UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

# Comparativa, instalación y testeo de un Sistema Operativo de bajo consumo y su aplicación en el Internet de las Cosas

**Autor: Alberto Flores Quintanilla**

**Tutor: Miguel Ángel Mateo Pla**

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-18

Valencia, 11 de septiembre de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación

Universitat Politècnica de València

Edificio 4D. Camino de Vera, s/n, 46022 Valencia

Tel. +34 96 387 71 90, ext. 77190

**www.etsit.upv.es**

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

## Resumen

El objetivo de este Trabajo de Fin de Grado (TFG) es realizar un estudio comparativo de diferentes Sistemas Operativos de bajo consumo y su aplicación en el Internet de las Cosas. Para ello, realizamos una búsqueda en internet de las principales soluciones software, las comparamos, seleccionamos una de ellas y analizamos su funcionamiento en un hardware compatible y asequible que permite llevar a cabo una serie de pruebas de rendimiento para determinar su utilidad dentro del ámbito del Internet de las Cosas. Adicionalmente se realiza una aplicación simple que permite darle utilidad práctica a este estudio. Por último, se expone su utilidad comercial como producto de consumo mediante posibles ampliaciones de este proyecto.

**Palabras clave:** IoT, Internet de las Cosas, Electrónica, Sistema Operativo, Open Source, Bajo consumo, Sensor, Mbed, STMicroelectronics.

## Resum

El propòsit d'aquest Treball de Fi de Grau és la realització d'un estudi comparatiu de diferents Sistemes Operatius de baix consum i la seua aplicació en l'Internet de les Coses. Per fer-ho, realitzem una recerca a internet de les principals solucions software, les comparem, seleccionem una i analitzem el seu funcionament en un hardware compatible i assequibe per tal de provar el seu rendiment i utilitat en l'àmbit de l'Internet de les Coses. Addicionalment, es realitza una aplicació simple que permet donar-li utilitat pràctica a aquest estudi. Finalment, el document exposa la utilitat comercial com a producte de consum mitjançant possibles ampliacions d'aquest projecte.

**Paraules clau:** IoT, Internet de les Coses, Electrònica, Sistema Operatiu, Open Source, Baix consum, Sensor, Mbed, STMicroelectronics.

# Abstract

The main purpose of this Work is to make a comparative study of Low Power Operative System and its application in the Internet of Things. For this, a certain software present on the internet and a compatible and affordable hardware are selected from a list of popular software solutions found in Internet in order to test its performance and usefulness in the Internet of Things. Additionally, a simple application is made to give practical use of this study. Finally, the document exposes the commercial utility as a consumer product through possible extensions of this project.

**Key words:** IoT, Internet of Things, Electronics, Operative System, Open Source, Low Power, Sensor, Mbed, STMicroelectronics.

# Índice:

# Índice de ilustraciones:



# Índice de tablas:

# 1. Introducción:

En este proyecto se quiere realizar un estudio del panorama actual de los *Sistemas Operativos de Tiempo Real* (*SOTR*), *Open Source* y de bajo consumo aplicados al *Internet de las Cosas* o *IoT*, por su abreviatura de *Internet of Things* en inglés, y la instalación y testeo de uno de ellos en un hardware compatible y asequible para evaluar su aplicación en este ámbito. Para ello se realizará una búsqueda de las diferentes opciones disponibles en Internet tanto a nivel de software como de hardware para averiguar cuáles son los más representativos en esta materia.

Debido a la gran cantidad de soluciones que reúnen las características que buscamos, basaremos nuestra selección en función de los siguientes aspectos:

- **Popularidad**: Los proyectos que gocen de mayor popularidad también serán probablemente los más revisados a nivel de desarrollo, con soporte para multitud de componentes hardware, y también por parte del fabricante.

- **Madurez**: Si la popularidad de un *SOTR* no implica un desarrollo maduro capaz de ofrecer soluciones estables y seguras dentro del ámbito del *IoT*, bien por una reciente aparición en el mercado o bien por su obsolescencia con el paso del tiempo, será descartado. Si ofrece soporte para muchos tipos de hardware, pero su avance es lento y no tenemos capacidad de maniobrar a otros productos del mismo fabricante para salir de un apuro, quizá sea mejor optar por otra solución software.

- **Soporte**: Como ya hemos mencionado previamente, cuanto mayor sea la ayuda ofrecida por los desarrolladores del *SOTR*, de la comunidad o de un determinado fabricante de hardware, más probabilidades tendremos de resolver los problemas que puedan aparecer a la hora de acometer nuestro proyecto. La abundancia de tutoriales u otro tipo de ayuda será siempre bien recibido.

En lo que respecta a la elección del fabricante, los aspectos a tener en cuenta serán los siguientes:

- **Soporte**: Una buena combinación software más hardware significa poder disponer de manuales, tutoriales o ejemplos que nos permitan aprovechar todos los recursos que nos ofrece cada uno, como ya hemos dicho previamente. Tanto si es el *SOTR* el que ofrece el soporte para determinado hardware, como si es su fabricante el que nos brinda esas facilidades, será un buen candidato para nuestro proyecto.

- **Disponibilidad**: El fabricante deberá ofrecer hardware no obsoleto, al menos a corto plazo, y un stock en distribuidores reconocidos que permita ampliar nuestro proyecto en el futuro, si así lo deseamos.

- **Precio**: Evidentemente, el coste de nuestro proyecto es un aspecto importante a la hora de escoger el hardware final. De ello dependerá la viabilidad y la escalabilidad de nuestro proyecto. Realizaremos un pequeño cálculo del coste de una cantidad considerable de dispositivos, en el caso de que se quisiera llegar a comercializar un producto parecido al ejecutado en este proyecto.

# 2. Objetivos:

Definir los objetivos principales y separarlos de los secundarios o menos importantes nos ayudará a establecer y organizar plazos para su cumplimiento, así como para determinar qué es importante y qué no lo es a la hora de acometer nuestro proyecto. Para ello, se definen los siguientes objetivos:

1. Analizar el estado actual de los principales *SOTR* con soporte para el *IoT*.
2. Seleccionar la solución o soluciones que mejor se ajustan a nuestro proyecto.
3. Analizar el hardware disponible para nuestro software y seleccionar también el más apropiado, en base a una aplicación o aplicaciones determinadas.
4. Comprobar que la combinación del *SOTR* y el *hardware* dispone de soporte, stock considerable en un distribuidor de confianza, tutoriales y ayuda para acometer nuestro proyecto.
5. Testear el *SOTR* en el hardware seleccionado y ejecutar una serie de pruebas básicas que nos ayuden a determinar el funcionamiento de nuestro sistema, como por ejemplo un pequeño programa que utilice al menos una parte de las características que ofrece el hardware.
6. Evaluar aspectos como el rendimiento, sus posibles aplicaciones, ampliaciones futuras, problemas que han surgido, etc.

## 2.1 Objetivos secundarios o deseables:

1. Utilizar múltiples nodos para testear la escalabilidad de la red.
2. Elaborar una aplicación real con utilidad práctica dentro del marco del Internet de las Cosas.
3. Elaborar una aplicación en Android para interactuar con el dispositivo montado.

# 3. Metodología:

## 3.1 Gestión del proyecto

Para acometer las tareas de este proyecto hubo que compaginar con prácticas en empresa durante la práctica totalidad del tiempo transcurrido, por lo que la duración de este proyecto se ha visto afectada en cuanto a disponibilidad horaria para dedicarse al proyecto. Ha sido necesario dedicar una media de dos o tres horas por día de lunes a viernes durante varios meses.

## 3.2 Distribución de tareas

Para la elaboración de este proyecto y de la presente memoria, se ha procedido a realizar un desglose de este en multitud de pasos:

1. Preparación de la memoria.
2. Definir *IoT* y *Sistema Operativo de Tiempo Real* (SOTR).
3. Búsqueda y comparativa de *SOTR* disponibles en la red, a ser posible de tipo *Open Source*.
4. Búsqueda y aprendizaje de las tecnologías de uso habituales en el ámbito del *IoT*.
5. Selección de uno o más Sistemas Operativos según los aspectos mencionados anteriormente.
6. Definición de las tecnologías encontradas.
7. Búsqueda y selección de fabricantes de hardware que cumplan con los requisitos establecidos anteriormente.
8. Definición de las características que ofrece. Conexiones, alimentación, montaje etc.
9. Recopilación de la documentación disponible tanto del software como del hardware.
10. Puesta a punto del hardware. Actualización del firmware.
11. Análisis del entorno de programación y el resto de las herramientas que ofrece el SOTR.
12. Diseño de pruebas básicas para test.
13. Comprobación del correcto funcionamiento del sistema y comparativa entre los distintos Sistemas Operativos si los hubiera.

14. Problemas encontrados y cómo solucionarlos en la medida de lo posible.
15. Planteamiento de una aplicación práctica dentro del marco del Internet de las Cosas.
16. Revisión de los objetivos alcanzables.
17. Conclusiones y ampliaciones futuras del proyecto
18. Redacción de la memoria.

## 3.3 Diagrama temporal

Para realizar un seguimiento temporal de los apartados enumerados en el punto anterior realizamos un diagrama temporal con una estimación de tiempo a emplear en cada uno de ellos. Conforme se completaron las tareas anotaremos la duración aproximada real y evaluaremos los motivos que han llevado a adelantar o a retrasar objetivos. A continuación, se presenta una tabla que resume este seguimiento:

| Tarea | Duración estimada (h) | Duración real (h) |
|---|---|---|
| Preparación de la memoria | 2 | 2 |
| Definición de IoT y SOTR | 2 | 4’5 |
| Búsqueda y comparativa SOTR | 15 | 29 |
| Tecnologías del IoT | 3 | 3’5 |
| Selección del software | 3 | 7 |
| Búsqueda y comparativa de hardware | 15 | 23 |
| Selección y compra del hardware | 3 días | 6 días* |
| Aprendizaje de las características hardware | 4 | 4 |
| Recopilación de la documentación | 2 | 2 |
| Puesta a punto del hardware | 3 | 30 |
| Aprendizaje del entorno de trabajo | 2 | 2 |
| Búsqueda y diseño de pruebas básicas | 7 | 8 |
| Redacción de la memoria | 25 | 45 |

**Tabla 1-. Tabla de duración de tareas**

La primera de las tareas planteadas, la preparación de la memoria, tuvo una duración acorde con lo previsto debido a que se realizó en una reunión con el tutor para planear el trabajo. En cuanto a la definición de los términos *IoT* y *SOTR*, la duración fue un poco mayor debido a la pequeña dificultad de encontrar unas definiciones de nuestro agrado. Cuando llegó el momento de buscar información sobre los *SOTR* disponibles, la gran cantidad de información encontrada y las tecnologías relacionadas con el Internet de las Cosas fue tal que sobrepasó con creces las expectativas temporales. Fueron necesarias casi tres semanas para recopilar una buena cantidad de recursos de Internet, clasificar la información en una hoja Excel y organizar los enlaces para tenerla más accesible desde el navegador web, llegando a acumular más de cien.

En el momento de enumerar las características de cada una de las opciones hardware disponibles se recurrió a la misma hoja en Excel con las características del software, de manera que pudiéramos asociar ventajas e inconvenientes de multitud de combinaciones software más hardware. Se tuvo que priorizar en cuanto a relevancia dentro del ámbito en el que se desarrolla este proyecto, y escoger una de las opciones basándonos tanto en criterios técnicos como personales. Con todo ello, el tiempo empleado en hacer una criba de los menos importantes llevó también bastante más tiempo de lo esperado.

A la hora de adquirir el hardware a través de un distribuidor de confianza, hubo retrasos inesperados debido a mi condición laboral. Si bien el paquete pudo haber sido entregado a los pocos días de haber hecho la compra de los componentes, su recepción se demoró unos días más hasta que coincidió que la entrega tuvo lugar cuando estaba en casa para recibirlo. Casi al mismo tiempo, se recibió el hardware encargado por parte del tutor.

Una vez que comienza el trabajo con el material ya en nuestras manos el tiempo dedicado a recopilar los tutoriales y familiarizarnos con el hardware y sus características entra dentro de lo esperado. Del mismo modo sucede con la documentación del software y el entorno de trabajo.

Sin embargo, en lo que respecta a la puesta a punto del hardware, surgieron ciertos problemas que retrasaron considerablemente el ritmo de trabajo. Estos problemas se explicarán debidamente en los apartados correspondientes a lo largo de esta memoria.

Por último, la redacción de la memoria llevó casi el doble de tiempo de lo esperado debido a diversos factores. El primero de ellos fue la intención inicial de redactarla usando un editor de látex. El aprendizaje fue lento y dilatado en el tiempo, pero al final se desechó la idea y se optó por Microsoft Office 365. En segundo lugar, los retrasos que se produjeron con el uso de determinado hardware provocaron un avance nulo a lo largo de una semana aproximadamente.

Sin los retrasos sufridos muy probablemente se habrían logrado los objetivos planteados, e incluso probablemente haber esbozado cómo abordar los objetivos secundarios o deseables.

# 4. Desarrollo y resultados:

En este apartado empezaremos por explicar algunos conceptos que nos ayudarán a entender mejor el desarrollo de este proyecto. Conforme vayamos hablando sobre protocolos de comunicación, tecnologías relacionadas, empresas etc., presentes en nuestro proyecto, las iremos explicando con detenimiento. A continuación, hablaremos sobre el proceso de búsqueda, selección y valoración del SOTR y el hardware que emplearemos. Posteriormente hablaremos sobre los problemas que hayan surgido y sus posibles soluciones. Por último, se explorarán diferentes ampliaciones o salidas comerciales para un proyecto de estas características.

## 4.1 Definiciones:

Debido a que nuestro proyecto trata sobre el estudio de un *Sistema Operativo de Tiempo Real*, a ser posible *Open Source*, la mejor manera de comenzar es explicando con detenimiento estos conceptos.

### 4.1.1 *Sistema Operativo de Tiempo Real:*

Para definir correctamente este concepto, empezaremos por definir primero qué es un Sistema Operativo, para luego matizar el apellido "*de tiempo real*". Así pues, entendemos como Sistema Operativo al programa que actúa como intermediario entre el usuario y el hardware para ejecutar programas que resuelvan de forma fácil los problemas del primero, dándole utilidad y eficiencia al segundo (Silberschatz, Galvin, Gagne, & & Silberschatz, 1998). Gestiona el hardware de manera que el usuario acceda de forma correcta y segura a los recursos que el hardware ofrece, sin comprometer la integridad del sistema.

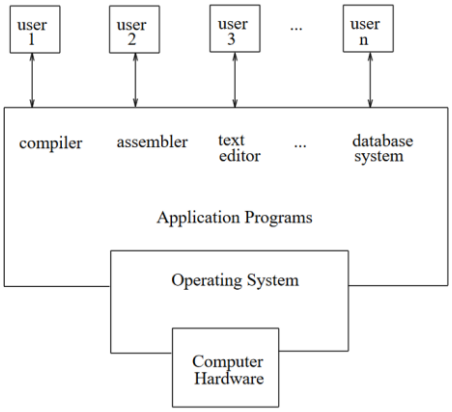



**Ilustración 1.- Esquema que representa la interacción usuario y hardware**

En lo que respecta a la etiqueta "*de tiempo real*", este concepto se aplica, en palabras de Alfonso Crespo y Alejandro Alonso, a los "*sistemas informáticos en los que la respuesta de la aplicación ante estímulos externos debe realizarse dentro de un plazo de tiempo establecido. La predictibilidad del tiempo de respuesta determina que el sistema será capaz de ofrecer una respuesta correcta ante la llegada de un determinado estímulo en un tiempo acotado.*" Teniendo en cuenta el tratamiento secuencial de los eventos por parte de un sistema operativo, y su capacidad en tiempo de respuesta ante estos eventos, podemos distinguir entre críticos y acríticos. En los primeros las acciones del sistema deben producirse obligatoriamente dentro del plazo especificado. Si se producen fallos de plazo, el sistema puede dejar de funcionar. Esta situación se da, por ejemplo, en sistemas aeronáuticos, de control de satélites etc. En el segundo tipo, sin embargo, se tolera la pérdida puntual de alguno de los plazos, provocando un funcionamiento degradado del sistema, pero se debe asegurar la recuperación automática y acotada de este. Ejemplos de este tipo pueden ser los sistemas multimedia o de transmisión de voz (Crespo & Alonso, 2010).

Como tercera característica, pero no menos importante, el o los candidatos finales para nuestro análisis deberán estar enfocados al bajo consumo, debido al ámbito al que se destina nuestro proyecto y cualquier otro que esté enfocado al *IoT*. Si podemos escoger una combinación software más hardware en la que ambos dispongan de recursos para reducir el consumo, todavía mejor.

### 4.1.2 *Internet of Things:*

El *Internet de las Cosas*, o en inglés *Internet of Things*, es un concepto que se puede definir como la presencia generalizada de dispositivos que, a través de protocolos y conexiones inalámbricas o cableadas eficientes, esquemas de direccionamiento únicos, sensores, hardware económico y la multitud de empresas implicadas tanto nuevas como asentadas, son capaces de interactuar entre sí y cooperar con otros para crear nuevas aplicaciones o servicios. Se crea pues una convergencia entre personas, dispositivos y datos, fomentada por la nube, que crean servicios que no serían obvios sin esta unión (European Research Cluster on the Internet of Things, 2015). Esto hace más necesario obtener grandes cantidades de información en tiempo real y analizarla para obtener algún tipo de beneficio en ciudades, empresas, agricultura, comercio etc. La explosión de soluciones nuevas o ya existentes que se benefician de estar conectados a otros dispositivos y a Internet entra dentro del IoT.

Si hasta hace algunos años lo máximo que vislumbrábamos en cuanto a lo que se define como *hogar conectado* eran comodidades como la de poder abrir y cerrar persianas cuando amanece sin tocar nada o encender el aire acondicionado de forma automática en función de la temperatura del hogar, hoy en día las posibilidades son casi ilimitadas.

En el hogar tenemos frigoríficos que adquieren los alimentos de forma automática o te avisan cuando alguno de estos escasea (Pulido Cañabate, 2016), dispositivos que hacen compras en Amazon cuando los pulsamos (Amazon Inc., s.f.), dispositivos inteligentes de Nest que nos permiten gestionar la temperatura de una casa (Nest Labs, Alphabet Inc., s.f.) o el sistema de seguridad (Nest Labs, Alphabet Inc., s.f.); altavoces inteligentes que se conectan al móvil (Google Inc., s.f.) para no tener que interactuar siempre con los pulgares, y un largo etcétera.


**Ilustración 2.- Amazon Dash Button, Termostato Nest, Altavoz inteligente Google Home.**

No nos olvidemos también del coche inteligente o las ciudades inteligentes. Ya no es extraño ver noticias sobre la monitorización del tráfico, el aire, ruido o incluso personas en las ciudades mediante farolas inteligentes (Pulido Cañabate, 2016), la búsqueda de aparcamiento o gestión del tráfico y su consulta a través de Internet. El *Internet de las Cosas* abarca tanto que incluso esta subcategoría ya tiene nombre propio: *Smartcities*.

En cuanto al coche, podríamos hablar de *CarPlay* de Apple (Apple Inc., s.f.) o *Android Auto* de Google (Google Inc., s.f.), pero también de las soluciones propias que han implementado los fabricantes de vehículos como Mercedes y su sistema *MBUX* (Mercedes-Benz, Daimler AG, s.f.). No hay duda de que en el coche se están gestando una serie de cambios como los que sufrieron los teléfonos móviles hace apenas una década, sacudiendo el mercado y cambiando por completo nuestra interacción y capacidades con éstos.

El sector de la agricultura también ha dado un paso más allá con la gestión de los cultivos mediante dispositivos que miden parámetros decisivos en el proceso de evapotranspiración (López, Chavez, & Sánchez, 2017), drones que patrullan los cultivos en busca de posibles plagas o para evitarlas fumigando de forma más precisa que el tradicional uso de avionetas o costosas máquinas fumigadoras pilotadas por humanos. Este tipo de técnicas permiten que el rendimiento tanto económico como productivo mejore.

No hay duda de que el Internet de las Cosas ha venido para quedarse, y su popularidad seguirá creciendo con el paso del tiempo.

### 4.1.3 *Open Source:*

Este término fue acuñado por la *Open Source Initiative*, una organización sin ánimo de lucro fundada en 1998 por un grupo de programadores que querían promover la adopción de este tipo de licencias. Los criterios que la definen se basaron en las condiciones de uso de la distribución de *Linux* llamada *Debian*. Una licencia de este tipo no sólo implica el acceso al código fuente o al compilado, sino también su modificación y redistribución, incluida la copia original, mediante el mismo tipo de licencia. Sin embargo, no obliga a que los programas que se distribuyan junto con un programa licenciado como *Open Source*, tengan el mismo tipo de licencia.

Esto significa, por tanto, que podemos desarrollar un producto comercial con licencia privativa o no siempre y cuando mantengamos el tipo de licencia original el código que tomemos prestado.

Algunos ejemplos de licencias *Open Source* que nos encontraremos a lo largo de esta memoria son los siguientes (Open Source Initiative, s.f.):

- **Licencia MIT:** Como bien indica su nombre, es originaria del *Instituto Tecnológico de Massachusetts*, *MIT* por sus siglas en inglés. Este tipo de licencia es permisiva, por lo que suele escogerse cuando se buscan pocas limitaciones en la reutilización de un software. Otorga permiso para usar, copiar, modificar, unir, publicar, distribuir, sublicenciar, vender copias del software y a las personas que lo reciben hacer lo propio.
- **Apache License 2.0:** También catalogada como permisiva, esta licencia creada por la *Apache Software Foundation* permite usar, distribuir, modificar y distribuir versiones modificadas del software.
- **BSD:** Otorgada principalmente para los sistemas *Berkeley Software Distribution*, esta licencia permisiva es muy similar a la *licencia MIT*. Más próxima al dominio público que la *GPL*, permite el uso del código en otro software no libre.
- **LGPLv2.1:** Esta licencia fue de las primeras en ser diseñada para garantizar que el software libre permanezca como libre para todo aquel que reciba una copia original o modificada del software bajo esta licencia. La *licencia MIT* es compatible con ella, pero no al contrario. A diferencia de otras licencias, permite cobrar o no por la distribución de

una copia del software. Sus sucesivas versiones respecto de la original de *Richard Stallman* en 1989 han buscado cubrir resquicios legales que a lo largo de los años amenazaban empresas con software privativo a base de disputas por patentes.

## 4.2 Búsqueda y selección del SOTR y el hardware

Pasaremos ahora a describir el proceso de selección tanto del *SOTR* como del hardware, y describiremos las tecnologías involucradas y su importancia para nuestro proyecto a medida que vayan apareciendo. Dentro de cada apartado dedicaremos una sección a describir las elecciones finales que se han hecho tras el proceso de selección que describiremos.

### 4.2.1 *Sistema Operativo:*

En este apartado empezaremos por hablar sobre el proceso de selección del *SOTR Open Source*. Una rápida búsqueda en internet nos arroja multitud de resultados como el amplio listado que aparece en la *Tabla 1* (Osrtos, s.f.). Si bien en algunos casos la continuidad de su desarrollo finalizó hace unos años, los hay que siguen estando muy activos gracias a su comunidad, y también nos sirve como punto de partida para nuestro proyecto. Como se puede comprobar a continuación, existen multitud de soluciones software:

| | | | | |
|---|---|---|---|---|
| FreeRTOS | TizenRT | Trampoline | IntrOS | RT-Thread |
| LiteOS | seL4 | LibreRTOS | Contiki OS | AliOS Things |
| uKOS | TNeo | Mbed OS | eChronos | embox |
| TI-TROS Kernel | ChibiOS/RT | BitThunder | F9 Microkernel | Atomthreads |
| NuttX | scmRTOS | State OS | Stratify OS | RIOT-OS |
| Erika Enterprise | Zephyr | MOE | Mongoose OS | BeRTOS |
| DistortOS | Femto OS | Protothreads | Apache Mynewt | Xenomai |
| RTAI | eCos | MARK3 | Nut/OS | Prex |
| Freescale MQX | TNKernel | uSmartX | DuinOS | FunkOS |
| Nano-RK | RTEMS | Tiny OS | OpenWSN | MansOS |
| Yocto Project | Fusion Embedded | | | |

**Tabla 2. - Ejemplos de Sistemas Operativos encontrados en la red**

Aquí ya aparecen proyectos de grandes marcas como por ejemplo *Apache* con *MyNewt*, *ARM* con *Mbed OS* o la *Linux Software Foundation* y su *Yocto Project*, pero una búsqueda en profundidad nos devuelve multitud de proyectos más.

Una vez analizadas las webs y repositorios en *GitHub* (GitHub Inc., s.f.) de los proyectos expuestos anteriormente, podemos descartar aquellos que, si bien ofrecen soporte para determinado hardware de bajo consumo, no tienen un desarrollo enfocado al *IoT*. Además, en más de una ocasión nos encontramos con proyectos desarrollados por una sola persona o por un grupo reducido de ellas, con la consecuente falta de soporte para hardware accesible o un desarrollo lento e incluso descontinuado. En casos como el del popular *Contiki OS*, muy tenido en cuenta en multitud de publicaciones en blogs de tecnología, optaremos por descartarlo debido a que su desarrollo se detuvo en 2015 con su versión 3.0.

Según una encuesta del *Eclipse IoT Working Group* (Cabé), en colaboración con otras organizaciones como *IEEE*, *AGILE IoT* o la *Open Mobile Alliance*, los sistemas operativos más utilizados dentro del *IoT* son *FreeRTOS*, *Mbed OS*, *Contiki*, *Tiny OS*, *RIOT-OS* y *Zephyr*, mientras que otras como *Raspbian* o *Ubuntu Core*, y no *Yocto Project*, que aparecía en nuestra tabla, son las variantes *Linux* más populares. Soluciones privativas como *Windows Embedded* de *Microsoft* y *VxWorks* de *Wind River*, que también aparecen en el estudio como opciones frecuentemente utilizadas, quedan por tanto fuera de nuestro foco de atención. En el caso de *Google* con *Android Things*, debido a su reciente aparición, no tiene hoy en día la relevancia suficiente como para entrar en nuestro estudio.

Otros ejemplos muy populares son *RaspberryPi* (Fundación Raspberry Pi, s.f.), con su versión de *Debian* (Debian Project, s.f.) llamada *Raspbian*, y la plataforma *Arduino* (Arduino, s.f.). Sin embargo, descartaremos estas opciones porque, aunque tendrían aplicación en el *IoT*, tienen un ámbito de destino demasiado genérico. Además, con la cantidad de información disponible en Internet, no supondría un reto para nuestro estudio.

Otro aspecto sobre el que filtrar resultados es el del bajo consumo. Pese a que muchos de los recursos mostrados en la tabla anterior presentan soporte para hardware del tipo *ARM Cortex-M* y multitud de características interesantes para el desarrollo de aplicaciones embebidas y de bajo consumo, como la gestión de memoria o el *sleep mode* para apagar el dispositivo cuando no es

necesario recibir información del nodo, las desventajas como el desarrollo descontinuado o la falta de otras tecnologías importantes hacen que los descartemos.

Para mayor comprensión, realizaremos una serie de tablas en las que visualizar más cómodamente las especificaciones de los *SOTR* que vamos preseleccionando:

| | ***FreeRTOS*** | ***Mbed OS*** | ***Contiki*** | ***Tiny OS*** | ***RIOT OS*** | ***Zephyr*** |
|---|---|---|---|---|---|---|
| *Licencia* | MIT | Apache License 2.0 | BSD | BSD | LGPLv2.1 | Apache License 2.0 |

**Tabla 3.- SOTR más populares y su tipo de licencia Open Source**

Como ya hemos descrito previamente, estos tipos de licencia otorgan mayor o menor libertad a la hora de utilizar el software para determinados propósitos. Sin embargo, para el propósito de nuestro estudio no tendremos ningún problema en utilizar cualquiera de los sistemas operativos preseleccionados.

A continuación, se muestra una tabla con multitud de características que hemos podido verificar que están soportadas por los distintos sistemas operativos. Esto nos ayudará también a escoger cuál de ellos puede resultar apropiado o interesante para conocer más sobre él:

| ***Características*** | ***FreeRTOS*** | ***Mbed OS*** | ***Contiki*** | ***Tiny OS*** | ***RIOT OS*** | ***Zephyr*** |
|---|---|---|---|---|---|---|
| *Lenguaje* | C | C | C | nesC (C) | C/C++ | C/Ensamblador |
| *Bluetooth* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *LoRA* | - | ✓ | - | - | ✓ | ✓ |
| *6LoWPAN* | - | ✓ | ✓ | ✓ | ✓ | ✓ |
| *RPL* | - | ✓ | ✓ | ✓ | ✓ | ✓ |
| *CoAP* | - | ✓ | ✓ | ✓ | ✓ | ✓ |
| *Thread* | - | ✓ | - | - | - | ✓ |
| *Ethernet* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *WiFi* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *HTTP* | - | ✓ | ✓ | - | - | ✓ |
| *NFC* | - | ✓ | - | - | ✓ | ✓ |
| *RFID* | - | ✓ | - | - | - | ✓ |
| *UDP* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *TCP* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *OMA LWM2M* | - | ✓ | - | - | - | ✓ |
| *CBOR* | - | ✓ | - | - | ✓ | - |
| *SSL/TLS* | ✓ | ✓ | - | - | - | - |
| *Sleep Mode* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *Gestión de memoria* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *Sim. / Virtualiz.* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| *Nube* | ✓ | ✓ | - | - | - | ✓ |

**Tabla 4.- Características básicas de los SOTR más populares**

Una vez realizada esta tabla podemos observar ya de primeras que no tenemos la seguridad de que *FreeRTOS* tenga soporte para ciertas características que consideramos vitales para nuestro proyecto e incluso cualquier otro. Por tanto, optaremos por descartarlo de ahora en adelante. Por otro lado, vemos que *Mbed OS* se posiciona como la opción de la cual hemos obtenido más respuestas positivas en su documentación, seguido de *Zephyr*.

Pasaremos ahora a describir brevemente cada una de las tecnologías y su utilidad para nuestro proyecto:

- **Lenguaje:** Todos los candidatos finalistas salvo *Tiny OS* utilizan *C*, *C++* o *ensamblador* para desarrollar aplicaciones. Esto es un punto a favor debido a que conocemos en mayor o menor medida cómo programar con ellos.
- **Bluetooth:** Dependiendo de la aplicación que uno desee realizar, se podría utilizar en lugar del *WiFi*. Por ejemplo, en el caso de querer comunicar dispositivos que estén situados en un rango de 100 m sería una buena opción si utilizamos la versión de bajo consumo *BLE* (*Bluetooth Low Energy*), con consumos de hasta 0.5 W.
- **LoRa:** Esta tecnología permite una comunicación inalámbrica de largo alcance mediante el uso de bandas de frecuencia por debajo de 1 GHz. Esto permite más de 10 km de alcance, por lo que es idóneo para aplicaciones en el ámbito rural, por ejemplo.
- **6LoWPAN (*IPv6 over Low-Power Personal Area Networks*):** Es la versión par el internet de las cosas del protocolo *IPv6* desarrollado por la *Internet Engineering Task Force* (*IETF*), una organización de estándares abiertos. Es un protocolo básico si queremos enviar información a través de internet. Cualquier sistema operativo de nuestro análisis debería tener soporte para ellos.
- **RPL (*Routing protocol for Low Power and Lossy Networks*):** Como bien dicen sus siglas, se trata de un protocolo de enrutamiento enfocado a paliar los efectos negativos de las redes de baja potencia y propensas a tener pérdidas.
- **CoAP (*Constrained Application Protocol*):** Es un protocolo de capa de servicio para dispositivos de reducidas prestaciones o constreñidos, de ahí el nombre del protocolo, habilitando su comunicación desde redes de nodos con Internet.
- **Thread:** Protocolo de red basado en *IPv6* y enfocado en la seguridad. Utilizarlo no supone coste, aunque sí tiene condiciones de uso con la empresa desarrolladora, *Thread*

*Group*. Existen alternativas como *OpenThread*, solución open source implementada por *Nest* y soportada por *Zephyr*.

- **Ethernet y WiFi:** Evidentemente indispensables para dispositivos conectados a Internet.
- **HTTP:** Este protocolo web nos podría servir para realizar peticiones a través de internet, bien solicitando información o bien recibiéndola.
- **NFC (*Near Field Communication*):** Este protocolo de comunicaciones entre dispositivos a una distancia de unos pocos centímetros sería necesaria en el caso de querer desarrollar una aplicación de registro mediante un smartphone o una tarjeta de acceso a un recinto.
- **RFID (*Radio-frequency identification*):** Esta forma de reconocimiento a distancia podría usarse para el etiquetado de paquetes de mensajería y su seguimiento a través de la cadena de distribución.
- **UDP y TCP:** Protocolos de red indispensables para el uso de nuestros dispositivos en el *IoT*. Sería raro que no lo implementasen las soluciones software que hemos seleccionado.
- **HTML:** Este lenguaje web nos permitiría desarrollar aplicaciones cuyos dispositivos enviaran páginas web al usuario de forma que éste tuviera que rellenar formularios y enviarlos de vuelta. Ejemplo: Una tienda de ropa que te envía un formulario mediante HTML y otros lenguajes para inscribirte en promociones nada más salir de la tienda o al pasar cerca.
- **OMA LWM2M (*OMA Lightweight M2M*):** Es un protocolo de la *Open Mobile Alliance* para las comunicaciones directas *machine to machine* enfocado al *IoT*.
- **CBOR (*Concise Binary Object Representation*):** Este formato de datos permite reducir la cantidad de datos enviados para transmitir una determinada información, lo que repercute positivamente en la velocidad de transmisión y el tiempo que nuestro dispositivo está consumiendo más energía para transmitirla.
- **SSL/TLS:** Protocolos de seguridad útiles cuando buscamos privacidad en nuestras comunicaciones. Muy a tener en cuenta a la hora de seleccionar una solución software para su estudio en detenimiento.
- **Sleep Mode:** Esta característica significa básicamente la capacidad de nuestro dispositivo de entrar en un estado de muy bajo consumo para aumentar la autonomía. Si unimos esta capacidad a la de un hardware que ya de por sí consume pocos recursos, el resultado

puede aportarnos autonomías de varios años en dispositivos aislados, reduciendo la necesidad de mantenimiento.

- **Gestión de memoria:** Esta característica es interesante para mejorar el rendimiento de nuestros dispositivos, y evitar así un consumo innecesario de energía que repercuta en la autonomía de la batería, en caso de disponer de ella.
- **Simulación / Virtualización:** Disponer de herramientas para simular nuestro código con un determinado hardware sin necesidad de disponer de él puede ayudarnos en el desarrollo de nuestro proyecto.
- **Nube:** Esencial en multitud de aplicaciones en las que la gran cantidad de datos debe almacenarse en ella para su posterior tratamiento y análisis.

Las conclusiones que podemos extraer de este análisis de características nos llevan a descartar a los últimos candidatos en favor de *Mbed OS*. Esto no significa que el resto sean malos candidatos, sino que el conjunto de ventajas respecto a los demás, unido a los recursos que estudiaremos más adelante, nos hacen escoger este sistema operativo como mejor opción para nuestro análisis. Como segunda opción, es posible que hubiéramos escogido a *Zephyr* por sus múltiples características y la facilidad de búsqueda de información, así como por su soporte hardware y actualizaciones frecuentes.

#### 4.2.1.1 El Sistema Operativo: ARM Mbed OS

Este sistema operativo embebido O*pen Source* fue creado por la empresa británica de semiconductores y software *ARM Holdings* en el año 2009. Está enfocado a dispositivos de 32 bits con arquitectura *ARM* de la familia *Cortex-M* y soporta las principales tecnologías de comunicación asociadas al *IoT*, como son *Bluetooth Low Energy*, *Thread*, *6LoWPAN*, *NFC*, *LoRa LPWAN*, *RFID*, *Ethernet* y *WiFi* entre otros, pero tampoco se olvida de la necesidad de mantener la seguridad con los protocolos *SSL* y *TLS* y un kernel específico a nivel de hardware llamado uVisor, que restringe el acceso a memoria y periféricos.

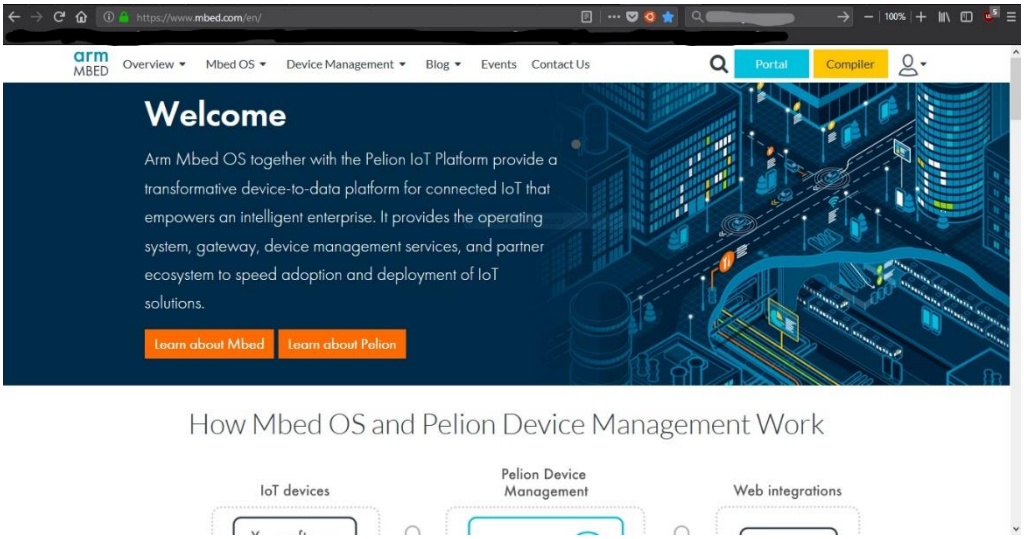



**Ilustración 3.- Página de inicio de ARM Mbed OS**

*Mbed* tiene un diseño modular, lo que significa que sólo las librerías necesarias son incluidas en los proyectos generados con su IDE online, *Mbed Compiler*. Este espacio de trabajo nos permite navegar por un repositorio de proyectos y librerías tanto de fabricantes como los adaptados por los usuarios registrados en la plataforma. También disponemos de un foro donde resolver los problemas que puedan surgir cuando el sistema operativo o los proyectos se actualizan.

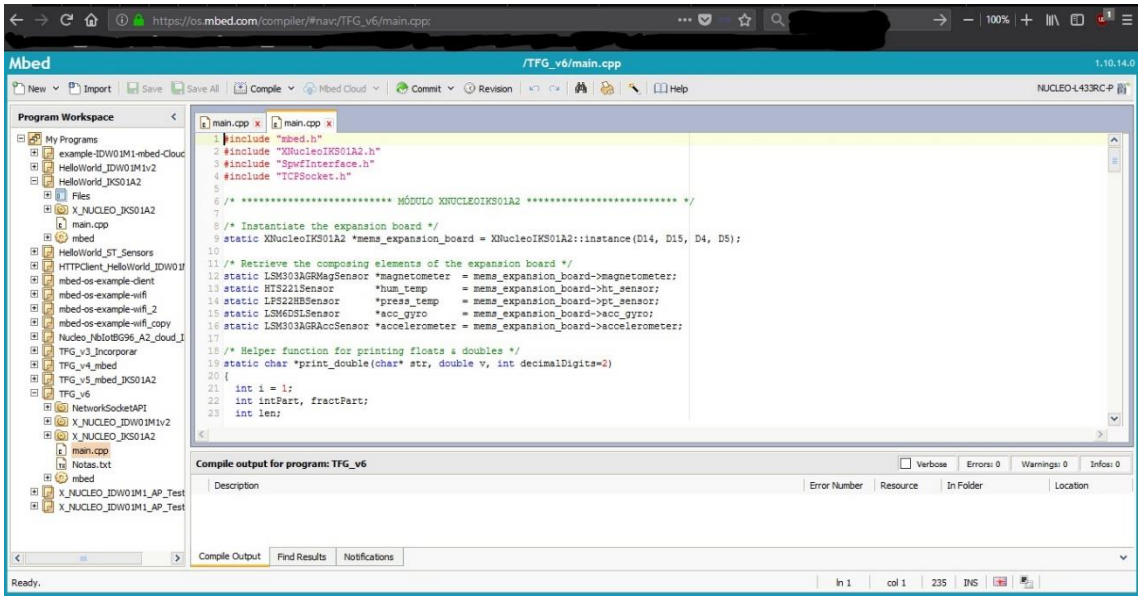



**Ilustración 4.- Entorno de desarrollo de Mbed OS: Mbed Compiler**

Es importante mantener actualizada la carpeta con los recursos de *Mbed* para que no dé errores al compilar. En caso de que un programa esté diseñado para otra placa, basta con utilizar el código que nos interese y *Mbed* se encargará de utilizar los archivos de configuración de nuestra placa, ya que cuando creamos nuestro espacio de trabajo podemos seleccionar la placa sobre la que trabajaremos. Periódicamente se va añadiendo soporte a nuevo hardware, por lo que este motivo es también importante para mantener todo actualizado a la última versión.

### 4.2.2 Hardware:

Tras consultar el soporte de cada una de las soluciones software que hemos estudiado previamente, analizando sus páginas web y las herramientas de que dispone para facilitarle el proceso de desarrollo al programador, hemos escogido a *STMicroelectronics* para nuestro proyecto. Con sede en Suiza, lleva desde 1957 fabricando todo tipo de componentes semiconductores. El motivo de esta elección ha sido la buena disponibilidad de sus productos en distribuidores de confianza en el sector como *Digi-Key*, así como su buena compatibilidad con el Sistema Operativo escogido. Este fabricante ofrece un catálogo amplio de productos, sobre todo en lo relacionado con los microcontroladores de la serie *ARM Cortex-M*. No escatima tampoco en ofrecer las herramientas básicas para empezar a desarrollar aplicaciones, y que veremos a lo largo de este apartado. La organización de sus productos en categorías bien diferenciadas nos hace distinguir con facilidad aquellos enfocados al *IoT*, y escoger de entre un amplio rango de prestaciones para no sobredimensionar las capacidades del producto respecto de las necesidades que nos hemos marcado.

A continuación, se explicará con detalle el hardware escogido y los motivos que llevaron a su elección. También se describirán las características técnicas de cada uno de los dispositivos escogidos, así como sus ventajas e inconvenientes y los problemas encontrados.

#### 4.2.2.1 STM32 Nucleo-L433RC-P:

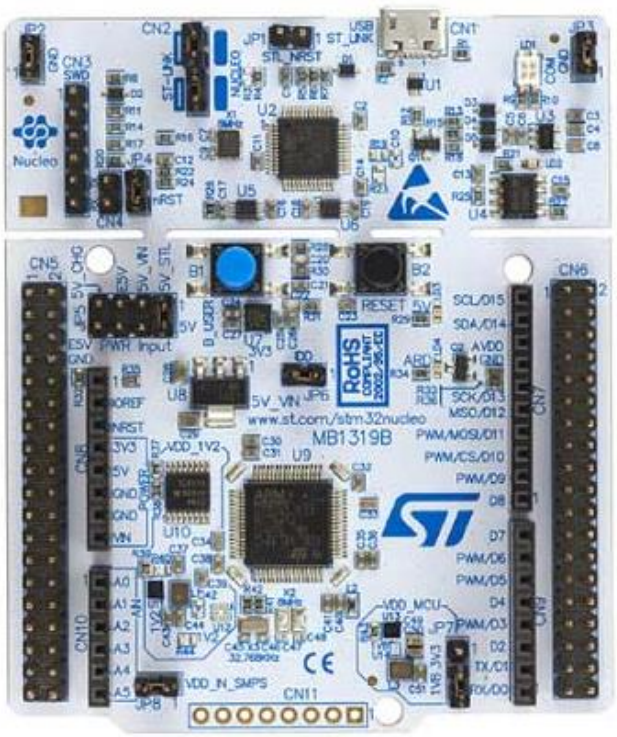



**Ilustración 5.- Vista previa de la placa STM32 Nucleo-L433RC-P**

Es la placa principal sobre la que colocaremos otros dos dispositivos: El que implementa una serie de sensores y el de conectividad Wifi. Si consultamos las prestaciones de este dispositivo podremos comprobar que dispone de multitud de recursos, tanto hardware como software.

Las características básicas se pueden resumir en los siguientes puntos:

- Frecuencia de reloj de 4 a 48 MHz. 32 KHz en modo *Real Time Clock* (RTC).
- CPU ARM Cortex M4 de 32 bits con frecuencia de hasta 80 MHz.
- 256 KB de memoria Flash.
- 64 KB de SRAM.
- Alimentación de 1.71 V hasta 3.6 V, y consumo de corriente del orden de *nA* en *standby* y *uA* en modo de ejecución.

- Puertos *mini-USB*, *Arduino Uno V3* (*CN5* y *CN6*), extensión ST para conectividad con otros productos.

La codificación utilizada para nombrar las distintas placas de este fabricante nos da pistas de sus especificaciones. Su significado se puede encontrar en el documento *DM00387966.pdf* de la web de ST, y lo desglosaremos en los siguientes puntos:

- STM32: Indica los bits de la CPU. En el apartado de microcontroladores de la web de ST podemos encontrar dispositivos de 8 y 32 bits, pero con distinto enfoque a la hora de desarrollar proyectos.
- L433: Indica la línea de productos del fabricante ST. Otros ejemplos son AF, AL, L, S, F, H y la familia dedicada a automoción SPC5.
- RC-P: Cada una de estas letras nos indica un aspecto diferente.
    - La R es por los 64 pines del ARM Cortex M4 que monta nuestra placa. Para los 48 y 100 pines de otros productos nos encontramos con los indicadores C y V, respectivamente.
    - La C indica el tamaño de memoria Flash de 256 KB. Aquí nos encontramos con que, si no lleva letra, la memoria Flash es de 8 a 64 KB, pero cuando lleva, la B es para 128 KB, C para 256 KB, E para 512 KB, G para 1 MB, H para 1.5MB e I para 2 MB.
    - La P indica *External SMPS architecture*. *SMPS* viene de *Switched-mode Power Supply*, que permite un menor consumo mediante dos perfiles de alimentación: *SMPS U11* (1.1 V y 30 mA), y *SMPS U13* (1.8 V y 50 mA).

Nuestra placa pertenece a la familia de 32 bits de ultra bajo consumo L4. La inmediata superior, L4+, aumenta las prestaciones con una frecuencia de reloj de hasta 120 MHz, 2 MB de memoria Flash y 640 KB de SRAM, entre otras características.

En cuanto al precio, esta placa fue adquirida por 13’63 € en *Digi-Key*, sin contar con el *IVA*.

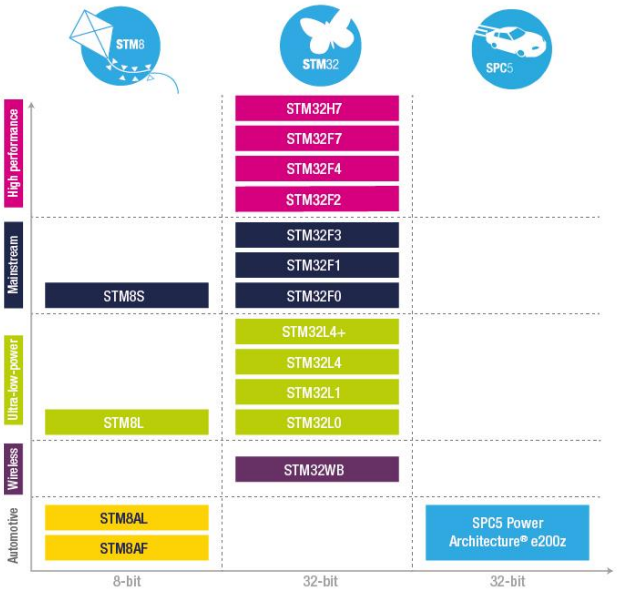



**Ilustración 6.- Línea de productos para el IoT de STMicroelectronics**

Para poder empezar a utilizar nuestra placa será necesario comprobar que su firmware está actualizado. Para ello debemos instalar en nuestro ordenador el controlador *ST-LINK/V2-1 USB driver*, que podemos encontrar en el apartado *Herramientas y Software* de la web del dispositivo. Con ello dispondremos de la utilidad *ST-Link Upgrade* para actualizar nuestra *Nucleo L433RC-P*, conectada mediante un cable mini-USB a USB:

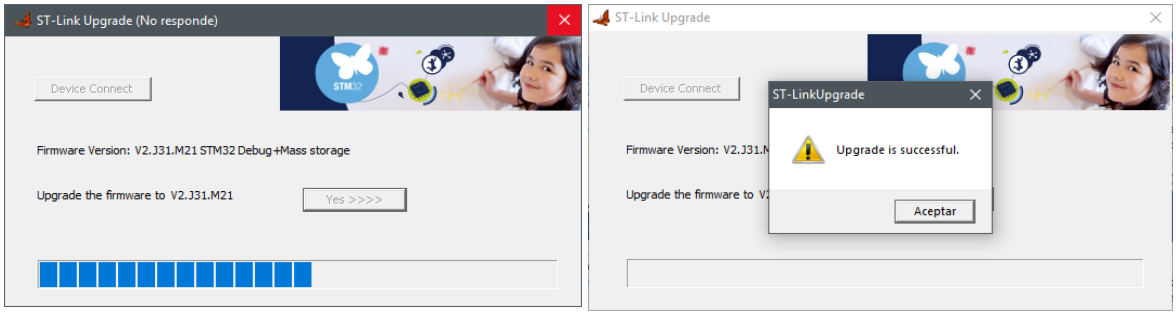



**Ilustración 7.- Actualización del firmware mediante ST-Link Upgrade**

ST ofrece multitud de herramientas para ahorrarnos tiempo a la hora de crear proyectos específicos para nuestra placa. Disponemos de la herramienta *STM32CubeMX*, que nos permite seleccionar las características del *pinout*, la configuración de reloj y hacer una estimación de consumo simulando distintos perfiles energéticos. Posteriormente, podemos generar los ficheros resultantes para incorporarlos a nuestro proyecto.

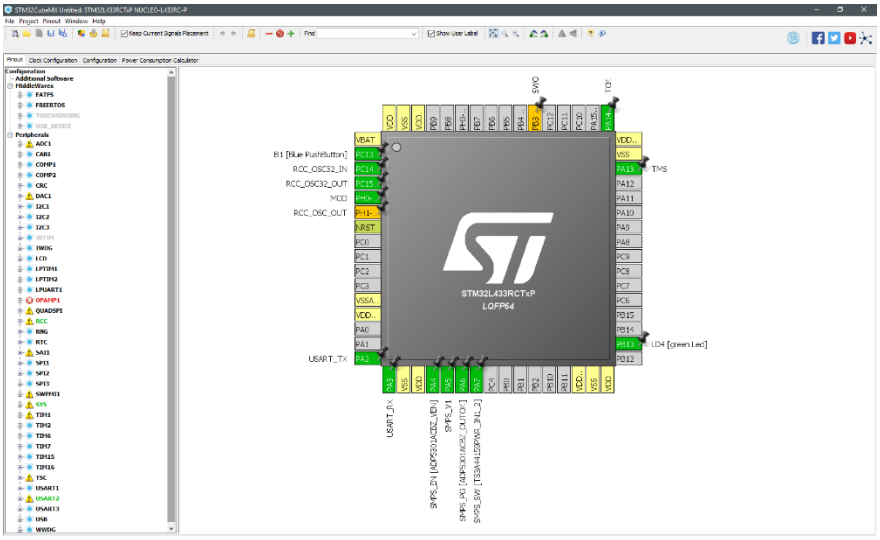



**Ilustración 8.- Configuración del *pinout***

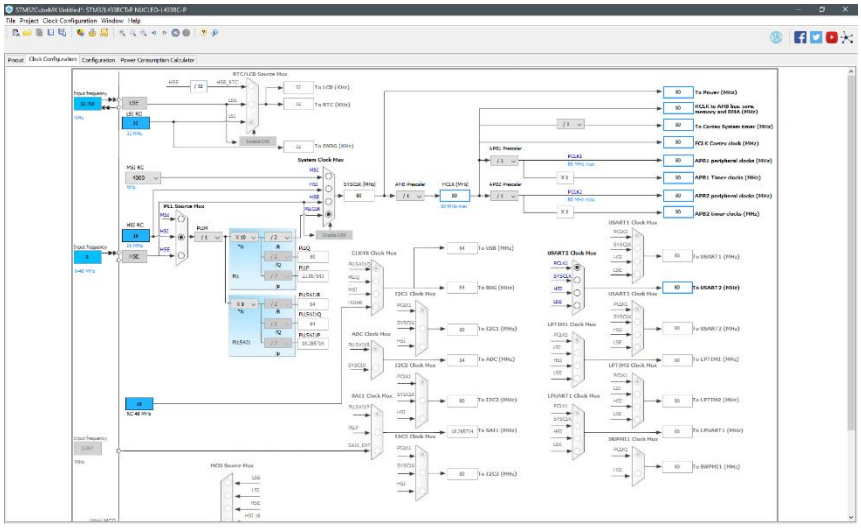


**Ilustración 9.- Configuración del reloj**

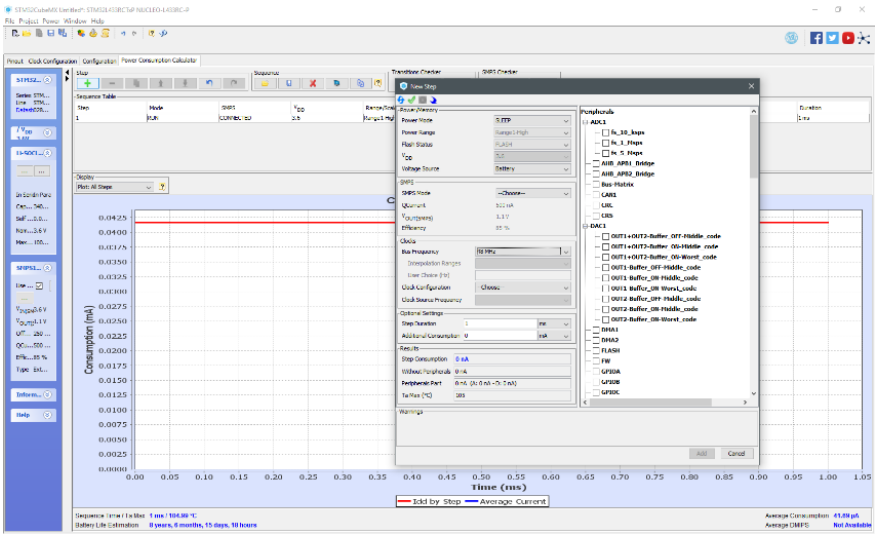



**Ilustración 10.- Estimación de consumo**

Gracias a estas herramientas, podemos simular el consumo aproximado de nuestro dispositivo como por ejemplo en modo de ejecución o en *sleep mode*, configurar la frecuencia de reloj, activar o no el *Switched-mode Power Supply* (SMPS) y escoger una alimentación por baterías de distintos tamaños y tipos para conseguir variaciones de autonomía de entre unos pocos días a incluso más de 8 años.

#### 4.2.2.2 X-Nucleo-IDW01M1:

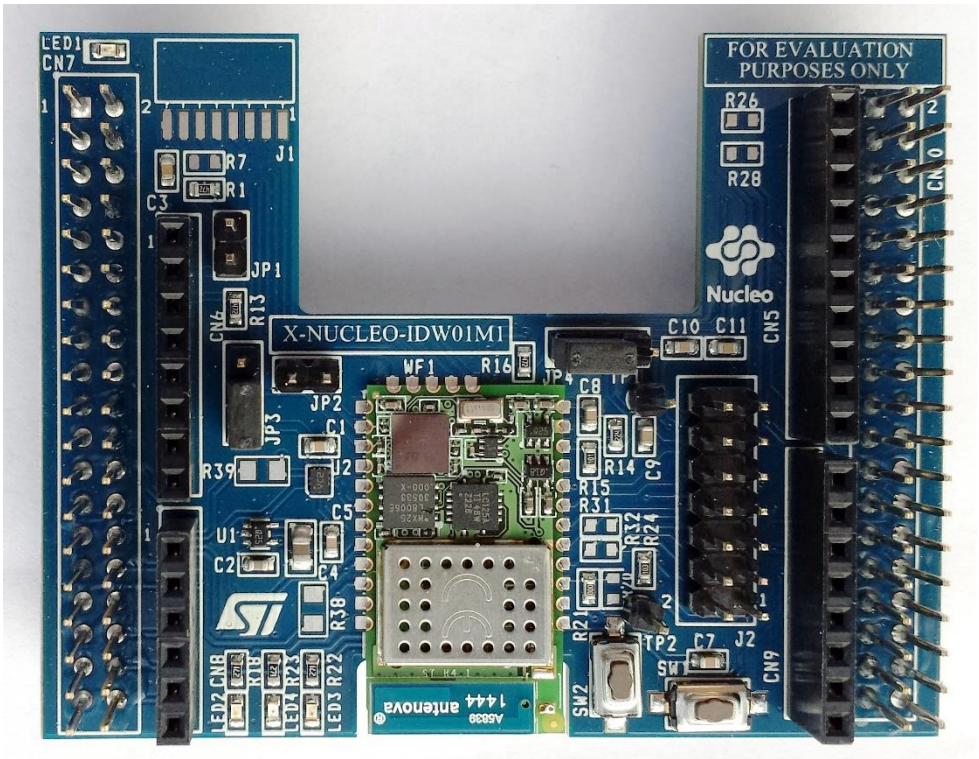



**Ilustración 11.- Vista previa de la placa X-Nucleo-IDW01M1**

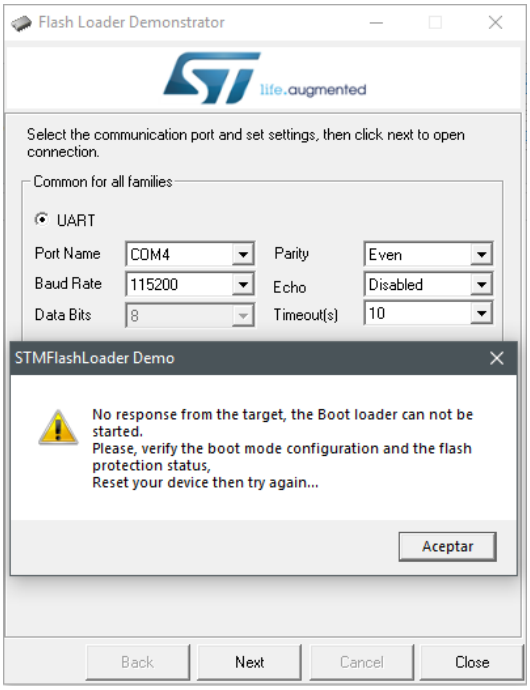
Este dispositivo nos proporcionará la conexión WiFi necesaria para enviar la información de la placa que contiene los sensores. Hablaremos de esta última posteriormente.

Lo primero que debemos hacer para utilizar este dispositivo es actualizar su firmware descargando un archivo llamado *X-Cube-WIFI1*. En él encontramos un documento que nos guía en el proceso previo con nuestra placa *Nucleo L433RC-P*. Los pasos a seguir son:

- Abrir la carpeta correspondiente a la familia a la que pertenece nuestra placa.
- Conectar la placa a un PC mediante el puerto mini-USB.
- Arrastrar el archivo .bin contenido en la carpeta en la que hemos entrado previamente.
- Desconectar la placa.
- Colocar un jumper en las patillas *JP2* para cambiar al modo “firmware download”.
- Conectar la placa *X-Nucleo-IDW01M1* con la placa *Nucleo L433RC-P*.

- Conectarlas al PC como hemos hecho anteriormente.
- Pulsar el botón *SW1* (*reset*) de la placa *X-Nucleo-IDW01M1*.
- Descargar el archivo stsw-mcu0005.zip de la web de ST.
- Abrir la utilidad *Flash Loader Tool* que hay contenida y seguir los pasos que indica.

Aquí nos encontramos con nuestro primer error de procedimiento. Si bien indica escoger en función de la familia, nos encontramos poco después con que el dispositivo no está soportado por la herramienta que actualiza el firmware, ya que nuestra placa es la *L433RC-P* y los modelos testeados son el *L053R8*, *F103RB*, *F401RE* o *L476RG*. Cuando intentamos establecer comunicación entre *Flash Loader Tool* y nuestra placa, la utilidad nos reporta errores. Repasando el documento e intentando multitud de combinaciones en el proceso tanto previo como posterior, por si hubiera algún error en el documento, no obtenemos resultados diferentes.




Consultando de nuevo las webs tanto de *STMicroelectronics* como de *Mbed OS*, nos encontramos con que nuestra placa *X-Nucleo-IDW01M1* ha dejado de tener soporte, si bien todavía se pueden consultar las librerías y ejemplos aplicados a ella.

Una forma de evitar esto es asegurarnos de que el hardware mantiene el soporte tanto en la web del fabricante como en la del sistema operativo, dado que nuestra suposición fue que por la parte del fabricante no habría ningún problema en utilizarla.

El precio de adquisición de esta placa fue de 20’44 € en *Digi-Key*, sin contar con el *IVA*.

#### 4.2.2.3 X-Nucleo-IKS01A2:

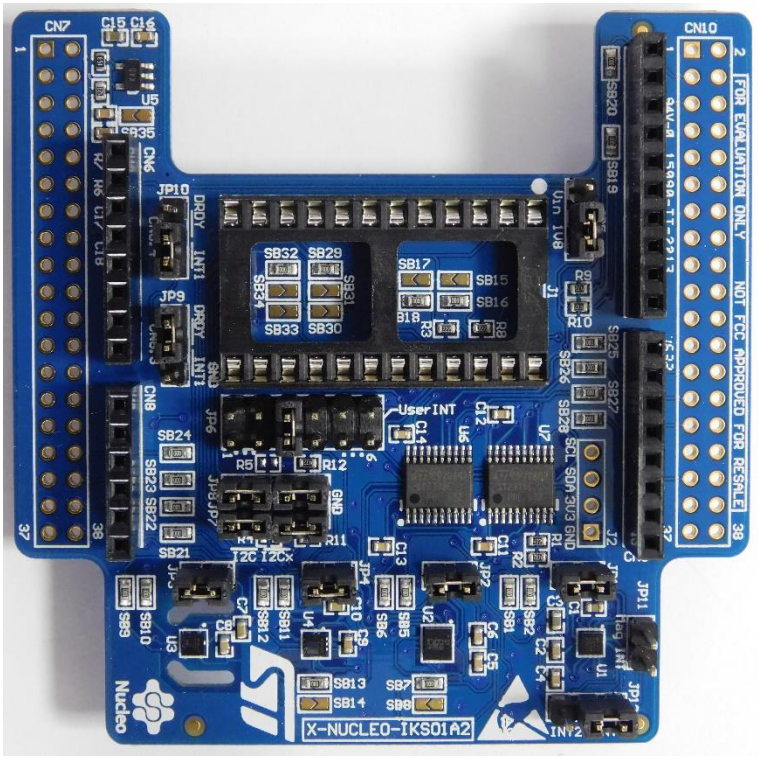



**Ilustración 12.- Vista previa de la placa X-Nucleo-IKS01A2**

Se trata de la placa que conectaremos a la *X-Nucleo-IDW01M1* a través del puerto *Arduino UNO R3* (*CN5*, *CN6*, *CN8* y *CN9*). El orden de conexión entre las 3 placas no tiene pérdida: Es el mismo que el orden de presentación de las distintas placas. Esta, por tanto, irá conectada en último lugar. Dispone de multitud de sensores que describiremos a continuación:

- ***HTS221***: Sensor capacitivo digital de humedad relativa y temperatura.
- ***LPS22HB MEMS***: Sensor de presión con rango comprendido entre 260 - 1260 hPa.

- ***LSM6DSL MEMS 3D***: Acelerómetro (+-2/+-4/+-8/+-16 g) y giróscopo 3D (+-125/+-245/+-500/+-1000/+-2000 dps).
- ***LSM303AGR MEMS 3D***: Acelerómetro (+-2/+-4/+-8/+-16 g) y MEMS3D magnetómetro (+-50 gauss)
- ***DIL24***: Conector para sensores adicionales.

El precio de adquisición de esta placa fue de 17’03 € en *Digi-Key*, sin contar con el *IVA*.

Para comprobar el correcto funcionamiento de la placa, disponemos también de una herramienta que nos proporciona *ST* que incluye unas demos para cada uno de los sensores. Basta con conectar por *miniUSB* nuestra *Nucleo L433RC-P* junto con la *X-Nucleo-IKS01A2* e iniciar la utilidad *Unicleo-GUI*, previamente instalada. Si consultamos el apartado 2.6 del documento *UM1859* (STMicroelectronics, s.f.), podemos encontrar información sobre el modo de empleo de esta utilidad para verificar que ninguno de los sensores presenta algún error de funcionamiento.

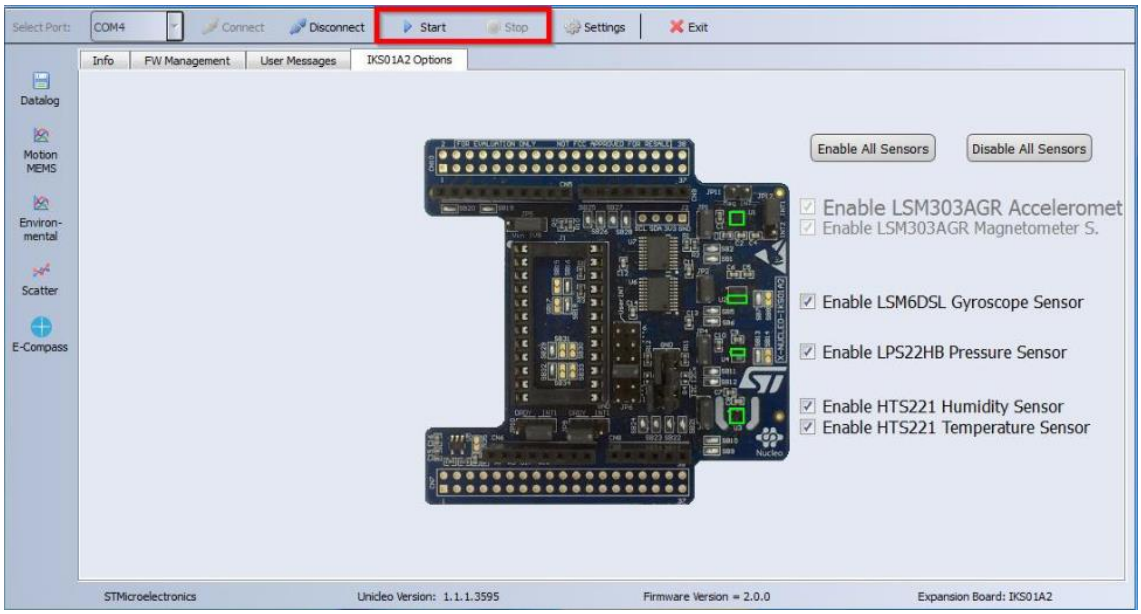



**Ilustración 13-. Vista de la utilidad Unicleo-GUI con el hardware conectado para test.**

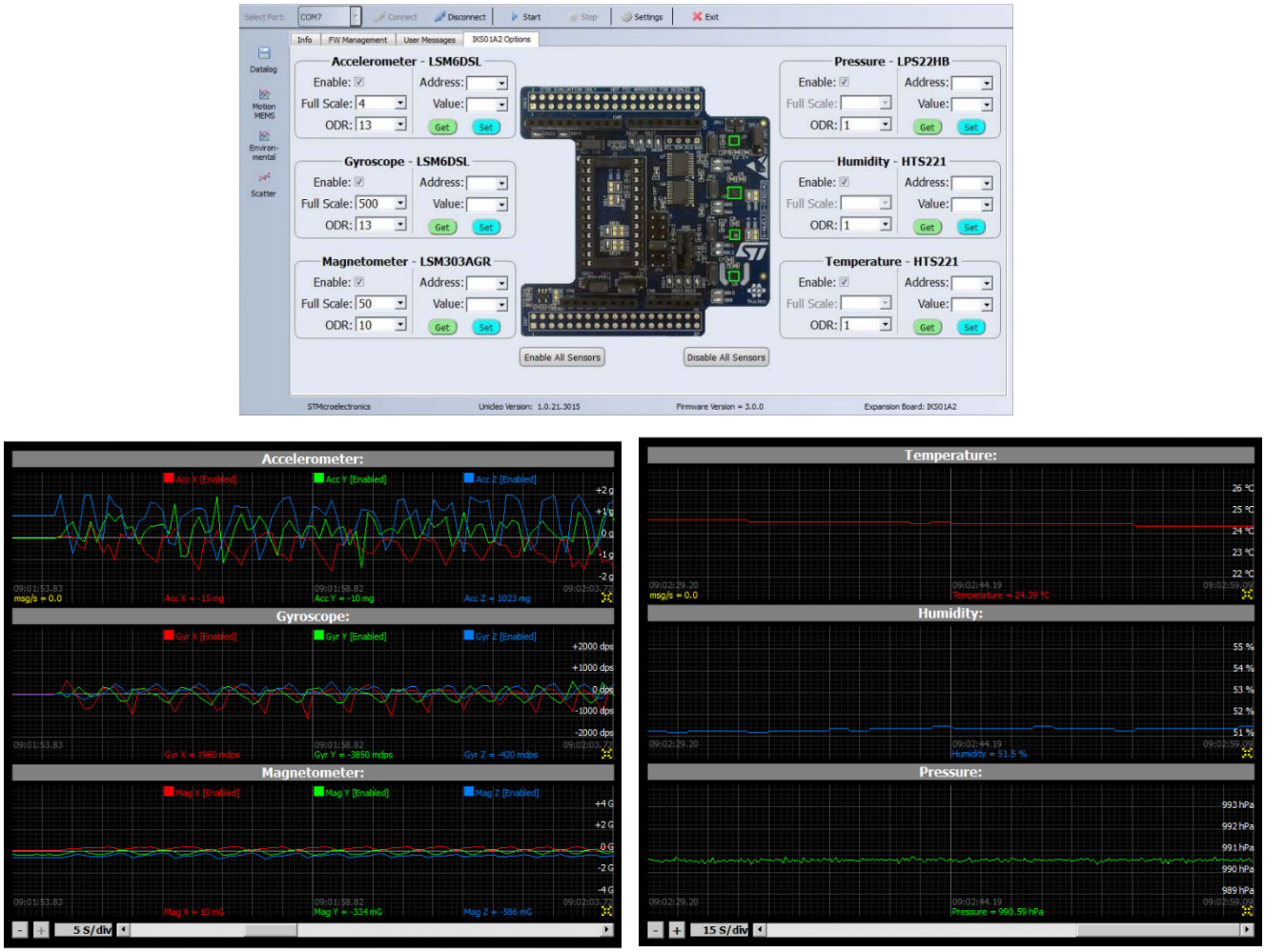



**Ilustración 14.- Ejemplos de representación de datos de los sensores**

Por último, con ayuda de *Mbed Compiler* elaboraremos un pequeño programa que reciba los datos obtenidos por los sensores a través del pin *I2C* (*D14* y *D15*) del dispositivo y lo envíen por el puerto serie de la placa *Nucleo L433RC-P* a un ordenador ya que, como hemos comentado anteriormente en el apartado del módulo *WiFi*, no ha sido posible utilizar ningún ejemplo relacionado con este módulo para implementarlo en nuestra idea de proyecto. Utilizaremos pues el ejemplo *HelloWorld_IKS01A2*, disponible en el repositorio de *Mbed Compiler*, y lo adaptaremos a nuestro gusto.

A continuación, tenemos una muestra de lo recibido por el puerto serie:

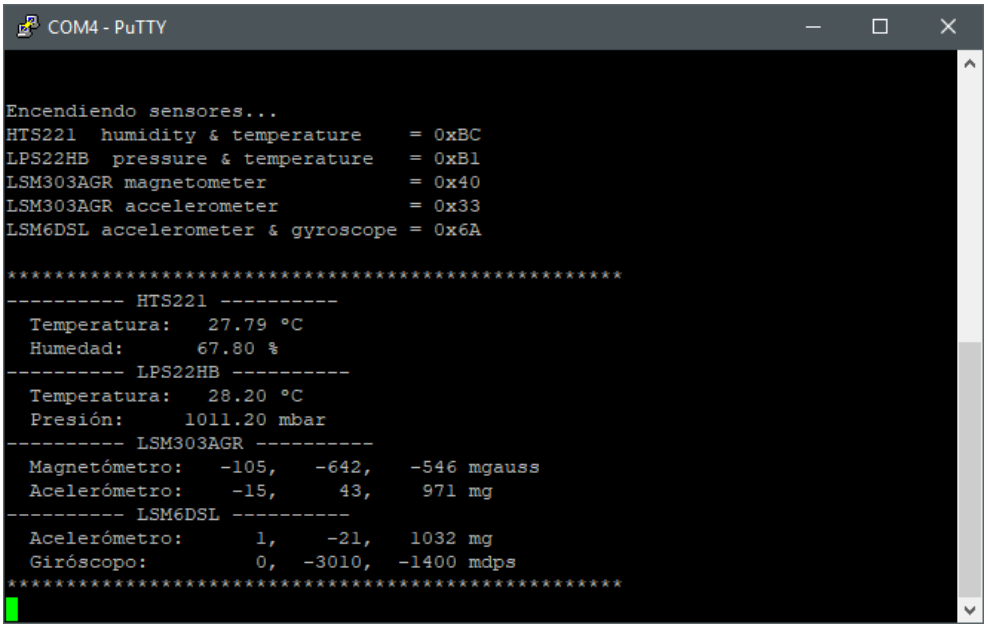



**Ilustración 15-. Medidas tomadas con el dispositivo en reposo.**

Como se puede observar, podemos obtener dos medidas de temperatura diferentes, una procedente del *HTS221* y otra del *LPS22HB*. El motivo de que sean diferentes se puede deber a la tolerancia de medida de cada uno de ellos, al tipo de sensor, a una colocación en el dispositivo donde recibe el calor de otro componente etc. Si acercamos unos imanes de gran potencia, también observaremos un cambio en los resultados del magnetómetro considerablemente mayor al de la propia deriva del dispositivo, alcanzando valor como "-13635, 13911, 35703" *mgauss* en los distintos ejes. Del mismo modo, si desplazamos en distintas direcciones nuestro dispositivo también comprobaremos la variación de medidas en el acelerómetro. Con esto podemos confirmar que los sensores reaccionan a los cambios de magnitud y que las medidas entran dentro del rango indicado por el fabricante.

# 5. Conclusiones y propuesta de trabajo futuro:

Una vez terminado el proyecto y habiendo repasado los objetivos planteados al principio de esta memoria, puedo decir que he completado la mayor parte de ellos, si bien he tenido que sacrificar algunos puntos concretos debido a elecciones no del todo acertadas y a la falta de presupuesto y tiempo para solucionarlas.

He podido analizar la situación de los principales *SOTR* dedicados al *Internet de las Cosas*, concluyendo que es un sector en auge y en constante cambio debido a la necesidad de adaptarse día a día al nuevo hardware que desarrollan grandes fabricantes y las nuevas aplicaciones y servicios que surgen de la ya mencionada unión de los usuarios con los datos que extraemos del entorno.

A la hora de escoger una combinación de sistema operativo y hardware compatible he aprendido a seleccionar en base a unos criterios específicos, descartando aquellos que no los cumplían o que resultaban menos aptos para mi primer acercamiento al *Internet de las Cosas*. He desarrollado la capacidad de indagar en lo que el fabricante del software o del hardware me ofrece con sus productos, las herramientas de configuración, de estimación de consumo etc.

Si bien los resultados no han sido del todo exitosos, me ha servido como experiencia futura bien por tema laboral o bien como aficionado a la tecnología. Pese a ello, considero provechoso este estudio de *Mbed OS* porque ahora tengo un poco más de práctica a la hora de buscar información sobre algo que desconozco, en este caso un conjunto de software más hardware, seleccionar una opción de entre las múltiples encontradas y estudiar su utilidad. He aprendido sobre el Internet de las Cosas y he podido observar la rápida evolución del sector. Además, me sirve como prueba escrita de que aún queda mucho por descubrir y aprender de las aplicaciones futuras de este tipo de software y dispositivos dedicados.

Durante la elaboración del proyecto y de esta memoria he sentido la necesidad de saber más, de en un futuro poner en prácticas ideas que me han venido a la mente durante estos meses. Acompañar las placas de ST de que dispongo con una aplicación en Android para mi móvil que solicite la información sería interesante. Podría utilizar otro sistema operativo como *Zephyr*, que

he visto en mi análisis que también podría ser interesante saber más sobre él u otro hardware como la *RaspberryPi* o *Arduino*, que antes de realizar este proyecto ya tenía curiosidad por ellos.

## 5.1 Trabajo futuro

En el caso de que mi experiencia hubiera sido mayor y de que las dificultades encontradas no hubieran hecho más difícil alcanzar los objetivos principales, una posible ampliación de este proyecto habría sido la siguiente:

- Comparativa de múltiples *SOTR* en función del rendimiento y autonomía de los nodos al estar conectados a un batería en vez de a la red eléctrica.
- Análisis de la escalabilidad del proyecto a la hora de deducir el número máximo de nodos soportables.
- Almacenamiento de los datos en la nube para su posterior análisis.
- Elaboración de una aplicación en Android para solicitar los datos de los sensores y la autonomía restante del dispositivo de una forma sencilla y práctica.

## 5.2 Conclusiones personales

En primer lugar, la elaboración de este trabajo me ha permitido evaluar mi capacidad para hallar la solución a un problema sin la ayuda previa de un profesor, como sí ha venido siendo a lo largo de la carrera en la mayoría de las asignaturas. Disponer únicamente de un punto de partida y el asesoramiento del tutor en determinadas ocasiones para alcanzar el objetivo me ha empujado a depender prácticamente de mí mismo y al menos intentar trabajar como un ingeniero de telecomunicaciones.

Escogí este proyecto por su relación con la especialidad de electrónica que he cursado en la carrera, como complemento a lo aprendido en asignaturas como Aplicaciones de los Microcontroladores. Sin embargo, no le vi sentido a escoger de nuevo Arduino, que es lo que utilizamos en las clases prácticas. Decidí, como ya he explicado en anteriores apartados, descartar ésta y otras opciones como *RaspberryPi*, aunque en el caso de querer seguir aprendiendo sobre la programación de microcontroladores y otros dispositivos de poca potencia quizá escogería estas

plataformas para aplicaciones como drones, consolas retro u otros hobbies debido a su gran popularidad y a su comprometida comunidad.

Una vez finalizado este trabajo, puedo decir que sé que podría haber hecho algunas cosas de forma diferente. Ahora sé cómo encontrar más fácilmente los ejemplos y tutoriales que he utilizado para comprobar que un determinado hardware va a tener soporte para más tiempo que el que me ha llevado realizar este proyecto. También sé que me queda mucho por aprender tanto del *IoT*, como de electrónica, programación de microcontroladores y de la resolución de proyectos de cierta envergadura, pero creo que me ha servido como base para afrontar el mundo laboral con un poco más de seguridad en mí mismo. Esto se sumará a las prácticas en empresa que ya he podido realizar y que me han formado como persona y como ingeniero para conseguir mis objetivos con esfuerzo y dedicación.

# 6. Bibliografía:

Amazon Inc. (s.f.). *Amazon Dash Button*. Obtenido de https://www.amazon.es/b/ref=nav_shopall_db_beauty?ie=UTF8&node=10909716031

Apple Inc. (s.f.). *CarPlay*. Obtenido de https://www.apple.com/es/ios/carplay/

Arduino. (s.f.). Obtenido de https://www.arduino.cc/

Cabé, B. (s.f.). *Eclipse IoT Working Group. IoT Developer Survey 2018.* Obtenido de https://blog.benjamin-cabe.com/2018/04/17/key-trends-iot-developer-survey-2018

Crespo, A., & Alonso, A. (2010). Una Panorámica de los Sistemas de Tiempo Real. *Revista Iberoamericana de Automática e Informática Industrial*. Obtenido de https://polipapers.upv.es/index.php/RIAI/article/view/8121

Debian Project. (s.f.). *Págna principal*. Obtenido de https://www.debian.org/index.es.html

European Research Cluster on the Internet of Things. (Octubre de 2015). *Internet of Things Applications. AIOTI WG01 – IERC.* Obtenido de https://aioti.eu/wp-content/uploads/2017/03/AIOTIWG01Report2015-Applications.pdf

Fundación Raspberry Pi. (s.f.). *Página principal*. Obtenido de https://www.raspberrypi.org/

GitHub Inc. (s.f.). Obtenido de https://github.com/

Google Inc. (s.f.). *Android Auto*. Obtenido de https://www.android.com/auto/

Google Inc. (s.f.). *Google Home*. Obtenido de https://store.google.com/product/google_home

Internet Engineering Task Force (IETF). (s.f.). Obtenido de https://www.ietf.org/

López, J. E., Chavez, J. C., & Sánchez, A. K. (Marzo de 2017). *Modelado de una red de sensores y actuadores inalámbrica para aplicaciones en agricultura de precisión. In Humanitarian Technology Conference (MHTC), IEEE Mexican (pp. 109-116). IEEE.*

Mercedes-Benz, Daimler AG. (s.f.). *MBUX - Mercedes-Benz User Experience*. Obtenido de https://www.mercedes-benz.com/en/mercedes-benz/innovation/mbux-mercedes-benz-user-experience-revolution-in-the-cockpit/

Nest Labs, Alphabet Inc. (s.f.). *Cámara de vigilancia inteligente Nest*. Obtenido de https://nest.com/es/cameras/nest-cam-iq-outdoor/overview/

Nest Labs, Alphabet Inc. (s.f.). *Termostato Nest*. Obtenido de https://nest.com/es/thermostats/nest-learning-thermostat/overview/

Open Mobile Alliance. (s.f.). Obtenido de https://www.omaspecworks.org/

Open Source Initiative. (s.f.). *Tipos de Licencia Open Source*. Obtenido de https://opensource.org/licenses/alphabetical

Osrtos. (s.f.). *List of open source real-time operating systems*. Obtenido de https://www.osrtos.com/

Pulido Cañabate, E. (2016). *Big data: ¿Solución o problema? Encuentros Multidisciplinares. Características básicas del Big data. Párrafo 6.*

Pulido Cañabate, E. (2016). *Big data: ¿Solución o problema? Encuentros Multidisciplinares. Características básicas del Big data. Párrafo 9.*

Silberschatz, A., Galvin, P. B., Gagne, G., & & Silberschatz, A. (1998). Operating system concepts (Vol. 4). Reading: Addison-wesley.

STMicroelectronics. (s.f.). *Manal de usuario UM1859*. Obtenido de https://www.st.com/content/ccc/resource/technical/document/user_manual/c8/04/7f/d1/e9/24/47/09/DM00157069.pdf/files/DM00157069.pdf/jcr:content/translations/en.DM00157069.pdf

Thread Group. (s.f.). Obtenido de https://www.threadgroup.org/

# 7. Anexos:

## 7.1 Vista previa de la conexión entre las placas