado. Los *anchors* en el sistema envían un mensaje de confirmación de unión utilizando una parte extendida del mensaje *beacon*. Esto se repite durante tres supertramas desde la última recepción del mensaje de petición de unión.

Durante este proceso, todos los *anchors* involucrados están bloqueados y no se podrán unir nuevos *anchors*. Si otro *anchor* intenta unirse a la red su petición será ignorada, por tanto, solo un *anchor* puede unirse a la vez a la red. Una vez todos los *anchors* han confirmado el número de identificación del nuevo *anchor*, el registro se ha completado correctamente y el nuevo *anchor* comenzará a enviar los mensajes *beacon* y de servicio y podrá participar en el DRTLS.

Si un *anchor* detecta que dos *anchors* comparten el mismo número de identificación se le llamará colisión, si el contador de colisiones llega a un umbral determinado, es decir, están compartiendo

identificación durante un tiempo predeterminado, el *anchor* informará mediante una ranura de servicio (SVC) a uno de los *anchors* implicados, este *anchor* que recibe el informe de colisión abandonará la red y probará a unirse con un nuevo número de identificación libre.

5.4.3. Funcionamiento de un tag

Un *tag* tiene un funcionamiento parecido al del *anchor*. En un principio se encuentra "durmiendo" y se "despierta" periódicamente durante un tiempo de 500 ms, 5 supertramas, para escuchar los mensajes *beacon* (BCN) y de servicio (SVC) del *anchor*. Existen varias formas de "despertar" a un *tag* cuando está en modo de baja potencia:

- **RTC:** es utilizado para activar el UWB y la capa MAC.
- **Acelerómetro:** si se detecta movimiento el dispositivo saldrá del modo baja potencia y el *tag* pasará a utilizar la ubicación no estacionaria.
- **UART GPIO**: el usuario *host* puede "despertar" el dispositivo para que pueda comunicarse a través de las API de UART.
- **SPI:** al igual que con UART, el usuario *host* puede utilizar la señal SPI para "despertar" el dispositivo.
- **Botón "user":** el botón "user" está conectado al GPIO2 que es el utilizado para activar el dispositivo.

Si el *tag* recibe mensajes válidos, comprueba que tiene una versión *hardware*, *firmware* y un tamaño *firmware* compatible con los *anchors* de la red. Si estas no son compatibles se inicia un proceso de actualización de *firmware*, si son compatibles, el *tag* continuará con el proceso TWR.

Un *tag* tiene dos modos de funcionamiento:

- **Modo responsivo:** después del intercambio TWR se programará el siguiente periodo de escucha. El DW1000 no se pondrá en su estado de baja potencia, pero permanecerá en modo inactivo, con el reloj en funcionamiento. El nRF52832 estará en modo de suspensión si no hay otras tareas ejecutándose. Este modo básicamente se utiliza si se requiere *Bluetooth*.

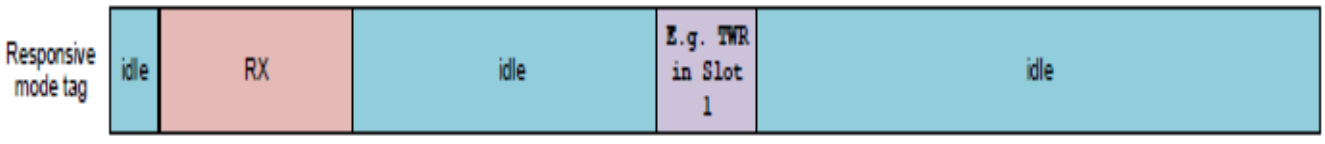


Figura 15. Modo de funcionamiento de tag modo responsivo [7]

- **Modo de baja potencia**: después del intercambio TWR, el DW1000 se pondrá en su estado de baja potencia y se activará antes del próximo intercambio TWR. Se pondrán en

modo de suspensión todos los componentes excepto el RTC y el acelerómetro. El módulo no escuchará las balizas a menos que se mueva fuera de la zona en la que se encuentran los *anchors* actualmente.

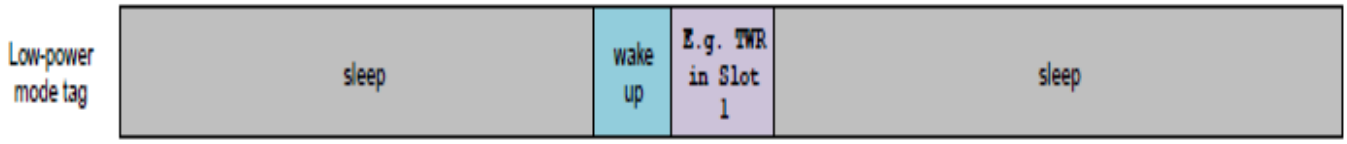


Figura 16. Modo de funcionamiento del tag en modo de baja potencia [7]

Un *tag* recibe información de los *anchors* y crea una lista con los *anchors* que ve y que ya recibió su posición. A continuación, calculará las distancias a cada una de los *anchors* de la lista a partir de su posición actual. Si hay más de cuatro *anchors*, el *tag* puede decidir cuáles de ellos elegir para calcular su ubicación. Los criterios que sigue para ello son:

- Primero el *tag* escogerá un *anchor* de cada cuadrante. Los cuadrantes se consiguen partiendo el *tag* por la mitad en vertical y en horizontal (ver ejemplo del *tag* T0 de la Figura 17).
- Si no es posible escoger un *anchor* de cada cuadrante, escogerán los *anchors* más cercanos al *tag*.

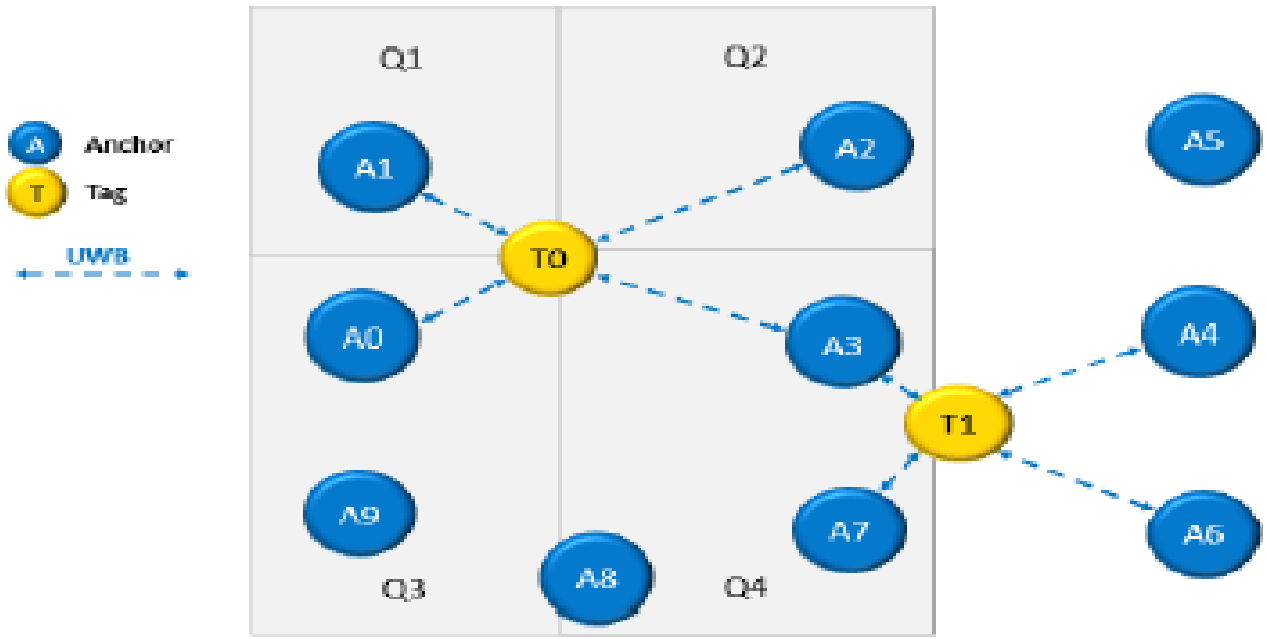


Figura 17. Cuadrantes para escoger anchors [7]

5.5. TWR (Two-Way Ranging)

El método *two-way ranging* o traducido del inglés rango bidireccional es el utilizado por *Decawave* para el módulo DWM1001. Este método se utiliza para calcular la distancia entre dos objetos determinando el tiempo de vuelo (TOF) de las señales que viajan entre ellos. Esta distancia se calcula haciendo uso de la fórmula 4.

$$Distancia = Velocidad\ de\ la\ luz\ \ x\ Tiempo\ de\ vuelo\ (TOF) \tag{4}$$

En este método dos unidades funcionan como par, una es el *tag* que inicia el intercambio y la otra es un *anchor* que escucha los mensajes del *tag* y realiza intercambios bidireccionales con él. Esto se muestra en la Figura 18.

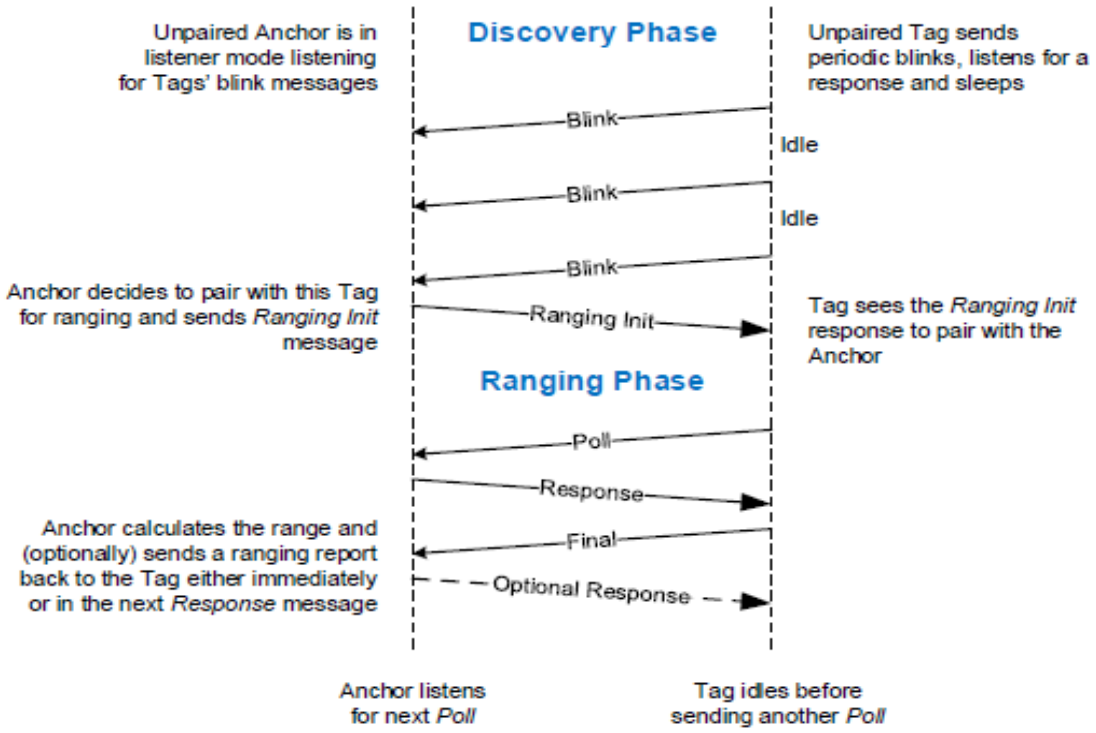


Figura 18. Método TWR [16]

Estos intercambios tienen dos fases:

- **Fase de descubrimiento**: el *tag* se encuentra en fase de detección y envía periódicamente un mensaje *Blink* que contiene su dirección para anunciarse a los *anchors* cercanos y escucha una respuesta del *anchor* llamada *Ranging Init*. Si el *tag* no recibe respuesta se duerme durante un tiempo predeterminado y vuelve a intentarlo. El *anchor*

escucha los mensajes *Blink* y cuando quiera emparejarse con un *tag* enviará automáticamente el mensaje *Ranging Init* al *tag* y se pasará a la fase de rango.

- **Fase de rango:** Esta fase se puede observar mejor en la Figura 19. Se realizan intercambios bidireccionales periódicos entre *tag* y *anchor* para medir la distancia entre ellos. Primero el *tag* envía un mensaje *Poll* o de sondeo a la dirección del *anchor* conocida, en el tiempo llamado TSP (*Time of Sending Poll*). El *anchor* almacena el tiempo de recepción del mensaje *Poll* (TRP) y responde con el tiempo TSR. El *tag* recibe la respuesta, almacena el tiempo TRR y compone el mensaje final que consta de su ID, TSP, TRR y TSF. Con estos tiempos y con el instante de tiempo de recepción a la que el *anchor* recibe el mensaje final, el *anchor* determina el tiempo de vuelo, TOF. Una vez calculado opcionalmente se puede comunicar al *tag* o a cualquier dispositivo UWB. Una distancia óptima entre el *anchor* y el *tag* para que el método TWR funcione correctamente sería entre 20 y 30 metros. La ecuación 5 es la utilizada para calcular el rango por el *tag* y la ecuación 6 es la utilizada por el *anchor* para calcular el tiempo de vuelo.

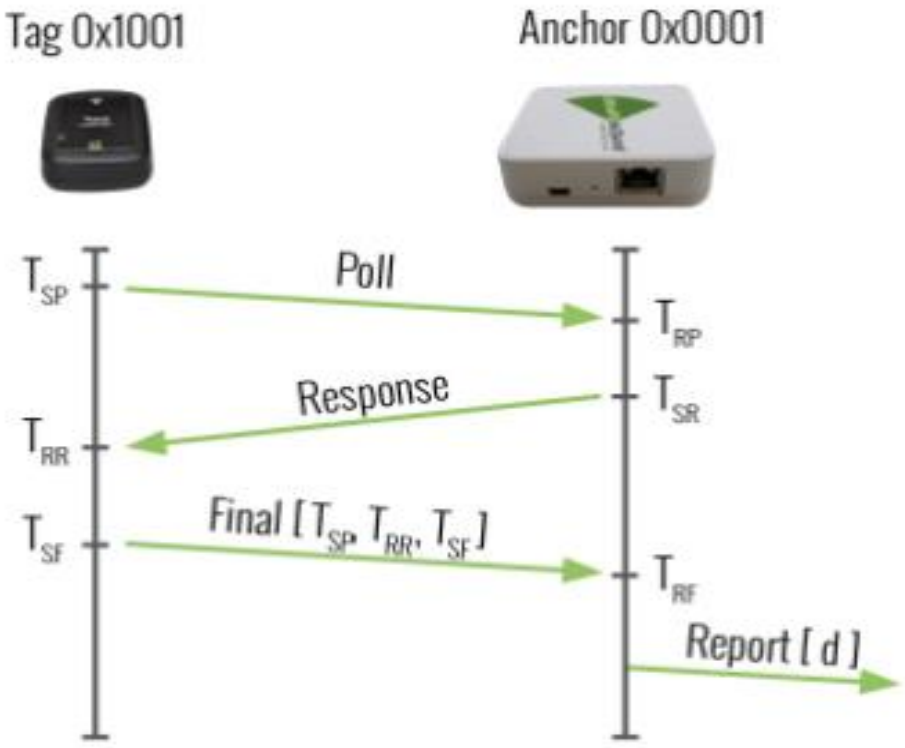


Figura 19. Fase de rango [3]

$$Rango = \frac{t_{receive\ response} - t_{sent\ initialization} - t_{tag\ delay}}{c} \quad (5)$$

$$TOF = \frac{[(TRR-TSP)-(TSR-TRP)+(TRF-TSR)-(TSF-TRR)]}{4} \quad (6)$$

Una vez calculadas las distancias del *tag* a cada *anchor* y sabiendo las coordenadas de cada *anchor* se pueden calcular las coordenadas en las que se encuentra el *tag* utilizando las ecuaciones 7.

$$\begin{cases} d_{10} = \sqrt{(X_1 - x)^2 + (Y_1 - y)^2} \\ d_{20} = \sqrt{(X_2 - x)^2 + (Y_2 - y)^2} \\ d_{30} = \sqrt{(X_3 - x)^2 + (Y_3 - y)^2} \end{cases} \quad (7)$$

Siendo d la distancia a cada uno de los *anchors*, X e Y las coordenadas de los *anchors* y por último x e y las coordenadas del *tag* que se desean calcular.

5.5.1. Protocolo TWR

Cada supertrama de 100 ms cuenta con 15 ranuras de rango TWR, es decir que admite 15 *tag*s como máximo en el área de la supertrama a 10 Hz. Si se requieren más *tag*s habría que disminuir la frecuencia o lo que es lo mismo aumentar el tiempo de la supertrama en múltiplos de 100 ms. Cada una de estas ranuras está diseñada para que al *tag* le dé el tiempo suficiente para realizar un rango bidireccional con cuatro *anchors*, dando una capacidad máxima de velocidad de ubicación de 150 Hz. Si las 15 ranuras de rango están completas, no se admitirán nuevos *tags* hasta que los existentes salgan del área o renuncien a sus ranuras.

El *tag* manda un mensaje de sondeo de grupo a los *anchors* dentro de su alcance, el cual contiene los siguientes datos: su rango, una lista de las cuatro direcciones de los *anchors* con los que desea intercambiar información y los números de identificación de ellos. Una vez recibido el mensaje, los *anchors* enumerados en el mensaje responden enviando otro mensaje de sondeo con el tiempo de transmisión determinado por el índice de su dirección, de cero a tres, en la lista. El *tag* manda otro mensaje de respuesta. A continuación, los *anchors* mandan un mensaje final.

Si existe algún mensaje IoT de *dowlink*, el *tag* lo podrá escuchar. Si no existe, podrá transmitir un mensaje IoT de *uplink* hacia la puerta de enlace.

Por último, con los rangos o medidas de distancia y las ubicaciones de los *anchors*, el *tag* utiliza su motor de localización interno y mediante trilateración esférica calcula su ubicación o posición (x,y,z). En la Figura 20 se observa lo descrito que es lo que sucede en una ranura TWR.

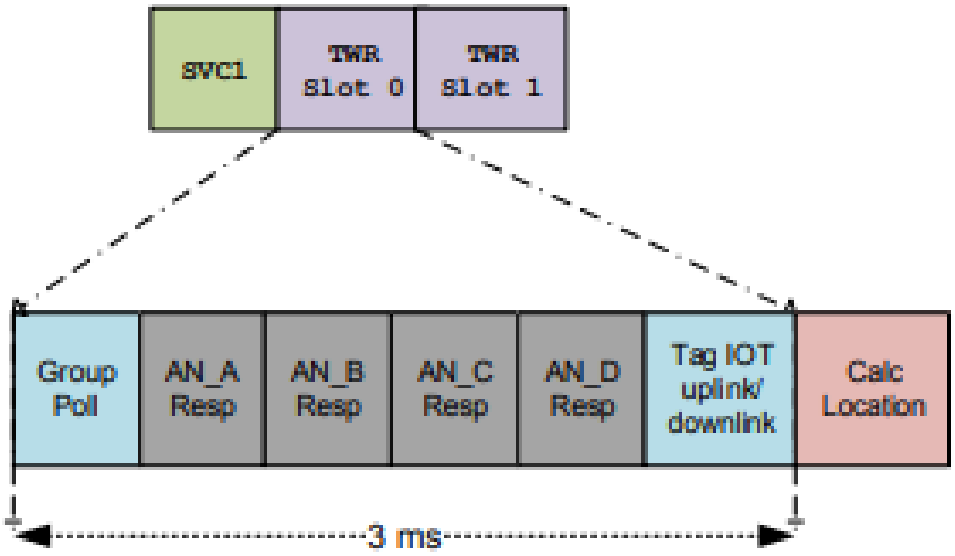


Figura 20. Tramas TWR de la supertrama [7]

Puede ocurrir que dos *tags* transmitan al mismo tiempo o que dos *anchors* compartan el mismo número de identificación en el mismo área, a esto se le llama colisión. Para ayudar a que esto no ocurra los dispositivos DWM1001 tienen un algoritmo de detección y resolución de colisiones. El *tag* señala una colisión si se recibe una trama con una dirección de destino diferente, si se recibe un mensaje no esperado o si no se recibe un mensaje de sondeo.

5.6. Motor de localización

Como se ha comentado anteriormente, los *tag*s utilizan su motor de localización interno para calcular su posición. La estrategia utilizada por *Decawave* en el *firmware* integrado es la intersección de esferas, más conocido como método de trilateración esférica. Para ello, utiliza los rangos bidireccionales enviados por los *anchors* y las posiciones de estos.

Cuando este método se utiliza de forma simple, para el cálculo de localización en dos dimensiones, es suficiente utilizar círculos intersecándose entre ellos para calcular su ubicación como se observa en la Figura 21.

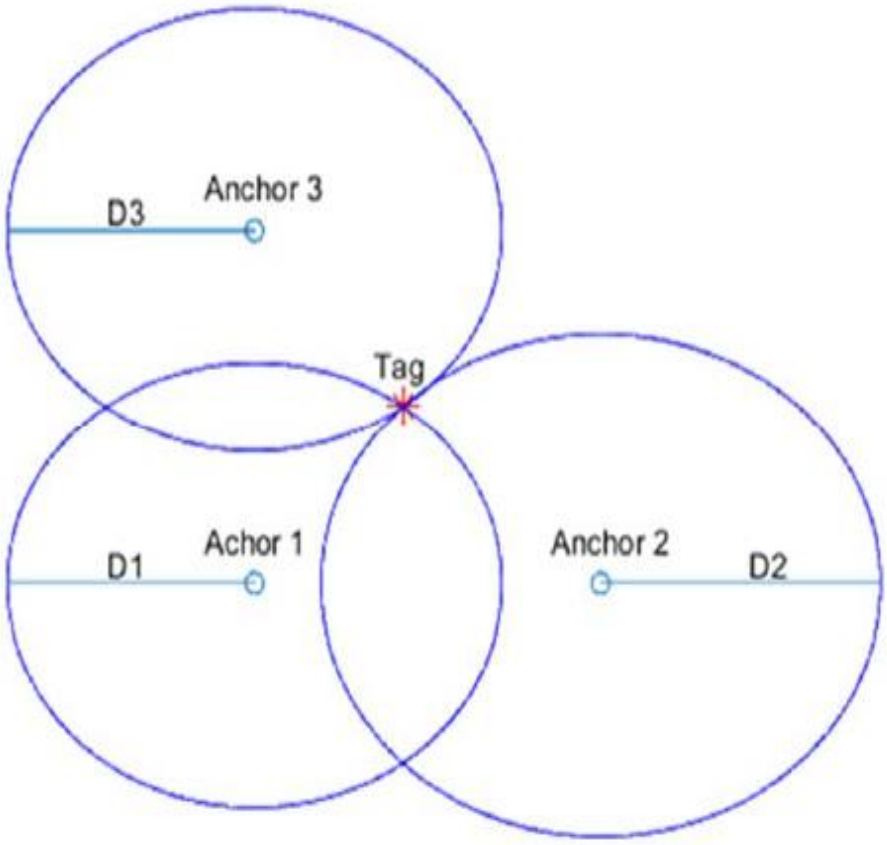


Figura 21. Trilateración para dos dimensiones [8]

Para calcular la ubicación del *tag* en tres dimensiones es necesario utilizar esferas. Para ello, se necesita tres o cuatro resultado de rangos, es decir, aunque un *anchor* de los cuatro no le envíe respuesta con el rango, puede calcular la estimación de la ubicación del *tag*. El código integrado calcula la posición de un punto utilizando la intersección de 3 esferas como mínimo y de 4 como máximo. Si la intersección de las tres esferas es de dos puntos, se calcula una línea entre ellos y la intersección con esa línea de una cuarta esfera será la localización buscada del *tag,* aunque normalmente es suficiente con tres esferas. Si con las tres primeras esferas no se encuentra una intersección, el *software* aumenta el radio de ellas para intentar buscar la intersección y la localización.

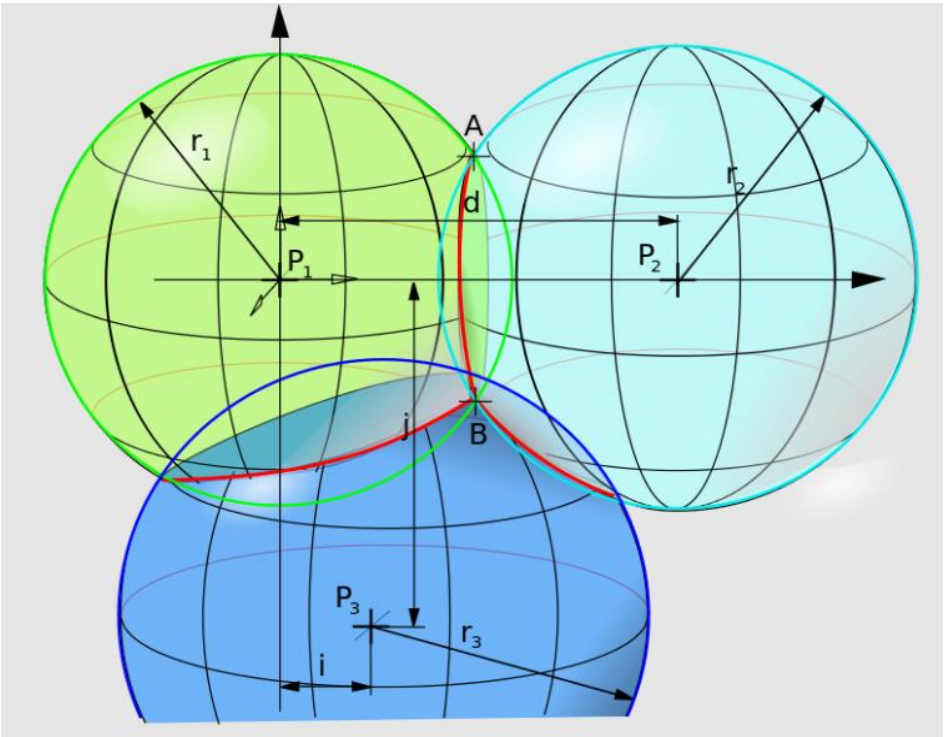


Figura 22. Trilateración de esferas para 3 dimensiones [9]

Cuando hay cuatro *anchors* y cuatro resultados de rango, el motor de localización crea varios conjuntos de rangos. Estos conjuntos se utilizan para calcular unas posibles posiciones del *tag*. Para reducir el tiempo del cálculo de las posiciones, el dispositivo dispone de una memoria caché donde almacena el conjunto de rangos. Si para hacer las mediciones se utilizan los mismos *anchors*, el motor de ubicación utiliza estos conjuntos almacenados y no es necesario que se vuelvan a calcular. A continuación, el motor de localización calcula la estimación de la posición combinando estos conjuntos y lo compara con distancias reales. Luego calcula los errores entre las distancias estimadas y las reales y elimina las posiciones con errores elevados. La posición final se calcula haciendo una media de las tres últimas posiciones estimadas. A este método se le conoce como estimación de máxima verosimilitud. Además, el motor de localización dará una idea de cómo de bueno es el cálculo de la estimación de la posición, notificando un factor de calidad entre cero y cien basado en los criterios y los errores calculados.

5.7. Expansión de la red

Si se requiere cubrir espacios más grandes, la red se puede ampliar con más *anchors*, más *tags* y más puertas de enlace como se observa en el ejemplo de la Figura 23. Para ello hay que cumplir una serie de normas y limitaciones:

- El rango de alcance entre dispositivos puede ser hasta de 60 metros con condiciones ideales de línea de visión y con la antena situada verticalmente, aunque dentro de una red

un *tag* se moverá por el sistema y no siempre tendrá la misma orientación y por tanto es complicado alcanzar el alcance máximo.

- Se recomienda espaciar los *anchors* entre ellos entre 20 y 25 metros en área cuadrada, dependiendo de la línea de visión que tengan, para asegurar un correcto funcionamiento.
- Para las puertas de enlace se recomienda considerar un rango máximo entre 30 y 35 metros dependiendo de las líneas de visión directa, si se desea cubrir un espacio con muchos obstáculos esta distancia se disminuiría.

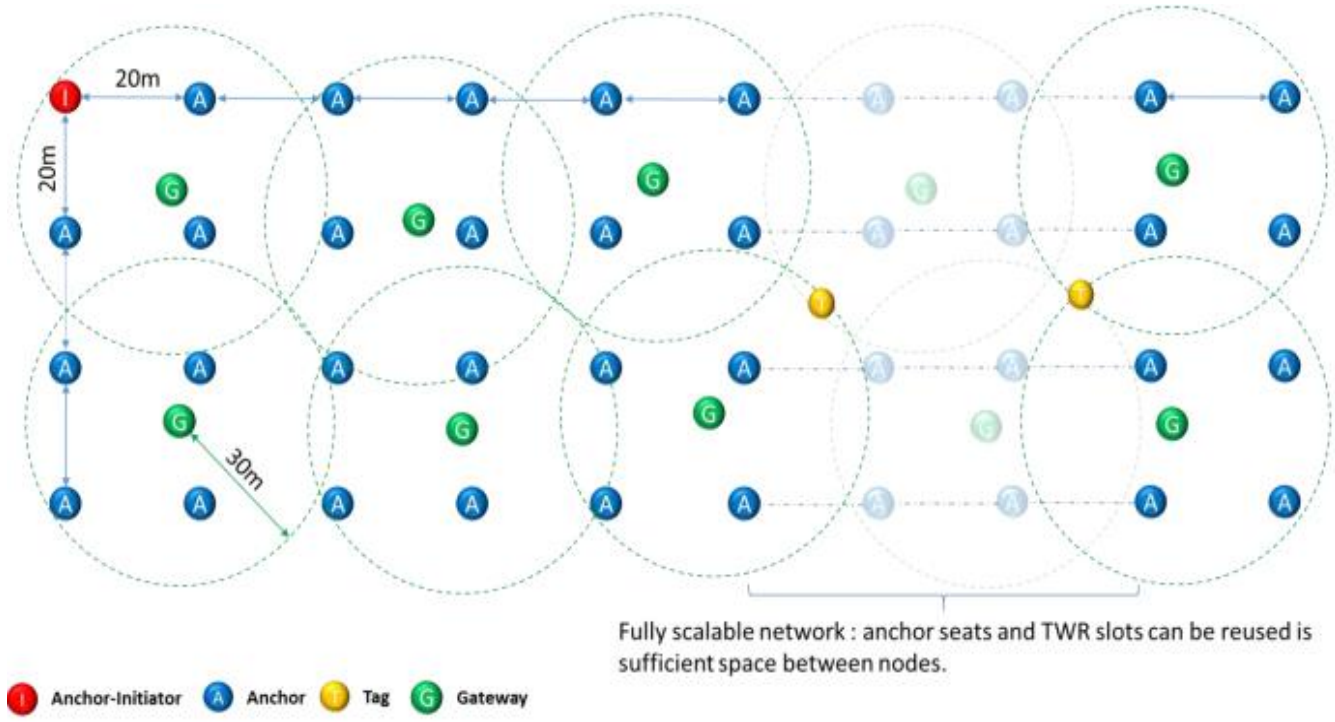


Figura 23. Ejemplo de expansión de la red [7]

5.7.1. Normas y limitaciones de los *anchors*

Para formar una red de localización existen unas reglas esenciales que los *anchors* siempre deben de cumplir:

- La red utiliza un esquema de acceso múltiple por división de tiempo y por tanto todos los *anchors* deben estar sincronizados con la supertrama del iniciador.
- Cada *anchor* debe de tener un número de identificación entre 0 y 29.
- Está prohibido que algún *anchor* de la red oiga dos *anchors* con el mismo número de identificación y si así sucede se activaría las colisiones y uno de ellos abandonaría la red.
- El número máximo de nivel de reloj debe de ser 127.

Cumpliendo siempre estas reglas se puede ampliar la red todo lo necesario. Como los números de identificación van del 0 al 29 solo hay 30 números de identificación, por tanto, si se requieren más de 30 *anchors* se debe repetir algún número de identificación. Para ello hay que espaciar los *anchors* lo suficientemente lejos para que ningún *anchor* oiga dos veces el mismo número de identificación. Una vez colocados los *anchors* suficientemente lejos, los demás *anchors* que se encuentran en el rango del *anchor* que se desea unir a la red, deben confirmar que existe un número de identificción libre que se le asignará al nuevo *anchor*.

Por ejemplo, la red de la Figura 24 se podría aumentar ya que al *anchor* número 31 se le puede asignar un número de identificación libre que no oigan los *anchors* en su rango, por ejemplo, el *anchor* número 28 o el número 21 se encuentran dentro del rango del 31, pero no oyen al *anchor* con número de identificación 0, por tanto al número 31 se le podría asignar el número de identificación 0 ya que así ningún *anchor* en la red oiría dos *anchors* con el mismo número de identificación.

Por todo ello, no se recomienda colocar demasiados *anchors* en un espacio reducido, si no colocar los *anchors* suficientemente distanciados para permitir reasignar los números de identificación.

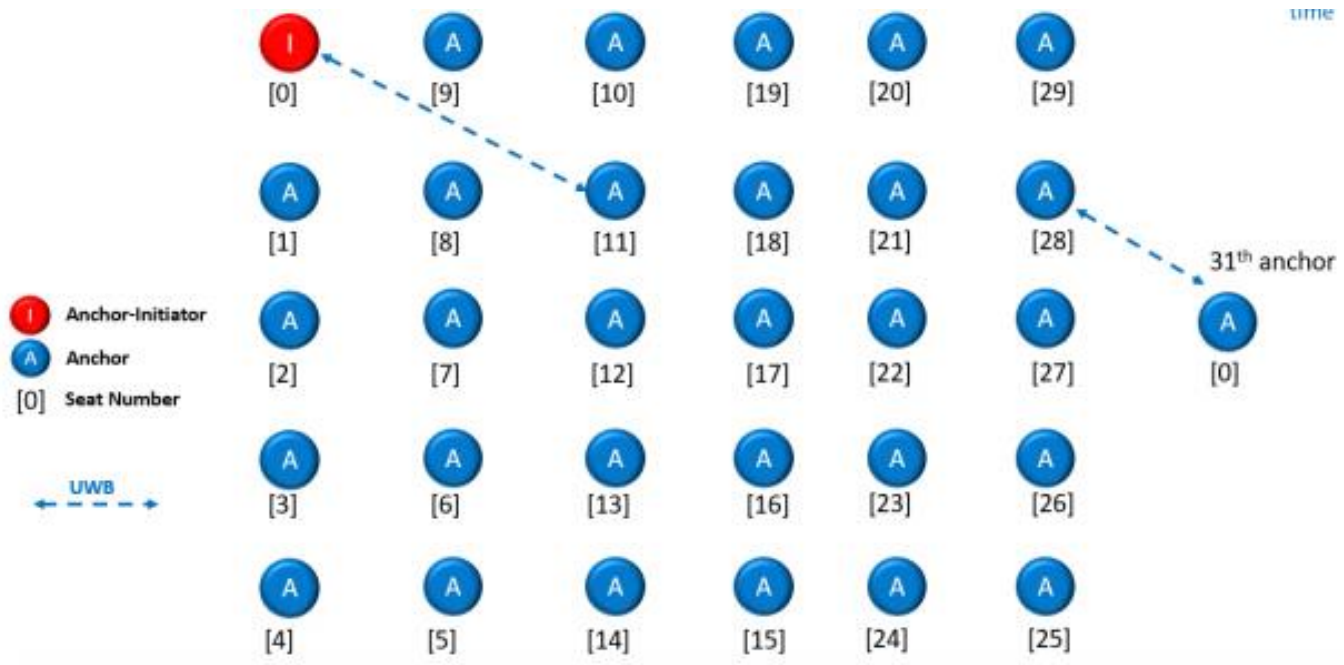


Figura 24. Ejemplo de expansión de red con números de identificación[7]

Como se ha comentado anteriormente, otra regla esencial es que el nivel de reloj máximo es de 127. Como se puede observar en la Figura 25, cada *anchor* tiene el nivel de reloj derivado del *anchor iniciador* o del *anchor* en el rango más cercano al iniciador. Esto quiere decir que los

anchors dentro del rango del iniciador tendrán su mismo nivel de reloj el cual es 1, a los *anchors* que se encuentran en el rango de los *anchors* de nivel 1 pero no en el rango del iniciador se les asignará el nivel de reloj 2 y sucesivamente hasta llegar al máximo nivel de reloj, 127. Por tanto, si un *anchor* nuevo que desea unirse a la red solo tiene en su rango a *anchors* de nivel 127 ya no podrá unirse a la red.

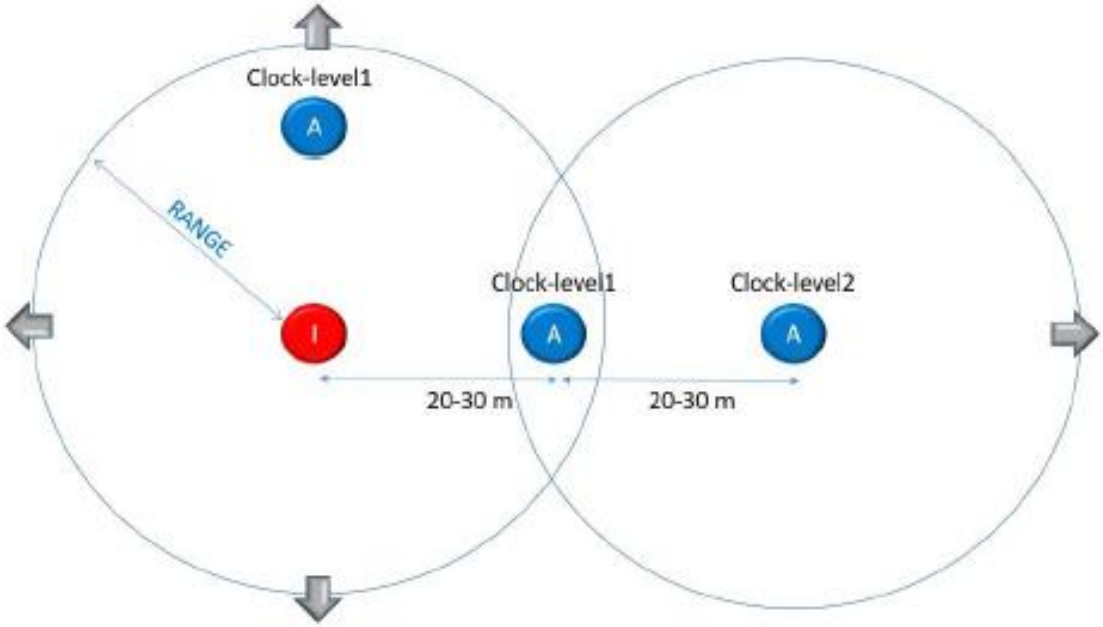


Figura 25. Expansión de la red dependiendo del nivel de reloj [7]

5.7.2. Normas y limitaciones de los *Tags*

Las principales reglas que se deben de seguir para aumentar los *tag*s es que el tiempo sea múltiplo de 100 ms y que la frecuencia total del sistema sea de 150 Hz. Como es evidente, si se requieren más *tag*s el tiempo en calcular la ubicación será mayor. En los siguientes ejemplos se comprenderá de mejor manera:

1. La frecuencia máxima es de 10 Hz, para esta frecuencia el número máximo de *tag*s sería de 15. Por tanto 15 *tag*s por 10 Hz es igual a 150 Hz. Además, T = 1/f = 1/10 Hz = 100 ms el cual será el tiempo mínimo y, por tanto, se tendría una velocidad de ubicación máxima.
2. Si por ejemplo se quieren aumentar los *tags* a 60, para que la frecuencia total de 150 Hz se cumpla, habría que utilizar una frecuencia de 2.5 Hz, es decir, 60 *tags* por 2.5 Hz es igual a 150 Hz. Además, el tiempo seguiría siendo múltiplo de 100 ms, T = 1/2.5 Hz = 400 ms. Siguiendo este criterio se tendrían los siguientes ejemplos.
3. 150 *tags* por 1 Hz. El tiempo de ubicación sería de 1 segundo.

4. 300 *tags* por 0.5 Hz. El tiempo de ubicación sería de 2 segundos.
5. De este modo se llega al máximo número de *tags* y por tanto a la mínima frecuencia. 9000 *tag*s por 0.016666 Hz. El tiempo sería de 1 minuto la cual sería una velocidad de ubicación mínima.

La última regla que hay que tener en cuenta, es que solo puede haber 15 *tags* a ser identificados y localizados cada 4 *anchors* ya que en una supertrama solo hay 15 ranuras. Si se requieren más *tag*s a localizar en una misma zona, habría que colocar más *anchors*.

5.8. Firmware

La tarjeta de desarrollo DWM1001 viene precargada con el *firmware* DRTLS que permite el desarrollo de un sistema de localización en tiempo real (RTLS) sin necesidad de programar un nuevo *firmware* para conseguirlo. Este *firmware* se divide básicamente en dos partes: Interfaz de Programación de Aplicaciones (API) de PANS para un más alto nivel y con la cual, se puede configurar y controlar el módulo y con la que el usuario podrá personalizar parte de su funcionamiento y agregar sus propias funciones haciendo uso de la cadena de herramientas y la biblioteca PANS para proporcionar funciones de más bajo nivel.

Para comenzar a utilizar el módulo lo primero que hay que hacer es actualizar el *firmware* con la última versión disponible. Una vez actualizado el *firmware*, la memoria flash queda estructurada como se observa en la Figura 26.

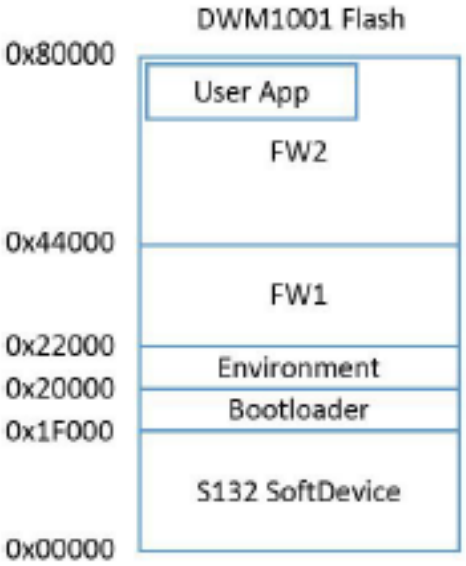


Figura 26. Memoria flash del DWM1001 [10]

5.8.1. Descripción del Firmware

Empezando con la dirección más baja, 0x00000, hasta la más alta, 0x80000, la memoria tiene un tamaño total de 512 KB y se divide en las siguientes partes:

- ***S132_SoftDevice*:** va de la dirección 0 a la dirección 0x1F000, por tanto, tiene un tamaño de 124 KB. Este es el encargado de proporcionar el Bluetooth de baja energía y una solución de la pila de protocolos.
- ***Bootloader*:** va de la dirección 0x1F000 hasta la 0x20000 ocupando por tanto 4 KB de tamaño. Su función es controlar la elección del FW1 o del FW2 durante el arranque del dispositivo.
- ***Environment*:** comienza en la dirección 0x20000 y finaliza en la dirección 0x22000, teniendo un tamaño de 8 KB. Esta es una sección reservada donde se almacena la información de configuración del usuario. Al apagar el dispositivo o si se vuelve a cargar un nuevo *firmware*, esta parte no se modificará ni se eliminará. Para eliminar y reestablecer esta parte es necesario realizar un borrado completo del *firmware*.
- **FW1:** tiene un tamaño de 136 KB y empieza en la dirección 0x22000 y finaliza en la dirección 0x44000. Es el encargado de la actualización del *firmware* FW2 mediante el aire, por UWB.
- **FW2:** es la última parte y comienza en la dirección 0x44000 y termina en la dirección 0x80000 teniendo por tanto un tamaño de 240 KB. Esta parte contiene la aplicación de usuario de código C y el bloque de funcionamiento principal. El bloque principal incluye la biblioteca PANS completa. La aplicación de usuario es un subproceso del FW2 y puede ocupar como máximo 3 KB de memoria RAM y 60 KB de memoria Flash.

5.8.2. Métodos de actualización

Para que una red de localización funcione todos los *anchors* de la red deben tener la misma versión de *firmware* que el iniciador. Si no tienen la misma versión es necesario actualizar el *firmware* y para conseguirlo existen distintos métodos.

5.8.2.1. Actualización de firmware manualmente

Una forma de actualización es manualmente a través de la depuración por el puerto serie (SWD). Con este método hay que actualizar los dispositivos uno a uno y es necesario por ejemplo, si se necesita actualizar el dispositivo de la versión PANS R1 a la versión PANS R2, ya que de una versión a otra cambian algunas capas y no es posible actualizarlo a través de otros métodos. Otra función para el que es útil este método es cuando se requiere hacer un borrado completo

del *firmware* y así poder actualizar la parte de la memoria llamada *environment*. Se recomienda actualizar todos los dispositivos mediante este método antes de formar la red.

5.8.2.1.1. Actualización de firmware por Bluetooth

Una manera muy cómoda de actualizar la red una vez se encuentra operativa es mediante este método. Solamente es necesario actualizar el *anchor iniciador* por *Bluetooth*, una vez el iniciador este actualizado, propagará el *firmware* a los demás dispositivos automáticamente a través de UWB.

5.8.2.1.2. Actualización de firmware por UWB

Para utilizar este método se debe de tener activada la actualización automática de *firmware* en los dispositivos. Si un dispositivo que desea unirse a la red no tiene el mismo firmware, elegirá el *anchor* más cercano y le solicitará una actualización de *firmware*. Este *anchor* le envía los datos de actualización mediante UWB. Cualquier *anchor* cercano que desee actualizar el *firmware* podrá recibir y procesar estos datos de actualización.

Una vez recibidos los datos de actualización se almacenan en un *buffer* intermedio, cuando estos datos alcanzan el tamaño del flash interno, se escribe la nueva imagen de actualización. Durante la actualización de *firmware* el *Bluetooth* se encuentra deshabilitado ya que escribir a través de *SoftDevice* sería muy lento.

Si ocurre algún error en la actualización, el *anchor* volvería a solicitar la actualización a partir del error anterior. Una vez recibidos todos los datos de actualización se ejecutará la suma de comprobación de todo el *firmware* para verificar que los datos son correctos. Si son correctos se guardará la nueva memoria, si no lo son comenzaría el proceso de nuevo. El orden que sigue para actualizar el *firmware* sería ejecutar primeramente el FW2 y actualizar el FW1 y luego actualizará el FW2.

5.8.3. Arquitectura del firmware

En la Figura 27 se observa la arquitectura de alto nivel y los componentes del *firmware*. En ella se pueden diferenciar las distintas partes del *firmware* como los módulos externos para interactuar con el DWM1001, la biblioteca PANS o la API de PANS que incluye una API genérica para comunicar las API de usuario como son C, UART, SPI y BLE con la biblioteca PANS.

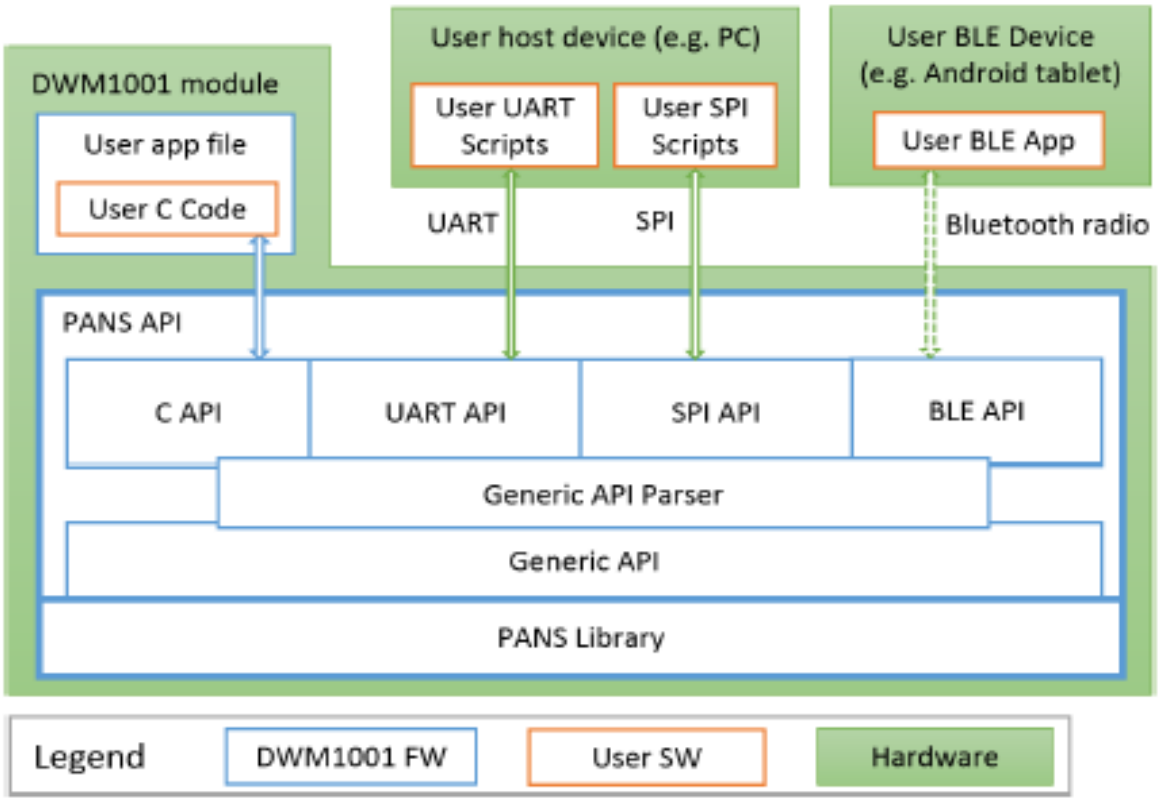


Figura 27. Arquitectura del firmware [10]

5.8.3.1. BLE API

Se puede utilizar con cualquier dispositivo que disponga de una conexión *Bluetooth* de baja energía y que cuente con la aplicación proporcionada por *Decawave* llamada *DRTLS Manager* y que puede ser utilizada por varios dispositivos como pueden ser un teléfono móvil o una Tablet. Con una conexión *bluetooth* a través de la aplicación es posible controlar y configurar los módulos para formar una red de localización. El módulo *Bluetooth* se activa en el encendido del módulo DWM1001 y cuando el usuario presiona el botón “user”. Si un dispositivo con la aplicación se conecta al módulo *tag*, la conexión permanece constante y por tanto el *tag* no podrá pasar al modo suspensión. Tan pronto como el dispositivo se desconecte, el módulo enviará datos *bluetooth* durante 20 segundos y después se desactivará liberando los recursos *bluetooth* y permitiendo que el *tag* pueda entrar en suspensión para ahorrar energía.

La forma de actuar de la API BLE es la mostrada en la Figura 28. El controlador de eventos del DWM1001 convierte las operaciones del Atributo Genérico (GATT) el cual, controla el flujo de datos, en comandos API genéricos. Si existe algún evento relacionado con el *bluetooth*, el controlador de eventos le envía la información correspondiente al dispositivo externo.

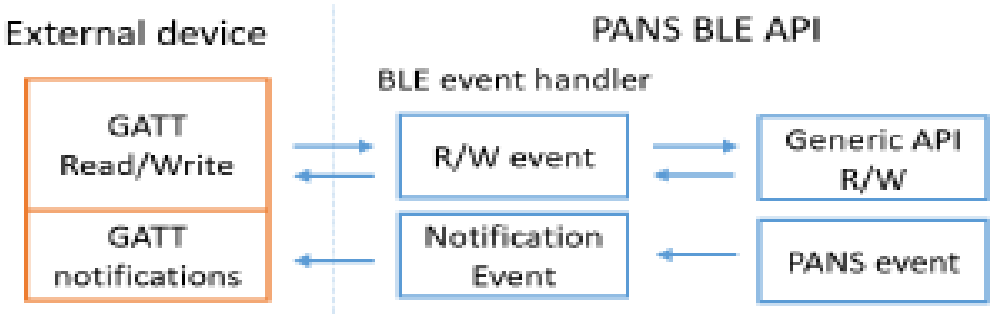


Figura 28. API BLE [11]

5.8.3.2. SPI API

Se utiliza en dispositivos que ofrecen varios servicios como pueden ser transferencia de datos o conexión remota como por ejemplo un PC. Este dispositivo en el modo maestro se comunica con el módulo DWM1001 que funciona como esclavo a través de TLV (Tipo-Longitud-Valor), es decir, envía datos comenzando con un byte de tipo, otro byte indicando la longitud y por último un número de bytes de valor indicados por la longitud entre 0 y 253. Un flujo completo de intercambio de datos a través de TLV incluye cuatro pasos: primero el *host* envía la solicitud TLV, luego el DWM1001 prepara la respuesta, más tarde el *host* lee el número de transferencias y el tamaño de cada respuesta y por último el *host* lee los datos de las respuestas. Una vez entra en modo SPI, el microcontrolador del *tag* no entra en modo suspensión y el *host* puede controlarlo a través de una llamada API.

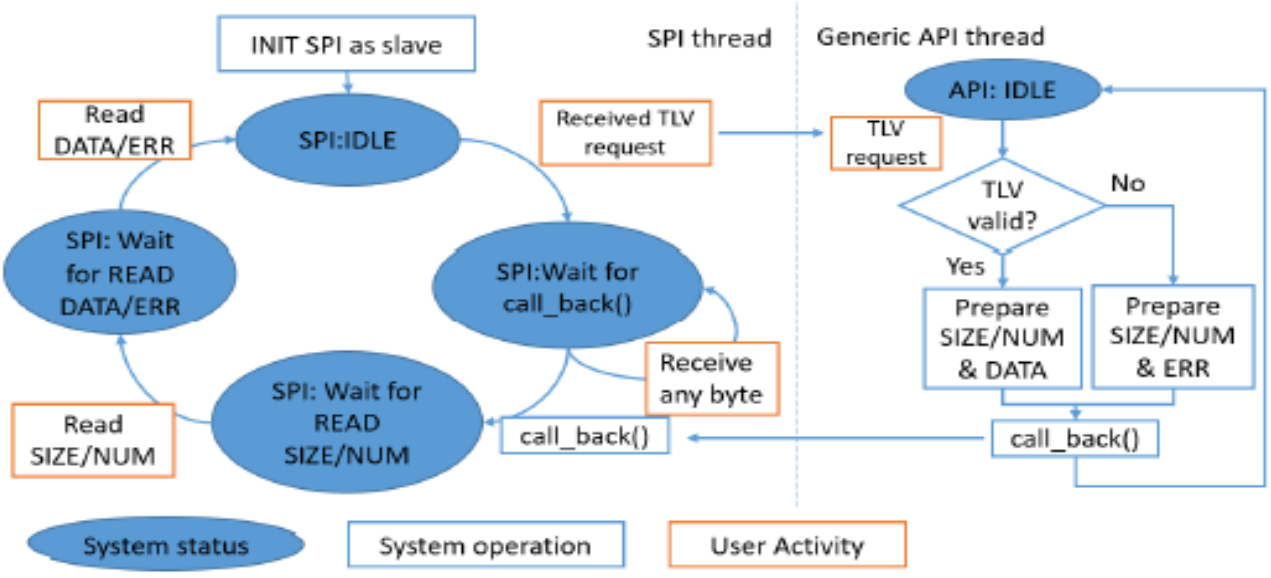


Figura 29. Funcionamiento SPI [11]

En la Figura 29 se muestra el diagrama de flujo correspondiente al funcionamiento de SPI en el DWM1001. Este diagrama tiene cuatro estados y su funcionamiento se explica a continuación:

- **SPI: IDLE**: es el estado en la inicialización del dispositivo y después de cada transmisión de datos exitosa. Aquí se reciben las solicitudes TLV. Si se recibe una solicitud, esta se pasa a la API genérica la cual analiza la solicitud y ve si es correcta o no y prepara los datos devueltos. Si es correcto, enviará el dato y si no, enviará un error a la función *call_back().* Una vez se recibe una solicitud TLV se pasa al siguiente estado.
- **SPI**: **Wait for call_back():** el SPI espera a que la API genérica le envíe la respuesta. Si se recibe algún byte del *host,* se ignora y sigue en el mismo estado. Una vez la API genérica analiza la solicitud, llama a la función *call_back()* y se pasará al siguiente estado.
- **SPI**: **Wait for READ SIZE/NUM:** la API genérica ha preparado la respuesta y envía el byte SIZE con un total de NUM transferencias. Estos son datos de dos bytes distintos de cero y puede contener el dato a leer o un mensaje de error. Una vez el dispositivo *host* lee los dos bytes, se pasa al siguiente estado.
- **SPI: Wait for READ DATA/ERR:** SPI DWM1001 envía los bytes SIZE de datos si el mensaje es correcto o un mensaje de error si la solicitud TLV era incorrecta. En este estado se espera a que el *host* lea o los datos de respuesta o el mensaje de error. Una vez el *host* lee la respuesta, se vuelve al estado inicial y vuelve a comenzar con el proceso.

5.8.3.3. UART API

Al igual que con SPI se necesita un dispositivo como *host* como puede ser un PC para comunicarse a través del bus UART a una velocidad de 115200 baudios. En este caso existen dos modos de comunicación con el DWM1001: uno es el modo genérico utilizando solicitudes y respuestas con TLV y la otra usando el modo UART Shell mediante comandos de solicitud de terminal. El modo predeterminado es el modo genérico, si se requiere pasar al modo *Shell* es necesario pulsar la tecla “Enter” dos veces en un segundo o introducir los bytes 0x0D, 0x0D. Si por el contrario, el usuario se encuentra en el modo *Shell* y desea pasar al modo genérico se debe introducir el comando “quit”. En la Figura 30 se encuentra el diagrama de flujo del funcionamiento de la interfaz UART en el módulo DWM1001, en los siguientes apartados se irán explicando los distintos estados y acciones.

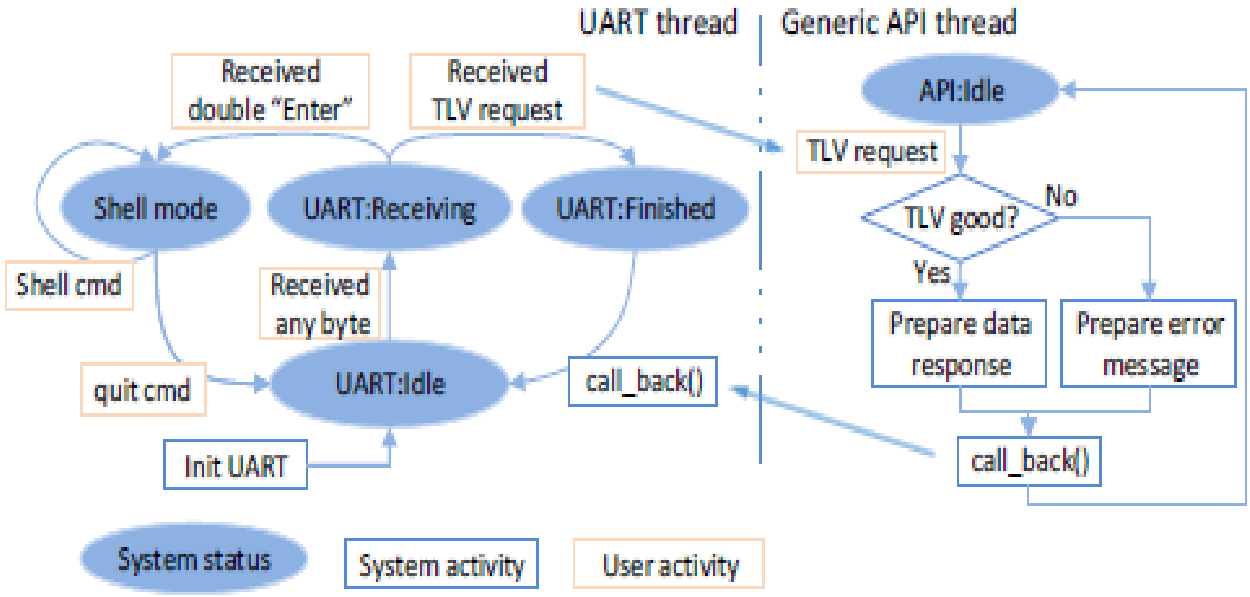


Figura 30. Funcionamiento UART [11]

5.8.3.3.1. UART modo genérico TLV

En este modo, el *host* y el dispositivo DWM1001 se comunican a través de datos TLV. El proceso que sigue la API genérica para este tipo de interfaz es el mismo que el explicado anteriormente para la interfaz de SPI. Los estados de la Figura 30 que actúan en este modo son tres:

- **UART: Idle:** es el estado inicial donde empieza al iniciar la conexión o después de cada respuesta TLV. En este estado se espera a recibir un byte para inicializar la solicitud TLV e inicia el temporizador de retardo para los siguientes datos. Una vez recibe un byte se pasa al siguiente estado.
- **UART: Receiving:** en él se espera a terminar de recibir la solicitud TLV. Una vez reciba cualquier dato en este estado se actualiza el temporizador de retardo. Si el *host* termina de mandar datos, el temporizador expirará. Al terminar de recibir la solicitud hay dos opciones: si la solicitud son dos bytes o doble "Enter" se pasará al estado *Shell mode*, si no es así, se envía la solicitud TLV a la API genérica para que estudie la solicitud y se pasa al estado final.
- **UART: Finished:** es el último estado del proceso donde espera a que el API genérico estudie la solicitud y vea si es correcta o no. Una vez analizada se enviará el mensaje de error o los datos de la respuesta llamando a la función *call_back()* y se pasará de nuevo al estado inicial.

5.8.3.3.2. UART modo Shell

En este modo se proporciona una terminal y se utilizan comandos *Shell* para la comunicación los cuales son cadenas de caracteres seguidos de la tecla “Enter”. Una vez el *tag* entra en modo Shell el microcontrolador integrado no podrá entrar en modo suspensión y ahorrar energía, hasta que el usuario salga de este modo introduciendo el comando “quit”. Los estados en la Figura 30 que influyen en el funcionamiento son tres, dos de ellos coincidiendo con el modo genérico TLV:

- **UART: Idle:** es el estado donde inicia el proceso después de iniciar la comunicación UART o después de recibir el comando “quit”. Una vez se envía un byte de solicitud se pasa al siguiente estado.
- **UART: Receiving:** es el mismo estado que para el modo genérico. Si se reciben dos bytes [0x0D, 0x0D] o si se pulsa dos veces la tecla “Enter” se pasa al estado de *Shell mode*.
- **Shell mode:** el estado donde se mandan los comandos para intercambiar información. Una vez entra en este estado, el DWM1001 permanece aquí permanentemente hasta que el usuario introduzca el comando “quit” y se pase al estado inicial.

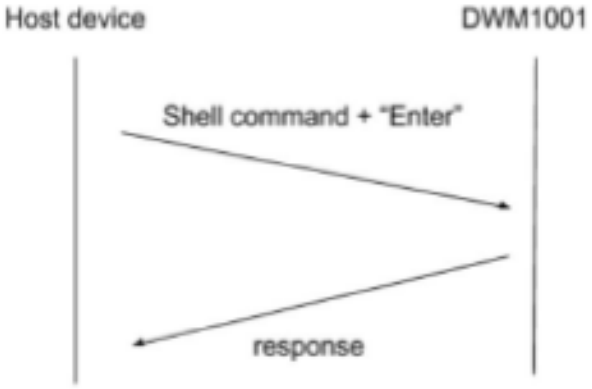


Figura 31. Intercambio en modo Shell [11]

En la Figura 31 se muestra un intercambio de información completo en este estado. El usuario a través del *host* envía un comando *Shell* más la tecla “Enter”, luego el DWM1001 responde al *host* con el mensaje correspondiente. Si no hubiera nada que responder el DWM1001 no enviaría nada.

5.8.3.4. C API

Esta interfaz será donde el usuario puede añadir un código al archivo de aplicación de usuario, utilizando la cadena de herramientas proporcionada por *Decawave*. Para añadir nuevas funcionalidades al *firmware* el usuario debe tener en cuenta varias cuestiones:

- Se debe de basar en las bibliotecas eCos RTOS y DWM.

- Los archivos utilizados para el desarrollo de aplicación de usuario son:
    a) dwm.h: es el archivo de encabezado utilizado para todas las funciones que se requieran crear.
    b) libdwm.a: es una biblioteca estática la cual no es de código abierto.
    c) extras.o, vectors.o y libtarget.a: es la biblioteca estática de eCos.
    d) target_s132_fw2.ld: para enlazar con el *firmware* 2.
- API proporciona unas funciones que define la aplicación de usuario como son: funciones comunes como la asignación de memoria o el acceso a las interfaces, funciones de inicialización, configuración y mantenimiento de la pila y funciones de devoluciones de llamadas para recibir datos o mediciones.

5.9. Biblioteca PANS

Una vez vista la arquitectura de más alto nivel, se va a estudiar más en detalle la de más bajo nivel como es la biblioteca PANS. La arquitectura de esta biblioteca es la mostrada en la Figura 32 donde se diferencian varias partes para tener un mejor control del módulo DWM1001.

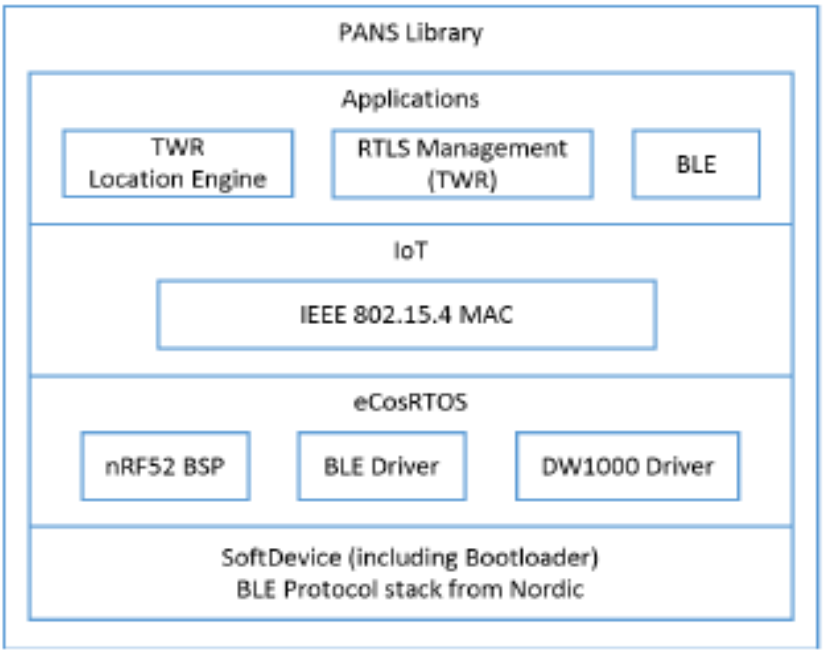


Figura 32. Biblioteca PANS [11]

5.9.1. SoftDevice y controlador BLE

SoftDevice es una biblioteca utilizada para funciones *bluetooth* de baja energía de *Nordic Semiconductor*. El *SoftDevice* utilizado es el S132 y es el encargado de enlazar múltiples cometidos simultáneamente con las aplicaciones bluetooth. El controlador BLE es el encargado de proporcionar al módulo DWM1001 características para crear redes complejas, comunicaciones y actualizaciones de *firmware* mediante el *bluetooth*, a través del aire.

5.9.2. eCosRTOS

Es un sistema operativo en tiempo real de código abierto. Es utilizado ya que genera buen rendimiento y lleva integrado un paquete de soporte de placa o en ingles *Board Support Package* (BSP) que permite que el sistema operativo funcione en el *hardware* del módulo DWM1001. Este sistema operativo incluye los componentes necesarios para el funcionamiento del módulo como pueden ser acelerómetro, *bluetooth* de baja energía y el controlador DW1000.

El controlador DW1000 es un conjunto de funciones de bajo nivel encargadas de ejecutar las funciones básicas del transceptor DW1000 sin necesidad de acceder al dispositivo a través de su interfaz SPI.

5.9.3. IoT IEEE 802.15.4 MAC

Las tramas UWB se envían con el formato estándar 802.15.4. La capa MAC es la encargada de controlar la red para instalar y poner en funcionamiento los nodos, enviar las tramas TWR, de realizar intercambios de datos, acceso múltiple con prevención de colisiones, transmisiones garantizadas y del sincronismo de los nodos. En la Figura 33 se puede ver el formato que utilizan estos tipos de tramas.

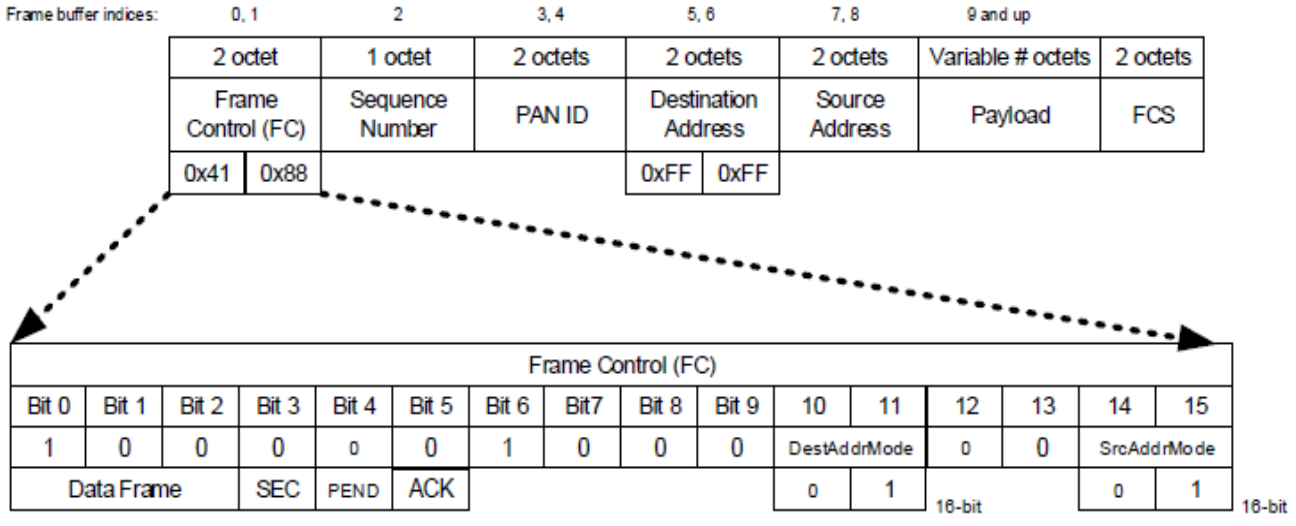


Figura 33. Formato IEEE 802.15.4 [7]

5.9.4. RTLS Management y solver TWR o Location Engine

La aplicación *RTLS Management* es usada para la configuración de los nodos como *anchors* o *tags*. Una vez configurados los nodos, el motor de ubicación interno en los *tags* calcula las coordenadas x, y, z enviando los resultados a través de la API.

5.10. Cadena de herramientas del Firmware

La cadena de herramientas es la mostrada en la Figura 34. Es utilizada para desarrollar una nueva aplicación añadiendo funcionalidades al módulo DWM1001 que se ejecutarán en la parte superior de la biblioteca PANS.

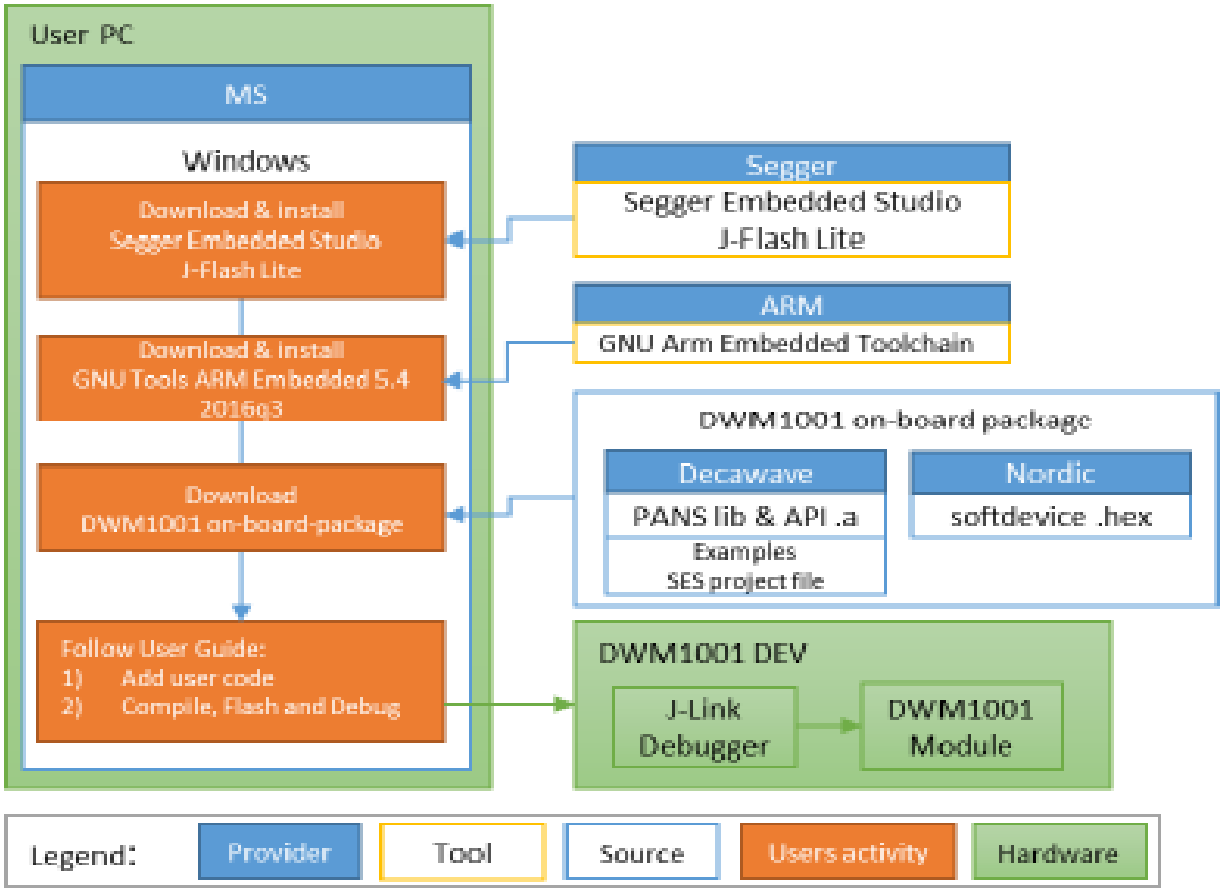


Figura 34. Cadena de herramientas [10]

Las herramientas principales para el desarrollo del *firmware* son el programa *Segger Embedded Studio*, el cual es ideal para el desarrollo firmware ya que cuenta con una licencia gratuita para

el microcontrolador nRF52832 de *Nordic Semiconductor* que es el utilizado en nuestro estudio y el Toolchain GNU ARM Embedded 5.4 el cual es un conjunto de herramientas de código abierto, que permite usar y compilar lenguajes de programación C, C++ y programación de ensamblajes para las familias de procesadores *Arm Cortex-A*, *Arm Cortex-M* y *Arm Cortex-R* de 32 bits.

Además de estas dos herramientas principales para el desarrollo del *firmware* es necesario disponer de una parte *hardware*, vista en la Figura 34 en verde, como son un ordenador con sistema operativo *Microsoft Windows* y la tarjeta de desarrollo DWM1001-DEV la cual consta del módulo DWM1001 y el depurador J-Link para realizar la depuración y la descarga del nuevo *firmware*.

Otra parte *software* necesaria es el programa J-Flash Lite con el que se puede realizar tanto un borrado total del *firmware* del módulo DWM1001 como la descarga de un nuevo *firmware*.

El paquete integrado DWM1001 con el cual se empezará a desarrollar y añadir funcionalidades y distribuido por *Decawave* contiene los archivos de origen de la aplicación de usuario y las bibliotecas necesarias para compilar y construir el *firmware* de usuario DWM1001. Este paquete está compuesto por la biblioteca PANS, los archivos binarios de *firmware* como el *softdevice.hex* o el gestor de arranque y ejemplos de aplicaciones de usuario para mostrar el funcionamiento de las interfaces UART, SPI o ioT. En la Figura 35 se muestra la estructura del paquete a bordo para el módulo DWM1001.

```
├── examples                                  // Examples directory
│   ├── dwm-simple-example                    // Simple Example
│   │   ├── dwm-simple.c                      // C source file
│   │   ├── dwm-simple-example.emProject      // SES project file
│   │   └── Output                            // Output directory
│   │       ├── compiler                      // Compiler output: .o
│   │       └── linker                        // Linker output: .elf
│   ├── dwm-range-iot
│   ├── dwm-timer
│   └── dwm-uart
├── include                                   // PANS Header file
├── lib                                       // PANS compiled library
├── nordicsemi                                // Nordic SDK 12.1.0
├── recovery                                  // PANS reference binaries
└── utilities                                 // Firmware complementary images
    └── generate_example_hex                  // Softdevice driver for BLE
```

Figura 35. Estructura del paquete a bordo [10]

5.11. Puerta de enlace o Gateway

Una puerta de enlace se consigue configurando una tarjeta de desarrollo DWM1001-DEV como nodo puente o *bridge node* y conectándola a una *Raspberry Pi* a través de los pines como se observa en la Figura 36.

Figura 36. Raspberry Pi con tarjeta de desarrollo (puerta de enlace) [12]

Una vez configurada la tarjeta DWM1001-DEV, es hora de configurar la *Raspberry Pi*. Para ello, primero se flashea la imagen proporcionada por *Decawave* en una tarjeta micro-SD que será la que proporciona el sistema operativo a la *Raspberry Pi*. Una vez funcionando el sistema operativo, hay que conocer la dirección IP de esta ya que será necesaria para la configuración y acceder al intermediario MQTT y a la página web integrada en la imagen y de esta manera poder ver la información de la red e interactuar con los nodos.

Luego, en el script de configuración dwm1001.config, situado en el directorio de la *Raspberry Pi* /etc/dwm1001/, se deben de configurar las características del nodo puente como son PAN ID, es decir, la dirección que identifica una red específica, la clave de cifrado, por si se quieren mandar los mensajes UWB encriptados y por último la configuración de la dirección IP del servidor *proxy*. Dependiendo de si la puerta de enlace es el servidor *proxy* principal o no, el Daemon se debe de configurar con unos valores u otros.

- Puerta de enlace actuando como servidor *proxy host*: en este caso la configuración es la observada en la Figura 37. Como se ve en la imagen esta puerta de enlace con IP 192.168.72.94 es el *host proxy* y por tanto la variable *proxy_server_host* se configura como “localhost”.

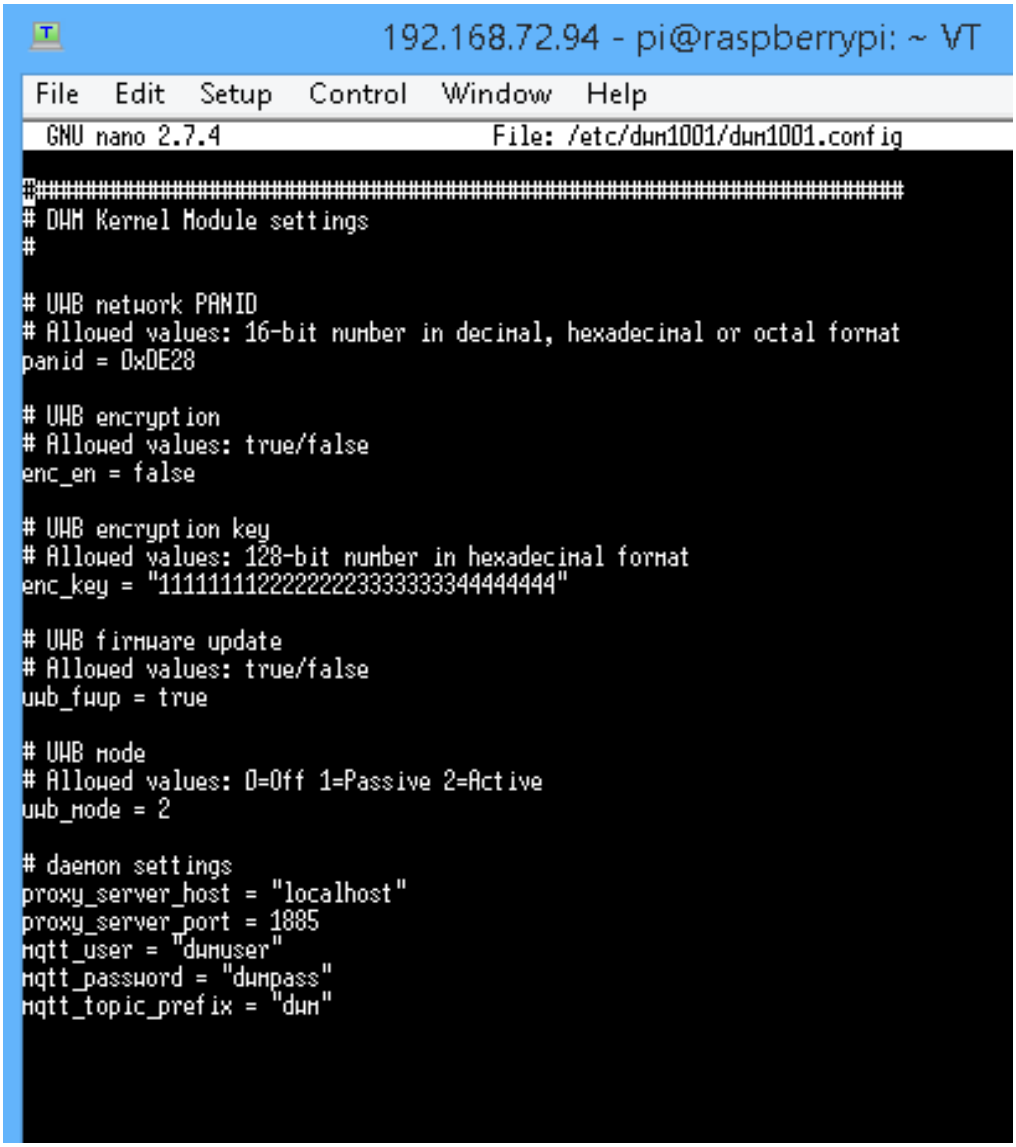


Figura 37. Configuración de la puerta de enlace servidor proxy

- Puerta de enlace actuando como *Daemon*: en este caso la configuración es la observada en la Figura 38. Ahora hay que redireccionar los datos a la IP del servidor *proxy* que es el que interactúa con el MQTT y, por tanto, la variable *proxy_server_host* se configura con la IP del *proxy* “192.168.72.94”.

```
################################################################################
# DWM Kernel Module settings
#

# UWB network PANID
# Allowed values: 16-bit number in decimal, hexadecimal or octal format
panid = 0xDE28

# UWB encryption
# Allowed values: true/false
enc_en = false

# UWB encryption key
# Allowed values: 128-bit number in hexadecimal format
enc_key = "11111111222222223333333344444444"

# UWB firmware update
# Allowed values: true/false
uwb_fwup = true

# UWB mode
# Allowed values: 0=Off 1=Passive 2=Active
uwb_mode = 2

# daemon settings
proxy_server_host = "192.168.72.94"
proxy_server_port = 1885
mqtt_user = "dwmuser"
mqtt_password = "dwmpass"
mqtt_topic_prefix = "dwm"
```

Figura 38. Configuración de puerta de enlace de modo Daemon

Con la puerta de enlace funcionando, la red de localización queda como muestran la Figura 39 y Figura 40 y se puede acceder a los datos en MQTT o en la página web a través de la dirección IP del servidor *proxy host* 192.168.72.94. En las dos figuras se muestra la misma red, pero en la primera se ve más a bajo nivel ya que muestra también el MQTT.

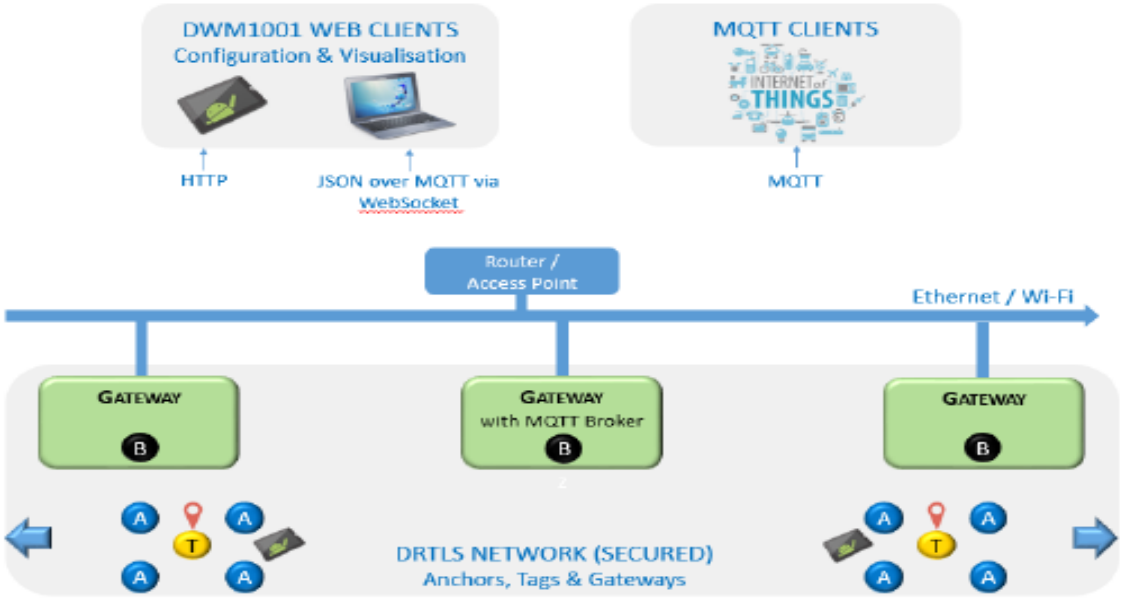


Figura 39. Ejemplo de red a bajo nivel [7]

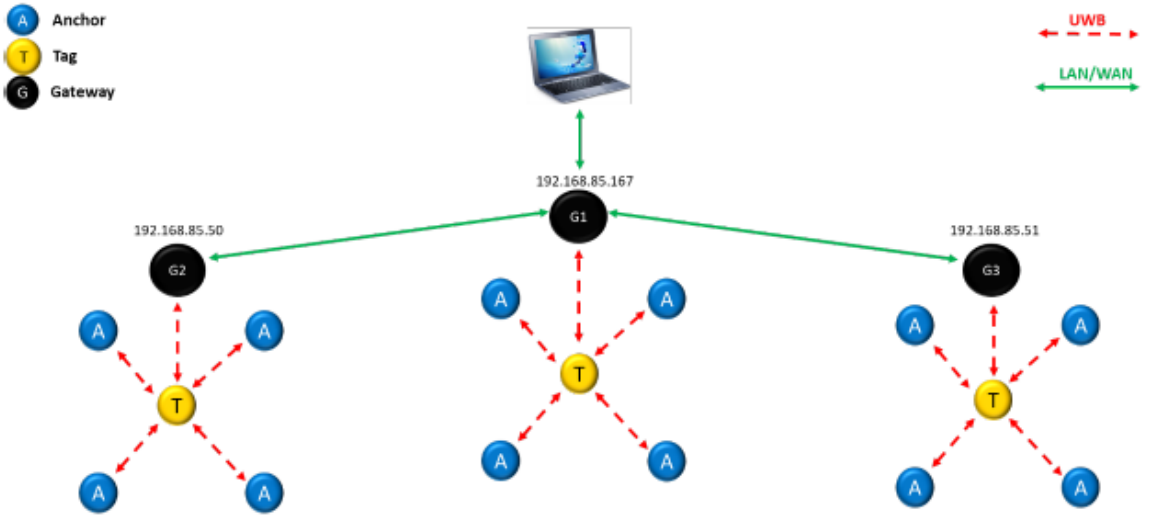


Figura 40. Ejemplo de red DRTLS [12]

La puerta de enlace se basa en MQTT Broker, el cual es un protocolo de comunicación basado en la pila TCP/IP. Los componentes de la puerta de enlace se observan en la Figura 41.

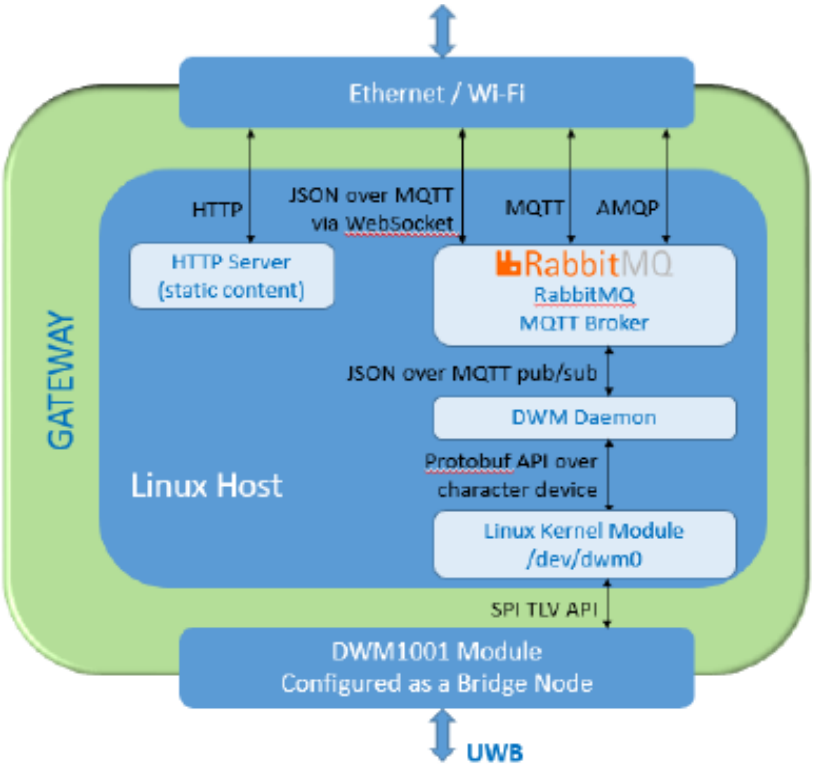


Figura 41. Estructura de una puerta de enlace [7]

5.11.1. Funcionamiento MQTT

MQTT es un servicio de mensajería con un patrón de publicador y suscriptor. Los clientes se conectan a un servidor al cual se le conoce como *Broker*. En este servidor el cliente se puede suscribir a un tema en concreto y verá los mensajes publicados en ese tema por el cliente que lo publicó. Este funcionamiento se ve en la Figura 42.

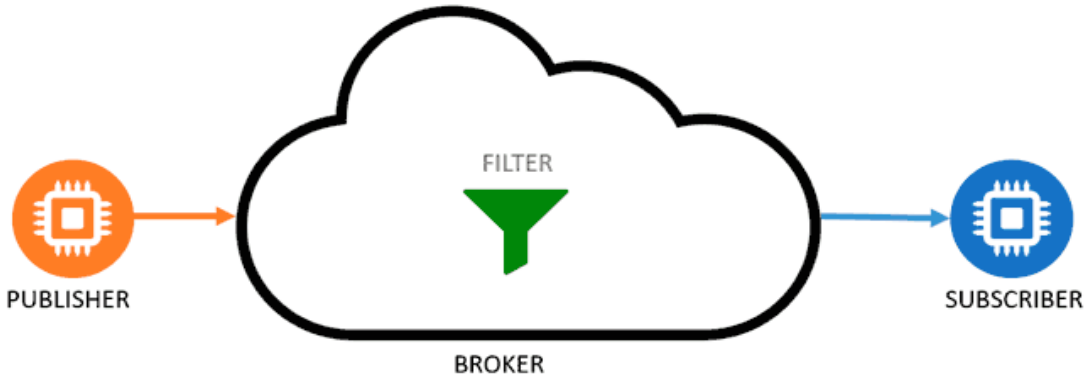


Figura 42. Funcionamiento MQTT [13]

En este caso la comunicación puede variar en dos posiciones: sentido ascendente, es decir, desde el nodo DWM1001 hacia MQTT/WEB o en sentido descendente, desde MQTT/WEB hacia el nodo DWM1001.

5.11.1.1. Enlace ascendente o uplink

La cadena de datos en sentido ascendente pasa por las siguientes fases: Nodo DWM1001 – Nodo puente – Módulo de Kernel Raspberry Pi – Daemon de DWM1001 – Proxy DWM1001 – MQTT – Web Manager o Cliente MQTT.

1. Una vez el *Tag* o el *anchor* envía los mensajes mediante UWB, el nodo actuando como puente recopila los datos de localización y de IoT y se los pasa a la *Raspberry Pi* mediante SPI. La puerta de enlace (Nodo puente más *Raspberry Pi*) podrá recibir información de los nodos dentro de su rango.

2. Los datos recibidos por la puerta de enlace se envían al *Daemon* de DWM1001 correspondiente, el cual habrá uno por puerta de enlace. El *Daemon* es un proceso de la *Raspberry Pi* que se ejecuta de forma continua en segundo plano y es útil para tener activos constantemente procesos como SSH o los servidores web.

3. El *Daemon* correspondiente reenviará los datos al *proxy* principal y único *proxy* de la red situado en la puerta de enlace principal. Este *proxy* funciona como filtro de datos para este enlace.

4. El *proxy* manda los datos filtrados al MQTT *Broker*.
5. Por último, los datos son recibidos por la página web que se comunica con MQTT a través de *WebSockets*. Además, estos datos son publicados en un tema de MQTT y se pueden observar suscribiéndote en el tema correspondiente utilizando una aplicación como MQTT.fx.

5.11.1.2. Enlace descendente o dowlink

El flujo de datos en sentido descendente es el siguiente: Web/Cliente MQTT – MQTT – *Proxy* principal – *Daemon* – *Raspberry Pi* – Nodo puente – Nodo DWM1001.

1. El cliente publica los datos en el MQTT *Broker* mediante la web o mediante el cliente MQTT. Estos datos se podrán ver suscribiéndose en el tema correspondiente.
2. El *Proxy* que en este caso hace de enrutador de los datos, se suscribe al MQTT *Broker* para recibir los temas del enlace y obtendrá los datos para enviárselos al *Daemon* correspondiente.
3. Este *Daemon* tramitará los datos en la *Raspberry Pi* y los enviará a través de SPI al nodo que actúa de puente.
4. El nodo puente envía los datos recibidos al *anchor* o el *tag* correspondiente a través de *Ultra Wade Band.*
5. Como los mensajes no tiene confirmación a menos que se programe un mensaje de vuelta, hay datos que pueden perderse debido a que en ese momento el nodo de recepción no esté en el rango. Por eso es recomendable programar un mensaje de confirmación de envío.

5.12. Material necesario

El material necesario para desarrollar el trabajo son dos kits MDEK1001, las fuentes de alimentación para alimentar los dispositivos y dos *Raspberry Pi*.

5.12.1. Kit MDEK1001

El contenido del kit MDEK1001 es el siguiente:

- 12 tarjetas de desarrollo DWM1001-DEV que pueden ser configurables como *anchors*, *tags* o *bridge*, estas tarjetas vienen insertadas en su respectiva carcasa de plástico.

Figura 43. Dispositivo DWM1001-DEV [14]

- 1 cable USB para depurar y descargar el *firmware* en la tarjeta de desarrollo.
- 4 conectores de ángulo recto que pueden servir para realizar una mejor conexión de la tarjeta.
- 8 almohadillas adhesivas de pared para poder fijar los dispositivos.
- Una guía de inicio rápido para conectar un sistema sencillo de posicionamiento.

5.12.2. Fuentes de alimentación

Para alimentar la tarjeta de desarrollo DWM1001-DEV están permitidas varias fuentes de energía.

- **Alimentación mediante conexión USB**: es la forma recomendada sobre todo para los *anchors* ya que están fijos y no son tan eficientes en energía como los *tag*s ya que están continuamente encendidos y esperando a estar en el rango para intercambiar datos con los *tag*s. La tarjeta de desarrollo requiere una alimentación de 5 voltios y al menos 500 mA.
- **Power Bank o batería externa portátil:** es una forma más cómoda de alimentar los *tags* ya que estos se estarán moviendo por la red y no pueden estar conectados mediante USB a la luz. Es válida cualquier batería que cumpla con un voltaje entre 3.6 V y 5.5 V y 500mA.

- **Baterías o pilas de litio:** al igual que con las baterías portátiles se recomienda su uso para los *tags*. Existen dos modelos válidos que cumplen con los requisitos como son: 3.7V RCR123a y 16340 recargable.

5.12.3. Raspberry Pi 3 Modelo B

La *Raspberry Pi* escogida es la *Raspberry Pi* 3 modelo B ya que el sistema operativo proporcionado por *Decawave* está diseñada para ella.

La *Raspberry Pi* es un pequeño ordenador del tamaño de una tarjeta de crédito muy funcional y de bajo precio y consta de una tarjeta base sobre la que se monta un procesador, un chip gráfico y una memoria RAM.

Para conectar correctamente la tarjeta de desarrollo DWM1001-DEV a la *Raspberry Pi* es necesario conocer para que se usa cada pin y esto es mostrado en la Figura 44.

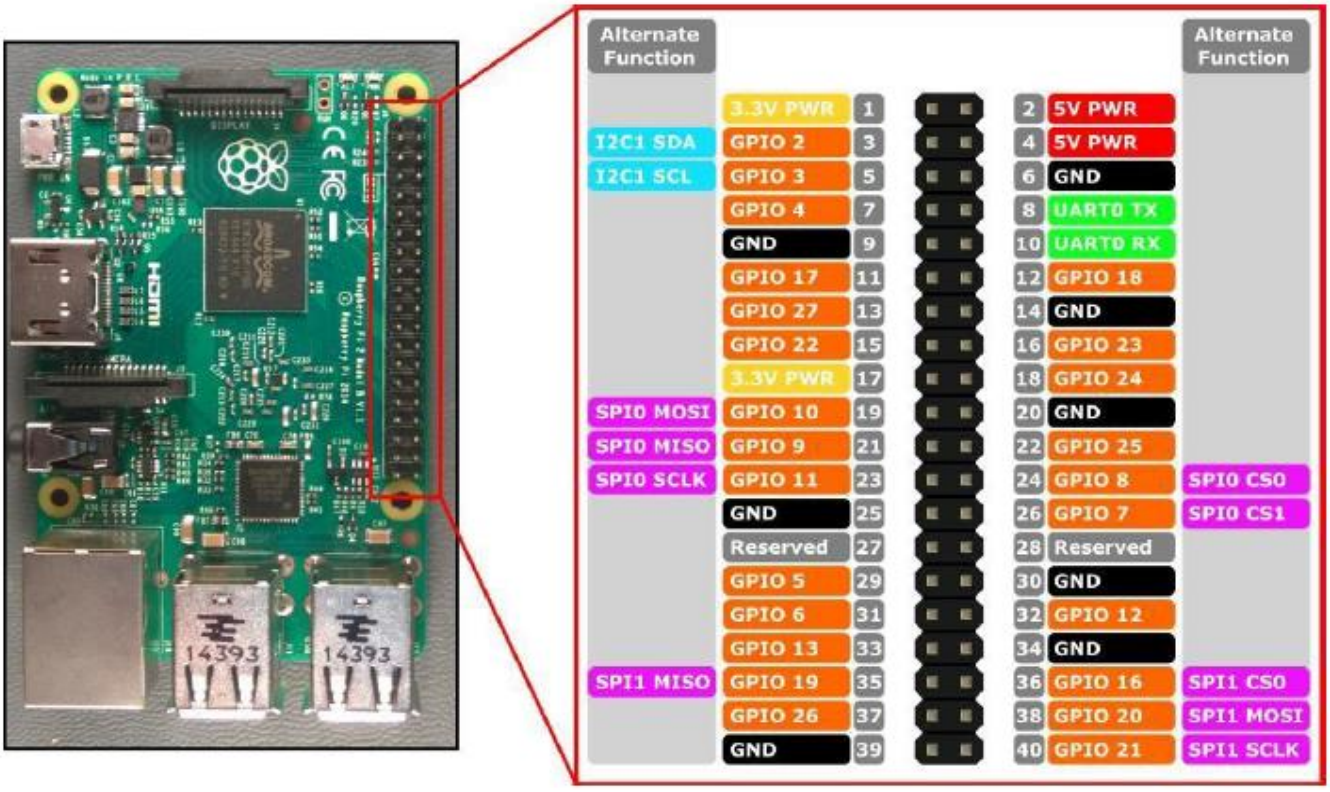


Figura 44. Raspberry Pi con conexionado [5]

6. PRUEBAS EXPERIMENTALES

6.1. Introducción

En esta sección se llevarán a cabo las pruebas experimentales del sistema para ver su funcionamiento. Se configura e instala una red en la Escuela Politécnica Superior de la Universidad de Alcalá y se realizan varias pruebas para comprobar el correcto funcionamiento. Primero se explicará el funcionamiento del código para calcular la ubicación, luego donde se instalan los *anchors* y las distintas opciones que existen para configurarlos y observar la red y por último se realizarán varias pruebas para ver el funcionamiento y se tratarán los datos obtenidos con *MatLab* para ver el error cometido a la hora de localizar el *tag*.

La Escuela Politécnica de la Universidad de Alcalá se divide físicamente en 4 sectores llamados según los puntos cardinales, Norte, Sur, Este y Oeste y que corresponden a los colores azul, amarillo, verde y rojo respectivamente tal y como se observa en la Figura 45.

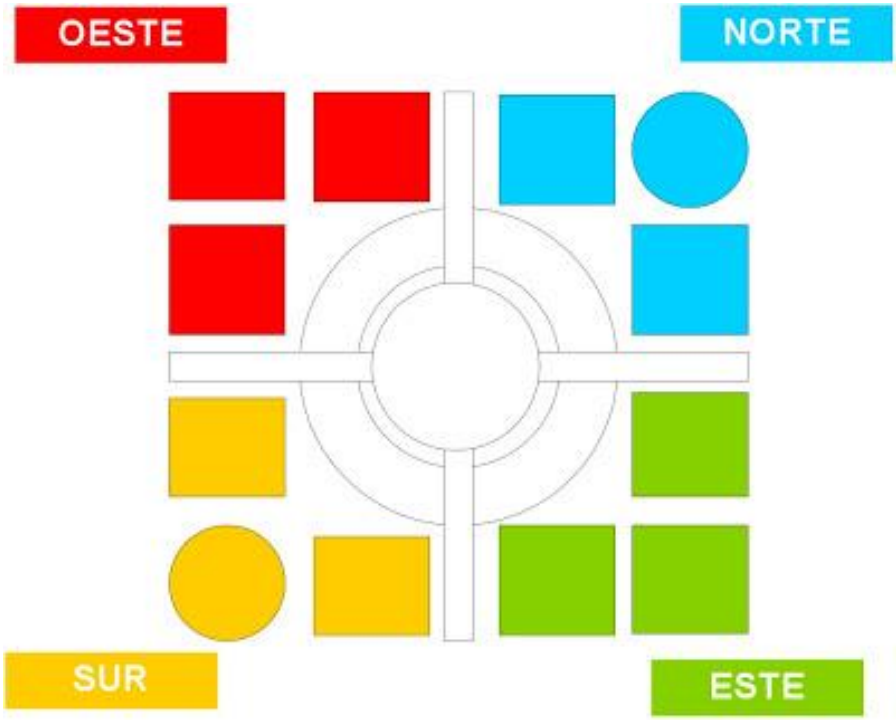


Figura 45. Mapa de la Escuela Politécnica Superior [15]

La red de localización se ha utilizado para balizar el Oeste de la segunda planta y la sala en el fondo del primer pasillo del Norte de la segunda planta N-21.

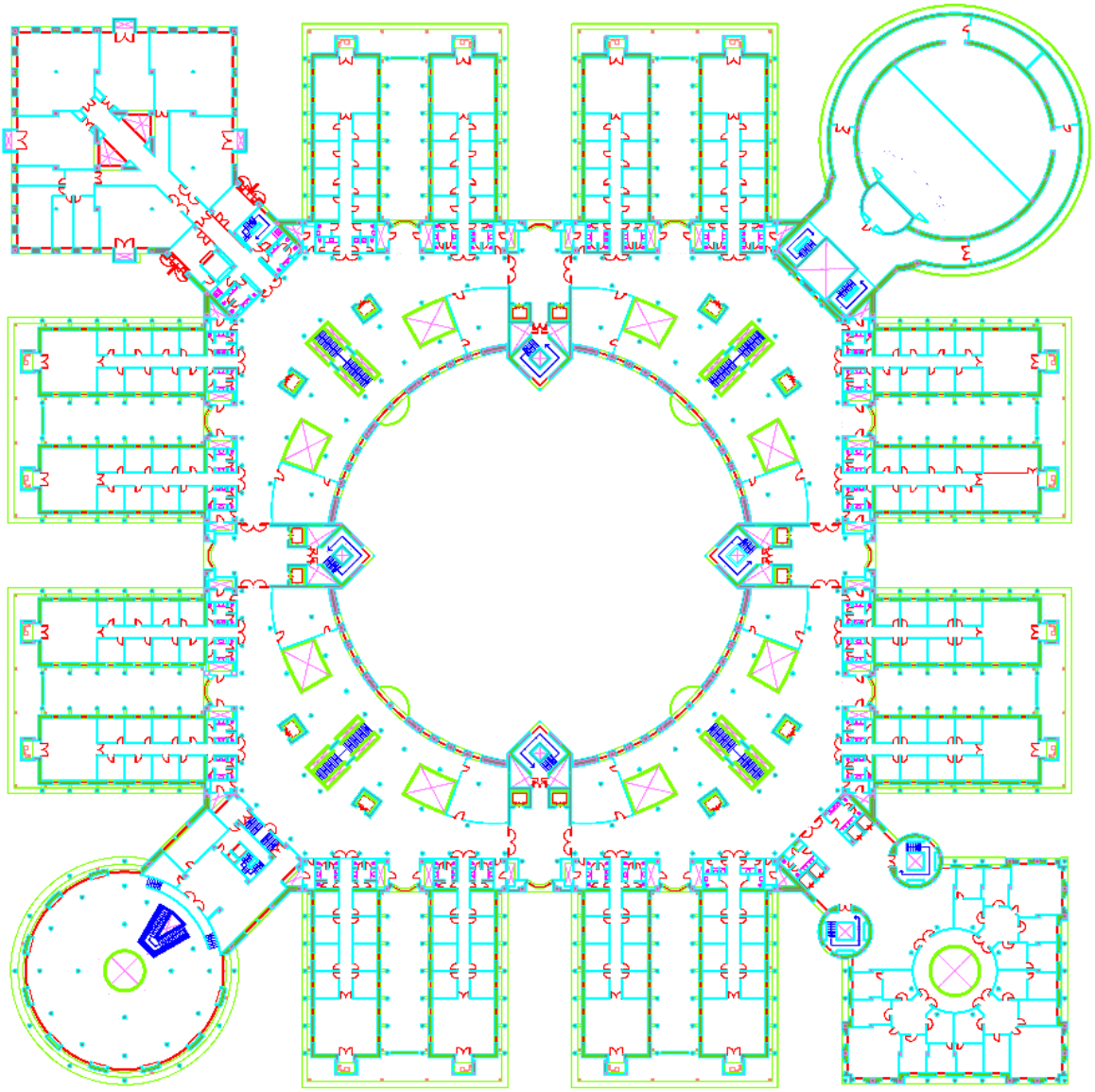


Figura 46. Mapa de la segunda planta de la Escuela Politécnica Superior

6.2. Desarrollo software del *Tag*

El funcionamiento del código es el mostrado en el flujograma de la Figura 47.

- **Función *dwm_user_start():*** cuando se inicia o se resetea el *tag*, el programa empieza con la función *dwm_user_start()* donde se inicializan y se añaden a la compilación el modo UART *Shell*, el modo *bluetooth*, el motor de localización y el modo SPI. Después se crea un hilo de nombre *app_thread_entry()* que será un subproceso encargado de ejecutar el programa del sistema DWM1001. Una vez creado el hilo, se iniciará y se pasa a la función *app_thread_entry().*

- **Función *app_thread_entry():*** esta es la función principal del sistema, es el equivalente a la función *main* en otros sistemas. En esta función se configura la tasa de actualización y la tasa de actualización estacionaria.

  Luego, con la función *dwm_evt_listener_register()* se registran los eventos requeridos como por ejemplo, que ha finalizado el cálculo de la posición. Por último, se introduce en la función *while (1)* donde se permanecerá constantemente esperando un evento, para ello se llama a la función *dwm_evt_wait()*, cuando esta función devuelva DWM_OK es que ha ocurrido un evento y se pasa a la función *on_dwm_evt()*. Si no se utilizara la función *dwm_evt_wait()* para esperar un evento, el *buffer* de los eventos se desbordaría.

- **Función *on_dwm_evt()*:** esta función es la encargada de identificar y ejecutar el evento ocurrido. Pueden existir 5 eventos diferentes:

  - **DWM_EVT_LOC_READY**: tiene el identificador número 1. Este evento ocurre cuando una localización es calculada por el motor de localización y la muestra al usuario.

  - **DWM_EVT_UWBMAC_JOINED_CHANGED**: tiene el identificador número 2 y ocurre cuando un dispositivo se une a una red de UWB.

  - **DWM_EVT_BH_INITIALIZED_CHANGED:** tiene un número identificador de 16 y ocurre cuando una inicialización de un dispositivo ha cambiado.

  - **DWM_EVT_USR_DATA_READY:** tiene el identificador 64. Este evento ocurre cuando un dato es recibido por IoT desde MQTT o desde la página web. Este dato se muestra al usuario con la función *dwm_usr_data_read().* Dependiendo del dato recibido, se puede configurar el dispositivo para que devuelva una respuesta o reaccione de alguna manera como por ejemplo encender un led.

  - **DWM_EVT_USR_DATA_SENT:** tiene el identificador número 128 y ocurre cuando se requiere enviar un dato por IoT hacia MQTT y la página web. Los datos recibidos y enviados deben de ser en hexadecimal y con una longitud máxima de 34 bytes.

  Después del evento ocurrido el dispositivo puede volver a dormir y se volverá al while (1) de la función *app_thread_entry()* y esperará el siguiente evento.

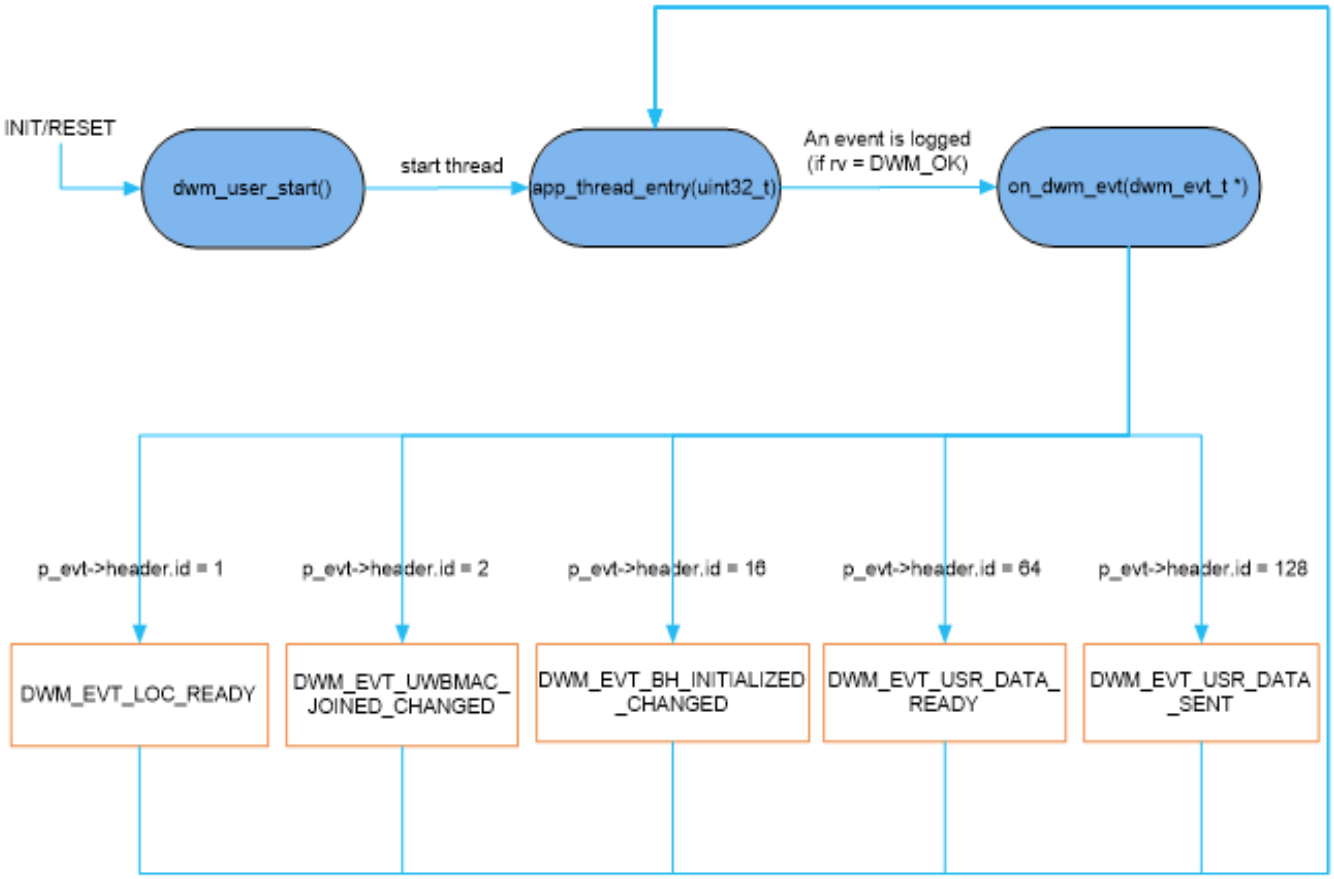


Figura 47. Flujograma del funcionamiento del código

6.3. Configuración y estudio de la red y obtención de datos

La red de localización y los distintos dispositivos que la componen pueden ser configurados mediante varios métodos como son: a través de UART *Shell*, mediante la aplicación de *Android DRTLS Manager*, con una página web conectada en la puerta de enlace o mediante el protocolo MQTT. Además de la configuración de los dispositivos, en la aplicación de *Android* y en la página web es posible introducir un mapa del edificio que se desea balizar y así poder observar la posición de los *anchors* y el *tag* moviéndose en tiempo real.

6.3.1. UART Shell

Para configurar y obtener datos de la red a través de UART *Shell* se hace uso de los siguientes comandos los cuales se dividen en varios grupos según la aplicación:

- **Comandos Base:**
  - **?:** muestra ayuda de los comandos disponibles.
  - **help:** tiene la misma función que el anterior.

- **quit:** se utiliza para salir del modo *Shell*.

- **Comandos GPIO:**
  - **gc:** borrar el pin GPIO seleccionado.
  - **gg:** lee el nivel del pin GPIO seleccionado, 0 o 1.
  - **gs:** establece el pin GPIO seleccionado como salida y con su nivel deseado.
  - **gt:** alterna el nivel del pin GPIO.

- **Comandos del sistema:**
  - **f:** muestra la memoria libre.
  - **ps:** muestra los subprocesos en ejecución.
  - **pms:** muestra tareas PM.
  - **reset:** reinicia el dispositivo.
  - **si:** muestra información del sistema.
  - **ut:** muestra el tiempo de actividad del sistema.
  - **frst:** restablece el dispositivo a valores de fábrica.

- **Comandos de sensibilidad:**
  - **twi:** lectura de I2C de propósito general, por ejemplo, para leer el id del acelerómetro.
  - **aid:** muestra el id del acelerómetro.
  - **av:** lee los valores del acelerómetro.
  - **scs:** establecer configuración estacionaria.
  - **scg:** obtener configuración estacionaria.

- **Comandos del motor de localización:**
  - **les:** muestra las distancias del *tag* a cada *anchor* y la posición del *tag*.
  - **lec:** muestra las medidas del *tag* a cada *anchor* y la posición del *tag* en formato CSV.
  - **lep:** muestra la posición del *tag* en formato CSV.

- **Comandos UWB:**
  - **utpg:** obtiene la potencia de transmisión.
  - **utps:** establece la potencia de transmisión.

- **Comandos de la capa UWBMAC:**
  - **nmg:** obtiene información del modo del nodo.
  - **nmp:** configura un nodo como pasivo.
  - **nmo:** configura el nodo sin conexión pasiva.
  - **nma:** configura el nodo como un *anchor*, activo.
  - **nmi:** configura el nodo como un *anchor* iniciador, activo.
  - **nmt:** configura el nodo como *tag*, activo.
  - **nmtl:** configura el nodo como *tag*, activo y de baja potencia.

- **nmb:** configura el nodo como *bridge*.
- **bpc:** alterna entre BW y TxWR.
- **la:** muestra una lista de los *anchors*.
- **lb:** muestra una lista de los nodos configurados como *bridge*.
- **nis:** establece un identificador ID para la red.
- **nls:** obtiene el nombre dl nodo.
- **stg:** muestra estadísticas como tiempo del sistema desde el reinicio, contador de errores de la API, número de interrupciones del DW100, etc.
- **stc:** borra las estadísticas.

- **Comandos de la API:**
  - **tlv:** se utiliza para enviar tramas TLV.
  - **uui:** envía datos de subida por IoT.
  - **udi:** muestra los datos IoT recibidos.
  - **aurs:** establece la tasa de actualización para el cálculo de la posición.
  - **aurg:** obtiene la tasa de actualización.
  - **apg:** obtiene la posición del nodo.
  - **aps:** establece la posición del nodo.
  - **acas:** configura el nodo como *anchor* con las opciones deseadas.
  - **acts:** configura el nodo como *tag* con las opciones deseadas.
  - **aks:** establece la clave de cifrado.
  - **akc:** borra y deshabilita la clave de cifrado.
  - **ans:** escribe datos de usuario en memoria no volátil.
  - **ang:** lee datos de la memoria no volátil.

6.3.2. MQTT

Para utilizar el protocolo de transportes de mensajes MQTT, se ha utilizado la aplicación para Windows, MQTT.fx.

Para la configuración del MQTT *Broker* es necesario introducir la dirección IP del servidor de la red, es decir, la dirección IP de la *Raspberry Pi* que actúa como servidor. Esta configuración se puede observar en la Figura 48.

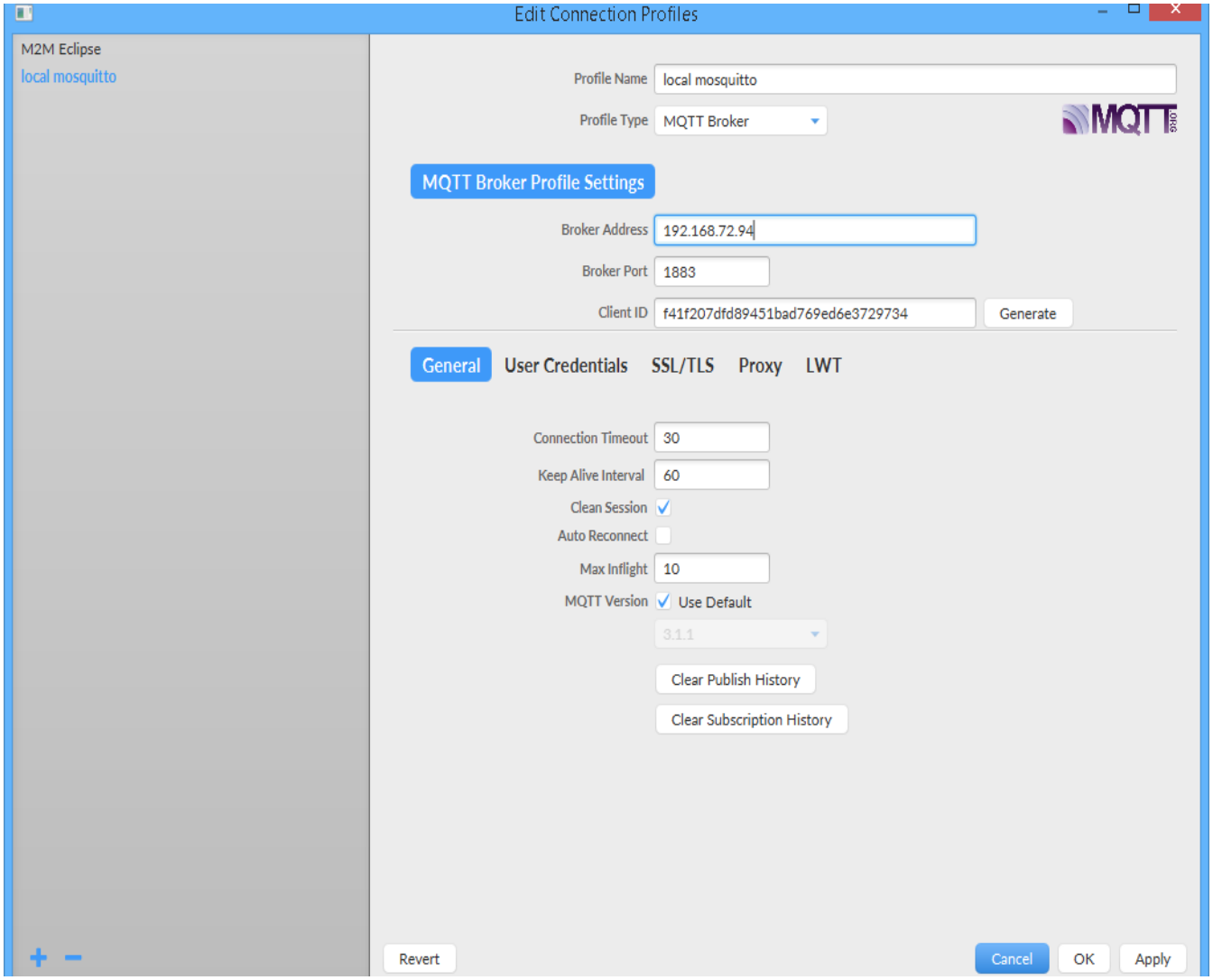


Figura 48. Configuración MQTT

Una vez configurado, ya es posible conectarse y así poder obtener los datos de localización de la red e interactuar con ella enviando o recibiendo mensajes IoT. Para ello, es necesario inscribirse en los temas necesarios como localización del *tag*, mensajes recibidos del *tag* o mensajes enviados del *tag*. Como se observa en la Figura 49, abajo en la izquierda, en la ventana llamada *Topics Collector* se pueden ver todos los temas en los que el usuario se puede subscribir.

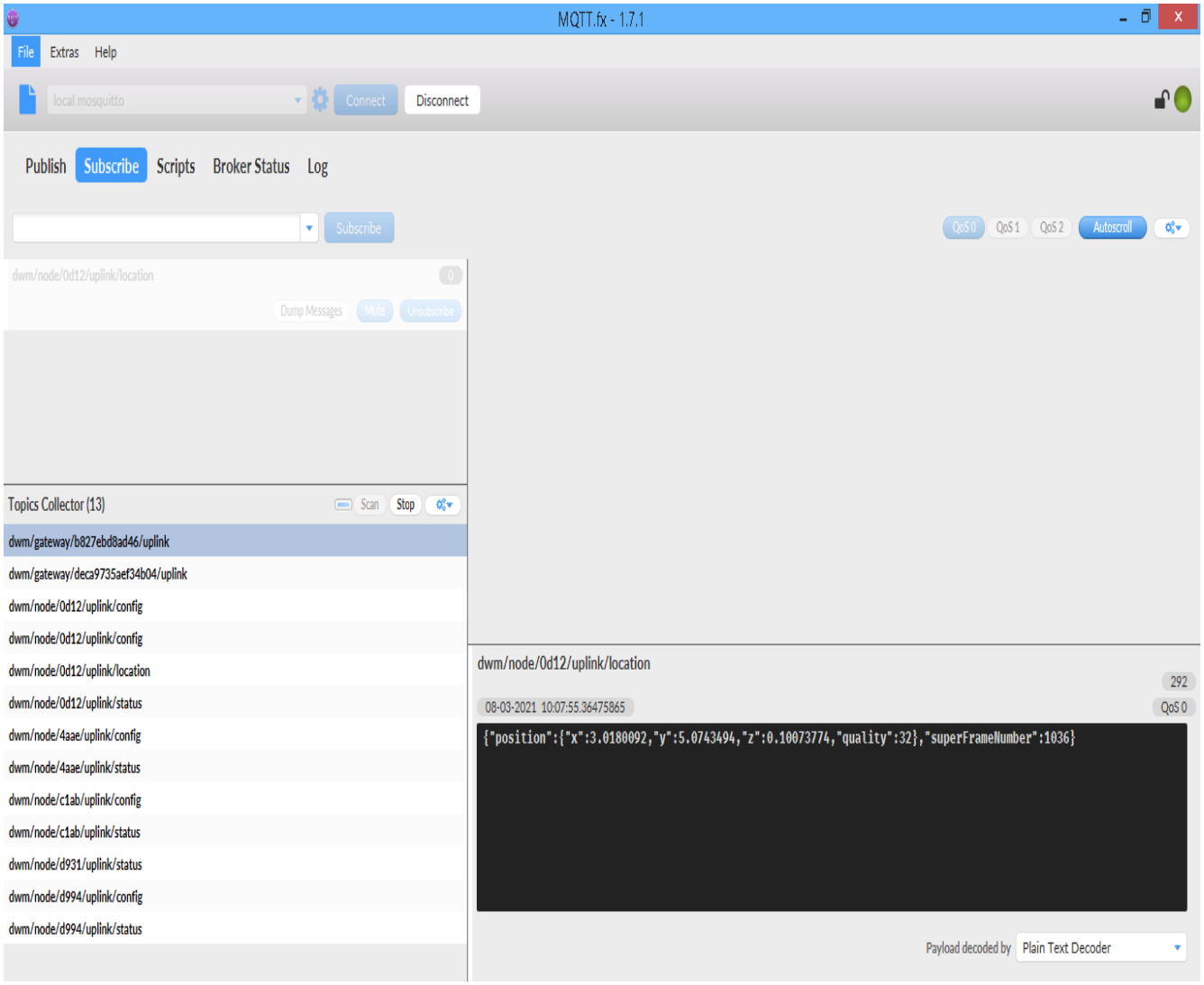


Figura 49. Pantalla principal para subscribirse los temas en MQTT.fx

Una vez el usuario se subscribe a un tema, los datos de ese tema serán mostrados en la ventana inferior derecha. Los temas para subscribir pueden ser:

- Configuración del dispositivo: pueden ser de un *tag* (Figura 50) o de un *anchor* (Figura 51) y donde se muestran las variables configuradas en ellos.

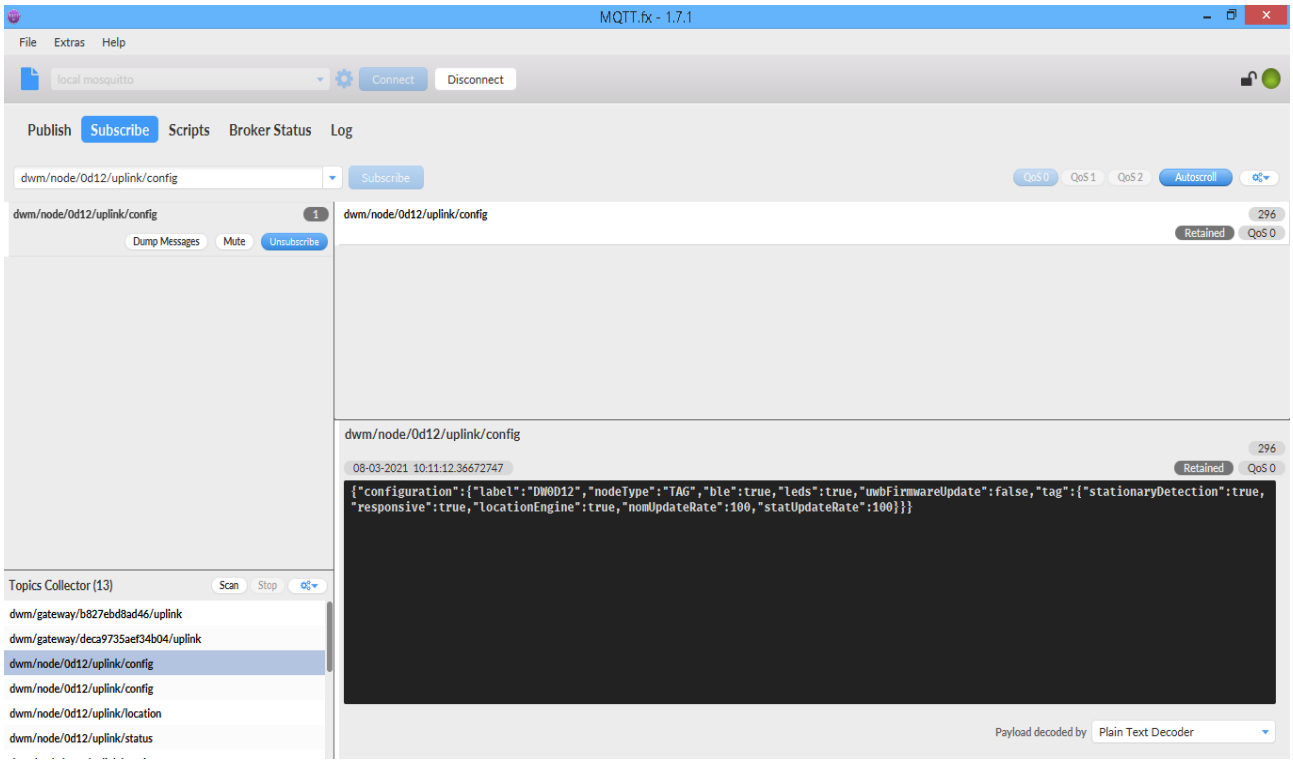


Figura 50. Suscrito en tema de configuración del tag

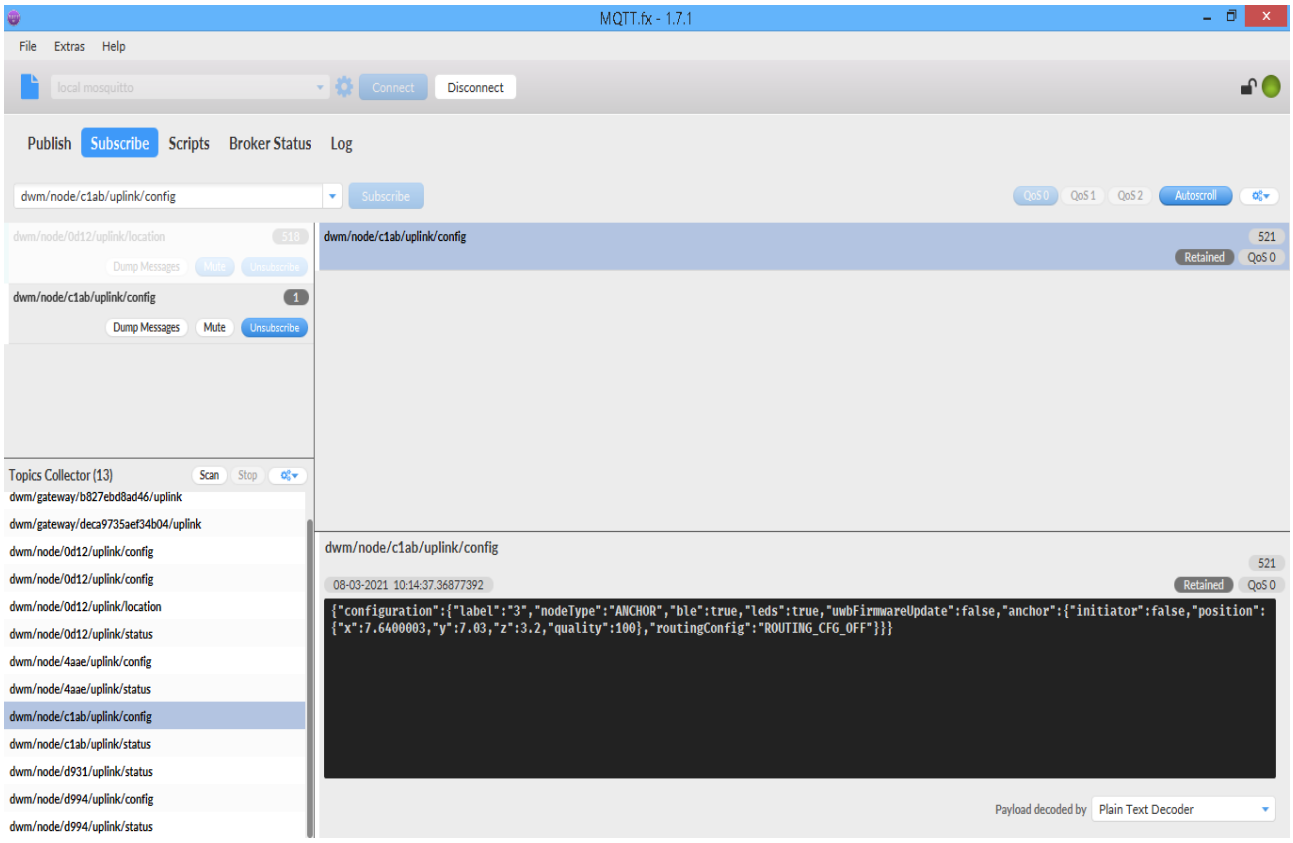


Figura 51. Suscrito en tema de configuración del anchor

- Localización del *tag*: muestra las coordenadas x, y, z y la calidad en las medidas del *tag*.

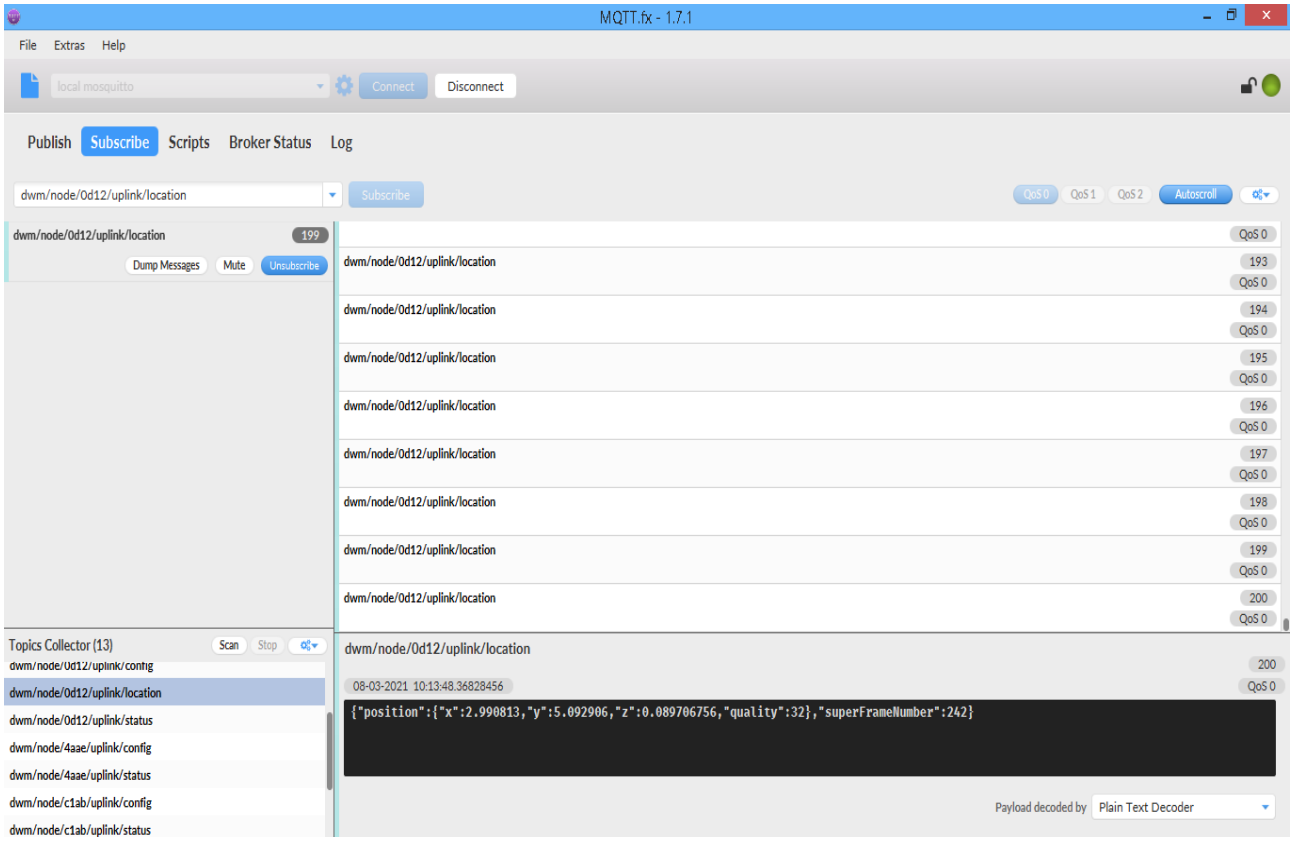


Figura 52. Tema de posicionamiento del tag

- Datos enviados del dispositivo a la web: los datos enviados desde un dispositivo hacia la web o el MQTT tienen el tema de /uplink/data.
  Un ejemplo de aplicación puede ser que el usuario pulsa el botón “user” del *tag* y este envía un mensaje de ayuda hacia el MQTT *Broker* y la web. Como se ve en la Figura 53, el mensaje mostrado por MQTT es “QVIVREE=”, este mensaje está codificado en Base64 y si se decodifica, su significado es “AYUDA”.
  Otro ejemplo creado en el dispositivo mostrado en la Figura 54, es que muestre un mensaje de una posible caída cuando se detecta mediante el acelerómetro una caída en el dispositivo. El mensaje en base64 mostrado es “UE9TSUJMRSBDQUlEQQ==” que si se decodifica significa “POSIBLE CAIDA”.
  Para una buena aplicación y ver los mensajes recibidos inmediatamente, se debería crear un decodificador de Base64, lo cual no es el propósito en este trabajo.

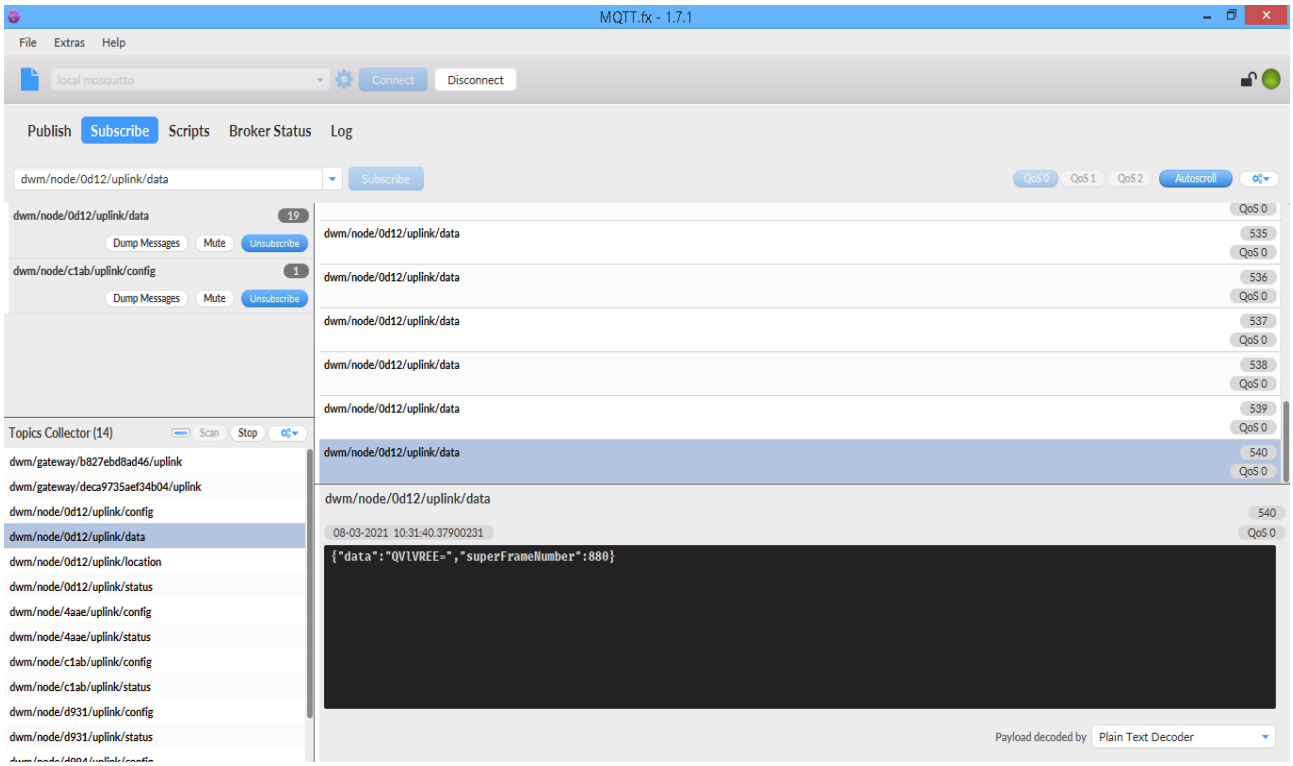


Figura 53. Mensajes uplink de AYUDA

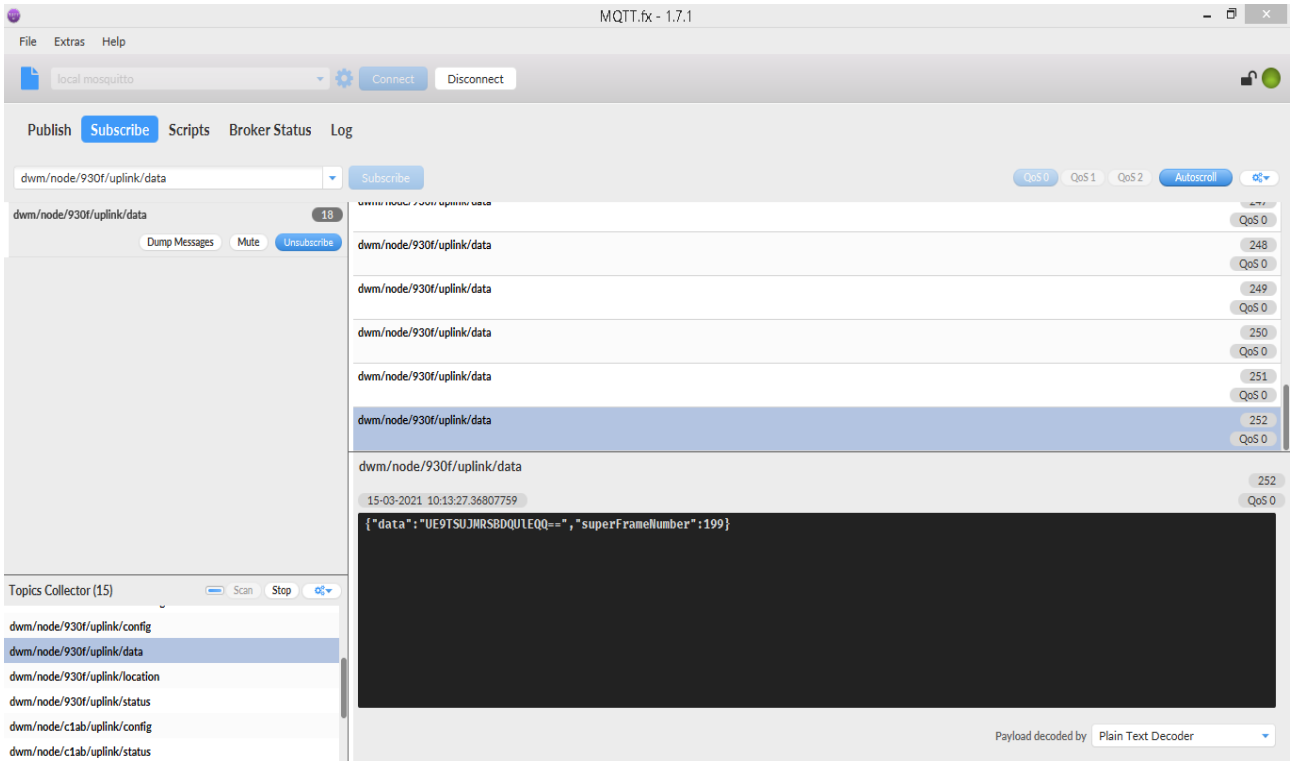


Figura 54. Mensaje uplink de POSIBLE CAIDA

- Datos enviados desde la web al dispositivo: estos mensajes se pueden ver inscribiéndose en el tema /dowlink/data. Desde la web solo es posible mandar mensajes en hexadecimal, por tanto, en estos ejemplos se envía el dato 0A (“Cg==” en Base 64) para que encienda un led del dispositivo y un dato 0B (“Cw==” en Base64) para que lo apague.

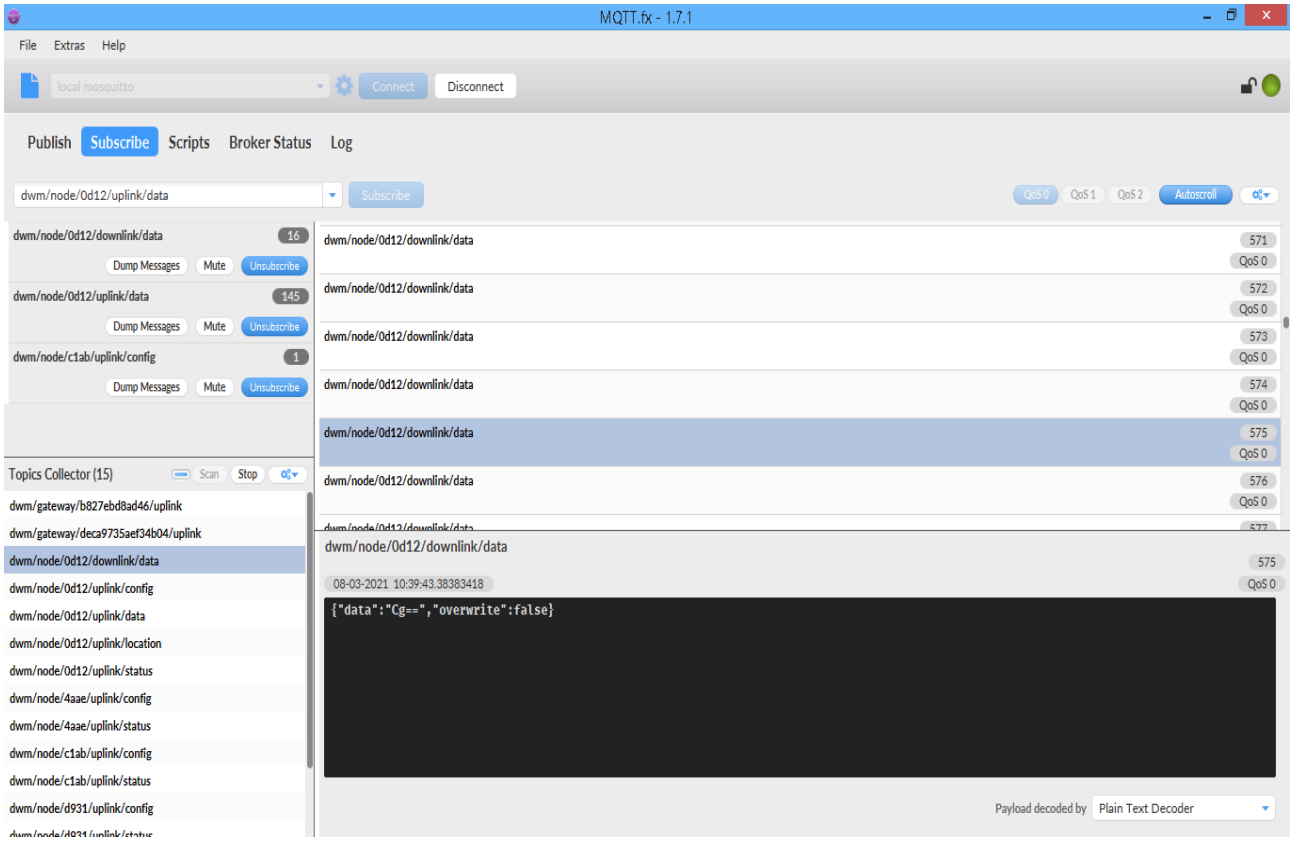


Figura 55. Mensaje dowlink 0x0A

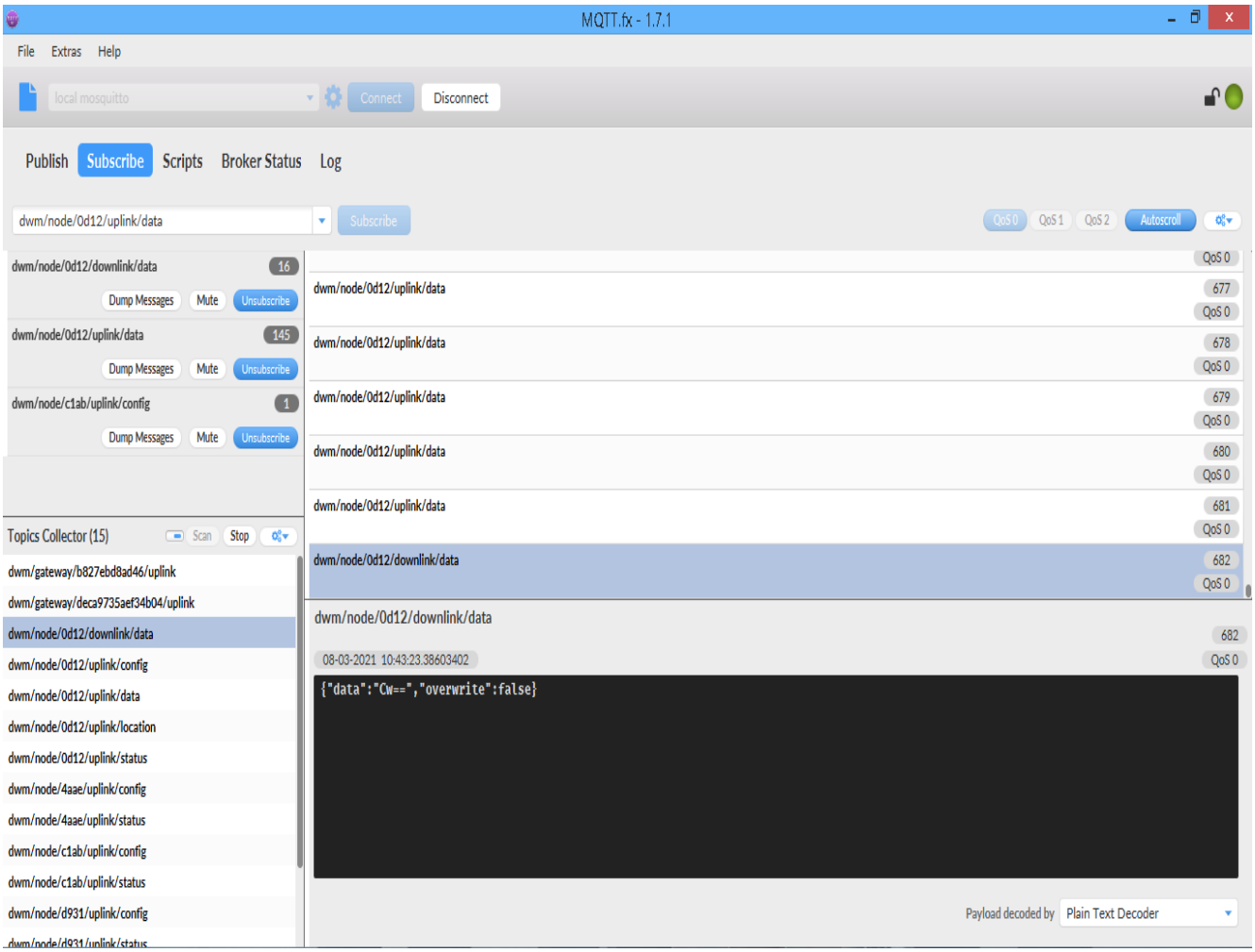


Figura 56. Mensaje dowlink 0x0B

6.3.3. App DRTLS Manager

La aplicación para Android es la manera más sencilla de configurar y ver la red de localización. El único inconveniente de este método es que para ver la localización del *tag* o para configurar los nodos es necesario conectarse a través de Bluetooth al dispositivo y por tanto, no es posible alejarse más de 10 metros del *tag* o del nodo que se desea configurar.

Para formar una red de localización se añaden dispositivos a la red configurando como mínimo 3 *anchors* y 1 *tag*, en nuestro caso se han configurado 11 dispositivos como *anchors*, 2 como *tags* y 1 oyente para almacenar la información en el ordenador. Una vez configurada la red, se verán los dispositivos como se observa en la Figura 57, donde se identifican los dispositivos por un nombre, por un identificador y por un icono, a los *anchors* con un triángulo vacío, a los *anchors iniciadores* con un triángulo relleno y a los *tags* con círculos rellenos.

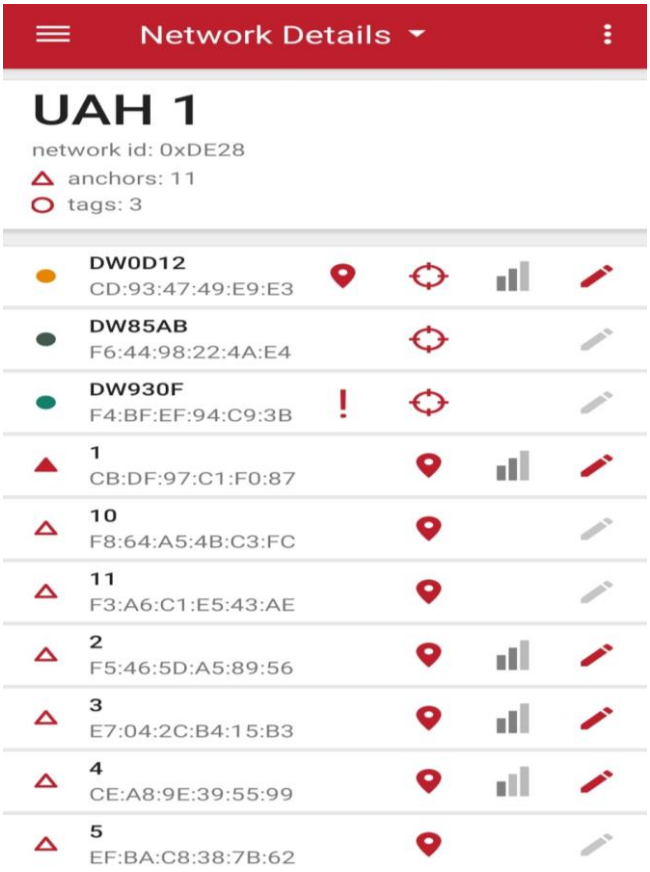


Figura 57. Dispositivos observados desde la app

Como se observa en la figura anterior, existen unos iconos de configuración dependiendo si es un *tag* o un *anchor*.

- Icono de ubicación.
- No muestra el *tag* en el mapa.
- Muestra el *tag* en el mapa.
- Muestra el *tag* en el mapa con las distancias a los *anchors* dentro del rango.
- Intensidad de la señal de *Bluetooth* al dispositivo.
- Icono para configurar el dispositivo.
- Icono que identifica el dispositivo como *tag* en el mapa. Cada *tag* muestra un color diferente.
- Icono que identifica un dispositivo como *anchor* en el mapa.
- Icono que identifica un dispositivo como *anchor* iniciador en el mapa.
- Icono de advertencia que indica que UWB está desactivado o configurado de forma pasivo como puede ser un oyente.

Para configurar un dispositivo se pulsa sobre el icono y aparece la pantalla mostrada en la Figura 58 o Figura 59 dependiendo si es un *anchor* o un *tag* respectivamente. En esta pantalla

aparecen cierta información del dispositivo como el identificador del dispositivo o la dirección *Bluetooth* del dispositivo. Luego, aparecen unos parámetros que pueden ser configurados como el nombre del dispositivo, la actualización de *firmware* automática por UWB, activar o desactivar los leds de la tarjeta, el nombre de la red, activar o desactivar o usar de forma pasiva UWB y el modo del nodo. Dependiendo si se configura el nodo como *tag* o como *anchor*, aparecerán unos parámetros a configurar distintos:

- **Modo *anchor*:** se debe de configurar si es un *anchor iniciador* o no y se pueden configurar las coordenadas en las que están situados.
- **Modo *tag*:** se configuran tanto la frecuencia de actualización normal como la estacionaria, si se requiere un modo responsivo o no y si se desea activar el motor de localización, necesario para posicionar el *tag*.

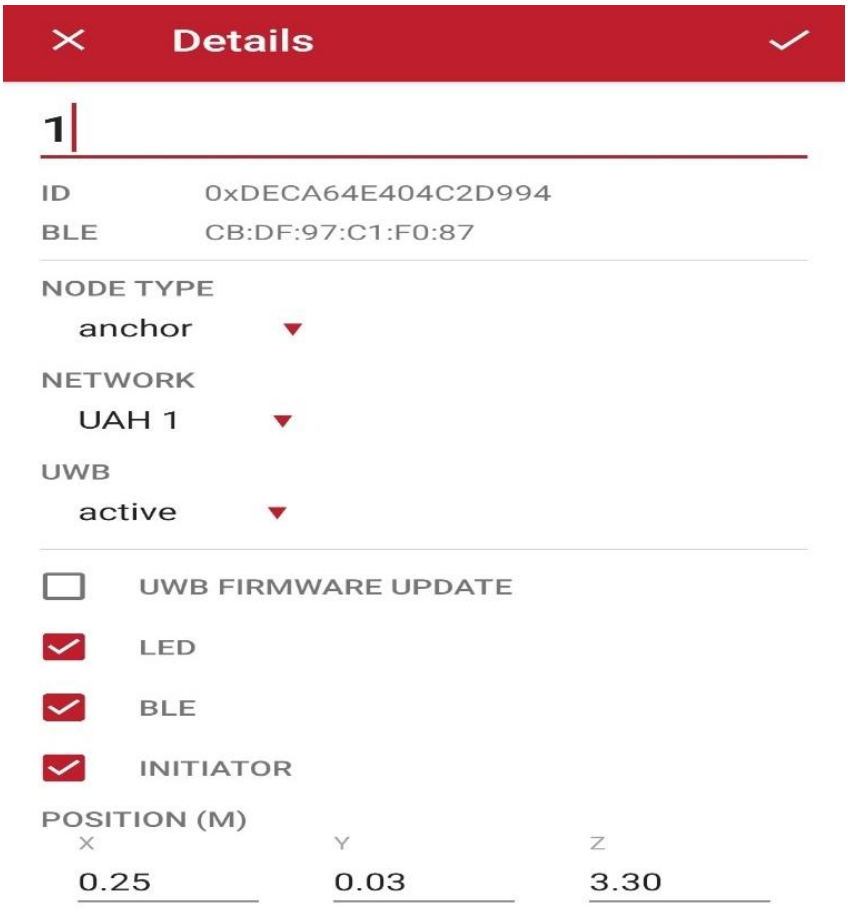


Figura 58. Configuración de anchor en app

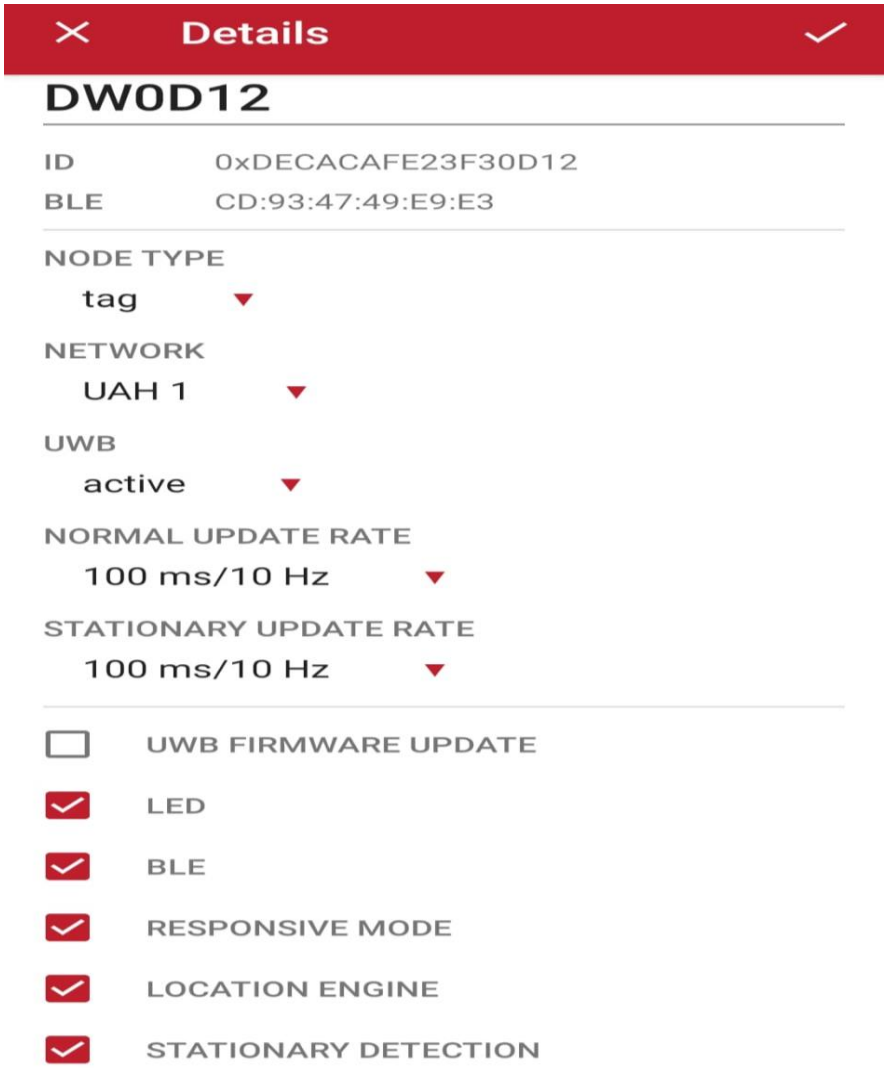


Figura 59. Configuración de tag en app

Una vez configurada la red de localización, en la pestaña *"Grid"* se introduce un mapa de la segunda planta de la Escuela Politécnica de la Universidad de Alcalá y se ajusta a las posiciones de los *anchors* situados en la sala del fondo del primer pasillo del sector Norte, sala N-21, de la EPS como se observa en la Figura 60 donde aparecen los *anchors* situados en las esquinas de la sala y el *tag* con su posición y con la distancia a cada uno de los *anchors*.

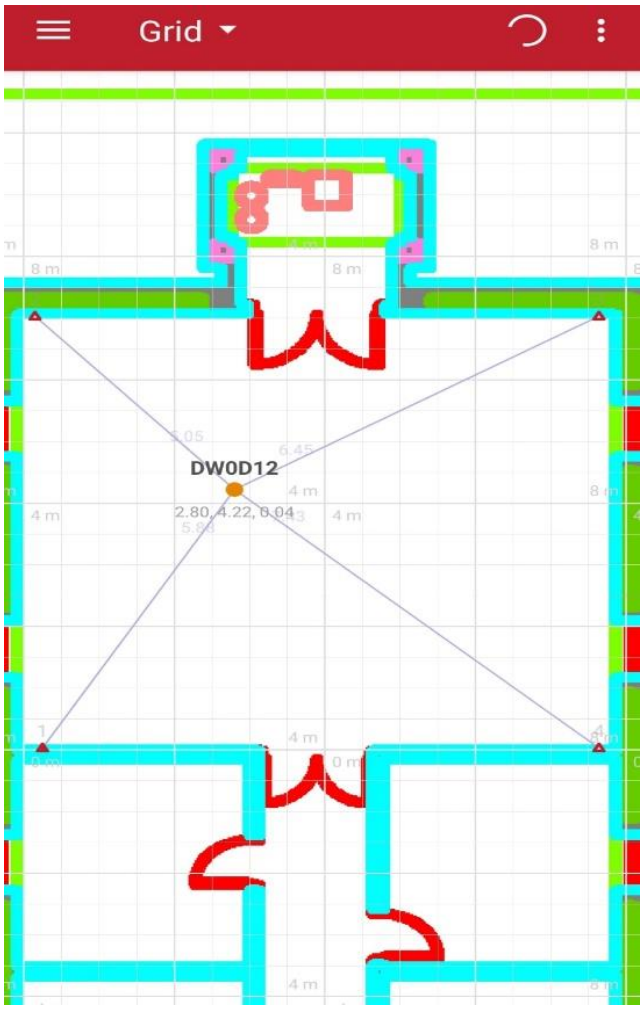


Figura 60. Mapa de la sala N21 en la APP

6.3.4. Página web DRTLS Manager Web

La web integrada en la imagen de la *Raspberry Pi* está basada en IP y por tanto, se necesita la dirección IP de la *Raspberry Pi* principal que actúa como *proxy* y que se encuentra situada en la sala del fondo del primer pasillo de la segunda planta del sector Norte (N-21) de la Escuela Politécnica Superior.

Si un ordenador se encuentra conectado en la misma red que esta *Raspberry Pi*, se puede observar y configurar la red de localización introduciendo en un navegador como *Google Chrome* o *Mozilla Firefox*, la dirección IP “192.168.72.94”. La red de localización en la web tendrá un aspecto como el mostrado en la Figura 61.

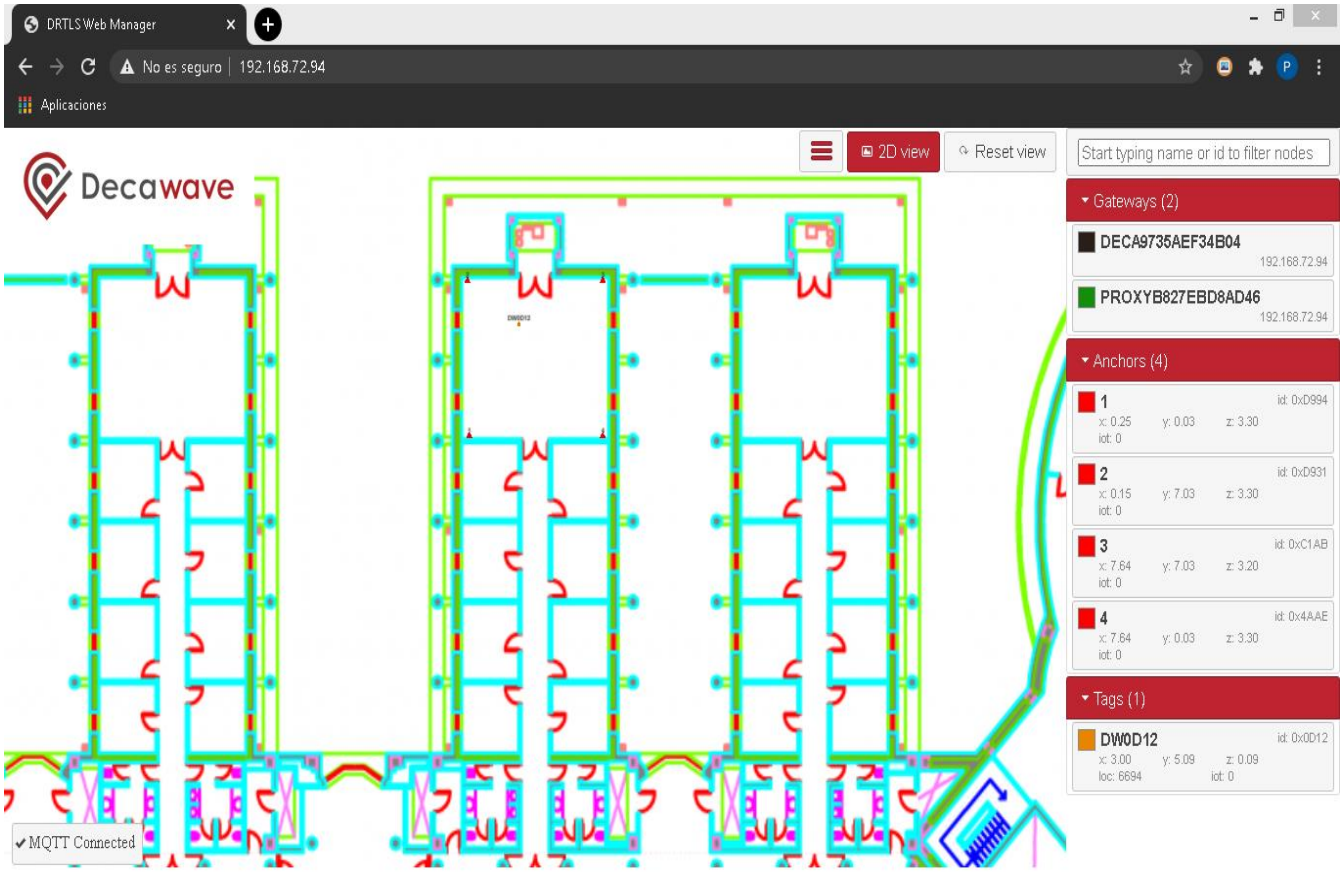


Figura 61. WEB DRTLS , mapa del primer y segundo pasillo del sector Norte 2ª planta donde se encuentra la sala N21

Para que las posiciones de las balizas coincidan en la posición real del mapa es necesario configurar unos valores dependiendo las dimensiones y donde se encuentre el origen de coordenadas en el mapa. En este caso los valores que se han configurado son mostrados en la Figura 62:

- **Aspect ratio x,y (m):** configura el tamaño real del eje x y el eje y.
- **Origin x,y (m):** configura donde se encuentra el origen. Configurando el eje x, se moverá la imagen hacia izquierda o derecha y configurando el eje y, se moverá hacia arriba o hacia abajo.

Figura 62. Configuración del mapa en la web

A la derecha de la página web se encuentran las puertas de enlace, los *anchors* y los *tag*s que se encuentran en el rango de visión del dispositivo que actúa como puerta de enlace. Además, dentro de cada dispositivo, se observa el nombre, la localización de cada uno de ellos y los mensajes IoT que se han intercambiado con ellos.

Si se requiere configurar un dispositivo se hace *click* sobre el dispositivo deseado y se abrirán las ventanas mostradas en las figuras 63 y 64, dependiendo si es un *anchor* o un *tag* respectivamente.

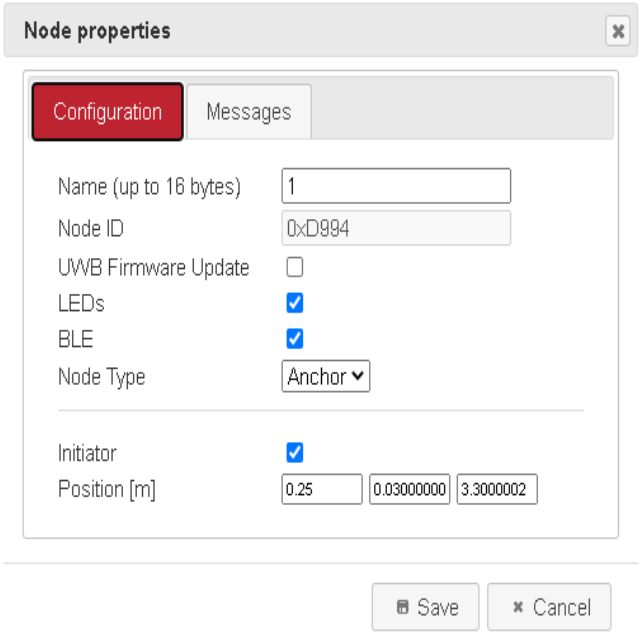


Figura 64. Configuración de anchor en web

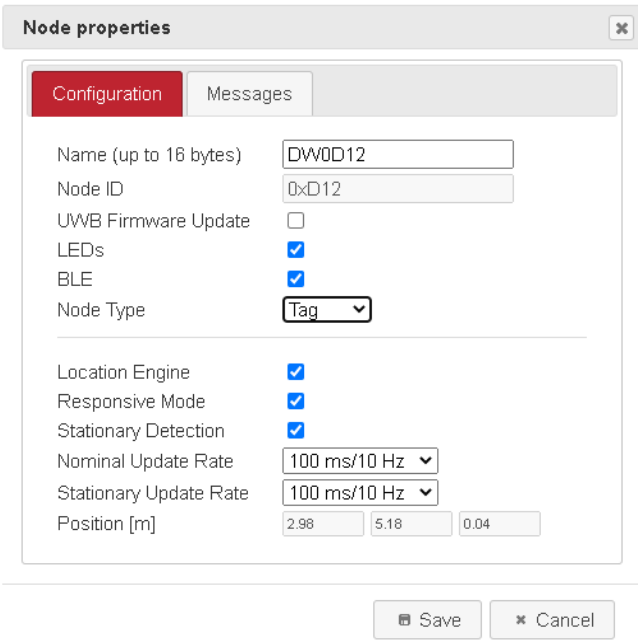


Figura 63. Configuración de tag en web

Además de la configuración de los dispositivos, en la ventana de las propiedades del dispositivo existe una pestaña donde se observan los mensajes recibidos desde el dispositivo y se pueden enviar mensajes hacia el dispositivo. En este caso los datos deben ser en hexadecimal y tienen un máximo de 34 bytes por mensaje.

En la Figura 65 se observa un ejemplo de mensajes recibidos y enviados. En este caso se ha recibido el mensaje en hexadecimal “4159554441”, que si se decodifica a texto significa “AYUDA”, esto quiere decir que el usuario ha pulsado el botón “user” del dispositivo y este ha mandado el mensaje de ayuda. Por otro lado, se envía el mensaje en hexadecimal “0b” que haría que se apagara el led D12 del dispositivo.

En la Figura 66 se recibe el mensaje en hexadecimal “504f5349424c45204341494441” que decodificado significa “POSIBLE CAIDA”, esto ocurre cuando el *tag* mediante el acelerómetro detecta una posible caída y manda un mensaje a la web.

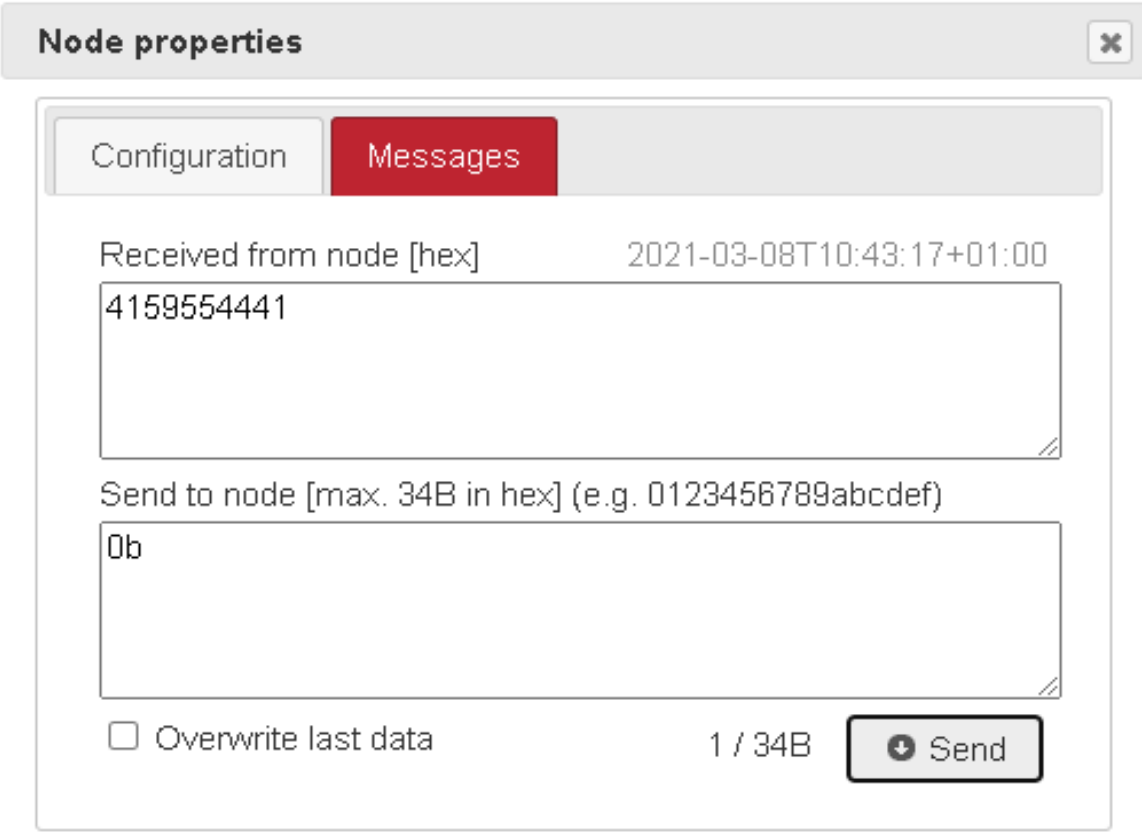


Figura 65. Mensajes recibidos y enviados en el tag en la web

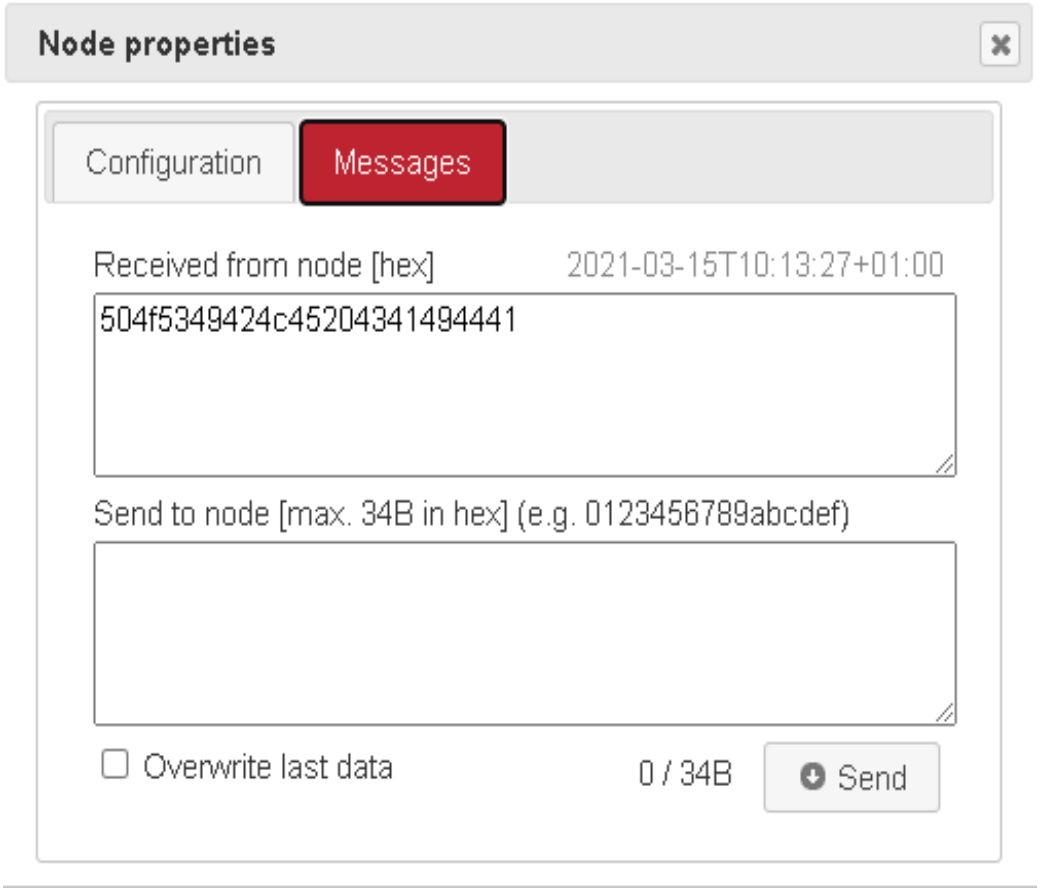


Figura 66. Mensaje de POSIBLE CAIDA recibido en tag en la web

6.4. Pruebas y resultados

Para probar el correcto funcionamiento del sistema de localización y comprobar el error cometido por el sistema se ha balizado la sala del fondo del primer pasillo de la segunda planta del sector Norte (sala N21) y el pasillo de la segunda planta del sector Oeste de la Escuela Politécnica Superior. Para ello se han colocado cuatro *anchors* en la sala N21, cada uno de ellos en una esquina de la sala, y 7 *anchors* distribuidos por columnas y paredes de forma que se cubra toda la zona del pasillo del Oeste vista en la Figura 67. Cada uno de estos *anchors* es configurado con una ubicación en metros a partir de la coordenada (0,0) que es la que se tendrá de referencia y que se encuentra situada en la esquina de la sala donde se coloca el *anchor* configurado como iniciador.

Figura 67. Planta segunda Oeste

Para una mejor obtención de los datos es necesario situar los *anchors* y la puerta de enlace con la *Raspberry Pi* lo más alto posible para que de esta forma cubra una mayor área y se obtenga una mayor señal con el resto de los dispositivos. En la Figura 68 se puede ver la posición de la *Raspberry Pi* que hace de puerta de enlace en el pasillo del Oeste, situada en la zona alta de una de las columnas.

Figura 68. Posición de la puerta de enlace en Oeste

Los *anchors* se encuentran situados en la siguiente posición:

Tabla 2. Posición de los anchors

Anchor	Situación	x (m)	y (m)	z (m)
1 iniciador	Sala N21	0.25	0.03	3.30
2	Sala N21	0.15	7.03	3.30
3	Sala N21	7.64	7.03	3.20
4	Sala N21	7.64	0.03	3.30
5	Oeste	-17.70	-19.00	2.20
6	Oeste	-22.00	-34.80	2.28
7	Oeste	-29.50	-42.70	2.33
8 iniciador	Oeste	-44.00	-48.80	2.26
9	Oeste	-41.61	-24.30	2.29
10	Oeste	-26.60	-27.70	2.22
11	Oeste	-32.50	-33.00	2.28

Las dimensiones de la sala N21 son de 7.856 x 7.1 metros. Para poner en movimiento el *tag* por la sala se ha hecho uso de un “robot sigue líneas” llamado “mbot” de la marca *Makeblock*, visto en la Figura 69 y que seguirá la cinta negra con la que se han creado dos recorridos en el suelo de la sala vistos en la Figura 70. El recorrido exterior tiene unas dimensiones de 2.802 x 3.61 metros. El *tag* se coloca encima del robot en posición vertical ya que los resultados serán mejores que si se coloca en posición horizontal debido a que la antena está polarizada verticalmente.

Figura 69. Robot con tag utilizado para hacer mediciones

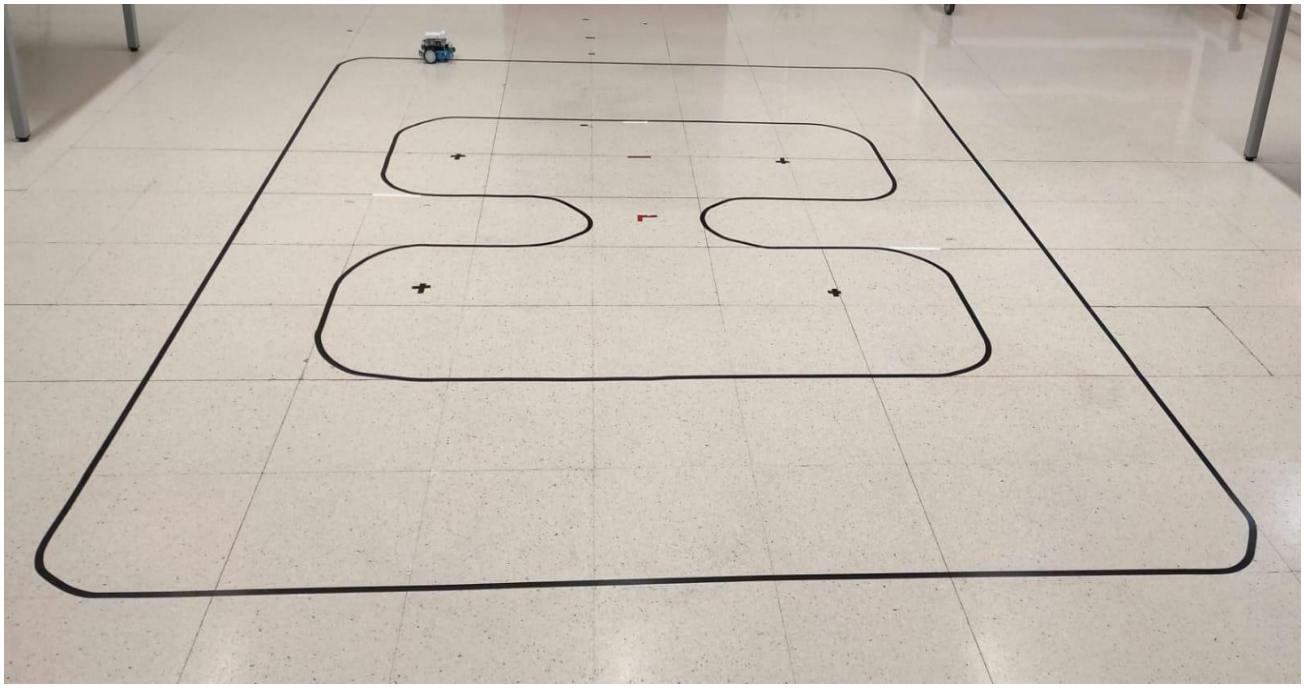


Figura 70. Recorridos para realizar mediciones

Además de estos recorridos y la posición de los *anchors* en la sala N21 vistos en la Figura 71, se han realizado varias pruebas por el sector Oeste de la segunda planta de la Escuela para comprobar que el *tag* se posiciona en todos los lugares del sector. Estas pruebas se han realizado con el *tag* obteniendo su posicionamiento a 5 Hz y portado por una persona caminando por todo el sector Oeste de la segunda planta. Los resultados son los mostrados en la Figura 72 donde además de los recorridos realizados se muestra la posición de los *anchors*.

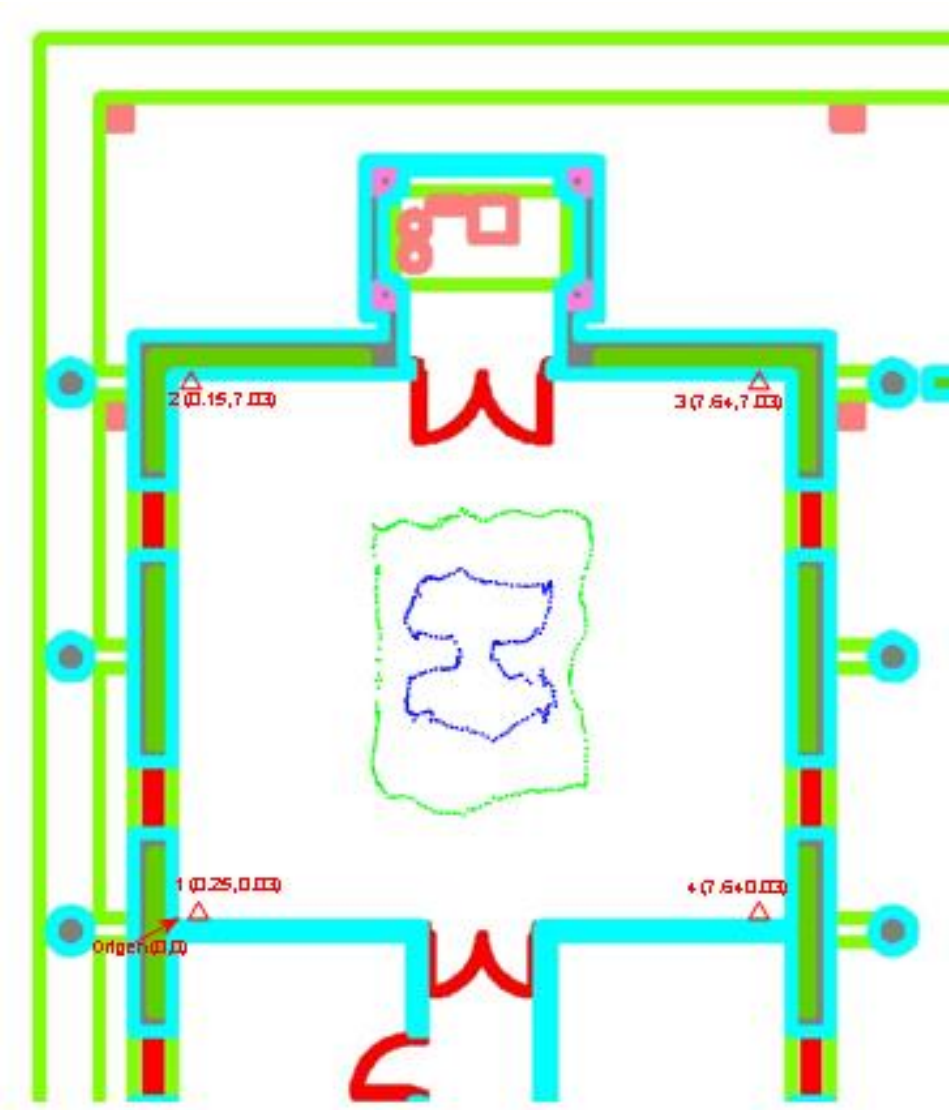


Figura 71. Anchors y recorridos medidos en sala N21

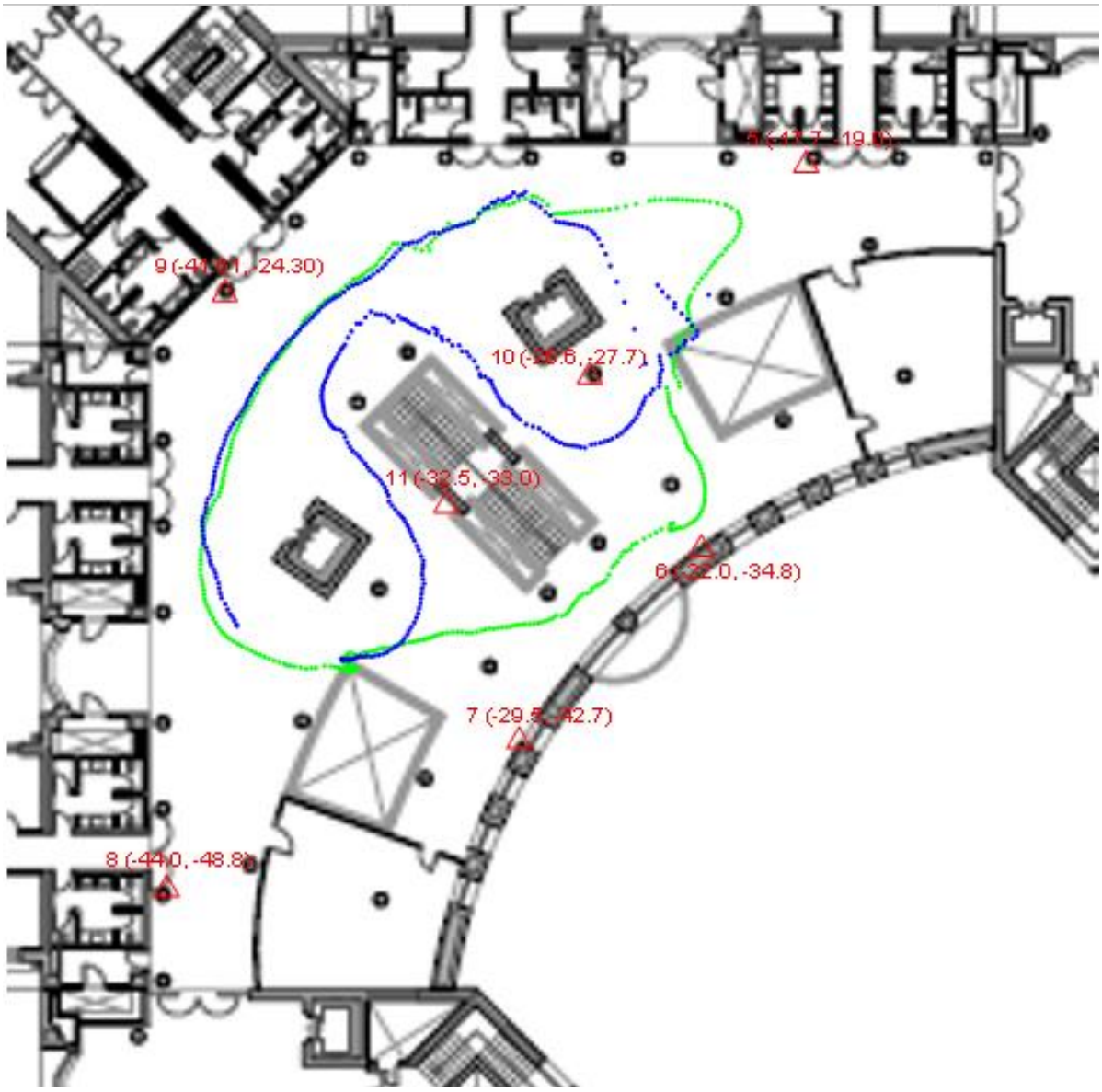


Figura 72. Anchors y recorridos en Oeste

Para la obtención de los datos de posicionamiento se hace uso de otro dispositivo configurado como *listener* u oyente, el cual recibirá los datos de localización del *tag* y los transmitirá al ordenador a través de UART haciendo uso del comando lep para que se obtengan los datos separados por comas (ver Figura 74). Estos datos se guardan en un archivo de texto haciendo un *log* para más tarde representarlos en *MatLab*. El sistema de localización es como el mostrado en la Figura 73.

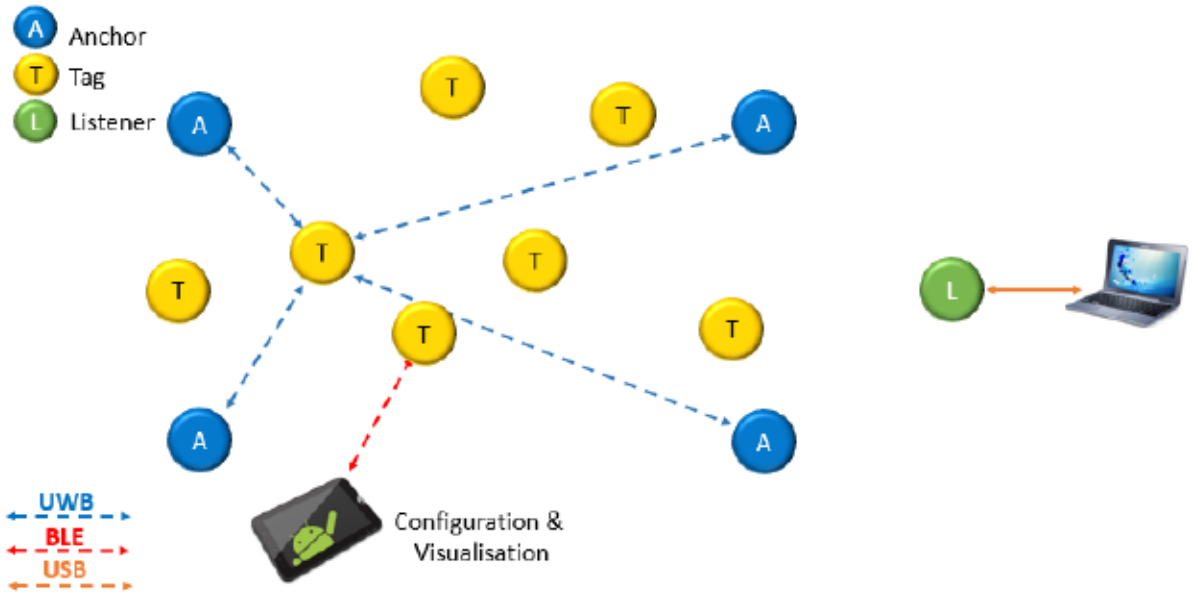


Figura 73. Red DRTLS para realizar mediciones [14]

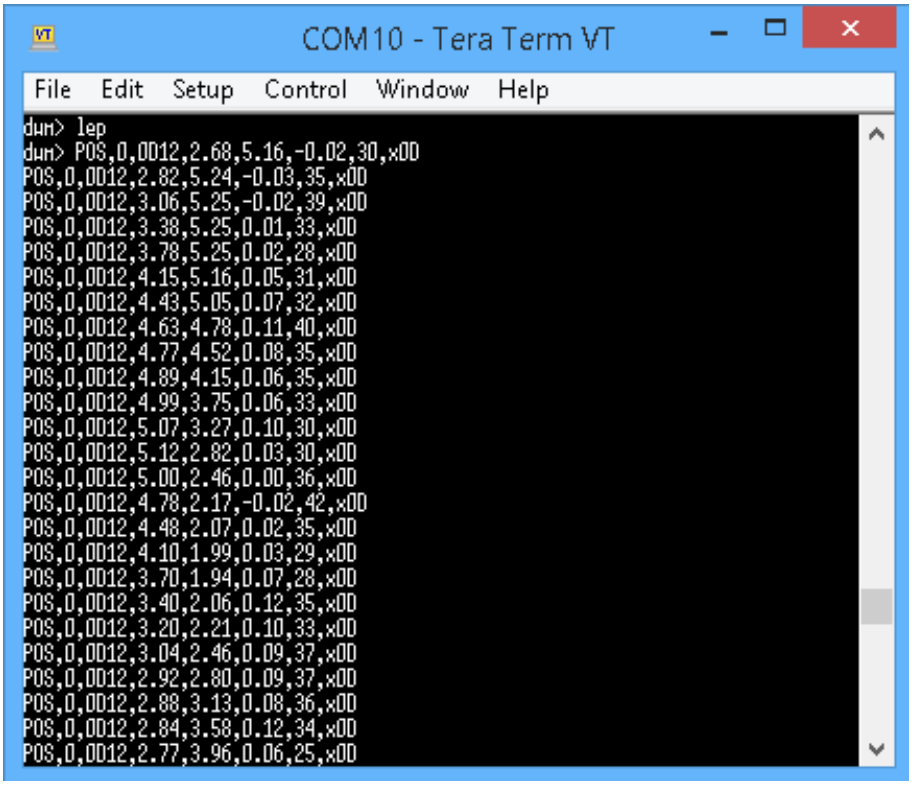


Figura 74. Datos recibidos por UART

El robot con el *tag* recorre primeramente el recorrido externo mientras obtiene los datos con una frecuencia de 5 Hz. Una vez ha realizado el recorrido externo, se coloca el robot en el recorrido interno y almacena sus datos.

Además de los recorridos realizados, se coloca el *tag* quieto en cada una de las esquinas y en el centro del recorrido exterior y poder comprobar cómo cambian las medidas cuando el *tag* se encuentra en movimiento y cuando se encuentra parado.

Una vez se han obtenido las localizaciones de los dos recorridos y de los demás puntos, se realiza un script en Matlab donde se leen los datos del fichero de texto, se calcula el error con las posiciones de los *ground truth* y las posiciones del centro y las esquinas reales, se dibujan los dos recorridos y las esquinas tanto los reales como los medidos y se representa el error mediante una función de distribución acumulativa (cdf).

- **Recorrido externo:** se realiza una vuelta con el robot y se obtienen los datos de la Figura 75. Como se observa en la imagen, los datos obtenidos por el *tag* son bastante buenos ya que sigue en su mayoría al *ground truth*. Se ven algunas desviaciones mayores en alguna zona que pueden ser causadas por la orientación del *tag* o por algún obstáculo que en esos puntos influyan en la línea de visión como mesas o estanterías de la sala.

  También se ha comprobado que si una persona camina cerca del *tag* los resultados son peores ya que afecta a la línea de visión con alguno de los *anchors*.

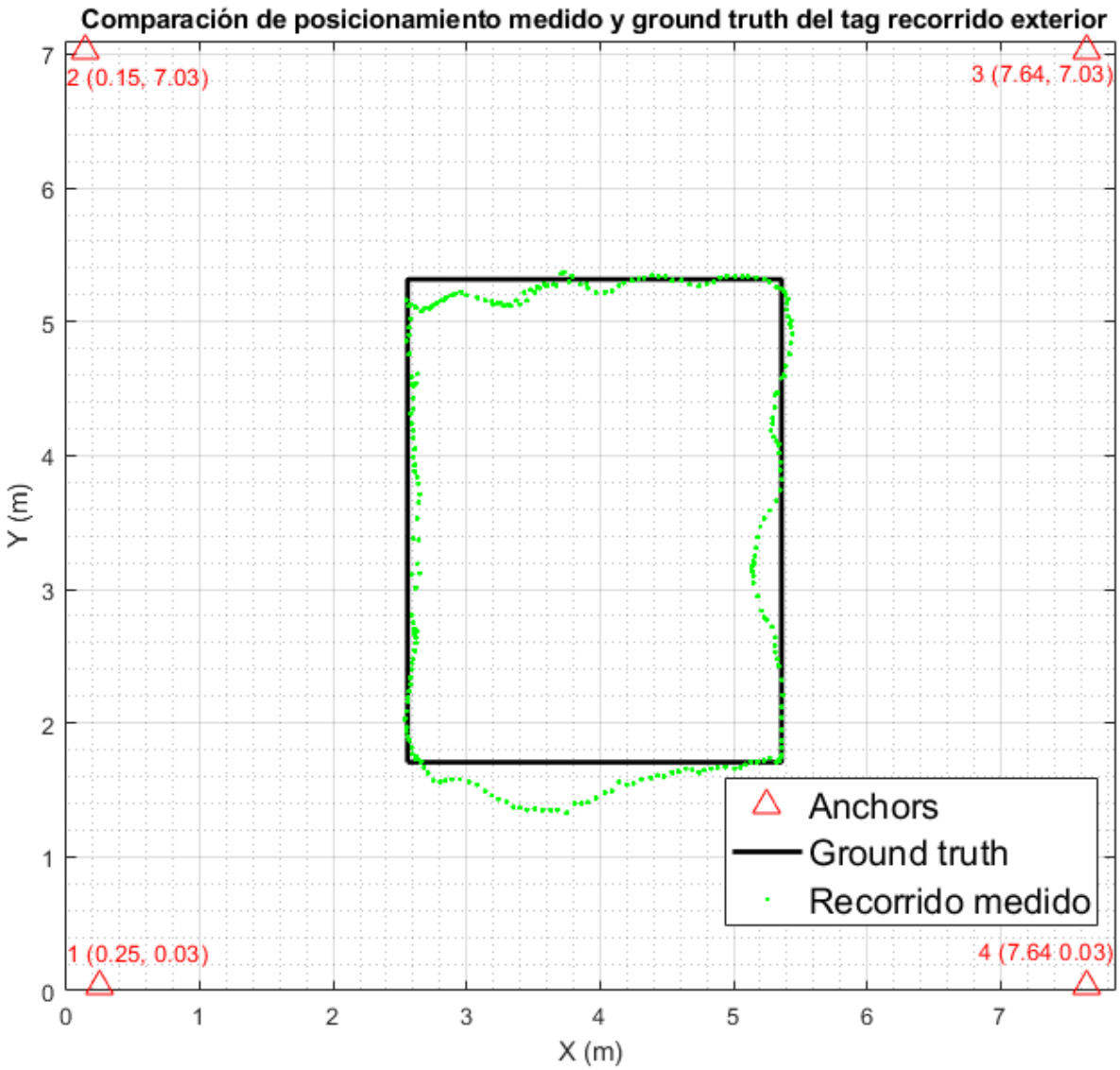


Figura 75. Medidas otenidas por el tag en el recorrido exterior

- **Recorrido interior:** el recorrido interior es más complicado de realizar ya que tiene más curvas. El *ground truth* mostrado en la Figura 76 no es el recorrido exacto ya que las esquinas deberían ser curvas, pero vale para hacernos una idea de los errores que se está cometiendo al realizar las medidas.

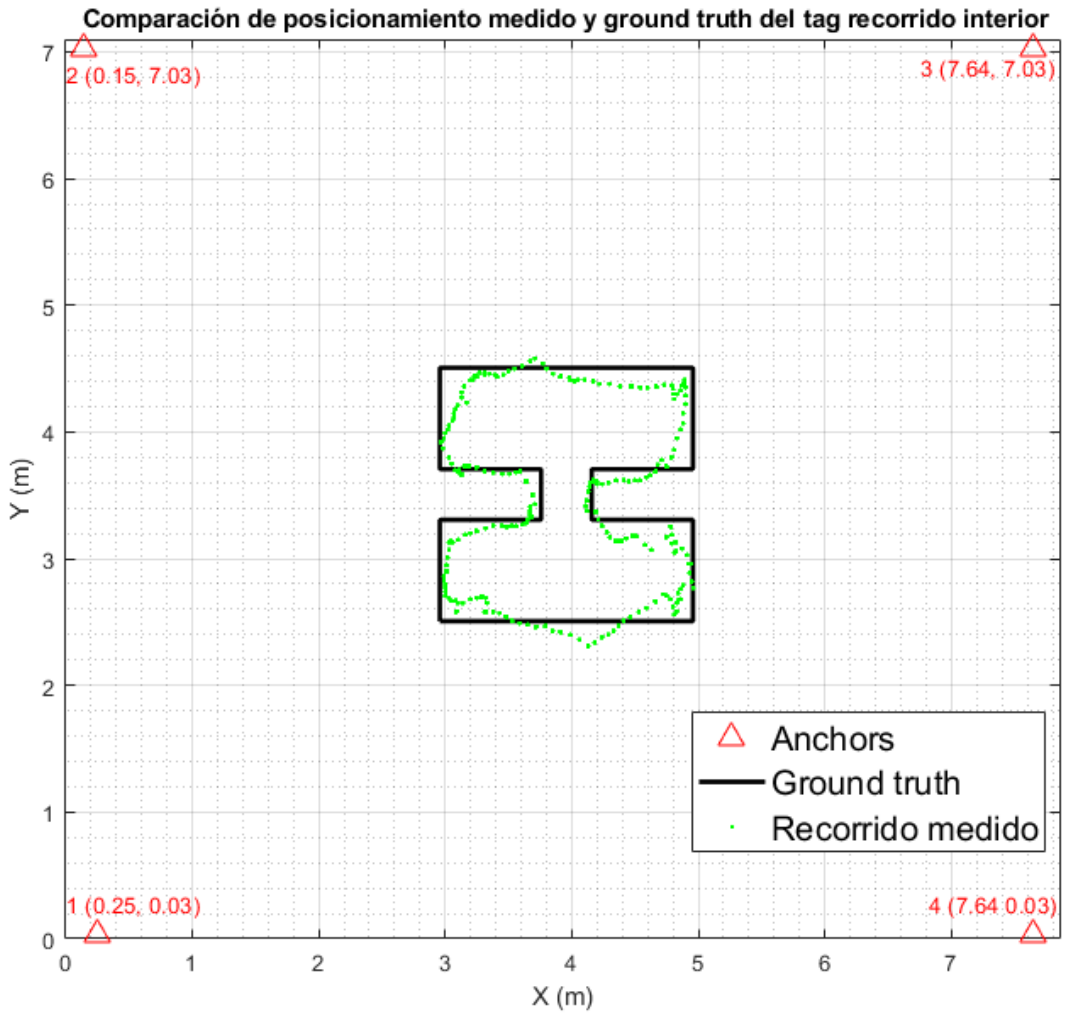


Figura 76. Medidas obtenidas por el tag en el recorrido interior

- ***Tag* posado sobre esquinas y centro:** una vez probado el funcionamiento del *tag* en movimiento, se comprueba el funcionamiento posando el *tag* en las cuatro esquinas del recorrido exterior y en el centro de este. Las coordenadas de las esquinas y el centro son las siguientes:
    - Esquina 1: (2.558, 1.707) m
    - Esquina 2: (2.558, 5.317) m
    - Esquina 3: (5.36, 5.317) m
    - Esquina 4: (5.36, 1.707) m
    - Centro: (3.959, 3.512) m

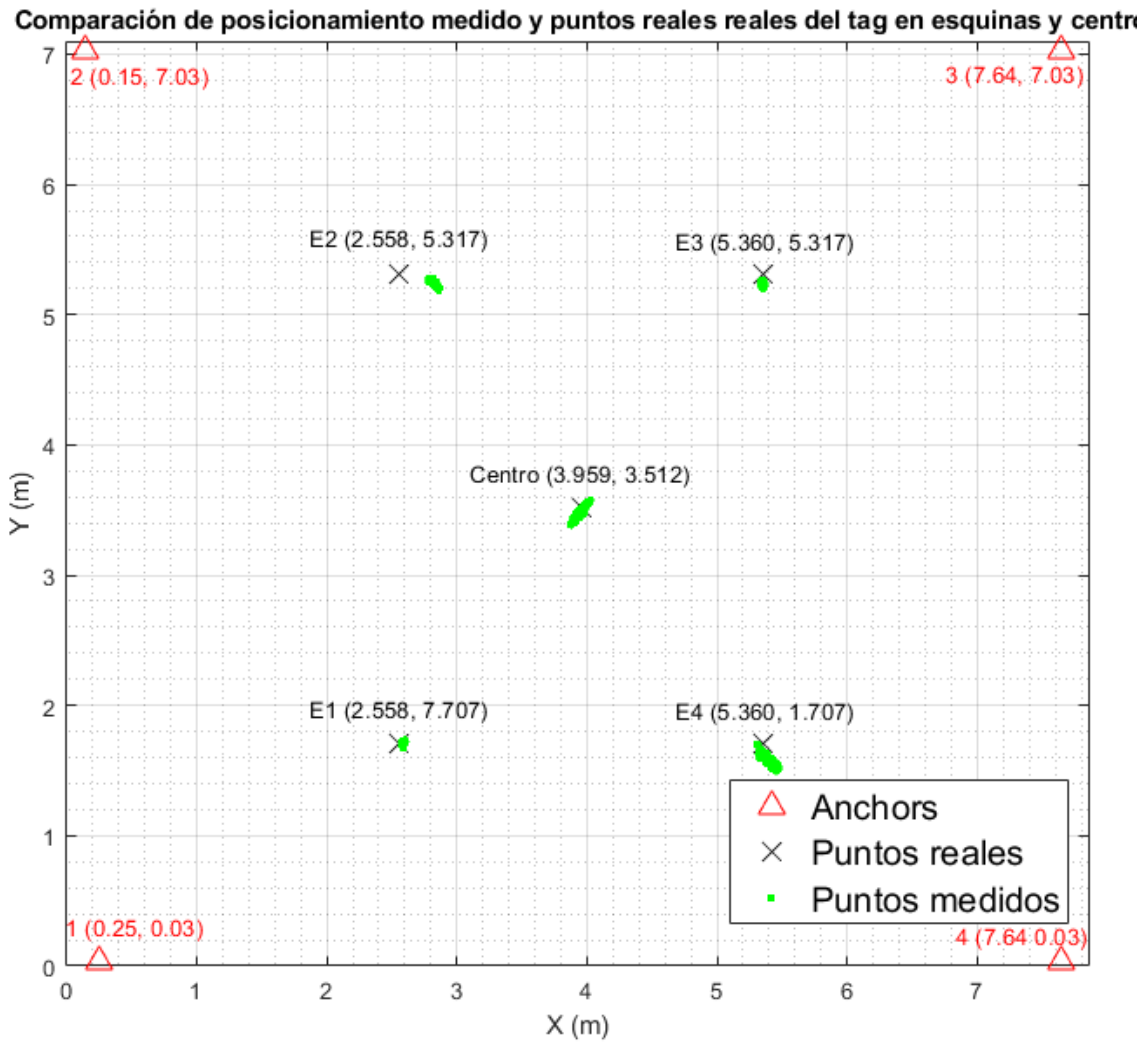


Figura 77. Mediciones en esquinas y punto central del recorrido exterior

- **Errores CDF:** por último, se realiza una función de distribución acumulativa para observar los errores de las medidas tomadas. Esta gráfica muestra el error cometido en metros en función del porcentaje de ese error, por ejemplo, en la curva del recorrido exterior, la curva de color azul celeste, se observa cómo se tiene un error de 0 en el 20% del recorrido, para el 75 % del recorrido se obtiene un error acumulado de unos 10 cm y para el 100 % de unos 35 cm. Por tanto, como se puede comprobar los resultados son buenos siendo la mayor parte de los errores menores de 20 cm.

  Con el *tag* parado, posado en las esquinas y en el centro del recorrido, los resultados son variados pudiendo ser debido a la orientación del *tag* o a la existencia de algún objeto cercano que pueda obstaculizar. La mayor parte de los resultados son buenos bajando el error de 20 cm excepto en la esquina 2, curva amarilla, donde se obtiene un error mayor debido a la orientación y a la presencia de una mesa cercana que puede influir en la línea de visión con el *anchor* número 2.

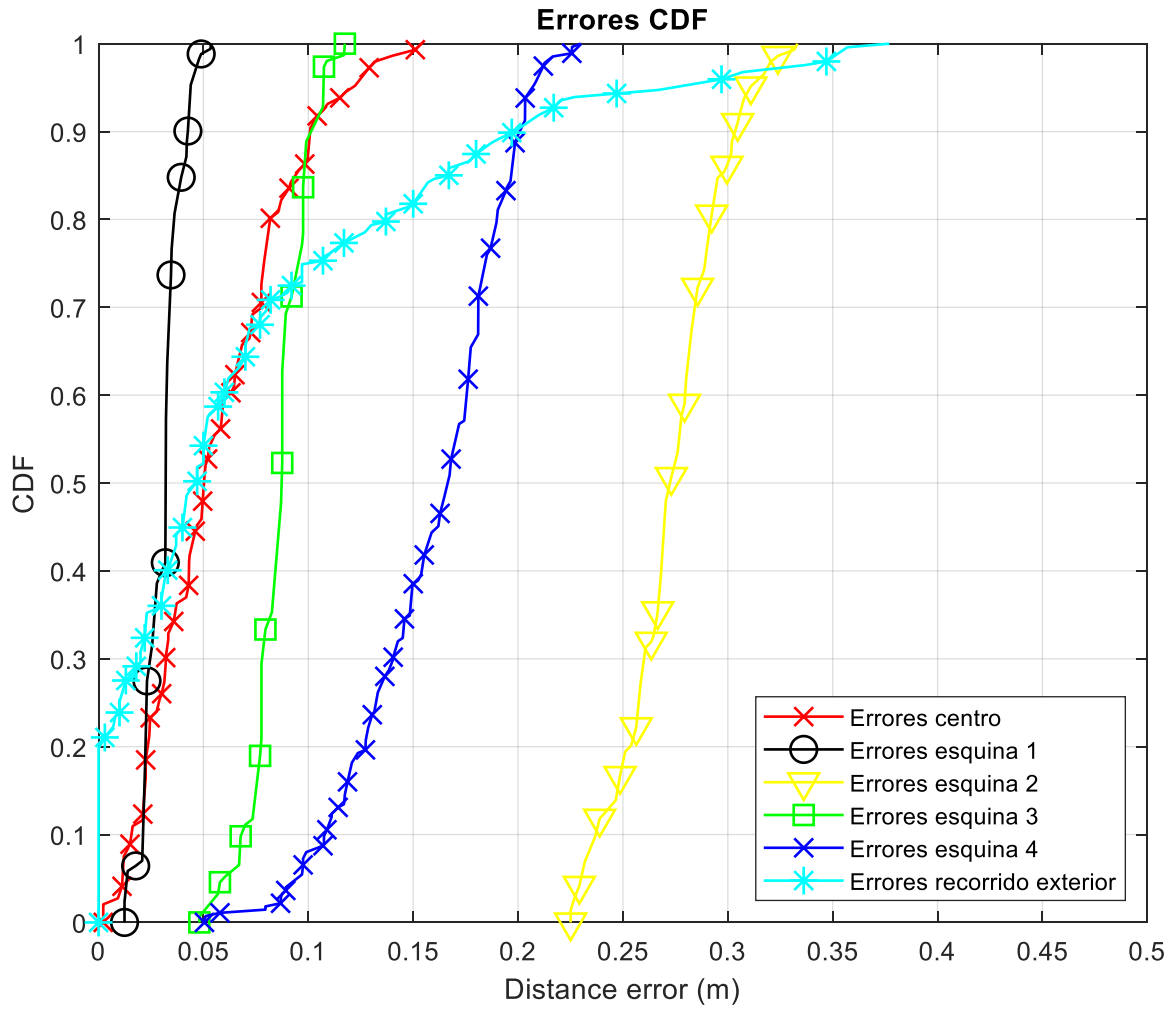


Figura 78. Errores CDF

6.5. MEJORAS Y APLICACIÓN

Las tarjetas de desarrollo DWM1001-DEV pueden ser utilizadas para muchas aplicaciones, un ejemplo de ello que puede ser útil en la situación actual que nos encontramos es su utilización para el distanciamiento social. Para ello es necesario medir las distancias entre los *tags* constantemente y cuando estos se encuentren por debajo de un umbral indicado, mandar un aviso. [18]

Otra posible aplicación de los módulos DWM1001 es para seguridad en distintos trabajos de riesgo como pueden ser las minas subterráneas, donde otros sistemas de posicionamiento como GPS son inviables. En este caso se podría balizar la mina y acoplar un *tag* a cada trabajador y así ante cualquier percance que pueda suceder, se pueda localizar la posición del trabajador en todo momento. [19]

Además de estas aplicaciones, existen una infinidad de situaciones en las que podría ser útil estos dispositivos. Una de ellas puede ser en residencias de mayores o centros donde sea necesario tener localizados a sus internos para una mayor seguridad, esto también se podría aplicar al ejemplo anterior del posicionamiento en una mina. Con este enfoque se han desarrollado dos funcionalidades para los dispositivos:

La primera es el aviso del usuario pulsando un botón del *tag*. Aprovechando que el dispositivo lleva integrado dos botones, se aprovechará uno de ellos, el llamado "user". El software estará continuamente chequeando si se activa ese botón que en este caso se activará con nivel bajo cuando es pulsado. Si el *software* detecta que el usuario está pulsando el botón, un led del dispositivo se encenderá y un aviso con el mensaje "AYUDA" será enviado a la página web y al protocolo MQTT donde se podrá observar e ir inmediatamente a atenderlo.

Otra funcionalidad que puede ser de gran ayuda, es implementando un detector de caídas aprovechando el acelerómetro integrado en el módulo DWM1001. Este es un acelerómetro de bajo consumo y que transmite los datos a través de SPI o I2C. Para poder acceder a los datos del acelerómetro es necesario desactivar la detección estacionaria del *tag*. La dirección necesaria para configurar y acceder al acelerómetro es 0x19, esta es la dirección a la que se accederá cuando se quiera leer o escribir a través de I2C. Para utilizar el acelerómetro hay que configurarlo escribiendo en los registros de configuración pertinentes:

- Se habilita el acelerómetro en los ejes x, y, z y se configuran con una frecuencia de 100 Hz.
- Se activa la actualización de datos de bloque, es decir, los registros de salida no se actualizan hasta que no son leídos.
- Se activa el modo de operación de alta resolución, por tanto, los datos de salida serán de 12 bits justificados a la izquierda.
- Se configura una escala de +/-2g

Una vez configurado el acelerómetro, se leen los datos del eje x, y, z. Para cada uno de los ejes se deben de leer primero los datos menos significativos y luego los más significativos. Después se unirán estas lecturas y se desplaza el resultado cuatro bits a la derecha, ya que los datos de salida son datos de 12 bits y justificados a la izquierda.

Por último, se calcula una aceleración normalizada de los tres ejes. Con el *tag* en reposo, normalmente esta aceleración será de uno ya que uno de los ejes es uno dependiendo de la orientación del *tag*. Cuando alguno de los ejes detecta un movimiento brusco, es decir una variación de la aceleración brusca, esta aceleración bajará su valor y si es menor a 0.5 durante varios segundos se considerará que es posible que esta variación haya sido provocada por una caída del dispositivo, si este dispositivo lo lleva una persona seria posible detectar la caída de ella. Cuando se detecta esta posible caída, el *software* manda de inmediato un mensaje de aviso de "POSIBLE CAIDA" hacia la página web y el protocolo MQTT.

7. CONCLUSIONES Y PROPUESTAS DE MEJORA

Después de estudiar, realizar el montaje de un sistema de localización en tiempo real para interiores y realizar varias pruebas con la tarjeta de desarrollo DWM1001-DEV se ha llegado a las siguientes conclusiones:

- Se puede realizar una red de localización en tiempo real con un bajo coste y unos resultados con errores normalmente menores de 20 cm.
- La tecnología de *Ultra Wide Band* para el posicionamiento en interiores irá creciendo en un futuro ya que es una tecnología que resulta más barata comparada con otros métodos como pueden ser ultrasonidos e infrarrojos, aunque estos puedan tener mejores resultados, dependerá de la aplicación para la que se requiera la localización.
- Los resultados de las pruebas realizadas en la sala N21 han sido muy buenos. El error cometido es bajo, aunque aumenta considerablemente cuando existe algún objeto o impedimento cerca del *tag* que hace que la visión con el anchor no sea la ideal y por tanto el cálculo de la localización será peor.
- Los resultados de las pruebas en el Oeste han sido peores. Como es un área más grande se necesitan un número de *anchors* mayor. Según la documentación, en una situación ideal sin obstáculos de por medio, estos dispositivos tienen un alcance entre ellos de entre 30 metros y 40 metros. Por ello en un principio se utilizan solamente 4 *anchors* que deberían ser suficientes para cubrir la zona deseada. Pero al realizar las medidas, se observa que esto no es así ya que el *tag* no visualiza los suficientes *anchors* para realizar las medidas, el alcance de los dispositivos es bastante menor de unos 20 metros si no existen obstáculos de por medio. Si existe algún obstáculo de por medio como columnas, paredes, ascensor, etc, el alcance se reduce aún más siendo de unos 10 metros pudiendo ser menor dependiendo del obstáculo a evitar. Por todo ello, se aumentan los *anchors* hasta conseguir que el *tag* calcule su posición en todas las partes de la planta del del sector Oeste.
- Los mensajes enviados por el dispositivo se muestran en la página web como código ASCII en hexadecimal y codificado en Base64 en el MQTT Broker. Por tanto, sería necesario decodificarlo a texto para tener un mejor funcionamiento del sistema.
- El acelerómetro incorporado nos puede dar datos muy útiles para distintas aplicaciones y tratar de mejorar los dispositivos.

Por todo lo anterior, se llega a la conclusión que para posicionar áreas de pequeño tamaño esta tecnología es la ideal ya que son necesarios pocos *anchors* y solamente una puerta de enlace y, por tanto, el coste sería menor que para otras tecnologías. En cambio, al aumentar el área a posicionar, habría que aumentar los *anchors* y poner más puertas de enlaces lo que conlleva a un mayor costo y, por tanto, habría que ver si merece la pena económicamente la aplicación de este módulo frente a otros o frente a otras tecnologías para cubrir áreas mayores.

Para mejorar los dispositivos en un futuro habría que aumentar la distancia de alcance. Para ello, sería necesario cambiar el canal en el que trabaja y bajar la velocidad de datos que está fijada a

6.8 Mbps. Esto nos llevaría a un alcance entre dispositivos mayor, aunque se sacrificaría la velocidad en el cálculo de la posición. Además, habría que estudiar la posibilidad de cambiar la antena utilizada para tener una mayor potencia.

8. PRESUPUESTO

En este apartado se realiza un desglose de los cotes del proyecto tanto de *hardware*, como de *software* y como de mano de obra.

8.1. Coste de Software

El desarrollo del *software* para este proyecto se ha realizado con los siguientes programas:

- **SEGGER Embedded Studio**: tiene licencia gratuita.
- **TeraTerm:** tiene licencia gratuita.
- **MatLab:** la Universidad de Alcalá cuenta con una licencia universitaria y por tanto no se atribuye ningún coste.
- **Segger J-Flash Lite** para flashear la tarjeta DWM1001-DEV: cuenta con licencia gratuita.
- **Etcher** para escribir la imagen en la Raspberry Pi: tiene licencia gratuita.
- **Microsoft Office:** al igual que *MatLab* la Universidad cuenta con licencia universitaria y no se le atribuye ningún gasto.

Por tanto, el coste de *software* total es de cero euros ya que la mayoría de los *software* utilizados tienen licencias gratuitas o suministradas por la Universidad de Alcalá.

8.2. Coste de Hardware

Tabla 3. Coste del material

Descripción	Precio (€/ud)	Cantidad	Coste total
Ordenador Portátil HP 15 TouchSmart Notebook i3	503,86 €	1	Duración = 72 meses Uso = 4 meses 28 €
Kit de desarrollo MDEK1001	167,13 €	2	334,26 €
Raspberry Pi 3 B	28,88 €	2	57,76 €
Fuente de alimentación Raspberry Pi	6,57 €	2	13,14 €
Cables USB conexión de módulos	6,41 €	11	70,51 €
Adaptador USB-Corriente	1,37 €	11	15,07 €
Tarjeta micro-SD	6,46 €	2	12,92 €
Cabezal GPIO 2X13	7,26 €	2	14,52 €

Cable Ethernet	2,95 €	3	8,85 €
Batería de litio recargable RCR123A	8,49 €	8	67,92 €
Robot sigue-líneas mBot	95 €	1	95 €
Coste Total del Material			717,95 €

8.3. Coste de mano de obra

Tabla 4. Coste de mano de obra

Concepto	Categoría	€/hora	Horas totales	Coste total
Pablo Amores Cuenca	Ingeniero de investigación	50	120	6000 €
Coste Total de mano de obra				6000 €

8.4. Coste de ejecución material

Tabla 5. Coste de ejecución material

Concepto	Coste Total
Coste de material	717,95 €
Coste de mano de obra	6000 €
Total	6717,95 €

8.5. Presupuesto de ejecución por contrata

Tabla 6. Presupuesto de ejecución por contrata

Concepto	Coste Total
Ejecución material	6717,95 €
Gastos generales (17%)	1142,05 €
Beneficio industrial (6 %)	403,08 €
Total	8263,08 €

8.6. Presupuesto Total

Tabla 7. Presupuesto Total

Concepto	Coste Total
Ejecución por contrata	8263,08 €
I.V.A. (21 %)	1735,25 €
Presupuesto Total	9998,33 €

9. BIBLIOGRAFÍA

[1] Ultra Wide Band (UWB).; https://www.ramonmillan.com/documentos/uwb.pdf

[2] DWM1001 Product Brief.; https://www.decawave.com/dwm1001/productbrief/

[3] Two Way Ranging (TWR) o Rango Bidireccional.; https://www.sewio.net/uwb-technology/two-way-ranging/

[4] DWM1001-DEV Product Brief.; https://www.decawave.com/dwm1001dev/productbrief/

[5] DWM1001-DEV Datasheet https://www.decawave.com/dwm1001dev/datasheet/

[6] DWM1001 Datasheet.; https://www.decawave.com/dwm1001/datasheet/

[7] DWM1001 System Overview and Performance.; https://www.decawave.com/dwm1001/systemoverview/

[8] Master Thesis: Heading Estimation Of A Mobile Robot Using Multiple Uwb Position Sensors By Marc Krumbein.; https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=case1555001007552678&disposition=inline

[9] Trilateración.; https://es.wikipedia.org/wiki/Trilateraci%C3%B3n

[10] DWM1001 Firmware User Guide.; https://www.decawave.com/dwm1001/firmware/

[11] DWM1001 Firmware API Guide.; https://www.decawave.com/dwm1001/api/

[12] DWM1001 Gateway Quick Deployment Guide.; https://www.decawave.com/dwm1001/gatewayguide/

[13] Protocolo IoT por MQTT.; https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[14] MDEK1001 User Manual.; https://www.decawave.com/mdek1001/usermanual/

[15] Mapa Escuela Politécnica.; http://ingenieria-telematica-uah.blogspot.com/2011/12/quien-es-quien-en-la-eps.html

[16] Rapidly Implement a RTLS.; https://www.digikey.es/es/articles/rapidly-implement-a-real-time-location-system

[17] Accelerometer LIS2DH12 Datasheet.; https://www.st.com/resource/en/datasheet/lis2dh12.pdf

[18] Social Distancing Monitor.; https://312bd7de-1ecc-4366-a239-93a3dbab7211.filesusr.com/ugd/6563b9_eb748053d6e84803a6d4dc0f5125ae61.pdf?index=true

[19] Human Real Time Localization System in Underground Mines using UWB.; https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9085295

[20] Pulso UWB.; https://www.researchgate.net/figure/UWB-pulse-in-time-domain-up-and-corresponding-spectrum-down_fig8_312190420

10. ANEXOS

10.1. Anexo 1. Código dwm-simple.c de SEGGER Embedded Studio

```
 * Application entry point. Initialize application thread.
 *
 * @warning ONLY ENABLING OF LOCATION ENGINE OR BLE AND CREATION AND STARTING OF
 * USER THREADS CAN BE DONE IN THIS FUNCTION
 */
void dwm_user_start(void)
{
        uint8_t hndl;
        int rv;

        dwm_shell_compile();
        //Disabling ble by default as softdevice prevents debugging with breakpoints (due to priority)
        dwm_ble_compile();
        dwm_le_compile();
        dwm_serial_spi_compile();

          //configura el tag
        dwm_cfg_tag_t cfg;
        cfg.stnry_en = 0;
        cfg.meas_mode = DWM_MEAS_MODE_TWR;
        cfg.low_power_en = 0;
        cfg.loc_engine_en = 1;
        cfg.common.enc_en = 1;
        cfg.common.led_en = 1;
        cfg.common.ble_en = 1;
        cfg.common.fw_update_en = 0;
        cfg.common.uwb_mode = DWM_UWB_MODE_ACTIVE;
        rv = dwm_cfg_tag_set(&cfg);
        APP_ERR_CHECK(rv);

        /* Create thread */
        rv = dwm_thread_create(THREAD_APP_PRIO, app_thread_entry, (void*)NULL,
                        "app", THREAD_APP_STACK_SIZE, &hndl);
        APP_ERR_CHECK(rv);

        /* Start the thread */
        dwm_thread_resume(hndl);
}
```

```
void CheckButton(void)

{

bool value;

int res;

// #define      DWM_OK  (0)

res = dwm_gpio_value_get( SLEEP_BTN_GPIO, &value);


if ( res != DWM_OK ) {

res = dwm_gpio_value_set( LED_BTN_GPIO, 0);


// in case of any error assuming button is not pressed


}

// check pin value, light LED if pressed and return current button state

if ( value == 0 ) {

        dwm_gpio_value_set( DWM_GPIO_IDX_14, false);


        printf("LED ENCENDIDO\n");
        dwm_usr_data_write("AYUDA",5,false);

}
}
```

```
/**
 * Event callback
 *
 * @param[in] p_evt  Pointer to event structure
 */

void on_dwm_evt(dwm_evt_t *p_evt)
{
        int len;
        int i;
        uint16_t static range_request = 0;
        uint16_t static cnt_data_out = 0;

        // Data to send in IoT data
        uint8_t data_out[34] = {0};
        // Data to send in IoT data
        uint8_t data_in[34] = {0};
        // Current Idc to store in data
        uint8_t idx = 0;

        switch (p_evt->header.id) {
        /* New location data */
        case DWM_EVT_LOC_READY:
                printf("\nT:%lu ", dwm_systime_us_get());
                if (p_evt->loc.pos_available) {
                        printf("POS:[%ld,%ld,%ld,%u] ", p_evt->loc.pos.x,
                                        p_evt->loc.pos.y, p_evt->loc.pos.z,
                                        p_evt->loc.pos.qf);
                } else {
                        printf("POS:N/A ");
                }
```

```
        for (i = 0; i < p_evt->loc.anchors.dist.cnt; ++i) {
                printf("DIST%d:", i);

                printf("0x%04X", (unsigned int)(p_evt->loc.anchors.dist.addr[i] & 0xffff));
                if (i < p_evt->loc.anchors.an_pos.cnt) {
                        printf("[%ld,%ld,%ld]",
                                        p_evt->loc.anchors.an_pos.pos[i].x,
                                        p_evt->loc.anchors.an_pos.pos[i].y,
                                        p_evt->loc.anchors.an_pos.pos[i].z);
                }

                printf("=[%lu,%u] ", p_evt->loc.anchors.dist.dist[i],
                                p_evt->loc.anchors.dist.qf[i]);
        }
        printf("\n");
        break;

case DWM_EVT_USR_DATA_READY:
        len = p_evt->header.len
        - sizeof(dwm_evt_hdr_t);
        if (len <= 0)
                break;

        dwm_usr_data_read(data_in,sizeof(p_evt->header.len));
        printf("iot received, len=%d:", len);
        for (i = 0; i < len; ++i) {
                printf(" %02X", p_evt->usr_data[i]);
                if (p_evt->usr_data[i] == 0x0A)
                        dwm_gpio_cfg_output(DWM_GPIO_IDX_14, false);
                else if (p_evt->usr_data[i] == 0x0B)
                        dwm_gpio_cfg_output(DWM_GPIO_IDX_14, true);

                printf("%c", p_evt->usr_data[i]);
        }
        break;
```

```
        case DWM_EVT_USR_DATA_SENT:
                printf("iot sent\n");
                break;

        case DWM_EVT_BH_INITIALIZED_CHANGED:
                printf("uwbmac: backhaul = %d\n", p_evt->bh_initialized);
                break;

        case DWM_EVT_UWBMAC_JOINED_CHANGED:
                printf("uwbmac: joined = %d\n", p_evt->uwbmac_joined);
                break;

        default:
                break;
        }
}

/**
 * Application thread
 *
 * @param[in] data  Pointer to user data
 */
void app_thread_entry(uint32_t data)
{
        dwm_cfg_t cfg;
        uint8_t i2cbyte;
        dwm_evt_t evt;
        int rv, cnt_caida = 0, flag_caida = 0;
        uint8_t label[DWM_LABEL_LEN_MAX];
        uint8_t label_len = DWM_LABEL_LEN_MAX;
        dwm_pos_t pos;
        uint8_t acc_x_L, acc_x_H, acc_y_L, acc_y_H, acc_z_L, acc_z_H, temp_H, temp_L, temp;
        int16_t acc_x_g, acc_y_g, acc_z_g;
        uint8_t config [2];
        int16_t acc_x, acc_y, acc_z, temp_16;
        float_t acc;
```

```
 dwm_gpio_cfg_input( SLEEP_BTN_GPIO, DWM_GPIO_PIN_PULLUP); //configura boton user como entrada

// configure LED pin, blink twice

dwm_gpio_cfg_output( DWM_GPIO_IDX_14, true);

/* Initial message */
printf(MSG_INIT);

/* Get node configuration */
APP_ERR_CHECK(dwm_cfg_get(&cfg));

/* Update rate set to 1 second, stationary update rate set to 5 seconds */
APP_ERR_CHECK(dwm_upd_rate_set(10, 50));

/* Sensitivity for switching between stationary and normal update rate */
APP_ERR_CHECK(dwm_stnry_cfg_set(DWM_STNRY_SENSITIVITY_NORMAL));

///* Register event callback */
dwm_evt_listener_register(
                DWM_EVT_LOC_READY | DWM_EVT_USR_DATA_READY |
                DWM_EVT_BH_INITIALIZED_CHANGED |
                DWM_EVT_UWBMAC_JOINED_CHANGED, NULL);


i2cbyte = 0x0f;
rv = dwm_i2c_write(0x33 >> 1, &i2cbyte, 1, true);


if (rv == DWM_OK) {
        rv = dwm_i2c_read(0x33 >> 1, &i2cbyte, 1);


        if (rv == DWM_OK) {
                printf("Accelerometer chip ID: %u\n", i2cbyte);

        } else {
                printf("i2c: read failed (%d)\n", rv);
        }
} else {
        printf("i2c: write failed (%d)\n", rv);
}


// Configurar registros para el acelerometro
config[0] = 0x1F;
config[1] = 0xC0 ;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);

config[0] = 0x20;
config[1] = 0x00 ;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);

config[0] = 0x24;
config[1] = 0x00;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);

config[0] = 0x20;
config[1] = 0x57;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);

config[0] = 0x21;
config[1] = 0x00 ;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);
```

```
config[0] = 0x22;
config[1] = 0x10 ;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);

config[0] = 0x23;
config[1] = 0x88 ;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);

config[0] = 0x25;
config[1] = 0x00 ;
rv = dwm_i2c_write(0x33 >> 1, config, 2, true);


rv = dwm_label_read(label, &label_len);

if (rv == DWM_OK) {
        printf("LABEL(len=%d):", label_len);
        for (rv = 0; rv < label_len; ++rv) {
                printf(" %02x", label[rv]);
        }
        printf("\n");
} else {
        printf("can't read label len=%d, error %d\n", label_len, rv);
}

while (1) {
        CheckButton() ;
        // delay 10ms
        dwm_thread_delay(10);

        dwm_pos_get(&pos);
```

```
//Leer temperatura
temp_L = 0X0C;
rv = dwm_i2c_write(0x33 >> 1, &temp_L, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &temp_L, 1);
//printf ("Temperatura L:%d \n",temp_L);
temp_H = 0X0D;
rv = dwm_i2c_write(0x33 >> 1, &temp_H, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &temp_H, 1);
//printf ("Temperatura H:%d \n",temp_H);

temp_16 = ( (temp_H << 8) | temp_L);
temp_16 = (temp_16 >> 6); //10 bits de la izquierda para high y normal resolution
printf ("Temperatura:%d \n",temp_16);



//Leer valores del acelerometro
acc_x_L = 0x28;
rv = dwm_i2c_write(0x33 >> 1, &acc_x_L, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &acc_x_L, 1);
acc_x_H = 0x29;
rv = dwm_i2c_write(0x33 >> 1, &acc_x_H, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &acc_x_H, 1);

acc_y_L = 0x2A;
rv = dwm_i2c_write(0x33 >> 1, &acc_y_L, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &acc_y_L, 1);
acc_y_H = 0x2B;
rv = dwm_i2c_write(0x33 >> 1, &acc_y_H, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &acc_y_H, 1);

acc_z_L = 0x2C;
rv = dwm_i2c_write(0x33 >> 1, &acc_z_L, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &acc_z_L, 1);
acc_z_H = 0x2D;
rv = dwm_i2c_write(0x33 >> 1, &acc_z_H, 1, true);
rv = dwm_i2c_read(0x33 >> 1, &acc_z_H, 1);
```

```
        acc_x = ( (acc_x_H << 8) | acc_x_L);
        acc_y = ( (acc_y_H << 8) | acc_y_L);
        acc_z = ( (acc_z_H << 8) | acc_z_L);
        acc_x_g = (acc_x >> 4); //12 bits de la izquierda para high resolution (resultado en mg)
        acc_y_g = (acc_y >> 4); //12 bits de la izquierda  para high resolution (resultado en mg)
        acc_z_g = (acc_z >> 4); //12 bits de la izquierda  para high resolution (resultado en mg)

        printf ("Acelerometro:%d, %d, %d \n\n", acc_x_g, acc_y_g, acc_z_g);

        //Detector de caídas
        acc = sqrt((pow(((float)acc_x_g/1000),2))+(pow(((float)acc_y_g/1000),2))+(pow(((float)acc_z_g/1000),2)));
        printf("Aceleracion: %.3f \n\n", acc);
        if(acc < 0.5){
          flag_caida = 1;
        }
        if (flag_caida){
          cnt_caida++;
          if (acc > 0.75){
            flag_caida = 0;
            }
        }
        if (cnt_caida > 3){
          dwm_usr_data_write("POSIBLE CAIDA",13,false);
          cnt_caida = 0;
        }
        printf("x=%ld, y=%ld, z=%ld, qf=%u \n", pos.x, pos.y, pos.z, pos.qf);
        printf("\t\t time=%lu \n", dwm_systime_us_get());

        /* Thread loop */
        rv = dwm_evt_wait(&evt);

        if (rv != DWM_OK) {
                printf("dwm_evt_wait, error %d\n", rv);
        } else {
                on_dwm_evt(&evt);
        }
}
```

10.2. Anexo 2. Código MatLab para la extracción de las coordenadas del tag y el dibujo de las figuras

```
% Script para extraer las coordenadas del tag obtenidas de sus respectivos
% ficheros log .

clear all ; close all ; clc;

%Recopila los datos del recorrido exterior UWB
fichero = '25_02_recorrido_fuera';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero txt y conversion a array de las coordenadas del tag.
tabla_datos = readtable ( fichero_txt );
circ_ext_UWB = table2array ( tabla_datos (: ,4:5) );
% Almacenamiento de las coordenadas en un fichero .mat
save ( 'circ_ext_UWB' ,'circ_ext_UWB');

%Recopila datos del recorrido interior UWB
fichero = '25_02_recorrido_dentro';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_recorrido_dentro = readtable ( fichero_txt );
circ_int_UWB = table2array ( tabla_datos_recorrido_dentro (: ,4:5) );
% Almacenamiento de las coordenadas en un fichero .mat
save ( 'circ_int_UWB' ,'circ_int_UWB');

%Recopila datos puntos del centro
fichero = 'teratermN21_centro_s1';
fichero_txt = strcat ( fichero , '.txt');
fichero_mat = strcat ( fichero , '.mat');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_centro = readtable ( fichero_txt );
posiciones_tag_centro = table2array ( tabla_datos_centro (: ,4:5) );
% Almacenamiento de las coordenadas en un fichero .mat
save ( fichero_mat ,'posiciones_tag_centro');

%Recopila datos de la esquina 1
fichero = 'teratermN21_esquina1_s1';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_esquina1 = readtable ( fichero_txt );
posiciones_tag_esquina1 = table2array ( tabla_datos_esquina1 (: ,4:5) );

%Recopila datos de la esquina 2
fichero = 'teratermN21_esquina2_s1';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_esquina2 = readtable ( fichero_txt );
posiciones_tag_esquina2 = table2array ( tabla_datos_esquina2 (: ,4:5) );

%Recopila datos de la esquina 3
fichero = 'teratermN21_esquina3_s1';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_esquina3 = readtable ( fichero_txt );
```

```
posiciones_tag_esquina3 = table2array ( tabla_datos_esquina3 (: ,4:5) );

%Recopila datos de la esquina 4
fichero = 'teratermN21_esquina4_s1';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_esquina4 = readtable ( fichero_txt );
posiciones_tag_esquina4 = table2array ( tabla_datos_esquina4 (: ,4:5) );

%Recopila datos del recorrido 1 OESTE
fichero = '17_03_PRUEBA1_OESTE';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_PRUEBA1_OESTE = readtable ( fichero_txt );
circ_UWB_OESTE1 = table2array ( tabla_datos_PRUEBA1_OESTE (: ,4:5) );

%Recopila datos del recorrido 2 OESTE
fichero = '17_03_PRUEBA2_OESTE';
fichero_txt = strcat ( fichero , '.txt');
% Lectura del fichero log y conversion a array de las coordenadas del tag.
tabla_datos_PRUEBA2_OESTE = readtable ( fichero_txt );
circ_UWB_OESTE2 = table2array ( tabla_datos_PRUEBA2_OESTE (: ,4:5) );

%Posiciones de los anchors, de las esquinas, del centro y de los ground
%truth
Recorrido_real_fuera_x = [2.558 2.558 5.36 5.36 2.558];
Recorrido_real_fuera_y = [1.707 5.317 5.317 1.707 1.707];
Recorrido_real_dentro_x = [2.958 2.958 3.758 3.758 2.958 2.958 4.958 4.958 4.158
4.158 4.958 4.958 2.958];
Recorrido_real_dentro_y = [2.507 3.307 3.307 3.707 3.707 4.507 4.507 3.707 3.707
3.307 3.307 2.507 2.507];
posicion_anchor_uno = [0.25 0.03];
posicion_anchor_dos = [0.15 7.03];
posicion_anchor_tres = [7.64 7.03];
posicion_anchor_cuatro = [7.64 0.03];
posicion_anchor_cinco = [-17.70 -19];
posicion_anchor_seis = [-22 -34.8];
posicion_anchor_siete = [-29.5 -42.7];
posicion_anchor_ocho = [-44 -48.8];
posicion_anchor_nueve = [-41.61 -24.3];
posicion_anchor_diez = [-26.60 -27.70];
posicion_anchor_once = [-32.50 -33];

posicion_tag_centro_real = [3.959 3.512];
Punto_esquina_uno_real = [2.558 1.707];
Punto_esquina_dos_real = [2.558 5.317];
Punto_esquina_tres_real = [5.36 5.317];
Punto_esquina_cuatro_real = [5.36 1.707];

%Calculo de errores del recorrido exterior UWB
for i = 1 :1:size ( circ_ext_UWB,1)

    if ((circ_ext_UWB(i,1) >= 2.81)&& (circ_ext_UWB(i,1) <= 5.11)) %2.81 < x < 5.11
        if((circ_ext_UWB(i,2) >= 0)&& (circ_ext_UWB(i,2) <= 2.6)) % 0 < y < 2.6
            error_UWB(i) = abs (circ_ext_UWB(i,2)- Recorrido_real_fuera_y(1));

        elseif((circ_ext_UWB(i,2) >= 4)&& (circ_ext_UWB(i,2) <= 7)) % 4 < y < 7
            error_UWB(i) = abs(circ_ext_UWB(i,2)- Recorrido_real_fuera_y(2));
        end
```

```
    elseif ((circ_ext_UWB(i,2) >= 1.957)&& (circ_ext_UWB(i,2) <= 5.067)) % 1.957 < y
< 5.067
        if ((circ_ext_UWB(i,1) >= 0)&& (circ_ext_UWB(i,1) <= 3.30)) % 0 < x < 3.30
            error_UWB(i) = abs(circ_ext_UWB(i,1)- Recorrido_real_fuera_x(1));

        elseif ((circ_ext_UWB(i,1) >= 4.60)&& (circ_ext_UWB(i,1) <= 7)) % 4.60 < x <
7
            error_UWB(i) = abs(circ_ext_UWB(i,1)- Recorrido_real_fuera_x(3));
        end
    end

end
error_circ_ext_UWB = error_UWB.';

%Calcula el error al centro
dist_centro = posicion_tag_centro_real - posiciones_tag_centro;
for i = 1:1:size ( dist_centro,1)
    dist_error(i) = norm(dist_centro(i,:));
end
errores_centro = dist_error.';


%Calcula el error a la esquina 1
dist_esquina1 = Punto_esquina_uno_real - posiciones_tag_esquina1;
for i = 1:1:size ( dist_esquina1,1)
    dist_error_esquina1(i) = norm(dist_esquina1(i,:));
end
errores_esquina_1 = dist_error_esquina1.';


%Calcula el error a la esquina 2
dist_esquina2 = Punto_esquina_dos_real - posiciones_tag_esquina2;
for i = 1:1:size ( dist_esquina2,1)
    dist_error_esquina2(i) = norm(dist_esquina2(i,:));
end
errores_esquina_2 = dist_error_esquina2.';


%Calcula el error a la esquina 3
dist_esquina3 = Punto_esquina_tres_real - posiciones_tag_esquina3;
for i = 1:1:size ( dist_esquina3,1)
    dist_error_esquina3(i) = norm(dist_esquina3(i,:));
end
errores_esquina_3 = dist_error_esquina3.';


%Calcula el error a la esquina 4
dist_esquina4 = Punto_esquina_cuatro_real - posiciones_tag_esquina4;
for i = 1:1:size ( dist_esquina4,1)
    dist_error_esquina4(i) = norm(dist_esquina4(i,:));
end
errores_esquina_4 = dist_error_esquina4.';

% Se indica el titulo de la grafica
titulo = sprintf (' Comparación de posicionamiento medido y ground truth del tag
recorrido exterior' );
```

```
titulo1 = sprintf (' Comparación de posicionamiento medido y ground truth del tag
recorrido interior' );
titulo2 = sprintf (' Comparación de posicionamiento medido y puntos reales del tag en
esquinas y centro' );
titulo3 = sprintf (' Errores CDF ');
titulo4 = sprintf (' Recorridos en el OESTE ');


%%%%%%% RECORRIDO EXTERIOR %%%%%%%%%%%

%Representa los puntos donde se situan los anchors
figure (1); p = plot ( posicion_anchor_uno (1) , posicion_anchor_uno (2));
p. LineStyle = 'none '; p. Marker = '^'; p. MarkerSize = 10; p. Color = 'r';
xlabel ('X (m)'); ylabel ('Y (m)'); title ( titulo );
axis equal; axis ([0 7.856 0 7.1]) ; grid on; grid minor ;
text(posicion_anchor_uno (1), posicion_anchor_uno (2), '1 (0.25, 0.03)');
hold on;

p1 = plot ( posicion_anchor_dos (1) , posicion_anchor_dos (2));
p1. LineStyle = 'none '; p1. Marker = '^'; p1. MarkerSize = 10; p1. Color = 'r';
text(posicion_anchor_dos (1), posicion_anchor_dos (2), '2 (0.15, 7.03)');
hold on;

p2 = plot ( posicion_anchor_tres (1) , posicion_anchor_tres (2));
p2. LineStyle = 'none '; p2. Marker = '^'; p2. MarkerSize = 10; p2. Color = 'r';
text(posicion_anchor_tres (1), posicion_anchor_tres (2), '3 (7.64, 7.03)');
hold on;

p3 = plot ( posicion_anchor_cuatro (1) , posicion_anchor_cuatro (2));
p3. LineStyle = 'none '; p3. Marker = '^'; p3. MarkerSize = 10; p3. Color = 'r';
text(posicion_anchor_cuatro (1), posicion_anchor_cuatro (2), '4 (7.64 0.03)');
hold on;

%Representa el recorrido exterior ground truth
p4 = plot (Recorrido_real_fuera_x,Recorrido_real_fuera_y);
p4. Color = 'k'; p4. LineWidth = 2;
hold on;

% Recorrido exterior medido con UWB .
p5 = plot ( circ_ext_UWB (: ,1) , circ_ext_UWB (: ,2));
p5. LineStyle = 'none '; p5. Marker = '.'; p5. MarkerSize = 7; p5. Color = 'g';

legend([p1 p4 p5],{'Anchors','Ground truth','Recorrido medido'},'FontSize',15,
'Location','southeast');




%%%%%%%%%%%%%% RECORRIDO INTERIOR %%%%%%%%%%%%%%%%%%%%

%Representa los puntos donde se situan los anchors
figure (2); r = plot ( posicion_anchor_uno (1) , posicion_anchor_uno (2));
r. LineStyle = 'none '; r. Marker = '^'; r. MarkerSize = 10; r. Color = 'r';
xlabel ('X (m)'); ylabel ('Y (m)'); title ( titulo1 );
axis equal; axis ([0 7.856 0 7.1]) ;  grid on; grid minor ;
text(posicion_anchor_uno (1), posicion_anchor_uno (2), '1 (0.25, 0.03)');
hold on;
```

```
r1 = plot ( posicion_anchor_dos (1) , posicion_anchor_dos (2));
r1. LineStyle = 'none '; r1. Marker = '^'; r1. MarkerSize = 10; r1. Color = 'r';
text(posicion_anchor_dos (1), posicion_anchor_dos (2), '2 (0.15, 7.03)');
hold on;

r2 = plot ( posicion_anchor_tres (1) , posicion_anchor_tres (2));
r2. LineStyle = 'none '; r2. Marker = '^'; r2. MarkerSize = 10; r2. Color = 'r';
text(posicion_anchor_tres (1), posicion_anchor_tres (2), '3 (7.64, 7.03)');
hold on;

r3 = plot ( posicion_anchor_cuatro (1) , posicion_anchor_cuatro (2));
r3. LineStyle = 'none '; r3. Marker = '^'; r3. MarkerSize = 10; r3. Color = 'r';
text(posicion_anchor_cuatro (1), posicion_anchor_cuatro (2), '4 (7.64 0.03)');
hold on;

%Representa el recorrido interior ground truth
r4 = plot (Recorrido_real_dentro_x,Recorrido_real_dentro_y);
r4. Color = 'k'; r4. LineWidth = 2;
hold on;

%Recorrido interior medido UWB.
r5 = plot ( circ_int_UWB (: ,1) , circ_int_UWB (: ,2));
r5. LineStyle = 'none '; r5. Marker = '.'; r5. MarkerSize = 7; r5. Color = 'g';

legend([r1 r4 r5],{'Anchors','Ground truth','Recorrido medido'},'FontSize',15,
'Location','southeast');


%%%%%%%%%%%%%%%%%% ESQUINAS Y CENTRO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Representa los puntos donde se situan los anchors
figure (3); q = plot ( posicion_anchor_uno (1) , posicion_anchor_uno (2));
q. LineStyle = 'none '; q. Marker = '^'; q. MarkerSize = 10; q. Color = 'r';
xlabel ('X (m)'); ylabel ('Y (m)'); title ( titulo2 );
axis equal; axis ([0 7.856 0 7.1]) ;  grid on; grid minor ;
text(posicion_anchor_uno (1), posicion_anchor_uno (2), '1 (0.25, 0.03)');
hold on;

q1 = plot ( posicion_anchor_dos (1) , posicion_anchor_dos (2));
q1. LineStyle = 'none '; q1. Marker = '^'; q1. MarkerSize = 10; q1. Color = 'r';
text(posicion_anchor_dos (1), posicion_anchor_dos (2), '2 (0.15, 7.03)');
hold on;

q2 = plot ( posicion_anchor_tres (1) , posicion_anchor_tres (2));
q2. LineStyle = 'none '; q2. Marker = '^'; q2. MarkerSize = 10; q2. Color = 'r';
text(posicion_anchor_tres (1), posicion_anchor_tres (2), '3 (7.64, 7.03)');
hold on;

q3 = plot ( posicion_anchor_cuatro (1) , posicion_anchor_cuatro (2));
q3. LineStyle = 'none '; q3. Marker = '^'; q3. MarkerSize = 10; q3. Color = 'r';
text(posicion_anchor_cuatro (1), posicion_anchor_cuatro (2), '4 (7.64 0.03)');
hold on;

%Representa el punto central real
q4 = plot( posicion_tag_centro_real (1), posicion_tag_centro_real (2));
q4. LineStyle = 'none '; q4. Marker = 'x'; q4. MarkerSize = 10; q4. Color = 'k';
text(posicion_tag_centro_real (1), posicion_tag_centro_real (2), 'Centro (3.959,
3.512)');
```

```
hold on;

%Representa la esquina 1 real
q5 = plot( Punto_esquina_uno_real (1), Punto_esquina_uno_real (2));
q5. LineStyle = 'none '; q5. Marker = 'x'; q5. MarkerSize = 10; q5. Color = 'k';
text(Punto_esquina_uno_real (1), Punto_esquina_uno_real (2), 'E1 (2.558, 7.707)');
hold on;

%Representa la esquina 2 real
q6 = plot( Punto_esquina_dos_real (1), Punto_esquina_dos_real (2));
q6. LineStyle = 'none '; q6. Marker = 'x'; q6. MarkerSize = 10; q6. Color = 'k';
text(Punto_esquina_dos_real (1), Punto_esquina_dos_real (2), 'E2 (2.558, 5.317)');
hold on;

%Representa la esquina 3 real
q7 = plot( Punto_esquina_tres_real (1), Punto_esquina_tres_real (2));
q7. LineStyle = 'none '; q7. Marker = 'x'; q7. MarkerSize = 10; q7. Color = 'k';
text(Punto_esquina_tres_real (1), Punto_esquina_tres_real (2), 'E3 (5.360, 5.317)');
hold on;

%Representa la esquina 4 real
q8 = plot( Punto_esquina_cuatro_real (1), Punto_esquina_cuatro_real (2));
q8. LineStyle = 'none '; q8. Marker = 'x'; q8. MarkerSize = 10; q8. Color = 'k';
text(Punto_esquina_cuatro_real (1), Punto_esquina_cuatro_real (2), 'E4 (5.360,
1.707)');
hold on;

%Representa los puntos medidos del tag en el centro
q9 = plot ( posiciones_tag_centro (: ,1) , posiciones_tag_centro (: ,2));
q9. LineStyle = 'none '; q9. Marker = '.'; q9. MarkerSize = 10; q9. Color = 'g';
hold on;

%Representa los puntos medidos del tag en la esquina 1
q10 = plot ( posiciones_tag_esquina1 (: ,1) , posiciones_tag_esquina1 (: ,2));
q10. LineStyle = 'none '; q10. Marker = '.'; q10. MarkerSize = 10; q10. Color = 'g';
hold on;

%Representa los puntos medidos del tag en la esquina 2
q11 = plot ( posiciones_tag_esquina2 (: ,1) , posiciones_tag_esquina2 (: ,2));
q11. LineStyle = 'none '; q11. Marker = '.'; q11. MarkerSize = 10; q11. Color = 'g';
hold on;

%Representa los puntos medidos del tag en la esquina 3
q12 = plot ( posiciones_tag_esquina3 (: ,1) , posiciones_tag_esquina3 (: ,2));
q12. LineStyle = 'none '; q12. Marker = '.'; q12. MarkerSize = 10; q12. Color = 'g';
hold on;

%Representa los puntos medidos del tag en la esquina 4
q13 = plot ( posiciones_tag_esquina4 (: ,1) , posiciones_tag_esquina4 (: ,2));
q13. LineStyle = 'none '; q13. Marker = '.'; q13. MarkerSize = 11; q13. Color = 'g';
hold on;

legend([q1 q4 q13],{'Anchors','Puntos reales','Puntos medidos'},'FontSize',15,
'Location','southeast');



%%%%%%% RECORRIDO OESTE %%%%%%%%%%%
```

```
%Representa los puntos donde se situan los anchors
figure (4); s = plot ( posicion_anchor_cinco (1) , posicion_anchor_cinco (2));
s. LineStyle = 'none '; s. Marker = '^'; s. MarkerSize = 10; s. Color = 'r';
xlabel ('X (m)'); ylabel ('Y (m)'); title ( titulo4 );
axis equal; axis ([-50 -15 -50 -15]) ;  grid on; grid minor ;
text(posicion_anchor_cinco (1), posicion_anchor_cinco (2), '5 (-17.7, -19.0)');
hold on;

s1 = plot ( posicion_anchor_seis (1) , posicion_anchor_seis (2));
s1. LineStyle = 'none '; s1. Marker = '^'; s1. MarkerSize = 10; s1. Color = 'r';
text(posicion_anchor_seis (1), posicion_anchor_seis (2), '6 (-22.0, -34.8)');
hold on;

s2 = plot ( posicion_anchor_siete (1) , posicion_anchor_siete (2));
s2. LineStyle = 'none '; s2. Marker = '^'; s2. MarkerSize = 10; s2. Color = 'r';
text(posicion_anchor_siete (1), posicion_anchor_siete (2), '7 (-29.5, -42.7)');
hold on;

s3 = plot ( posicion_anchor_ocho (1) , posicion_anchor_ocho (2));
s3. LineStyle = 'none '; s3. Marker = '^'; s3. MarkerSize = 10; s3. Color = 'r';
text(posicion_anchor_ocho (1), posicion_anchor_ocho (2), '8 (-44.0, -48.8)');
hold on;

s4 = plot ( posicion_anchor_nueve (1) , posicion_anchor_nueve (2));
s4. LineStyle = 'none '; s4. Marker = '^'; s4. MarkerSize = 10; s4. Color = 'r';
text(posicion_anchor_nueve (1), posicion_anchor_nueve (2), '9 (-41.61, -24.30)');
hold on;

s5 = plot ( posicion_anchor_diez (1) , posicion_anchor_diez (2));
s5. LineStyle = 'none '; s5. Marker = '^'; s5. MarkerSize = 10; s5. Color = 'r';
text(posicion_anchor_diez (1), posicion_anchor_diez (2), '10 (-26.6, -27.7)');
hold on;

s6 = plot ( posicion_anchor_once (1) , posicion_anchor_once (2));
s6. LineStyle = 'none '; s6. Marker = '^'; s6. MarkerSize = 10; s6. Color = 'r';
text(posicion_anchor_once (1), posicion_anchor_once (2), '11 (-32.5, -33.0)');
hold on;

% Recorrido 1 medido con UWB .
s7 = plot ( circ_UWB_OESTE1 (: ,1) , circ_UWB_OESTE1 (: ,2));
s7. LineStyle = 'none '; s7. Marker = '.'; s7. MarkerSize = 7; s7. Color = 'g';
hold on;

% Recorrido 2 medido con UWB .
s8 = plot ( circ_UWB_OESTE2 (: ,1) , circ_UWB_OESTE2 (: ,2));
s8. LineStyle = 'none '; s8. Marker = '.'; s8. MarkerSize = 7; s8. Color = 'b';

legend([s1 s7 s8],{'Anchors','Recorrido medido 1', 'Recorrido medido
2'},'FontSize',15, 'Location','southeast');


%%%%%%% RECORRIDO OESTE Sin ejes %%%%%%%%%%%%
%Representa los puntos donde se situan los anchors
figure (5); s = plot ( posicion_anchor_cinco (1) , posicion_anchor_cinco (2));
s. LineStyle = 'none '; s. Marker = '^'; s. MarkerSize = 10; s. Color = 'r';
axis equal; axis ([-50 -15 -50 -15]) ;
axis off;
```

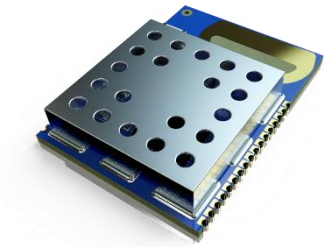
```
text(posicion_anchor_cinco (1), posicion_anchor_cinco (2), '5 (-17.7, -19.0)');
hold on;

s1 = plot ( posicion_anchor_seis (1) , posicion_anchor_seis (2));
s1. LineStyle = 'none '; s1. Marker = '^'; s1. MarkerSize = 10; s1. Color = 'r';
text(posicion_anchor_seis (1), posicion_anchor_seis (2), '6 (-22.0, -34.8)');
hold on;

s2 = plot ( posicion_anchor_siete (1) , posicion_anchor_siete (2));
s2. LineStyle = 'none '; s2. Marker = '^'; s2. MarkerSize = 10; s2. Color = 'r';
text(posicion_anchor_siete (1), posicion_anchor_siete (2), '7 (-29.5, -42.7)');
hold on;

s3 = plot ( posicion_anchor_ocho (1) , posicion_anchor_ocho (2));
s3. LineStyle = 'none '; s3. Marker = '^'; s3. MarkerSize = 10; s3. Color = 'r';
text(posicion_anchor_ocho (1), posicion_anchor_ocho (2), '8 (-44.0, -48.8)');
hold on;

s4 = plot ( posicion_anchor_nueve (1) , posicion_anchor_nueve (2));
s4. LineStyle = 'none '; s4. Marker = '^'; s4. MarkerSize = 10; s4. Color = 'r';
text(posicion_anchor_nueve (1), posicion_anchor_nueve (2), '9 (-41.61, -24.30)');
hold on;

s5 = plot ( posicion_anchor_diez (1) , posicion_anchor_diez (2));
s5. LineStyle = 'none '; s5. Marker = '^'; s5. MarkerSize = 10; s5. Color = 'r';
text(posicion_anchor_diez (1), posicion_anchor_diez (2), '10 (-26.6, -27.7)');
hold on;

s6 = plot ( posicion_anchor_once (1) , posicion_anchor_once (2));
s6. LineStyle = 'none '; s6. Marker = '^'; s6. MarkerSize = 10; s6. Color = 'r';
text(posicion_anchor_once (1), posicion_anchor_once (2), '11 (-32.5, -33.0)');
hold on;

% Recorrido 1 medido con UWB .
s7 = plot ( circ_UWB_OESTE1 (: ,1) , circ_UWB_OESTE1 (: ,2));
s7. LineStyle = 'none '; s7. Marker = '.'; s7. MarkerSize = 7; s7. Color = 'g';
hold on;

% Recorrido 2 medido con UWB .
s8 = plot ( circ_UWB_OESTE2 (: ,1) , circ_UWB_OESTE2 (: ,2));
s8. LineStyle = 'none '; s8. Marker = '.'; s8. MarkerSize = 7; s8. Color = 'b';




%%%%%%% RECORRIDO EXTERIOR E INTERIOR SIN EJES %%%%%%%%%%%
%Representa los puntos donde se situan los anchors
figure (6); t = plot ( posicion_anchor_uno (1) , posicion_anchor_uno (2));
t. LineStyle = 'none '; t. Marker = '^'; t. MarkerSize = 10; t. Color = 'r';
axis equal; axis ([-2 9.856 -2 7.1]) ;
axis off;
text(posicion_anchor_uno (1), posicion_anchor_uno (2), '1 (0.25, 0.03)');
hold on;

t1 = plot ( posicion_anchor_dos (1) , posicion_anchor_dos (2));
t1. LineStyle = 'none '; t1. Marker = '^'; t1. MarkerSize = 10; t1. Color = 'r';
text(posicion_anchor_dos (1), posicion_anchor_dos (2), '2 (0.15, 7.03)');
hold on;
```

```
t2 = plot ( posicion_anchor_tres (1) , posicion_anchor_tres (2));
t2. LineStyle = 'none '; t2. Marker = '^'; t2. MarkerSize = 10; t2. Color = 'r';
text(posicion_anchor_tres (1), posicion_anchor_tres (2), '3 (7.64, 7.03)');
hold on;

t3 = plot ( posicion_anchor_cuatro (1) , posicion_anchor_cuatro (2));
t3. LineStyle = 'none '; t3. Marker = '^'; t3. MarkerSize = 10; t3. Color = 'r';
text(posicion_anchor_cuatro (1), posicion_anchor_cuatro (2), '4 (7.64 0.03)');
text(0, 0, 'Origen (0,0)');
hold on;

% Recorrido exterior medido con UWB .
t4 = plot ( circ_ext_UWB (: ,1) , circ_ext_UWB (: ,2));
t4. LineStyle = 'none '; t4. Marker = '.'; t4. MarkerSize = 7; t4. Color = 'g';
hold on;

%Recorrido interior medido UWB.
t5 = plot ( circ_int_UWB (: ,1) , circ_int_UWB (: ,2));
t5. LineStyle = 'none '; t5. Marker = '.'; t5. MarkerSize = 7; t5. Color = 'b';


%%%%%%%%%%%% ERRORES CDF %%%%%%%%%%%%%%%%

% Representa todos los errores cdf
figure(7);
Func_Representacion_CDF(3,10,1,[0 0.5 0 1],'southeast',errores_centro,'Errores
centro','r','x',errores_esquina_1,'Errores esquina
1','k','o',errores_esquina_2,'Errores esquina 2','y','v',errores_esquina_3,'Errores
esquina 3','g','s' ...
,errores_esquina_4,'Errores esquina 4','b','x', error_circ_ext_UWB,'Errores recorrido
exterior','c','*');
title ( titulo3 );
```

10.3. Datasheet DWM1001, DWM1001-DEV y Acelerómetro lis2dh12

Product Overview

The DWM1001 module is based on Decawave's DW1000 Ultra Wideband (UWB) transceiver IC, which is an IEEE 802.15.4-2011 UWB implementation. It integrates UWB and Bluetooth® antenna, all RF circuitry, Nordic Semiconductor nRF52832 and a motion sensor.



Key Features

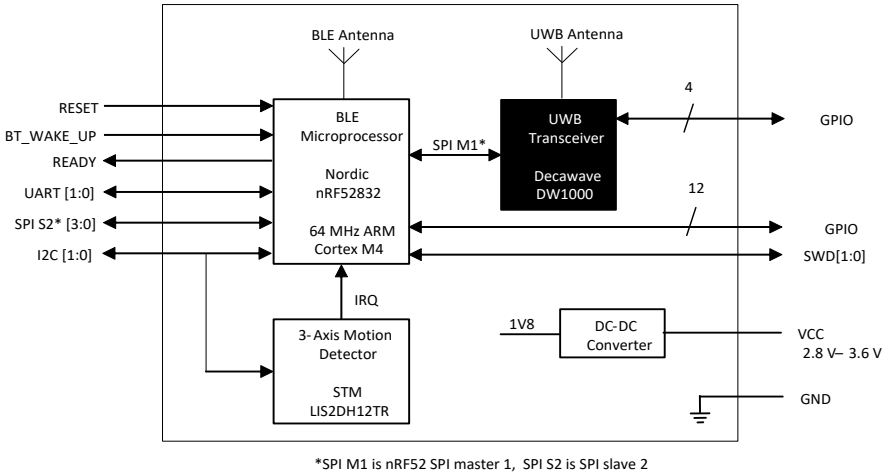
- Ranging accuracy to within 10cm.
- UWB Channel 5 printed PCB antenna (6.5 GHz)
- 6.8 Mbps data rate IEEE 802.15.4-2011 UWB compliant
- Nordic Semiconductor nRF52832
- Bluetooth® connectivity
- Bluetooth® chip antenna
- Motion sensor: 3-axis accelerometer
- Current consumption optimised for low power sleep mode: <15µA
- Supply voltage: 2.8 V to 3.6 V
- Size: 19.1 mm x 26.2 mm x 2.6 mm
- Modules marked DWM1001C are certified to ETSI, FCC and ISED regulations
    - FCC ID: 2AQ33-DWM1001, IC: 23794-DWM1001
    - See the module marking section of this datasheet

Key Benefits

Enables anchors, tags & gateways to quickly get an entire RTLS system up-and-running

- Accelerates product designs for faster Time-to-Market & reduced development costs
- Low-power hardware design and software architecture for longer battery life
- SPI, UART, I2C and *Bluetooth*® to interface DWM1001 with an external device
- Ready-to-go embedded firmware for Two Way Ranging RTLS application:
    - User API to DWM1001 firmware (available as a library) for user code customisation
    - On-board *Bluetooth*® SMART for connectivity to phones/tablets/PCs
    - SPI, UART and *Bluetooth*® APIs to access DWM1001 firmware from an external device

See MDEK1001 and PAN on www.decawave.com for additional information



*SPI M1 is nRF52 SPI master 1, SPI S2 is SPI slave 2

Block Diagram

Table of Contents

List of Figures

List of Tables

DOCUMENT INFORMATION

Disclaimer

Decawave reserves the right to change product specifications without notice. As far as possible changes to functionality and specifications will be issued in product specific errata sheets or in new versions of this document. Customers are advised to check with Decawave for the most recent updates on this product.

Modules labelled "**DWM1001**" are pre-loaded with PANS firmware, please refer to the "DWM1001 Firmware User Guide" for disclaimer and license terms.

Modules labelled "**DWM1001C**" are delivered without firmware - blank -.

LIFE SUPPORT POLICY

Decawave products are not authorized for use in safety-critical applications (such as life support) where a failure of the Decawave product would reasonably be expected to cause severe personal injury or death. Decawave customers using or selling Decawave products in such a manner do so entirely at their own risk and agree to fully indemnify Decawave and its representatives against any damages arising out of the use of Decawave products in such safety-critical applications.

Caution! ESD sensitive device. Precaution should be used when handling the device in order to prevent permanent damage.

REGULATORY APPROVALS

See Regulatory Information ..... Page 23

1 OVERVIEW

The block diagram on page 1 of this data sheet shows the major sections of the DWM1001. An overview of these blocks is given below.

1.1 UWB Transceiver DW1000

The module has a DW1000 UWB transceiver mounted on the PCB. The DW1000 uses a 38.4 MHz reference crystal. The crystal has been trimmed in production to reduce the initial frequency error to approximately 3 ppm, using the DW1000 IC's internal on-chip crystal trimming circuit.

Always-On (AON) memory can be used to retain DW1000 configuration data during the lowest power operational states when the on-chip voltage regulators are disabled. This data is uploaded and downloaded automatically. Use of DW1000 AON memory is configurable.

The on-chip voltage and temperature monitors allow the host to read the voltage on the VDDAON pin and the internal die temperature information from the DW1000.

See the DW1000 Datasheet [2] for more detailed information on device functionality, electrical specifications and typical performance.

1.2 Bluetooth® Microprocessor Nordic nRF52832

The nRF52832 is an ultra-low power 2.4 GHz wireless system on chip (SoC) integrating the nRF52 Series 2.4 GHz transceiver and an ARM Cortex-M4 CPU with 512kB flash memory and 64kB RAM.

See the nRF52832 Datasheet[1] for more detailed information on device functionality, electrical specifications and typical performance.

1.3 Power Supply and Power management

The power management circuit consists of a switch mode regulator. It is a buck convertor or step-down convertor. The input voltage to the DWM1001 can be in the range 2.8V to 3.6V. Outputs from the convertor provides 1.8V which is required by the DW1000[2]

1.4 Three Axis Motion Detector STMicroelectronics LIS2DH12TR

The LIS2DH12 is an ultra-low-power high performance three-axis linear accelerometer with digital I2C/SPI serial interface standard output. The LIS2DH12 has user-selectable full scales of ±2g/±4g/±8g/±16g and is capable of measuring accelerations with output data rates from 1 Hz to 5.3 kHz. The self-test capability allows the user to check the functionality of the sensor in the final application. The device may be configured to generate interrupt signals by detecting two independent inertial wake-up/free-fall events as well as by the position of the device itself. The LIS2DH12 is guaranteed to operate over an extended temperature range from -40 °C to +85 °C.

See the LIS2DH12TR Datasheet[4] for more detailed information on device functionality, electrical specifications and typical performance.

1.5 Software on board

The DWM1001C modules are delivered without firmware - blank -.

The DWM1001 modules come pre-loaded with embedded firmware which provides two-way ranging (TWR) and real time location system (RTLS) functionality. See the details in the DWM1001 Firmware User Guide [6]. The module can be configured and controlled via its API, which can be accessed through a number of different interfaces, allowing flexibility to the product designer. The details of the API are described in the DWM1001 Firmware API Guide [5]. Decawave also provides the module firmware in the form of binary libraries and some source code. A build environment is provided, so that the user can customise the operation and if required add their own functions[6].

2 DWM1001 CALIBRATION

Depending on the end-use applications and the system design, DWM1001 settings may need to be tuned. To help with this tuning several built in functions such as continuous wave transmission and continuous packet transmission can be enabled. See the DW1000 User Manual [3] for further details.

2.1.1 Crystal Oscillator Trim

DWM1001 modules are calibrated at production to minimise initial frequency error to reduce carrier frequency offset between modules and thus improve receiver sensitivity. The calibration carried out at module production will trim the initial frequency offset to less than 3 ppm, typically.

2.1.2 Transmitter Calibration

The DWM1001C is calibrated in module production, the calibrated values are permanently stored in the DW1000 OTP. This module is calibrated to meet the regulatory power spectral density requirement of less than -41.3 dBm/MHz.

2.1.3 Antenna Delay Calibration

In order to measure range accurately, precise calculation of timestamps is required. To do this the antenna delay must be known. The DWM1001 allows this delay to be calibrated and provides the facility to compensate for delays introduced by PCB, external components, antenna and internal DWM1001 delays.

The DWM1001's antenna delay was pre-calibrated for the RF configuration used within PANS. The antenna delay is stored in OTP memory.

To calibrate the antenna delay, range is measured at a known distance using two DWM1001 systems. Antenna delay is adjusted until the known distance and reported range agree.

Antenna delay calibration must be carried out as a once off measurement for each DWM1001 design implementation. If required, for greater accuracy, antenna delay calibration should be carried out on a per DWM1001 module basis, see DW1000 User Manual [3] for full details. Further details can be found in the Antenna Delay Application Note [8].

3 DWM1001 PIN CONNECTIONS

3.1 Pin Numbering

DWM1001 module pin assignments are as follows (viewed from top): -

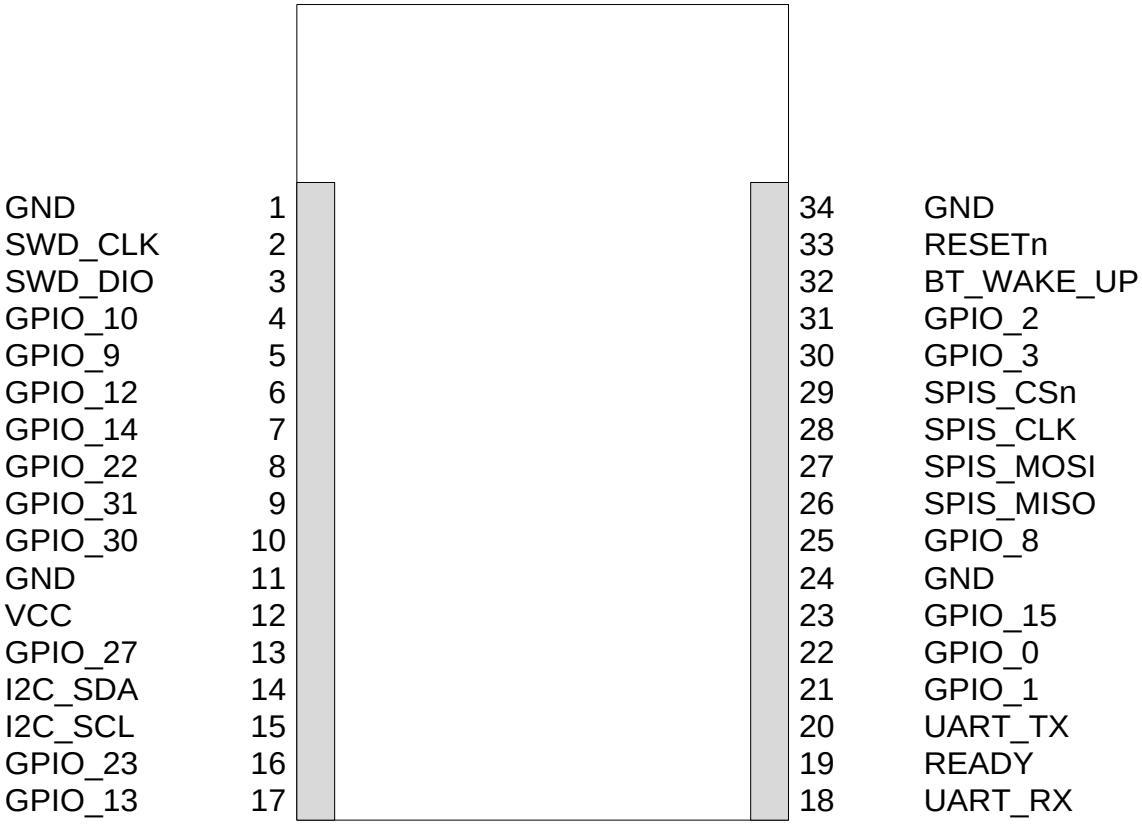


Figure 1: DWM1001 Pin Diagram

3.2 Pin Descriptions

Pin details are given in

Table 1: DWM1001 Pin functions

SIGNAL NAME	PIN	I/O (Default)	DESCRIPTION	REFERENCE (Pin designation)
Digital Interface				
SWD_CLK	2	DI	Serial wire debug clock input for debug and programming of Nordic Processor	[N] SWDCLK
SWD_DIO	3	DIO	Serial wire debug I/O for debug and programming of Nordic Processor	[N] SWDIO
GPIO_10	4	DIO	General purpose I/O pin.	[N] P0.10
GPIO_9	5	DIO	General purpose I/O pin.	[N] P0.9
GPIO_12	6	DIO	General purpose I/O pin.	[N] P0.12
GPIO_14	7	DIO	General purpose I/O pin.	[N] P0.14
GPIO_22	8	DIO	General purpose I/O pin.	[N] P0.22
GPIO_31	9	DIO	General purpose I/O pin. ADC function of nRF52	[N] P0.31
GPIO_30	10	DIO	General purpose I/O pin. ADC function of nRF52	[N] P0.30
GPIO_27	13	DIO	General purpose I/O pin.	[N] P0.27
I2C_SDA (Master)	14	DIO	Master I2C Data Line.	[N] P0.29

SIGNAL NAME	PIN	I/O (Default)	DESCRIPTION	REFERENCE (Pin designation)
I2C_SCL (Master)	15	DO	Master I2C Clock Line	[N] P0.28
GPIO_23	16	DIO	General purpose I/O pin.	[N] P0.23
GPIO_13	17	DIO	General purpose I/O pin.	[N] P0.13
UART_RX	18	DI	UART_RX	[N] P0.11
READY	19	DO	Generated interrupt from the device. Indicates events such as SPI data ready, or location data ready. See the function dwm_int_cfg() in the DWM1001 Firmware API Guide for details[5].	[N] P0.26
UART_TX	20	DO	UART_TX, This is also the ADC function of the nRF52	[N] P0.05
GPIO_1	21	DIO	General purpose I/O pin of the DW1000. It may be configured for use as a SFDLED driving pin that can be used to light a LED when SFD (Start Frame Delimiter) is found by the receiver. Refer to the DW1000 User Manual [1] for details of LED use.	[DW] GPIO1
GPIO_0	22	DIO	General purpose I/O pin of the DW1000. It may be configured for use as a RXOKLED driving pin that can be used to light a LED on reception of a good frame. Refer to the DW1000 User Manual [1] for details of LED use.	[DW] GPIO0
GPIO_15	23	DIO	General purpose I/O pin.	[N] P0.15
GPIO_8	25	DIO	General purpose I/O pin.	[N] P0.08
SPIS_MISO	26	DI	Configured as a SPI slave this pin is the SPI data output. Refer to Datasheet for more details [1].	[N] P0.07
SPIS_MOSI	27	DO	Configured as a SPI slave this pin is the SPI data input. Refer to Datasheet for more details [1].	[N] P0.06
SPIS_CLK	28	DI	Configured as a SPI slave this pin is the SPI clock. This is also the ADC function of the nRF52	[N] P0.04
SPIS_CSn	29	DI	Configured as a SPI slave this pin is the SPI chip select. This is an active low enable input. The high-to-low transition on SPICSn signals the start of a new SPI transaction. This is also the ADC function of the nRF52	[N] P0.03
GPIO_3	30	DO	This pin is configured for use as a TXLED driving pin that can be used to light a LED during transmit mode. Refer to the DW1000 User Manual [2] for details of LED use.	[DW] GPIO3
GPIO_2	31	DO	This pin is configured for use as a RXLED driving pin that can be used to light a LED during receive mode. Refer to the DW1000 User Manual [2] for details of LED use.	[DW] GPIO2
BT_WAKE_UP	32	DI	When this pin is asserted to its active low state the Bluetooth device will advertise its availability for 20 seconds by broadcasting advertising packets. This is also the ADC function of the nRF52.	[N] P0.02
RESETn	33	DI	Reset pin. Active Low Input.	[N] P0.21
Power Supplies				
VCC	12	P	External supply for the module. 2.8V - 3.6V	

SIGNAL NAME	PIN	I/O (Default)	DESCRIPTION	REFERENCE (Pin designation)
Ground				
GND	1, 11, 24, 34	G	Common ground.	

Table 2: Explanation of Abbreviations

ABBREVIATION	EXPLANATION
DI	Digital Input
DIO	Digital Input / Output
DO	Digital Output
G	Ground
P	Power Supply
N	nRF52832
DW	DW1000
Note: Any signal with the suffix 'n' indicates an active low signal.	

Table 3: Internal nRF52832 pins used and their function

nRF52832 Pin	Function
PO.19	DW_IRQ
PO.16	DW_SCK
PO.20	DW_MOSI
PO.18	DW_MISO
PO.17	DW_SPI_CS
PO.24	DW_RST
PO.25	ACC_IRQ
PO.29	I2C_SDA
PO.28	I2C_SCL

DW1000's GPIOs 5,6 control the DW1000 SPI mode configuration. Within the DWM1001 module, those GPIOs are unconnected and will be internally pulled down. Consequently, SPI will be set to mode 0. For more details, please refer to DW1000 data sheet [2].

Table 4: I2C slave devices address I2C

I2C slave device	Address
LIS2DH12	0X19

4 ELECTRICAL SPECIFICATIONS

The following tables give detailed specifications for the DWM1001 module. $T_{amb} = 25\ ^{\circ}C$ for all specifications given.

4.1 Nominal Operating Conditions

Table 5: DWM1001 Operating Conditions

Parameter	Min.	Typ.	Max.	Units	Condition/Note
Operating temperature	-40		+85	˚C	
Supply voltage VCC	2.8	3.3	3.6	V	Normal operation
Voltage on VDDIO for programming OTP	3.7	3.8	3.9	V	Note that for programming the OTP in the DWM1001 this supply is connected to the VDDIO test point which is underneath the PCB. (See Figure 6)

4.2 DC Characteristics

Table 6: DWM1001 DC Characteristics

Parameter	Min.	Typ.	Max.	Units	Condition/Note
Supply current in DEEP SLEEP mode		4		µA	All peripherals in lowest power consumption mode Achievable where RTC and accelerometer are disabled with custom firmware.
Supply current in DEEP SLEEP mode		12		µA	RTC and accelerometer operational, all other peripherals in lowest power consumption mode*
Supply current in IDLE mode		13		mA	MCU and DW1000 awake
TX peak current		111		mA	
TX mean current		82		mA	
RX peak current		154		mA	
RX mean current		134		mA	
Current in Bluetooth® discovery mode		6		mA	
Digital input voltage high	0.7 x VCC		VCC	V	
Digital input voltage low	GND		0.3 x VCC	V	
Digital output voltage high	0.7 x VCC		VCC	V	
Digital output voltage low	GND		0.3 x VCC	V	

* Using a ranging update rate of 1 Hz

4.3 Receiver AC Characteristics

Table 7: DWM1001 Receiver AC Characteristics

Parameter	Min.	Typ.	Max.	Units	Condition/Note
Frequency range	6240		6739.2	MHz	Centre Frequency 6489.6 MHz

4.4 Receiver Sensitivity Characteristics

T_{amb} = 25 ˚C, 20 byte payload. These sensitivity figures assume an antenna gain of 0 dBi and should be modified by the antenna characteristics, depending on the orientation of the DWM1001.

Table 8: DWM1001 Typical Receiver Sensitivity Characteristics

Packet Error Rate	Data Rate	Receiver Sensitivity	Units	Condition/Note		
1%	6.8 Mbps	-98*(-92)	dBm/500 MHz	Preamble 128	Carrier frequency offset ±10 ppm	All measurements performed on Channel 5, PRF 64 MHz
10%	6.8 Mbps	-99*(-93)	dBm/500 MHz	Preamble 128		

*equivalent sensitivity with Smart TX Power enabled. This is enabled in the onboard firmware.

4.5 Transmitter AC Characteristics

Table 9: DWM1001 Transmitter AC Characteristics

Parameter	Min.	Typ.	Max.	Units	Condition/Note
Frequency range	6240		6739.2	MHz	
Output power spectral density			-41.3*	dBm/MHz	See DW1000 Datasheet [1]
Output Channel Power		-17		dBm/500MHz	
Output power variation with temperature*	-1		+1	dB	Using on board compensation.

* When using the Decawave supplied embedded firmware for the DWM1001 module

4.5.1 Absolute Maximum Ratings

Table 10: DWM1001 Absolute Maximum Ratings

Parameter	Min.	Max.	Units
Supply voltage	2.8	3.9	V
Receiver power		0	dBm
Temperature - Storage temperature	-40	+85	˚C
Temperature – Operating temperature	-40	+85	˚C
ESD (Human Body Model)		2000	V
DWM1001 pins other than VCC, VDDIO and GND		3.6	Note that 3.6 V is the max voltage that may be applied to these pins

Stresses beyond those listed in this table may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions beyond those indicated in the operating conditions of the specification is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

5 TRANSMIT AND RECEIVE POWER CONSUMPTION

The following Figures give power profiles for the DWM1001 on a DWM1001-DEV PCB when used for Two Way Ranging, see Figure 2. Peak values are given.

Figure 2 shows an example of the power consumption of a DWM1001 tag running the factory loaded firmware.

The tag is in low-power mode, and two-way ranging with 3 anchors. The deep-sleep current occurs while the tag is sleeping with only the RTC and accelerometer active.

Once awake, the tag transmits at its allocated time in the TDMA-slotting and awaits the anchors responses. This can be observed as 1 transmission followed by 3 receives, repeated once. After this is completed, the tag spends some time computing its location, before returning to sleep. The total time awake is dependent on the number of anchors within range of the tag. For more details on the system operation, see the DWM1001 System Overview document[9].

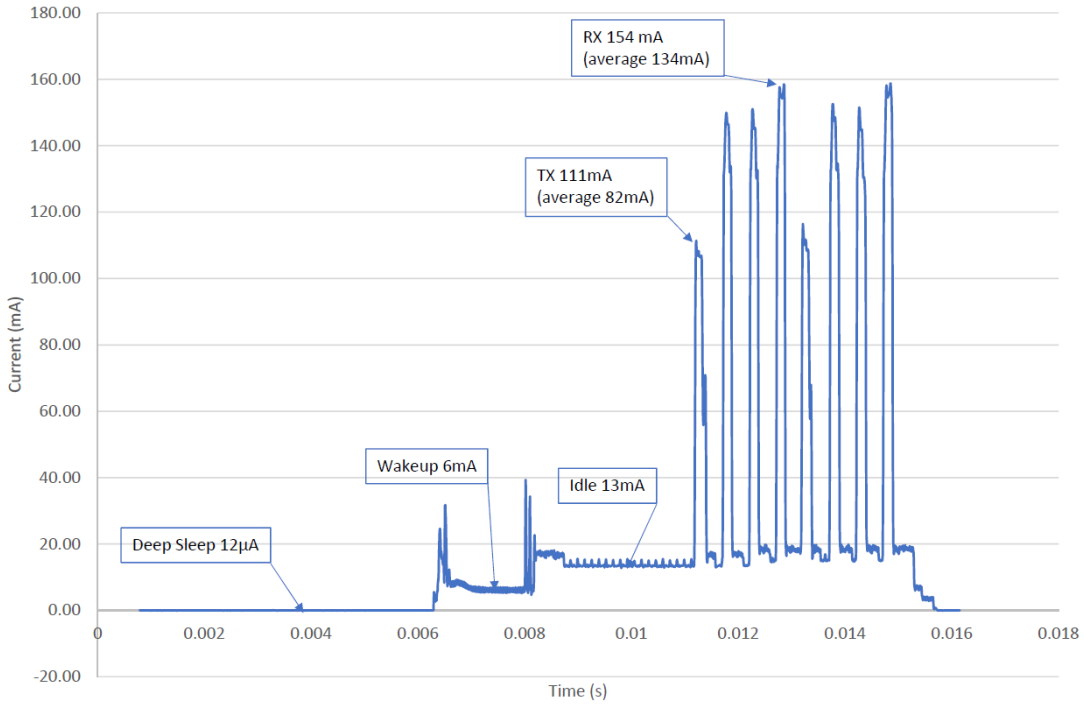


Figure 2: power consumption during Two Way Ranging

6 ANTENNA PERFORMANCE

This section details antenna radiation patterns for the DWM1001-Dev board. Figure 3 presents a view of the measurement planes considered in this document.

Table 11 shows antenna radiation patterns for the DWM1001 module mounted on the DWM1001-Dev board. Three planes in the spherical space about the centre of the board are measured, with theta and phi plots representing perpendicular polarisations.

The DWM1001 antenna is vertically polarised, meaning that the module is intended to be positioned vertically upright when used in an RTLS system. An omnidirectional radiation pattern is seen in the XZ plane when observed by another antenna which is also vertically polarised. This is shown in the XZ plane antenna patterns, where the vertically polarised plot, phi, has a circular, or omnidirectional shape.

If the antennas are oriented perpendicular relative to each other, then the polarisation changes. In this case, the horizontally polarised pattern, theta, applies and there are nulls at certain angles which can limit range and introduce location inaccuracy.

Table 12 presents the key characteristics of the DWM1001's antenna.

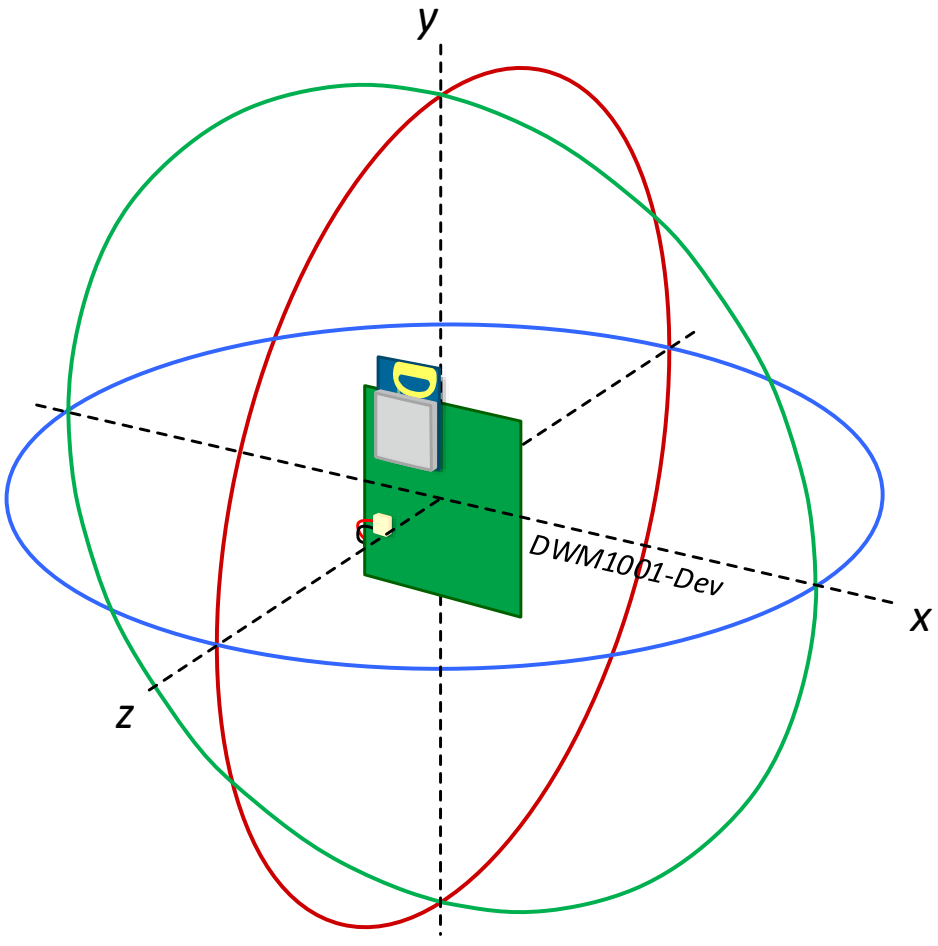


Figure 3. Antenna Radiation Pattern Planes

Table 11. Antenna Radiation Patterns

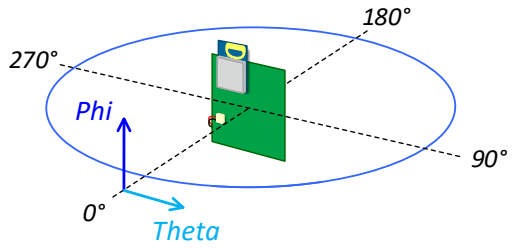
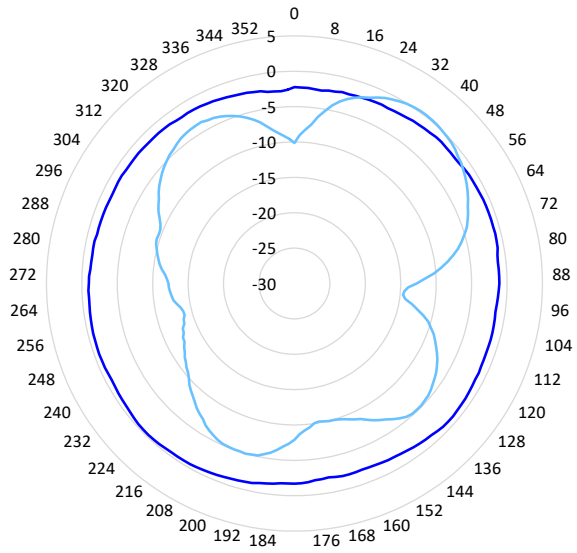
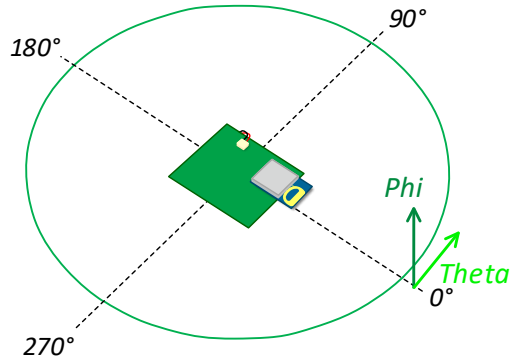
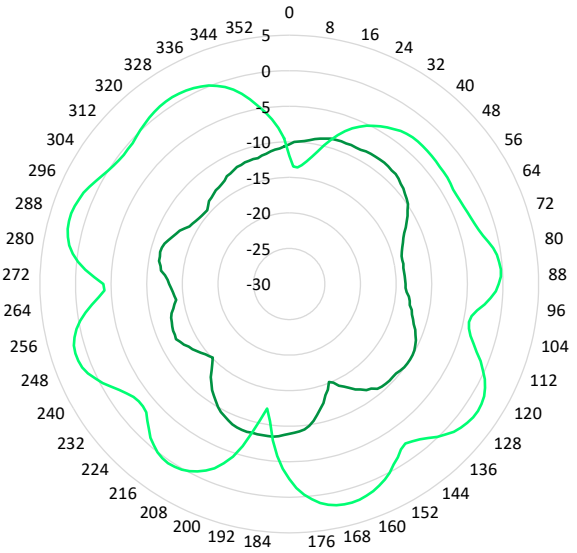
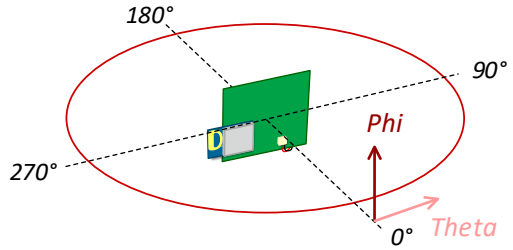
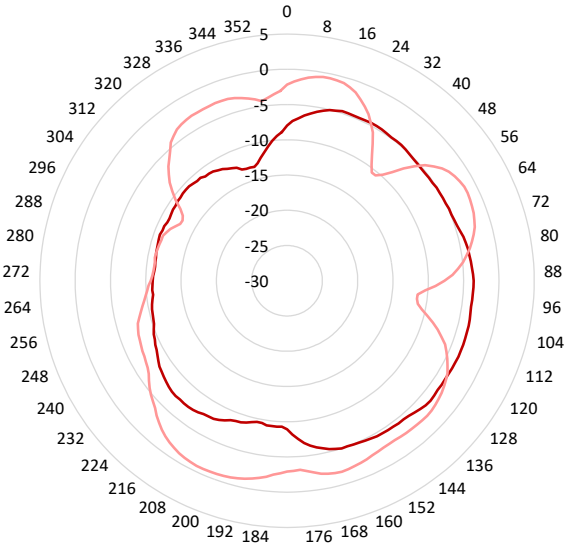
Planes:	Antenna Gain (dBi) vs. Angle (°)
XZ Plane: 	
XY Plane: 	
YZ Plane: 	

Table 12 : WB003 antenna characteristics

Antenna Model	Decawave WB003 – MiniHoe
Antenna Type	PCB Trace Monopole Antenna
Peak gain (Measured on DWM1001-DEV)	2.5 dBi
Frequency range	5500 ~ 7500 MHz

Please note the "Peak gain" and overall antenna performance are dependent on the carrier PCB geometry. When integrating the DWM1001 module to custom PCB, a variation in antenna performance may be observed.

7 APPLICATION INFORMATION

7.1 Application Board Layout Guidelines

When designing the PCB onto which the DWM1001 will be soldered, the proximity of the DWM1001 on-board antenna to metal and other non-RF transparent materials needs to considered carefully. Two suggested placement schemes are shown below.

For best RF performance, ground copper should be flooded in all areas of the application board, except in the areas marked "Keep-Out Area", where there should be no metal either side, above or below (e.g. do not place battery under antenna).

The two placement schemes in Figure 4 show an application board with no metallic material in the keep-out area. The diagram on the right is an application board with the antenna projecting off of the board so that the keep out area is in free-space. The diagram on the left shows an application board which does not have the module in free space but has the PCB copper removed on either side (and behind) the module antenna.

(Note: the rectangular area above the shield on the module is the antenna area)

It is also important to note that the ground plane on the application board affects the DWM1001 antenna radiation pattern. There must be a minimum spacing of 10 mm (d) without metal either side of the module antenna.

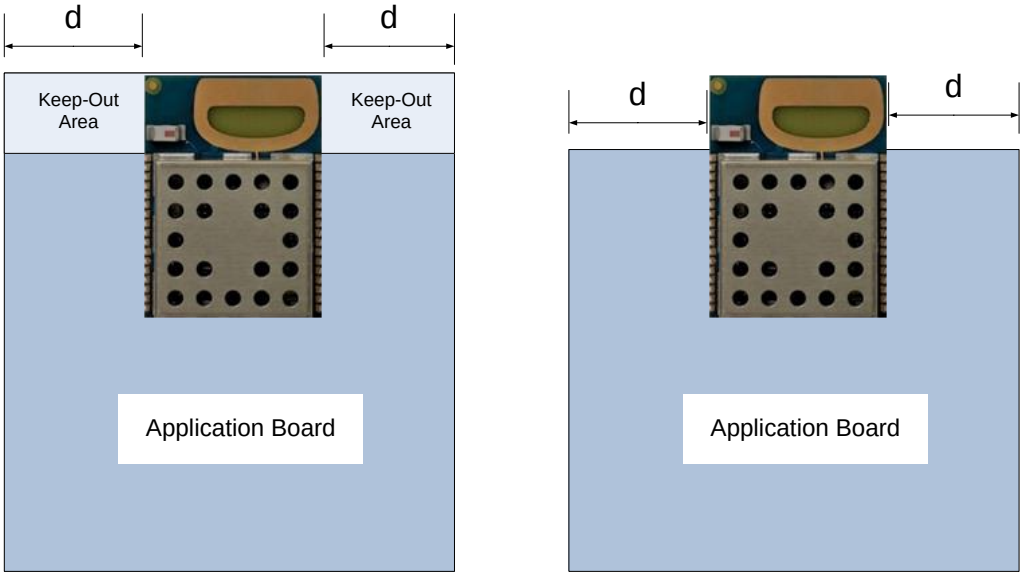


Figure 4: DWM1001 Application Board Keep-Out Areas

8 PACKAGE INFORMATION

8.1 Module Drawings

All measurements are given in millimetres.

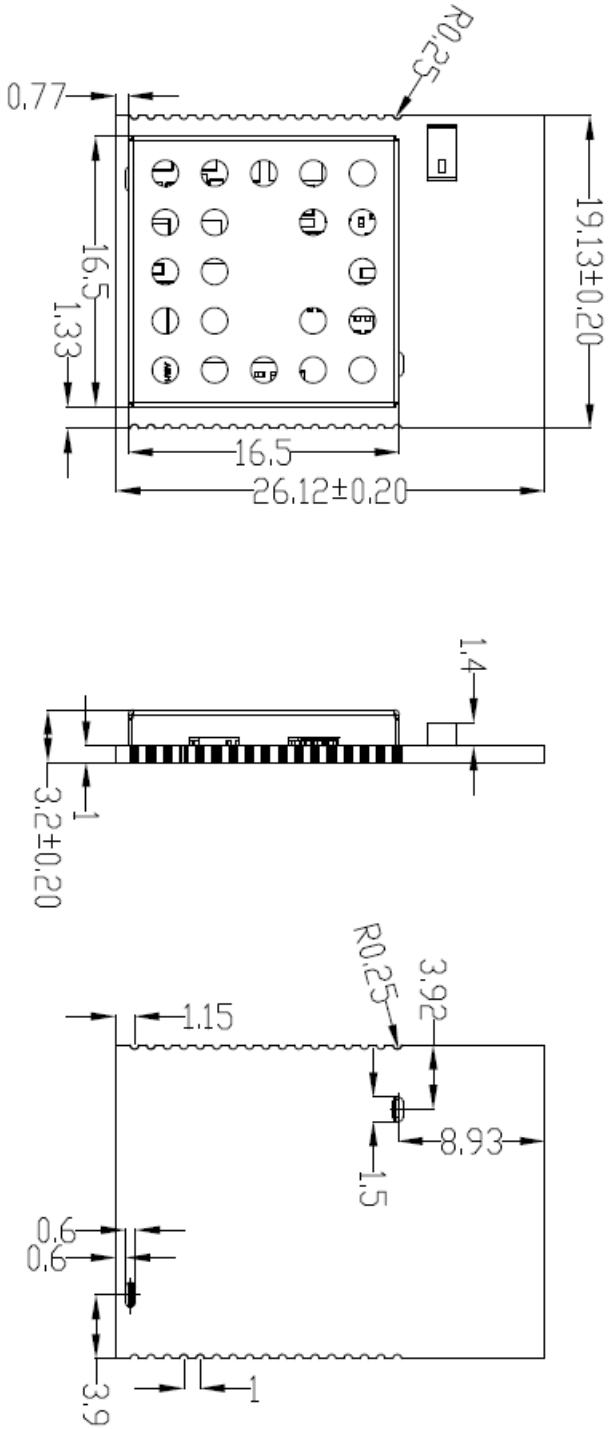


Figure 5: Module Package Size (units: mm)

8.2 Module Land Pattern

The diagram below shows the DWM1001 module land pattern.

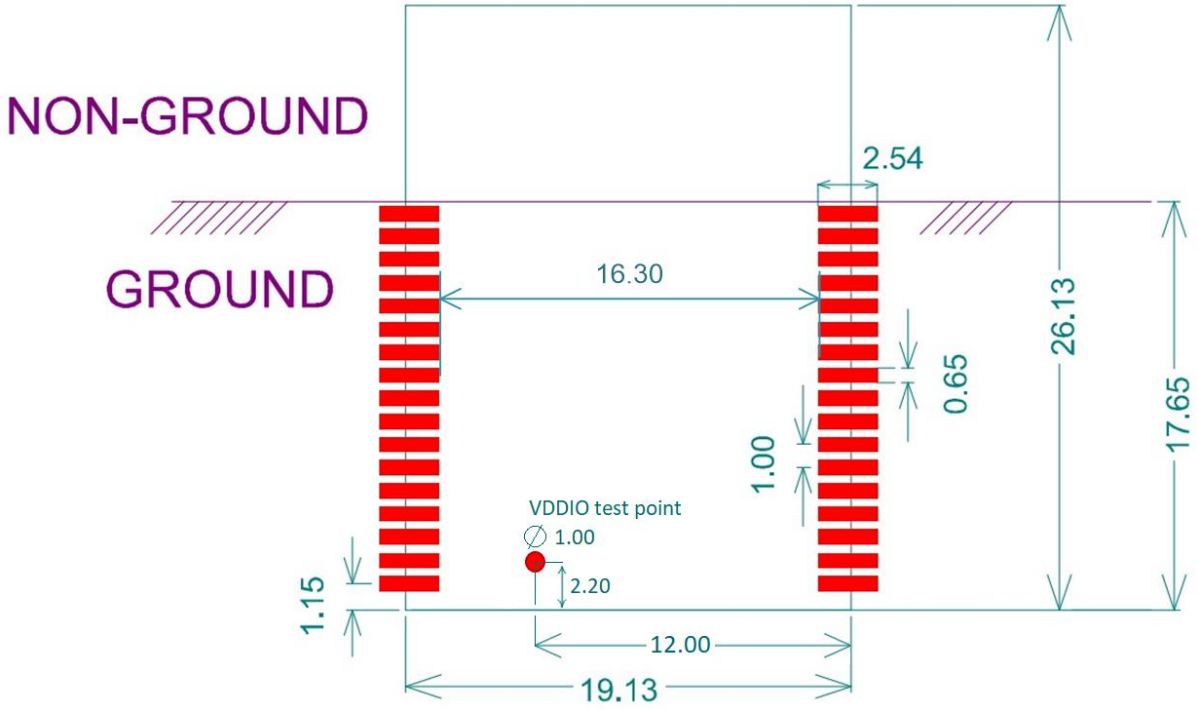


Figure 6: DWM1001 Module Land Pattern (units: mm)

8.3 Module Marking Information

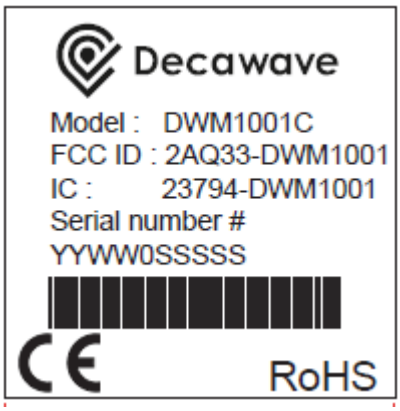
Each module has a label on the shield with a serial number in the following format:

YY WW 0 SSSSS

Where:

YY	indicates the year
WW	indicates the week of the year
0	indicates the DWM1001 module
SSSSS	indicates the module manufacturing number

Modules marked with DWM1001C are the certified version of the DWM1001.



Modules marked as DWM1001 are a non-certified version of the DWM1001

8.4 Module Solder Profile

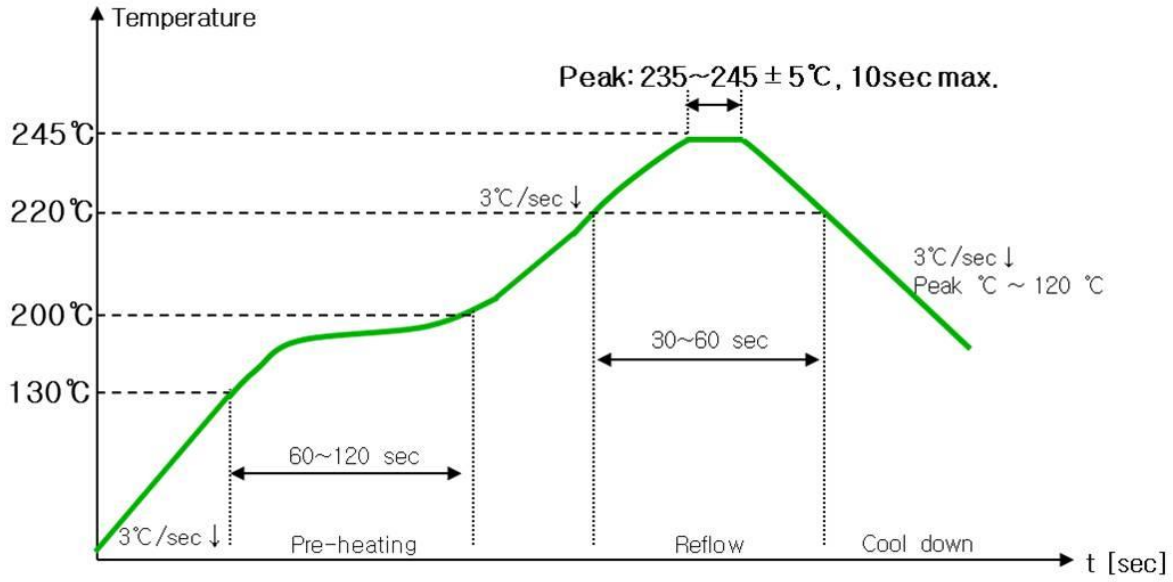


Figure 7: DWM1001 Module Solder Profile

9 ORDERING INFORMATION

9.1 Tape and Reel Information

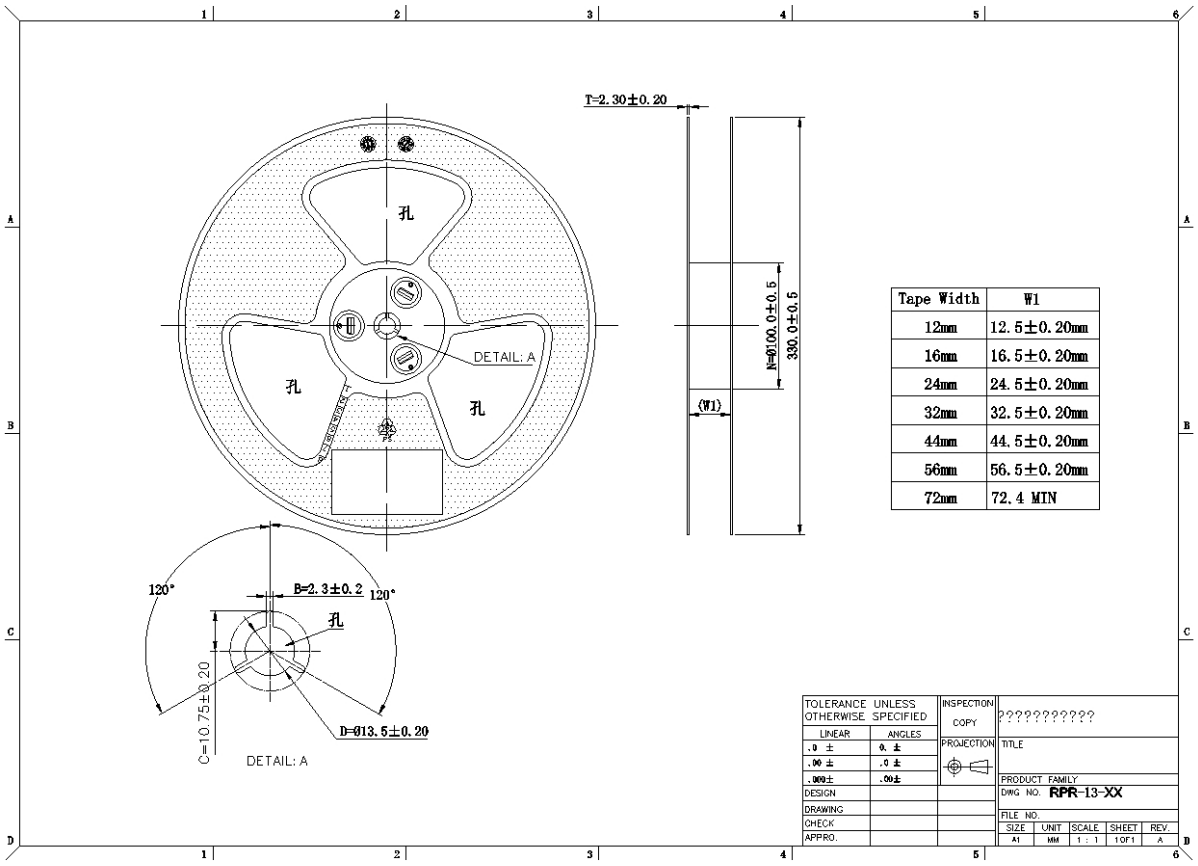


Figure 8: DWM1001 Tape and Reel Dimensions

10 REGULATORY INFORMATION

Model Number: DWM1001C FCC ID: 2AQ33-DWM1001 IC: 23794-DWM1001

The information below is valid for the DWM1001C module only.

10.1 Agency Certifications

10.1.1 United States (FCC)

This device complies with Part 15 of the FCC Rules:

Operation is subject to the following conditions:
1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation

Changes and Modifications not expressly approved by Decawave Ltd. can void your authority to operate this equipment under Federal Communications Commission rules.

Warning: Changes or modifications to this unit not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

To fulfil FCC Certification requirements, an OEM manufacturer must comply with the following regulations:

1. The DWM1001C modular transmitter must be labelled with its own FCC ID number, and, if the FCC ID is not visible when the module is installed inside another device, then the outside of the device into which the module is installed must also display a label referring to the enclosed module. This exterior label can use wording such as the following:

IMPORTANT: Contains FCC ID: 2AQ33-DWM1001. This equipment complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation (FCC 15.19).

10.1.2 Radio and Television Interference

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.

- Increase the separation between the equipment and the receiver.

- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

- Consult the dealer or an experienced radio/TV technician for help.

You may also find helpful the following booklet, prepared by the FCC: "How to Identify and Resolve Radio-TV Interference Problems." This booklet is available from the U.S. Government Printing Office, Washington D.C. 20402.

10.1.3 European Union (ETSI)

The DWM1001C Module has been certified for use in European Union and EEA countries and complies with ETSI EN 302 065-2 (V2.1.1 - November 2016). If these modules are incorporated into a product, the manufacturer must assess the compliance of the final product with the Radio Equipment Directive (and potentially other applicable Directives depending on the product category). A Declaration of Conformity must be issued as prescribed in Article 18 and Annex VI of the Radio Equipment Directive.

Furthermore, the manufacturer must maintain a copy of the modules' documentation and ensure the final product does not exceed the specified power ratings, antenna specifications, and/or installation requirements as specified in the user manual. If any of these specifications are exceeded in the final product, the manufacturer must perform an own, complete conformity assessment.

IMPORTANT:
The CE marking shall also be affixed visibly and legibly to the packaging. If the nature of the radio equipment does not allow a marking of at least 5 mm, the manufacturer may affix a CE marking that is smaller than 5 mm to the product under the condition that it remains visible and legible. If it not possible or not warranted on account of the nature of the radio equipment to affix a CE marking on the product, the manufacturer may affix it visibly and legibly only to the packaging. The CE mark shall consist of the initials "CE" taking the following form:

CE

10.1.4 Industry Canada (IC) Compliance Statements

This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

CAUTION: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

The OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed (for example, digital device emissions, PC peripheral requirements, etc.). This Module is labelled with its own IC ID. If the IC ID Certification Number is not visible while installed inside another device, then the device should display the label on it referring the enclosed module. In that case, the final end product must be labelled in a visible area with the following:

"Contains Transmitter Module IC: 23794-DWM1001"
OR
"Contains IC: 23794-DWM1001"

Ce module est étiqueté avec son propre ID IC. Si le numéro de certification IC ID n'est pas visible lorsqu'il est installé à l'intérieur d'un autre appareil, l'appareil doit afficher l'étiquette sur le module de référence ci-joint. Dans ce cas, le produit final doit être étiqueté dans un endroit visible par le texte suivant:

"Contains Transmitter Module IC: 23794-DWM1001"
OR
"Contains IC: 23794-DWM1001"

10.2 Regulatory Module Integration Instructions

This module has been granted modular approval for mobile applications. OEM integrators for host products may use the module in their final products without additional FCC / ISED (Innovation, Science and Economic Development Canada) certification if they meet the following conditions. Otherwise, additional FCC / ISED approvals must be obtained.

- The host product with the module installed must be evaluated for simultaneous transmission requirements.
- The user's manual for the host product must clearly indicate the operating requirements and conditions that must be observed to ensure compliance with current FCC / IC RF exposure guidelines.
- To comply with FCC / ISED regulations limiting both maximum RF output power and human exposure to RF radiation, the maximum antenna gain including cable loss in a mobile-only exposure condition must not exceed:

  Peak UWB Antenna Gain: 2.5 dBi
  Peak BLE Antenna Gain: 0.5 dBi

- A label must be affixed to the outside of the host product with the following statements:

  This device contains FCC ID: 2AQ33-DWM1001
  This equipment contains equipment certified under IC: 23794-DWM1001

The final host / module combination may also need to be evaluated against the FCC Part 15B criteria for unintentional radiators in order to be properly authorized for operation as a Part 15 digital device.
If the final host / module combination is intended for use as a portable device (see classifications below) the host manufacturer is responsible for separate approvals for the SAR requirements from FCC Part 2.1093 and RSS-102.

10.3 Device Classifications

Since host devices vary widely with design features and configurations module integrators shall follow the guidelines below regarding device classification and simultaneous transmission, and seek guidance from their preferred regulatory test lab to determine how regulatory guidelines will impact the device compliance. Proactive management of the regulatory process will minimize unexpected schedule delays and costs due to unplanned testing activities.

The module integrator must determine the minimum distance required between their host device and the user's body. The FCC provides device classification definitions to assist in making the correct determination. Note that these classifications are guidelines only; strict adherence to a device classification may not satisfy the regulatory requirement as near-body device design details may vary widely. Your preferred test lab will be able to assist in determining the appropriate device category for your host product and if a KDB or PBA must be submitted to the FCC.

Note, the module you are using has been granted modular approval for mobile applications. Portable applications may require further RF exposure (SAR) evaluations. It is also likely that the host / module combination will need to undergo testing for FCC Part 15 regardless of the device classification. Your preferred test lab will be able to assist in determining the exact tests which are required on the host / module combination.

10.4 FCC Definitions

Portable: (§2.1093) — A portable device is defined as a transmitting device designed to be used so that the radiating structure(s) of the device is / are within 20 centimeters of the body of the user.

Mobile: (§2.1091) (b) — A mobile device is defined as a transmitting device designed to be used in other than fixed locations and to generally be used in such a way that a separation distance of at least 20 centimeters is normally maintained between the transmitter's radiating structure(s) and the body of the user or nearby persons. Per §2.1091d(d)(4) In some cases (for example, modular or desktop transmitters), the potential conditions of use of a device may not allow easy classification of that device as either Mobile or Portable. In these cases, applicants are responsible for determining minimum distances for compliance for the intended use and installation of the device based on evaluation of either specific absorption rate (SAR), field strength, or power density, whichever is most appropriate.

10.5 Simultaneous Transmission Evaluation

This module has not been evaluated or approved for simultaneous transmission as it is impossible to determine the exact multi-transmission scenario that a host manufacturer may choose. Any simultaneous transmission condition established through module integration into a host product must be evaluated per the requirements in KDB447498D01(8) and KDB616217D01,D03 (for laptop, notebook, netbook, and tablet applications).

These requirements include, but are not limited to:

- Transmitters and modules certified for mobile or portable exposure conditions can be incorporated in mobile host devices without further testing or certification when:
- The closest separation among all simultaneous transmitting antennas is >20 cm,

Or

- Antenna separation distance and MPE compliance requirements for ALL simultaneous transmitting antennas have been specified in the application filing of at least one of the certified transmitters within the host device. In addition, when transmitters certified for portable use are incorporated in a mobile host device, the antenna(s) must be >5 cm from all other simultaneous transmitting antennas.
- All antennas in the final product must be at least 20 cm from users and nearby persons.

10.6 Operating Requirements and Conditions

The design of DWM1001 complies with U.S. Federal Communications Commission (FCC) guidelines respecting safety levels of radio frequency (RF) exposure for Mobile or Portable devices.

FCC ID:
This product contains FCC ID: 2AQ33-DWM1001

Note: In the case where the Host / Module combination has been re-certified the FCC ID shall appear in the product manual as follows:

FCC ID: 2AQ33-DWM1001

10.7 Mobile Device RF Exposure Statement

RF Exposure - This device is only authorized for use in a mobile application. At least 20 cm of separation distance between the DWM1001 device and the user's body must be maintained at all times.

Caution Statement for Modifications:
CAUTION: Any changes or modifications not expressly approved by Decawave Ltd could void the user’s authority to operate the equipment.

11 GLOSSARY

Table 13: Glossary of Terms

Abbreviation	Full Title	Explanation
EIRP	Equivalent Isotropically Radiated Power	The amount of power that a theoretical isotropic antenna (which evenly distributes power in all directions) would emit to produce the peak power density observed in the direction of maximum gain of the antenna being used
ETSI	European Telecommunication Standards Institute	Regulatory body in the EU charged with the management of the radio spectrum and the setting of regulations for devices that use it
FCC	Federal Communications Commission	Regulatory body in the USA charged with the management of the radio spectrum and the setting of regulations for devices that use it
GPIO	General Purpose Input / Output	Pin of an IC that can be configured as an input or output under software control and has no specifically identified function
IEEE	Institute of Electrical and Electronic Engineers	The world's largest technical professional society. It is designed to serve professionals involved in all aspects of the electrical, electronic and computing fields and related areas of science and technology
LIFS	Long Inter-Frame Spacing	Defined in the context of the IEEE 802.15.4-2011 [7] standard
LNA	Low Noise Amplifier	Circuit normally found at the front-end of a radio receiver designed to amplify very low level signals while keeping any added noise to as low a level as possible
LOS	Line of Sight	Physical radio channel configuration in which there is a direct line of sight between the transmitter and the receiver
NLOS	Non Line of Sight	Physical radio channel configuration in which there is no direct line of sight between the transmitter and the receiver
PGA	Programmable Gain Amplifier	Amplifier whose gain can be set / changed via a control mechanism usually by changing register values
PLL	Phase Locked Loop	Circuit designed to generate a signal at a particular frequency whose phase is related to an incoming "reference" signal.
PPM	Parts Per Million	Used to quantify very small relative proportions. Just as 1% is one out of a hundred, 1 ppm is one part in a million
RF	Radio Frequency	Generally used to refer to signals in the range of 3 kHz to 300 GHz. In the context of a radio receiver, the term is generally used to refer to circuits in a receiver before down-conversion takes place and in a transmitter after up-conversion takes place
RTLS	Real Time Location System	System intended to provide information on the location of various items in real-time.
SFD	Start of Frame Delimiter	Defined in the context of the IEEE 802.15.4-2011 [7] standard.
SPI	Serial Peripheral Interface	An industry standard method for interfacing between IC's using a synchronous serial scheme first introduced by Motorola
TCXO	Temperature Controlled Crystal Oscillator	A crystal oscillator whose output frequency is very accurately maintained at its specified value over its specified temperature range of operation.
TWR	Two Way Ranging	Method of measuring the physical distance between two radio units by exchanging messages between the units and noting the times of transmission and reception. Refer to Decawave's website for further information
TDOA	Time Difference of Arrival	Method of deriving information on the location of a transmitter. The time of arrival of a transmission at two physically different locations whose clocks are synchronized is noted and the difference in the arrival times provides information on the location of the transmitter. A number of such TDOA measurements at different locations can be used to uniquely determine the position of the transmitter. Refer to Decawave's website for further information.
UWB	Ultra Wideband	A radio scheme employing channel bandwidths of, or in excess of, 500MHz
WSN	Wireless Sensor Network	A network of wireless nodes intended to enable the monitoring and control of the physical environment
BLE	Bluetooth Low Energy.	A low power means of data communication.

12 REFERENCES

[1] nRF52832 Product Specification v1.3 www.nordicsemi.com
[2] Decawave DW1000 Datasheet www.decawave.com
[3] Decawave DW1000 User Manual www.decawave.com
[4] STMicroelectronics LIS2DH12TR www.st.com
[5] DWM1001 Firmware API Guide
[6] DWM1001 Firmware User Guide
[7] IEEE802.15.4-2011 or "IEEE Std 802.15.4™-2011" (Revision of IEEE Std 802.15.4-2006). IEEE Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Computer Society Sponsored by the LAN/MAN Standards Committee. Available from http://standards.ieee.org/
[8] APS014 Antenna Delay Calibration of DW1000-based products and systems
[9] DWM1001 System Overview

13 DOCUMENT HISTORY

Table 14: Document History

Revision	Date	Description
1.4	19/12/2018	Update
1.3	22/10/2018	Update
1.2	07/08/2018	Update
1.1	27/02/18	Core update

14 MAJOR CHANGES

Revision 1.1

Page	Change Description
All	Update of version number to 1.10
9	New table detailing internal connections between nRF52 and DW1000
9	Adding I2C slave devices address
9	Specifying that nrF52 to DW1000 SPI interface mode is 0
14,15	New details on Antenna Radiation pattern
18	Adding accurate position of VDDIO test point on figure 6

Revision 1.2

Page	Change Description
All	Logo Change

Revision 1.3

Page	Change Description
1	Key benefits update

Revision 1.4

Page	Change Description
All	DWM1001C and regulation information added

15 FURTHER INFORMATION

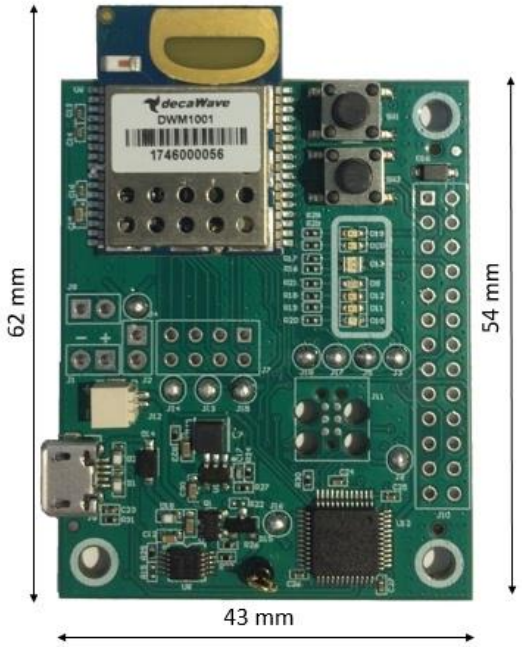
Decawave develops semiconductors solutions, software, modules, reference designs - that enable real-time, ultra-accurate, ultra-reliable local area micro-location services. Decawave's technology enables an entirely new class of easy to implement, highly secure, intelligent location functionality and services for IoT and smart consumer products and applications.

For further information on this or any other Decawave product, please refer to our website www.decawave.com.

Product Overview: DWM1001-DEV

DWM1001 Module Development Board

- Plug-and-Play Development Board for evaluating the performance of the Decawave DWM1001 module
- Easily assemble a fully wireless RTLS system, including anchors, tags & gateways, without designing any hardware or writing a single line of code – and quickly progress into developing your application



Key Features and Benefits

- DWM1001 module mounted (See DWM1001 data sheet for details)
- USB connection for reprogramming, debug & power supply
- On board JLINK
- External API via SPI, UART & BLE for configuration & control
- 26-pin Raspberry Pi compatible header
- Reset and user-defined buttons and LEDs
- Battery charging circuit
- Allows access to DWM1001 pins (castellation) via on board headers

Table of Contents

List of Figures

List of Tables

DOCUMENT INFORMATION

Disclaimer

Decawave reserves the right to change product specifications without notice. As far as possible changes to functionality and specifications will be issued in product specific errata sheets or in new versions of this document. Customers are advised to check with Decawave for the most recent updates on this product.

The DWM1001 module mounted on the DWM1001-DEV PCB is pre-loaded with firmware, please refer to the "DWM1001 Firmware User Guide" for disclaimer and license terms.

LIFE SUPPORT POLICY

Decawave products are not authorized for use in safety-critical applications (such as life support) where a failure of the Decawave product would reasonably be expected to cause severe personal injury or death. Decawave customers using or selling Decawave products in such a manner do so entirely at their own risk and agree to fully indemnify Decawave and its representatives against any damages arising out of the use of Decawave products in such safety-critical applications.

Caution! ESD sensitive device. Precaution should be used when handling the device in order to prevent permanent damage.

REGULATORY APPROVALS

The DWM1001, as supplied from Decawave, has not been certified for use in any particular geographic region by the appropriate regulatory body governing radio emissions in that region although it is capable of such certification depending on the region and the manner in which it is used.

All products developed by the user incorporating the DWM1001 must be approved by the relevant authority governing radio emissions in any given jurisdiction prior to the marketing or sale of such products in that jurisdiction and user bears all responsibility for obtaining such approval as needed from the appropriate authorities.

1 OVERVIEW

This document gives technical details of the DWM1001 module development board, called the DWM1001-DEV. All the functions of the DWM1001 module can be exercised with this board. Figure 1 gives an overview of the main components of the module development board.

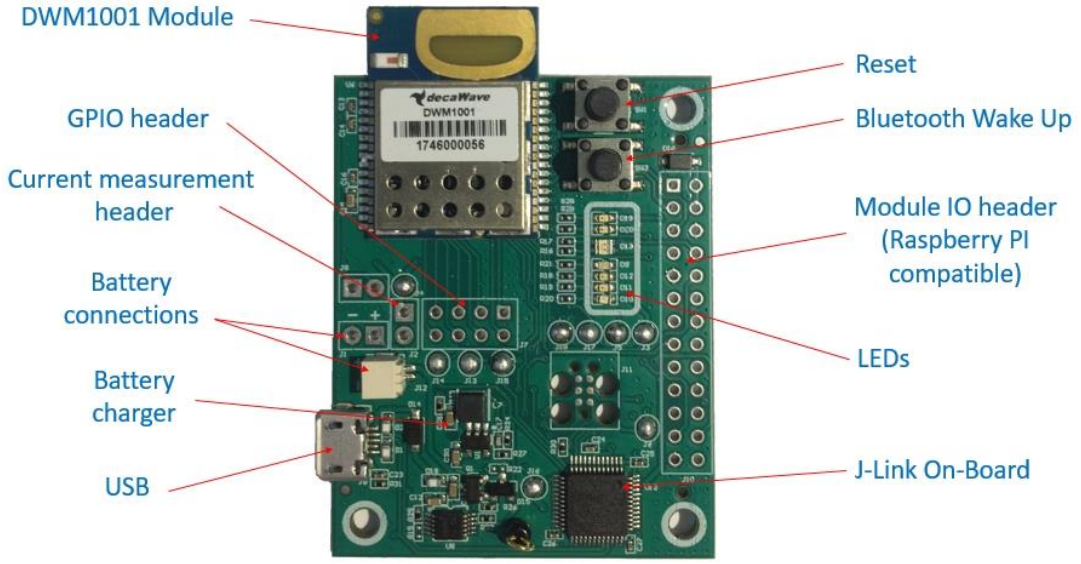


Figure 1 The main components of the module development board are shown

The module development board can be used to create an Anchor or a Tag for an RTLS system. This is shown in Figure 2. It can also be combined with a Raspberry Pi to create a gateway device. Figure 3 shows the configuration of an RTLS system where the module development board can be an Anchor, Tag or Gateway device.

Figure 2: The module development board can be used to create an Anchor, a Tag or a Gateway

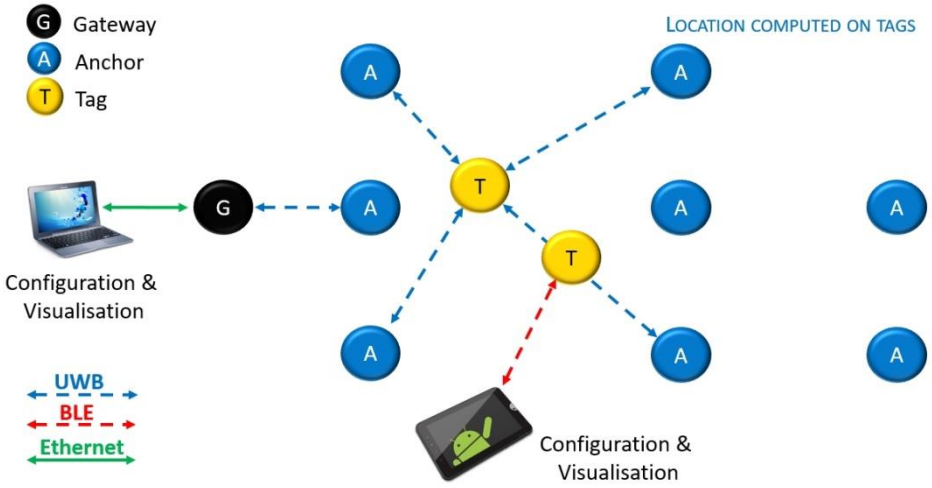


Figure 3: Module development board used as an Anchor, Tag or Gateway device, in an RTLS system

1.1 Suitable Power Supply Options

The module development board has a voltage supply requirement of 3.6V to 5.5V.
The module development board can be powered from three different sources. Details are given in Table 1.

Table 1: Possible sources of power for the module development board

Power Source	Voltage level	Current level (Recommended)	Notes
USB Connection	+5V	500mA	The board requires a connection to a high power USB connection. Check that it can supply at least 500mA.
Battery	3.6V - 5.5V	500mA	Any battery that meets the 3.6V to 5.5V voltage supply will suffice.
Raspberry Pi Power	+5V	500mA	

1.2 Block Diagram of the development board

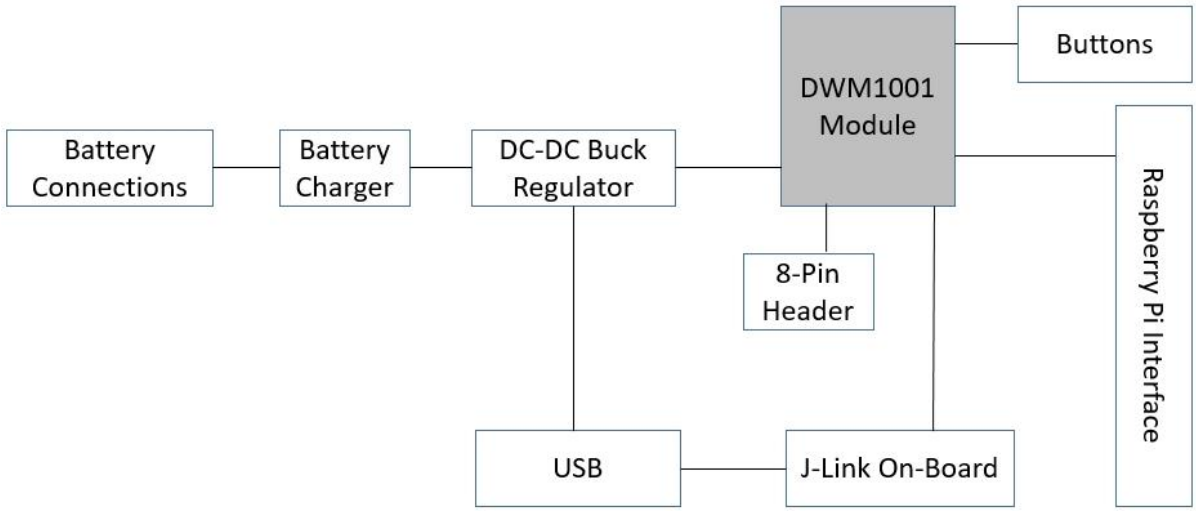


Figure 4: The main sections of the DWM1001 Module Development Board

Figure 4 shows the main sections of the Module Development Board. A brief overview of these sections is given below with further details given in later sections of this document.

The DWM1001 module is based on Decawave's DW1000 Ultra Wideband (UWB) transceiver IC, which is an IEEE 802.15.4-2011 UWB implementation. It integrates UWB and Bluetooth antenna, all RF circuitry, Nordic Semiconductor nRF52832 and motion sensor [1].

The USB connection can provide power to the Module Development Board and also allows for the capability to flash the DWM1001 module and furthermore to debug software running on the DWM1001 module.

The Power Supply takes its input from USB or from a Battery or from a connected Raspberry Pi. It powers the module and the other devices on the Module Development Board. It can also charge a connected battery when powered by USB or the Raspberry Pi.

Two buttons and a number of LEDs are provided for end user applications. A header to interface to the Raspberry Pi is also provided.

2 DWM1001 MODULE

Figure 5 shows a block diagram of the module. All major sections of the module are shown, along with the source of signals coming to the module's pins.

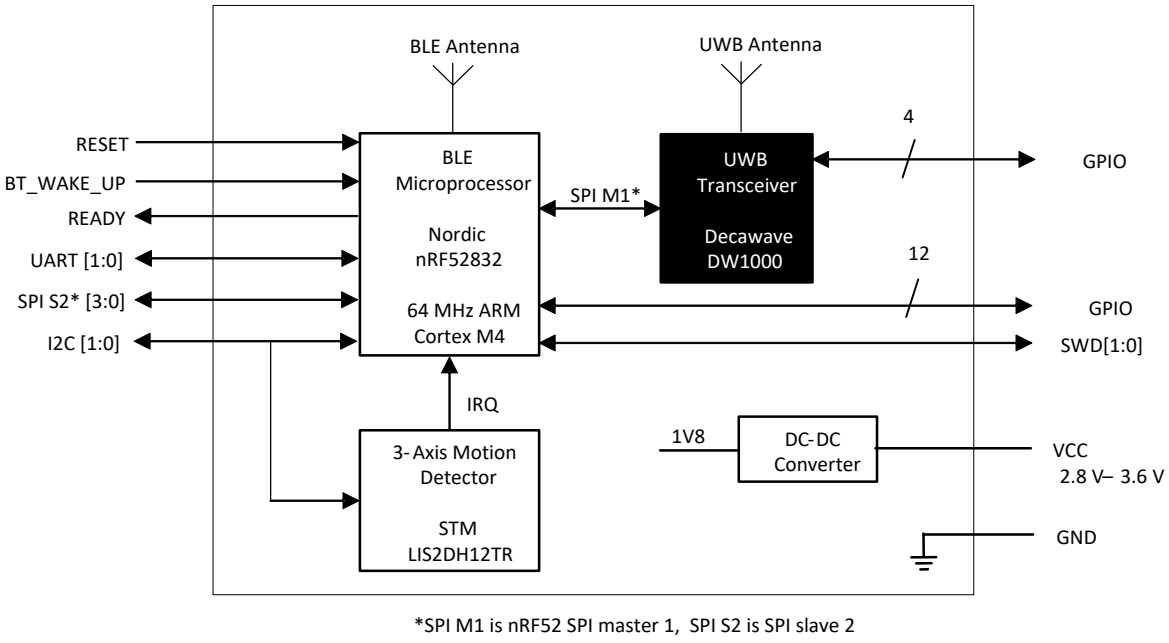


Figure 5: Block diagram of DWM1001 module

3 RASPBERRY PI INTERFACE

3.1 Means of connection

There are a number of types of Raspberry Pi. The preferred options are the A and B variants. To use the A variant you will require a ribbon cable to connect to the Module Development Board. Figure 6 below shows a Raspberry Pi Model A with the ribbon cable connected.

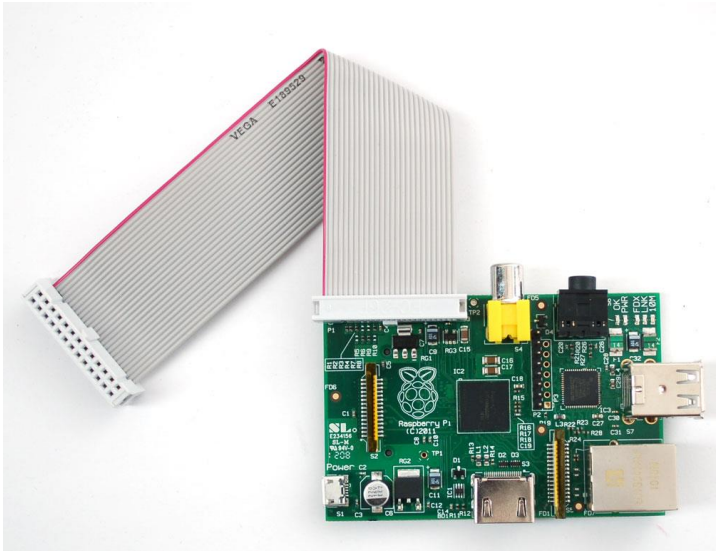


Figure 6: Raspberry Pi Model A Type with ribbon cable for connection to the Module Development Board

As an alternative to the ribbon cable it is possible to get header connectors with extra long pins. Such a connector is shown in Figure 7. One supplier of these connectors is https://www.modmypi.com. Search for part number MMP-0275.

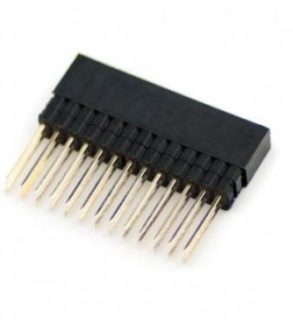


Figure 7: Header connector with extended pin lengths

DWM1001 Module Pin to Raspberry Pi Connector MappingTable 2 below gives the connection details between the Raspberry Pi connector and the DWM1001 module. See the DWM1001 datasheet[1] and the schematic for the DWM1001 Module Development Board at the end of this document.

Table 2: Connections between Raspberry Pi and DWM1001 module

Module Development Board RPi connector		**Module Pin Number (and Name) from DWM1001 Module Data Sheet**
Pin Number	Schematic Net Name	
3	SDA_RPI	Pin 23 (GPIO_15)
5	SCL_RPI	Pin 25 (GPIO_8)
9	GND	GND
15	GPIO_RPI	Pin 19 (READY)
19	SPI1_MOSI	Pin 27 (SPIS_MOSI)
21	SPI1_MISO	Pin 26 (SPIS_MISO)
23	SPI1_CLK	Pin 25 (GPIO_8)
25	GND	GND
2	VRPI	Provides input power to Module Development Board. (Not connected directly to module)
4	VRPI	
6	GND	GND
8	TXD	Pin 18 (UART_RX)
10	RXD_RPI/RXD	Pin 20 (UART_TX)
12	RESET	Pin 33 (RESETn)
14	GND	GND
20	GND	GND
24	CS_RPI	Pin29 (SPIS_CSn)

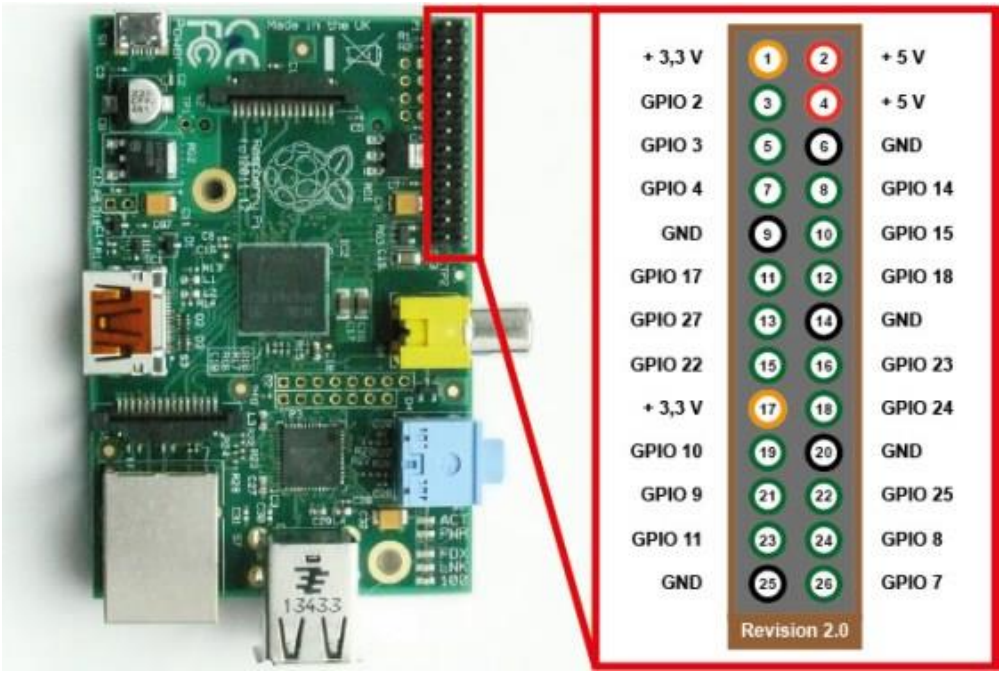


Figure 8: Pin designations of the Raspberry Pi Model A variant.

Figure 8 shows the pin designations for the model A Raspberry Pi variant. The Raspberry Pi Model B variant has a larger header connector with 40 pins. Figure 9 below shows its header and connector pin designations.

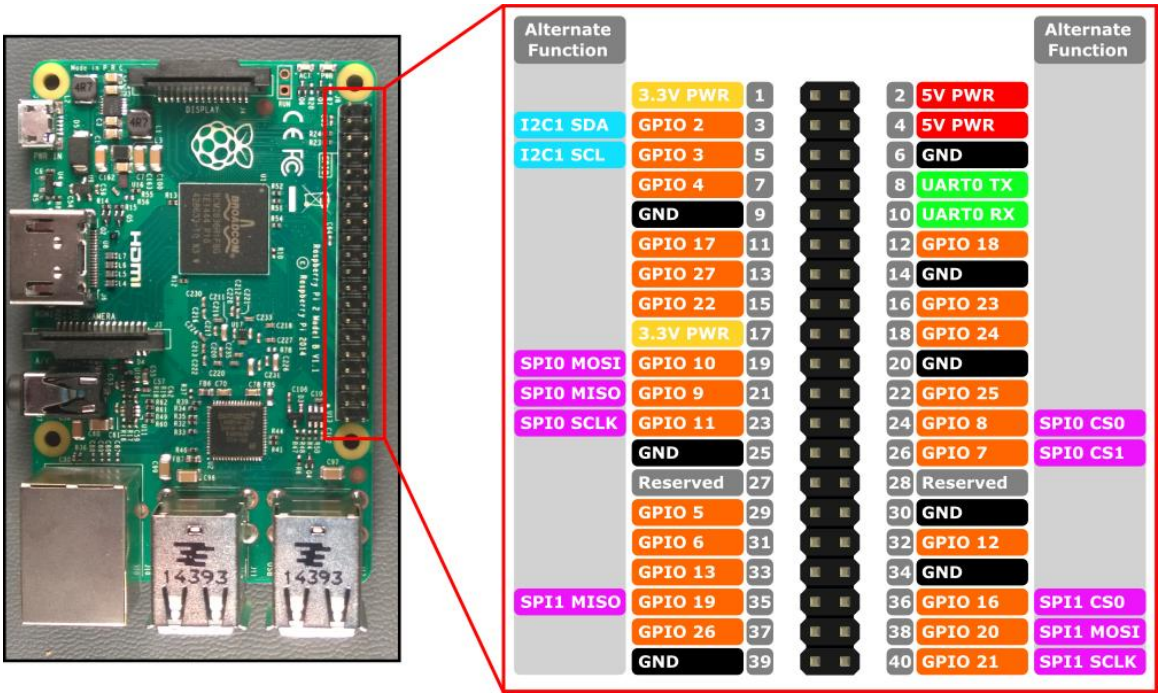


Figure 9: Pin designations of the Raspberry Pi Model B variant.

4 THE DEVELOPMENT BOARD LEDS

The Development board has a number of LEDs for indication purposes. They give useful indication of a number of events within the system. Figure 10 shows the LEDs and their use with the pre programmed firmware. Table 3 gives further specific details on D9, D10 and D11.

Figure 10: Front View of the DWM1001-DEV Module Development Board

Note (1): See DW1000 IC User Manual for description of D13 TX/RX LEDs
Note (2): F/W function of LEDs D9, D11, D10 is shown in the table below

Table 3: The Firmware Indication LEDs

GREEN D9	RED D11	BLUE D10	Function	Description
Blink Fast	Blink Fast	Blink Fast	Bootloader Active	LEDs blink twice
Blink Fast	Blink Fast	Blink Fast	Firmware Update in Progress	LEDs alternate between Blue and Green/Red
On	On		Mode Status	UWB Passive Mode
On	On			UWB Off Mode
Blink Fast	Blink Fast			Anchor Node (no UWB signal detected for more than 8 s)
Blink Slow	Blink Slow			Tag Node (no UWB signal detected for more than 8 s)
On				Tag Low Power Mode: ON
Off				Tag Low Power Mode: SLEEP
On			MAC Status	Connected anchor or tag
Blink Slow				Connected anchor initiator
Blink Fast				UWB communication detected
Off				Low Power node: no signal detected for more than 6 s
-		On	Bluetooth Status	Bluetooth Connected
-		Off		Bluetooth Disconnected
-	On		Data / Measurement Status	UWB TX/RX Active
-	Off			Idle

5 ON BOARD JLINK

The processor on the Module Development Board provides USB to SWD (Serial Wire Debug) conversion to allow programming and debug of software on the DWM1001 module. Figure 11 below shows the relevant sections on the Module Development Board.

Serial Wire Debug is a replacement for the more traditional 5-pin JTAG port. It uses a clock (SWDCLK) and a Single bi-directional data pin (SWDIO), providing all the normal JTAG debug and test functionality. SWDIO and SWCLK are overlaid on the TMS and TCK pins. In order to communicate with a SWD device, J-Link sends out data on SWDIO, synchronous to the SWCLK. With every rising edge of SWCLK, one bit of data is transmitted or received on the SWDIO. The data read from SWDIO can then be retrieved from the input buffer.

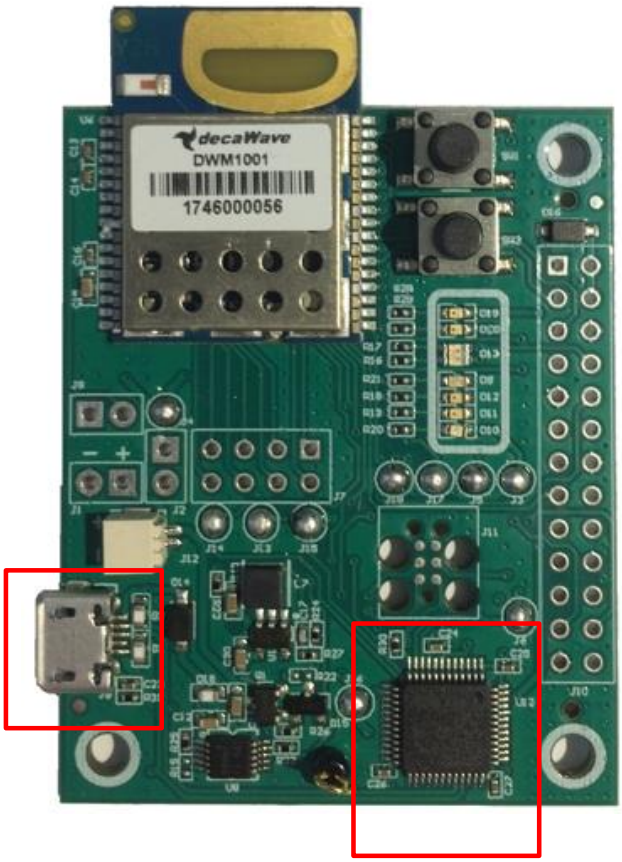


Figure 11: Module Development Board JLink components

6 POWER SUPPLY AND BATTERY CONNECTIONS

The Power Supply takes its input from USB or from a Battery or from a connected Raspberry Pi. It powers the module and the other devices on the Module Development Board. It can also charge a connected battery when powered by USB or the Raspberry Pi.

The Battery Charger is a Lithium - Ion battery charger. Batteries can be connected to the module development board at the connectors shown in Figure 12.

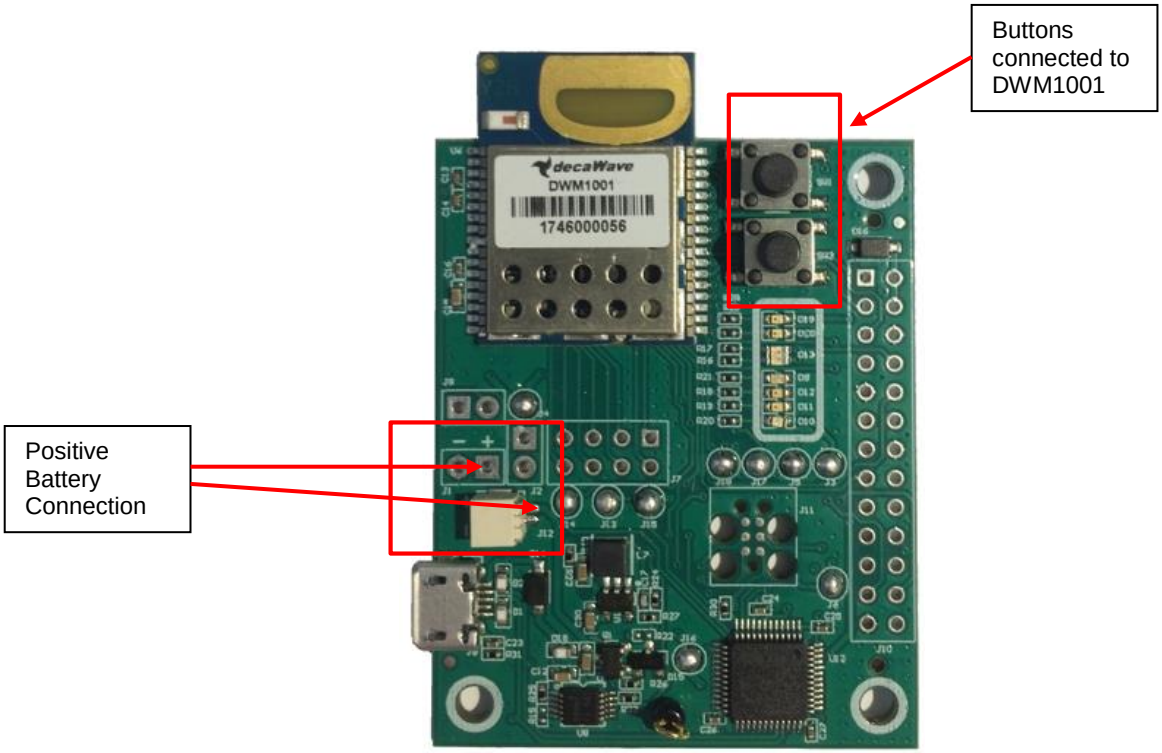


Figure 12: Picture of the module development board showing the battery connection points and buttons

7 BUTTONS SW1 AND SW2

There are two buttons on the PCB, SW1 and SW2. SW1 is connected to the RESETn pin on the DWM1001 module and SW2 is connected to the BT_WAKE_UP pin on the DWM1001 module. SW2 wakes up the Bluetooth functionality when a tag is in low-power mode, as described in the System Overview document[3]. Buttons are shown above in Figure 12.

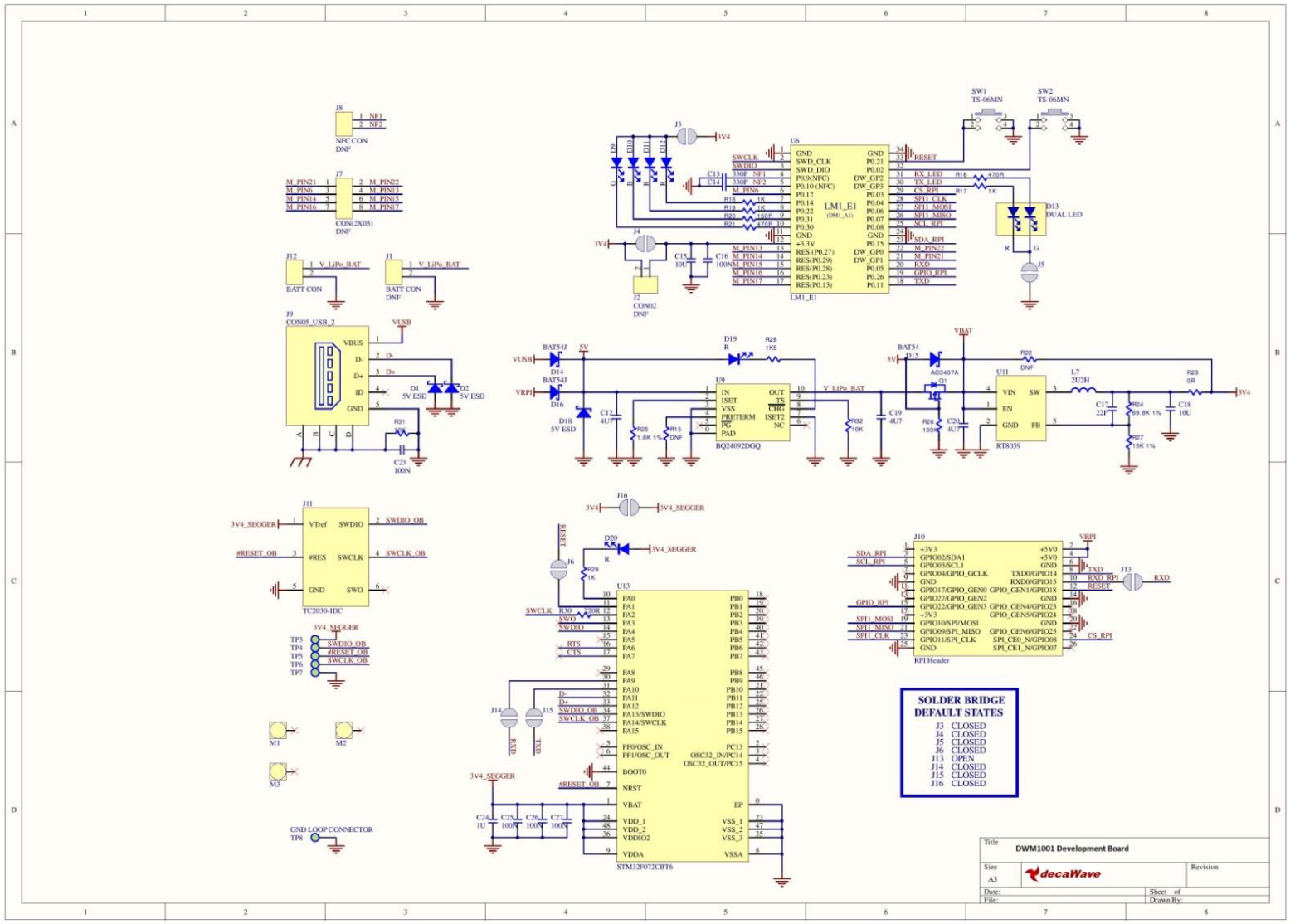
8 DEVELOPMENT BOARD SOLDER BRIDGE JUMPERS

The Module Development Board has eight solder bridge jumpers. These can be used to allow evaluation of different aspects of the DWM1001 modules performance. For example Solder Bridge J4 can be desoldered and a resistor placed across connector J2 to allow measurement of the modules current consumption. Table 4 gives a list of these jumpers, their purpose and the state they are in when leaving the factory. Investigation of the schematic of the Module Development board at the end of this document will give further details on their use.

Table 4: A list of solder jumpers available on the Module Development Board

Jumper Number	Purpose	Default State
J3	Desolder to disconnect user LEDs from the module	Closed
J5	Desolder to disconnect Tx and Rx LEDs from the module	Closed
J4	Desolder to measure module current in J2	Closed
J6	Desolder to disconnect Reset button from JLINK	Closed
J13	Solder to connect UART Rx between Module and Raspberry Pi	Closed
J14	Desolder to disconnect module RXD from JLINK	Closed
J15	Desolder to disconnect module TXD from JLINK	Closed
J16	Desolder to disconnect power to JLINK	Closed

9 DEVELOPMENT BOARD SCHEMATIC



10 REFERENCES

[1] Decawave DWM1001 Datasheet www.decawave.com
[2] IEEE802.15.4-2011 or "IEEE Std 802.15.4™-2011" (Revision of IEEE Std 802.15.4-2006). IEEE Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Computer Society Sponsored by the LAN/MAN Standards Committee. Available from http://standards.ieee.org/
[3] DWM1001 System Overview

11 DOCUMENT HISTORY

Table 5: Document History

Revision	Date	Description
1.0	20/12/17	First Version

12 MAJOR CHANGES

To be completed when document is updated.

13 ABOUT DECAWAVE

Decawave is a pioneering fabless semiconductor company whose flagship product, the DW1000, is a complete, single chip CMOS Ultra-Wideband IC based on the IEEE 802.15.4-2011[2] UWB standard. This device is the first in a family of parts that will operate at data rates of 110 kbps, 850 kbps, 6.8 Mbps.

The resulting silicon has a wide range of standards-based applications for both Real Time Location Systems (RTLS) and Ultra Low Power Wireless Transceivers in areas as diverse as manufacturing, healthcare, lighting, security, transport, inventory & supply chain management.

Further Information

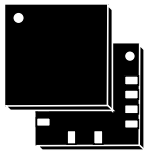
For further information on this or any other Decawave product contact a sales representative as follows: -

Decawave Ltd
Adelaide Chambers
Peter Street
Dublin
D08 T6YA
Ireland
+353 1 6975030
e: sales@decawave.com
w: www.decawave.com

LIS2DH12

MEMS digital output motion sensor: ultra-low-power high-performance 3-axis "femto" accelerometer

Datasheet - production data



LGA-12 (2.0x2.0x1 mm)

Features

- Wide supply voltage, 1.71 V to 3.6 V
- Independent IO supply (1.8 V) and supply voltage compatible
- Ultra-low power consumption down to 2 µA
- ±2*g*/±4*g*/±8*g*/±16*g* selectable full scales
- I^2C/SPI digital output interface
- 2 independent programmable interrupt generators for free-fall and motion detection
- 6D/4D orientation detection
- “Sleep-to-wake” and “return-to-sleep” functions
- Free-fall detection
- Motion detection
- Embedded temperature sensor
- Embedded FIFO
- ECOPACK®, RoHS and “Green” compliant

Applications

- Motion-activated functions
- Display orientation
- Shake control
- Pedometer
- Gaming and virtual reality input devices
- Impact recognition and logging

Description

The LIS2DH12 is an ultra-low-power high-performance three-axis linear accelerometer belonging to the “femto” family with digital I^2C/SPI serial interface standard output.

The LIS2DH12 has user-selectable full scales of ±2*g*/±4*g*/±8*g*/±16*g* and is capable of measuring accelerations with output data rates from 1 Hz to 5.3 kHz.

The self-test capability allows the user to check the functionality of the sensor in the final application.

The device may be configured to generate interrupt signals by detecting two independent inertial wake-up/free-fall events as well as by the position of the device itself.

The LIS2DH12 is available in a small thin plastic land grid array package (LGA) and is guaranteed to operate over an extended temperature range from -40 °C to +85 °C.

Table 1. Device summary

Order code	Temp. range [°C]	Package	Packaging
LIS2DH12TR	-40 to +85	LGA-12	Tape and reel

Contents

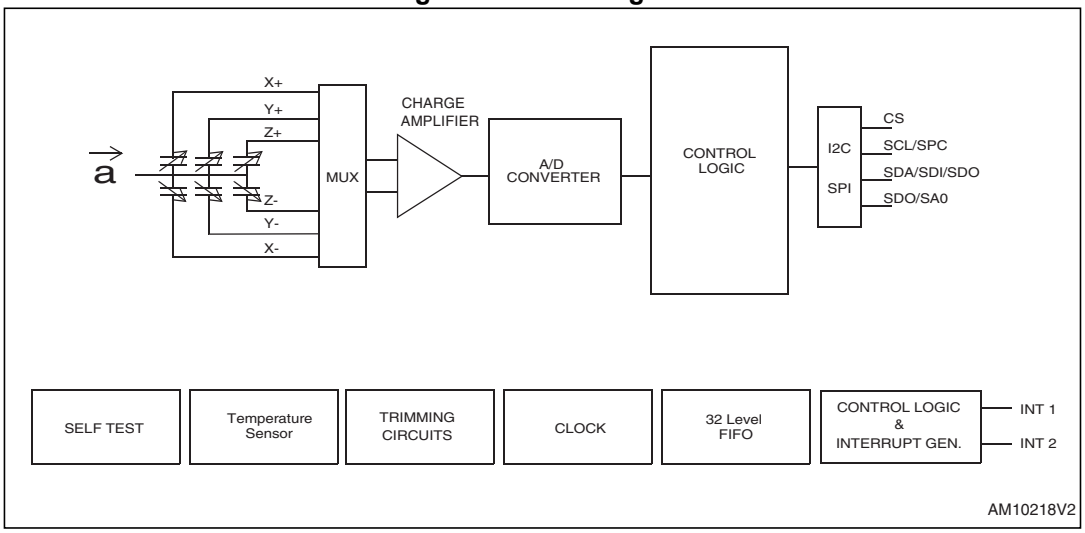
List of tables

List of figures

1 Block diagram and pin description

1.1 Block diagram

Figure 1. Block diagram



1.2 Pin description

Figure 2. Pin connections

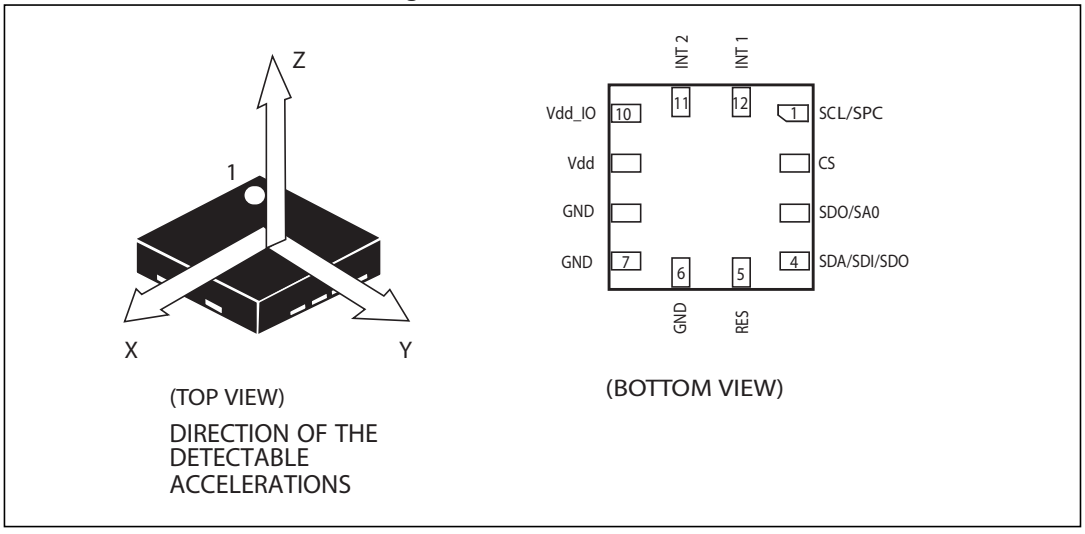


Table 2. Pin description

Pin#	Name	Function
1	SCL SPC	I^2C serial clock (SCL) SPI serial port clock (SPC)
2	CS	SPI enable I^2C/SPI mode selection: 1: SPI idle mode / I^2C communication enabled 0: SPI communication mode / I^2C disabled
3(1)	SDO SA0	SPI serial data output (SDO) I^2C less significant bit of the device address (SA0)
4	SDA SDI SDO	I^2C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
5	Res	Connect to GND
6	GND	0 V supply
7	GND	0 V supply
8	GND	0 V supply
9	Vdd	Power supply
10	Vdd_IO	Power supply for I/O pins
11	INT2	Interrupt pin 2
12	INT1	Interrupt pin 1

1. SDO/SA0 pin is internally pulled up. Refer to *Table 3* for the internal pull-up values (typ).

Table 3. Internal pull-up values (typ.) for SDO/SA0 pin

Vdd_IO	Resistor value for SDO/SA0 pin
	Typ. (kΩ)
1.7 V	54.4
1.8 V	49.2
2.5 V	30.4
3.6 V	20.4

2 Mechanical and electrical specifications

2.1 Mechanical characteristics

@ Vdd = 2.5 V, T = 25 °C unless otherwise noted[a]

Table 4. Mechanical characteristics

Symbol	Parameter	Test conditions	Min.	Typ.[1]	Max.	Unit
FS	Measurement range[2]	FS bit set to 00		±2.0		*g*
		FS bit set to 01		±4.0		
		FS bit set to 10		±8.0		
		FS bit set to 11		±16.0		
So	Sensitivity	FS bit set to 00; High-resolution mode		1		mg/digit
		FS bit set to 00; Normal mode		4		
		FS bit set to 00; Low-power mode		16		
		FS bit set to 01; High-resolution mode		2		mg/digit
		FS bit set to 01; Normal mode		8		
		FS bit set to 01; Low-power mode		32		
		FS bit set to 10; High-resolution mode		4		mg/digit
		FS bit set to 10; Normal mode		16		
		FS bit set to 10; Low-power mode		64		
		FS bit set to 11; High-resolution mode		12		mg/digit
		FS bit set to 11; Normal mode		48		
		FS bit set to 11; Low-power mode		192		
TCSo	Sensitivity change vs. temperature	FS bit set to 00		±0.01		%/°C
TyOff	Typical zero-*g* level offset accuracy[3]	FS bit set to 00		±40		mg

a. The product is factory calibrated at 2.5 V. The operational power supply range is from 1.71 V to 3.6 V.

Table 4. Mechanical characteristics (continued)

Symbol	Parameter	Test conditions	Min.	Typ.(1)	Max.	Unit
TCOff	Zero-*g* level change vs. temperature	Max delta from 25 °C		±0.5		m*g*/°C
An	Acceleration noise density	FS bit set to 00, High-Resolution mode (*Table 10*), ODR > 1300 Hz		220		µ*g*/√Hz
Vst	Self-test output change(4) (5) (6)	FS bit set to 00 X-axis; Normal mode	17		360	LSb
		FS bit set to 00 Y-axis; Normal mode	17		360	LSb
		FS bit set to 00 Z-axis; Normal mode	17		360	LSb
Top	Operating temperature range		-40		+85	°C

1. Typical specifications are not guaranteed.
2. Verified by wafer level test and measurement of initial offset and sensitivity.
3. Typical zero-g level offset value after factory calibration test at socket level.
4. The sign of “Self-test output change” is defined by the ST bits in *CTRL_REG4 (23h)*, for all axes.
5. “Self-test output change” is defined as the absolute value of: $OUTPUT[LSb]_{(Self\ test\ enabled)}$ - $OUTPUT[LSb]_{(Self\ test\ disabled)}$. 1LSb = 4 m*g* at 10-bit representation, ±2 *g* full scale
6. After enabling the self-test, correct data is obtained after two samples (low-power mode / normal mode) or after eight samples (high-resolution mode).

2.2 Temperature sensor characteristics

@ Vdd = 2.5 V, T = 25 °C unless otherwise noted(b)

Table 5. Temperature sensor characteristics

Symbol	Parameter	Min.	Typ.(1)	Max.	Unit
TSDr	Temperature sensor output change vs. temperature		1		digit/°C(2)
TODR	Temperature refresh rate		ODR(3)		Hz
Top	Operating temperature range	-40		+85	°C

1. Typical specifications are not guaranteed.
2. 8-bit resolution.
3. Refer to *Table 31*.

b. The product is factory calibrated at 2.5 V. Temperature sensor operation is guaranteed in the range 2 V - 3.6 V.

2.3 Electrical characteristics

@ Vdd = 2.5 V, T = 25 °C unless otherwise noted[c]

Table 6. Electrical characteristics

Symbol	Parameter	Test conditions	Min.	Typ.[1]	Max.	Unit
Vdd	Supply voltage		1.71	2.5	3.6	V
Vdd_IO	I/O pins supply voltage[2]		1.71		Vdd+0.1	V
Idd	Current consumption in normal mode	50 Hz ODR		11		µA
		1 Hz ODR		2		µA
IddLP	Current consumption in low-power mode	50 Hz ODR		6		µA
IddPdn	Current consumption in power-down mode			0.5		µA
VIH	Digital high-level input voltage		0.8*Vdd_IO			V
VIL	Digital low-level input voltage				0.2*Vdd_IO	V
VOH	High-level output voltage		0.9*Vdd_IO			V
VOL	Low-level output voltage				0.1*Vdd_IO	V
Top	Operating temperature range		-40		+85	°C

1. Typical specification are not guaranteed.
2. It is possible to remove Vdd maintaining Vdd_IO without blocking the communication busses, in this condition the measurement chain is powered off.

c. The product is factory calibrated at 2.5 V. The operational power supply range is from 1.71 V to 3.6 V.

2.4 Communication interface characteristics

2.4.1 SPI - serial peripheral interface

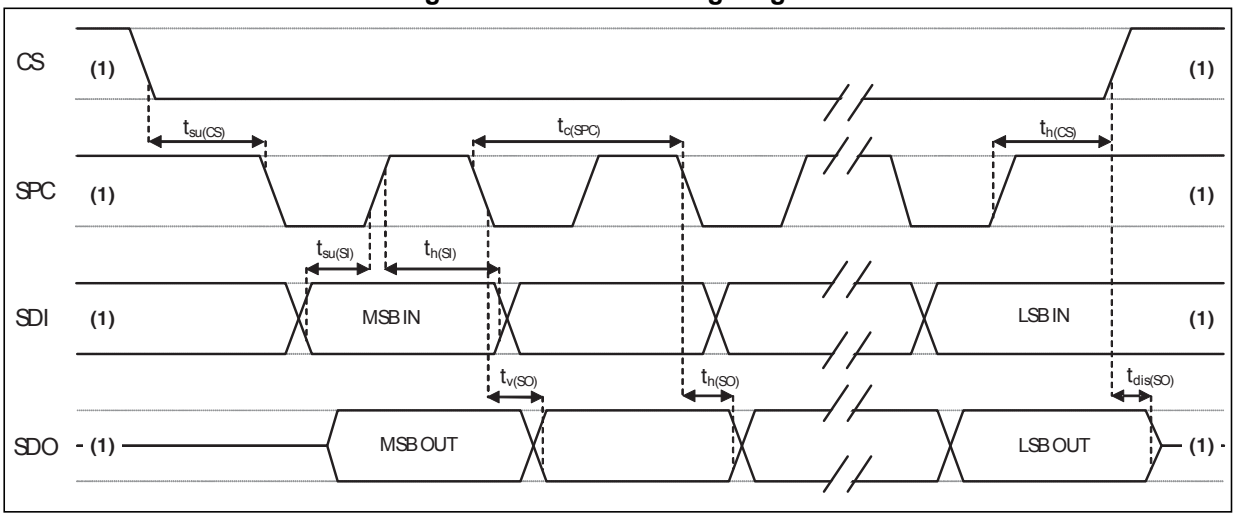
Subject to general operating conditions for Vdd and Top.

Table 7. SPI slave timing values

Symbol	Parameter	Value (1)		Unit
		Min	Max	
$t_{c(SPC)}$	SPI clock cycle	100		ns
$f_{c(SPC)}$	SPI clock frequency		10	MHz
$t_{su(CS)}$	CS setup time	5		ns
$t_{h(CS)}$	CS hold time	20		
$t_{su(SI)}$	SDI input setup time	5		
$t_{h(SI)}$	SDI input hold time	15		
$t_{v(SO)}$	SDO valid output time		50	
$t_{h(SO)}$	SDO output hold time	5		
$t_{dis(SO)}$	SDO output disable time		50	

1. Values are guaranteed at 10 MHz clock frequency for SPI with both 4 and 3 wires, based on characterization results, not tested in production.

Figure 3. SPI slave timing diagram



1. When no communication is ongoing, data on SDO is driven by internal pull-up resistors.

Note: Measurement points are done at 0.2·Vdd_IO and 0.8·Vdd_IO, for both input and output ports.

2.4.2 I²C - inter-IC control interface

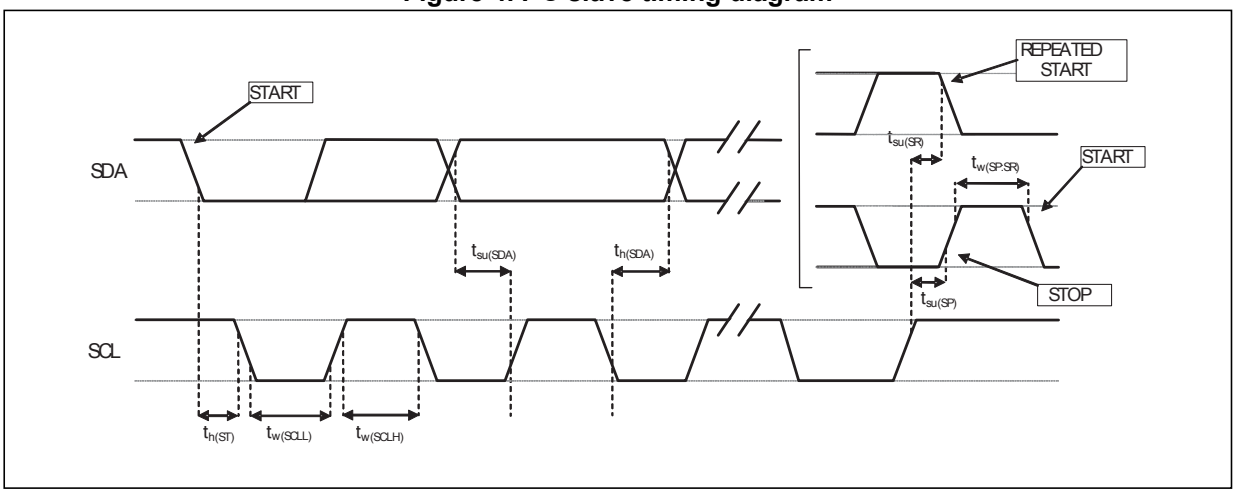
Subject to general operating conditions for Vdd and top.

Table 8. I²C slave timing values

Symbol	Parameter	I²C standard mode (1)		I²C fast mode (1)		Unit
		Min	Max	Min	Max	
$f_{(SCL)}$	SCL clock frequency	0	100	0	400	kHz
$t_{w(SCLL)}$	SCL clock low time	4.7		1.3		µs
$t_{w(SCLH)}$	SCL clock high time	4.0		0.6		
$t_{su(SDA)}$	SDA setup time	250		100		ns
$t_{h(SDA)}$	SDA data hold time	0	3.45	0	0.9	µs
$t_{h(ST)}$	START condition hold time	4		0.6		µs
$t_{su(SR)}$	Repeated START condition setup time	4.7		0.6		
$t_{su(SP)}$	STOP condition setup time	4		0.6		
$t_{w(SP:SR)}$	Bus free time between STOP and START condition	4.7		1.3		

1. Data based on standard I²C protocol requirement, not tested in production.

Figure 4. I²C slave timing diagram



Note: Measurement points are done at 0.2·Vdd_IO and 0.8·Vdd_IO, for both ports.

2.5 Absolute maximum ratings

Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 9. Absolute maximum ratings

Symbol	Ratings	Maximum value	Unit
Vdd	Supply voltage	-0.3 to 4.8	V
Vdd_IO	Supply voltage on I/O pins	-0.3 to 4.8	V
Vin	Input voltage on any control pin (CS, SCL/SPC, SDA/SDI/SDO, SDO/SA0)	-0.3 to Vdd_IO +0.3	V
A_{POW}	Acceleration (any axis, powered, Vdd = 2.5 V)	3000 *g* for 0.5 ms	
		10000 *g* for 0.2 ms	
A_{UNP}	Acceleration (any axis, unpowered)	3000 *g* for 0.5 ms	
		10000 *g* for 0.2 ms	
T_{OP}	Operating temperature range	-40 to +85	°C
T_{STG}	Storage temperature range	-40 to +125	°C
ESD	Electrostatic discharge protection (HBM)	2	kV

Note: Supply voltage on any pin should never exceed 4.8 V.

This device is sensitive to mechanical shock, improper handling can cause permanent damage to the part.

This device is sensitive to electrostatic discharge (ESD), improper handling can cause permanent damage to the part.

3 Terminology and functionality

3.1 Terminology

3.1.1 Sensitivity

Sensitivity describes the gain of the sensor and can be determined by applying 1 *g* acceleration to it. As the sensor can measure DC accelerations, this can be done easily by pointing the axis of interest towards the center of the Earth, noting the output value, rotating the sensor by 180 degrees (pointing to the sky) and noting the output value again. By doing so, ±1 *g* acceleration is applied to the sensor. Subtracting the larger output value from the smaller one, and dividing the result by 2, leads to the actual sensitivity of the sensor. This value changes very little over temperature and time. The sensitivity tolerance describes the range of sensitivities of a large population of sensors.

3.1.2 Zero-*g* level

The zero-*g* level offset (TyOff) describes the deviation of an actual output signal from the ideal output signal if no acceleration is present. A sensor in a steady state on a horizontal surface will measure 0 *g* for the X-axis and 0 *g* for the Y-axis whereas the Z-axis will measure 1 *g*. The output is ideally in the middle of the dynamic range of the sensor (content of OUT registers 00h, data expressed as two's complement number). A deviation from the ideal value in this case is called zero-*g* offset. Offset is to some extent a result of stress to the MEMS sensor and therefore the offset can slightly change after mounting the sensor on a printed circuit board or exposing it to extensive mechanical stress. Offset changes little over temperature, see *Table 4* "Zero-*g* level change vs. temperature" (TCOff). The zero-*g* level tolerance (TyOff) describes the standard deviation of the range of zero-*g* levels of a population of sensors.

3.2 Functionality

3.2.1 High-resolution, normal mode, low-power mode

The LIS2DH12 provides three different operating modes: *high-resolution mode*, *normal mode* and *low-power mode*.

The table below summarizes how to select the different operating modes.

Table 10. Operating mode selection

Operating mode	CTRL_REG1[3] (LPen bit)	CTRL_REG4[3] (HR bit)	BW [Hz]	Turn-on time [ms]	So @ ±2*g* [mg/digit]
Low-power mode (8-bit data output)	1	0	ODR/2	1	16
Normal mode (10-bit data output)	0	0	ODR/2	1.6	4
High-resolution mode (12-bit data output)(1)	0	1	ODR/9	7/ODR	1
Not allowed	1	1	--	--	--

1. By design, when the device from high-resolution configuration (HR) is set to power-down mode (PD), it is recommended to read register *REFERENCE (26h)* for a complete reset of the filtering block before switching to normal/high-performance mode again.

The turn-on time to transition to another operating mode is given in *Table 11*.

Table 11. Turn-on time for operating mode transition

Operating mode change	Turn-on time [ms]
12-bit mode to 8-bit mode	1/ODR
12-bit mode to 10-bit mode	1/ODR
10-bit mode to 8-bit mode	1/ODR
10-bit mode to 12-bit mode	7/ODR
8-bit mode to 10-bit mode	1/ODR
8-bit mode to 12-bit mode	7/ODR

Table 12. Current consumption of operating modes

Operating mode [Hz]	Low-power mode (8-bit data output) [µA]	Normal mode (10-bit data output) [µA]	High resolution (12-bit data output) [µA]
1	2	2	2
10	3	4	4
25	4	6	6
50	6	11	11
100	10	20	20
200	18	38	38
400	36	73	73
1344	--	185	185
1620	100	--	--
5376	185	--	--

3.2.2 Self-test

The self-test allows the user to check the sensor functionality without moving it. When the self-test is enabled, an actuation force is applied to the sensor, simulating a definite input acceleration. In this case the sensor outputs will exhibit a change in their DC levels which are related to the selected full scale through the device sensitivity. When the self-test is activated, the device output level is given by the algebraic sum of the signals produced by the acceleration acting on the sensor and by the electrostatic test-force. If the output signals change within the amplitude specified inside *Table 4*, then the sensor is working properly and the parameters of the interface chip are within the defined specifications.

3.2.3 6D / 4D orientation detection

The LIS2DH12 provides the capability to detect the orientation of the device in space, enabling easy implementation of energy-saving procedures and automatic image rotation for mobile devices.

The 4D detection is a subset of the 6D function especially defined to be implemented in mobile devices for portrait and landscape computation. In 4D configuration, the Z-axis position detection is disabled.

3.2.4 “Sleep-to-wake” and “Return-to-sleep”

The LIS2DH12 can be programmed to automatically switch to low-power mode upon recognition of a determined event.
Once the event condition is over, the device returns back to the preset normal or high-resolution mode.

To enable this function the desired threshold value must be stored inside the *ACT_THS (3Eh)* register while the duration value is written inside the *ACT_DUR (3Fh)* register.

When the acceleration falls below the threshold value, the device automatically switches to low-power mode (10Hz ODR).
During this condition, the ODR[3:0] bits and the LPen bit inside *CTRL_REG1 (20h)* and the HR bit in *CTRL_REG4 (23h)* are not considered.

As soon as the acceleration rises above threshold, the module restores the operating mode and ODRs as determined by the *CTRL_REG1 (20h)* and *CTRL_REG4 (23h)* settings.

3.3 Sensing element

A proprietary process is used to create a surface micromachined accelerometer. The technology processes suspended silicon structures which are attached to the substrate in a few points called anchors and are free to move in the direction of the sensed acceleration. To be compatible with traditional packaging techniques, a cap is placed on top of the sensing element to avoid blocking the moving parts during the molding phase of the plastic encapsulation.

When an acceleration is applied to the sensor, the proof mass displaces from its nominal position, causing an imbalance in the capacitive half-bridge. This imbalance is measured using charge integration in response to a voltage pulse applied to the capacitor.

At steady state the nominal value of the capacitors are a few pF and when an acceleration is applied, the maximum variation of the capacitive load is in the fF range.

3.4 IC interface

The complete measurement chain is composed of a low-noise capacitive amplifier which converts the capacitive unbalance of the MEMS sensor into an analog voltage that will be available to the user through an analog-to-digital converter.

The acceleration data may be accessed through an I^2C/SPI interface, thus making the device particularly suitable for direct interfacing with a microcontroller.

The LIS2DH12 features a data-ready signal (DRDY) which indicates when a new set of measured acceleration data is available, thus simplifying data synchronization in the digital system that uses the device.

The LIS2DH12 may also be configured to generate an inertial wake-up and free-fall interrupt signal according to a programmed acceleration event along the enabled axes. Both free-fall and wake-up can be available simultaneously on two different pins.

3.5 Factory calibration

The IC interface is factory calibrated for sensitivity (So) and zero-*g* level (TyOff).

The trim values are stored inside the device in non-volatile memory. Any time the device is turned on, these values are downloaded into the registers to be used during active operation. This allows using the device without further calibration.

3.6 FIFO

The LIS2DH12 contains a 10-bit, 32-level FIFO. Buffered output allows the following operation modes: FIFO, Stream, Stream-to-FIFO and FIFO bypass. When FIFO bypass mode is activated, FIFO is not operating and remains empty. In FIFO mode, measurement data from acceleration detection on the x, y, and z-axes are stored in the FIFO buffer.

3.7 Temperature sensor

In order to enable the internal temperature sensor, bits TEMP_EN[1:0] in register *TEMP_CFG_REG (1Fh)* and the BDU bit in *CTRL_REG4 (23h)* have to be set.

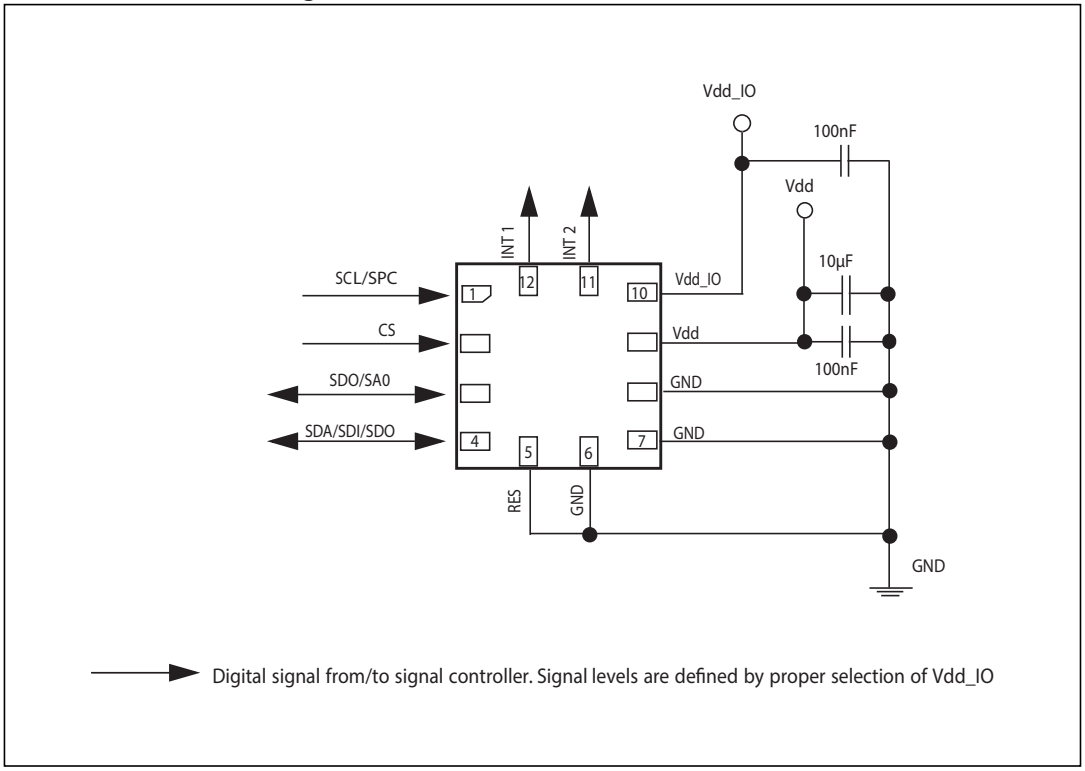
The temperature is available in *OUT_TEMP_L (0Ch), OUT_TEMP_H (0Dh)* stored as two's complement data, left-justified.

The temperature data format can be 10 bits if LPen (bit 3) in *CTRL_REG1 (20h)* is cleared (high-resolution / normal mode), otherwise, in low-power mode, the ADC resolution is 8-bit.

Refer to *Table 5: Temperature sensor characteristics* for the conversion factor.

4 Application hints

Figure 5. LIS2DH12 electrical connections



The device core is supplied through the Vdd line while the I/O pads are supplied through the Vdd_IO line. Power supply decoupling capacitors (100 nF ceramic, 10 µF aluminum) should be placed as near as possible to pin 9 of the device (common design practice).

All the voltage and ground supplies must be present at the same time to have proper behavior of the IC (refer to *Figure 5*). It is possible to remove Vdd while maintaining Vdd_IO without blocking the communication bus, in this condition the measurement chain is powered off.

The functionality of the device and the measured acceleration data is selectable and accessible through the I^2C or SPI interfaces. When using the I^2C, CS must be tied high.

The functions, the threshold and the timing of the two interrupt pins (INT1 and INT2) can be completely programmed by the user through the I^2C/SPI interface.

Table 13. Internal pin status

Pin#	Name	Function	Pin status
1	SCL SPC	I²C serial clock (SCL) SPI serial port clock (SPC)	Default: input high impedance
2	CS	SPI enable I²C/SPI mode selection: 1: SPI idle mode / I²C communication enabled 0: SPI communication mode / I²C disabled	Default: input high impedance
3	SDO SA0	SPI serial data output (SDO) I²C less significant bit of the device address (SA0)	Default: input with internal pull-up(1)
4	SDA SDI SDO	I²C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)	Default: (SDA) input high impedance
5	Res	Connect to GND	
6	GND	0 V supply	
7	GND	0 V supply	
8	GND	0 V supply	
9	Vdd	Power supply	
10	Vdd_IO	Power supply for I/O pins	
11	INT2	Interrupt pin 2	Default: push-pull output forced to GND
12	INT1	Interrupt pin 1	Default: push-pull output forced to GND

1. In order to disable the internal pull-up on the SDO/SA0 pin, write 90h in *CTRL_REG0 (1Eh)*.

4.1 Soldering information

The LGA package is compliant with the ECOPACK®, RoHS and “Green” standard.
It is qualified for soldering heat resistance according to JEDEC J-STD-020.

Leave “Pin 1 Indicator” unconnected during soldering.

Land pattern and soldering recommendations are available at www.st.com.

5 Digital main blocks

5.1 FIFO

The LIS2DH12 embeds a 32-level FIFO for each of the three output channels, X, Y and Z. This allows consistent power saving for the system, since the host processor does not need to continuously poll data from the sensor, but it can wake up only when needed and burst the significant data out from the FIFO.

In order to enable the FIFO buffer, the FIFO_EN bit in *CTRL_REG5 (24h)* must be set to '1'.

This buffer can work according to the following different modes: Bypass mode, FIFO mode, Stream mode and Stream-to-FIFO mode. Each mode is selected by the FM [1:0] bits in *FIFO_CTRL_REG (2Eh)*. Programmable FIFO watermark level, FIFO empty or FIFO overrun events can be enabled to generate dedicated interrupts on the INT1 pin (configuration through *CTRL_REG3 (22h)*).

In the *FIFO_SRC_REG (2Fh)* register the EMPTY bit is equal to '1' when all FIFO samples are ready and FIFO is empty.

In the *FIFO_SRC_REG (2Fh)* register the WTM bit goes to '1' if new data is written in the buffer and *FIFO_SRC_REG (2Fh)* (FSS [4:0]) is greater than or equal to *FIFO_CTRL_REG (2Eh)* (FTH [4:0]). *FIFO_SRC_REG (2Fh)* (WTM) goes to '0' if reading an X, Y, Z data slot from FIFO and *FIFO_SRC_REG (2Fh)* (FSS [4:0]) is less than or equal to *FIFO_CTRL_REG (2Eh)* (FTH [4:0]).

In the *FIFO_SRC_REG (2Fh)* register the OVRN_FIFO bit is equal to '1' if the FIFO slot is overwritten.

5.1.1 Bypass mode

In Bypass mode the FIFO is not operational and for this reason it remains empty. For each channel only the first address is used. The remaining FIFO levels are empty.

Bypass mode must be used in order to reset the FIFO buffer when a different mode is operating (i.e. FIFO mode).

5.1.2 FIFO mode

In FIFO mode, the buffer continues filling data from the X, Y and Z accelerometer channels until it is full (a set of 32 samples stored). When the FIFO is full, it stops collecting data from the input channels and the FIFO content remains unchanged.

An overrun interrupt can be enabled, I1_OVERRUN = '1' in the *CTRL_REG3 (22h)* register, in order to be raised when the FIFO stops collecting data. When the overrun interrupt occurs, the first data has been overwritten and the FIFO stops collecting data from the input channels.

After the last read it is necessary to transit from Bypass mode in order to reset the FIFO content. After this reset command, it is possible to restart FIFO mode just by selecting the FIFO mode configuration (FM[1:0] bits) in register *FIFO_CTRL_REG (2Eh)*.

5.1.3 Stream mode

In Stream mode the FIFO continues filling data from the X, Y, and Z accelerometer channels until the buffer is full (a set of 32 samples stored) at which point the FIFO buffer index restarts from the beginning and older data is replaced by the current data. The oldest values continue to be overwritten until a read operation frees the FIFO slots.

An overrun interrupt can be enabled, I1_OVERRUN = '1' in the *CTRL_REG3 (22h)* register, in order to read the entire contents of the FIFO at once. If, in the application, it is mandatory not to lose data and it is not possible to read at least one sample for each axis within one ODR period, a watermark interrupt can be enabled in order to read partially the FIFO and leave memory slots free for incoming data.

Setting the FTH [4:0] bit in the *FIFO_CTRL_REG (2Eh)* register to an N value, the number of X, Y and Z data samples that should be read at the rise of the watermark interrupt is up to (N+1).

5.1.4 Stream-to-FIFO mode

In Stream-to-FIFO mode, data from the X, Y and Z accelerometer channels are collected in a combination of Stream mode and FIFO mode. The FIFO buffer starts operating in Stream mode and switches to FIFO mode when the selected interrupt occurs.

The FIFO operating mode changes according to the INT1 pin value if the TR bit is set to ‘0’ in the *FIFO_CTRL_REG (2Eh)* register or the INT2 pin value if the TR bit is set to‘1’ in the *FIFO_CTRL_REG (2Eh)* register.

When the interrupt pin is selected and the interrupt event is configured on the corresponding pin, the FIFO operates in Stream mode if the pin value is equal to ‘0’ and it operates in FIFO mode if the pin value is equal to ‘1’. Switching modes is dynamically performed according to the pin value.

Stream-to-FIFO can be used in order to analyze the sampling history that generates an interrupt. The standard operation is to read the contents of FIFO when the FIFO mode is triggered and the FIFO buffer is full and stopped.

5.1.5 Retrieving data from FIFO

FIFO data is read from *OUT_X_L (28h), OUT_X_H (29h)*, *OUT_Y_L (2Ah), OUT_Y_H (2Bh)* and *OUT_Z_L (2Ch), OUT_Z_H (2Dh)*. When the FIFO is in Stream, Stream-to-FIFO or FIFO mode, a read operation to the *OUT_X_L (28h), OUT_X_H (29h)*, *OUT_Y_L (2Ah), OUT_Y_H (2Bh)* or *OUT_Z_L (2Ch), OUT_Z_H (2Dh)* registers provides the data stored in the FIFO. Each time data is read from the FIFO, the oldest X, Y and Z data are placed in the *OUT_X_L (28h), OUT_X_H (29h)*, *OUT_Y_L (2Ah), OUT_Y_H (2Bh)* and *OUT_Z_L (2Ch), OUT_Z_H (2Dh)* registers and both single read and read_burst operations can be used.

The address to be read is automatically updated by the device and it rolls back to 0x28 when register 0x2D is reached. In order to read all FIFO levels in a multiple byte read,192 bytes (6 output registers of 32 levels) have to be read.

6 Digital interfaces

The registers embedded inside the LIS2DH12 may be accessed through both the I^2C and SPI serial interfaces. The latter may be SW configured to operate either in 3-wire or 4-wire interface mode.

The serial interfaces are mapped to the same pads. To select/exploit the I^2C interface, the CS line must be tied high (i.e. connected to Vdd_IO).

Table 14. Serial interface pin description

Pin name	Pin description
CS	SPI enable I^2C/SPI mode selection: 1: SPI idle mode / I^2C communication enabled 0: SPI communication mode / I^2C disabled
SCL SPC	I^2C serial clock (SCL) SPI serial port clock (SPC)
SDA SDI SDO	I^2C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
SA0 SDO	I^2C less significant bit of the device address (SA0) SPI serial data output (SDO)

6.1 I^2C serial interface

The LIS2DH12 I^2C is a bus slave. The I^2C is employed to write data into registers whose content can also be read back.

The relevant I^2C terminology is given in the table below.

Table 15. I^2C terminology

Term	Description
Transmitter	The device which sends data to the bus
Receiver	The device which receives data from the bus
Master	The device which initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by the master

There are two signals associated with the I^2C bus: the serial clock line (SCL) and the serial data line (SDA). The latter is a bidirectional line used for sending and receiving data to/from the interface. Both the lines must be connected to Vdd_IO through an external pull-up resistor. When the bus is free, both the lines are high.

The I^2C interface is compliant with fast mode (400 kHz) I^2C standards as well as with normal mode.

6.1.1 I²C operation

The transaction on the bus is started through a START (ST) signal. A START condition is defined as a HIGH-to-LOW transition on the data line while the SCL line is held HIGH. After this has been transmitted by the master, the bus is considered busy. The next byte of data transmitted after the start condition contains the address of the slave in the first 7 bits and the eighth bit tells whether the master is receiving data from the slave or transmitting data to the slave. When an address is sent, each device in the system compares the first seven bits after a start condition with its address. If they match, the device considers itself addressed by the master.

The Slave ADdress (SAD) associated to the LIS2DH12 is 001100xb. The **SDO**/**SA0** pad can be used to modify the less significant bit of the device address. If the SA0 pad is connected to the voltage supply, LSb is ‘1’ (address 0011001b), else if the SA0 pad is connected to ground, the LSb value is ‘0’ (address 0011000b). This solution permits to connect and address two different accelerometers to the same I²C lines.

Data transfer with acknowledge is mandatory. The transmitter must release the SDA line during the acknowledge pulse. The receiver must then pull the data line LOW so that it remains stable low during the HIGH period of the acknowledge clock pulse. A receiver which has been addressed is obliged to generate an acknowledge after each byte of data received.

The I²C embedded inside the LIS2DH12 behaves like a slave device and the following protocol must be adhered to. After the start condition (ST) a slave address is sent, once a slave acknowledge (SAK) has been returned, an 8-bit sub-address (SUB) is transmitted: the 7 LSb represent the actual register address while the MSb enables address auto increment. If the MSb of the SUB field is ‘1’, the SUB (register address) is automatically increased to allow multiple data read/writes.

The slave address is completed with a Read/Write bit. If the bit is ‘1’ (Read), a repeated START (SR) condition must be issued after the two sub-address bytes; if the bit is ‘0’ (Write) the master will transmit to the slave with direction unchanged. *Table 16* explains how the SAD+read/write bit pattern is composed, listing all the possible configurations.

Table 16. SAD+read/write patterns

Command	SAD[6:1]	SAD[0] = SA0	R/W	SAD+R/W
Read	001100	0	1	00110001 (31h)
Write	001100	0	0	00110000 (30h)
Read	001100	1	1	00110011 (33h)
Write	001100	1	0	00110010 (32h)

Table 17. Transfer when master is writing one byte to slave

Master	ST	SAD + W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

Table 18. Transfer when master is writing multiple bytes to slave

Master	ST	SAD + W		SUB		DATA		DATA		SP
Slave			SAK		SAK		SAK		SAK	

Table 19. Transfer when master is receiving (reading) one byte of data from slave

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

Table 20. Transfer when master is receiving (reading) multiple bytes of data from slave

Master	ST	SAD+W		SUB		SR	SAD+R			MAK		MAK		NMAK	SP
Slave			SAK		SAK			SAK	DATA		DATA		DATA		

Data are transmitted in byte format (DATA). Each data transfer contains 8 bits. The number of bytes transferred per transfer is unlimited. Data is transferred with the Most Significant bit (MSb) first. If a receiver can't receive another complete byte of data until it has performed some other function, it can hold the clock line, SCL low to force the transmitter into a wait state. Data transfer only continues when the receiver is ready for another byte and releases the data line. If a slave receiver doesn't acknowledge the slave address (i.e. it is not able to receive because it is performing some real-time function) the data line must be left HIGH by the slave. The master can then abort the transfer. A low-to-high transition on the SDA line while the SCL line is HIGH is defined as a STOP condition. Each data transfer must be terminated by the generation of a STOP (SP) condition.

In order to read multiple bytes, it is necessary to assert the most significant bit of the sub-address field. In other words, SUB(7) must be equal to 1 while SUB(6-0) represents the address of the first register to be read.

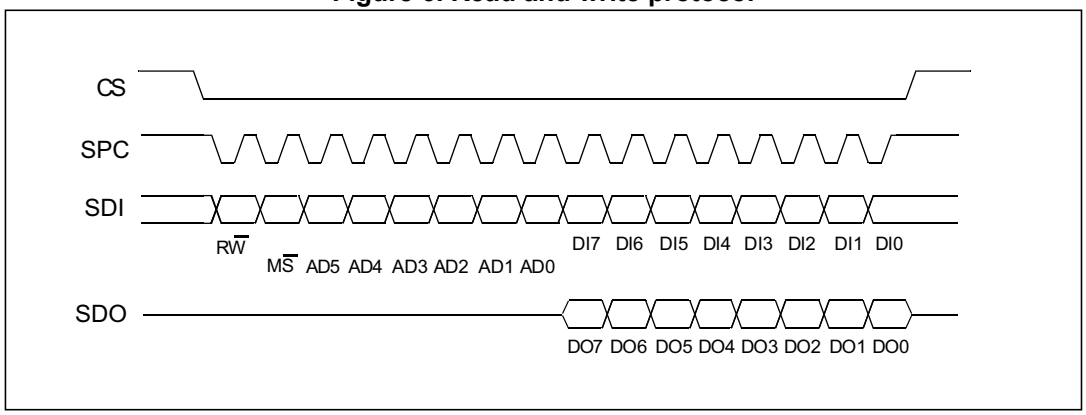
In the presented communication format MAK is Master acknowledge and NMAK is No Master Acknowledge.

6.2 SPI bus interface

The LIS2DH12 SPI is a bus slave. The SPI allows writing to and reading from the registers of the device.

The serial interface interacts with the application using 4 wires: **CS**, **SPC**, **SDI** and **SDO**.

Figure 6. Read and write protocol



CS is the serial port enable and it is controlled by the SPI master. It goes low at the start of the transmission and goes back high at the end. **SPC** is the serial port clock and it is controlled by the SPI master. It is stopped high when **CS** is high (no transmission). **SDI** and **SDO** are respectively the serial port data input and output. These lines are driven at the falling edge of **SPC** and should be captured at the rising edge of **SPC**.

Both the read register and write register commands are completed in 16 clock pulses or in multiples of 8 in case of multiple read/write bytes. Bit duration is the time between two falling edges of **SPC**. The first bit (bit 0) starts at the first falling edge of **SPC** after the falling edge of **CS** while the last bit (bit 15, bit 23, ...) starts at the last falling edge of SPC just before the rising edge of **CS**.

bit 0: R$\overline{W}$ bit. When 0, the data DI(7:0) is written into the device. When 1, the data DO(7:0) from the device is read. In the latter case, the chip will drive **SDO** at the start of bit 8.

bit 1: M$\overline{S}$ bit. When 0, the address will remain unchanged in multiple read/write commands. When 1, the address is auto incremented in multiple read/write commands.

bit 2-7: address AD(5:0). This is the address field of the indexed register.

bit 8-15: data DI(7:0) (write mode). This is the data that is written into the device (MSb first).

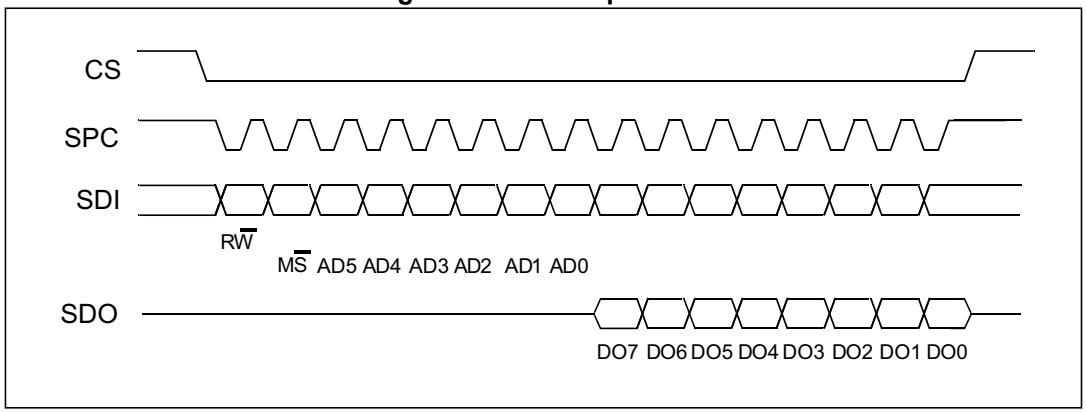
bit 8-15: data DO(7:0) (read mode). This is the data that is read from the device (MSb first).

In multiple read/write commands further blocks of 8 clock periods will be added. When the M$\overline{S}$ bit is ‘0’, the address used to read/write data remains the same for every block. When the M$\overline{S}$ bit is ‘1’, the address used to read/write data is increased at every block.

The function and the behavior of **SDI** and **SDO** remain unchanged.

6.2.1 SPI read

Figure 7. SPI read protocol



The SPI read command is performed with 16 clock pulses. A multiple byte read command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: READ bit. The value is 1.

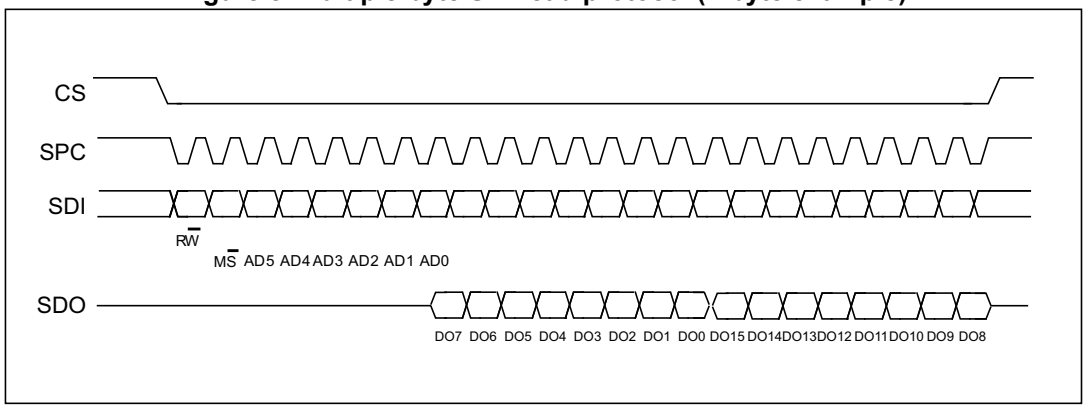
bit 1: MS bit. When 0, does not increment the address; when 1, increments the address in multiple reads.

bit 2-7: address AD(5:0). This is the address field of the indexed register.

bit 8-15: data DO(7:0) (read mode). This is the data that will be read from the device (MSb first).

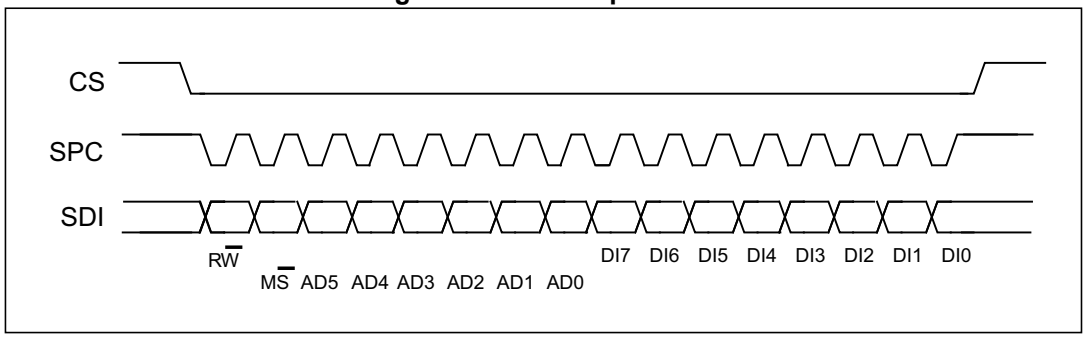
bit 16-... : data DO(...-8). Further data in multiple byte reads.

Figure 8. Multiple byte SPI read protocol (2-byte example)



6.2.2 SPI write

Figure 9. SPI write protocol



The SPI write command is performed with 16 clock pulses. A multiple byte write command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: WRITE bit. The value is 0.

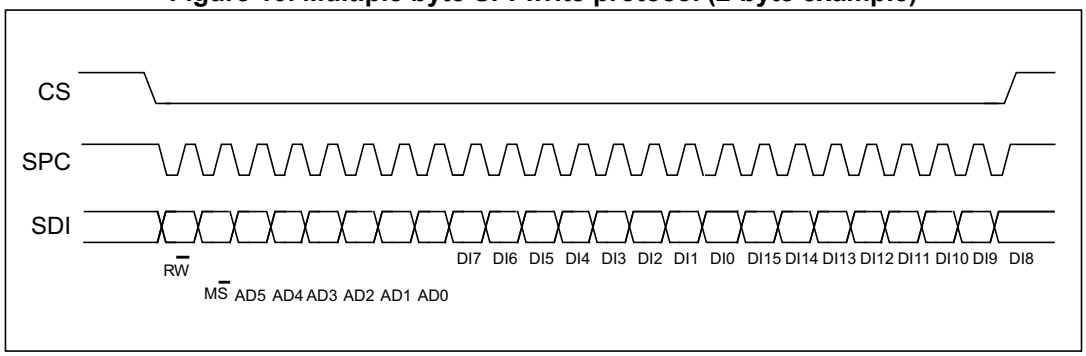
bit 1: MS bit. When 0, does not increment the address; when 1, increments the address in multiple writes.

bit 2 -7: address AD(5:0). This is the address field of the indexed register.

bit 8-15: data DI(7:0) (write mode). This is the data that is written inside the device (MSb first).

bit 16-... : data DI(...-8). Further data in multiple byte writes.

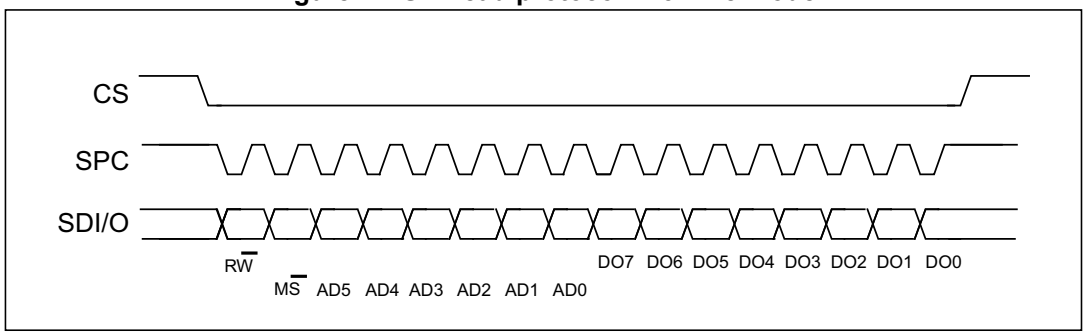
Figure 10. Multiple byte SPI write protocol (2-byte example)



6.2.3 SPI read in 3-wire mode

3-wire mode is entered by setting the SIM bit (SPI serial interface mode selection) to ‘1’ in *CTRL_REG4 (23h)*.

Figure 11. SPI read protocol in 3-wire mode



The SPI read command is performed with 16 clock pulses.

bit 0: READ bit. The value is 1.

bit 1: MS bit. When 0, does not increment the address; when 1, increments the address in multiple reads.

bit 2-7: address AD(5:0). This is the address field of the indexed register.

bit 8-15: data DO(7:0) (read mode). This is the data that is read from the device (MSb first).

The multiple read command is also available in 3-wire mode.

7 Register mapping

The table given below provides a list of the 8-bit registers embedded in the device and the corresponding addresses.

Table 21. Register address map

Name	Type	Register address		Default	Comment
		Hex	Binary		
Reserved	-	00 - 06			Reserved
STATUS_REG_AUX	r	07	000 0111	Output	
Reserved	-	08-0B			Reserved
OUT_TEMP_L	r	0C	000 1100	Output	
OUT_TEMP_H	r	0D	000 1101	Output	
Reserved	-	0E	000 1110		Reserved
WHO_AM_I	r	0F	000 1111	00110011	Dummy register
Reserved	-	10 - 1D			Reserved
CTRL_REG0	rw	1E	001 1110	00010000	
TEMP_CFG_REG	rw	1F	001 1111	00000000	
CTRL_REG1	rw	20	010 0000	00000111	
CTRL_REG2	rw	21	010 0001	00000000	
CTRL_REG3	rw	22	010 0010	00000000	
CTRL_REG4	rw	23	010 0011	00000000	
CTRL_REG5	rw	24	010 0100	00000000	
CTRL_REG6	rw	25	010 0101	00000000	
REFERENCE	rw	26	010 0110	00000000	
STATUS_REG	r	27	010 0111	Output	
OUT_X_L	r	28	010 1000	Output	
OUT_X_H	r	29	010 1001	Output	
OUT_Y_L	r	2A	010 1010	Output	
OUT_Y_H	r	2B	010 1011	Output	
OUT_Z_L	r	2C	010 1100	Output	
OUT_Z_H	r	2D	010 1101	Output	
FIFO_CTRL_REG	rw	2E	010 1110	00000000	
FIFO_SRC_REG	r	2F	010 1111	Output	
INT1_CFG	rw	30	011 0000	00000000	
INT1_SRC	r	31	011 0001	Output	
INT1_THS	rw	32	011 0010	00000000	

Table 21. Register address map (continued)

Name	Type	Register address		Default	Comment
		Hex	Binary		
INT1_DURATION	rw	33	011 0011	00000000	
INT2_CFG	rw	34	011 0100	00000000	
INT2_SRC	r	35	011 0101	Output	
INT2_THS	rw	36	011 0110	00000000	
INT2_DURATION	rw	37	011 0111	00000000	
CLICK_CFG	rw	38	011 1000	00000000	
CLICK_SRC	r	39	011 1001	Output	
CLICK_THS	rw	3A	011 1010	00000000	
TIME_LIMIT	rw	3B	011 1011	00000000	
TIME_LATENCY	rw	3C	011 1100	00000000	
TIME_WINDOW	rw	3D	011 1101	00000000	
ACT_THS	rw	3E	011 1110	00000000	
ACT_DUR	rw	3F	011 1111	00000000	

Registers marked as *Reserved* or not listed in the table above must not be changed. Writing to those registers may cause permanent damage to the device.

The content of the registers that are loaded at boot should not be changed. They contain the factory calibration values. Their content is automatically restored when the device is powered up.

The boot procedure is complete within 5 milliseconds after device power-up.

8 Register description

8.1 STATUS_REG_AUX (07h)

Table 22. STATUS_REG_AUX register

--	TOR	--	--	--	TDA	--	--

Table 23. STATUS_REG_AUX description

TOR	Temperature data overrun. Default value: 0 (0: no overrun has occurred; 1: new temperature data has overwritten the previous data)
TDA	Temperature new data available. Default value: 0 (0: new temperature data is not yet available; 1: new temperature data is available)

8.2 OUT_TEMP_L (0Ch), OUT_TEMP_H (0Dh)

Temperature sensor data. Refer to *Section 3.7: Temperature sensor* for details on how to enable and read the temperature sensor output data.

8.3 WHO_AM_I (0Fh)

Table 24. WHO_AM_I register

0	0	1	1	0	0	1	1

Device identification register.

8.4 CTRL_REG0 (1Eh)

Table 25. CTRL_REG0 register

SDO_PU_DISC	0(1)	0(1)	1(2)	0(1)	0(1)	0(1)	0(1)

1. This bit must be set to 0 for correct operation of the device.
2. This bit must be set to 1 for correct operation of the device.

Table 26. CTRL_REG0 description

SDO_PU_DISC	Disconnect SDO/SA0 pull-up. Default value: 00010000 (0: pull-up connected to SDO/SA0 pin; 1: pull-up disconnected to SDO/SA0 pin)

Note: Leave bits 0 through 6 at the default value in order to ensure correct operation of the device.

8.5 TEMP_CFG_REG (1Fh)

Table 27. TEMP_CFG_REG register

TEMP_EN1	TEMP_EN0	0	0	0	0	0	0

Table 28. TEMP_CFG_REG description

TEMP_EN[1:0]	Temperature sensor (T) enable. Default value: 00 (00: T disabled; 11: T enabled)

8.6 CTRL_REG1 (20h)

Table 29. CTRL_REG1 register

ODR3	ODR2	ODR1	ODR0	LPen	Zen	Yen	Xen

Table 30. CTRL_REG1 description

ODR[3:0]	Data rate selection. Default value: 0000 (0000: power-down mode; others: refer to *Table 31*)
LPen	Low-power mode enable. Default value: 0 (0: high-resolution / normal mode, 1: low-power mode) (Refer to section *3.2.1: High-resolution, normal mode, low-power mode*)
Zen	Z-axis enable. Default value: 1 (0: Z-axis disabled; 1: Z-axis enabled)
Yen	Y-axis enable. Default value: 1 (0: Y-axis disabled; 1: Y-axis enabled)
Xen	X-axis enable. Default value: 1 (0: X-axis disabled; 1: X-axis enabled)

ODR[3:0] is used to set the power mode and ODR selection. The following table indicates the frequency of each combination of ODR[3:0].

Table 31. Data rate configuration

ODR3	ODR2	ODR1	ODR0	Power mode selection
0	0	0	0	Power-down mode
0	0	0	1	HR / Normal / Low-power mode (1 Hz)
0	0	1	0	HR / Normal / Low-power mode (10 Hz)
0	0	1	1	HR / Normal / Low-power mode (25 Hz)
0	1	0	0	HR / Normal / Low-power mode (50 Hz)
0	1	0	1	HR / Normal / Low-power mode (100 Hz)
0	1	1	0	HR / Normal / Low-power mode (200 Hz)
0	1	1	1	HR/ Normal / Low-power mode (400 Hz)
1	0	0	0	Low-power mode (1.620 kHz)
1	0	0	1	HR/ Normal (1.344 kHz); Low-power mode (5.376 kHz)

By design, when the device from high-resolution configuration (HR) is set to power-down mode (PD), it is recommended to read register *REFERENCE (26h)* for a complete reset of the filtering block before switching to normal/high-performance mode again for proper device functionality.

8.7 CTRL_REG2 (21h)

Table 32. CTRL_REG2 register

HPM1	HPM0	HPCF2	HPCF1	FDS	HPCLICK	HP_IA2	HP_IA1

Table 33. CTRL_REG2 description

HPM[1:0]	High-pass filter mode selection. Default value: 00 Refer to *Table 34* for filter mode configuration
HPCF[2:1]	High-pass filter cutoff frequency selection
FDS	Filtered data selection. Default value: 0 (0: internal filter bypassed; 1: data from internal filter sent to output register and FIFO)
HPCLICK	High-pass filter enabled for CLICK function. (0: filter bypassed; 1: filter enabled)
HP_IA2	High-pass filter enabled for AOI function on Interrupt 2. (0: filter bypassed; 1: filter enabled)
HP_IA1	High-pass filter enabled for AOI function on Interrupt 1. (0: filter bypassed; 1: filter enabled)

Table 34. High-pass filter mode configuration

HPM1	HPM0	High-pass filter mode
0	0	Normal mode (reset by reading *REFERENCE (26h)* register)
0	1	Reference signal for filtering
1	0	Normal mode
1	1	Autoreset on interrupt event

8.8 CTRL_REG3 (22h)

Table 35. CTRL_REG3 register

I1_CLICK	I1_IA1	I1_IA2	I1_ZYXDA	0(1)	I1_WTM	I1_OVERRUN	--

1. This bit must be set to ‘0’ for correct operation of the device.

Table 36. CTRL_REG3 description

I1_CLICK	CLICK interrupt on INT1 pin. Default value: 0 (0: disable; 1: enable)
I1_IA1	IA1 interrupt on INT1 pin. Default value: 0 (0: disable; 1: enable)
I1_IA2	IA2 interrupt on INT1 pin. Default value: 0 (0: disable; 1: enable)
I1_ZYXDA	ZYXDA interrupt on INT1 pin. Default value: 0 (0: disable; 1: enable)
I1_WTM	FIFO watermark interrupt on INT1 pin. Default value: 0 (0: disable; 1: enable)
I1_OVERRUN	FIFO overrun interrupt on INT1 pin. Default value: 0 (0: disable; 1: enable)

8.9 CTRL_REG4 (23h)

Table 37. CTRL_REG4 register

BDU	BLE(1)	FS1	FS0	HR	ST1	ST0	SIM

1. The BLE function can be activated only in high-resolution mode

Table 38. CTRL_REG4 description

BDU	Block data update. Default value: 0 (0: continuous update; 1: output registers not updated until MSB and LSB have been read)
BLE	Big/Little Endian data selection. Default value: 0 (0: data LSb at lower address; 1: data MSb at lower address) The BLE function can be activated only in high-resolution mode
FS[1:0]	Full-scale selection. Default value: 00 (00: ±2 *g*; 01: ±4 *g*; 10: ±8 *g*; 11: ±16 *g*)
HR	Operating mode selection (refer to section *3.2.1: High-resolution, normal mode, low-power mode*)(1)
ST[1:0]	Self-test enable. Default value: 00 (00: self-test disabled; other: see *Table 39*)
SIM	SPI serial interface mode selection. Default value: 0 (0: 4-wire interface; 1: 3-wire interface).

1. By design, when the device from high-resolution configuration (HR) is set to power-down mode (PD), it is recommended to read register *REFERENCE (26h)* for a complete reset of the filtering block before switching to normal/high-performance mode again for proper device functionality.

Table 39. Self-test mode configuration

ST1	ST0	Self-test mode
0	0	Normal mode
0	1	Self test 0
1	0	Self test 1
1	1	--

8.10 CTRL_REG5 (24h)

Table 40. CTRL_REG5 register

BOOT	FIFO_EN	--	--	LIR_INT1	D4D_INT1	LIR_INT2	D4D_INT2

Table 41. CTRL_REG5 description

BOOT	Reboot memory content. Default value: 0 (0: normal mode; 1: reboot memory content)
FIFO_EN	FIFO enable. Default value: 0 (0: FIFO disabled; 1: FIFO enabled)
LIR_INT1	Latch interrupt request on *INT1_SRC (31h)*, with *INT1_SRC (31h)* register cleared by reading *INT1_SRC (31h)* itself. Default value: 0. (0: interrupt request not latched; 1: interrupt request latched)
D4D_INT1	4D enable: 4D detection is enabled on INT1 pin when 6D bit on *INT1_CFG (30h)* is set to 1.
LIR_INT2	Latch interrupt request on *INT2_SRC (35h)* register, with *INT2_SRC (35h)* register cleared by reading *INT2_SRC (35h)* itself. Default value: 0. (0: interrupt request not latched; 1: interrupt request latched)
D4D_INT2	4D enable: 4D detection is enabled on INT2 pin when 6D bit on *INT2_CFG (34h)* is set to 1.

8.11 CTRL_REG6 (25h)

Table 42. CTRL_REG6 register

I2_CLICK	I2_IA1	I2_IA2	I2_BOOT	I2_ACT	- -	INT_POLARITY	-

Table 43. CTRL_REG6 description

I2_CLICK	Click interrupt on INT2 pin. Default value: 0 (0: disabled; 1: enabled)
I2_IA1	Enable interrupt 1 function on INT2 pin. Default value: 0 (0: function disabled; 1: function enabled)
I2_IA2	Enable interrupt 2 function on INT2 pin. Default value: 0 (0: function disabled; 1: function enabled)
I2_BOOT	Enable boot on INT2 pin. Default value: 0 (0: disabled; 1: enabled)
I2_ACT	Enable activity interrupt on INT2 pin. Default value: 0 (0: disabled; 1: enabled)
INT_POLARITY	INT1 and INT2 pin polarity. Default value: 0 (0: active-high; 1: active-low)

8.12 REFERENCE (26h)

Table 44. REFERENCE register

Ref7	Ref6	Ref5	Ref4	Ref3	Ref2	Ref1	Ref0

Table 45. REFERENCE description

Ref [7:0]	Reference value for interrupt generation. Default value: 0

8.13 STATUS_REG (27h)

Table 46. STATUS_REG register

ZYXOR	ZOR	YOR	XOR	ZYXDA	ZDA	YDA	XDA

Table 47. STATUS_REG description

ZYXOR	X-, Y- and Z-axis data overrun. Default value: 0 (0: no overrun has occurred; 1: a new set of data has overwritten the previous set)
ZOR	Z-axis data overrun. Default value: 0 (0: no overrun has occurred; 1: new data for the Z-axis has overwritten the previous data)
YOR	Y-axis data overrun. Default value: 0 (0: no overrun has occurred; 1: new data for the Y-axis has overwritten the previous data)
XOR	X-axis data overrun. Default value: 0 (0: no overrun has occurred; 1: new data for the X-axis has overwritten the previous data)
ZYXDA	X-, Y- and Z-axis new data available. Default value: 0 (0: a new set of data is not yet available; 1: a new set of data is available)
ZDA	Z-axis new data available. Default value: 0 (0: new data for the Z-axis is not yet available; 1: new data for the Z-axis is available)
YDA	Y-axis new data available. Default value: 0 (0: new data for the Y-axis is not yet available; 1: new data for the Y-axis is available)

8.14 OUT_X_L (28h), OUT_X_H (29h)

X-axis acceleration data. The value is expressed as two's complement left-justified.
Please refer to *Section 3.2.1: High-resolution, normal mode, low-power mode*.

8.15 OUT_Y_L (2Ah), OUT_Y_H (2Bh)

Y-axis acceleration data. The value is expressed as two's complement left-justified.
Please refer to *Section 3.2.1: High-resolution, normal mode, low-power mode*.

8.16 OUT_Z_L (2Ch), OUT_Z_H (2Dh)

Z-axis acceleration data. The value is expressed as two's complement left-justified.
Please refer to *Section 3.2.1: High-resolution, normal mode, low-power mode*.

8.17 FIFO_CTRL_REG (2Eh)

Table 48. FIFO_CTRL_REG register

FM1	FM0	TR	FTH4	FTH3	FTH2	FTH1	FTH0

Table 49. FIFO_CTRL_REG description

FM[1:0]	FIFO mode selection. Default value: 00 (see *Table 50*)
TR	Trigger selection. Default value: 0 0: trigger event allows triggering signal on INT1 1: trigger event allows triggering signal on INT2
FTH[4:0]	Default value: 00000

Table 50. FIFO mode configuration

FM1	FM0	FIFO mode
0	0	Bypass mode
0	1	FIFO mode
1	0	Stream mode
1	1	Stream-to-FIFO mode

8.18 FIFO_SRC_REG (2Fh)

Table 51. FIFO_SRC_REG register

WTM	OVRN_FIFO	EMPTY	FSS4	FSS3	FSS2	FSS1	FSS0

Table 52. FIFO_SRC_REG description

WTM	WTM bit is set high when FIFO content exceeds watermark level
OVRN_FIFO	OVRN bit is set high when FIFO buffer is full; this means that the FIFO buffer contains 32 unread samples. At the following ODR a new sample set replaces the oldest FIFO value. The OVRN bit is set to 0 when the first sample set has been read
EMPTY	EMPTY flag is set high when all FIFO samples have been read and FIFO is empty
FSS [4:0]	FSS [4:0] field always contains the current number of unread samples stored in the FIFO buffer. When FIFO is enabled, this value increases at ODR frequency until the buffer is full, whereas, it decreases every time one sample set is retrieved from FIFO

8.19 INT1_CFG (30h)

Table 53. INT1_CFG register

AOI	6D	ZHIE/	ZLIE	YHIE	YLIE	XHIE	XLIE

Table 54. INT1_CFG description

AOI	And/Or combination of interrupt events. Default value: 0. Refer to *Table 55*
6D	6-direction detection function enabled. Default value: 0. Refer to *Table 55*
ZHIE	Enable interrupt generation on Z high event or on direction recognition. Default value: 0 (0: disable interrupt request;1: enable interrupt request)
ZLIE	Enable interrupt generation on Z low event or on direction recognition. Default value: 0 (0: disable interrupt request;1: enable interrupt request)
YHIE	Enable interrupt generation on Y high event or on direction recognition. Default value: 0 (0: disable interrupt request; 1: enable interrupt request.)
YLIE	Enable interrupt generation on Y low event or on direction recognition. Default value: 0 (0: disable interrupt request; 1: enable interrupt request.)
XHIE	Enable interrupt generation on X high event or on direction recognition. Default value: 0 (0: disable interrupt request; 1: enable interrupt request.)
XLIE	Enable interrupt generation on X low event or on direction recognition. Default value: 0 (0: disable interrupt request; 1: enable interrupt request.)

The content of this register is loaded at boot.

A write operation to this address is possible only after system boot.

Table 55. Interrupt mode

AOI	6D	Interrupt mode
0	0	OR combination of interrupt events
0	1	6-direction movement recognition
1	0	AND combination of interrupt events
1	1	6-direction position recognition

The difference between AOI-6D = ‘01’ and AOI-6D = ‘11’.

AOI-6D = ‘01’ is movement recognition. An interrupt is generated when the orientation moves from an unknown zone to a known zone. The interrupt signal remains for a duration ODR.

AOI-6D = ‘11’ is direction recognition. An interrupt is generated when the orientation is inside a known zone. The interrupt signal remains while the orientation is inside the zone.

8.20 INT1_SRC (31h)

Table 56. INT1_SRC register

0	IA	ZH	ZL	YH	YL	XH	XL

Table 57. INT1_SRC description

IA	Interrupt active. Default value: 0 (0: no interrupt has been generated; 1: one or more interrupts have been generated)
ZH	Z high. Default value: 0 (0: no interrupt, 1: Z high event has occurred)
ZL	Z low. Default value: 0 (0: no interrupt; 1: Z low event has occurred)
YH	Y high. Default value: 0 (0: no interrupt, 1: Y high event has occurred)
YL	Y low. Default value: 0 (0: no interrupt, 1: Y low event has occurred)
XH	X high. Default value: 0 (0: no interrupt, 1: X high event has occurred)
XL	X low. Default value: 0 (0: no interrupt, 1: X low event has occurred)

Interrupt 1 source register. Read-only register.

Reading at this address clears the *INT1_SRC (31h)* IA bit (and the interrupt signal on the INT1 pin) and allows the refresh of data in the *INT1_SRC (31h)* register if the latched option was chosen.

8.21 INT1_THS (32h)

Table 58. INT1_THS register

0	THS6	THS5	THS4	THS3	THS2	THS1	THS0

Table 59. INT1_THS description

THS[6:0]	Interrupt 1 threshold. Default value: 000 0000 1 LSb = 16 m*g* @ FS = 2 *g* 1 LSb = 32 m*g* @ FS = 4 *g* 1 LSb = 62 m*g* @ FS = 8 *g* 1 LSb = 186 m*g* @ FS = 16 *g*

8.22 INT1_DURATION (33h)

Table 60. INT1_DURATION register

0	D6	D5	D4	D3	D2	D1	D0

Table 61. INT1_DURATION description

D[6:0]	Duration value. Default value: 000 0000 1 LSb = 1/ODR

The **D[6:0]** bits set the minimum duration of the Interrupt 2 event to be recognized. Duration steps and maximum values depend on the ODR chosen.

Duration time is measured in N/ODR, where N is the content of the duration register.

8.23 INT2_CFG (34h)

Table 62. INT2_CFG register

AOI	6D	ZHIE	ZLIE	YHIE	YLIE	XHIE	XLIE

Table 63. INT2_CFG description

AOI	AND/OR combination of interrupt events. Default value: 0 (see *Table 64*)
6D	6-direction detection function enabled. Default value: 0. Refer to *Table 64.*
ZHIE	Enable interrupt generation on Z high event. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
ZLIE	Enable interrupt generation on Z low event. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value lower than preset threshold)
YHIE	Enable interrupt generation on Y high event. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
YLIE	Enable interrupt generation on Y low event. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value lower than preset threshold)
XHIE	Enable interrupt generation on X high event. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
XLIE	Enable interrupt generation on X low event. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value lower than preset threshold)

The content of this register is loaded at boot.

A write operation to this address is possible only after system boot.

Table 64. Interrupt mode

AOI	6D	Interrupt mode
0	0	OR combination of interrupt events
0	1	6-direction movement recognition
1	0	AND combination of interrupt events
1	1	6-direction position recognition

The difference between AOI-6D = ‘01’ and AOI-6D = ‘11’.

AOI-6D = ‘01’ is movement recognition. An interrupt is generated when the orientation moves from an unknown zone to a known zone. The interrupt signal remains for a duration ODR.

AOI-6D = ‘11’ is direction recognition. An interrupt is generated when the orientation is inside a known zone. The interrupt signal remains while the orientation is inside the zone.

8.24 INT2_SRC (35h)

Table 65. INT2_SRC register

0	IA	ZH	ZL	YH	YL	XH	XL

Table 66. INT2_SRC description

IA	Interrupt active. Default value: 0 (0: no interrupt has been generated; 1: one or more interrupts have been generated)
ZH	Z high. Default value: 0 (0: no interrupt, 1: Z high event has occurred)
ZL	Z low. Default value: 0 (0: no interrupt; 1: Z low event has occurred)
YH	Y high. Default value: 0 (0: no interrupt, 1: Y high event has occurred)
YL	Y low. Default value: 0 (0: no interrupt, 1: Y low event has occurred)
XH	X high. Default value: 0 (0: no interrupt, 1: X high event has occurred)
XL	X low. Default value: 0 (0: no interrupt, 1: X low event has occurred)

Interrupt 2 source register. Read-only register.

Reading at this address clears the *INT2_SRC (35h)* IA bit (and the interrupt signal on the INT2 pin) and allows the refresh of data in the *INT2_SRC (35h)* register if the latched option was chosen.

8.25 INT2_THS (36h)

Table 67. INT2_THS register

0	THS6	THS5	THS4	THS3	THS2	THS1	THS0

Table 68. INT2_THS description

THS[6:0]	Interrupt 2 threshold. Default value: 000 0000 1 LSb = 16 m*g* @ FS = 2 *g* 1 LSb = 32 m*g* @ FS = 4 *g* 1 LSb = 62 m*g* @ FS = 8 *g* 1 LSb = 186 m*g* @ FS = 16 *g*

8.26 INT2_DURATION (37h)

Table 69. INT2_DURATION register

0	D6	D5	D4	D3	D2	D1	D0

Table 70. INT2_DURATION description

D[6:0]	Duration value. Default value: 000 0000 1 LSb = 1/ODR(1)

1. Duration time is measured in N/ODR, where N is the content of the duration register.

The **D[6:0]** bits set the minimum duration of the Interrupt 2 event to be recognized. Duration time steps and maximum values depend on the ODR chosen.

8.27 CLICK_CFG (38h)

Table 71. CLICK_CFG register

--	--	ZD	ZS	YD	YS	XD	XS

Table 72. CLICK_CFG description

ZD	Enable interrupt double-click on Z-axis. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
ZS	Enable interrupt single-click on Z-axis. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
YD	Enable interrupt double-click on Y-axis. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
YS	Enable interrupt single-click on Y-axis. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
XD	Enable interrupt double-click on X-axis. Default value: 0 (0: disable interrupt request; 1: enable interrupt request on measured accel. value higher than preset threshold)
XS	Enable interrupt single-click on X-axis. Default value: 0 (0: disable interrupt request; 1 : enable interrupt request on measured accel. value higher than preset threshold)

8.28 CLICK_SRC (39h)

Table 73. CLICK_SRC register

	IA	DClick	SClick	Sign	Z	Y	X

Table 74. CLICK_SRC description

IA	Interrupt active. Default value: 0 (0: no interrupt has been generated; 1: one or more interrupts have been generated)
DClick	Double-click enable. Default value: 0 (0: double-click detection disabled, 1: double-click detection enabled)
SClick	Single-click enable. Default value: 0 (0: single-click detection disabled, 1: single-click detection enabled)
Sign	Click sign. 0: positive detection, 1: negative detection
Z	Z click detection. Default value: 0 (0: no interrupt, 1: Z high event has occurred)
Y	Y click detection. Default value: 0 (0: no interrupt, 1: Y high event has occurred)
X	X click detection. Default value: 0 (0: no interrupt, 1: X high event has occurred)

8.29 CLICK_THS (3Ah)

Table 75. CLICK_THS register

LIR_Click	Ths6	Ths5	Ths4	Ths3	Ths2	Ths1	Ths0

Table 76. CLICK_THS register description

LIR_Click	If the LIR_Click bit is not set, the interrupt is kept high for the duration of the latency window. If the LIR_Click bit is set, the interrupt is kept high until the *CLICK_SRC (39h)* register is read.
Ths[6:0]	Click threshold. Default value: 000 0000

8.30 TIME_LIMIT (3Bh)

Table 77. TIME_LIMIT register

-	TLI6	TLI5	TLI4	TLI3	TLI2	TLI1	TLI0

Table 78. TIME_LIMIT description

TLI[6:0]	Click time limit. Default value: 000 0000

8.31 TIME_LATENCY (3Ch)

Table 79. TIME_LATENCY register

TLA7	TLA6	TLA5	TLA4	TLA3	TLA2	TLA1	TLA0

Table 80. TIME_LATENCY description

TLA[7:0]	Click time latency. Default value: 0000 0000

8.32 TIME_WINDOW (3Dh)

Table 81. TIME_WINDOW register

TW7	TW6	TW5	TW4	TW3	TW2	TW1	TW0

Table 82. TIME_WINDOW description

TW[7:0]	Click time window

8.33 ACT_THS (3Eh)

Table 83. ACT_THS register

--	Acth6	Acth5	Acth4	Acth3	Acth2	Acth1	Acth0

Table 84. ACT_THS description

Acth[6:0]	Sleep-to-wake, return-to-sleep activation threshold in low-power mode 1 LSb = 16 m*g* @ FS = 2 *g* 1 LSb = 32 m*g* @ FS = 4 *g* 1 LSb = 62 m*g* @ FS = 8 *g* 1 LSb = 186 m*g* @ FS = 16 *g*

8.34 ACT_DUR (3Fh)

Table 85. ACT_DUR register

ActD7	ActD6	ActD5	ActD4	ActD3	ActD2	ActD1	ActD0

Table 86. ACT_DUR description

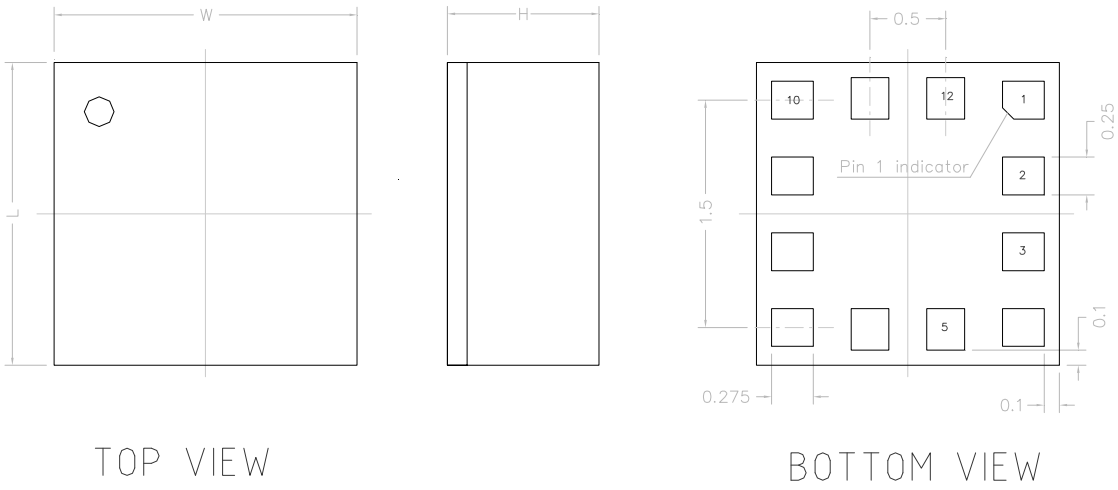
ActD[7:0]	Sleep-to-wake, return-to-sleep duration. 1 LSb = (8*1[LSb]+1)/ODR

9 Package information

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: www.st.com.
ECOPACK is an ST trademark.

9.1 LGA-12 package information

Figure 12. LGA-12: package outline and mechanical data



Dimensions are in millimeter unless otherwise specified
General Tolerance is +/-0.1mm unless otherwise specified

OUTER DIMENSIONS

ITEM	DIMENSION [mm]	TOLERANCE [mm]
Length [L]	2.00	±0.15
Width [W]	2.00	±0.15
Height [H]	1.0	MAX

8365767_2

9.2 LGA-12 packing information

Figure 13. Carrier tape information for LGA-12 package

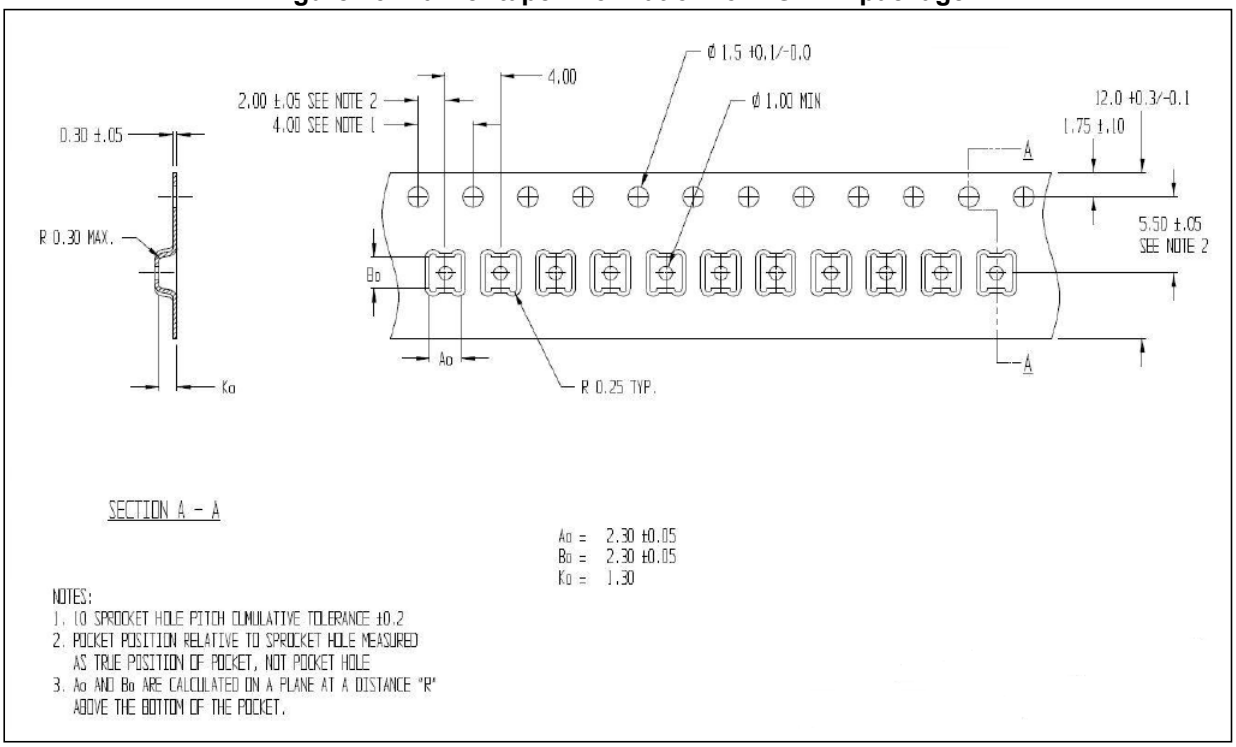


Figure 14. LGA-12 package orientation in carrier tape

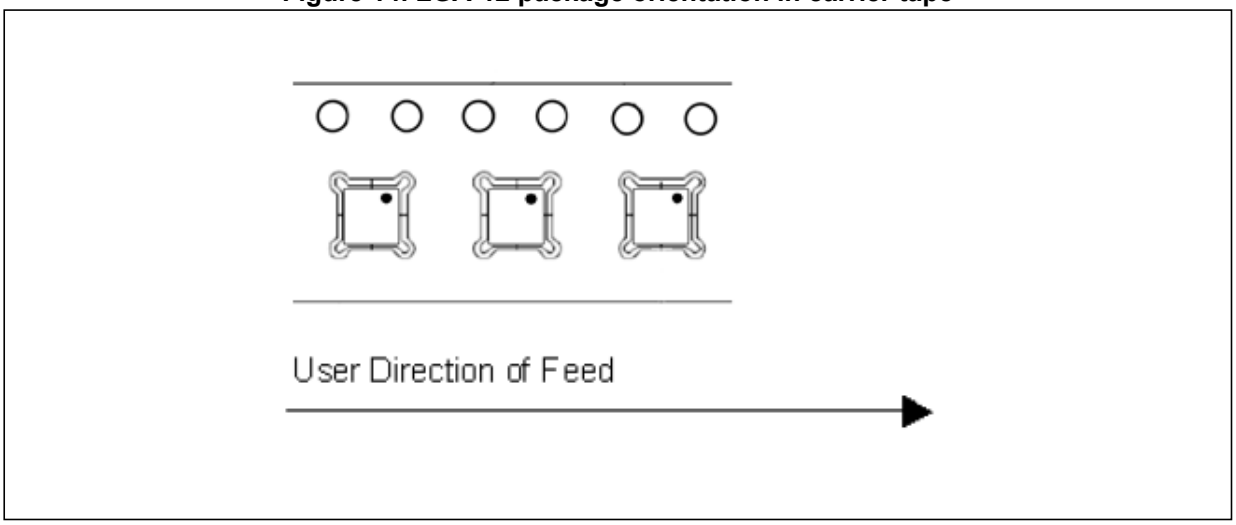


Figure 15. Reel information for carrier tape of LGA-12 package

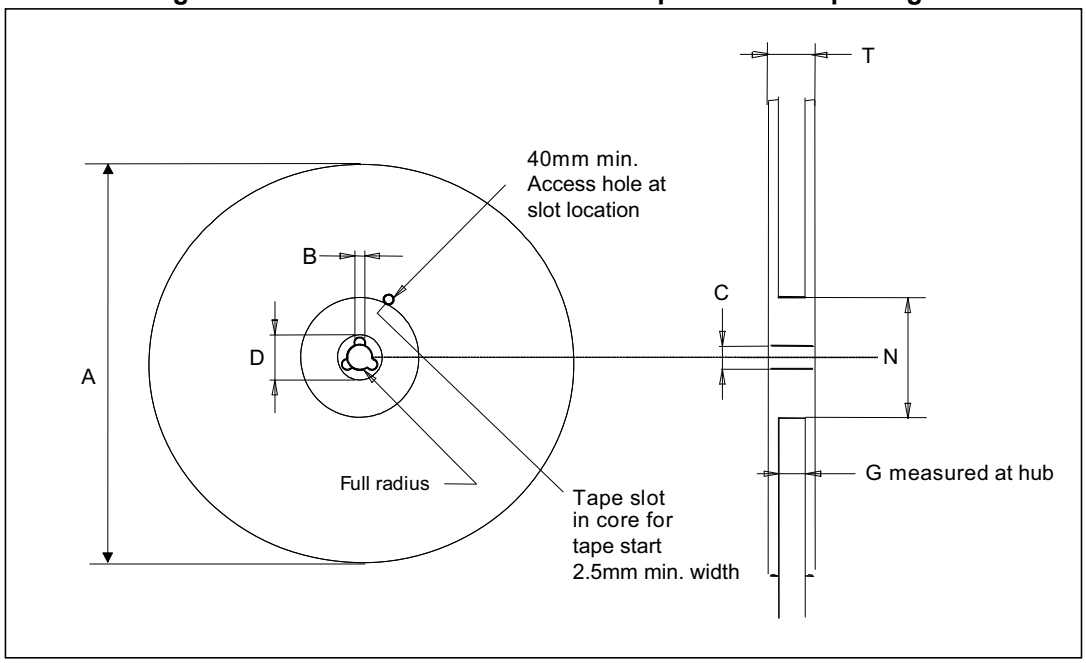


Table 87. Reel dimensions for carrier tape of LGA-12 package

Reel dimensions (mm)	
A (max)	330
B (min)	1.5
C	13 ±0.25
D (min)	20.2
N (min)	60
G	12.4 +2/-0
T (max)	18.4

10 Revision history

Table 88. Document revision history

Date	Revision	Changes
06-Aug-2013	1	Initial release
23-Oct-2015	2	Added *Section 9.2: LGA-12 packing information*
02-Dec-2015	3	Corrected orientation of X and Y axes in *Figure 2: Pin connections* Corrected chamfer of pin 1 in *Figure 5: LIS2DH12 electrical connections* Updated default values in *Table 21: Register address map* Modified register 0Eh to “Reserved” in *Table 21* and removed from *Section 8: Register description* Corrected typo in *Table 87: Reel dimensions for carrier tape of LGA-12 package*
23-May-2016	4	Updated *Table 1: Device summary* Updated A_{POW} and A_{UNP} in *Table 9: Absolute maximum ratings*
08-Nov-2016	5	Updated *Table 2: Pin description* Added *Table 3: Internal pull-up values (typ.) for SDO/SA0 pin* Updated *Section 3.7: Temperature sensor* Added *Table 13: Internal pin status* Updated *CTRL_REG2 (21h)* Updated *Section 8: Register description* Minor textual updates
05-May-2017	6	Added footnote *1* to *Table 10* concerning power-down from high-resolution mode Updated *CTRL_REG1 (20h)* Added footnote *1* to HR bit description in *CTRL_REG4 (23h)*

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá