DESARROLLO E IMPLEMENTACIÓN DE UNA ESTRATEGIA DE DESPLIEGUE Y MANTENIMIENTO DE RED DE NODOS SENSORES

FRANCISCO JAVIER SÁNCHEZ MORENO

MÁSTER EN INTERNET DE LAS COSAS, FACULTAD DE INFORMÁTICA,

UNIVERSIDAD COMPLUTENSE DE MADRID

Convocatoria de Septiembre

Calificación: 6.0

03/09/2018

Directores:

Jose Ignacio Gómez Pérez

Daniel Ángel Chaver Martínez

Autorización de Difusión

FRANCISCO JAVIER SÁNCHEZ MORENO

03/09/2018

El/la abajo firmante, matriculado/a en el Máster en Internet de las Cosas de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: “Desarrollo e implementación de una estrategia de despliegue y mantenimiento de red de nodos sensores”, realizado durante el curso académico 2017-2018 bajo la dirección de José Ignacio Gómez Pérez y Daniel Ángel Chaver Martínez en el Departamento de Ingeniería de Computadores, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen

El uso de las energías renovables, y en concreto de la solar, implica numerosas ventajas, medioambientales y de sostenibilidad, por lo que predecir la radiación solar que habrá en un determinado lugar en un determinado momento puede ser fundamental para estudiar la producción de energía solar. Para ello, se ha creado un dispositivo con distintos sensores, conectados a una Pycom LoPy, placa de bajo consumo con conectividad WiFi y LoRa, capaz de ejecutar scripts en MicroPython, y todo ello geolocalizado para poder estudiar el punto exacto donde se toman los datos, además de tener un mejor control de cada nodo.

Este Trabajo Fin de Máster se ha centrado en una estrategia de despliegue de una red de cientos, incluso miles, de nodos. Esta estrategia describe el cómo se van a posicionar estos nodos, el desarrollo del código necesario para llevar a cabo las tareas necesarias de medición de los sensores, geolocalización, obtención de las redes, envío de los datos a un servidor MQTT, configuración de un servidor y desarrollo de una base de datos para la administración de las horas de sueño del dispositivo, redes cercanas y la geolocalización de cada nodo. Además de estudiar otros aspectos de la LoPy, como su conectividad con certificados, consumo energético y posible retraso del reloj.

Palabras clave

Lopy, Radiación, Energía solar, Geolocalización, MQTT, LoRa, WiFi

Abstract

The use of renewable energies, and specially of the solar, implies many advantages, environmental and sustainability, so predicting solar radiation what will do in a certain place at a certain moment could be fundamental to study the solar energy production. For doing it, a device has been created with different sensors, connected to a Pycom LoPy, a low consumption board with WiFi and LoRa connectivity, able to execute scripts in MicroPython, and all of it geolocated to be able to study the exact point where the data is taken, and also, having a better control of each node.

This Master's Thesis is focused on a deployment strategy of a network of hundreds, even thousands, of nodes. This strategy describes how these nodes are going to be positioned, the development of the necessary code to carry out the necessary tasks of measuring of the sensors, geolocation, obtaining the networks, sending the data to a MQTT server, configuring the server and the development of the data base for the administration of the sleep hours of the device, nearby networks and the geolocation of each node. In addition to study other aspects of the loPy, such as its connectivity with certificates, energy consumption and possible delay of the clock.

Keywords

LoPy, Radiation, Solar energy, Geolocation, MQTT, LoRa, WiFi

Índice de contenidos

Índice de figuras

Agradecimientos

Agradecer en primer lugar a mis directores José Ignacio Gómez Pérez y Daniel Ángel Chaver Martínez por toda su ayuda, dedicación y buen hacer.

Agradecer a mi familia y cercanos todo su cariño y ánimos. Mis tíos y tías, primos, primas, hermanos y sobrinos.

A la maravillosa gente que me rodea, Adri, Alicia, Andoni, Antonio, Blind y Adela, Carla, Carlos, Chema, Censi, Dani y Viki, David, Desi, Elena y Gabi, Emilio e Irene, Erika, Isaac, Isra y Victoria, Gema, Lorena y Achokarlos, Järi, Jimmy, Marco y Judith, Mercedes, Miguel, Nacho y Nathalie, Pery, Primož y Lana, Ra, Raquel y Rebeca, Rosana, Rü y Fabia, San, Toni, Tito, Valky y Vicente.

Toda la gente de Valkyrie tours, Suiza, Eslovenia, Wacken… Os quiero.

A Tita Lola y Blas, a Lin, Alicia y Víctor.

A todos mis compañeros del Máster.

Y por supuesto, a Jose María y a mi Madre

1 - Introducción

El consumo energético en general y el de energía eléctrica en particular, por parte de la sociedad actual, se incrementa día a día a un ritmo exponencial, (AIE) tanto a nivel de la industria como de la agricultura, en los hogares y medios de transporte, por citar algunos ejemplos.

Una de las fuentes de energía utilizadas tradicionalmente es la energía fósil, entre las que se encuentran el petróleo, el gas natural y el carbón. Estas fuentes tienen diversos inconvenientes, ya que, durante su extracción, procesado y combustión producen gases tóxicos y de efecto invernadero. Otro inconveniente que presenta esta fuente de energía es que no son renovables por lo que las reservas se van agotando (García, 2007), además de que su reparto por zonas geográficas es muy desigual.

Otra fuente es la energía nuclear. Entre sus ventajas cuenta con que puede generar gran cantidad de energía por unidad de masa empleada, pero presenta numerosos problemas, sobretodo en cuanto a la gestión de los residuos radiactivos que genera, peligrosos y difíciles de eliminar, por no mencionar las normas de seguridad que requieren tener estas instalaciones. (BOE, 2014)

A la vez que los requerimientos de consumo de energía aumentan, su producción es la principal responsable del cambio de temperatura del planeta. Según el Foro Económico Mundial, en un informe (EAPI, 2017), afirma que “en marzo de 2016, y por primera vez desde que se tiene registro, los niveles mundiales de dióxido de carbono se mantuvieron por encima de 400 partes por millón durante un mes”. Ante esta situación se hace cada vez más necesario el uso de energías renovables (energía limpia) para producir energía eléctrica, ya que estas se muestran como la respuesta ideal para solucionar los problemas medioambientales y de sostenibilidad a los que nos enfrentamos como sociedad. Entre las energías renovables se encuentran la eólica, la mareomotriz, la hidráulica y la energía solar. El presente estudio se centrará en esta última.

La energía solar tiene una serie de características que la convierten en ideal: es la principal fuente de energía de nuestro planeta y en principio inagotable, es de producción local, y como ya se comentó anteriormente, limpia. Estas características han hecho que en los últimos años se haya apostado por el desarrollo de tecnologías solares limpias.

Fundamentalmente existen dos tipos de energía solar en función de cómo se realice su aprovechamiento: solar fotovoltaica y solar térmica.

La energía solar fotovoltaica se basa en utilizar el espectro de luz visible del sol para generar electricidad, por medio de células fotovoltaicas. Actualmente el desarrollo tecnológico ha experimentado un fuerte aumento enfocado a conseguir la máxima eficiencia de las células y los paneles solares (EEA, 2005). El trabajo realizado se ha centrado en medir la radiación producida para aprovechamiento de este tipo de energía solar.

La energía solar térmica consiste básicamente en el aprovechamiento de la capacidad calorífica de la luz solar, tiene diversos usos como calentar agua para uso sanitario y climatizar espacios. Nuestro país, además, cuenta con el privilegio de ser unos de los países con más horas de radiación solar de Europa. La Asociación de la industria fotovoltaica española (ASIF) cuenta en su página web con información detallada por provincias.

A pesar de todas las ventajas que presenta el uso de la energía solar para producir energía eléctrica, también presenta el importante problema de la dificultad de predecir cuánta energía solar se producirá en las próximas horas y en un determinado lugar. Hay que tener en cuenta que la energía eléctrica no se puede almacenar como tal, habría que transformarla en energía mecánica o química, por lo que se hace necesario que la oferta y la demanda sean muy similares en cada instante de tiempo, lo que implica necesariamente una coordinación de la producción de energía eléctrica, que usa la energía solar como fuente, con otras fuentes de producción. El reto pues al que se enfrenta la sociedad actual en esta materia consiste en disponer de un suministro de energía fiable, respetuoso con el medio ambiente y a un precio asequible (AEMA, 2017).

Lograr que la energía solar se convierta en este suministro fiable, pasa, entre otras cosas, por la capacidad de predecir la cuantía de su producción a nivel local, por ello, disponer de un modelo de predicción reportaría grandes beneficios, económicos entre otros.

La problemática viene al elegir dónde poner, ya sean unos pocos paneles solares, ya sea una granja de éstos, para poder aprovechar al máximo la radiación solar con el propósito de lograr la mayor obtención de energía. Y aquí es dónde el IoT (Internet Of Things, Internet de las Cosas) cobra mucha importancia pues, se disponen de dispositivos dotados de una serie de sensores capaces de medir esta radiación solar en puntos concretos, que sean autónomos (alimentándose de hecho, de esta misma energía solar) y enviar estos datos por redes inalámbricas para su posterior estudio.

Para llevar a cabo este estudio, hemos configurado un dispositivo dotado de los sensores necesarios para estas mediciones, conectividades inalámbricas necesarias para su comunicación, un panel solar y una batería para su alimentación. Como ya se ha mencionado en el resumen, este dispositivo está dotado de:

- Una LoPy (puede verse en la Ilustración 1) (Pycom, 2018)
- Placa desarrollada por la UCM (Universidad Complutense de Madrid) para conectar más sensores y una batería externa, así como el panel solar
- Los sensores necesarios para medir radiación, temperatura, humedad y GPS (Global Position System).

Ilustración 1 – LoPy

Como ya se ha mencionado, el objetivo principal de este trabajo es el desarrollo de un plan de despliegue de estos sensores, el cual se encuentra más detallado en el Capítulo 5. Este plan de despliegue constará de:

- Problemática de las redes con cientos/miles de nodos
- Configuración
- Corrección de errores

1.1 Objetivos propuestos

El desarrollo de una red de nodos con sensores conectados a una red para el envío de sus medidas, para su posterior análisis. Esta red tiene que ser capaz de poder identificar si alguno de los nodos no está funcionando correctamente, ha sido reemplazado o desplazado y realizar su actualización, entre otros aspectos importantes que se describen a continuación.

1.1.1 Servidor de datos

La red de nodos recoge una serie de datos que deben ser enviados a un servidor para su posterior análisis. Elegir qué protocolos de comunicación y la funcionalidad global de éste es un elemento principal para el desarrollo del trabajo.

1.1.2 Geolocalización

Una de las principales características de este dispositivo en su geolocalización. Para ello, se ha conectado y configurado un sistema GPS al dispositivo.

1.1.3 Automatización de la administración

Automatizar la administración de los nodos es de vital importancia, dado que, al tener cientos o miles de nodos, automatizar muchas de las tareas comunes y diarias de los nodos es de gran importancia. De Estas tareas son:

- Asignación del nodeId para cada nodo dependiendo de sus coordenadas.
- Comunicación de un listado de redes a las que poder conectarse.
- Hora de dormir y despertar. Los nodos deben estar inactivos por la noche, pero las horas de sol varían de un día para otro agudizándose esta diferencia entre invierno y verano, por lo que los nodos deben cada día obtener a qué hora entrarán en modo de sueño profundo y cuándo despertarán de él
- Tener controlada la posición de cada nodo.

1.1.4 Sincronización de los nodos

Uno de los objetivos principales del proyecto es una correcta sincronización entre todos los nodos. Es importante que los nodos obtengan la fecha y hora de una fuente externa pues, cada nodo, al iniciar o despertar, tiene su reloj interno puesto en época (0:0:0-1/1/1970), y es de suma importancia que los datos recibidos de distintos nodos correspondan con el mismo momento.

Que esta fecha y hora sea obtenida de un servidor suele ser una buena solución, pero se puede tener el problema de que esta hora puede no llegar al mismo tiempo a todos los nodos, por lo que las tramas no coincidirían en el tiempo al no existir una correcta sincronización.

1.1.5 Conexión por redes

Por la naturaleza del IoT, y este proyecto en concreto, es necesario que el dispositivo esté conectado a la red. La casuística de este proyecto hace que la red LoRa (Long Range, largo alcance) resulte muy interesante para cubrir grandes áreas, donde puede no llegar ninguna red WiFi conocida, y dotarlas de conectividad a internet.

1.1.6 Dispositivo autónomo

Cada dispositivo debe ser capaz de ser autónomo, alimentándose de energía solar y en ciertos casos concretos, de una batería conectada a la placa, la cual se recarga también mediante la placa solar.

2 - Introduction

Energy consumption in general and electricity consumption in particular, by today's society, is increasing day by day at an exponential rate, (AIE) both at the level of industry and agriculture, in households and the media. transport, to name a few examples. One of the energy sources traditionally used is fossil energy, among which are petroleum, natural gas and coal. These sources have several drawbacks, since during their extraction, processing and combustion they produce toxic gases and greenhouse gases. Another drawback of this energy source is that they are not renewable, so the reserves are running out (García, 2007), in addition to their geographical distribution is very unequal.

Another source is nuclear energy. Among its advantages is that it can generate a large amount of energy per unit of mass used, but it presents numerous problems, especially in terms of the management of the radioactive waste it generates, dangerous and difficult to eliminate, not to mention the safety standards that they need to have these facilities. (BOE, 2014).

At the same time that the energy consumption requirements increase, its production is mainly responsible for the change in the temperature of the planet. According to the World Economic Forum, in a report (EAPI, 2017), states that "in March 2016, and for the first time since registration, global levels of carbon dioxide remained above 400 parts per million during one month". Faced with this situation, the use of renewable energies (clean energy) is becoming increasingly necessary to produce electrical energy, since these are shown as the ideal response to solve the environmental and sustainability problems that we face as a society. Among renewable energies are wind, tidal, hydraulics and solar energy. The present study will focus on the this one.

Solar energy has a series of characteristics that make it ideal: it is the main source of energy for our planet and, in principle, inexhaustible, it is locally produced, and as previously mentioned, it is clean. These characteristics have made in recent years a commitment to the development of clean solar technologies.

There are basically two types of solar energy depending on how it is used: solar photovoltaic and solar thermal.

Photovoltaic solar energy is based on using the visible light spectrum of the sun to generate electricity, by means of photovoltaic cells. Currently, technological development has

experienced a sharp increase focused on achieving the maximum efficiency of cells and solar panels (EEA, 2005). The work done has focused on measuring the radiation produced for the use of this type of solar energy.

Thermal solar energy basically consists of the use of the heat capacity of sunlight, has various uses such as heating water for sanitary use and air conditioning spaces. Our country also has the privilege of being one of the countries with the longest hours of solar radiation in Europe. The Association of the Spanish photovoltaic industry (ASIF) has on its website detailed information by provinces.

Despite all the advantages of using solar energy to produce electricity, it also presents the important problem of the difficulty of predicting how much solar energy will be produced in the next hours and in a certain place. Keep in mind that electrical energy cannot be stored as such, it should be transformed into mechanical or chemical energy, so it is necessary that supply and demand are very similar at each instant of time, which necessarily implies a coordination of the production of electrical energy, which uses solar energy as a source, with other sources of production. The challenge facing today's society in this matter is to have a reliable energy supply, respecting the environment and at an affordable price (EEA, 2017).

Achieving that solar energy becomes this reliable supply, happens, among other things, by the ability to predict the amount of its production at the local level, therefore, having a prediction model would yield great economic benefits, among others.

The problem comes when choosing where to put, whether a few solar panels, or a farm of these, in order to maximize the solar radiation in order to achieve the highest energy yield. And this is where the IoT (Internet of Things) is very important because there are devices equipped with a series of sensors capable of measuring this solar radiation at specific points, which are autonomous (feeding in fact, from this same solar energy) and send these data over wireless networks for further study.

To carry out this study, we have configured a device equipped with the necessary sensors for these measurements, wireless connectivity necessary for its communication, a solar panel and a battery for its power. As already mentioned in the summary, this device is equipped with:

- One LoPy (can be seen in Ilustración 1) (Pycom, 2018)

- Plate developed by the UCM (Universidad Complutense de Madrid) to connect more sensors and an external battery, as well as the solar panel.

- The sensors needed to measure radiation, temperature, humidity and GPS (Global Position System).

As already mentioned, the main objective of this work is the development of a deployment plan for these sensors, which is more detailed in 4. This deployment plan will consist of:

- Problems of networks with hundreds / thousands of nodes
- Configuration
- Error correction

2.1 Proposed objectives

El desarrollo de una red de nodos con sensores conectados a una red para el envío de sus medidas, para su posterior análisis. Esta red tiene que ser capaz de poder identificar si alguno de los nodos no está funcionando correctamente, ha sido reemplazado o desplazado y realizar su actualización, entre otros aspectos importantes que se describen a continuación.

2.1.1 Data server

The network of nodes collects a series of data that must be sent to a server for further analysis. Choose which communication protocols and the global functionality of this is a main element for the development of the work

2.1.2 Geolocation

One of the main features of this device is its geolocation. To do this, a GPS system has been connected and configured to the device.

2.1.3 Administration automation

Automating the administration of the nodes is of vital importance, since, having hundreds or thousands of nodes, automating many of the common and daily tasks of the nodes is of great importance. These tasks are:

- Assignment of the nodeId for each node depending on its coordinates.
- Communication of a list of networks to which you can connect.
- Time to sleep and wake up. The nodes must be inactive at night, but the sun hours

vary from one day to the next, making this difference between winter and summer worse, so the nodes must each day obtain at what time they will enter in deep sleep mode and when they will wake up from it

- Control the position of each node.

2.1.4 Synchronization of the nodes

One of the main objectives of the project is a correct synchronization between all the nodes. It is important that the nodes obtain the date and time from an external source, since each node, when starting or waking up, has its internal clock set at time (0: 0: 0-1 / 1/1970), and it is of utmost importance that the data received from different nodes correspond with the same moment.

That this date and time is obtained from a server is usually a good solution, but you can have the problem that this time may not arrive at all the nodes at the same time, so the frames would not coincide in time because they do not exist a correct synchronization

2.1.5 Connection by networks

Due to the nature of the IoT, and this project in particular, it is necessary that the device is connected to the network. The casuistry of this project makes the LoRa network (Long Range, long range) very interesting to cover large areas, where no known WiFi network can reach, and provide them with Internet connectivity.

2.1.6 Autonomous device

Each device must be able to be autonomous, feeding on solar energy and in certain specific cases, a battery connected to the plate, which is also recharged by the solar panel.

3 – Arquitectura global del sistema

El proyecto se ha realizado usando una LoPy de Pycom, con su placa de expansión y una placa creada en la propia Universidad para poder conectar más dispositivos, así como una entrada de alimentación mediante un panel solar y una entrada para una batería, que se carga cuando hay sol y se usa como alimentación cuando falta este.

Se ha elegido este dispositivo por su reducido tamaño, buena conectividad, bajo consumo y la capacidad de conectar múltiples sensores, así como su potencial por ser capaz de ejecutar scripts en MicroPython, además de su ajustado precio.

La arquitectura global del sistema puede apreciarse en la Ilustración 2. Se aprecia un conjunto de nodos, conectados a un Gateway ya sea LoRa o WiFi, para después estar conectado con el servidor de administración y el bróker.

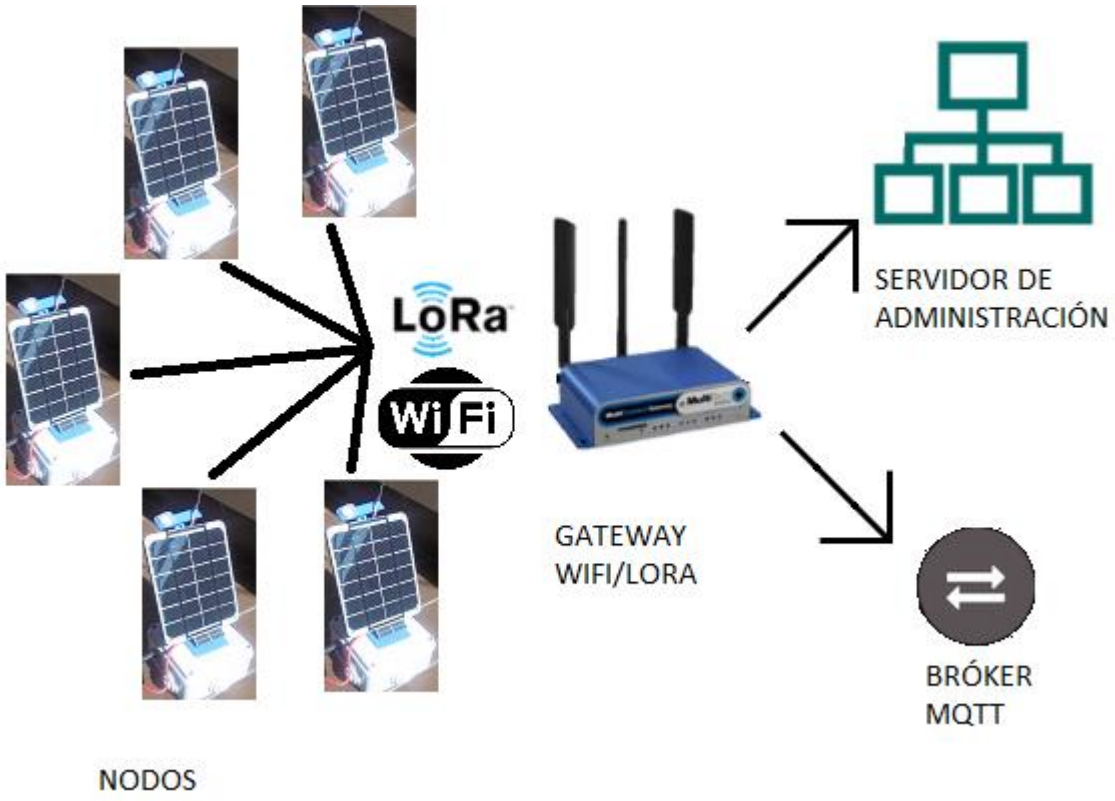

Ilustración 2 - Arquitectura del Sistema

3.1 Estudio de viabilidad

El siguiente apartado tiene como objetivo comprobar de qué tecnología se dispone en el momento actual, ver si satisface las necesidades requeridas por el proyecto, o al menos en qué medida, o comprobar qué dispositivos hardware hacen falta.

3.1.1 Estudio de la tecnología actual

Para realizar el proyecto hay que comprobar qué dispositivos necesitamos, y los productos de Pycom dan una buena solución para este tipo de despliegues. Pycom es una compañía especializada en el desarrollo de dispositivos IoT. Su objetivo es crear plataformas IoT revolucionarias, rápidas y fáciles de usar, dándole a los desarrolladores lo que necesitan: un camino corto desde el prototipado hasta el desarrollo final, cerrando la brecha que existía entre los desarrolladores y las compañías. Todo esto se consigue con un gran catálogo de desarrollos de código abierto, placas programables y MicroPython y módulos OEM (hardware de terceros integrados en sus productos).

En este momento, no hay ningún otro microcontrolador a este precio que tenga las 3 conectividades que este ofrece, en cuanto a su tamaño, potencia y consumo, por lo que es la solución perfecta para el problema que aquí se plantea. A continuación, explicaremos otros dispositivos de Pycom de características similares a LoPy (Pycom); la descripción detallada de LoPy se realizará más adelante (LoPy).

3.1.1.1 WiPy

Crea un punto de acceso WiFi y además tiene conectividad Bluetooth. Ya se encuentra en su versión 3.0, que mejora la RAM (4MB) y la FLASH (8MB). Su alcance de WiFi es de 1KM, muy útil para dar cobertura en grandes zonas.

3.1.1.2 SiPy

Incluye conectividad Sigfox, WiFi y Bluetooth, siendo el único microcontrolador a día de hoy que incluye estas 3 conexiones.

3.1.1.3 GPy

Incluye conectividad WiFi, Bluetooth y LTE, lo que lo hace muy útil para dotar de conexión a la red en zonas donde es difícil proporcionar un punto de acceso.

3.1.1.4 FiPy

Es el más completo de todos, pues incluye todas las conexiones: WiFi, Bluetooth, LoRa, Sigfox y LTE.

Tabla 1 - Comparativa distintos módulos Pycom

Module	WiFi	Bluetooth	LoRa	Sigfox	LTE CAT-M1NB-IoT
WiPy 3.0	✓	✓			
SiPy	✓	✓		✓	
GPy	✓	✓			✓
LoPy	✓	✓	✓		
LoPy4	✓	✓	✓	✓	
FiPy	✓	✓	✓	✓	✓

3.1.2 Tecnologías y protocolos de comunicación

3.1.2.1 Tecnologías a nivel físico

La elección de los protocolos de comunicación ha sido decisiva, ya que el dispositivo estará enviando datos constantemente y es una de las acciones que más energía consume. Además, los nodos van a estar desplegados en lugares remotos, por lo que la elección de una red con largo alcance, bajo consumo y que sea capaz de soportar el tamaño y número de tramas a enviar es fundamental. LoPy admite comunicación por WiFi y LoRa.

La red LoRa (S.L., 2017) es una tecnología de comunicación que se encuentra en el nivel 1 de la pila OSI, el nivel físico, que permite el envío y recepción de datos punto a punto. Entre sus ventajas, destaca su largo alcance, usando la técnica de espectro ensanchado (Rouse, 2012), que consiste en permitir que una señal que se transmite a través de una banda de frecuencia sea más ancha de lo realmente necesario. Para ello, el transmisor extiende la energía a través de un número de canales de la banda de frecuencia de un espectro electromagnético más amplio, permitiendo así una recepción de múltiples señales a la vez que tengan distinta velocidad. Las bandas en las que suele operar LoRa están por debajo de 1GHz, normalmente en los 433MHz, 868MHz y 915 MHz, aunque esta última no se puede usar en Europa. Su alcance puede llegar a ser de hasta unos 20KM en situaciones muy favorables.

Establecer redes LoRa de distancias cortas (1KM o menos) es relativamente sencillo. El problema viene cuando las distancias deben ser más grandes, ya que requieren más infraestructura a nivel de antenas y un mayor cifrado de los datos para evitar que terceros obtengan estos datos. Una vez establecida la red, lo único que hay que hacer es registrar e identificar el dispositivo dentro de una red.

Además, LoRa se va abriendo cada vez más camino en el mundo del IoT, ya que otros dispositivos como Arduino o Raspberry Pi ya están empezando a desarrollar proyectos con esta red. Además, existen varias empresas que crean y comercializan módulos para integrar LoRa haciéndolos prácticamente *"Plug and play"*, y entre esas compañías tenemos Pycom y el módulo elegido, el LoPy.

Pero no todo son ventajas en LoRa. Tiene una velocidad de trasmisión muy baja, de 0.3 a 27kbps. Esto hace que 1MB de información tarde entre 5 minutos y medio (a 27kbps) hasta 8 horas y 20 minutos (a 0.3kbps). Para este proyecto, las tramas a enviar son de 58-60 bytes, por lo que en el peor de los casos, toma 1 segundo para su envío.

Otra tecnología de red de bajo consumo y largo alcance es Sigfox (Sigfox, 2012), pero presenta desventajas, como que las tramas máximas son de 12 bytes, o la que la hace prácticamente inviable para este proyecto, sólo puede enviar un máximo de 140 mensajes por nodo al día, por lo que ha sido descartada.

3.1.2.2 Protocolos de comunicación

Existen varios protocolos de comunicación orientados a dispositivos IoT. Estos protocolos suelen tener características comunes, como bajo consumo. Entre los más destacados, se encuentran MQTT y CoAP (Constrained Application Protocol). Ambos pertenecen a la capa de *"Aplicación"* del modelo OSI.

MQTT es un protocolo basado en un sistema de publicación y subscripción, bastante parecido a Twitter. Los elementos principales son los publicadores, los subscriptores y el bróker. El servidor MQTT contiene *topics*, o temas en castellano, que son como una especie de canales en los que los publicadores publican mensajes y los subscriptores reciben los mensajes de los temas a los que están subscritos. El bróker es el que se encarga de controlar estos *topics* y de enviar a los subscriptores los mensajes enviados por los publicadores. En la Ilustración 3 se puede ver un ejemplo de su funcionamiento.

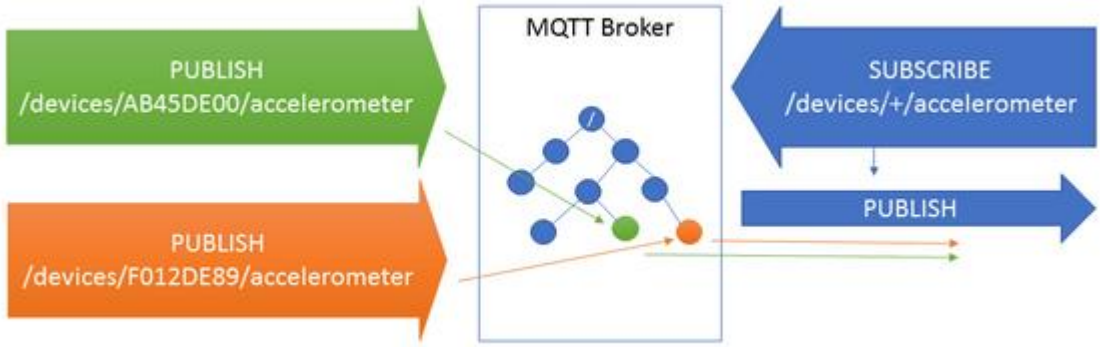

Ilustración 3 - Funcionamiento de MQTT

CoAP es más parecido al funcionamiento tradicional de servicios web en los que se piden recursos y a veces publican tus propios datos. Es compatible con HTTP y muy usado en comunicaciones M2M. Los dispositivos se registran en el servidor ofreciendo sus recursos disponibles, como servicios o sensores. Los clientes que se conectan al servidor CoAP para beneficiarse de estos recursos han de pedirlos. Se puede observar un ejemplo de su funcionamiento en Ilustración 4.

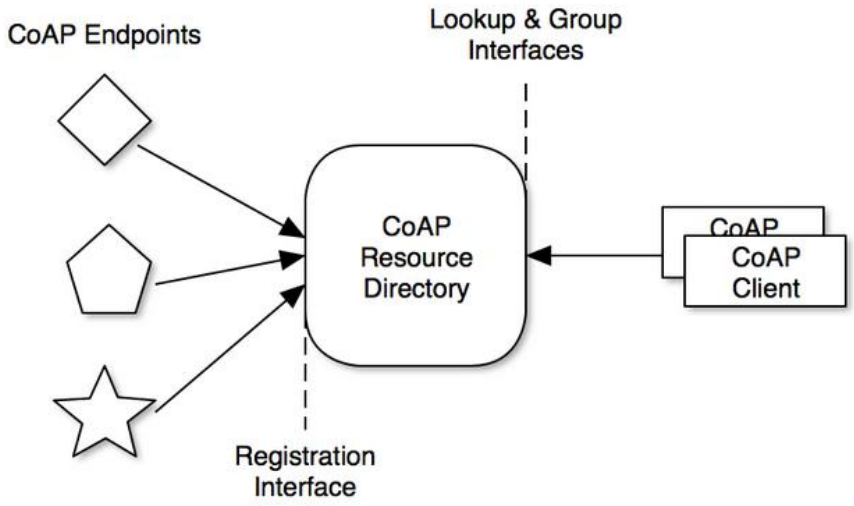

Ilustración 4 - Funcionamiento de CoAP

3.2 Selección de la solución

Por la naturaleza del IoT, y este proyecto en concreto, es necesario que el dispositivo esté conectado a la red. La casuística de este proyecto hace que la red LoRa (Long Range, largo alcance) resulte muy interesante para cubrir grandes áreas, donde puede no llegar ninguna red

WiFi conocida, y dotarlas de conectividad a internet. La elección del dispositivo LoPy para cada nodo es por tanto adecuada a este respecto, pues dispone de conexión LoRa. Además de ser una solución viable a un precio ajustado. Los sensores usados en el proyecto son el de radiación, temperatura exterior, temperatura y humedad relativa y GPS descritos en el apartado Descripción del LoPy y sensores. El dispositivo está dotado de una batería y un panel solar para poder ser autónomo.

Como protocolo de comunicación se ha elegido MQTT por su ligereza, fácil implementación y la posibilidad de poder distinguir entre los distintos dispositivos conectados, además de la opción de subscripción que puede ser muy útil y facilitadora a la hora de almacenar y estudiar los datos obtenidos de los nodos. El bróker elegido, el suministrado por Adafruit.

El servidor necesario para poder garantizar el correcto funcionamiento de este trabajo debe ser capaz de ejecutar Apache (servidor web), scripts en PHP y administrar una base de datos en MySQL, por lo que la instalación de XAMPP (paquete de software que incluye todas las funcionalidades descritas) es una opción muy recomendable.

Para realizar el código, se ha usado el programa Atom (IDE recomendado para programar en MicroPython para la LoPy) con el plugin Pymakr (Plugin que añade a Atom la funcionalidad necesaria para la comunicación del IDE con el dispositivo) con la LoPy conectada mediante USB. Además de programar, este programa nos proporciona una consola de comandos muy útil para probar trozos de código rápidamente y para poder comprobar, mediante trazas, el código.

3.3 Descripción del LoPy y sensores

Este trabajo comprende una parte hardware y otra parte software. Aunque el grueso del trabajo ha sido el desarrollo software, no ha faltado la interacción con el hardware, así como su entendimiento para poder conectar sensores.

3.3.1 LoPy

La LoPy de Pycom es un dispositivo con conectividad LoRa, WiFi y BLE, siendo así el único dispositivo que dispone de las 3 conectividades en el mercado hoy en día, perfecto para dispositivos IoT. Gracias al chipset Espressif ESP32, LoPy ofrece una combinación perfecta de consumo y flexibilidad, haciendo muy rápido y fácil crear y conectar dispositivos.

Puede actuar como Gateway de LoRa o una plataforma de desarrollo multi-portador (debido a sus 3 conectividades). Es programable con MicroPython usando el IDE de Pymakr. En este proyecto, la LoPy se ha conectado a varios sensores, para medir humedad, radiación y temperatura.

3.3.2 Sensor de radiación

El modelo elegido ha sido la SLMD121H04L de la compañía IXYS, modelo (IXYS, 2018). No es un sensor de radiacoón como tal, sino una placa solar pequeña de 4 células que funciona, la corriente generada por la radiación se asume como proporcional a la radiación recibida. Es decir, no mide la radiación como un piranómetro, pero es perfectamente válido para saber en qué zonas hay más radiación o menos de una forma proporcional. Debe colocarse como se muestra en la Ilustración 5. Los valores típicos deben de estar sobre los 3000. Está conectado al puerto situado en el lateral de la placa.

Ilustración 5 - Sensor de radiación

3.3.3 Dispositivo GPS

El GPS ha sido uno de los principales desafíos de este trabajo. No es un sensor como tal, sino un dispositivo dentro del nodo. Su función es vital, como se explica en el trabajo, ya que uno de los objetivos principales es la geolocalización. Es un GPS de tamaño pequeño y bajo consumo, para que tenga espacio dentro de la caja y no haga mucho uso de la batería/panel solar. Se puede observar el utilizado en este proyecto en la Ilustración 6.

La sigla GPS (Christensson, 2018) significa *"Global Position System"*. Es un sistema de navegación por satélite usado para determinar la posición en la Tierra de un objeto. En un principio, fue usado por el ejército de los EEUU en los años sesenta, más tarde se expandió para uso civil. Se puede encontrar en numerosos aparatos electrónicos de hoy en día, como Smartphones, tablets o dispositivos de navegación.

Ilustración 6 – GPS

GPS consta de 24 satélites desplegados en el espacio a unos 19,000KM por encima de la superficie de la Tierra, orbitan cada 12 horas a unos 11,200KM por hora. Cada satélite emite mensajes en los que se incluyen la posición del satélite, órbita y la hora exacta, dato usado en este trabajo.

Para determinar la posición de un objeto en la Tierra, un receptor GPS necesita de 3 satélites GPS y mediante un proceso llamado triangulación (Masmar, 2008), se obtiene la posición exacta del objeto. Esta triangulación consiste en tener comunicación con 3 satélites GPS. Sabiendo la distancia a la que se está de cada uno, cada satélite "dibujará" una circunferencia con eses radio. Las 3 circunferencias coincidirán en 2 puntos, uno de ellos muy improbable porque no estaría en la Tierra, sino sobre ella, y el punto restante es la posición del

dispositivo. Por este motivo, es muy difícil de usar en interiores, y ha traído algunos problemas a la hora de la realización de este trabajo.

El GPS usado en este trabajo ha sido uno de bajo coste, ya que no se requiere una precisión exacta del nodo, y dado que va a estar siempre en exterior, esto facilita la obtención de las coordenadas.

El dispositivo, el GPS está conectado en los pines 22 y 23 de la placa, además de la toma de tierra y a un conector de alimentación de 5V. Mediante este dispositivo se geolocaliza el dispositivo, además de obtener la fecha y hora, ya que cada vez que se inicia, su reloj está en época, y es vital saber la fecha y hora y que todos los dispositivos la adquieran del mismo lugar para que estén sincronizados. Si ésta sincronización fuese mediante un servidor, podría haber pequeños desajustes de segundos.

Las coordenadas recibidas de los satélites GPS están en formato *"Grados Decimales"*. A diferencia de las coordenadas en grados, minutos y segundos, las coordenadas son expresadas con sólo un número en decimal. La conversión es sencilla, como se aprecia en la Ilustración 7:

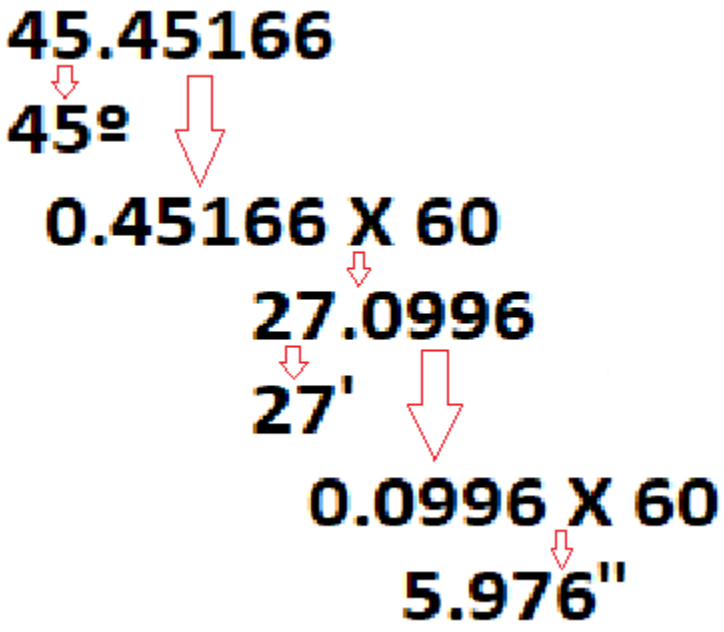

Ilustración 7 - Ejemplo conversión coordenadas

Además, el dispositivo GPS es capaz de administrar la fecha y la hora al dispositivo, y se ha elegido esta opción ya que su reloj es muy preciso y no genera ningún tipo de retraso, además de ser una fuente común para todos los nodos, por lo que la sincronización de los nodos está garantizada.

3.3.4 Sensor de temperatura exterior

Uno de los sensores de más interés que se han conectado a la placa es el de temperatura exterior, el cual está situado al lado del sensor de radiación. Es un sensor de sonda activa, es decir, que da un voltaje según la temperatura a la que esté y el regulador, con ese voltaje, lo calibra para obtener la temperatura. Puede observarse en la Ilustración 8.

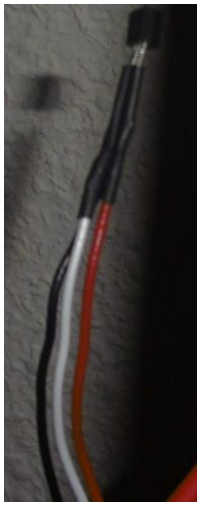

Ilustración 8 - Sensor de temperatura

3.3.5 Sensor de temperatura y humedad relativa

El sensor, modelo AM2302 (Adafruit, Adafruit AM2302, s.f.), que se coloca dentro de la caja y sirve para medir la humedad y una corrección de la temperatura usando la temperatura de dentro de la caja del dispositivo. Para su funcionamiento, lleva una especie de membrana para medir la humedad, según ésta va variando el voltaje o la intensidad según el tipo de sonda. Está desarrollado y distribuido por Adafruit. Puede observarse en la Ilustración 9.

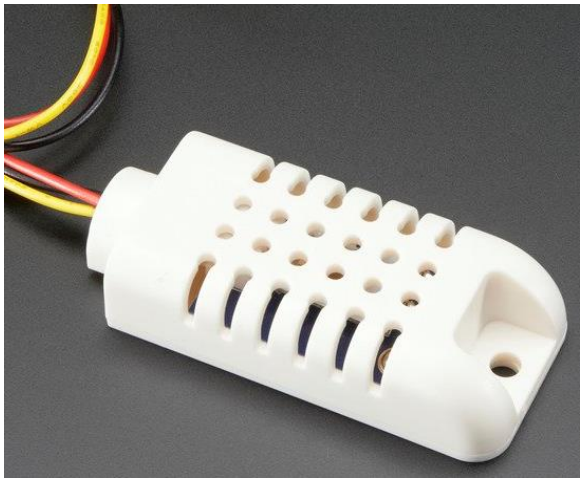

Ilustración 9 - Sensor humedad

3.3.6 Aspecto final del dispositivo

Con todos los nodos conectados y dentro de su caja, el dispositivo queda como se observa en la Ilustración 10.

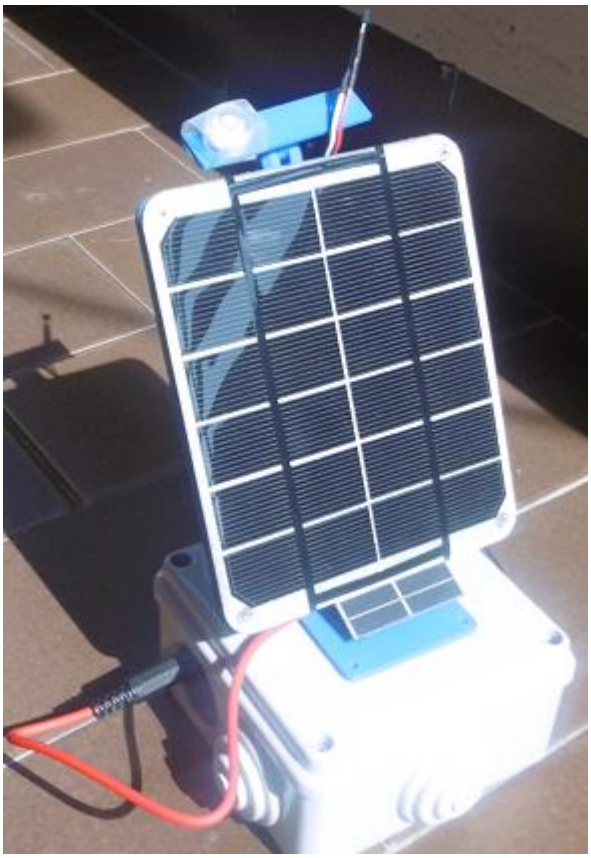

Ilustración 10 - Aspecto final del dispositivo

3.4 Servidor de administración de nodos

Para automatizar las tareas de administración y diarias de los nodos, se ha creado un servidor de administración. El dispositivo debe conectarse a esteservidor con una base de datos para establecer una serie de parámetros. Basándose en las coordenadas, el servidor realizará unas acciones u otras. Con el servidor podemos:

- Llevar un registro de los nodos y dónde están
- Asignar a cada nodo las redes WiFi a las que podrá conectarse según sus coordenadas
- Establecer la hora a la que el nodo se despierta y se duerme

Cada una de las funcionalidades del servidor es una tabla de la base de datos sin relación con el resto de tablas, como puede observarse en la Ilustración 11.

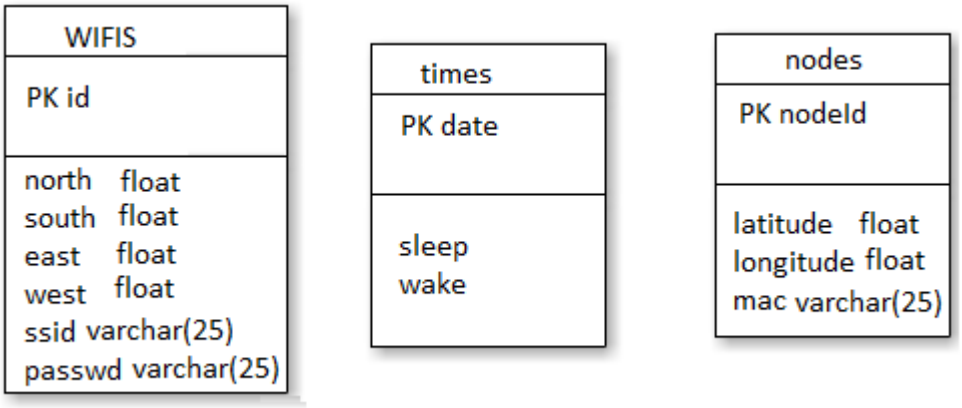

Ilustración 11 - Tablas base de datos

3.5 Servidor MQTT

El servidor MQTT y su bróker se han implementado en un servidor de Adafruit (io.adafruit.com, compañía desarrolladora y de venta de elementos de IoT, recomendada por Pycom para la configuración de este servidor), el cual permite la creación de varios *topics* y una monitorización de los nodos en tiempo real. Un ejemplo de esto se puede observar en la Ilustración 12.

Ilustración 12 - Monitorización servidor MQTT

Adafruit es una compañía de hardware *"open-source"*, lo que quiere decir, que se dedica a crear partes físicas de tecnología diseñadas para poder conectarse entre sí y de código abierto. Crea placas y sensores, entre otras cosas, y tiene un apartado dedicado especialmente al IoT. entre sus servicios disponibles está el de la creación de un servido MQTT, que es lo que recomienda la documentación de LoPy, que es gratuito la versión de prueba y tiene una versión de pago a razón de 10$ al mes, o 99$ al año. Se ha optado por la versión gratuita para comprobar su funcionamiento, la cual permite hasta 5 *topics* y guarda los mensajes recibidos durante un mes, suficiente para la realización de este trabajo.

Además, este servidor proporciona una opción de *"dashboard"* funcionalidad que implementa una interfaz gráfica en la que poder observar con mayor facilidad el volumen de datos que envía cada uno de los nodos.

4 Desarrollo del contenido del proyecto

El desarrollo de este trabajo ha requerido de un análisis de cómo implementar la solución elegida, desarrollo del código y configuración del servidor. El lenguaje ha sido MicroPython, no por elección, sino porque es el único lenguaje que acepta la LoPy.

4.1 Modelo de programación MicroPython

A medida que los sistemas integrados de tiempo real se vuelven más complejos y requieren un profundo conocimiento de los microcontroladores de 32 bits, sensores y protocolos, entre otros, los ciclos de desarrollo son más cortos, los equipos de desarrollo necesitan encontrar formas de acelerar el diseño y al mismo tiempo de poder ofrecer puertos al código para nuevos productos: una plataforma de desarrollo integrada y flexible es necesaria (Beningo, 2017).

Hay varias plataformas de microcontroladores disponibles para acelerar el proceso de desarrollo, pero está el problema de limitarse a un único proveedor, con los problemas que puede generar el depender de una compañía y que la integración con dispositivos de otras puede ser costoso. Por ello, una buena solución que está siendo muy aceptada es acoplar el microcontrolador de bajo nivel hardware con un lenguaje de programación de alto nivel, como Python, y una de esas soluciones es MicroPython. Se puede ejecutar en distintas piezas de proveedores de microcontroladores, es de código abierto y fácilmente personalizable por los desarrolladores según sus necesidades.

El modelo de programación MicroPython (George, 2014-2018) es una implementación liviana y eficiente de Python 3 (Python, 2008) que incluye un pequeño conjunto de librerías estándar y su optimización para ser ejecutadas en microcontroladores y entornos pequeños. MicroPython está dotado de muchas características avanzadas, como una línea de comandos interactiva o control de excepciones.

MicroPython proporciona una consola de comandos así como la opción de importar y ejecutar scripts almacenados en el sistema, y es tan parecido a Python que, si ya lo conoces, ya sabes MicroPython. Además, incluye módulos como “machine” para poder acceder a hardware de bajo nivel.

Como ya se ha mencionado, MicroPython se puede ejecutar en distintos microcontroladores, pero tiene unos requisitos mínimos de hardware:

- 256KB de flash
- 16KB de RAM
- 80 Mhz de CPU

Hay una serie de “buenas prácticas” a la hora de programar en MicroPython, que resultan de interés dado que, para este proyecto, la calidad y optimización del código son muy importantes:

- Utilizar búferes preasignados para canales de comunicación
- Utilizar el método `readinto` al utilizar periféricos de comunicación
- Evitar la documentación de Python tradicional usando ###
- Minimizar la creación y destrucción de objetos en tiempo de ejecución
- Supervisar los tiempos de ejecución de la aplicación

MicroPython es una interesante plataforma para desarrolladores que deseen implementar aplicaciones integradas en tiempo real para microcontroladores. El hecho de poder crear scripts en alto nivel y ejecutarlos en cualquier microcontrolador proporciona las siguientes ventajas:

- Mejora la reutilización de la aplicación
- Acelera el tiempo de llegada al mercado

Desacopla la aplicación del hardware

4.2 Análisis del código existente

Al comenzar a realizar el proyecto ya había parte del código desarrollado. Éste código aportaba la funcionalidad necesaria para obtener los datos de las medidas realizadas por los sensores y mostrarlas por pantalla. El código se ha completado con la implementación de la funcionalidad relacionada con MQTT, uso del GPS, conectividad con el servidor de administración y conectividad a redes.

4.3 Propuesta de mejora

En un principio, el dispositivo solo tomaba las medidas por los sensores cuando se le indicaba por medio de la consola de comandos. Las mejoras consistieron en:

- Creación y comunicación con el servidor
- Comunicación por WiFi y LoRa
- Toma de medidas periódicas y envío periódico de la media de estas por MQTT

- Geolocalización y establecimiento de fecha y hora por GPS

La Ilustración 13 muestra las distintas fases por las que pasa el dispositivo.

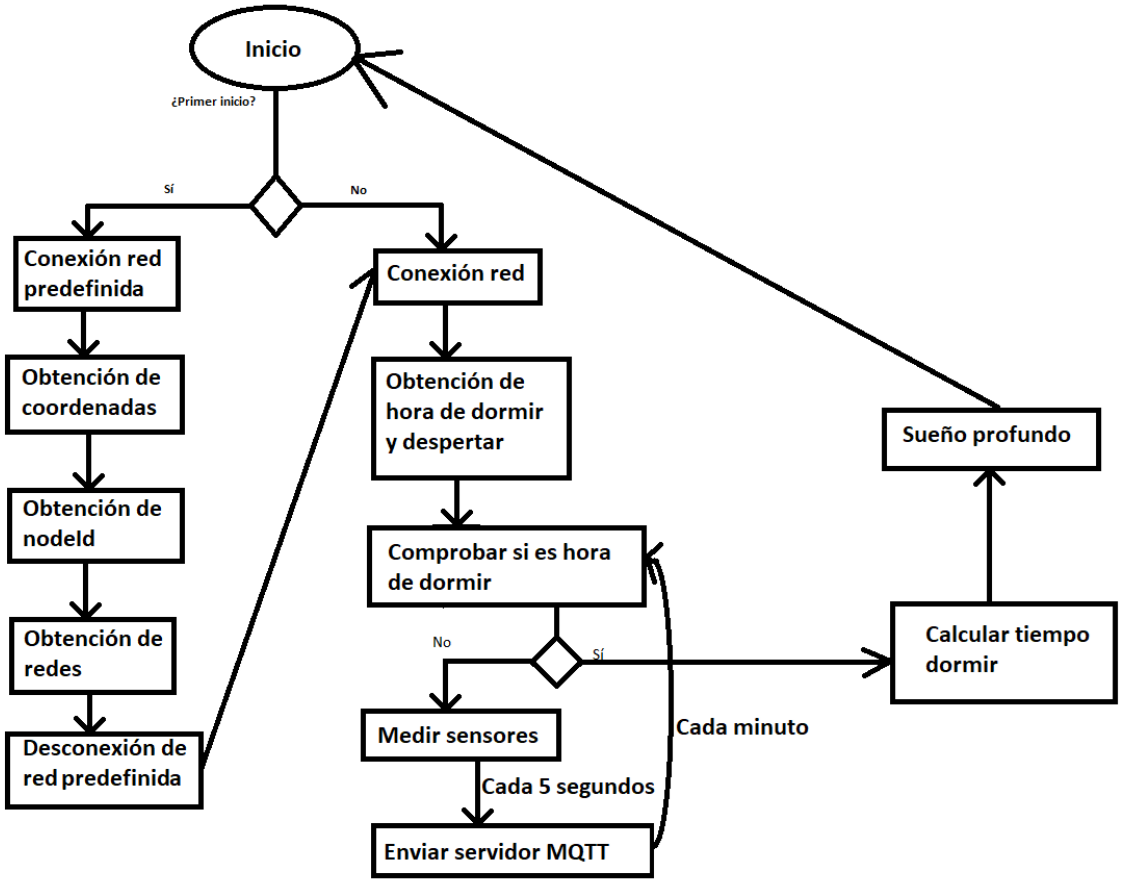

Ilustración 13 - Máquina de estados

4.3.1 GPS

Una de las mejoras más importantes que se han hecho ha sido la adición del código necesario para el uso del GPS, así como conectarlo a la placa para su uso. Para este último paso se estudió el *"datasheet"* de la LoPy. El GPS debe estar conectado a pines UART, en este caso, TX y RX. Los puertos RX se usan para la entrada de datos, dado que el GPS obtiene datos y se los envía al programa. El puerto TX se usa para poder enviarle instrucciones al dispositivo. Los puertos elegidos en el desarrollo del trabajo han sido el GPIO14 (TX) y GPIO 25 (RX), por su proximidad a los pines de alimentación y toma de tierra que hay en la placa. En la placa de

expansión corresponden con los conectores P22 y P23 Ilustración 14, dato importante, ya que en el código hay que hacer referencia a esos pines.

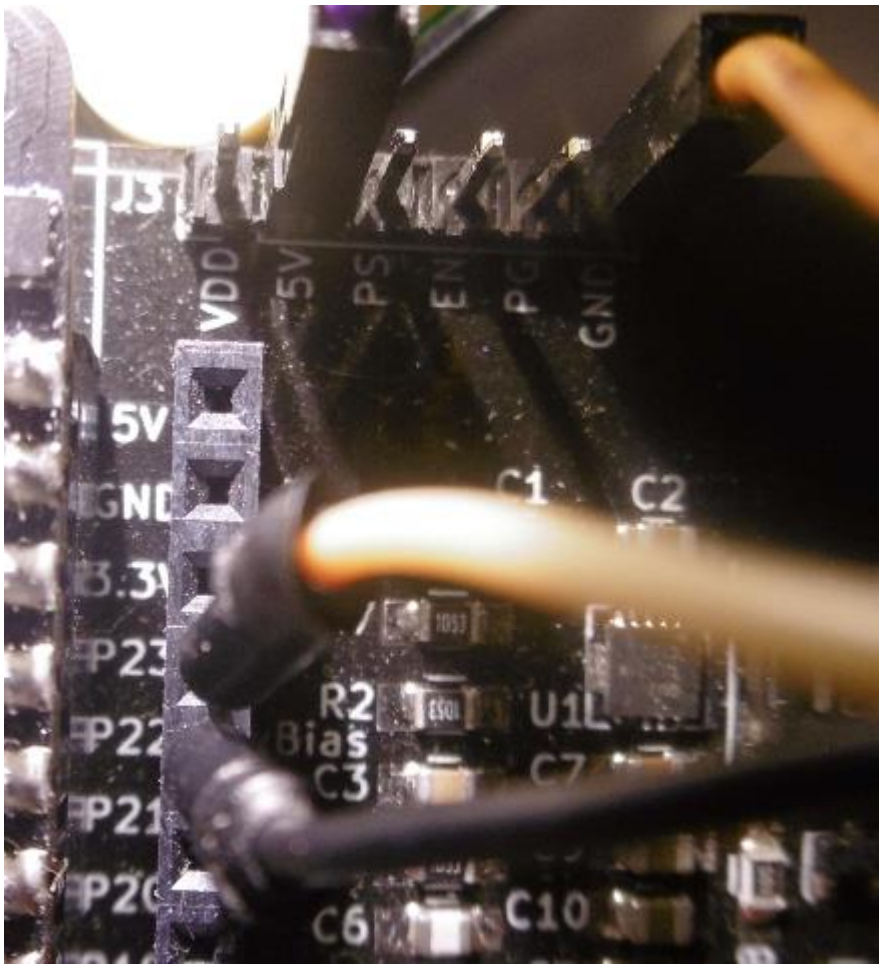

Ilustración 14 - Pines GPS

Tras su conexión con la placa, el siguiente paso a realizar es el código necesario para obtener los datos deseados del GPS. El dispositivo GPS recibe muchos tipos de tramas con mucha cantidad de información; la necesaria para este trabajo es la de la cabecera *"$'GPRMC"*, la cual tiene las coordenadas, fecha y hora. Hay que desmenuzar dicha trama, y puede que no siempre obtenga todos los datos (por ejemplo, si el dispositivo está en un lugar cerrado como una casa), por lo que el programa se asegura de no avanzar hasta tener todos los datos necesarios (coordenadas, fecha y hora). En exterior no hay problema en obtenerlos rápidamente, pero en interiores sí puede haberlo.

Una vez se tienen todos los datos, se pone en fecha y hora el reloj del dispositivo, establece comunicación con la base de datos, a partir de sus coordenadas se obtiene su nodeId (salvo que ya lo tuviese, en el fichero *"conf.txt"*) y las WiFi cercanas.

4.3.2 Hora de despertar y dormir de los nodos

Como indicamos al principio, el objetivo fundamental de este trabajo es medir la radiación solar, motivo por el cual, durante la noche, el nodo no debe obtener muestras ni enviar ningún tipo de información. Durante la noche, el nodo entrará en un modo de *"sueño profundo"* en el que lo único que funcionará es un contador para saber cuándo despertar. De este modo, el consumo de energía será mínimo, lo cual es indispensable, ya que en el tiempo que no haya sol, el dispositivo funcionará con la energía acumulada en la batería.

Para saber cuándo tiene que dormir y cuándo despertar, en cada arranque o despertar, el nodo consultará, en la base de datos, la tabla referida a las horas de dormir y despertar. En esta tabla sólo hay 3 campos, la fecha, hora de dormir y hora de despertar al día siguiente. Una vez tiene estos datos, el nodo empieza su funcionamiento normal y cada minuto comprobará si es el momento de dormir o no. Una vez llega este momento, con la hora de despertar, calcula el tiempo, en segundos, que debe estar dormido, y entrará en este estado. Una vez trascurrido este tiempo, el nodo despertará, empezará sus tareas de obtener la localización, por si ésta ha cambiado, las redes WiFi disponibles, y comenzará a medir y enviar datos.

Para mayor robustez del dispositivo, esta comprobación se realiza previa a empezar a medir, por lo que en el caso de que el dispositivo se encienda en horario de estar dormitando, entrará automáticamente en este modo. Los motivos de esta comprobación son:

- Reseteo accidental del dispositivo durante el tiempo de estar en sueño profundo.
- Colocación/puesta en marcha del dispositivo en horario de estar en sueño profundo.

4.3.3 Mediciones y envío de las tramas

Como ya se ha mencionado anteriormente, el objetivo de este trabajo es estudiar la radiación, para ello se ha desarrollado el código y configurado el dispositivo con los sensores dedicados a tal fin. Es muy importante obtener datos cada poco tiempo, por lo que el dispositivo ha sido programado para tomar muestras cada segundo, y enviar, de forma predeterminada, cada 5 segundos la media de estas 5 mediciones. Esta frecuencia de envío es modificable mediante una consulta al servidor, ya que esta acción puede repetirse hasta 12000 veces en un día, lo que,

multiplicado por la cantidad de nodos, puede resultar un volumen de datos muy grande a analizar y puede que esta frecuencia no añada información relevante.

Para que todos los nodos envíen en el mismo instante de tiempo, o lo más aproximado posible, ya se ha mencionado que sincronizan su reloj con la hora obtenida mediante el GPS, y en cada medición el dispositivo comprueba si ha de enviar los datos o no. Para saberlo, realiza un módulo 5 a los segundos del reloj. Si el resultado es 0, enviará los datos, y de éste modo está asegurado que todos los nodos enviarán los datos en los segundos terminados en 0 y 5. Es la mejor manera de que los nodos estén sincronizados según la hora obtenida por el GPS, y así no hay duda en saber en qué momento fueron enviadas las tramas, pues para todos los nodos va a ser en el mismo momento, por lo que comparar las tramas del mismo instante va a ser muy sencillo.

Es verdad que este sistema provoca que la primera trama del nodo puede ser muy poco precisa, ya que es posible, por ejemplo, que un nodo despierte en el segundo 57 y en 3 segundos envía una trama que no contiene información fiable, pero es preferible perder la primera trama de cada nodo y hacer este método de *"sincronización"* que a que cada nodo envíe en instantes distintos.

Cada trama de datos contiene en primer lugar el nodeId, seguido de la fecha y la hora expresada en HH:mm:ss, y después todos los datos obtenidos de los sensores, temperatura exterior, temperatura dentro del nodo y humedad relativa, ocupando un total de unos 60B por trama. De este modo, se puede hacer para cada nodo, un análisis de los datos de todo el día, y para todos los nodos, un análisis en el mismo instante de tiempo en todos ellos. Es importante incluir la marca de tiempo en la trama, pues, aunque el servidor MQTT tenga la hora de recepción de los mensajes, no es precisa, ya que entre el tiempo que tarda el nodo en enviar los datos y el retraso que puede haber durante el tiempo den envío desde el nodo hasta el servidor (puede ser de 1-2 segundos a veces), haría que los datos no mostrasen exactamente el instante de tiempo al que pertenecen y sería muy difícil, casi imposible, poder comparar las tramas del mismo instante de distintos nodos.

El servidor MQTT creado en Adafruit permite ver los datos en tiempo real según estén llegando, por cada uno de los subscriptores (los dispositivos en este caso), además de descargar todos los datos en formato *".csv"*, lo cual es de vital importancia para su posterior análisis.

5 - Plan de despliegue

En este capítulo se afrontan algunos de los desafíos más importantes de este proyecto: desplegar una red de cientos/mieles de nodos con sensores que estarán continuamente enviado datos, la instalación del software en cada uno de ellos, su administración, ser capaces de detectar errores en ellos y dónde posicionarlos, son los puntos que se explicarán en este apartado.

5.1 Problemática

Desplegar una red de nodos/sensores conlleva una serie de problemas adheridos a su naturaleza.

- Configuración: No es viable configurar cada nodo uno a uno. Cada nodo tendrá datos únicos, como es su nodeId. Por ello, el servidor tiene una base de datos en la que almacena datos únicos de cada dispositivo (la MAC) junto con sus coordenadas y así poder asignar un nodeId a cada nodo de forma única para poder ser reconocido dentro de la red de sensores.
- Mantenimiento: Los nodos pueden dar problemas. Para cada problema habrá que realizar una serie da tareas de mantenimiento:
    - Se detecta que los datos enviados por un nodo no son correctos. Al tener el nodo geolocalizado, no hay problema en saber cuál es. Las acciones a realizar podrían ir desde un reseteo, hasta reinstalar el programa en él, pasando por una recolocación de éste, si estaba mal orientado.
    - Las coordenadas han cambiado, es decir, su posición ha cambiado. Esto puede ser por varios motivos:
        - Intencionadamente se ha cambiado: Puede que al estudiar un punto concreto se haya llegado a la conclusión de que no es de interés, y se quiera reaprovechar el nodo en otro lugar.
        - Por accidente: debido al pequeño tamaño y peso del nodo, puede moverse por distintos motivos, como fuertes vientos, incluso que algún animal sea capaz de moverlo.
        - El nodo no se inicia. Puede ser resultado de una mala alimentación solar, por lo que habría que reorientarlo hasta subsanar el problema.

- Posicionamiento: Es importante saber dónde se van a desplegar los nodos. Hay que tener en cuenta una serie de factores para no llenar la red de nodos que, a priori, se puede saber que no van a tener buen resultado. Además, los nodos no deben estar tan cercanos que puedan producir confusión durante su mantenimiento:
    - No poner nodos donde se sabe a priori que no va a haber mucha radiación. No tiene sentido colocar un nodo en un lugar donde ya se saber que casi siempre hay sombra debido, por ejemplo, a un alto edificio. Lo lógico sería colocarlo en la azotea de este edificio.
    - Orientarlos de forma que aprovechen al máximo las horas de sol.
    - Además, la exactitud del GPS depende de con cuántos satélites GPS sea capaz de establecer contacto el dispositivo. En las pruebas realizadas tenía un rango de error de hasta unos 15 metros, por lo que, si 2 nodos se colocan a 30 metros o menos, podrían aparecer (aunque sea muy improbable) con las mismas coordenadas.
- Apagado: estos dispositivos no pueden apagarse como tal, el único botón que tiene una LoPy es un botón de reset, por lo que el modo de apagarlo es simplemente desconectando la fuente de alimentación.

5.2 Servidor de administración de nodos

La administración de los nodos se realiza mediante un servidor. Este servidor ejecuta XAMPP, el cual lleva un Apache, una base de datos en MySQL y scripts en PHP, cada uno para una función distinta, que se corresponden con las tablas de la base de datos. Cada nodo, al iniciarse, ya sea la primera vez o no, lo primero que hará al tener conexión es conectar con el servidor. El funcionamiento de cada uno de sus elementos es el siguiente:

5.2.1.1 Base de datos

Se compone de 3 tablas:

- Nodos
- Hora de despertar y dormir
- Áreas y WiFi de cada una

La primera tabla sirve para administrar los nodos, su nodeId, coordenadas y MAC. La segunda se encarga de almacenar a qué hora han de dormirse los nodos y a qué hora despertarse, y la última tiene un listado de las redes WiFi por áreas.

5.2.1.2 Control de nodos

Al establecer cada nodo la comunicación con el servidor, acción que ocurre al despertar el nodo, enviará sus coordenadas y su MAC. Con ellas, el script consultará la base de datos y dará una respuesta al nodo. Este script se encarga de controlar la asignación de Id del nodo, según sus coordenadas y su MAC. Hay 3 casos a tener en cuenta:

- Nodo nuevo en posición nueva: las coordenadas y la MAC no existen en la base de datos, por lo que se le asigna un nuevo ID
- Las coordenadas existen, pero no así la MAC. Puede deberse a que el nodo ha sido reemplazado, ya sea por nosotros o que ha sido suplantado. En ese caso, se avisará al administrador para que tome las medidas necesarias.
- La MAC existe, pero no las coordenadas. Puede deberse a que un nodo haya sido cambiado de sitio o robado, por lo que debe ser el administrador el que tome las medidas necesarias.

5.2.1.3 Asignación de red WiFi

Dependiendo de la zona en la que esté, necesitará una WiFi u otra. En la base de datos hay una tabla para administrar las redes WiFi, donde se encuentran sus coordenadas, el SSID y la contraseña. Las coordenadas de una red, al tener que cubrir un área, están expresadas como norte, sur, este y oeste, donde cada una se refiere al máximo al que llega. En la Ilustración 15 se puede observar cómo se delimitan las WiFi en esta tabla. Aunque las WiFi no tienen por qué tener un área de acción cuadrada, se ha elegido esta manera al ser aproximada y más fácil de representar en una base de datos, así como su facilidad para que el nodo realice las consultas necesarias para la obtención de las redes y facilidad de administración.

Ya que el nodo se moverá en un momento dado, no sólo obtiene las WiFi a su alcance, sino que obtiene algunas que estén cerca también, ya que si el nodo se mueve mucho desde el lugar donde ha sido arrancado por primera vez, es posible que las redes WiFi asignadas en primer lugar queden fuera de su alcance.

Para poder conectar a la WiFi, el nodo irá probando cada una de su listado *"WiFis.txt"* hasta que se conecte con alguna. Para asegurar la conectividad del nodo, la base de datos tiene que estar actualizada y ser correcta.

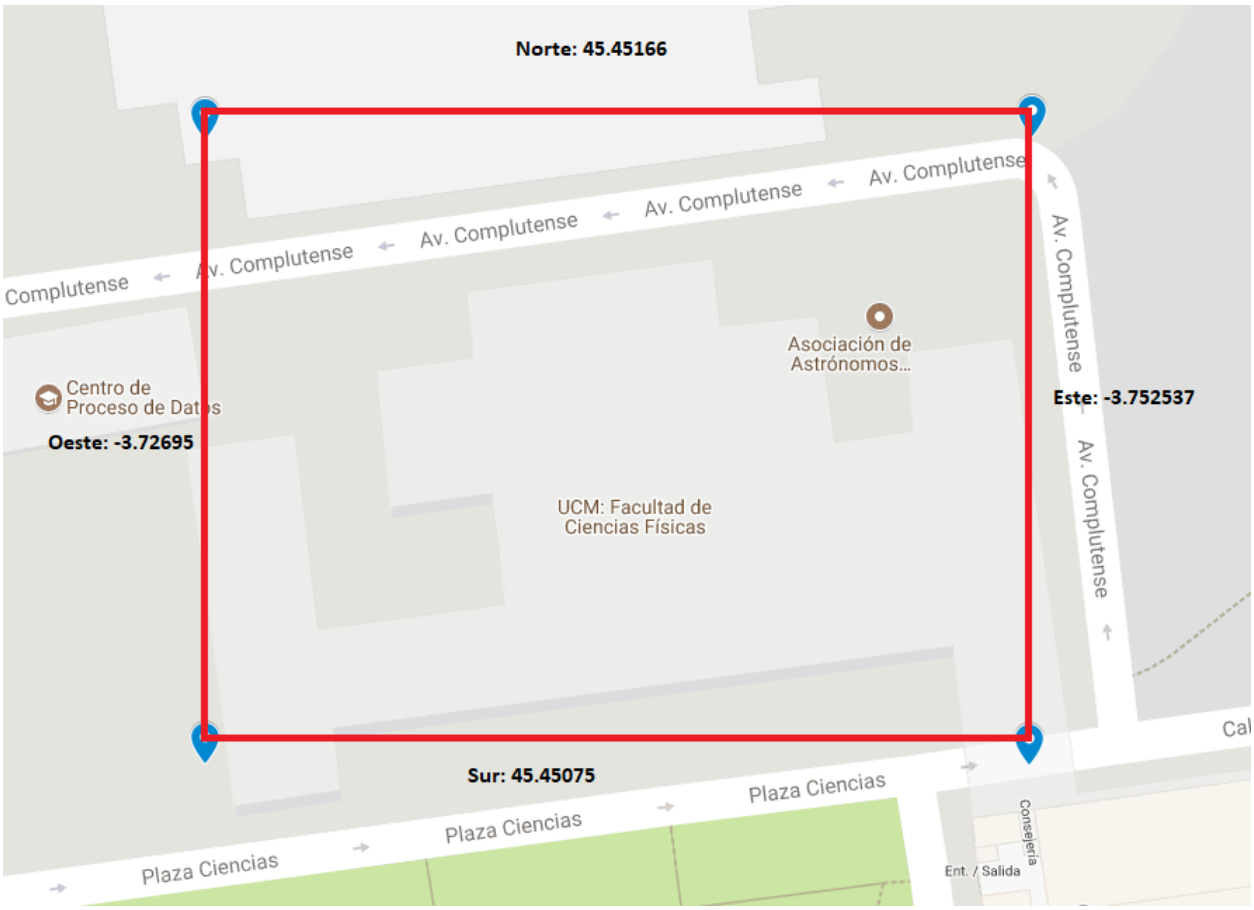

Ilustración 15 - Área de acción red WiFi

5.3 Instalación del software

Como ya se ha mencionado, los nodos que utilizaremos en el proyecto estarán conectados por WiFi o LoRa para el envío de las tramas con los datos recogidos de los sensores para su posterior análisis. Dado que se van a desplegar muchos nodos, es imprescindible que la instalación del software en ellos sea lo más rápida posible y que la intervención humana sea mínima. En un primer momento, la idea consistía en tener copias del programa en muchas tarjetas MicroSD, insertarlas en cada uno de los nodos y que estos, al iniciarse, copiasen el programa en su almacenamiento interno y se pusieran a funcionar. Lamentablemente, las LoPy no ofrecen esta opción. La única opción parecida es creando un script que hay que poner en cada

LoPy, por lo que la intervención humana es la misma, y sería un malgasto de dinero en las tarjetas MicroSD.

La segunda opción consiste en instalar el software en cada LoPy como se ha hecho durante el desarrollo, pero con una modificación. Como ya se ha explicado, el programa usado para el desarrollo del software y su instalación en la LoPy es el Atom con el plugin Pymakr. Hay un fichero de configuración llamado *"pymakr.conf"*, como se observa en la Ilustración 16, en el que se puede indicar qué archivos subir, carpeta, extensión… Así como el puerto al que está conectado la LoPy. De este modo, con una gran cantidad de puertos USB se podría realizar este paso de forma muy rápida, conectando muchas LoPy en un ordenador, o varios. Además, dado que los nodos pueden distribuirse en muchos lugares y en un primer lugar deben tener una conexión WiFi con un SSID y una contraseña que no conocen, de esta manera, se puede cambiar rápidamente para cada zona.

Ilustración 16 - Fichero Pymakr.conf

5.4 Funcionamiento habitual

En cada arranque del nodo, debe establecer comunicación GPS para obtener la hora y las coordenadas. Esto es importante porque al despertar, cada nodo tiene como fecha inicial época, y el nodo debe obtener la fecha y la hora del GPS, ya que es un modo seguro de obtener la hora correcta sin sufrir retrasos que puedan desencadenar una desincronización de los nodos. Hay que diferenciar si es la primera vez que se inician los nodos o no.

El primer inicio del nodo se caracteriza por no tener un fichero de configuración, que tendrá que crearlo. Para ello, se conecta a una red WiFi predeterminada en el código y con sus coordenadas obtendrá el Id del nodo (nodeId). Una vez tiene este dato, lo almacenará en un fichero llamado *"conf.txt"*, y cada vez que el nodo se inicie, comprobará si tiene este fichero o no.

El resto de inicios se caracterizan, por el contrario, por el hecho de que sí tiene el fichero de configuración, del cual obtienen su nodeId. Aun así, al iniciarse (o al despertar) comprobará

sus coordenadas, las contrastará con la base de datos y pondrá en hora su reloj, así como el calendario. Además, en cada inicio del nodo se comprobarán las redes WiFi disponibles, ya que el nodo ha podido ser cambiado de lugar o las redes han podido variar. Tras esto, se conectará a alguna de las redes disponibles y empezará a medir y a enviar las tramas con la información obtenida. Estas redes estarán almacenadas en un fichero llamado *"WiFis.txt"*.

5.5 Gateway

Dado el gran número de nodos que se van a tener conectados a un mismo Gateway, este debe ser bastante potente. El ancho de banda agregado que supone esta red será de:

- Tamaño de las tramas: 60B
- Tiempo de trasmisión: cada 5 segundos (de forma predefinida). Se va a tomar este tiempo para los cálculos.
- (60B*8)/5 = 96bps por cada nodo.
- El ancho de banda agregado aumentará con mayor número de nodos y menor tiempo entre trasmisiones.
- El ancho de banda agregado disminuirá con menor número de nodos y menor tiempo entre trasmisiones.

5.6 Actualización remota del software

Al igual que pasa con la instalación del software en el dispositivo, momento en el que la acción humana debe ser mínima, a la hora de la actualización del software tenemos el mismo problema, agregando el hecho de tener que, desplazarse, recoger los nodos, actualizarlos y volver a desplegarlos donde estaban. Además del tiempo que esto supone, se produce una importante pérdida de datos durante ese tiempo. Otra estrategia podría ser, para algún nodo que pueda dar problemas, desplazarse hasta él exclusivamente con un ordenador portátil a instalar una actualización del software, como medida puntual en caso extremo.

El modo de actualización de software debe hacerse de modo remoto. Atom permite la comunicación con nodos mediante su IP, lo cual facilita esta tarea al no tener que realizarse un desplazamiento a cada nodo, con todos los inconvenientes que conllevaría.

Esta actualización remota se puede contemplar de 2 maneras distintas:

- Con Atom poder actualizar en el mismo momento todos los nodos, asignando como dirección objetivo no la IP de un nodo en concreto, sino mediante un broadcast (envío a todos los nodos) de la red en la que se encuentren los nodos.
- En el servidor de administración alojar la nueva versión del software y, mediante una comprobación diaria, comprobar si hay una actualización del software y desarrollar un script que permita ésta actualización.

5.7 Identificación de los nodos

Como se ha explicado previamente, hay distintas causas por las que un nodo puede estar dando problemas, por lo que es de gran importancia identificarlos cuanto antes. Gracias al servidor se tiene un control de los nodos, su nodeId y sus coordenadas, datos que utilizaremos, junto a las tramas enviadas al bróker. Además, este nodeId está almacenado en un fichero en el nodo llamado "*conf.txt"*, con el cual, accediendo al nodo, ya sea remotamente o físicamente, se puede leer dicho fichero para saber si es el nodo.

Una vez se ha detectado el nodo que falla, se puede proceder a realizar una conexión con el IDE Atom a ese nodo en concreto e intentar realizar tareas para identificar el fallo y si es posible solventarlo, o incluso un reinicio. En caso de continuar el fallo, habría que hacer un desplazamiento físico hasta el lugar.

6 - Pruebas

La verificación del correcto funcionamiento del dispositivo es una parte fundamental del desarrollo. Los principales aspectos a probar del dispositivo han sido la conectividad WiFi, el servidor MQTT, el GPS, el correcto muestreo del dispositivo y las pruebas relacionadas con el servidor de administración, además de una serie de casos de uso para facilitar la resolución de algunos posibles problemas.

6.1 Casos de uso

En este apartado se describen los casos de uso de las principales tareas de cada nodo. Su objetivo principal es establecer las tareas, qué elementos realizan acciones, precondiciones y postcondiciones, y de este modo facilitar el mantenimiento en caso de fallos en algún nodo.

Caso de uso 1	**Comprobar si es primer inicio**
Descripción	- Comprueba si existe el fichero *"conf.txt"*, que es el modo de saber si es su primer inicio o no
Actores	- Nodo
Precondiciones	- Existencia o no del fichero *"conf.txt"*
Postcondiciones	- Permite saber si hay que realizar la tarea de darse de alta en la base de datos, así como la obtención del nodeId
Escenario principal	1. El nodo se enciende 2. Comprueba si tiene el fichero *"conf.txt"*

Caso de uso 2	**Obtención del nodeId para nuevos nodos**
Descripción	- Al no existir el fichero *"conf.txt"*, el nodo debe conectarse a la red predefinida y, con sus coordenadas y MAC, obtener el nodeId y crear el fichero *"conf.txt"*.
Actores	- Nodo - Servidor
Precondiciones	- Tener conexión a la red
Postcondiciones	- Permite obtener el nodeId

Escenario principal	1. Se conecta a la red 2. Obtiene las coordenadas 3. Se comunica con el servidor para registrar sus coordenadas, MAC y obtener su nodeId

Caso de uso 3	**Obtención de la hora de dormir y despertar**
Descripción	- Conexión con el servidor para saber a qué hora debe dormir el nodo y a cuál despertar
Actores	- Nodo - Servidor
Precondiciones	- Estar conectado a la red
Postcondiciones	- El nodo sabrá a qué hora dormir y a qué hora despertar
Escenario principal	1. El nodo dormirá y despertará a la hora que marque el servidor

Caso de uso 4	**Mediciones de los sensores**
Descripción	- Los sensores empiezan a medir la temperatura, humedad y radiación
Actores	- Nodo
Precondiciones	- Estar conectado a la red
Postcondiciones	- Se obtienen los datos para su posterior envío
Escenario principal	1. El nodo obtiene los datos de los sensores y calcula su media

Caso de uso 5	**Envío al servidor MQTT**
Descripción	- Tras obtener los datos de los sensores y calcular su media, éstos serán enviados al servidor MQTT
Actores	- Nodo - Servidor MQTT
Precondiciones	- Estar conectado a la red - Estar conectado al servidor MQTT
Postcondiciones	- Los datos de la media de las mediciones de los nodos han sido

	enviados al servidor
Escenario principal	1. El nodo envía por WiFi o LoRa los datos obtenidos al servidor MQTT

Caso de uso 6	**Dormir**
Descripción	- Cada minuto, el nodo comprobará sui es la hora de dormir. En caso afirmativo, entra en modo de sueño profundo.
Actores	- Nodo
Precondiciones	- Tener asignadas las horas de dormir y despertar
Postcondiciones	- Los datos de la media de las mediciones de los nodos han sido enviados al servidor
Escenario principal	2. El nodo envía por WiFi o LoRa los datos obtenidos al servidor MQTT

6.2 Conectividad WiFi

La primera prueba ha sido conectar el dispositivo a una red local conocida. El nodo va a buscar las redes de 2 maneras distintas:

1. Si no tiene el fichero con las WiFi llamado *"WiFis.txt"*, el nodo se conectará a una red predeterminada que ha sido puesta por el administrador.
2. Si tiene el fichero, irá probando una a una todas las redes disponibles hasta poder conectar con alguna.

Comprobar si esto ha ido bien es bastante sencillo. Con el nodo conectado al ordenador mediante el cable USB y la placa de expansión y el programa Atom en ejecución se pueden observar las trazas del código. Cuando el nodo se conecta a la red WiFi, muestra un mensaje por pantalla y su IP, como se puede observar en la Ilustración 17.

Ilustración 17 - Traza conexión WiFi

Para comprobar si realiza correctamente la creación del fichero *"WiFis.txt"* lo que se ha hecho ha sido insertar 2 redes WiFi en la base de datos con unas coordenadas parecidas a las del dispositivo, de modo que obtenga ambas. Al no disponer de 2 redes WiFi distintas con su contraseña, con un Smartphone se ha usado *"Tethering"*, es decir, usar el Smartphone como Gateway con WiFi. Se ha configurado en él otra red WiFi. Tras obtener las redes, se observa mediante las trazas en el Atom que el dispositivo se desconecta de la red en la que está e intenta conectarse a la red creada con el Smartphone con éxito. Otra prueba ha sido desconectar el *"tethering"* del Smartphone y comprobar si al no encontrar esa red se conectaba a otra de la lista, otra vez con éxito.

6.3 Servidor MQTT

Una vez hay conectividad con la red, el siguiente paso ha sido configurar y comprobar el correcto funcionamiento del servidor MQTT. Una vez configurado, hubo que plasmar en el código las funcionalidades necesarias para su comunicación con el servidor. Los datos necesarios para ello son: el nombre del servidor al que conectarse, el *topic* y el mensaje, en este caso, la trama con los datos de los sensores, nodeId y marca de tiempo.

Comprobar su funcionamiento es bastante sencillo, como se muestra anteriormente en la Ilustración 12, pues el propio servidor de Adafruit nos da una opción de monitorear los mensajes recibidos. Si en vez de este servidor se hubiese hecho con otro, se podría comprobar mediante un subscriptor subscrito al *topic* correspondiente y comprobar si llegan los mensajes correctamente.

6.4 GPS

El nodo consulta el GPS al iniciar, y una vez, mediante las trazas mostradas por el programa Atom, se puede comprobar si se han obtenido correctamente estas coordenadas o no. Además, para saber si éstas estaban bien, se han contrastado con Google Maps, con un margen de error de pocos metros (máximo de 15).

Otro modo de comprobar si el GPS funciona correctamente es comprobar si en la base de datos del servidor de administración ha habido alguna modificación, ya sea que se inserta un nuevo nodo, o que uno que ya existía ha cambiado sus coordenadas.

Otra de las pruebas referidas al GPS ha sido la realizada para comprobar el desfase en segundos que puede acumular el dispositivo a lo largo de un día de uso. Habiendo empezado a

las 9:30 de la mañana y terminado a las 20:55, su retraso acumulado ha sido de entre 1.9 y 2.1 segundos. Este retraso no supone un problema a lo largo de un día y que no es acumulable a lo largo de los días.

6.5 Muestreo

El objetivo principal del dispositivo es tomar muestras con los sensores y enviarlas al servidor MQTT. Las pruebas realizadas han sido:

1. Comprobar mediante mensajes trazas en Atom si los sensores medían correctamente. A continuación, si hacían la media correctamente.
2. Comprobar si la trama estaba bien formada y su correcto envío al servidor MQTT. Para comprobar esto, en la administración del servidor MQTT se puede observar qué nodos están enviando y qué envían.

El resultado obtenido son unas tramas en almacenadas en el servidor MQTT con el nodeId, fecha y hora, y a continuación los datos obtenidos de las mediciones separados por *";"*. Todos los datos obtenidos de las mediciones de los sensores están dentro de lo esperado.

6.6 Servidor de administración

Las pruebas realizadas se van a describir por cada una de las tablas.

Para la tabla de *"nodes"*, tenemos 2 casuísticas:

1. Se registra un nodo nuevo. Esto sucede cuando un nodo conecta con el servidor y no están registradas ni sus coordenadas ni su MAC.
    a. Para comprobar esto, se observa que en la base de datos se añada un nuevo registro.
2. Un nodo ya está registrado, pero se han modificado sus coordenadas o su MAC. Esto puede ser por 2 motivos:
    a. El nodo ha sido cambiado de sitio o sustraído (cambian las coordenadas).
    b. El nodo ha sido cambiado por desperfectos (cambia la MAC).

La tabla *"times"*, usada para indicar la hora de entrar en sueño profundo y la de despertar, se consulta cuando el nodo se inicia, estableciendo así estas horas. Para comprobar si ha funcionado correctamente, una vez más, se puede hacer que el Atom muestre por pantalla las horas de sueño profundo y de despertar.

Para comprobar que la funcionalidad es correcta, primero se hicieron pruebas con pequeños intervalos de sueño (1,3,5,20 minutos), hasta que se hicieron pruebas de funcionamiento completas. Observando el dashboard del servidor MQTT, se comprobaba que el nodo enviaba la última trama justo en el instante antes de la hora de entrar en sueño profundo. Por la mañana, se ha comprobado que al poco de la hora de despertar, ya está enviando tramas al servidor MQTT. No es automático ya que el despertar es aproximadamente como un inicio normal del dispositivo, con todas sus comprobaciones de ficheros, conexión a la red, obtención de coordenadas y contrastar datos con el servidor.

Por último, se ha probado la funcionalidad referida a la obtención de redes WiFi. Para su correcta comprobación, se han realizado trazas en se muestran por pantalla al recibirlas tras la consulta.

6.7 Otras pruebas

6.7.1 Redes

Como ya se ha mencionado, fue imposible conectar el dispositivo a la red Eduroam por falta del certificado para la LoPy. A su vez, se descubrió que para insertar el certificado en el dispositivo no se puede hacer como el resto de ficheros, usando el programa Atom, sino que hay que hacerlo desde una consola de comandos. Para ello, basta con el comando:

```
sudo ampy --port /dev/ttyUSB0/cert put ca.crt
```

6.7.2 Desfase temporal

Además, para comprobar la fiabilidad de los envíos de las tramas de datos al servidor y que la hora de dormir y despertar iba a ser fiable, se comprobó el retraso que puede haber en el dispositivo. Para ello, justo antes de irse a dormir, el dispositivo volvía a conectar con el GPS; en ese momento obtiene la hora, y al obtener la hora nueva, calcula la diferencia de tiempo que hay. Para ello, se compara la hora actual con la recibida del GPS, por lo que el retraso es mínimo. Se ha visto que el retraso puede ser de unos 2 segundos al día, elemento que no supone un gran problema ya que, al iniciar cada día el nodo, este vuelve a obtener la hora, por lo que no hay que temer por una acumulación de retrasos que al final del mes pueda ocasionar alguna incongruencia en los datos recibidos.

6.8 Pruebas de consumo de energía

Dado que este dispositivo tiene que ser autónomo, un punto vital es su consumo de energía. Para ello, se ha conectado el dispositivo a una máquina para medir la corriente, como se observa en la Ilustración 18. El cable negro se ha conectado a la toma de tierra del dispositivo, y el rojo, al de alimentación. Las pruebas se han hecho a 5.5 voltios y un límite de 2 amperios. La fórmula para calcular la potencia es P = V x I, mientras que la energía eléctrica se calcula como el producto de la potencia y tiempo: E= PxT.

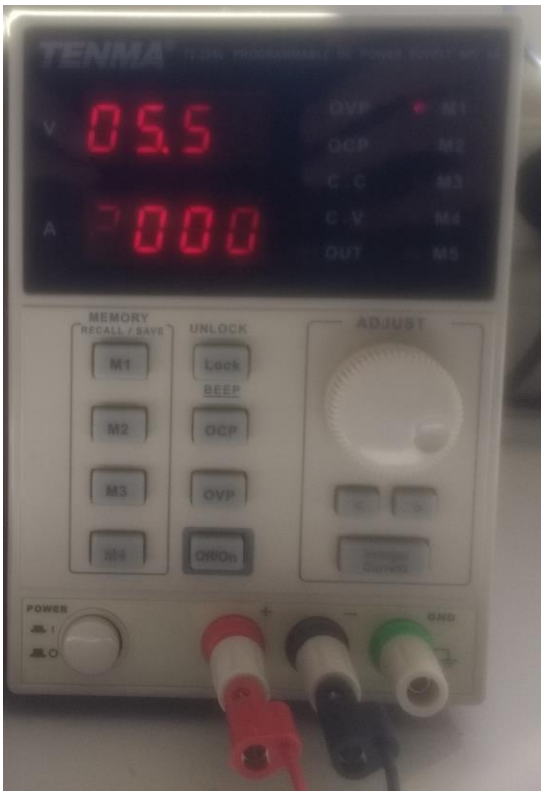

Ilustración 18 – Fuente de alimentación

Las pruebas realizadas para el consumo de energía coinciden con las funciones principales del dispositivo. Se ha diferenciado entre *"modo espera"* y *"sueño profundo"*, que presentan consumos muy diferentes de energía, ya que el primero es una espera que realiza el dispositivo, y el segundo es un modo de sueño más profundo en el que sólo hace uso de su reloj, y al terminar este sueño, es como un inicio del dispositivo. Además, se han hecho las medidas cargando la batería y sin ella, ya que en caso de estar conectada y necesitar cargarla, la potencia necesaria es mayor. Se ha realizado un estudio del tiempo necesario para cada una de las acciones, ya que las diferentes tareas que realiza el nodo no tienen por qué gastar la misma

energía. Las 3 primeras pruebas sólo se realizan una vez al día, y el total de las realizadas han sido:

1. Iniciar el nodo: tarda aproximadamente unos 10 segundos y se produce una vez al día.
2. Conectar red WiFi: es algo que solo se hará al iniciar. Tarda unos 15 segundos
3. Obtención de coordenadas GPS, que es algo que sólo se hará al iniciar. Supone unos 20 segundos en un caso malo, 10 en casos favorables, por lo que la media resulta de 15 segundos.
4. Medir los datos de los sensores: Requiere de alrededor de175ms de media cada segundo.
5. Envío de la trama: se lleva a cabo cada 5 segundos (de forma predeterminada). El envío del mensaje son 5ms
6. Carga de la batería: Desde que los rayos del sol llegan al panel solar del dispositivo hasta que esta esté cargada
7. Modo espera: durante la medición de los sensores, unos 825ms (de cada segundo) está en este estado. Obtener los datos de los sensores y todo lo necesario para su envío oscila entre 170 y 180ms. Como enviar el mensaje son 5ms y no generar un retardo acumulativo, se ha contado como 174ms.
8. Sueño profundo: el tiempo que no está despierto está en este modo. Varía según el día.

Además, el dispositivo tiene 2 estados importantes:

- Despierto
- Sueño profundo

Durante el uso ya se ha especificado el tiempo que conlleva cada acción. Algunas solo se realizan una vez al día, mientras que medir y realizar el envío de datos se realiza constantemente hasta que llega la hora del sueño profundo. Dado que el tiempo que esté despierto o en sueño profundo varía de la época del año en la que se esté, en invierno hay menos horas de sol y en verano hay más, se va a hacer el cálculo del consumo de todo el día usando el día más largo del año y el más corto.

El día más corto del año, el 21 de diciembre, tiene 9 horas y 17 minutos de luz, en segundos son 33420, en cambio, el día más largo, el 21 de junio tiene 15 horas y 3 minutos, que son un total de 54180 segundos (Instituo Geográfico Nacional, 2018). Descontando los 40

segundos de media que necesita el dispositivo al día para ponerse en marcha (las tareas de inicio, conexión y geolocalización), se obtiene que los segundos en que estará midiendo y enviando son 33380 y 54140, para invierno y verano. De cada uno de esos segundos, como se ha mencionado, 825ms los pasará en modo espera, 174 ms midiendo y el envío por MQTT se ha medido en 5ms, es decir, 0.005s. Para saber cuántas veces se envía, basta con dividir los segundos entre 5.

En total, el dispositivo pasa:

- Segundos en espera: 33380x0.825 = 27538.5 segundos en invierno, 54140x0.825= 44665.5 segundos en verano.
- Segundos midiendo = 33380x0.174 = 5808.12 segundos en invierno, 54140x0.174= 9420.36 segundos en verano.
- Segundos enviando: 33.38 segundos en invierno, 54.14 segundos en verano.
- Segundos en sueño profundo: 52980 segundos en invierno, 32220 segundos en verano.

Tabla 2 . Tabla de potencias y consumos

Variable / Acción	Corriente (A)	Potencia (W)	Tiempo diario(s)		Energía consumida (J)	
			Invierno	Verano	Invierno	Verano
Inicio	0.16	0.88	10		8.8	
Conexión WiFi	0.21	1.155	15		17.325	
Coordenadas	0.16	0.88	15		13.2	
Sensores	0.55	3.025	5808.12	9420.36	17.569,563	28.496,589
Envío	0.62	3.41	33.38	54.14	113.8258	184.6174
Modo espera	0.16	0.88	27538.5	44665.5	24233.88	39305.64
Sueño profundo	0.08	0.44	52980	32220	23311.5	14176.8
Total gasto					65267.7938	82163.6464

Dado que el voltaje es constante (5.5V), las variables medidas han sido la corriente y el tiempo. Con ello, se obtiene la potencia y el consumo total al cabo de un día del dispositivo. Dado que el panel solar es capaz de proporcionar hasta 6V en el mejor de los casos y 2W, es más que suficiente para poder alimentar el dispositivo, así como cargar la batería. Ya que esto depende de cómo el sol de en el panel solar, la nubosidad, la inclinación de los rayos… vamos a suponer todo esto en el mejor de los casos.

La batería, de 800mAh, en caso de estar completamente descargada, tardaría en cargar aproximadamente 1:47 minutos. Durante las pruebas realizadas, ha sido capaz de aguantar toda la noche sin problemas. Estas pruebas han sido realizadas en verano, donde las noches son más cortas.

El total de consumo de energía obtenido ha sido de 65267.7938J al día en invierno y 82163.6464J al día en verano. Para calcular el consumo diario (sin contar el modo sueño profundo) se han tenido en cuenta el consumo total de los sensores, envío y modo espera.

Quizás uno de los peores aspectos es que, el nodo, para estar alimentado completamente, necesita que el sol de muy bien en el panel solar. No siempre se va a conseguir esto, por lo que puede que en algunos momentos el panel no sea capaz de alimentar bien los sensores y el nodo haga uso de la batería para su alimentación, mermando así su uso durante el día e incluso el estado de sueño profundo por la noche.

7 - Conclusiones

El desarrollo del código para el dispositivo ha supuesto un reto en el sentido de tratarse de un lenguaje y una plataforma nuevos sumando la necesidad de realizar un código óptimo para no sobrecargar al dispositivo y que su consumo de energía sea el menor posible. La duración de algunas pruebas, como la de obtener el desfase del reloj, o muchas que dependían de obtener las coordenadas, ya que el desarrollo mayoritario del proyecto ha sido realizado en espacios cerrados, han requerido mucho tiempo, ya que al ser más difícil la obtención de las coordenadas, han requerido de bastante tiempo, o salir al aire libre durante estas pruebas.

A pesar de estos retos, se han conseguido casi todos los objetivos propuestos al inicio del trabajo. El nodo es capaz de ser autónomo; tiene un servidor que funciona correctamente para la administración de las tareas básicas que cada nodo necesita, como son asignar un nodeId a cada uno y saber dónde se encuentran, informar a cada nodo de qué redes tiene cerca; de cuando se va a ir a dormir y cuando se va a despertar. En cuanto a su funcionamiento durante días, desde su primer inicio; es capaz de dormir a la hora asignada y despertarse cuando tiene que hacerlo; de no funcionar en horas que no debe si ha sufrido algún reseteo o situación inesperada; capaz de simular una *"sincronización"* con el resto de nodos (simular porque hacerla con tantos nodos es prácticamente imposible).

8 - Conclusions

The development of the code for the device has been a challenge in the sense of being a new language and platform adding the need to make an optimal code so as not to overload the device and that its energy consumption is the least possible. The duration of some tests, such as obtaining the phase shift of the clock, or many that depended on obtaining the coordinates, since the majority development of the project has been carried out in closed spaces, have required a lot of time, since being more difficult the Obtaining the coordinates, have required quite a lot of time, or go outside during these tests.

Despite these challenges, almost all the objectives proposed at the beginning of the work have been achieved. The node is capable of being autonomous; it has a server that works correctly for the administration of the basic tasks that each node needs, such as assigning a nodeId to each one and knowing where they are, informing each node of which networks it has nearby; from when you are going to go to sleep and when you are going to wake up. As for its operation for days, from its first start; is able to sleep at the assigned time and wake up when he has to; of not working in hours that you should not if you have suffered a reset or unexpected situation; able to simulate a "synchronization" with the rest of the nodes (simulate why doing it with so many nodes is practically impossible).

9 - Vías futuras

9.1 Conexion LoRa

Como se ha descrito en el proyecto, las bondades de las redes LoRa son muy ventajosas para el IoT, y sus problemas no son, a priori, ningún impedimento para las necesidades de este proyecto. No obstante, aunque la facultad de Física dispone de un router LoRa, no se ha podido probar por falta de tiempo. En la documentación se encuentra cómo realizar esta comunicación (Pycom, 2018).

Una posibilidad sería crear, como se ha hecho con las WiFi, un fichero de texto con las redes LoRa cercanas y que éstas estén registradas en una base de datos para así saber a cuáles conectarse.

9.2 Servidor MQTT

De momento está creado en Adafruit, pero se podría crear un servidor usando como bróker Mosquitto (Mosquitto, 2017). Adafruit da muchas ventajas para pocos nodos y es gratuito, sin embargo, en el momento en que el proyecto empeice a crecer, será recomendable añadir esta mejora.

9.3 Copiar tramas no enviadas a tarjeta MicroSD

El dispositivo sólo envía tramas si está conectado a internet. No contempla qué pasaría si no está conectado, situación que podría pasar si en un momento dado falla la red. Una mejora para el futuro es que almacene estas tramas en una tarjeta MicroSD, y cuando se tenga conectividad con la red, las envíe. Como todas tienen su marca de tiempo dentro de la trama, no hay problema en el momento en el que lleguen, ya que serán tratadas con la información aquí contenida.

10 - Bibliografía

Adafruit. (s.f.). *Adafruit*. Obtenido de Adafruit: https://www.adafruit.com/product/3339

Adafruit. (s.f.). *Adafruit AM2302*. Obtenido de Adafruit products: https://www.adafruit.com/product/393

AEMA. (2017). *Agencia Europea de Medio Ambiente. Configuración del futuro de la energía en Europa: limpia, inteligente y renovable.* . Obtenido de https://www.eea.europa.eu/es/senales/senales-2017-configuracion-del-futuro/articulos.

AIE. (s.f.). *https://datos.bancomundial.org/indicador/EG.USE.ELEC.KH.PC*.

Beningo, J. (3 de 11 de 2017). *Digi-key*. Obtenido de https://www.digikey.es/es/articles/techzone/2017/sep/develop-real-time-mcu-based-applications-micropython

BOE. (21 de Febreo de 2014). (Boletín Oficial del Estado. 11 de marzo 2014. Real Decreto 102/2014, para la gestión responsable y segura del combustible nuclear gastado y los residuos radiactivos. España.

C, A. P. (s.f.). *Meridianos y Paralelos*. Obtenido de Pesos y medidas: https://sites.google.com/site/antoniopadillac/Home/pesosymedidas/meridianosyparalelos

Christensson, P. (28 de 08 de 2018). *GPS Definition*. Obtenido de TechTems: https://techterms.com

EAPI. (2017). *Informe sobre el índice de Rendimiento de la Arquitectura Energética Mundial, 2017. Foro Económico Mundial.* . Obtenido de (EAPI, 2017) (Informe sobre el índice de Rendimiento de la Arquitectura Energética Mun http://www3.weforum.org/docs/WEF_Energy_Architecture_Performance_Index_2017.pdf

EEA. (Mayo de 2005). *Propuesta para el desarrollo de la energía solar fotovoltaíca.* Obtenido de https://www.ecologistasenaccion.org/wp-content/uploads/adjuntos-spip/pdf/propuesta_fotovoltaica.pdf

García, J. (2007). Investigando el problema del uso de la energía. *Investigación en la escuela*, 29-45.

George, D. (2014-2018). *MicroPython*. Obtenido de MicroPython: https://micropython.org/

ingsoftware weebly. (s.f.). Obtenido de Ingeniería del Software: https://ingsoftware.weebly.com/modelo-evolutivo.html

Instituo Geográfico Nacional. (2018). *Información sobre las infraestructuras y trabajos de Astronomía del Instituto Geográfico Nacional.* Obtenido de https://astronomia.ign.es/rknowsys-theme/images/webAstro/paginas/documentos/pdf/verano2018.pdf

IXYS. (2018). Obtenido de IXYS: http://www.ixys.com/ProductPortfolio/ProductFamilyInfo.aspx?i=78

Karel Vanicek, T. F. (2000). *COST-713 "UVB Forecasting"*. Obtenido de http://www.temis.nl/uvradiation/info/Vanicek_et_al_COST-713_2000.pdf

Masmar. (5 de 5 de 2008). *Masmar*. Obtenido de http://www.masmar.net/esl/N%C3%A1utica/Seguridad-Naval-Naufragios/Sistema-de-Posicionamiento-Global.-%C2%BFQue-es-un-GPS/Triangulaci%C3%B3n-desde-los-sat%C3%A9lites

Ministerio de Medio Ambiente, y. M. (s.f.). *Aemet*. Obtenido de Aemet: http://www.aemet.es/documentos/es/eltiempo/observacion/radiacion/Radiacion_Solar.pdf

Mosquitto. (2017). *Mosquitto*. Obtenido de Mosquitto: https://mosquitto.org/

Organización Mundial de la Salud. (1994). *Programa internacional de seguridad química, criterios de salud ambiental. Radiación ultravioleta.* Ginebra: Organización Mundial de la Salud.

Ortega, J. C. (15 de 4 de 2013). *metodologiaevolutivo.blogspot*. Obtenido de http://metodologiaevolutivo.blogspot.com/

Pycom. (20 de 08 de 2018). *Pycom LoPy*. Obtenido de https://docs.pycom.io/tutorials/lora/lorawan-abp

Pycom. (s.f.). *Pycom about*. Obtenido de Pycom: https://pycom.io/about-2/

Pycom. (s.f.). *Pycom products*. Obtenido de Pycom: https://docs.pycom.io/chapter/products.html

Python. (3 de 12 de 2008). *Python*. Obtenido de Python: https://www.python.org/download/releases/3.0/

Rouse, M. (2012). *Whatls.com*. Obtenido de https://searchdatacenter.techtarget.com/es/definicion/Salto-de-frecuencia-de-espectro-ensanchado

S.L., D. (2017). *LoRaWAN*. Obtenido de LoRaWAN: http://lorawan.es/

Sigfox. (6 de 2012). *Sigfox*. Obtenido de Sigfox: https://www.sigfox.com/en