Universidad Católica Del Uruguay

Telemetría de luminarias en aeropuertos utilizando LoRaWAN

Memoria de grado

Ingeniería en Electrónica

Autores

Germán Baldo – Joaquín Correa – Fabian Rodríguez

Tutor: Dr. Ing. Matías Miguez

Montevideo, Uruguay - Noviembre 2020

Nuestro más sincero agradecimiento a nuestras familias y amigos por su apoyo incondicional. También a quienes formaron parte de la carrera, a profesores y compañeros, a la Universidad Católica del Uruguay y a nuestro tutor, Dr. Ing. Matías Miguez por su gran ayuda durante todo este proceso. Por último una mención especial a Ing. Guillermo Coduri y al D. Ind. Emilio Oteiza por sus aportes en diferentes áreas del proyecto.

Resumen

En los últimos veinte años, luego de la invención del internet, esta red ha pasado de ser una red global interconectando computadoras a interconectar cualquier tipo de dispositivo. Esta revolución tecnológica la conocemos como internet de las cosas o IoT.

En este documento de grado se presenta el desarrollo de un sistema de monitoreo a distancia, alimentado por batería mediante la utilización de LoRa y LoRaWAN. LoRa es una tecnología emergente de radiofrecuencia que permite la transmisión de información a un gran alcance y muy bajo consumo. La implementación de este sistema se orientó a una aplicación específica, pero con la particularidad de que puede ser utilizado para cualquier otro requerimiento, como por ejemplo telemetría en industria, geolocalización, entre otros.

La aplicación desarrollada permite determinar el correcto funcionamiento de las luces de los aeropuertos de manera remota. Para esto fue necesario diseñar una red LoRaWAN y un servidor que permita procesar los datos y visualizarlos. Esta red está compuesta por muchos “dispositivos inteligentes” o nodos colocados muy cerca de cada luz del aeropuerto, que envían el estado de las mismas a un gateway central que recepciona y concentra los datos.

Para el desarrollo de cada nodo o terminal se utilizó un chip moderno del mercado (RAK4260). Este microcontrolador tiene la ventaja de ser bajo consumo y ya tener incorporado el transceptor LoRa. Su tarea es medir el estado de la luz (mediante la lectura de un sensor óptico) y enviar los datos al servidor central mediante LoRa.

Abstract

In the last twenty years, after the invention of the internet, this network has gone from being a global network interconnecting computers to interconnecting any type of device. This technological revolution is known as the Internet of Things or IoT.

This document presents the development of a battery-powered monitoring system using LoRa and LoRaWAN. LoRa is an emerging RF technology that allows the transmission of information with long range and very low energy consumption. The implementation of this system is oriented to a specific application, but with the particularity that it can be used for any other requirement, such as telemetry in industry, geolocation, etc.

The application developed allows to monitor the correct operation of the airport lights remotely. For this, it was necessary to design a LoRaWAN network and a server that allows data to be processed and viewed. This network was built with "smart devices" or nodes placed very close to each light in the airport, which send their status to a central gateway that receives and concentrates the data.

For the development of each node or terminal was used a modern chip on the market (RAK4260). This microcontroller has the advantage of being low power consumption and has included the LoRa transceiver. Its task is to measure the state of light (by reading an optical sensor) and send the data to the central server using LoRa.

Índice

Agradecimientos **1**

Resumen **2**

Abstract **3**

Índice **4**

1. Introducción **9**

1.1 Problema a resolver 9

1.2 Estado del arte 10

1.3 Objetivo 10

1.4 Solución planteada 11

2. Requerimientos del proyecto **12**

2.1 Consumo Batería 12

2.2 Alcance de comunicación 12

2.3 Confiable 13

2.4 Precio 13

3. Tecnología inalámbrica **14**

3.1 Selección de tecnología 14

3.1.1 ¿Por qué LPWAN? 14

3.1.2 SigFox 15

3.1.3 LoRa 15

3.1.4 LTE-M 15

3.1.5 NB-IoT 16

3.1.6 ZigBee 16

3.1.7 Elección de LoRa 16

3.2 LoRa 17

3.3 LoRaWAN 17

3.3.1 Arquitectura de la red 18

3.3.2 Duración de batería 19

3.3.3 Capacidad de la red 20
3.3.4 Clases de nodos 20
3.3.5 Seguridad 21
3.3.6 Especificaciones regionales 21
3.4 The Things Network (TTN) 22
4. Elección del hardware 24
4.1 Microcontrolador - Transceptor 24
4.1.1 Transceptores familia SX12xx 24
4.1.2 Transceptores familia RFM95/96/97/98 25
4.1.3 Transceptores familia SAMR34 26
4.1.3 Análisis y selección 28
4.2 Gateway 29
4.2.1 RAK 2245 (Pi Hat) 29
4.2.2 RHF0M31 30
4.2.3 IC880A 31
4.2.4 Elección de gateway 32
4.3 Sensores de luz 32
4.3.1 Análisis de sensores luz 33
4.3.1.1 GY-49 (MAX 44009) 33
4.3.1.2 GY - VELM6070 34
4.3.1.3 TEMT 6000 34
4.3.1.4 CJMCU-3001 OPT3001 35
4.3.1.5 BH - 1750 36
4.3.2 Análisis y resultados 37
4.4 Antenas 39
4.4.1 Antena gateway 40
4.4.2 Antena nodo 41
4.5 Baterías 42
5. Desarrollo del producto 43
5.1 Hardware 43

5.1.1 Alimentación 43
5.1.2 Antena 44
5.1.3 Sensores 45
5.1.4 Medición de batería 46
5.1.5 Circuitos auxiliares 48
5.1.6 PCB 48
5.1.6.1 Versión 1 49
5.1.6.2 Versión 2 49
5.1.6.3 BOM 53
5.2 Firmware 53
5.2.1 Flujo principal de programa 54
5.2.2 Características del Firmware del Microcontrolador 56
5.2.3 I2C 57
5.2.3.1 Composición de mensaje 57
5.2.3.2 Comunicacion RAK con sensor de luz 58
5.2.3.3 Configuracion BH1750 58
5.2.3.4 Lectura I2C del sensor BH1750 58
5.2.3.4 Implementacion procolo I2C en el microcontrolador 59
5.2.4 Sleep mode 59
5.2.5 Debug 60
5.2.5.1 Análisis funcionamiento del código 60
5.2.5.2 Interpretación mensaje I2C 61
5.2.5.3 Testeo mensaje LoRa 62
5.2.6 Resumen de la evolución del código 62
5.3 Software 63
5.3.1 Diagrama bloques inicial 63
5.3.2 Diagrama bloques final 64
5.3.3 TTN en software 65
5.3.4 Python 66
5.3.4.1 Identificación aplicación TTN 66

5.3.4.2 Lectura datos TTN 67
5.3.4.3 Procesamiento y lectura sistema aeropuerto 67
5.3.5 MySQL 68
5.3.6 Sitio web 69
5.4 Encapsulado 71
6. Simulación del proyecto 72
6.1 Luminaria aeropuerto 72
6.2 Armado del nodo 75
6.3 Protección gateway 76
6.4 Sistema de control luminaria 77
7. Muestra de resultados 79
7.1 Alcance 79
7.1 Sensor de luz 80
7.2 Encapsulado 82
7.3 Consumo del nodo 83
7.3.1 Consumo fijo 83
7.3.2 Consumo Transitorio 85
8. Estimación y costos 90
8.1 Costo del producto 90
8.1.1 Cotización componentes 91
8.1.2 Cotización encapsulado 92
8.1.3 Presupuesto unitario 92
9. Conclusiones 94
9.1 Generales 94
9.2 Académicas 94
9.3 Reflexiones personales 95
9.4 Cronograma de proyecto 96
10. Trabajos a futuro 97
10.1 Seguridad y almacenamiento de la información 97
10.2 Automatización instalación 97

10.3 Comunicación con el nodo 97
10.4 Cambios en el circuito 98
10.5 Rediseñar el encapsulado 99
11. Bibliografía 100
12. Glosario 103
13. Anexo 105
13.1 Firmware Nodo 105
13.1.1 Board_init.c 105
13.1.2 conf_app.h 106
13.1.3 main.c 108
13.1.4 enddevice.c 109
13.1.5 I2C_nodo.c 117
13.2 Software servidor 119

1. Introducción

La iluminación de los aeropuertos es de vital importancia en la seguridad aeronáutica. Las tareas de mantenimiento y control de estas, deben ser realizadas por los operarios de los aeropuertos periódicamente, siendo estos procedimientos regularizados y definidos en diferentes estándares internacionales.

Este proyecto de grado está centrado en la creación de un sistema complementario, que permita visualizar y notificar al personal técnico del aeropuerto en caso de un mal funcionamiento de una luminaria. Esto facilita las tareas de control, reparaciones y agregaría valor a la seguridad del aeropuerto.

Por otra parte, este desarrollo si bien está orientado a una aplicación específica, puede ser utilizado en diversas ramas. Siendo que el gran avance de esta tesis se encuentra en la utilización de LoRaWAN como tecnología inalámbrica para la transmisión de datos y en la utilización de un microcontrolador moderno con transceptor LoRa incorporado de muy bajo consumo e ínfimo tamaño.

1.1 Problema a resolver

Actualmente en el Aeropuerto Internacional de Carrasco la inspección de las luces de pista y perimetral es realizada por diferentes operarios. Quienes se encargan de efectuar al menos una recorrida diaria por todas las luminarias con el objetivo de verificar su correcto funcionamiento. Estas tareas implican ingresar a lugares de difícil acceso y un gasto de tiempo considerable para el personal del aeropuerto.

En algunos aeropuertos de la región se han colocado sistemas complementarios, que permiten notificar al personal técnico en caso de un mal funcionamiento de una luminaria. Estos sistemas dan aviso al personal técnico del aeropuerto, quienes realizarán la reparación lo antes posible.

Estos sistemas tienen la particularidad de ser muy costosos y muchas veces presentan un tiempo de instalación importante, siendo este último un dato no menor por las dificultades que presenta acceder a zonas cercanas a las pistas de un aeropuerto.

En coordinación con el área de innovación del Aeropuerto Internacional de Carrasco y personal técnico del mismo, se planteó la posibilidad de realizar un proyecto capaz de monitorear el funcionamiento de las luminarias de

manera no invasiva, es decir sin modificar la instalación y funcionamiento de las mismas.

Por otra parte este sistema debe ser lo más económico posible, para que pueda ser viable y escalable a distintos aeropuertos pequeños de la región.

Estas limitantes generaron que el sistema que se desarrolló no puede ser alimentado por la energía de las lámparas actuales y los datos deben ser transmitidos de manera inalámbrica.

1.2 Estado del arte

Actualmente los aeropuertos controlan las luces de pista mediante sistemas cableados, estos fueron diseñados e instalados cuando se construyó la misma o en obras posteriores. El agregado de nuevos componentes o reformas invasivas, generan inconvenientes en el normal funcionamiento del aeropuerto.

Los sistemas actuales de monitoreo de luminarias son parte de otros sistemas más grandes que realizan varias tareas del aeropuerto. La gran mayoría de estos se basan en medir la corriente para determinar el correcto funcionamiento de las luces.

Estos sistemas no son posibles de instalar fácilmente debido a los grandes cambios que se deben realizar y por los altos costos asociados al mismo. Por este motivo surge la necesidad de realizar un producto capaz de monitorear las luces, detectando la luminosidad que emiten. Este producto será compatible con cualquier tecnología de lámparas, con una instalación muy simple y no invasiva con la instalación actual.

1.3 Objetivo

El objetivo principal en esta tesis de grado, es crear un sistema de monitoreo que permita determinar el correcto funcionamiento de las luces en los aeropuertos de forma remota. Esta tarea de gran importancia en la seguridad aeronáutica es realizada actualmente por los operarios de aeropuertos. Este proyecto tiene como objetivo realizar un sistema complementario al existente, ya que por las normas internacionales se deben continuar realizando inspecciones visuales.

Este desarrollo particular está orientado a todas las luminarias de los aeropuertos donde se necesite verificar su funcionamiento, como por ejemplo balizas de pista y aproximación, luces de alumbrado público, luminarias decorativas, etc. Dependiendo del tipo de luminaria se controlarán diferentes

parámetros, como su estado (prendidas o apagadas) y la intensidad lumínica de cada una.

1.4 Solución planteada

Para resolver este problema se implementará una red de sensores inalámbricos o nodos (colocados próximo a las luces), que se comunicarán con un servidor central informando su estado actual. En el caso de detectar una falla en alguna luz, notificará mediante una interfaz al usuario u operario correspondiente. Los nodos deberán ser de tamaño pequeño y de larga autonomía.

Los aeropuertos se caracterizan por su amplia dimensión, por este motivo para lograr la comunicación entre los dispositivos y el servidor se utilizará la tecnología LoRaWan, siendo esta especializada en comunicaciones de largo alcance y bajo costo energético. Para la implementación de esta red, los nodos se comunicarán con un gateway y mediante la plataforma TTN (The Things Network) podrán ser enviados a un servidor central que procesa los datos. Por último la información enviada por los nodos se mostrarán en un servidor privado a través de un sitio web.

Para el desarrollo de la solución planteada anteriormente, el proyecto se dividirá en tres ramas principales:

- Nodos o terminales
- Red LoRaWAN
- Servidor y Visualizador web

Los nodos estarán compuestos por un microcontrolador con transceiver de LoRa, sensor de luz y batería, además contarán con un encapsulado plástico diseñado a medida. Estos enviarán los datos de luz por radiofrecuencia y es el servidor quien analizará estos valores para luego ser mostrados en el visualizador web.

Es importante mencionar que si bien este proyecto está orientado a una aplicación particular, en las etapas previas y durante la ejecución del mismo se buscó diseñar un sistema versátil que permite ser utilizado en distintas aplicaciones.

2. Requerimientos del proyecto

En la etapa de planeamiento y durante la ejecución del proyecto se analizaron distintas áreas que lo caracterizaron. Estas permitieron justificar y definir la solución adoptada.

2.1 Consumo Batería

Una de las principales limitantes del proyecto es el consumo de batería, debido a que el dispositivo diseñado se colocará en las proximidades de pista de los aeropuertos. Esta zona es de difícil acceso y requiere un protocolo de seguridad para el ingreso. Debido a esto, es fundamental que la duración de la batería sea mayor a un periodo de tres años.

2.2 Alcance de comunicación

Los aeropuertos se caracterizan por tener grandes dimensiones, por lo que la red diseñada debe poder cubrir toda su superficie.

Específicamente, en el aeropuerto internacional de Carrasco la distancia máxima posible entre un nodo y gateway es de 2.8 kilómetros (Se visualiza en Imagen 5)

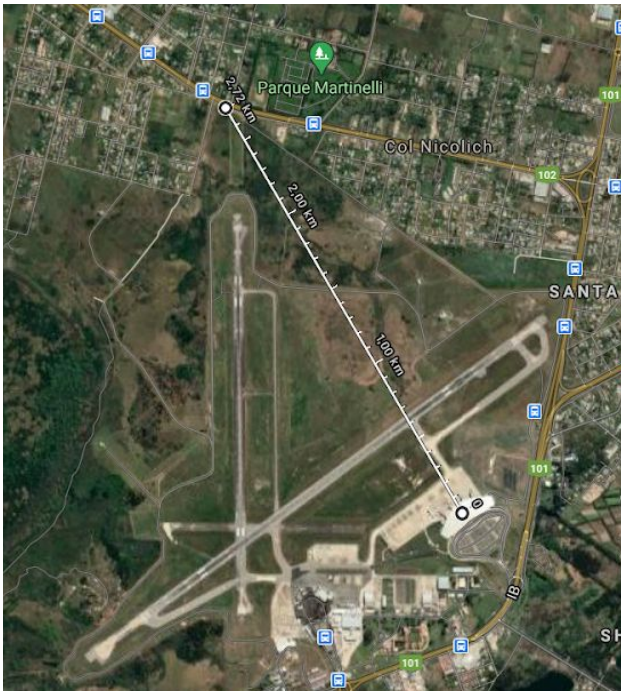

Imagen 5: Medición de máxima distancia del aeropuerto de carrasco [4]

La comunicación debe ser estable, es decir no pueden haber pérdidas de paquetes de información. Por esto se establece que la distancia mínima sea de cuatro kilómetros en cualquier situación climática, ya que este último factor puede afectar en la transmisión de los datos.

2.3 Confiable

La confiabilidad de un producto es un pilar fundamental en el desarrollo de proyectos. Esta característica presenta mayor importancia en la aeronáutica, debido a que un error puede generar un accidente fatal. Por eso, es que se debe ser muy cuidadoso en el desarrollo, buscando no dar ningún falso positivo y realizando una meticulosa etapa de testeo o pruebas del mismo.

2.4 Precio

El precio es también una característica muy importante a la hora de desarrollar un producto, este determina muchas veces la viabilidad del proyecto. Para determinar cuál sería un valor aceptable del producto se deben tomar en cuenta distintos factores, como por ejemplo la necesidad de utilización del producto, soluciones similares en el mercado y una estimación de costos del mismo.

Luego de investigar en internet y en reuniones con el personal técnico del Aeropuerto Internacional de Carrasco, las soluciones similares que se encuentran en el mercado son inviables por el costo elevado que presenta el producto y lo complicado que es su instalación.

Por último, en cuanto al costo de fabricación del producto se estima entre veinte y treinta dólares para una cantidad de producción de 10.000 unidades. Esta proyección será analizada y verificada en la sección de costos (8. Estimación y costos) donde se estimaron los valores para distintas cantidades de producción.

3. Tecnología inalámbrica

En esta sección se presenta la tecnología inalámbrica utilizada, siendo esta uno de los pilares más importante del proyecto. También se dará un marco teórico de la misma presentando sus principales características y ventajas frente a otro tipo de tecnologías.

3.1 Selección de tecnología

Para elegir la tecnología inalámbrica se analizaron diferentes parámetros de cada una, que se detallan a continuación.

Por un lado, se debe lograr una comunicación que consuma la menor cantidad posible de energía, ya que los mismos pueden contar únicamente con baterías para su funcionamiento.

Por otro lado, la tecnología que se utilice debe tener un alcance mayor a 4 km para poder cubrir las dimensiones de un aeropuerto de la región.

Debido a los requerimientos mencionados anteriormente, se analizaron las redes de tipo LPWAN, ya que este tipo de tecnología inalámbricas se caracterizan por ser de bajo consumo. Dentro de esta sección se profundizó en SigFox, LoRa, LTE-M, NB-IOT y ZigBee.

3.1.1 ¿Por qué LPWAN?

Es difícil que una tecnología se adapte a todas las aplicaciones y volúmenes proyectados para IoT.

Por ejemplo, WiFi y BLE son estándares ampliamente adoptados y funcionan bien con aplicaciones relacionadas con la comunicación de dispositivos personales. Sin embargo este tipo de tecnologías presentan un alto gasto de energía, lo que hace imposible su utilización en aplicaciones de bajo consumo. Algo similar sucede con la tecnología celular que es ideal para aplicaciones que necesitan un alto rendimiento de datos y tienen una fuente de energía.[1]

En cambio LPWAN está diseñado para sensores y aplicaciones que necesitan enviar pequeñas cantidades de datos a largas distancias, unas pocas veces por hora. Esto permite un consumo mínimo de energía, alargando la vida útil de las baterías. En la “Imagen 1” se puede ver en qué tipo de tecnología encaja LPWAN comparado con las ya existentes.

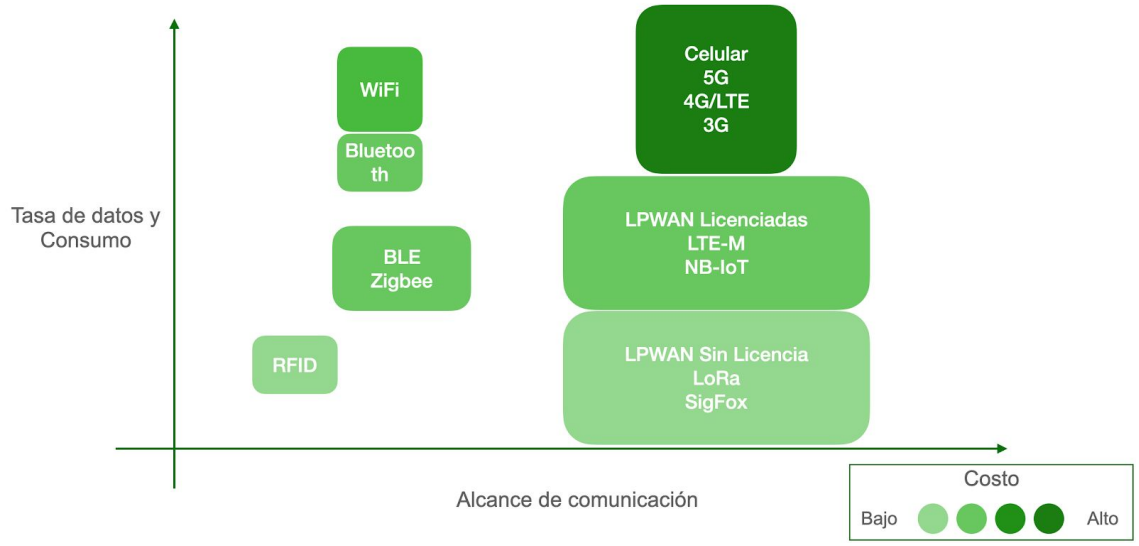

Imagen 1: Comparación de diferentes tecnologías. [2]

3.1.2 SigFox

Es una tecnología desarrollada por una empresa con el mismo nombre, que proporciona una infraestructura de comunicación propia. En esta solamente es necesario verificar que haya cobertura y adquirir un dispositivo con capacidad de comunicarse por SigFox, para comenzar a utilizarla. Es importante mencionar que esta tecnología requiere del pago de una cuota mensual por dispositivo.[3]

El principal problema de esta tecnología es que al momento de utilizarla se depende de que la empresa tenga cobertura en la zona de trabajo. En este caso, SigFox no tiene cobertura en el Aeropuerto Internacional de Carrasco.

3.1.3 LoRa

Es un tipo de modulación en radiofrecuencia similar en características a SigFox, pero en cambio la red puede ser implementada y gestionada por el usuario. Ya que solamente se necesita un gateway que reciba los datos de los sensores y luego los redireccione a un servidor local o alojado en internet.

3.1.4 LTE-M

LTE-M está basada en la red LTE/4G, pero optimizada para IoT. Esto le da la ventaja que mantiene los niveles de seguridad y privacidad de LTE, además de una implementación con un nivel de complejidad relativamente baja. Tiene además mayores capacidades de transmisión de datos que las tecnologías anteriores.

Por otro lado, LTE también depende del despliegue de la red que pueda haber en la zona donde se utilice, particularmente el Aeropuerto Internacional de Carrasco presenta cobertura. Sin embargo el gasto excesivo de energía, la descarta para aplicaciones de bajo consumo.

Por último, al igual que SigFox, es necesario contratar un plan de datos para hacer uso del servicio.

3.1.5 NB-IoT

Por su nombre “NarrowBand - Internet of Things” es una red de banda angosta desarrollada específicamente para IoT, la misma está basada en LTE para su funcionamiento.

Opera en las mismas frecuencias que LTE pero tiene menores consumos teóricos que LTE-M, además soporta mayor cantidad de usuarios conectados a una misma radio base. Para poder utilizar NB-IoT es necesario contratar un servicio de red debido a que son tecnologías que trabajan en frecuencias con licencia.

3.1.6 ZigBee

Basada en el estándar IEEE 802.15.4, es un protocolo diseñado para bajo consumo, que utiliza la comunicación punto a punto o punto-multipunto. Esto significa que para que la red pueda funcionar cada sensor se debe poder comunicar con al menos otro para formar una interconexión estrella. Debido a esto los sensores deben estar siempre en modo recepción, generando un consumo fijo elevado.

Por otra parte, la complejidad de la red aumenta a medida que se agregan más sensores.

3.1.7 Elección de LoRa

Al momento de la elección de la tecnología, se basó en tres pilares fundamentales, como lo son el bajo consumo, el largo alcance y la posibilidad de poder crear una red privada a medida, en cada situación sin pagar un servicio extra.

Debido al consumo mayor, se descartan LTE-M y ZigBee. A pesar que esta última da la posibilidad de crear un red privada, al momento de utilizar muchos dispositivos el consumo aumenta rápidamente.

Por otro lado, las redes donde es necesario utilizar un servicio pago de un tercero como lo son SigFox, NB-IoT y LTE-M, fueron descartadas. Debido a

no poder crear una red privada, si las mismas no tienen cobertura en el lugar no se pueden utilizar.

Por último, LoRa aporta al proyecto lo necesario para el desarrollo. El consumo de esta tecnología es muy bajo y brinda la posibilidad de implementar una red totalmente privada. Esto se adapta mejor a las necesidades del proyecto, brindando una seguridad extra a la red.

3.2 LoRa

La información referida a LoRa y LoRaWAN, tanto como la base de su funcionamiento, como sus especificaciones fueron obtenidas desde el sitio web de LoRa Alliance [1]. El mismo es una asociación sin fines de lucro que se dedica a la promoción y desarrollo del protocolo LoRaWAN, brindando certificaciones para garantizar la interoperabilidad de la red.

LoRa es la capa física o la modulación inalámbrica utilizada para crear el enlace de comunicación de largo alcance. Muchos sistemas inalámbricos utilizan modulación por desplazamiento de frecuencia (FSK, por sus siglas en inglés) como capa física porque es una modulación muy eficiente para bajo consumo. LoRa se basa en la modulación de espectro ensanchado, que mantiene las mismas características de baja potencia que la modulación FSK, pero aumenta significativamente el rango de comunicación. Esta modulación se ha utilizado tanto en comunicaciones militares como espaciales durante décadas, debido a las largas distancias de comunicación que puede lograr y la solidez que presenta ante la interferencia. LoRa es la primera implementación de bajo costo de esta tecnología para uso comercial.

La ventaja de LoRa está en la capacidad de largo alcance de la tecnología. Una sola puerta de enlace (Gateway) puede cubrir ciudades enteras o cientos de kilómetros cuadrados. El alcance depende en gran medida del entorno o de las obstrucciones en una ubicación determinada, pero LoRa y LoRaWAN tienen una capacidad mayor que cualquier otra tecnología de comunicación estandarizada. Con una cantidad mínima de infraestructura, se pueden cubrir fácilmente países enteros.

3.3 LoRaWAN

LoRaWAN define el protocolo de comunicación y la arquitectura del sistema para la red, mientras que la capa física LoRa habilita el enlace de comunicación de largo alcance. El protocolo y la arquitectura de la red tienen la mayor influencia en la determinación de la vida útil de la batería de un nodo, la

capacidad de la red, la calidad del servicio, la seguridad y la variedad de aplicaciones que atiende la red.

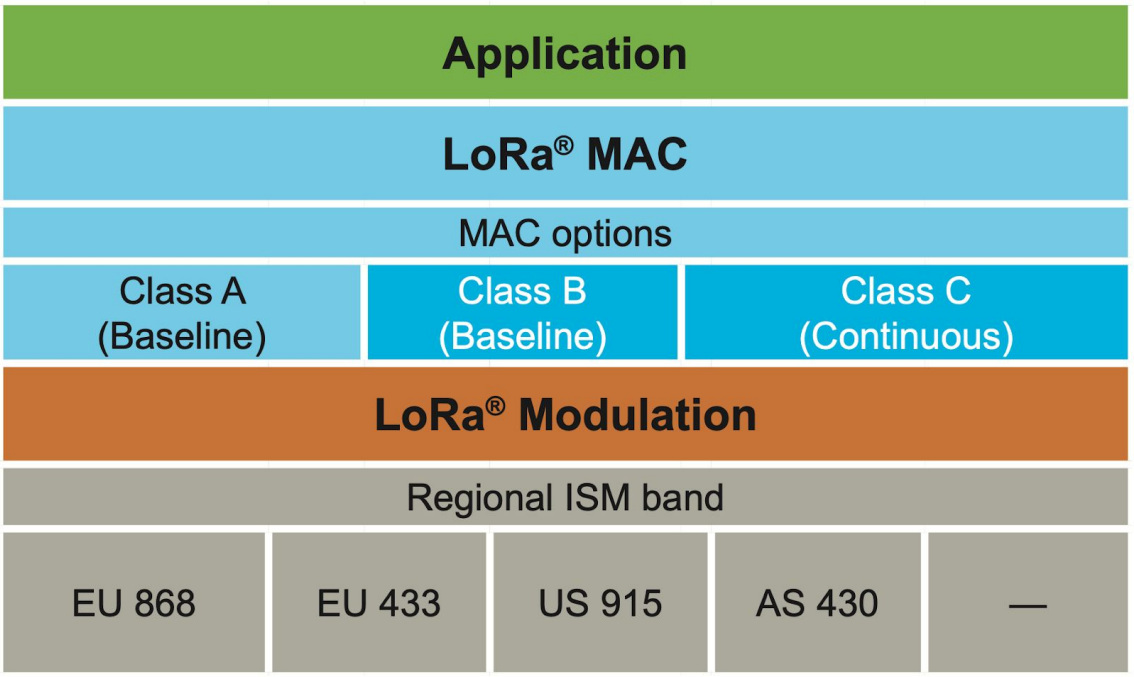

Imagen 2: Esquema de funcionamiento LoRaWan [1]

3.3.1 Arquitectura de la red

Muchas redes implementadas existentes utilizan una arquitectura de red en malla. En una red de malla, los nodos finales individuales envían la información de otros nodos para aumentar el rango de comunicación y el tamaño de celda de la red. Si bien esto aumenta el rango, también agrega complejidad, reduce la capacidad de la red y reduce la vida útil de la batería, ya que los nodos reciben y envían información de otros nodos que probablemente sea irrelevante para ellos. La arquitectura en estrella de largo alcance tiene más sentido para preservar la vida útil de la batería logrando una conectividad de largo alcance, por esta razón el diseño de la red LoRaWAN se implementó en estrella.

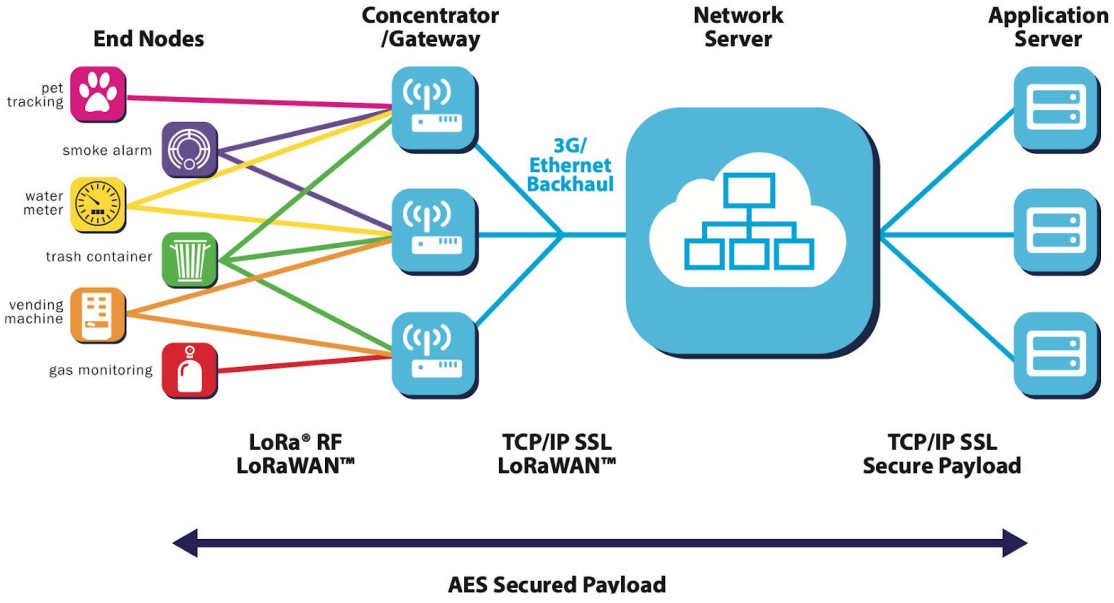

Imagen 3: Esquema de red LoRaWAN [1]

En una red LoRaWAN, los nodos no están asociados con un gateway específico. En cambio, los datos transmitidos por un nodo generalmente son recibidos por múltiples gateways. Cada uno reenviará el paquete recibido desde el nodo al servidor de red en la nube a través de alguna conexión a la red (ya sea celular, Ethernet, satélite o Wi-Fi).

La inteligencia y la complejidad de la red se envían al servidor, que es el encargado de administrar y filtrar los paquetes recibidos redundantes, realiza verificaciones de seguridad, programa respuestas por el gateway más adecuado, maneja la velocidad de los datos, etc. Si un nodo es móvil o está en movimiento, no es necesario realizar una transferencia de un gateway a otro, siendo esto una característica fundamental para habilitar las aplicaciones de seguimiento de activos, una de las principales aplicaciones de destino muy utilizado para IoT.

3.3.2 Duración de batería

Los nodos en una red LoRaWAN son asíncronos y se comunican cuando tienen datos listos para enviar, ya sea por eventos o programados. Este tipo de protocolo generalmente se conoce como el método Aloha. En una red de malla o con una red síncrona, como la celular, los nodos con frecuencia tienen que "despertarse" para sincronizarse con la red y buscar mensajes. Esta sincronización consume una cantidad significativa de energía y es el principal factor de reducción de la vida útil de la batería.

3.3.3 Capacidad de la red

Para que una red en estrella de largo alcance sea viable, el gateway debe tener una capacidad muy alta para recibir mensajes de un gran volumen de nodos. La alta capacidad de una red LoRaWAN se logra utilizando una velocidad de datos adaptativa y un transceptor del gateway multi-módem y multicanal para que de esta forma reciba mensajes simultáneos en varios canales.

Los factores críticos que afectan la capacidad son el número de canales concurrentes, la velocidad de datos(tiempo en el aire), la longitud del dato y la frecuencia con la que transmiten los nodos.

Dado que LoRa es una modulación basada en espectro ensanchado, las señales son prácticamente ortogonales entre sí cuando se utilizan diferentes factores de ensanchamiento. A medida que cambia el factor de dispersión, también cambia la velocidad de datos efectiva. El gateway aprovecha esta propiedad al poder recibir múltiples velocidades de datos diferentes en el mismo canal al mismo tiempo. Si un nodo tiene un buen enlace y está cerca de una puerta de enlace, no hay razón para que utilice siempre la velocidad de datos más baja y llene el espectro disponible más tiempo del necesario. Al aumentar la velocidad de datos, el tiempo en el aire se acorta, lo que abre más espacio potencial para que otros nodos transmitan.

La velocidad de datos adaptable también optimiza la vida útil de la batería de un nodo. Para que la velocidad de datos adaptable funcione, se requiere un uplink y downlink simétrico con suficiente capacidad downlink.

Estas características permiten que una red LoRaWAN tenga una capacidad muy alta y haga que la red sea escalable fácilmente. Es posible implementar una red con una cantidad mínima de infraestructura y, a medida que se necesita capacidad, se pueden agregar más gateways, lo que aumenta las velocidades de datos, reduce la cantidad de sobre escucha de otros gateway.

Otras alternativas LPWAN no tienen la escalabilidad de LoRaWAN debido a las compensaciones tecnológicas, que limitan la capacidad del downlink o hacen que el rango del downlink sea asimétrico con respecto al del uplink.

3.3.4 Clases de nodos

Los nodos finales son utilizados para diferentes aplicaciones, por lo tanto tienen diferentes requisitos. Para optimizar una variedad de perfiles de aplicaciones, LoRaWAN utiliza diferentes clases de nodos. Las clases de

nodos compensan la latencia de la comunicación del downlink de la red con la vida útil de la batería.

En una aplicación de control o de tipo actuador, la latencia de la comunicación del enlace descendente es un factor importante, no en cambio la medición de la altura de un río por ejemplo que varía muy poco en una hora.

Nodos(Clase A): Los nodos Clase A permiten comunicaciones bidireccionales. Cada transmisión uplink del nodo es seguida por dos ventanas de downlink cortas. El intervalo de transmisión programado por el nodo se basa en sus propias necesidades de comunicación, con una pequeña variación basada en un tiempo aleatorio (tipo de protocolo ALOHA). Esta Clase A es el sistema de comunicación de menor consumo para aplicaciones que solo requieren un downlink. Las comunicaciones downlink enviadas desde el servidor en cualquier otro momento tendrán que esperar hasta el próximo uplink.

Nodos bidireccionales con ventanas de recepción programadas (Clase B): además de las ventanas de recepción aleatoria de Clase A, los dispositivos de Clase B abren ventanas de recepción adicionales en horarios programados. Para que el nodo abra su ventana de recepción a la hora programada, recibe un beacon sincronizado desde el gateway. Esto permite que el servidor sepa cuándo está escuchando el dispositivo final.

Nodos bidireccionales con recepción continua (Clase C): Los dispositivos finales de Clase C tienen ventanas de recepción casi continuamente abiertas, solo cerradas durante la transmisión.

3.3.5 Seguridad

Es extremadamente importante para cualquier LPWAN incorporar seguridad. LoRaWAN utiliza dos capas de seguridad: una para la red y otra para la aplicación.

La seguridad de la red da autenticidad al nodo, mientras que la capa de seguridad de la aplicación garantiza que el gateway no tenga acceso a los datos de la aplicación del usuario final. La única forma de acceder a los datos de la aplicación es obteniendo las credenciales de acceso, que están ocultas dentro de cada aplicación. El cifrado AES es el encargado de manejar el intercambio de claves utilizando un identificador IEEE EUI64.

3.3.6 Especificaciones regionales

Las especificaciones LoRaWAN varían de una región a otra según las diferentes asignaciones regionales de espectro y los requisitos reglamentarios.

La especificación LoRaWAN para Europa y Norteamérica está definida, pero el comité técnico aún está definiendo otras regiones. Actualmente en Uruguay se utilizan las especificaciones de Norteamérica.

	Europe	North America	China	Korea	Japan	India
Frequency band	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz
Channels	10	64 + 8 +8	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee
Channel BW Up	125/250kHz	125/500kHz				
Channel BW Dn	125kHz	500kHz				
TX Power Up	+14dBm	+20dBm typ (+30dBm allowed)				
TX Power Dn	+14dBm	+27dBm				
SF Up	7-12	7-10				
Data rate	250bps- 50kbps	980bps-21.9kpbs				
Link Budget Up	155dB	154dB				
Link Budget Dn	155dB	157dB				

Imagen 4: Especificaciones LoRaWan por región [1]

La banda ISM para América del Norte, también utilizada en Uruguay es de 902-928 MHz. LoRaWAN define 64 canales de uplink de 125 kHz de ancho desde 902,3 MHz hasta 914,9 MHz en incrementos de 200 kHz. Hay ocho canales de uplink de 500 KHz de ancho adicionales en incrementos de 1,6 MHz de 903 MHz a 914,9 MHz. Los ocho canales de downlink tienen una ancho de 500 kHz, desde 923,3 MHz hasta 927,5 MHz. La potencia permitida máxima en la banda de 902-928 MHz de América del Norte es de +30 dBm, pero para la mayoría de los dispositivos +20dBm es suficiente. Según la FCC, no hay limitaciones de ciclo de trabajo, pero hay un tiempo máximo de permanencia de 400 ms por canal.

3.4 The Things Network (TTN)

Es una red abierta formada por comunidades de todo el mundo que interconecta redes LoRaWAN mediante internet. Además TTN posee servidores que centralizan todos los datos recibidos por distintos gateways. Esto hace posible el acceso a los datos de los nodos desde cualquier parte del mundo.

Los servidores de TTN son los que realizan el backend de toda la red manejando los datos recibidos y gestionando la integración con distintas plataformas. Esto brinda la posibilidad de integrar (mediante HTTP o MQTT) a sus servidores distintas aplicaciones para poder realizar un proyecto el que integre nodos, gateways, servidor TTN y una aplicación para visualización de los datos.

En caso de realizar un red privada, la implementación de TTN no es necesaria, ya que los datos son enviados directamente desde el gateway a un servidor privado de la empresa, evitando así la conexión del sistema a internet.

4. Elección del hardware

En esta sección se presentarán las diferentes alternativas de hardware que fueron analizadas en distintas partes del proyecto y la solución adoptada con la justificación correspondiente.

4.1 Microcontrolador - Transceptor

Elegir el microcontrolador y el transceptor Lora es una parte importante en el desarrollo de un producto. Este presenta la limitante que debe ser alimentado a batería, por lo que los componentes elegidos deben ser de muy bajo consumo.

Se realizó un estudio del mercado para ver cuales eran las mejores opciones disponibles, donde se obtuvieron tres familias distintas que se detallan a continuación.

4.1.1 Transceptores familia SX12xx

Estos tipos de transceptores LoRa, se pueden encontrar en distintos formatos como se observa en la siguiente figura. Como característica principal, es que son de muy bajo consumo y muy utilizados en el mercado.

Imagen 6: Transceptores de la familia SX12xx [5]

Por otra parte, esta familia tiene únicamente un transceptor LoRa por lo que para poder utilizarlos en cualquier aplicación, requieren de un microcontrolador.

Una opción posible para este proyecto es la placa de desarrollo LoPy de la empresa Pycom. Esta placa tiene un transceptor LoRa y un microcontrolador

de alta potencia el cual se puede programar en lenguaje de alto nivel como es Python .

Sin embargo, esta placa presenta un costo elevado (40 USD) y un tamaño grande para esta aplicación específica (55x20x25 mm). Por esta razones y por no ser una placa de ultra bajo consumo, hace que no sea viable para este proyecto.

Imagen 7: Lopy pycom (SX1272) [6]

4.1.2 Transceptores familia RFM95/96/97/98

Estos módulos son muy populares, por lo que tienen la ventaja de presentar mucha documentación. Se pueden encontrar distintos modelos, con diferentes potencia de transmisión y recepción que dependerá de la distancia que se quiera cubrir. Estos transceptores presentan un muy bajo costo en el mercado, aunque también necesitan un microcontrolador para ser utilizados.

Imagen 8: Transceptor RFM95 [7]

Una opción disponible es la placa de desarrollo Moteino basada en Atmel que incluye transceptores LoRa de la familia RFM95. Esta placa cuesta alrededor de 25 USD en el mercado y se puede programar desde el Arduino IDE.

Esta placa consume menos de 2 µA en modo sleep, por lo que podría ser utilizada para este tipo de aplicaciones. Sin embargo no se obtuvo un buen desempeño en las pruebas de alcance realizadas. En estas se obtuvo una distancia de comunicación estable máxima de un kilómetro, haciendo que no sea viable para el desarrollo.

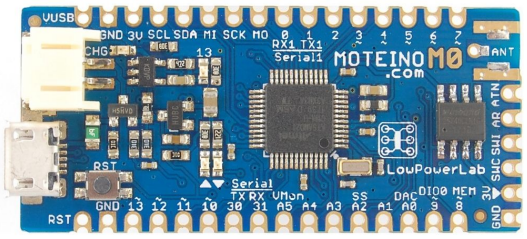

Imagen 9: Placa de desarrollo con transceptor RFM95 [8]

4.1.3 Transceptores familia SAMR34

El SAMR34 tiene la cualidad de tener un microcontrolador ARM y un transceiver LORA en el mismo integrado. Generando que sea muy versátil para diversas aplicaciones.

Imagen 10: SAMR34 [9]

Este chip, posee un pinout muy difícil de soldar siendo más habitual encontrarlo dentro de otras soluciones. Una de las empresas que lo utiliza en sus módulos es RAK, de donde seleccionamos para esta aplicación el RAK4260.

Imagen 11: RAK4260 [10]

Características Principales:

- Dispositivo LoRa® de alta calidad y baja potencia (Microchip ATSAMR34J18 SiP)
- Cobertura de frecuencia de 862 a 1020 MHz
- Alto nivel de precisión y estabilidad (32MHz TXCO)
- Potencia máxima de transmisión : 20dBm
- Sensibilidad máxima (SF12 BW125kHz) : -136dBM
- Corriente Rx: 17mA (típica)
- Una amplia selección de interfaces: I2C, SPI, ADC, UART, GPIO

Este microcontrolador cuesta actualmente aproximadamente 8 USD y tiene como gran ventaja de tener integrado el microcontrolador y el transceptor. Por otra parte es un chip de muy bajo consumo (aproximadamente 1 µA en sleep Mode) convirtiéndolo en el ideal para este tipo de aplicaciones a batería. Sin embargo, al ser un chip relativamente moderno, no se dispone de suficiente documentación.

La empresa RAK también ofrece una placa de desarrollo que permite realizar todas las principales pruebas antes de diseñar tu propio PCB , o utilizar directamente esta placa para aplicaciones específicas.

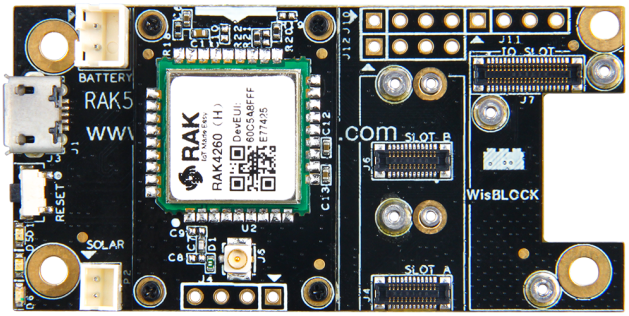

Imagen 12: placa de desarrollo para RAK4260 [10]

4.1.3 Análisis y selección

Para poder ver las características principales de las diferentes familias se creó la tabla que se muestra a continuación.

Microcontrolador - Transceptor	SX12xx	RFM9x	SAMR3x
Microcontrolador	no	no	**si**
Transceptor LoRa	si	si	si
Lenguaje	AT command	AT command	C
Costo Aproximado	4USD	3.5USD	5USD
Consumo sleep Max	1000nA	1000nA	**790nA**
Consumo standby	1.6mA	1.6mA	**1.4µA**
Potencia Transmisión	20dbm	20dbm	20dbm
Sensibilidad Máx	-148dbm	-148dbm	-148dbm
Typical Voltage	3.3V	3.3V	3.3V
Documentación	Mucha	Mucha	Poca

Tabla 1: Comparativas diferentes transceptores

Por su bajo consumo, gran alcance, tener todos los componentes en un mismo integrado y la posibilidad de utilizar I2C junto con entradas analógicas se decidió por el SAMR34.

El RAK4260 utiliza este microcontrolador, agregando únicamente los componentes para que sea más sencillo soldarlo y algunos elementos del transmisor LoRa.

Estas características del microcontrolador permite hacer un diseño pequeño con baja cantidad de componentes.

El RAK 4260 consta de un puerto JTAG únicamente para su programación. Por otra parte, este consta de puertos seriales con el fin de realizar tareas de debug y comunicación en general. En este caso se utiliza un

J-link de Segger que ya viene previsto para poder programar con Atmel studio 7.0.

4.2 Gateway

Actualmente en el mercado existen muchas marcas y modelos de gateways LoRaWAN. A raíz de esto la decisión se basó en algunas características básicas necesarias que debían cumplir para que la red desplegada funcione correctamente.

La principal característica que se toma en cuenta es la capacidad de gateway de soportar grandes cantidades de dispositivos conectados al mismo tiempo. Por esta razón el gateway debe ser de 8 canales, lo que le da la posibilidad de recibir cientos de mensajes al mismo tiempo sin perder información.

El gateway debe ser un dispositivo que sea ampliamente utilizado actualmente, lo que le da una confiabilidad mayor. Esto es una característica muy relevante a la hora de la elección debido a que el mismo debe funcionar perfectamente en cualquier momento. En caso de que el gateway tenga problemas, toda la red se verá afectada. Además la comunidad de desarrolladores debe ser amplia y con un buen soporte, apoyada con documentación clara.

Es importante que el gateway tenga entre sus características la opción de generar un servidor privado. De esta forma da la posibilidad de desplegar una red tanto pública como privada.

Se tomó en cuenta el precio del dispositivo debido a que según reglamentación de importaciones uruguayas, los envíos menores a U$S 200 pueden ser ingresados sin pago de impuestos extras.

4.2.1 RAK 2245 (Pi Hat)

El RAK 2245 es un módulo que se conecta a un Raspberry PI y funciona como un gateway con todas las funcionalidades. Este soporta 8 canales en todas las frecuencias de LoRaWAN del mundo. En su interior, consta de un transceptor Semtech SX1301 que es el encargado del manejo de paquetes LoRa. Además posee dos Semtech SX125x que son los transceptores I/Q de radio frecuencia.

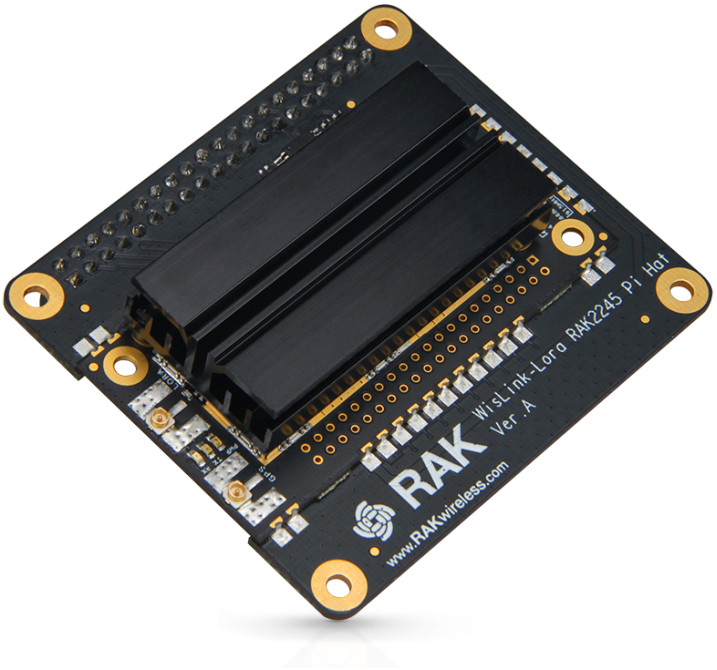

Imagen 13: Gateway LoRa RAK 2245 [11]

Características:

- Compatible con Raspberry Pi 3B + y 4.
- Modulo GPS Ublox MAX-7 integrado con disipador de calor.
- Procesador de banda base SX1301, emula 49 demoduladores LoRa, 10 rutas de modulación paralela.
- Soporta 8 canales de downlink y 1 de uplink
- Integra 2 unidades SX125x de Tx/Rx para alta y baja frecuencia.
- Admite alimentación 5V.
- Potencia Tx hasta 27 dBm
- Sensibilidad Rx hasta -139 dBm @SF12 y BW 125 kHz.
- Soporta el protocolo LoRaWan 1.0.2.
- Soporta todas las frecuencias de licencia libre del mundo (EU433, CN470,EU868, US915, AS923, AU915, KR920, IN865 y AS920
- Soporta interfaz SPI, UART e I2C.

4.2.2 RHF0M31

Posee el mismo chip que el RAK 2245 (Semtech SX1301), ofreciendo además todas las frecuencias utilizadas a nivel mundial. Soporta 8 canales de recepción lo que permite un mayor número de nodos. Tiene una sensibilidad de -142 dBm siendo esta mejor que el anterior.

Un problema importante que posee este gateway es que la documentación disponible en la web es muy pobre y tiene pocos usuarios utilizando este módulo actualmente.

Imagen 14: Módulo RHF0M301 [12]

4.2.3 IC880A

Es un transceptor que cumple con todos los requisitos planteados anteriormente, debido a que es un potente, bastante utilizado y con buen soporte en caso de tener problemas al momento del funcionamiento. Al igual que los anteriores trabaja con el Semtech SX1301. Por estas razones este módulo es una buena opción para utilizar.

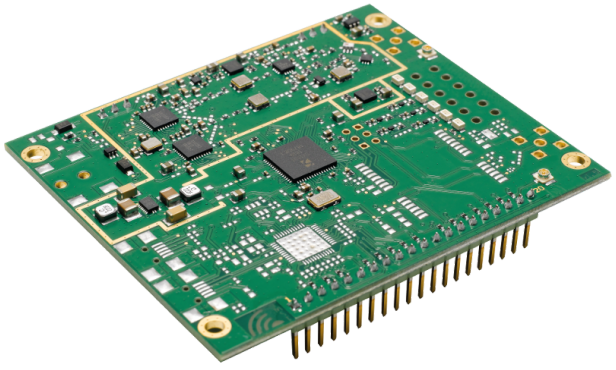

Imagen 15: Módulo IC880A [13]

4.2.4 Elección de gateway

Partiendo de la base de estos 3 módulos que cumplen con las características necesarias, se tomó la decisión basándose principalmente en el soporte que cada uno tiene. Se decidió por el RAK 2245, siendo este muy utilizado por la comunidad LoRaWAN, lo que lleva a que muchos problemas que se podrían enfrentar eventualmente, están resueltos con anterioridad. Además existen foros que tienen mucha actividad en los que usan este módulo.

Cabe destacar que este gateway es fabricado por "RAK Wireless" al igual que el nodo, como se puede ver en 4.1 Microcontrolador - Transceptor, siendo esta una ventaja muy importante al momento de realizar las compras, ahorrando costos de envíos y tiempo.

La configuración del gateway se realizó siguiendo la guia de RAKWireless que se puede encontrar en su página web [14].

En la siguiente tabla se comparan los gateway presentados y analizados en esta sección.

Marca	RAKWireless	RisingHF	WiMOD
Modelo	RAK 2245	RHF0M31	IC880A
Transceptor	SX1301	SX1301	SX1301
Sensibilidad	-139 dBm	-140 dBm	-137 dBm
Canales	8	8	8
Potencia transmisión	27 dBm	24.5 dBm	20 dBm
Soporte	Muy bueno	Pobre	Bueno

Tabla 2: Comparativas diferentes gateways

4.3 Sensores de luz

Encontrar el sensor indicado es un factor importante ya que esto determina si el proyecto es funcional, pudiendo diferenciar si la luces están o no prendidas. Para esta aplicación el sensor de luz que se utilice debe presentar un consumo máximo de corriente de 2µA en modo sleep (mientras no se encuentra sensando).

Por otra parte, los sensores pueden comunicarse de manera analógica o digital mediante el protocolo I2C, siendo estas dos formas compatibles con el microcontrolador RAK 4260.

Por último el sensor debe tener una buena precisión en la medida, para poder determinar correctamente el estado de la luz y no obtener resultados erróneos o inconcluyentes.

4.3.1 Análisis de sensores luz

Por los motivos mencionados anteriormente, se realizaron pruebas a diferentes tipos de sensores, buscando el óptimo para esta aplicación específica.

Las primeras pruebas de campo realizadas para poder determinar cuál era el sensor que mejor se comportaba, fue medir la luz solar durante un tiempo determinado, guardando las lecturas y luego analizando los datos obtenidos.

Estas pruebas se realizaron en condiciones similares de luz en un mismo día, buscando hallar la estabilidad y precisión de medida de los diferentes sensores. Es bueno mencionar que los sensores estudiados tienen distinto rango de espectro, por lo que la cantidad de lux en condiciones idénticas puede diferir.

4.3.1.1 GY-49 (MAX 44009)

Este sensor se comunica mediante el protocolo I2C y tiene un rango muy amplio de medida desde 0.045 lux a 188.000 lux. Su voltaje de funcionamiento va desde 1.7 V a 3.6 V y su consumo es menor a 1µA . Costo del sensor 1 USD.

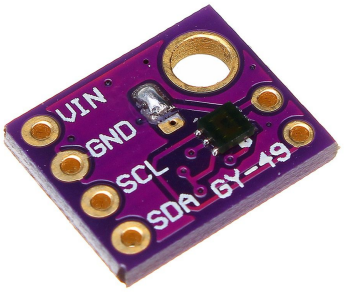

Imagen 16: Sensor lumínico GY-49 [15]

Se realizaron pruebas de campo con luz ambiente para observar la performance del sensor. Los resultados no fueron los adecuados para esta aplicación, ya que se generaban muchas fluctuaciones en la medida como se observa en el siguiente gráfico.

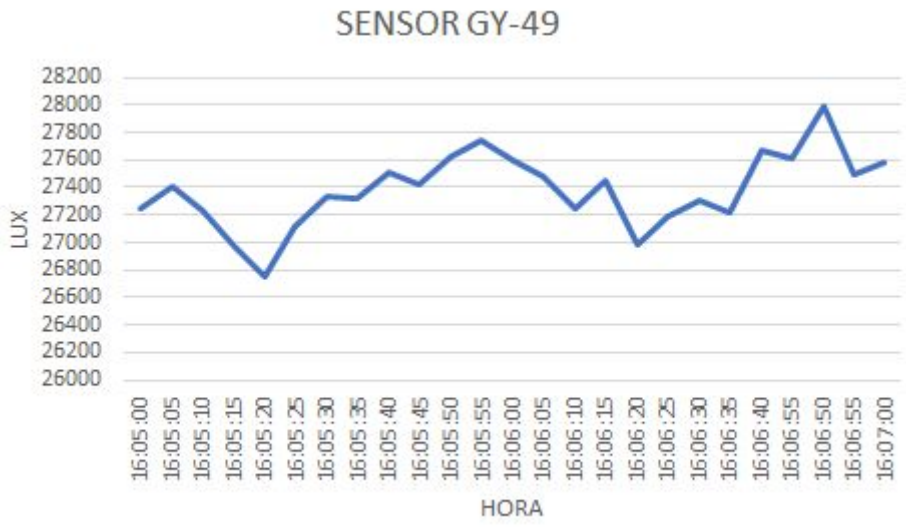

Imagen 17: Luminosidad, sensor GY-49

4.3.1.2 GY - VELM6070

Este sensor se comunica mediante el protocolo I2C y permite medir la luz ultravioleta con una sensibilidad del espectro de luz que va desde los 320 nm a los 410 nm. Su rango de alimentación es desde 2.5 V a 5.5 V y su consumo en sleep es menor a 1μA .

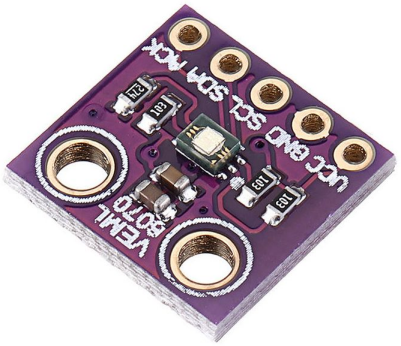

Imagen 18: Sensor VELM6070 [16]

El objetivo de utilizar este sensor era poder diferenciar la luz solar, de la luz que se necesita sensar. Sin embargo al momento de realizar las pruebas de campo, no se obtuvieron buenos resultados, ya que el sensor se veía afectado por cualquier tipo de luz en igual medida.

4.3.1.3 TEMT 6000

El TEMP 6000 es un sensor analógico que su voltaje de operación va entre 3.3V y 5.5V . Su sensibilidad máxima es de 570 nm y su rango de iluminación va entre 1 y 1000 Lux.

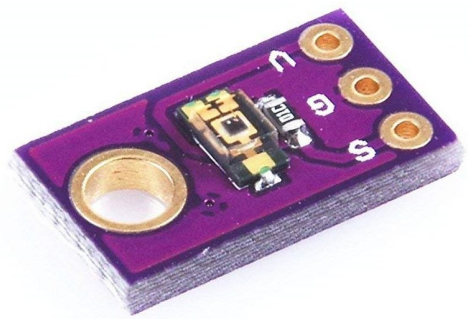

Imagen 19: Sensor TEMT6000 [17]

Se hicieron pruebas con este sensor y las fluctuaciones de medidas eran demasiado grandes. Por otra parte, al exponer el sensor a una fuente de luz grande saturaba rápidamente, debido a que el rango de medida en lux no era lo suficientemente grande.

4.3.1.4 CJMCU-3001 OPT3001

El sensor OPT3001 es un sensor de luz ambiental digital que se comunica mediante el protocolo I2C con el microcontrolador. Este sensor tiene una excelente apreciación de medida y su banda espectral va desde los 460nm a 655nm. Su rango de alimentación va desde 1.6 V a 3.6 V y su corriente en modo sleep es de 0.4µA.

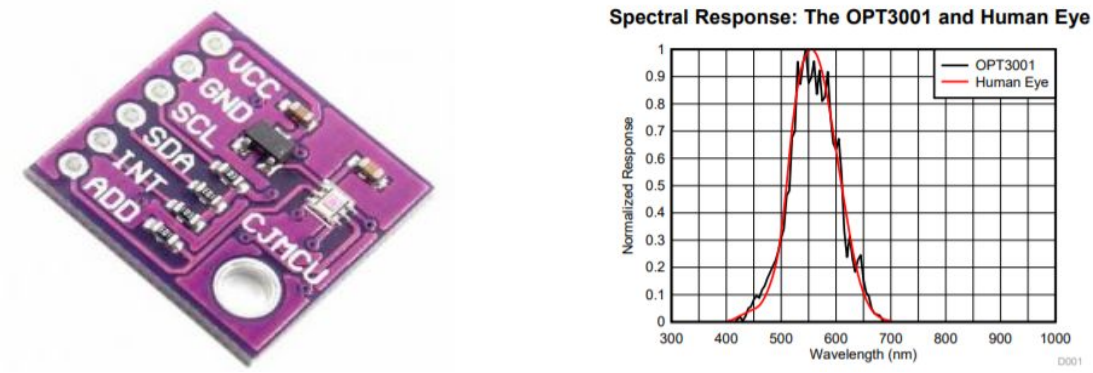

Imagen 20: Sensor OTP3001 y gráfico de respuesta en frecuencia [18]

Durante las pruebas realizadas, este sensor fue el más “estable” en las mediciones y el que mejor respondía a los requerimientos de la aplicación desarrollada. Esto se puede visualizar en el siguiente gráfico.

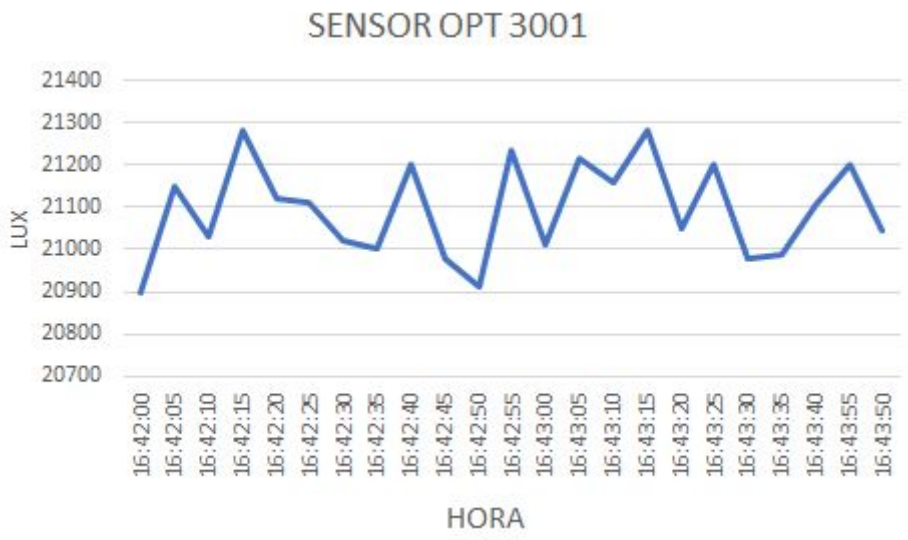

Imagen 21: Sensor OTP3001

Sin embargo, presenta un elevado costo en comparación a los otros sensores, aproximadamente 5 USD.

4.3.1.5 BH - 1750

El sensor de luz ambiente BH1750, se comunica digitalmente mediante el protocolo I2C. Su rango de alimentación va desde los 3V a los 5V y el consumo de corriente es menor a 1 µA en modo sleep. Cabe mencionar que cuando se realiza la medición cambia a un modo lectura, donde el consumo aumenta a aproximadamente 100 µA.

Su rango de medida va desde 1 lux a 65535 lux y puede obtener una apreciación de 0.5 lux. Por último su costo en el mercado es menor a 1 USD .

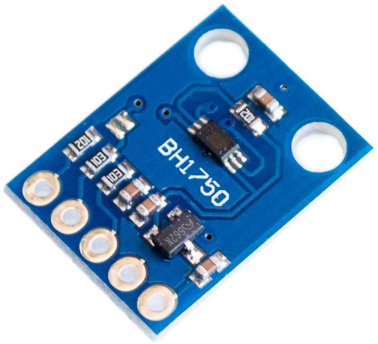

Imagen 22: Sensor BH1750 [19]

Durante las diferentes pruebas, el sensor se comportó de una manera aceptable, teniendo una fluctuación en la medida bastante buena en

comparación al resto. En el siguiente gráfico podemos visualizar las mediciones en condiciones de luz solar directa .

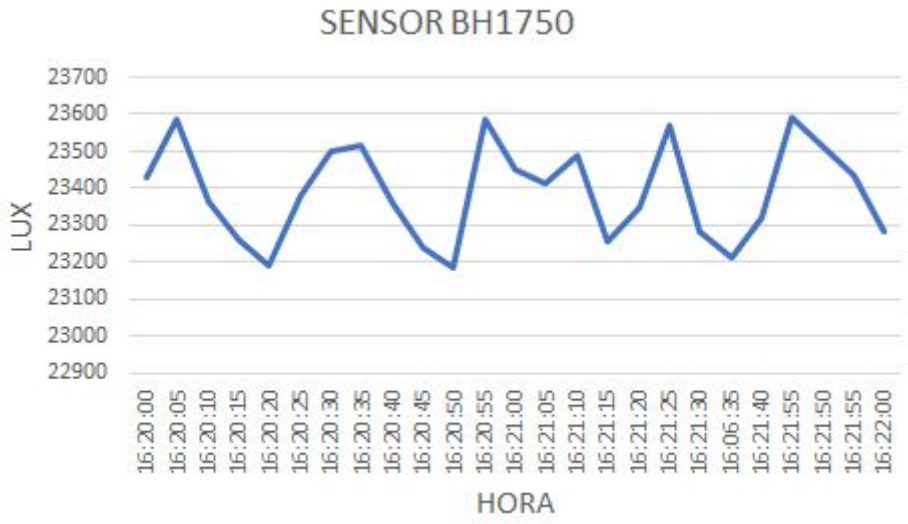

Imagen 23: Sensor BH1750

4.3.2 Análisis y resultados

Luego de realizar pruebas de campo a diferentes sensores se pudo observar que los que mejor respondieron fueron el BH1750 y el OPT3001. Como se puede visualizar en el siguiente gráfico, la fluctuaciones en las medidas de ambos fueron relativamente aceptable en comparación con el GY-49.

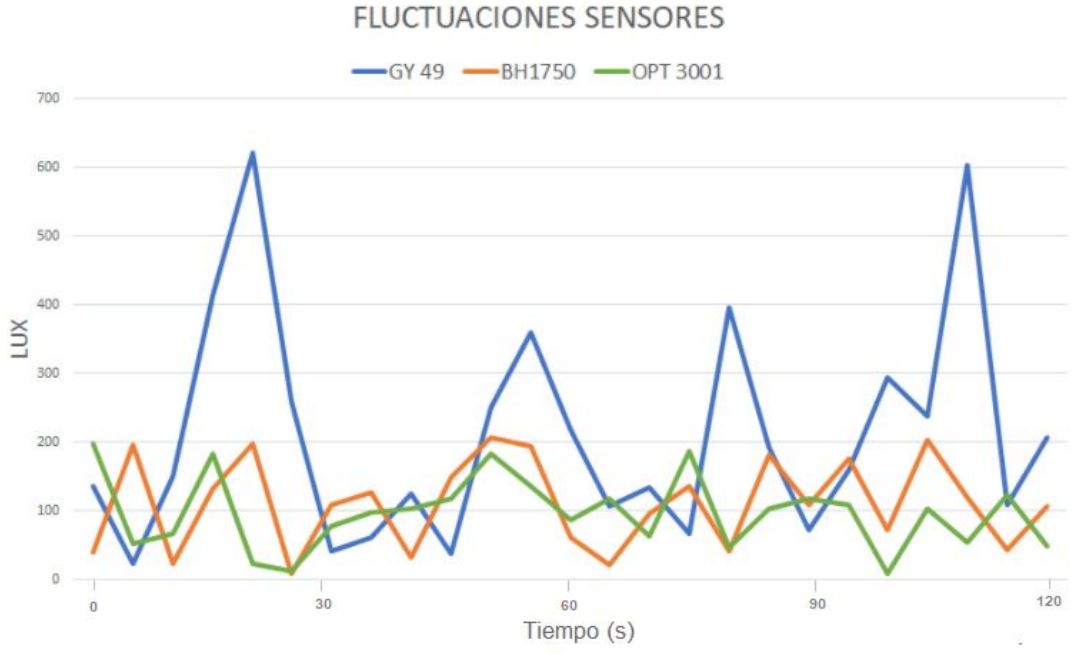

Imagen 24: Comparación de sensores en un lapso de 2 minutos.

Cabe mencionar que los valores medios de lux medidos por los diferentes sensores no son los mismos, debido a que no se hicieron al mismo tiempo generando que las condiciones de luz sean diferentes. Por otra parte la sensibilidad de cada sensor no era exactamente igual, es decir si se realizaban una medida al mismo tiempo con los tres sensores y en las mismas condiciones los valores de lux eran diferentes.

Los sensores analógicos también fueron descartados porque por defecto, el microcontrolador únicamente podía manejar una entrada analógica y debido al mejor funcionamiento de los sensores digitales por el protocolo I2C se optó por dejar esa entrada analógica libre para medir el nivel de la batería.

Para decidir finalmente entre qué sensor se adecuaba mejor a nuestro proyecto, se analizaron las diferentes características de cada uno, siendo el BH1750 más barato y con un consumo de corriente relativamente menor. Es relevante destacar que el costo del sensor es un parámetro de gran importancia en esta aplicación.

Sensor	OPT 3001	**BH 1750**	GY 49
Rango máximo (Lux)	83865	**65535**	188000
Resolución (Lux)	0.01	**0.5**	0.045
Consumo fijo (µA)	0.4	**1.0**	0.65
Costo (USD)	5	**1**	1
Desviación media (pruebas realizadas)	112	**131**	269

Tabla 3: Comparativa sensores digitales

Luego de realizarse las pruebas de campo a los diferentes sensores, se analizó cuál era la mejor forma para poder discernir si la luz estaba o no prendida.

En un principio se evaluó la posibilidad de utilizar dos sensores iguales donde uno captará la luz ambiente, y otro detectara la luz testigo. Luego, mediante una resta entre las mediciones se detectaría el estado de la luz. Sin embargo durante las pruebas se observó que debido a la sensibilidad de los sensores era imposible poder determinar el funcionamiento de las luces mediante este método.

Como alternativa, surgió la idea de utilizar un único sensor que solamente midiera la fuente de luz que se quería analizar y no se viera afectado por la luz solar o ambiente. Para poder lograrlo se colocó un tubo del diámetro del sensor de luz (3 mm aproximadamente) y con un largo variable.

Se construyó una primera maqueta para realizar las pruebas, con un microcontrolador, un sensor BH1750 y una luz testigo que simulaba la luz del aeropuerto pero con menor potencia. Con esta maqueta no solo se podía observar si el sol afectaba la medida sino que también cuál era la distancia y el ángulo óptimo que se debía colocar el sensor de la luminaria, para que funcionara lo mejor posible.

Imagen 25: Maqueta para pruebas.

En esta simulación se guardaba cada un minuto la medida del sensor de luz con y sin la luminaria prendida, posteriormente estos datos se procesaron y se realizaron gráficas para analizar los mismos.

4.4 Antenas

Para la implementación de la red se estudiaron distintos tipos de antenas tanto para el gateway como para los nodos. El objetivo del estudio es mejorar la potencia de transmisión y recepción en cada línea de comunicación, siendo que las antenas cumplen un rol muy importante en las comunicaciones inalámbricas.

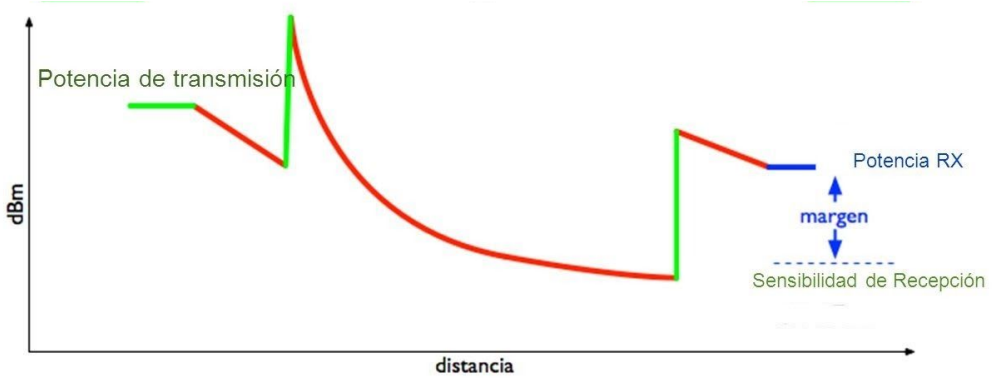

Imagen 26: Esquema de transmisión inalámbrica [19]

En la imagen anterior las líneas verdes verticales son las representaciones de las antenas en el gateway y el nodo. En esta se puede visualizar como aumenta tanto la potencia de transmisión como la de recepción a medida que se aumenta la ganancia de las antenas. Esto deriva a conexiones más estables y mayores distancias de comunicación.

Las antenas en los nodos presentan una limitación de tamaño no así la de los gateway, lo que hace que la elección de la antena en los nodos sea más acotada.

4.4.1 Antena gateway

Para la antena del gateway se utiliza la “860~930 MHz 3dBi LoRa Antena” de RAKWireless. En las siguientes imágenes se visualiza fotos y características de la misma.

Imagen 27: Antena 3dBi de RAKWireless [20]

Rango de frecuencias (Mhz)	860 - 930
Ganancia a 915 Mhz (dBi)	2.9

VSWR a 915Mhz	1.568
Eficiencia a 915 Mhz (%)	69.9
Impedancia (Ohms)	50
Polarización	Vertical
Radomo	Fibra de vidrio
Conector	Tipo N macho
Dimensiones (mm)	Φ25 x L360 ± 10

Tabla 4: Especificaciones antena 3dBi RAKWireles [20]

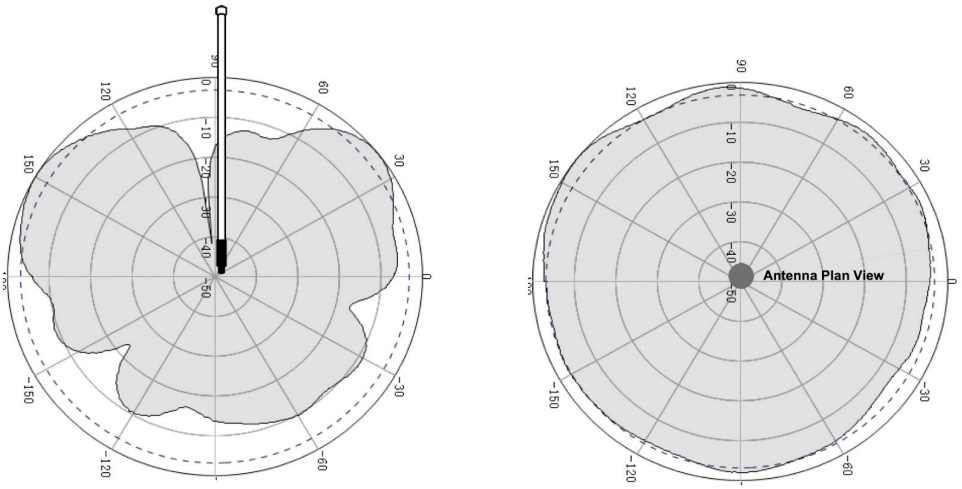

Imagen 28: Patron de radiacion antena 3dBi RAKWireless para 920Mhz [20]

4.4.2 Antena nodo

Para la antena del nodo se evaluaron dos tipos de antenas (ambas de 3dBi), una con un conector SMA exterior y otra con un conector iPEX. Esta última puede colocarse dentro del encapsulado fácilmente, debido a que posee un conector pequeño.

Para elegir estas antenas se toma en cuenta las limitaciones en tamaño, el que no puede superar los 5 cm. Finalmente a raíz de que ambas antenas mostraron un rendimiento similar, se opta por la que se puede colocar dentro del encapsulado, manteniendo de mejor manera la estanqueidad del nodo.

Imagen 29: Antena 3dBi conector SMA [21]

Imagen 30: Antena 3dBi conector iPEX [22]

4.5 Baterías

Debido a que el nodo debe funcionar con batería durante años se descarta la posibilidad de utilizar baterías secundarias (recargables), tomando en cuenta el hecho de que estas tienen una fuga de carga mayor al correr del tiempo.

Con respecto al tipo de batería primaria, una opción que se adapta muy bien a este tipo de proyecto son las baterías LiPo, en este caso éstas fueron descartadas por lo complicado que se hace la importación de las mismas. En cambio optó por pilas AAA, por el hecho de que se pueden adquirir fácilmente y tienen un bajo costo.

Para la alimentación se utiliza 3 pilas AAA en serie para de esta manera obtener un voltaje de 4.5 V, siendo esto suficiente para el funcionamiento del nodo.

5. Desarrollo del producto

Luego de definir el hardware que se utilizará, se continuó con el desarrollo y diseño del producto. El mismo se subdivide en cuatro partes principales: Hardware, Firmware, software y envolvente o encapsulado.

5.1 Hardware

En esta parte se explicará el diseño de los distintos componentes del proyecto, centrándose principalmente en el nodo, donde se encuentra el desarrollo principal.

Debido a que el hardware debe ser alimentado con batería primaria, se realizó su diseño en base a esa necesidad. En primer lugar se implementó un sistema de alimentación para que el microcontrolador funcione correctamente, posteriormente se realizó el circuito de la antena y el de los botones basándose en las recomendaciones del fabricante RAK [10].

Por otra parte se adaptó la línea de I2C y por último, se implementó un circuito de medición de voltaje de batería.

5.1.1 Alimentación

Teniendo en cuenta que el máximo voltaje tolerable por el microcontrolador es aproximadamente 3.5V, se debió reducir el voltaje de la batería y dejarlo en un valor estable y confiable. Para lograr esto último, se utilizó un regulador de voltaje a 3.3V. Este valor es el que fue utilizado en las pruebas preliminares realizadas en el desarrollo de este producto.

Para este tipo de proyecto se podía utilizar diferentes tipos de baterías primarias como por ejemplo de litio, lipo o alcalinas. Estas distintas tecnologías de baterías presentan celdas desde 1.2V a 4.2V, por lo que se decidió colocar un regulador con un voltaje máximo de 5.5V, para que funcione correctamente con cualquiera de ellas.

Según la hoja de datos, el microcontrolador en modo sleep tiene un consumo en el orden de 2μA y en transmisión puede tener un consumo de hasta 120mA.

Con todas estas características se decidió utilizar un regulador lineal S-1313 en SOT 23-5 de 3.3V. El mismo tiene un voltaje de dropout de 0.32 V (a 100mA) con un consumo fijo cercano a 1μA y una corriente máxima de 200 mA. Es un regulador de tamaño pequeño y tiene un costo inferior a un dólar.

Para su utilización se implementó el circuito de la siguiente imagen (Imagen 32), añadiendo capacitores filtro tanto en la entrada como en la salida. Además se planificó colocar capacitores adicionales en caso de ser necesario. A su vez se conectó el pin “enable” del regulador al voltaje de entrada a través de una resistencia para garantizar su encendido.

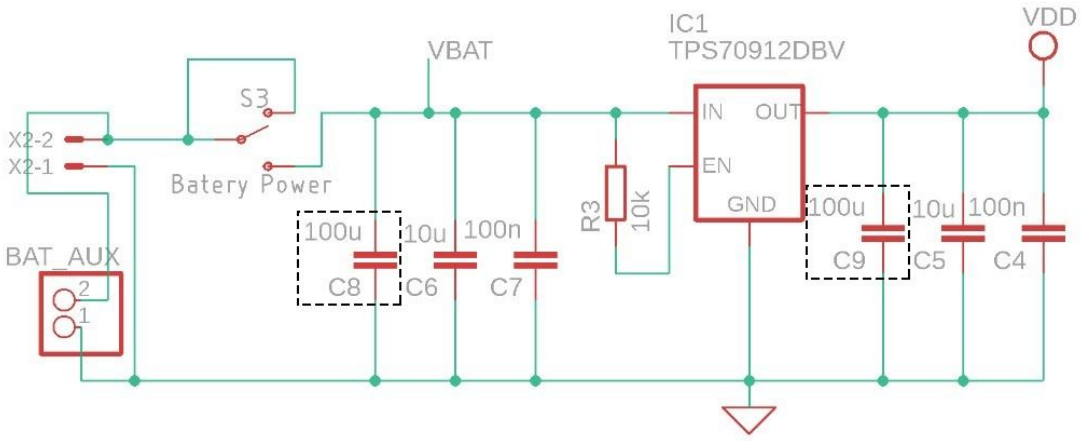

Imagen 31: Circuito diseñado para fuente de alimentación

Es importante mencionar, que los capacitores opcionales pueden llegar a extender la vida útil de la batería, filtrando el pico de consumo en transmisión. De esta manera se logra mantener el voltaje de encendido en un valor superior al voltaje mínimo del procesador, evitando el reinicio por bajo voltaje.

5.1.2 Antena

Para realizar el circuito de la antena se utilizó la recomendación de RAK, que consiste principalmente en la utilización de un capacitor de adaptación de 47pF y diodos de protección. Para este prototipo utilizamos la antena de RAK, previamente descrita en la sección 4.4 Antena. Este circuito se adapta adecuadamente a la antena elegida.

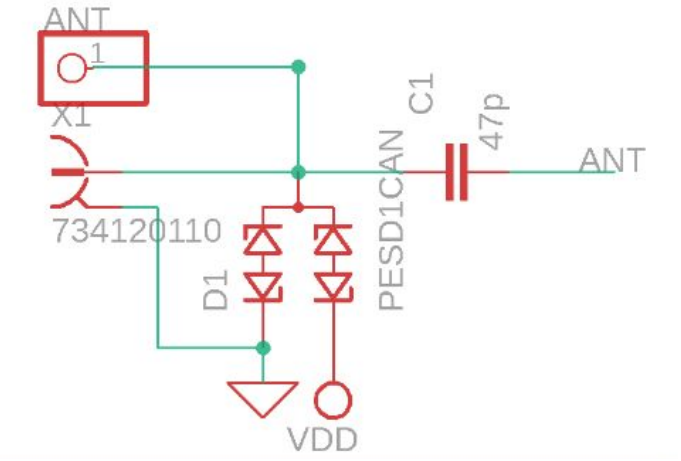

Imagen 32: Circuito diseñado para la antena.

Cabe mencionar que a nivel de PCB se previó un agujero para poder soldar otra antena en caso de no utilizar el conector SMA.

5.1.3 Sensores

El sensor de luz seleccionado (BH1750) utiliza el protocolo de comunicación I2C. Para alimentar el sensor se implementó un circuito con un transistor PMOS, controlando el encendido desde el microcontrolador. De todas formas, se dejó prevista una salida o entrada digital, para poder medir el estado del sensor o en caso de ser posible encenderlo directamente desde el RAK4260, evitando de esta manera la utilización de un transistor.

Además se colocó un “jumper” entre el sensor y la alimentación para poder simular que se apaga y estudiar la recuperación del sensor ante una falla esporádica.

Al utilizar el sensor BH1750 es necesario forzar la salida ADDR a tierra, ya que ésta determina el número de esclavo en función de su estado. A su vez la línea de comunicación I2C recomienda la colocación de resistencias de pull up, tal como se puede visualizar en el siguiente diagrama.

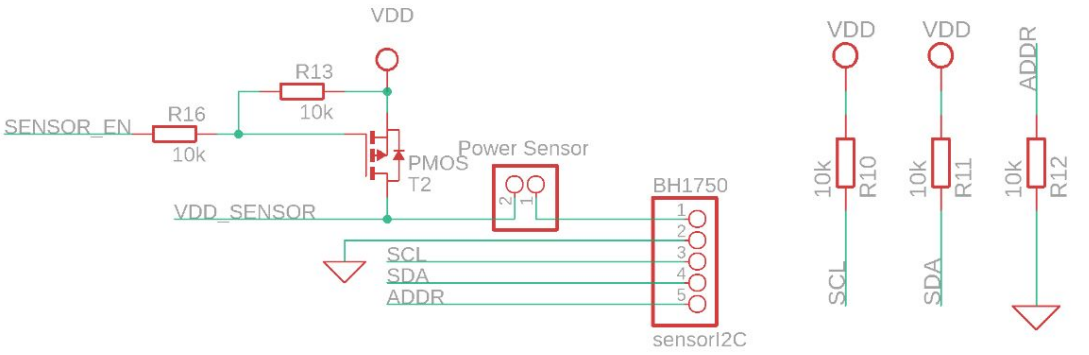

Imagen 33: Circuitos de control de sensor BH1750 y resistencias de pull up del I2C

5.1.4 Medición de batería

Con el fin de conocer y comunicar el estado de la batería se decidió implementar un circuito de medición de batería. Como las entradas analógicas son capaces de leer hasta un voltaje de 3V, se utilizó un divisor resistivo, permitiendo medir hasta 6V. Sin embargo, su implementación sin usar resistencias de valores muy grandes, lleva a un consumo fijo importante a largo plazo, por lo que se colocó un transistor PMOS para habilitar el circuito solamente cuando es necesario realizar la medición.

El RAK4260 utilizado posee un conversor analógico digital de 12 bits con un voltaje de referencia configurable. En esta aplicación se configuró en 3V, teniendo una resolución de 0.74 mV.

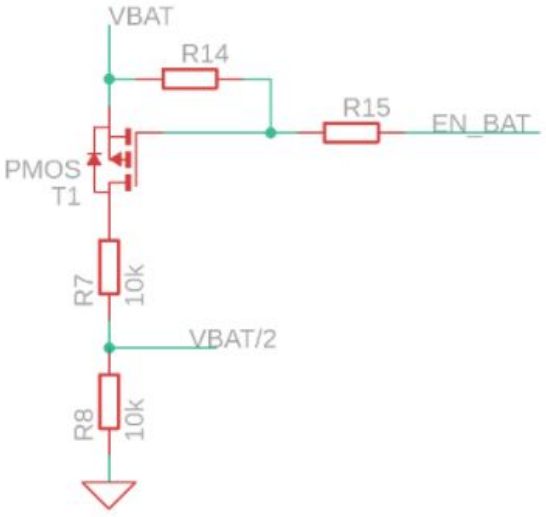

Imagen 34: Circuito de medición de batería

En las pruebas de consumo se observaron inconvenientes al apagar el transistor de medición de batería. Durante la misma el voltaje se redujo desde 4.5V hasta llegar asintóticamente a 3.3V, como se puede observar en la siguiente gráfica:

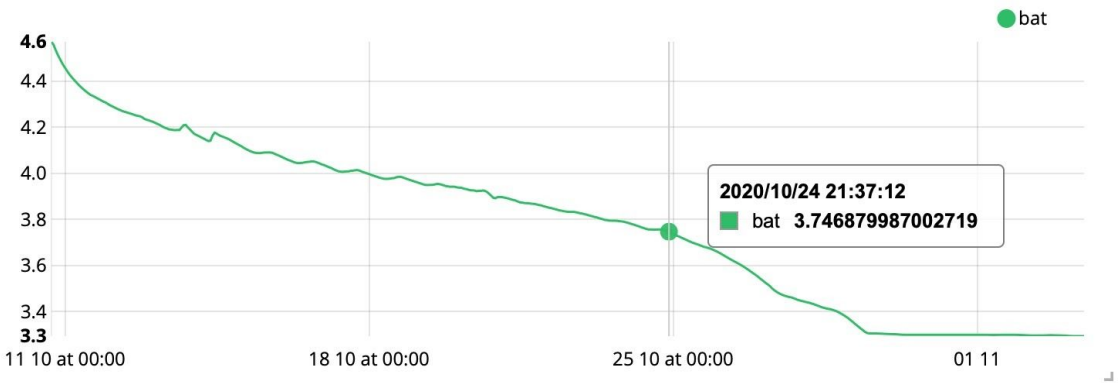

Imagen 35: Curva de descarga batería en función de los mensajes enviados.

Este error en la medida fue producto de abrir el transistor colocando el gate del mismo en 3.3V. Luego de analizar el problema se observó que no se había considerado que el voltaje de source del transistor llega hasta 4.5 V, produciendo una diferencia de 1.2V entre el source y el gate del transistor, generando de esta manera su autoencendido. Por este motivo al momento de descender el voltaje de batería hasta 3.3V el transistor abrió el circuito, produciendo el estancamiento en este valor.

Para solucionar este inconveniente se deberá colocar otro transistor NMOS con un pull up como se muestra en la figura.

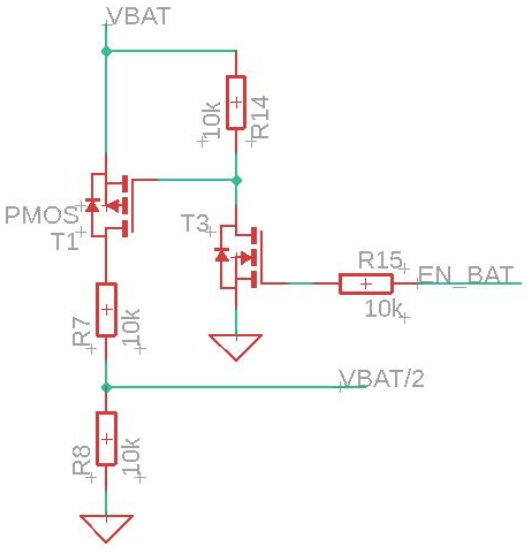

Imagen 36: Segunda versión circuito de medición de batería

En esta nueva configuración, la señal de mando que activa y desactiva el circuito de medición de batería quedará invertida, ya que llevar el transistor NMOS a 3.3V lo activa, forzando un voltaje de 0V en el gate del PMOS, lo que permite la medición de batería. En caso contrario, al desactivar el NMOS colocando su gate a 0V, el gate del PMOS subirá hasta el voltaje de batería, cortando la corriente del divisor resistivo. De esta manera se eliminaría el consumo fijo.

5.1.5 Circuitos auxiliares

Con el objetivo de poder realizar debug o alguna posible función de configuración o instalación, se dejó prevista la colocación de tres leds y un botón para interactuar con el microcontrolador. Por otra parte, se previó la colocación de un botón de reset tal como se observa en la siguiente figura.

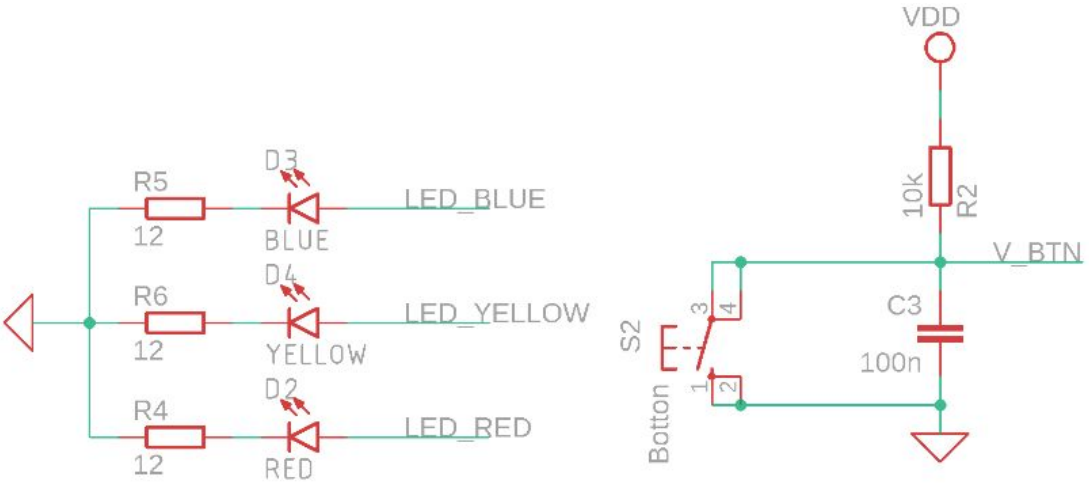

Imagen 37: Circuitos de los leds y de los botones

5.1.6 PCB

Durante la ejecución del proyecto se realizaron dos versiones de PCB, una preliminar para poder realizar pruebas y ver el comportamiento de los diferentes componentes. Luego de encontrar diferentes errores y correcciones se realizó la segunda versión del PCB, siendo ésta la utilizada como prototipo final. Para el diseño del PCB se utilizó el programa Eagle de Autocad con versión estudiantil.

5.1.6.1 Versión 1

Teniendo en cuenta las recomendaciones de RAK y los requerimientos estudiados, se procedió a hacer un primer circuito esquemático. Éste consistió en hacer un PCB relativamente grande y variado como para realizar las diferentes pruebas de funcionamiento, performances de sensores, etc.

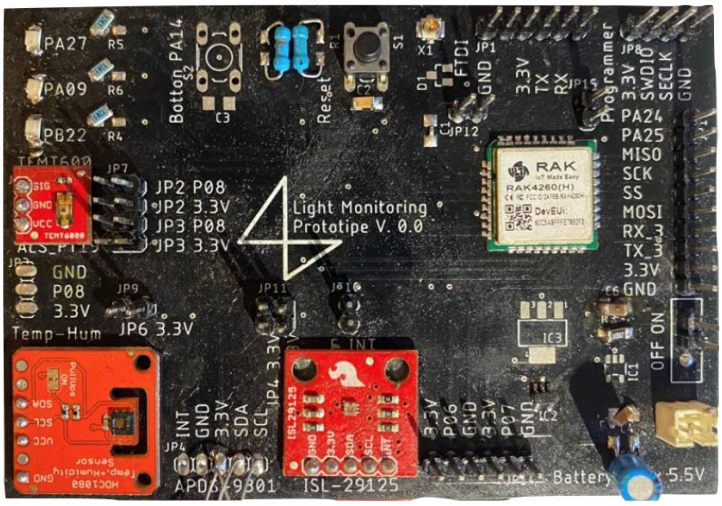

Imagen 38: Foto del primer PCB armado

En esta primera versión se encontraron varios inconvenientes o errores en el funcionamiento de la placa. El primero de ellos fue que faltaron las resistencias de pull up necesarias en el protocolo I2C, para la comunicación de los sensores de luz con el microcontrolador.

Por otra parte, a través de esta versión del PCB surgieron nuevas ideas y posibles mejoras que se podían realizar. Éstos son el circuito de medición de batería y un nuevo circuito con la capacidad de poder controlar sensores (ya sea directamente desde una salida digital o a través de un transistor).

También se encontraron problemas de reinicios imprevistos en baterías descargadas, para solucionarlo se agregaron capacitores más grandes. A su vez en este circuito los capacitores de alimentación estaban innecesariamente lejos del microcontrolador, por lo que fueron reubicados para la próxima versión.

5.1.6.2 Versión 2

En base a los conocimientos aprendidos y los errores cometidos en la primera versión, se procedió a hacer la segunda. En esta se redujo drásticamente el tamaño y se acercaron capacitores hacia la fuente de alimentación. También se le hicieron las mejoras mencionadas anteriormente

de agregar el circuito para medición de batería y un transistor de mando del sensor lumínico.

De todas formas se colocaron varios componentes de más por si llega a surgir algún inconveniente. Por ejemplo, dejar el “footprint” de capacitores, resistencias y transistores mencionados en la partes anteriores de 5.1 Hardware.

Por otra parte el sensor óptico BH1750 era necesario que se encontrara en otro plano, por este motivo se dejó previsto un lugar para poder soldar posteriormente este sensor en el PCB.

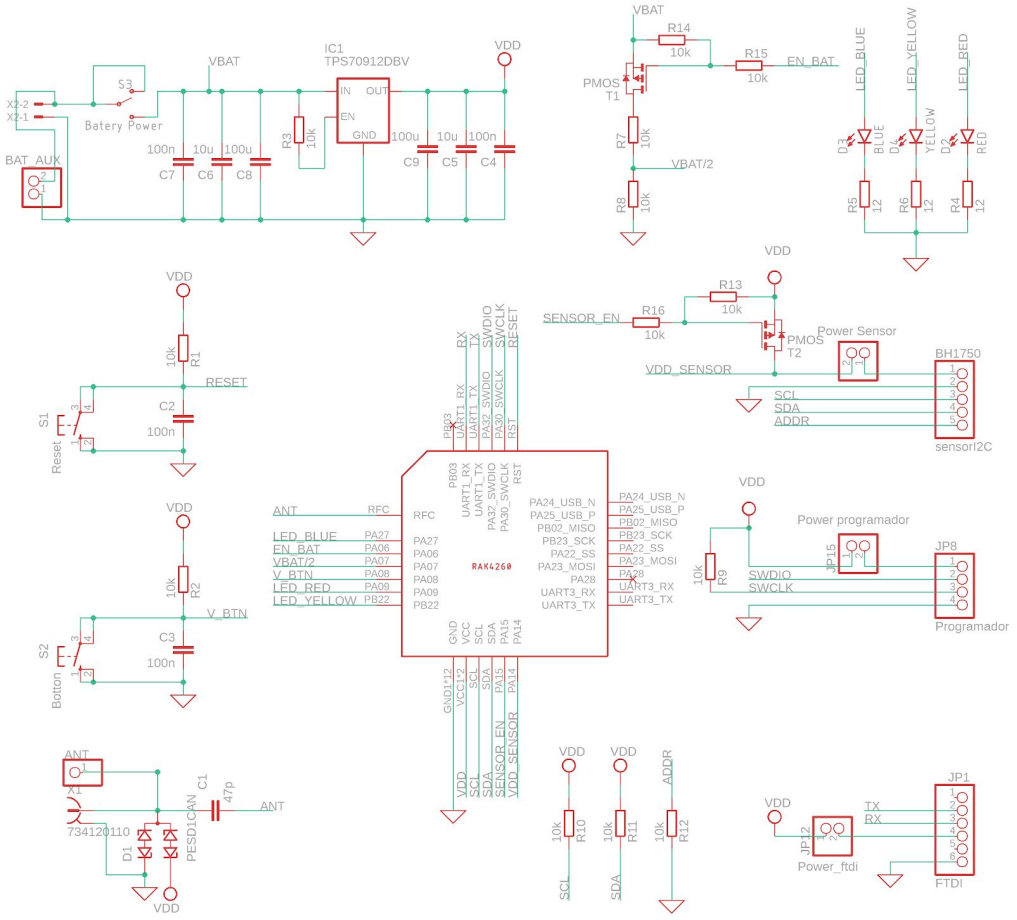

Imagen 39: Circuito esquemático del nodo

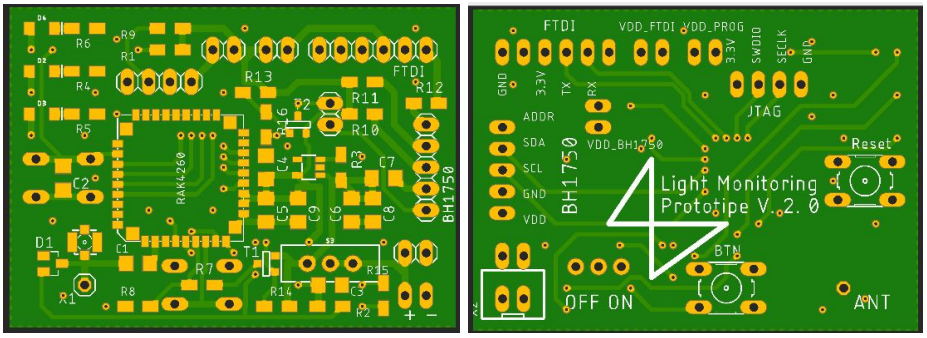

Imagen 40: Layout del PCB del nodo

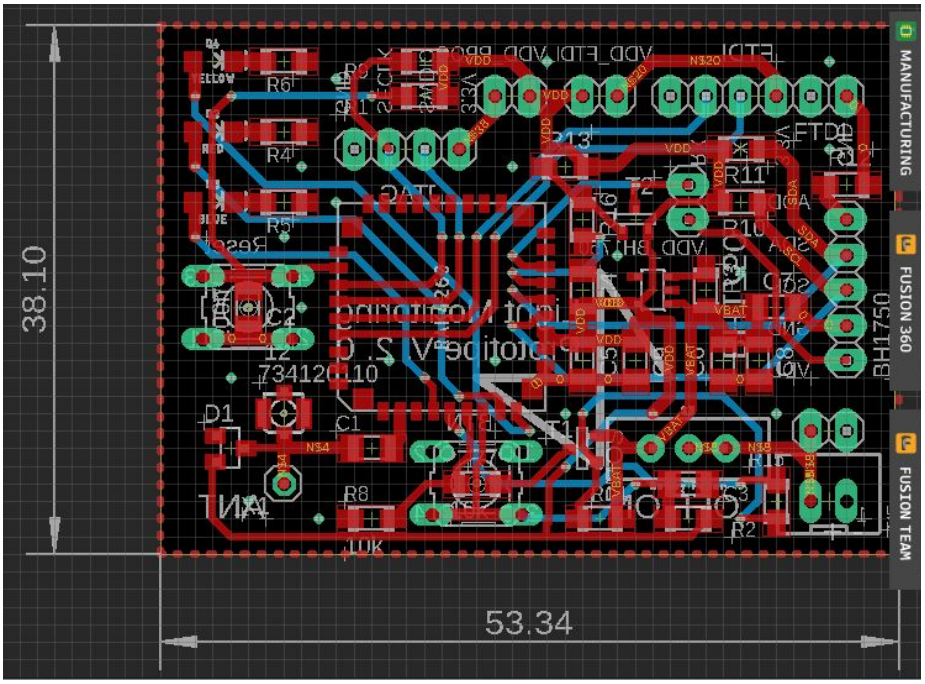

Imagen 41: Captura del programa Eagle del layout del PCB del nodo

La segunda versión del PCB final tiene un tamaño de 53.3mm de largo y 38.1mm de ancho, por lo que se puede considerar pequeño. Esto permite que el tamaño final del producto sea relativamente chico.

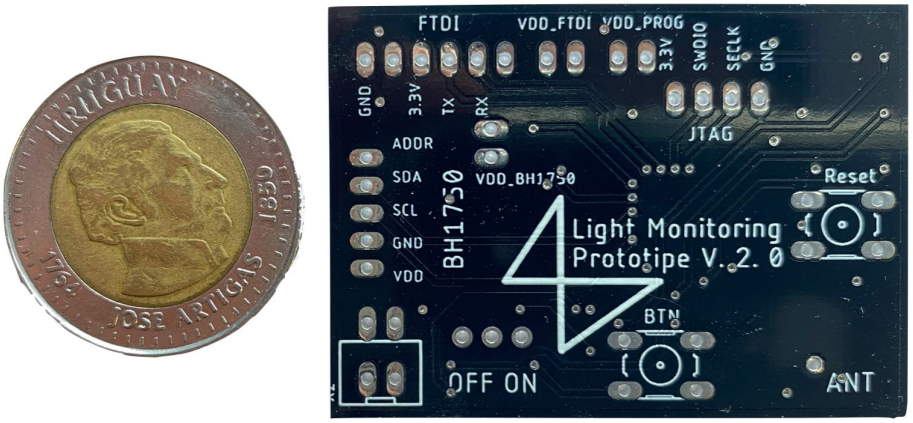

Imagen 42: PCB del nodo de frente sin componentes.

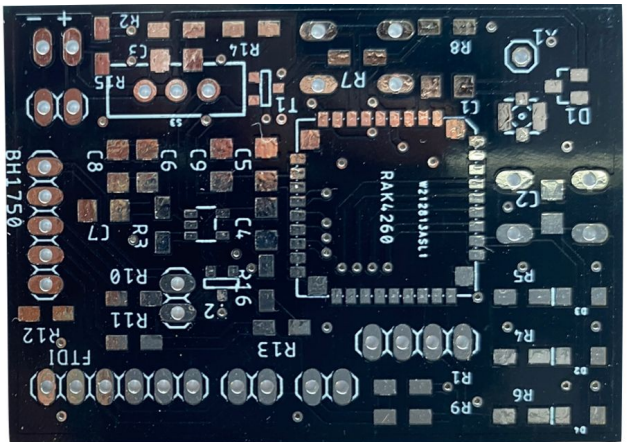

Imagen 43: PCB del nodo de atrás sin componentes.

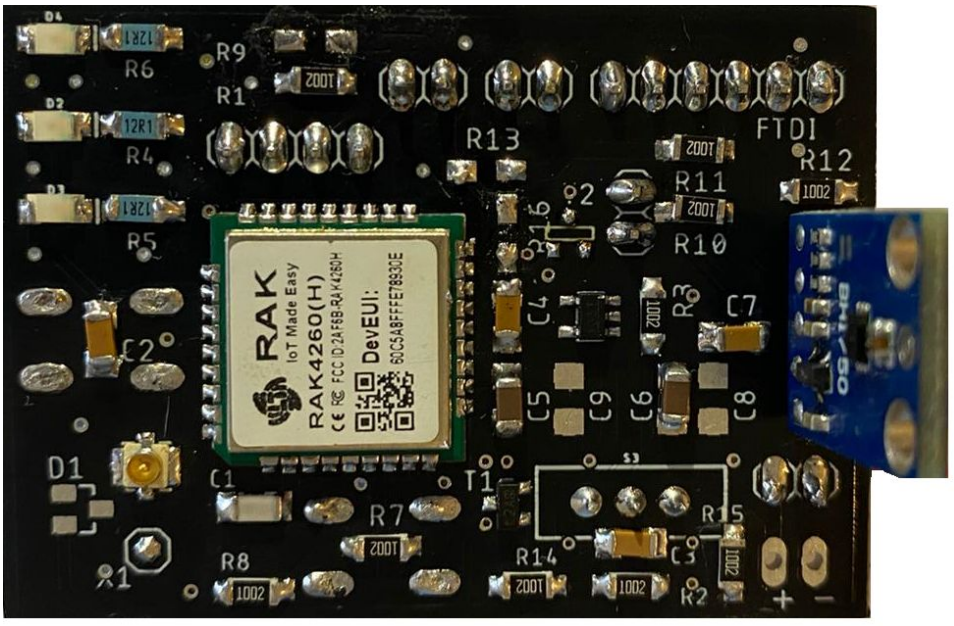

Imagen 44: PCB con los componentes soldados.

5.1.6.3 BOM

En la segunda Versión del PCB la BOM está compuesta por los siguientes elementos:

Componente	Valor	Cantidad
Resistencias 1206	10kΩ	10
Resistencias 1206	5.1Ω	3
Botones		2
Interruptor	2 posiciones	1
Transistor sot 23-3	PMOS baja fuga	1
Capacitor 1206	100nF	4
Capacitor 1206	10µF	2
Capacitor 1206	47pF	1
Regulador sot 23 - 5	3.3V bajo corriente de fuga	1
Leds 1206	Roja, azul y amarilla	3
Diodo protección		1
BH1750	Sensor lumínico	1
RAK4260	Transceptor LoRa	1

Tabla 5: BOM Version 2.

5.2 Firmware

En esta sección del proyecto, se presenta el firmware desarrollado en el nodo. Esto se podría explicar como el programa y las estructuras que definen el comportamiento del microcontrolador. Se utiliza un entorno de programación, que mediante un lenguaje específico se escriben diferentes instrucciones en el microcontrolador, generando de esta manera el comportamiento deseado. En

este proyecto y para programar el microcontrolador RAK4260 se utilizó el entorno Atmel studio 7.0.

El nodo no realiza tareas de gran complejidad, ya que simplemente cumple la función de sensor de telemetría. Sin embargo, debido a que el procesador es relativamente moderno, se generaron diferentes inconvenientes a la hora de programarlo.

El firmware del nodo en esta aplicación, se divide en tres tareas principales:

- Medir luminosidad del sensor de luz
- Medir voltaje de batería
- Enviar datos obtenidos al gateway principal

5.2.1 Flujo principal de programa

El flujo principal del programa muestra en un simple diagrama las acciones que toma el microcontrolador cíclicamente.

El nodo consta con un flujo de programa que inicia por identificarse ante TTN, asegurando la conexión a la red. En caso de no tener alcance y no poder conectarse lo volverá a intentar cada 30 segundos hasta conseguirlo. Luego de establecida la conexión, el nodo no necesitará volver a identificarse más.

Después de unirse a la red, comenzará a realizar su rutina principal. Esta comenzará cambiando el sensor de luz BH1750 a “modo lectura” durante el tiempo mínimo necesario para poder realizar la medida. Luego de obtener la luminosidad el sensor vuelve a “modo sleep”.

Siguiendo con la rutina, se prende el circuito de medición de batería del nodo y se obtiene su valor de voltaje. Posteriormente se deshabilita este circuito para disminuir el consumo de energía.

Por último, teniendo estos datos guardados, los envía a través de LoRa al servidor y luego el microcontrolador vuelve a entrar en modo sleep. A continuación se muestra un diagrama de flujo representando la rutina del nodo.

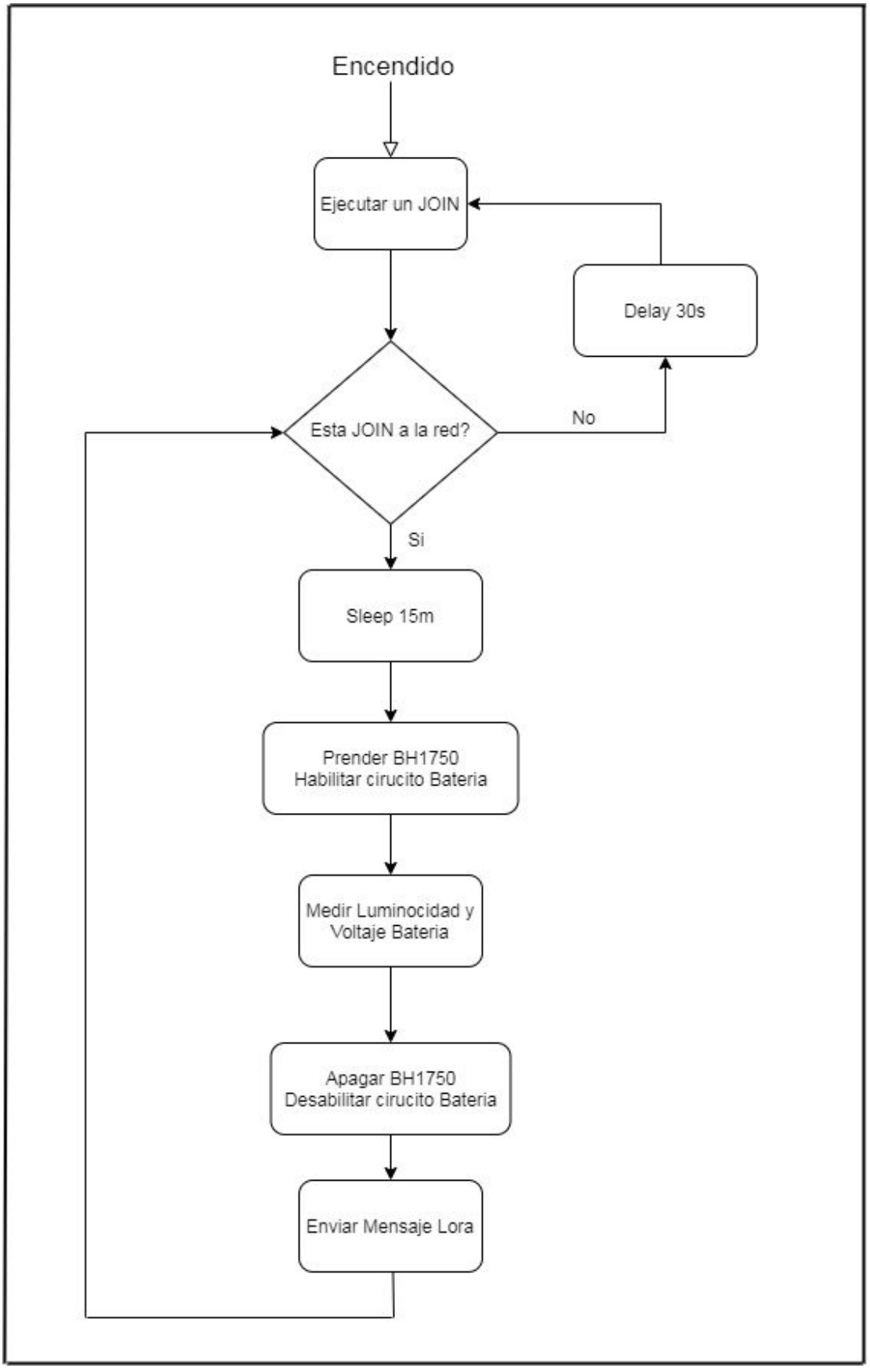

Imagen 45: Diagrama del flujo principal del programa

5.2.2 Características del Firmware del Microcontrolador

El RAK4260 tiene en su interior un SAMR34 con un puerto JTAG. Este puerto es únicamente de programación y está formado por dos pines de comunicación SWCLK y SWDIO. Por otra parte, este microcontrolador consta de puertos seriales con el fin de poder hacer tareas de debug y comunicación en general. En este proyecto particularmente se utilizó un J-link de Segger que ya viene previsto para poder utilizarse con Atmel studio 7.0 siendo este último el entorno de programación utilizado en el microcontrolador RAK4260. Este IDE consta con la posibilidad de hacer debug en el código mientras corre, permitiendo realizar break points y poder observar el flujo del código.

Imagen 46: Programador Jlink y logo del IDE atmel studio [23]

Para la programación del RAK4260 se utilizó como base un código de ejemplo previsto por RAK. Este código base no presentaba buenas prácticas en su desarrollo, como por ejemplo, la falta de linealidad del código que activaba tareas nuevas dentro de otras sin poder dar seguimiento al flujo del programa.

También por la falta de utilización de estructuras como “while” siendo reemplazadas por bucles autorentrantes. Esto provocó que sea una tarea muy compleja entender el funcionamiento del programa y la edición del mismo.

Por otra parte, al ser un microprocesador tan nuevo no se dispone de otro código de donde partir con la parte principal de LoRa implementada, por lo que se decidió utilizar el código ejemplo genérico a pesar de tener que destinar mucho tiempo a simplificar y eliminar las funciones que no eran necesarias.

Actualmente, existen versiones piloto de librerías en el IDE de Arduino para poder programarlo. Esto podría facilitar códigos de ejemplo y librerías más simples permitiendo acelerar el desarrollo al utilizar este microcontrolador.

El RAK4260 permite la utilización de multitarea, lo que es necesario para las funciones de LoRa, de esta manera recibir mensajes al mismo tiempo que se continúan otros procesos.

5.2.3 I2C

I2C es un protocolo de comunicación serial síncrono capaz de manejar varios esclavos y albergar varios maestros, consta de 2 líneas de comunicación SDA (Serial Data) y SCL (Serial Clock).

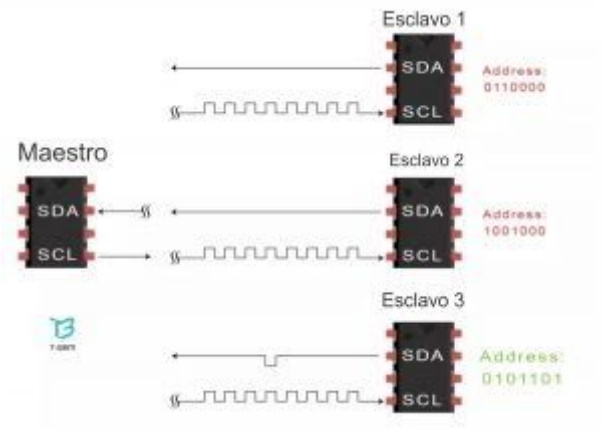

Imagen 47: Diagrama de comunicación I2C [24]

Existen varias configuraciones habituales de velocidad, desde 100 kbps a 5Mbps. En este proyecto no hay una necesidad de velocidad alta por esta razón se utiliza la mínima 100kbps.

5.2.3.1 Composición de mensaje

Los mensajes de este protocolo comienzan con un flanco de bajada en la línea SDA mientras en SCL cambia de alto a bajo. Luego de eso se envía el número de esclavo, es decir la dirección donde se quiere solicitar o enviar información. Posteriormente aparece el bit de lectura o escritura, seguido de un bit de comprobación.

A continuación se agrega la información, ya sea uno o dos bytes, finalizando cada uno con un bit de comprobación. Para terminar el mensaje, se pasa la señal SDA de un nivel bajo a un nivel alto mientras SCL cambia de bajo a alto.

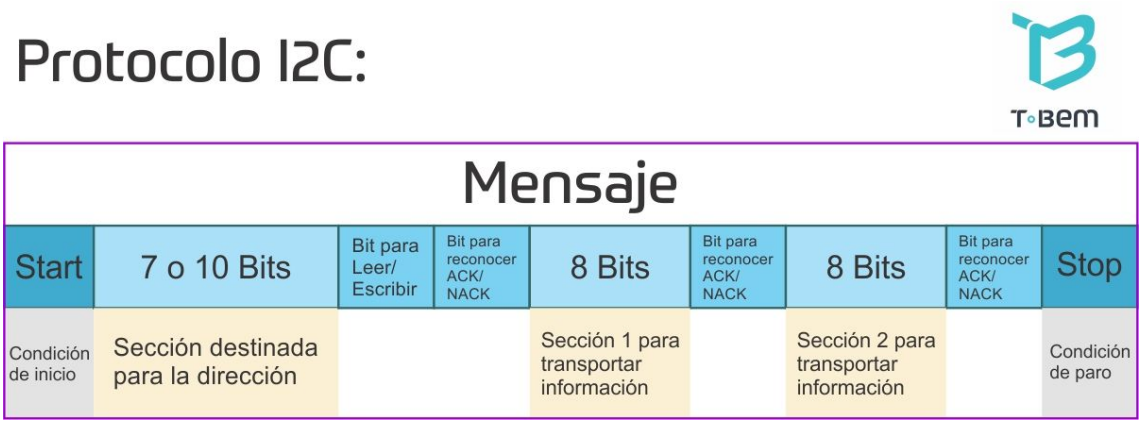

Imagen 48: Composición del mensaje I2C [24]

Dentro de estos atributos del mensaje los más importantes son: la dirección de esclavo (identifica a cada esclavo), el bit de lectura o escritura (permite indicar que se quiere hacer con el esclavo, ya sea leer o escribir uno de sus registros), el mensaje enviado desde el maestro contiene en la información el número de registro con el que se quiere interactuar.

5.2.3.2 Comunicacion RAK con sensor de luz

El sensor BH1750 puede escoger entre dos valores de esclavo dependiendo del estado del pin ADDR. Si el voltaje en este pin es superior a 70% de la alimentación la dirección sería 0x5C y en caso de ser inferior al 30% sería 0x23.

5.2.3.3 Configuracion BH1750

Por otra parte el BH1750 posee distintas configuraciones de sensado de luz, ya sea para modos nocturnos o a pleno sol del día. Cada uno de estos modos tienen distinta sensibilidad y error asociado. En este desarrollo se optó por utilizar la configuración de menor resolución y más rápida (codigo 0b00010000), con un error máximo de cuatro lux en la medida y un tiempo aproximado de 25 ms para obtener la información.

5.2.3.4 Lectura I2C del sensor BH1750

Para poder leer el sensor se envía la solicitud por I2C utilizando la dirección 0x23 y el microcontrolador espera la respuesta de dos bytes, donde el primero es el más significativo y el segundo el menos significativo. Luego de enviar la respuesta el sensor se pone en modo reposo, reduciendo su consumo a 1µA según su hoja de datos.

Instruction	Opecode	Comments
Power Down	0000_0000	No active state.
Power On	0000_0001	Waiting for measurement command.
Reset	0000_0111	Reset Data register value. Reset command is not acceptable in Power Down mode.
Continuously H-Resolution Mode	0001_0000	Start measurement at 1lx resolution. Measurement Time is typically 120ms.
Continuously H-Resolution Mode2	0001_0001	Start measurement at 0.5lx resolution. Measurement Time is typically 120ms.
Continuously L-Resolution Mode	0001_0011	Start measurement at 4lx resolution. Measurement Time is typically 16ms.
One Time H-Resolution Mode	0010_0000	Start measurement at 1lx resolution. Measurement Time is typically 120ms. It is automatically set to Power Down mode after measurement.
One Time H-Resolution Mode2	0010_0001	Start measurement at 0.5lx resolution. Measurement Time is typically 120ms. It is automatically set to Power Down mode after measurement.
One Time L-Resolution Mode	0010_0011	Start measurement at 4lx resolution. Measurement Time is typically 16ms. It is automatically set to Power Down mode after measurement.
Change Measurement time (High bit)	01000_MT[7,6,5]	Change measurement time. ※ Please refer "adjust measurement result for influence of optical window."
Change Masurement time (Low bit)	011_MT[4,3,2,1,0]	Change measurement time. ※ Please refer "adjust measurement result for influence of optical window."

※ Don't input the other opecode.

Imagen 49: Instrucciones y modo de funcionamiento sensor BH1750 [25]

5.2.3.4 Implementacion procolo I2C en el microcontrolador

Para implementar y configurar el protocolo I2C en el RAK4260, se utilizó la librería “I2C master”. En esta se editaron diferentes atributos de la estructura “I2C_master_config”, para poder lograr la comunicación con el sensor BH1750 como por ejemplo el baud_rate, generator_source, buffer_timeout, entre otros.

Luego de tener configuradas las características de la comunicación, es simplemente enviar y recibir paquetes con las funciones de la librería.

5.2.4 Sleep mode

En este tipo de aplicaciones donde se busca optimizar el rendimiento de batería al máximo, es necesario que el microcontrolador que se utilice sea de muy bajo consumo. Para lograr esto el RAK4260 presenta dos modos de sleep.

Por un lado, en el “modo sleep standby” el microcontrolador apaga todas las salidas digitales y el transceptor LoRa, tanto en recepción como transmisión. Además conserva los valores de memoria RAM sin utilizar la Flash, permitiendo al salir del modo sleep volver al flujo del programa a través de la función “wake up”.

Por otro lado, en el “modo sleep backup” se pierden los valores que contiene la memoria RAM. A su vez se apaga el LDO interno dejando en funcionamiento solamente la interrupción para volver a encenderlo. Al salir de este modo el microcontrolador se iniciará de la misma manera que un reset.

En la siguiente tabla se comparan y analizan las principales características de estos dos modos.

	STANDBY	BACKUP
Consumo	4 µA	2.5 µA
Pérdida datos RAM	No	Si
Transmisor LoRa	Apagado	Apagado
Salidas digitales	Encendido	Apagado
LDO interno	Encendido	Apagado

Tabla 6: Comparación de modos de sleep.

Luego de analizar los dos modos se decidió utilizar el “sleep mode standby”, ya que a pesar de tener un consumo superior no genera grandes cambios en la vida útil de la batería. Por otro lado este modo permite que no se pierdan los valores en la memoria RAM y no tener que volver a realizar el “join” facilitando el funcionamiento del código.

De todas formas se podría llegar a implementar el otro modo sleep utilizando el PDS, que es una memoria flash que se utiliza para almacenar los valores importantes de LoRa, evitando de esta forma volver a realizar el join en cada reset.

5.2.5 Debug

Las tareas de “Debug” son aquellas que permiten encontrar y eliminar errores en el funcionamiento de un programa, así como también analizar su comportamiento para poder mejorar la funcionamiento.

En este desarrollo se realizó “debugs” en tres puntos importantes, por un lado para poder comunicarse con el microcontrolador y confirmar la realización de los procedimientos correctamente. Por otro lado, para analizar la comunicación de los sensores mediante I2C y por último para controlar la comunicación mediante LoRa.

5.2.5.1 Análisis funcionamiento del código

Para poder probar el funcionamiento del código y verificar que el mismo estaba realizando las tareas programadas, se utilizó la herramienta Hércules.

Esta sirve para establecer una comunicación con el microcontrolador a través del puerto serial, utilizando un conversor FTDI.

Gracias a la comunicación en ambos sentidos con el RAK4260 se realizaron las primeras pruebas para conocer de una manera simple lo que estaba haciendo el microcontrolador en cada momento.

5.2.5.2 Interpretación mensaje I2C

Otro punto importante de análisis fue la configuración de los sensores de luz. Para esto se colocó un intérprete del protocolo I2C que era capaz de capturar las tramas de los mensajes enviados y recibidos entre el microcontrolador y los sensores de luz.

Al examinar estos mensajes se pudo identificar parámetros como la velocidad de conexión, los bit de parada, entre otros. Esto permitió ayudar a la programación y a la configuración del sensor de luz utilizado (BH1750).

En la siguiente imagen se puede ver una captura de un mensaje con el sensor ISL-29125, el mismo fue el primer sensor utilizado para las pruebas.

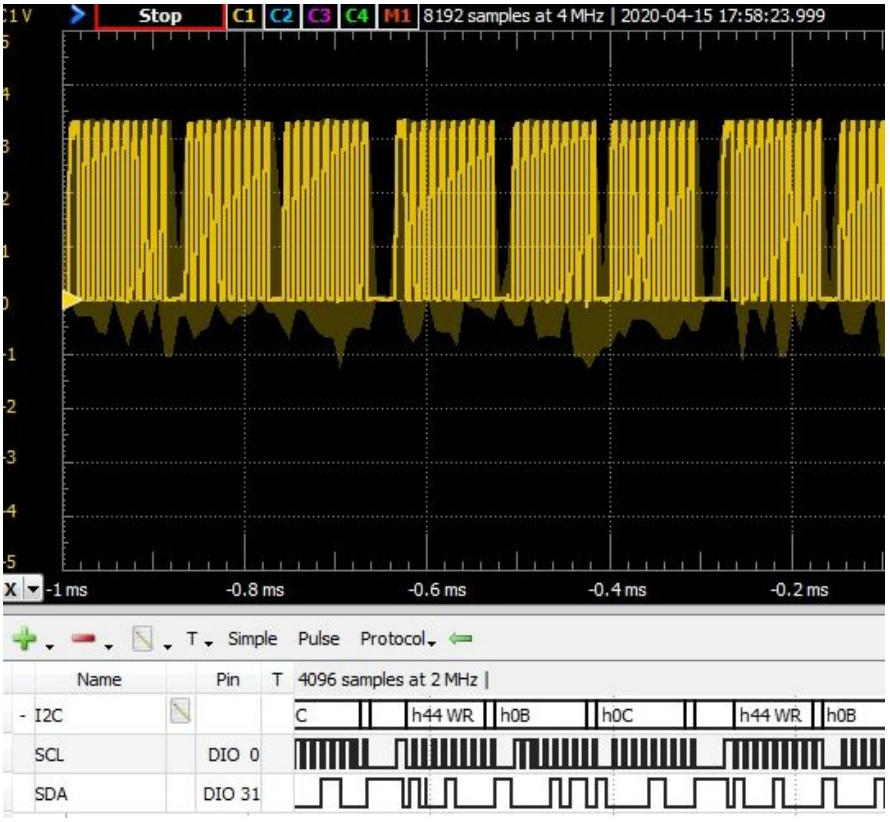

Imagen 50: Intérprete de I2C en waveforms

5.2.5.3 Testeo mensaje LoRa

El tercer punto donde se realizaron tareas de debug, fue para el testeo de la recepción de los mensajes LoRa. Para esto se utilizó la consola de TTN visualizando tanto el tráfico del gateway como el historial de mensajes desencriptados

En la siguiente imagen se puede ver mensajes de join y respuesta del gateway en un análisis de tráfico de TTN.

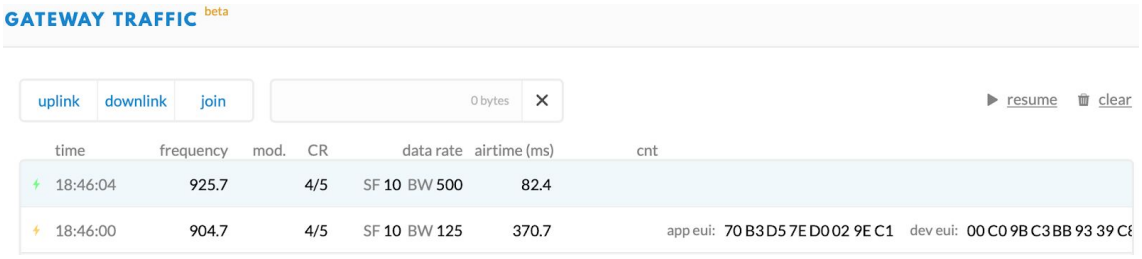

Imagen 51: Captura de Mensajes de join en TTN [26]

En la siguiente figura, se pueden visualizar los mensajes decodificados por la aplicación diseñada para este proyecto en TTN.

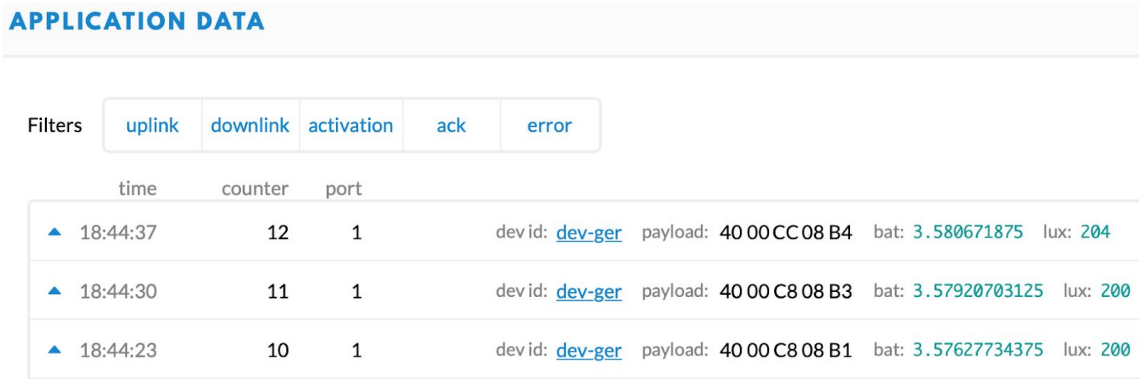

Imagen 52: Captura de Mensajes de join en TTN [26]

5.2.6 Resumen de la evolución del código

En esta sección se resumirá el trabajo que se hizo para llegar al código final. Se comenzó el proyecto utilizando un código de ejemplo brindado por RAK, como se mencionó anteriormente, el mismo presentaba malas pautas de programación. Sin embargo este era el único código donde el transceiver LoRa funcionaba correctamente, por lo que se decidió tomarlo de todas manera como base para el proyecto.

Este código contaba con un menú al que se accedía a través de serial, en el que se podía configurar la frecuencia de trabajo, realizar un join y enviar un mensaje. Este mensaje era la información de la temperatura interna del microcontrolador.

Dentro de las primeras modificaciones que se le realizaron al programa, fue mejorarlo de tal manera que se pudiera manipular las salidas digitales, para poder encender o apagar módulos o leds para debuggear.

Posteriormente se reconfiguró la lectura de la entrada analógica para poder leer un voltaje externo al microcontrolador y no la temperatura interna. Se configuró con un valor de 3 V de referencia en un conversor ADC de 12 bits.

Debido a la necesidad de utilizar sensores con comunicación digital se configuró e implementó la librería del protocolo I2C, siendo este el utilizado por el sensor de luz BH1750.

Finalmente el código se modificó utilizando las funciones desarrolladas anteriormente pero en un flujo de programa cerrado descrito en 5.2.1 Flujo principal

Actualmente el código utiliza un 43.5 % de la memoria destinada a código y un 38.2 % de la memoria destinada a datos. Esto permite agregar varias funcionalidades sin la preocupación inminente de quedarse sin memoria.

5.3 Software

El objetivo de esta sección es explicar el funcionamiento del producto, desde que el nodo envía un mensaje a TTN hasta la interfaz donde se visualizan los datos.

Para esto se presenta un diagrama de bloques del desarrollo del producto, luego se explicará en profundidad cada aplicación que compone el software implementado.

5.3.1 Diagrama bloques inicial

Inicialmente se realizó un diagrama de los distintos bloques del software. Estos bloques o aplicaciones cumplian una funcion determinada en el procesamiento y la visualización de los datos obtenidos por el nodo.

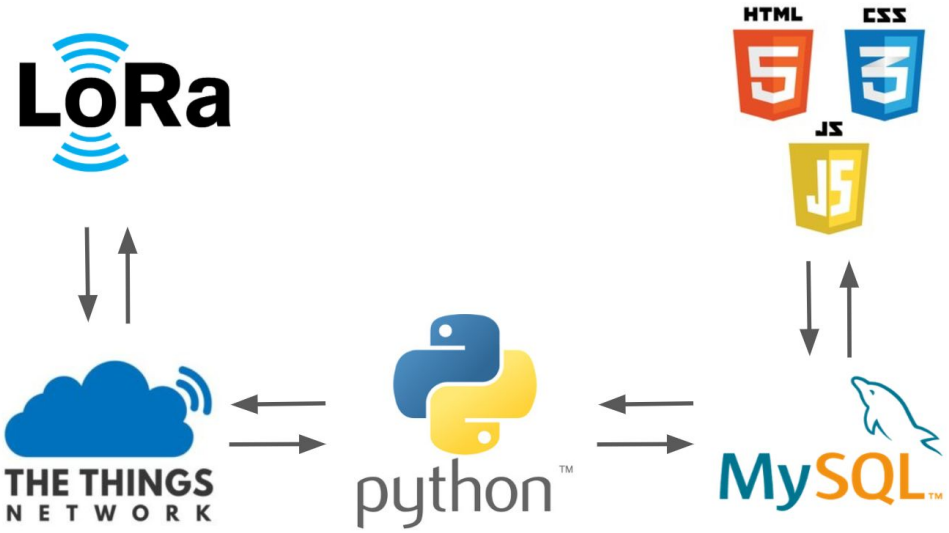

Imagen 53: Flujo inicial de programa

Este esquema, parte desde LoRa como punto inicial de salida con los datos provenientes del nodo. Lo recibe TTN que tiene la tarea de decodificar el mensaje.

Posteriormente, mediante Python, se procesan los datos obtenidos del nodo descargados desde TTN. Por otra lado, Python también lee el estado de las luminarias desde el sistema del aeropuerto (ejemplo SCADA). Luego de analizar toda la información se la envía a la base de datos. Esta contiene datos sobre si una luminaria está prendida, apagada o si presenta algún tipo de problema.

Por último, una vez procesados los datos se almacenan en un servidor MySQL, lo que permite la visualización en un sitio web en tiempo real.

Sin embargo debido a la situación sanitaria mundial (COVID-19), los objetivos del aeropuerto se vieron modificados, postergando el proyecto por tiempo indeterminado. Esto generó que no se llegara a analizar el funcionamiento interno del sistema del aeropuerto. Para emular el mismo, se agregó una placa Arduino Uno con una botonera asociada, que cumpliría el rol de un SCADA interno del aeropuerto. La cual permite accionar las luces de prueba y que Python pueda acceder a esta información, a través de un protocolo serial.

5.3.2 Diagrama bloques final

Finalmente durante la ejecución del proyecto surgieron cambios en el esquema final del software, debido a diversos inconvenientes mencionados anteriormente y elecciones de acuerdo a la necesidad de la aplicación.

Como primer modificación se encuentra la colocación de la placa arduino que se mencionó en la sección anterior.

Otro cambio importante que se realizó para simplificar el sistema fue dejar un vínculo unidireccional entre el nodo y TTN. De esta manera únicamente el nodo le puede enviar datos a TTN y no en sentido contrario. De todas formas, las ventajas de la bidireccionalidad en este vínculo se mencionan en 10. Trabajos a futuro.

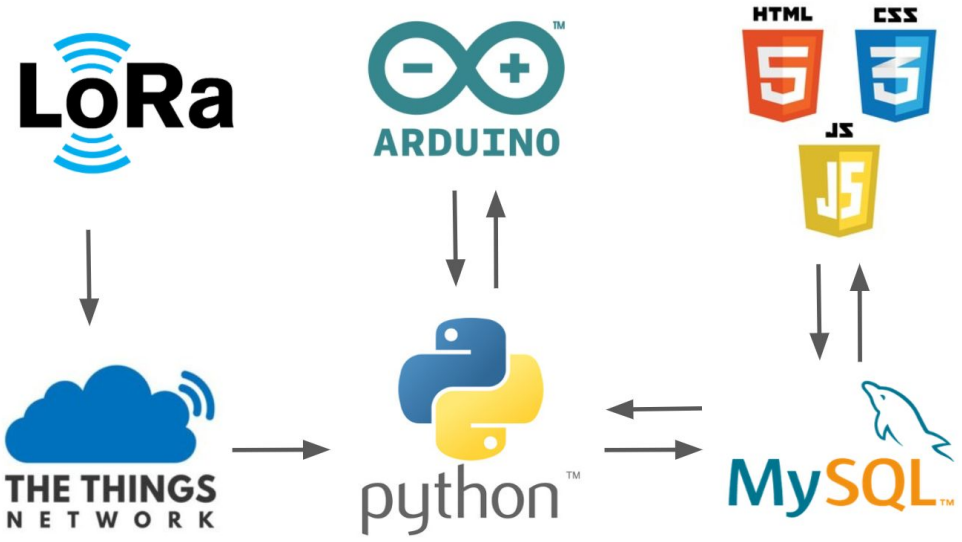

Imagen 54: Diagrama implementado de bloques del software

5.3.3 TTN en software

The Things Network o TTN es el gestor de la red LoRaWAN que mediante el gateway puede recibir y responder paquetes a los diversos nodos, el funcionamiento del mismo se explica en la sección 3.4 The Things Network.

Por como funciona la red LoRaWAN, los mensajes se envían desde el nodo encriptados a través de LoRa, brindando de esta manera mayor seguridad a la información.

Luego los datos son recepcionados por el gateway en TTN y se pueden visualizar en la consola.

En LoRa la información útil que se quiere enviar en cada mensaje se le denomina payload. En este desarrollo, el payload está compuesto por 5 bytes. Un byte de inicio del mensaje y cuatro bytes de información, donde dos corresponden al voltaje de batería y dos a la luminosidad del sensor.

Para poder interpretar el payload recibido se crea una “decoder” en TTN, que se encarga de traducirlo de bytes a las unidades correspondientes, obteniendo el valor de voltaje de batería y el valor de luminosidad del sensor.

```
function Decoder(bytes) {
  var lux = bytes[1]<<8 | bytes[2];
  var bat = bytes[3]<<8 | bytes[4];

  battery = (bat*6/4096)

 return {
    lux: lux,
    bat: battery
  }
}
```

Imagen 55: Funcion de conversion de valores [26]

Cabe mencionar que el payload es visible luego de desencriptar el mensaje recibido por el nodo. En la siguiente imagen se puede observar el payload y la interpretación del mismo.

Imagen 56: Captura de tráfico de la aplicación y despliegue de información del mensaje [26]

Por último, este mensaje con el valor ya interpretado queda almacenado temporalmente en los servidores de TTN lo cual puede ser accedido usando un programa, en este caso desarrollado en Python.

5.3.4 Python

Para el procesamiento de los datos, la recepción de los datos de LoRa y la lectura del SCADA del aeropuerto se desarrolló un software a medida implementado en Python.

Este lenguaje de programación de alto nivel permite realizar diferentes tareas en forma de rutinas que se utilizan ejecutando un script.

En este proyecto particular, se pensó en la utilización de una computadora local en el aeropuerto donde se desea instalar el sistema. En la misma, se ejecutaria el software desarrollado, para poder procesar los datos recibidos en TTN y los datos del SCADA del aeropuerto.

En esta sección se presentará las principales rutinas del código desarrollado en Python. Para esto se subdividió el programa en diferentes funciones donde cada una realiza una tarea de gran importancia.

5.3.4.1 Identificación aplicación TTN

Para poder acceder desde Python a TTN se requiere identificar la aplicación creada y tener una llave de acceso como se puede ver en la

siguiente imagen. De esta forma mediante MQTT se realiza una suscripción a los datos de la aplicación. Luego de establecida la conexión se procede a esperar los mensajes.

```
84    app_id = "appy"
85    access_key = "ttn-account-v2.cxsQJSM77baRYFMmcDx5ycnpvKS8KKuLbbkh5AwrqQY"
86    handler = ttn.HandlerClient(app_id, access_key)
87    mqtt_client = handler.data()
88    mqtt_client.set_uplink_callback(uplink_callback)
89    mqtt_client.connect()
```

Imagen 57: credenciales para acceder a TTN

5.3.4.2 Lectura datos TTN

Esta rutina se encarga de leer el servidor de TTN en tiempo real. Al mismo tiempo está solicitando y leyendo el arduino a través del puerto serial a la espera de modificaciones en los actuadores de las luces.

Al llegar un mensaje desde TTN, el programa lo analiza obteniendo el identificador “device EUI”, la luminosidad y el voltaje de batería.

Con el dato del identificador del nodo, se realiza una consulta a la base de datos del servidor privado en SQL. Esto se hace a través de la tabla de “identificación” buscando por número de hardware del nodo y se obtiene como respuesta el número de luminaria y el umbral asociado. La instrucción se puede ver en la siguiente figura.

```
# ABRIR CONEXION BASE DE DATOS Y ENVIO DATO
db = MySQLdb.connect("www.electronicaboost.com.uy","hboost_python","TesisLora","hboost_Base_de_Datos" )
cursor = db.cursor()
consulta= "SELECT * FROM `Identificacion` WHERE `Numero_Hardware` LIKE '" + str(nroHardware) +"'"
cursor.execute(consulta)
#respuesta base de datos
records = cursor.fetchall()
```

Imagen 58: Credenciales y consulta SQL

5.3.4.3 Procesamiento y lectura sistema aeropuerto

Luego de tener los datos de número de luminaria, umbral y la luminosidad se puede determinar si esta se encuentra encendida o apagada, lo que no quiere decir que necesariamente esté funcionando correctamente, es decir podría estar apagada o rota.

Con los datos obtenidos de la luminaria junto con la información obtenida por el arduino sobre el estado del actuador, se puede discernir si la lámpara está prendida, apagada o rota.

Para poder obtener la información del puerto serial sin tener que esperar la llegada de un mensaje, se implementó una interrupción por timer en el

software. De esta manera se refrescan los datos del panel del aeropuerto cada un tiempo fijo de cinco segundos. Es importante mencionar que cada vez que se realiza una lectura del puerto serie, se guarda el valor en una variable, de esta manera si se detecta una diferencia mayor a treinta segundos entre la hora actual y la guardada se dispara una falla de comunicación en el puerto serial. Esta falla se visualizará en el sitio web.

```
#Funcion Interrupcion Serial
def Serial_Interrupt():

    global estado
    global serial_last
    print('Actualizacion Serial')
    ard.write(b'\n')
    time.sleep(1)
    lectura = ard.read(ard.inWaiting())
    if (lectura[0] == 64):
        estado = [lectura[1]-48, lectura[2]-48, lectura[3]-48, lectura[4]-48, lectura[5]-48, lectura[6]-48, lect
        serial_last = datetime.datetime.now()
    else :
        estado = [-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]
    threading.Timer(5,Serial_Interrupt).start()
```

Imagen 59: Rutina para lectura del puerto serial

Luego de discernir el estado de la luz se envía la información al servidor SQL para posteriormente ser visualizada en el sitio web. En caso de haber una luz quemada o baja batería, se envía el mensaje a la tabla de error en el servidor.

5.3.5 MySQL

MySQL es un sistema de almacenamiento y gestión de bases de datos. En este proyecto se utilizó como herramienta para guardar y modificar los datos del servidor, que luego serán visualizados en el sitio web.

El servidor utilizado se encuentra en la nube y no localmente en alguna máquina. Este está compuesto por tres tablas principales:

La primera tabla denominada “Panel” almacena la información ya procesada de los mensajes proveniente de los nodos. Los atributos de esta son: el “ID” o número identificador de la luminaria, el “Estado” o estado de funcionamiento en el que se encuentra la luz, la “Luminosidad que indican los lux captados por el sensor lumínico y la “Última Hora” que es la hora en la que se ingresa a la base de datos, este campo se genera automáticamente al ingresar un nuevo valor.

Imagen 60: Tabla Panel de la base de datos

La segunda tabla de “Identificación” es donde se asocia el número de hardware con el número de luminaria. Sus campos son : “Número_Hardware” el que corresponde con “device EUI”. El “Identificador” el cual se corresponde con “ID” de la tabla Panel y el “Umbral” el cual se decide si la luminosidad recibida es suficiente para considerar que está encendida.

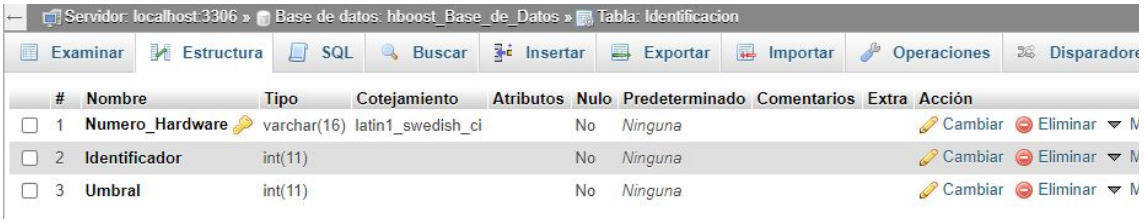

Imagen 61: Tabla Identificación de la base de datos

Por último la tercer tabla es donde se almacenan los errores, ya sea los que se van a visualizar en el tablero o algún mensaje de baja batería, entre otros. Sus atributos son: “ID Luminaria” el identificador del número de luminaria, ”Descripción” de la falla, ”Tipo” de falla y ”Hora” del evento.

Imagen 62: Tabla Historial error de la base de datos

5.3.6 Sitio web

El sitio web es la interfaz que se pensó para que los operarios del aeropuerto pudieran ver en tiempo real la información de la luminaria. En un futuro si el proyecto se lleva a cabo, se podría realizar una aplicación para celulares y tablets, permitiendo visualizar la información en cualquier lugar y simplemente con un smartphone.

Este sitio se encuentra montado sobre el mismo servidor en la nube de la base de datos. La url del sitio es electronicaboost.com.uy/Tesis. Este, utiliza css para la estética de la página y javascript junto con json para poder acceder a la base de datos y actualizarse en tiempo real.

El sitio principal posee javascript el cual solicita a tres páginas php donde cada una de estas páginas accede a una tabla distinta. De esta manera se puede obtener toda la información de los nodos.

Ya entrando en el panel principal del sitio se puede visualizar señalizadores que indica el estado que presentan diez luces distintas.

Cada indicador de luminaria puede tener cinco estados diferentes :

- Prendido
- Apagado
- Falla
- Warning timeout Serial
- Warning timeout LoRa

Los primeros cuatro estados los decide y analiza el código de python mientra que el Warning timeout LoRa, lo determina el sitio web al no recibir ningún mensaje por más de media hora. Esto tiene como objetivo que en el caso de no recibir mensajes, el sitio web entienda que hay una falla en la comunicación de LoRa.

Por otro lado en todos los indicadores, se visualiza la luminosidad que tiene cada nodo, para poder brindarle más información al operador de lo que está sucediendo en cada luz.

Otro dato importante que se podría agregar en los indicadores, es simplemente un aviso de batería baja en el nodo.

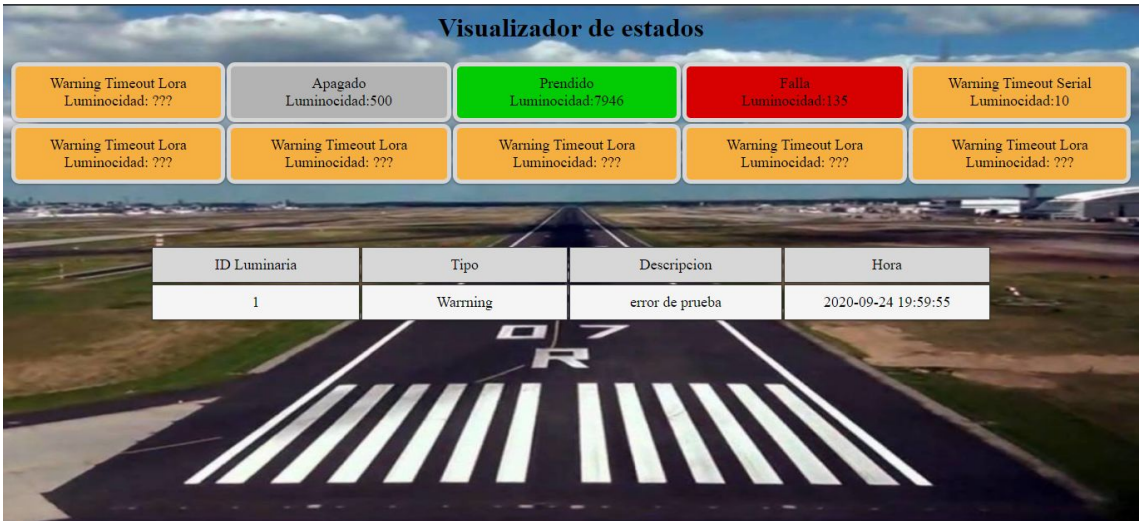

Imagen 63: Captura de sitio web con varios estados de botones

En la parte inferior del panel principal se encuentra una tabla de eventos, para poder registrar posibles fallas en las luminarias indicando el tipo de error, una breve descripción del mismo y la hora en el que ocurrió.

5.4 Encapsulado

Para realizar el diseño del encapsulado, se utilizó la herramienta “Autodesk Inventor” con una licencia estudiantil. Este programa presenta un gran potencial para el diseño industrial.

Este encapsulado fue pensado para resistir diferentes condiciones climáticas, buscando que tenga un alto grado de estanqueidad, sin afectar las mediciones del sensor.

A raíz de lo analizado en el capítulo “4.3 Sensores de luz” , se diseña un encapsulado que también funciona como tubo para direccionar el sensor lumínico. De esta forma la luz medida por el sensor es solamente la de la fuente de luz y no la de ambiente. Para esto se realizaron dos versiones; en la primera, el tubo está situado perpendicular al plano del PCB y en la segunda está paralelo al PCB. Finalmente se opta por la segunda versión debido a que las dimensiones del encapsulado son más pequeñas incluso llevando el largo del tubo a 5.5cm.

Para el visor del sensor, se utiliza en la punta del tubo un cubreobjeto, dando como resultado un visor que permite el ingreso de luz al mismo pero no de polvo y agua al interior del nodo.

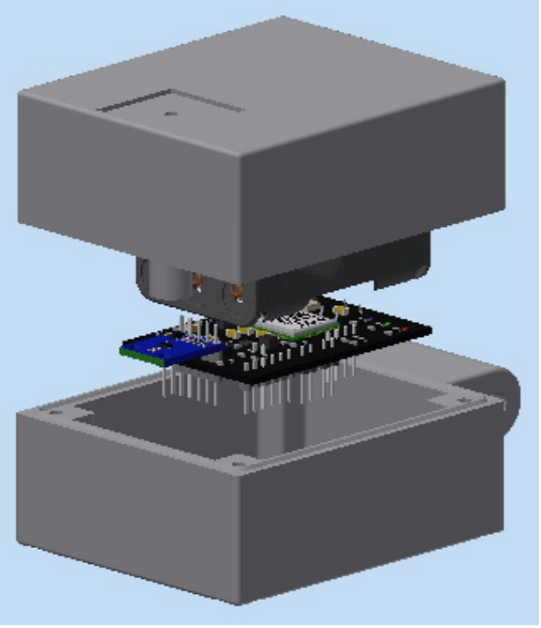

Imagen 64: Encapsulado versión 1

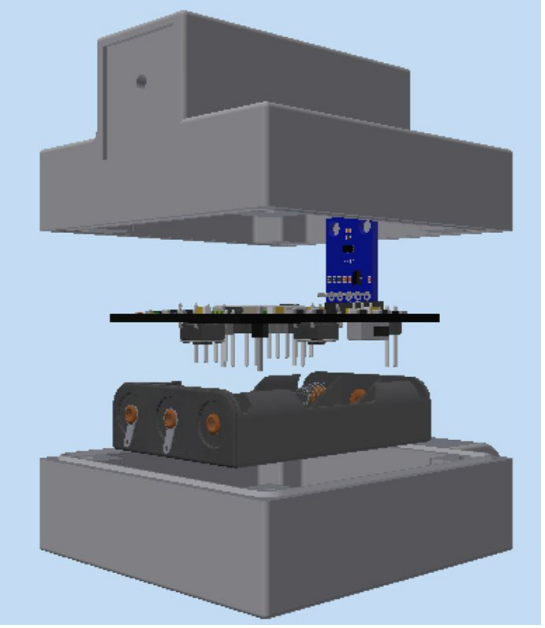

Imagen 65: Encapsulado versión 2

6. Simulación del proyecto

En esta sección del proyecto, se presentará el proceso que se llevó a cabo para poder simular las condiciones del aeropuerto. Para lograr esto se buscó hacer una réplica de una luminaria del aeropuerto, ya que no había disponibilidad o repuestos para poder utilizar como muestra. Por otra parte, se realizó un sistema de control de las luces con protocolo de comunicación serial, para poder simular el “estado” de las luces y por último se realizó una estructura necesaria para colocar y oriental el nodo en la luminaria.

6.1 Luminaria aeropuerto

Para poder probar el funcionamiento del proyecto, se buscó crear una luminaria lo más parecida a la realidad. Durante la primera visita del aeropuerto se observaron las luminarias, su estructura y su orientación en la pista. Estos datos fueron utilizados para realizar físicamente la base de la luminaria. En base a eso se construyó una estructura similar, utilizando un tubo metálico como mástil, al que se le puso un disco circular en la base y una tuerca en la parte superior. En esta tuerca se enrosca la luminaria.

Imagen 66: Maqueta de simulación de luminaria aeropuerto

En cuanto a la luminaria, la empresa del aeropuerto interesada en el proyecto brindó un documento de especificaciones de las mismas (modelo FAE-1200) de donde se obtuvieron algunos datos técnicos.

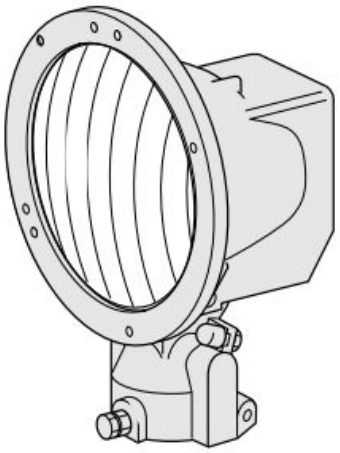

Imagen 67: Luminaria FEA-1200 [27]

Sin embargo en este documento no especificaba en ningún momento la potencia lumínica de las mismas, por lo tanto al no conocer este parámetro se buscó crear una luminaria de alta potencia. Para esto se colocó un chip led de 100 Watts con un disipador y un ventilador en la parte posterior debido a la alta temperatura que genera. Este Led se colocó en el interior de un soporte de acero que se enrosca en la base de la luminaria.

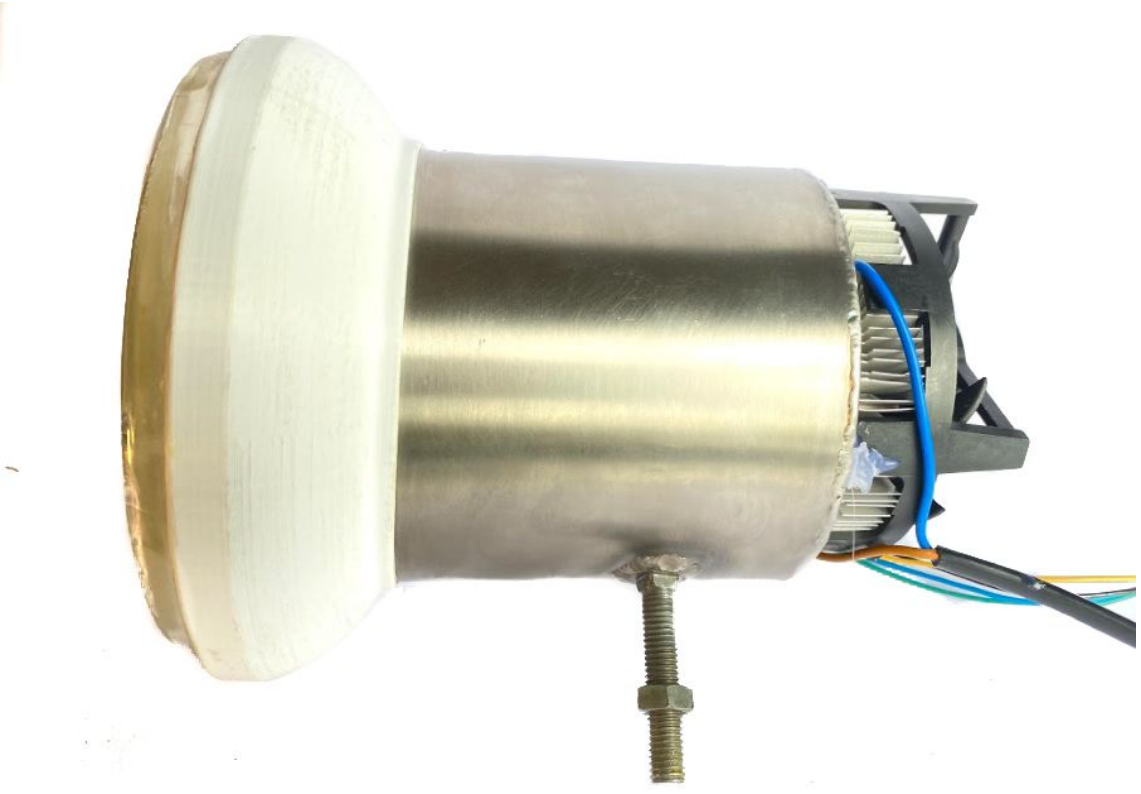

Imagen 68: Estructura del luminaria.

Imagen 69: Estructura del led 100W con disipador frontal

Por último,en la parte delantera de la luminaria se colocó un vidrio de un foco automotriz y para acoplarlo de forma óptima con el soporte de acero circular, se realizó una impresión en 3D de un adaptador como se observa en la siguiente figura.

Imagen 70: Adaptador de óptica

6.2 Armado del nodo

En esta sección se mostrará el armado completo del nodo con todos sus componentes y la colocación en la estructura o maqueta de la luminaria. Luego de tener todos los elementos necesarios del nodo se montaron en el interior del encapsulado.

Primeramente se soldó los cables del portapilas y se conectó la antena al nodo. En la siguiente imagen se puede ver esta implementación.

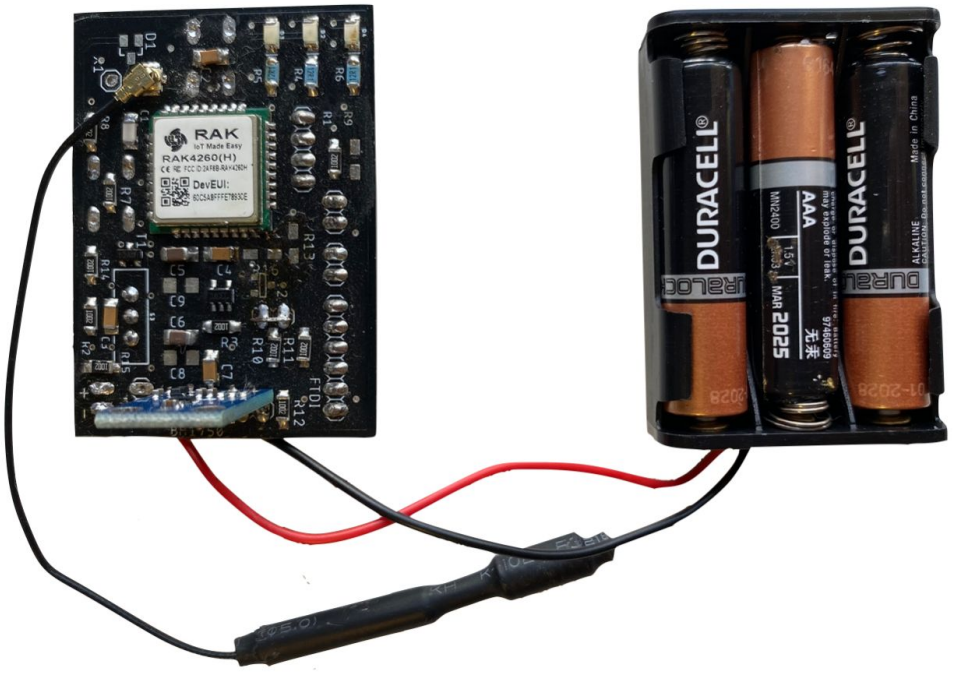

Imagen 71: PCB prototipo terminado

Posteriormente, se colocó el PCB diseñado en el interior del encapsulado, de manera que el sensor de luz quedará alineado con el orificio diseñado para este

El siguiente paso, fue pegar el portapilas en el interior del encapsulado mediante silicona. De esta manera en el interior de una de las tapas quedó colocado el portapilas y en la otra el PCB.

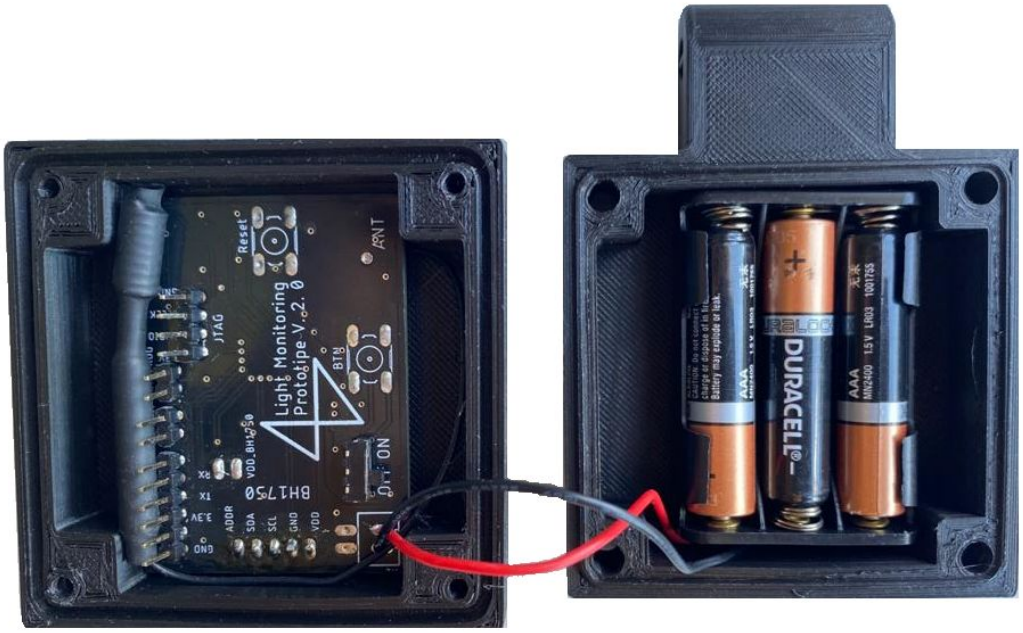

Imagen 72: Prototipo abierto en dos partes

Por último para evitar que ingrese agua y tenga resistencia a la intemperie, se colocó el vidrio protector del sensor donde ingresa la luz y también se instaló una goma que sobre la unión entre las tapas para poder lograr que el nodo sea lo más “estanco posible”.

Luego de tener el nodo armado, se acopló a la estructura de la luminaria mediante un brazo. Esta permite regular y posicionar correctamente el nodo a la estructura, regulando la altura y dirección. De esta forma se puede medir la mayor luz posible proveniente de la luminaria.

Es importante mencionar que sería bueno diseñar algún sistema de calibración de la posición, debido a que actualmente se realiza manualmente de manera iterativa, es decir buscando la posición en la cual el sensor recibe la mayor cantidad de luz.

6.3 Protección gateway

Para poder utilizar el gateway en el exterior se construyó una protección de manera que no le afecte las inclemencias del tiempo. Para esto se utilizó un tubo de PVC de 70mm de diámetro por 1m de largo, con tapas estancas en las puntas. En el mismo se insertó el gateway, la antena y el transformador. De esta forma el único cable que sale hacia afuera es el de alimentación a 220V.

Imagen 73: Protección gateway

6.4 Sistema de control luminaria

Para poder simular el estado de las luces, se creó una especie de tablero o botonera, que se comunica con el servidor mediante protocolo serial. Con la información del estado de las luminarias más los datos del nodo, se puede discernir si la luz está rota o funciona correctamente. Por ejemplo, si la luz aparece como prendida en el tablero del aeropuerto pero sin embargo el nodo muestra como que se encuentra apagada, probablemente la luz esté fallando o se haya dañado.

Imagen 74: Maqueta simulando un SCADA

Este sistema es una simulación de lo que podría ser el centro de control o el tablero de mando de cualquier aeropuerto, debido a que no se sabe exactamente cómo funciona realmente. Se colocó el protocolo serial para obtener los datos, ya que el mismo es muy utilizado en la industria y de gran compatibilidad con la plataforma donde se desarrolló el servidor.

7. Muestra de resultados

Luego de tener el prototipo funcionando correctamente se realizaron diferentes pruebas finales para poder testear su funcionalidad en diferentes áreas. Este proceso de validación del producto permite determinar si los resultados son los necesarios para que el proyecto sea viable.

Por otra parte estas pruebas sirven también como una primera etapa de testeo del prototipo, siendo esta área muy importante para poder visualizar diferentes errores en el funcionamiento.

7.1 Alcance

El alcance del dispositivo es un factor de estudio, ya que se colocó una distancia mínima de tres kilómetros en el envío de datos para que el proyecto sea funcional.

Estas pruebas de distancia de comunicación se realizaron en una área rural de manera de simular lo mejor posible las condiciones del aeropuerto. Es bueno mencionar que en este último se tiene línea de vista entre los nodos y la antena central del gateway, lo que mejoraría la estabilidad de la comunicación.

El gateway central se colocó a unos tres metros de altura para obtener una mejor transmisión y recepción de los datos. Este parámetro es importante, ya que existe una relación entre su elevación y el correcto funcionamiento de la comunicación, es decir a mayor altura del gateway mejor enlace.

Por otra parte, con el prototipo o nodo terminado y totalmente cerrado, se realizó un mapeo con gps mediante el servicio de ttnmapper.org [28]. De esta forma se obtuvieron las distancias máxima de funcionamiento del dispositivo y se visualizó en qué momento comenzaba una pérdida de paquetes significativa (mayor al 10%).

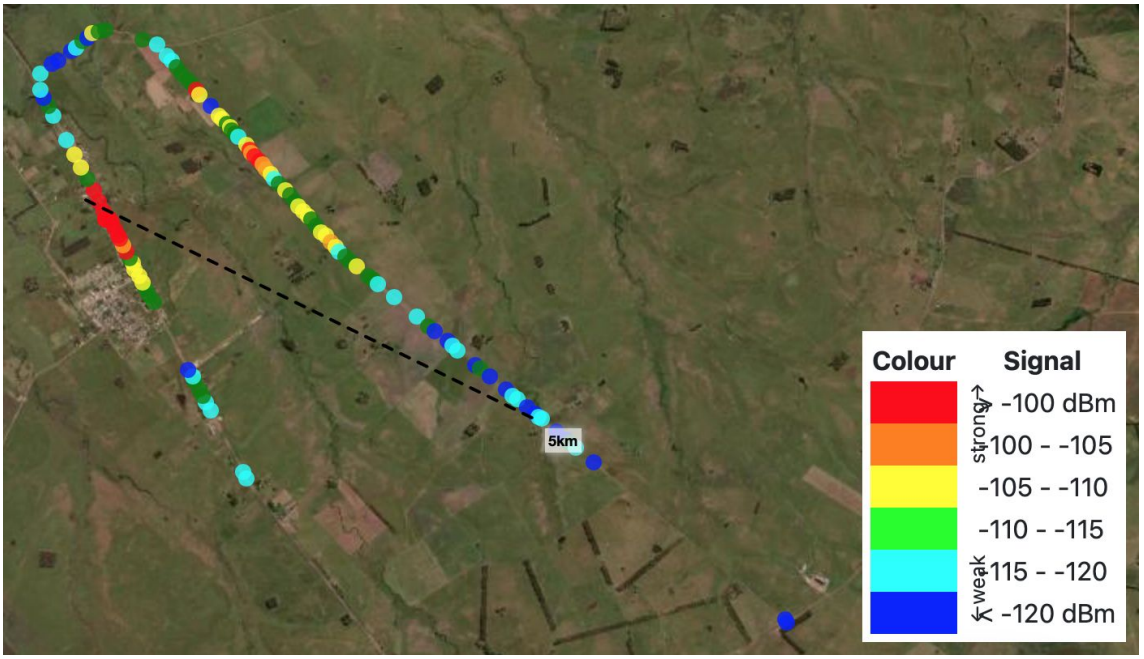

Imagen 75: Mapas de pruebas de alcance [28]

7.1 Sensor de luz

El objetivo principal de este producto está en poder discernir si las luces de pista están prendidas o apagadas. Para esto se realizaron diferentes pruebas al prototipo final, evaluando si esto se lograba en cualquier condición climática y en cualquier horario.

El nodo está diseñado de manera que por el orificio únicamente le entre luz de la óptica o fuente de luz que se quiere medir, evitando que ingrese luz solar directa o reflejada y generando falsos positivos.

Para la prueba principal, se colocó el prototipo final en la maqueta diseñada. Se orientó el sensor de luz del nodo al punto de mayor intensidad lumínica de la luminaria. Esta calibración se realizó mediante un proceso iterativo obteniendo la posición óptima.

Luego del posicionamiento, el desarrollo de la prueba consistía en estar durante todo un día midiendo la intensidad de luz del sensor, con la luz apagada y en el instante siguiente con la luz prendida. Con esto se logró visualizar datos relevantes como son saber cuánta luz solar ingresaba al sensor, ya sea reflejada o directamente.

Imagen 76: Representación de luz reflejada en la optica

Exportando y analizando los datos obtenidos, la relación de luz solar frente a la luz generada por la fuente que se quiere sensar es insignificante (menor al 10%). Esto se puede observar en el siguiente gráfico con los datos obtenidos en la prueba.

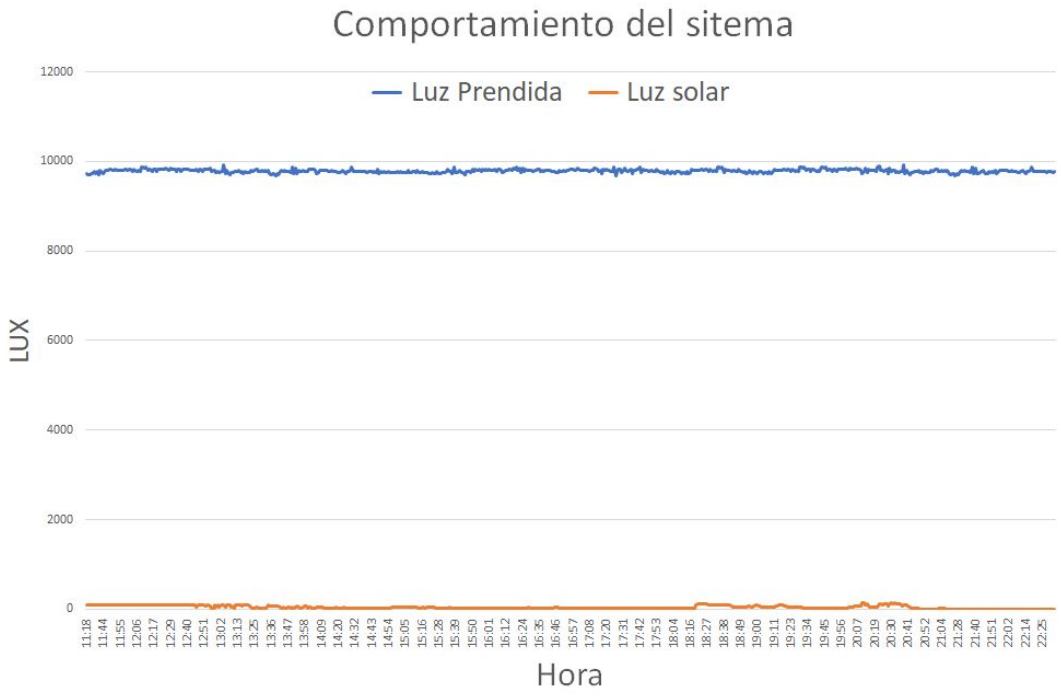

Imagen 77: Luz fuente y luz reflejada.

Una primera conclusión de los datos obtenidos es que en “teoría” se puede discernir el funcionamiento de la luz en cualquier hora del día y cualquiera sea la condición climática. Esto nos permite colocar un umbral para determinar si la luz está o no encendida a un 50% del valor máximo de

luminosidad proyectada por la luminaria. Dejando un margen de seguridad suficiente, ya que la potencia de la luz solar varía durante el día y en diferentes épocas del año.

7.2 Encapsulado

En esta sección, se le realizaron pruebas al encapsulado de estanqueidad, para saber que tan resistente es el mismo a diversas condiciones climáticas.

Cabe mencionar que este tipo de pruebas se realizan en cualquier envolvente que contiene electrónica, al enfrentarse a diferentes condiciones climáticas. Esto define la efectividad de sellado contra el ingreso de cuerpos extraños o agua al mismo y se conocen como clasificación IP (International Protection).

En la siguiente imagen se observa la tabla de grados de protección IP según la norma IEC 60529 para agua y polvo.

PRIMER DÍGITO	GRADO DE PROTECCIÓN POLVO DESCRIPCIÓN	SEGUNDO DÍGITO	GRADO DE PROTECCIÓN AL AGUA DESCRIPCIÓN
0	No protegido.	0	No protegido.
1	Protegido contra objetos sólidos de diámetro ≥ 50 (mm).	1	Protegido contra goteos que descienden verticalmente.
		2	Protegido contra goteos que descienden verticalmente cuando la cubierta esta inclinada hasta 15º
2	Protegido contra objetos sólidos de diámetro ≥ 12,5 (mm).	3	Protegido contra agua pulverizada.
3	Protegido contra objetos sólidos de diámetro ≥ 2,5 (mm)	4	Protegido contra salpicaduras de agua.
		5	Protegido contra chorros de agua a presión.
4	Protegido contra objetos sólidos de diámetro ≥ 1 (mm)	6	Protegido contra chorros de agua a presión de alta potencia.
5	Protegido contra suciedad.	7	Protegido contra efectos de inmersión temporal en agua.
6	Hermético contra suciedad.	8	Protección contra los efectos de continuas inmersiones en agua.

Imagen 78: Norma IP de protección. [29]

Las pruebas en el encapsulado se realizaron solamente contra agua. Una de estas consistió en proyectar agua a presión directamente al nodo y ver si la misma ingresaba a su interior. La prueba de inmersión, se basaba en sumergir el encapsulado bajo el agua durante varios minutos y luego verificar el ingreso de agua al mismo.

El resultado fue que está protegido contra chorros de agua a presión. En cambio en la prueba de inmersión ingresó humedad al interior del encapsulado. Debido a estos resultado se concluye que el diseño logra un grado de protección IPX6.

Es importante destacar que las pruebas no fueron realizadas por el ente certificador, sino que se emularon las pruebas reales.

7.3 Consumo del nodo

Las pruebas de consumo son muy importantes en el proyecto, ya que definen una de las características principales del mismo, como es la duración de las baterías del nodo.

Como se mencionó anteriormente, la programación del microcontrolador define dos grandes estados. Por un lado se tiene al RAK prendido donde mide el estado de la luz y envía estos datos. Mientras que en otro momento, el microcontrolador se encuentra en modo sleep un tiempo definido anteriormente por el usuario. Ambos estados definen dos tipos de consumo, uno transitorio y un consumo fijo, que se analizan por separado a continuación.

7.3.1 Consumo fijo

Para conocer el consumo fijo de la placa, se utilizó la rutina del programa final y se midió cuando el microcontrolador se encuentra dormido. Esto se realizó calculando la caída de tensión en una resistencia de 68 KΩ con un voltaje de alimentación de 4.5V, tal como se observa en el circuito de la figura.

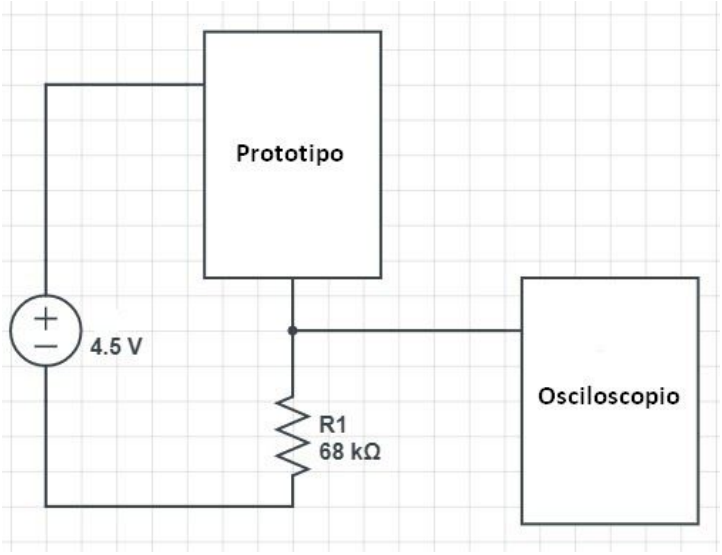

Imagen 79: Medición de consumo fijo

Los resultados obtenidos fueron de aproximadamente 12.5 µA de consumo fijo de toda la placa, siendo estos valores muy parecidos a los que obtuvo la empresa RAKWireless, en una prueba de consumo de un solo microcontrolador de la misma familia (11.2 µA). Cabe mencionar que en los 12.5 µA obtenidos se encuentran todos los componentes de la placa, incluyendo el LDO y el sensor de luz BH1750 que consumen aproximadamente 2 µA.

Por otra parte, se realizó una tabla con los consumos de los componentes más importantes de la placa desarrollada, como se observa a continuación.

Componente	Consumo (µA)
RAK 4260	10.5
BH1750	1
LDO	0.985

Tabla 7: Lista de consumos por componentes.

Durante las pruebas de consumo se encontró una pequeña dispersión en distintos componentes del mismo tipo. Por ejemplo se utilizaron dos microcontroladores RAK4260 con mismo programa cargado y se obtuvieron medidas de consumos con una pequeña diferencia.

Los consumos obtenidos del microcontrolador RAK4260 durante las pruebas son mayores a los que especifica el fabricante en la hoja de datos

(2μA). Sin embargo estos consumos siguen siendo muy pequeños, lo que no afecta significativamente en la duración de la batería.

7.3.2 Consumo Transitorio

Para medir el consumo transitorio del microcontrolador, se realizó un circuito especial con una resistencia graduada de 6.432 Ω. La misma se generó a través de un “hilo de nicrom” que tiene una resistividad de 5.36Ω por metro (51mm de diámetro). Su valor de resistencia se mantiene constante en un amplio rango de temperaturas. Mediante el siguiente circuito se obtuvieron los consumos durante la transmisión y recepción de la placa.

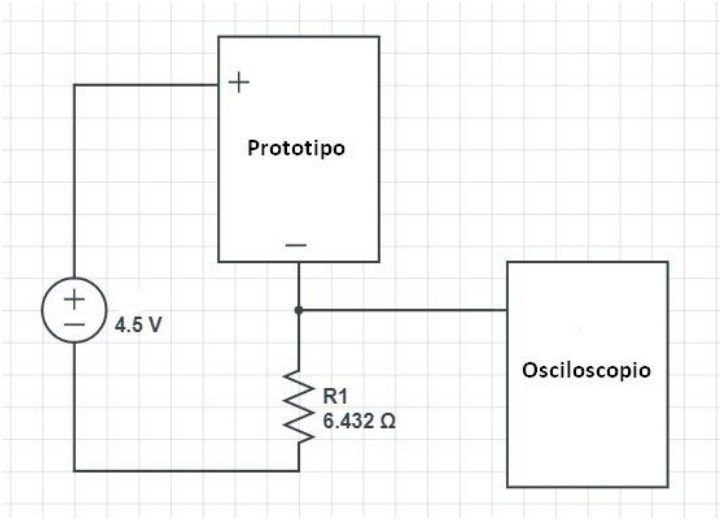

Imagen 80: Circuito medición de consumo transitorio

También se utilizó esta resistencia graduada para medir la cantidad de energía que se gasta en el envío de un dato desde que prende el micro hasta que se apaga. Esto se realizó calculando el área bajo la curva del gráfico de corriente medido a través del osciloscopio, obteniendo un valor aproximado de 5,8 μAh. Estos datos recién mencionados se pueden visualizar en el siguiente gráfico.

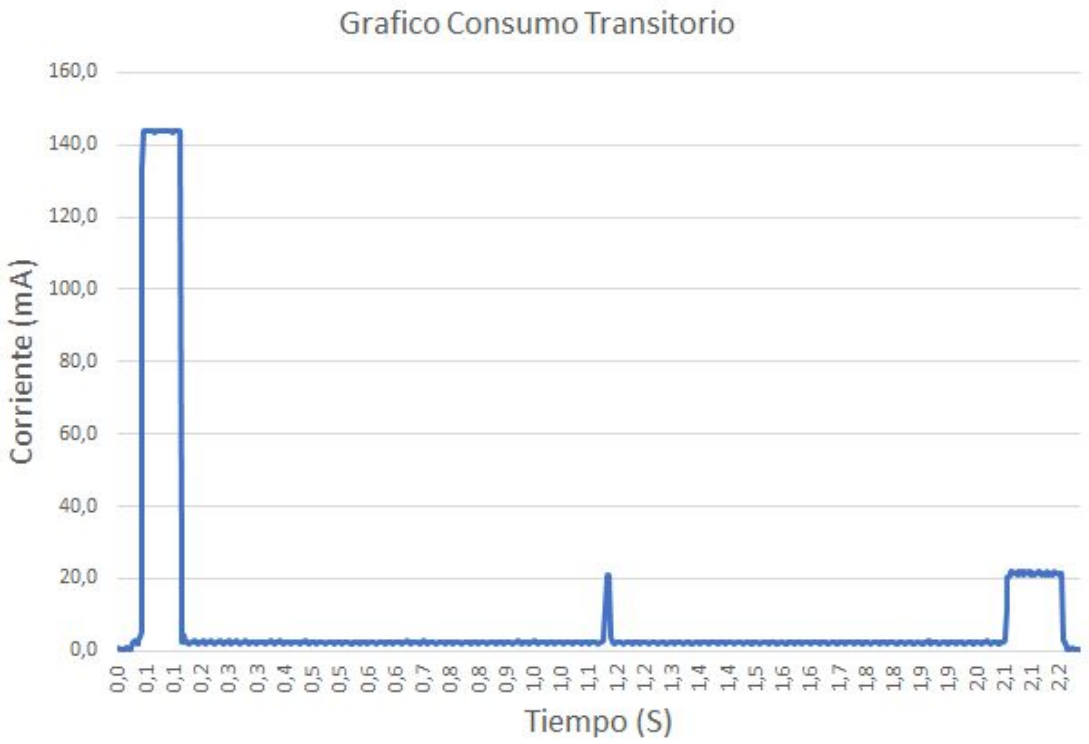

Imagen 81: Perfil de consumo transitorio

Periodo	Consumo (mA)
Transmisión	144
Recepción	21

Tabla 8: Comparación consumo transitorio y fijo

Por otra parte, se realizó otra prueba para ver el consumo de energía transitorio del micro y visualizar un perfil de descarga de batería. La misma se hizo creando un código donde se enviará un dato cada un periodo de 5 segundos, minimizando de esta forma el impacto del gasto de energía de consumo fijo.

En esta prueba se lograron enviar 295.000 mensajes durando la batería aproximadamente 23 días. Cabe mencionar que el micro estaba dormido 5 segundos y prendido 2 segundos aproximadamente mientras hacía las tareas del envío de datos.

En el siguiente gráfico se observa el perfil de descarga de la batería durante la prueba.

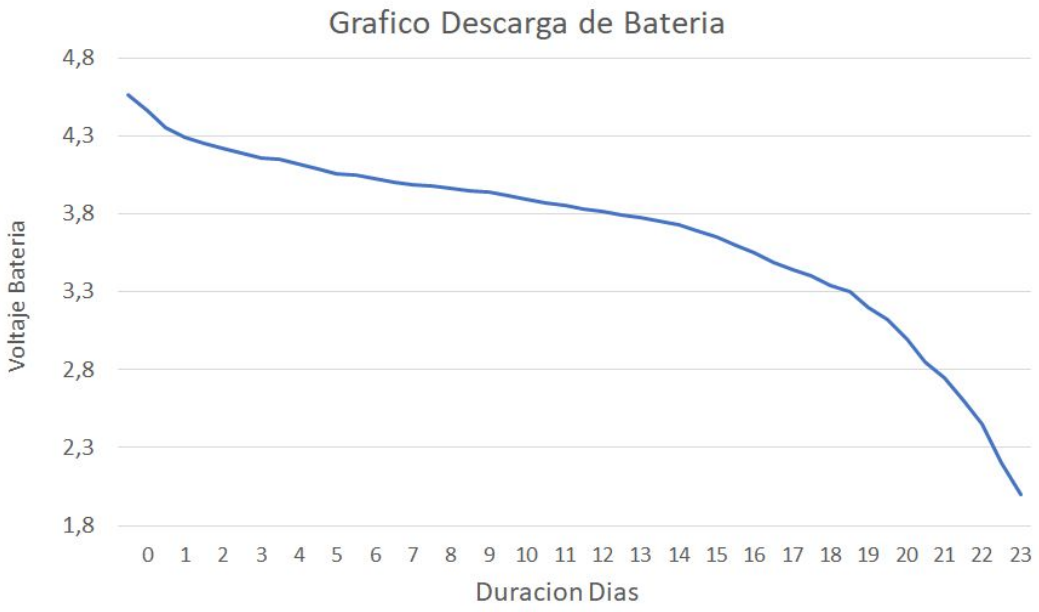

Imagen 82: Descarga de bateria

Las baterías utilizadas para la prueba fueron tres pilas alcalinas AAA Duracell, las mismas tienen una hoja de datos donde se observa el perfil de descarga de las pilas para diferentes consumos. Con esta información se obtuvo que con un consumo entre 1mA y 2 mA (el calculado durante la prueba) la capacidad de las pilas es de aproximadamente 1400 mAh.[30]

En el siguiente gráfico de la hoja de datos de las pilas Duracell se observa que el perfil de descarga con un consumo de 5mA es muy parecida a la curva obtenida en la prueba de descarga de batería.

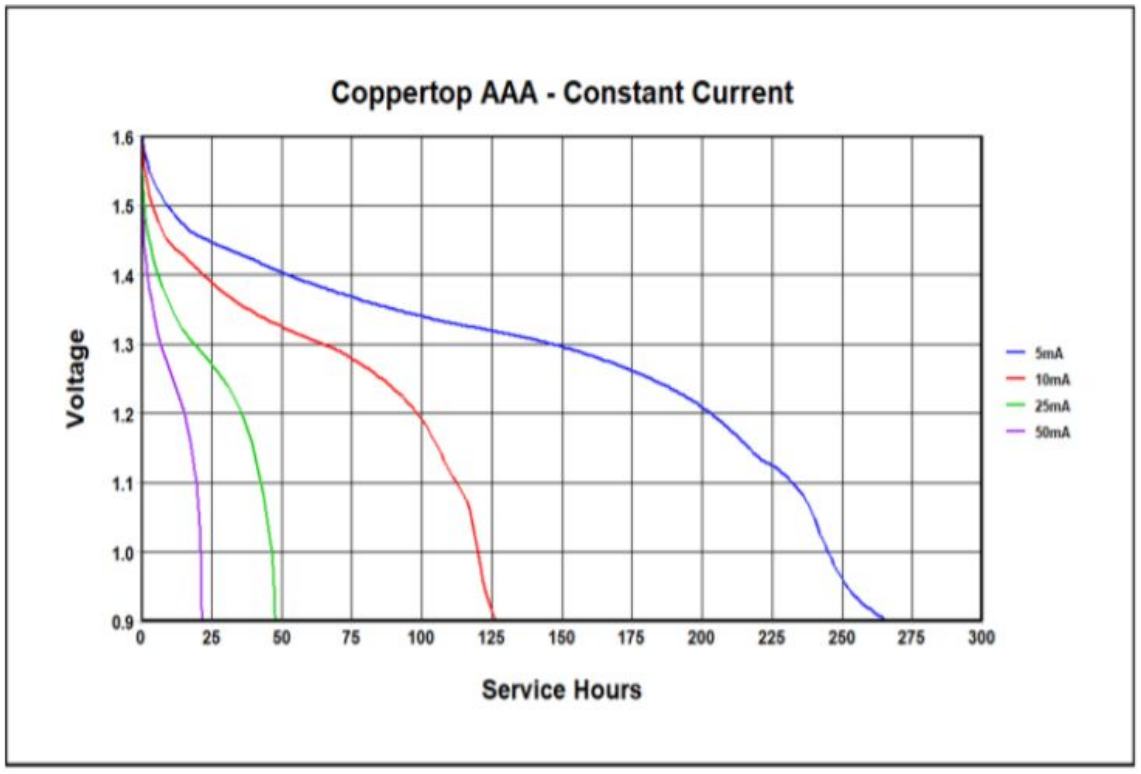

Imagen 83: Descarga de batería para una corriente constante [30]

Luego de obtener la capacidad de la pila a partir del consumo promedio de batería durante la prueba realizada, se calculó el consumo de energía por mensaje, llegando a un valor de 4,7 µAh. Siendo este resultado obtenido muy similar al hallado mediante la resistencia graduada de nicrom.

Método	Consumo de energía por mensaje µAh
Medida osciloscopio	5.8
Prueba descarga bateria	4.7

Tabla 9: Comparación de energía por mensaje.

Por último con los datos obtenidos, se hizo una proyección de cuánto tiempo podría durar la batería en función de los mensajes que se mandan por hora, ya que esto genera el consumo más importante. Cabe mencionar que para estos gráficos la capacidad de las pilas alcalinas AAA se fijó en 1500 mAh, ya que el consumo promedio es menor a 0,1 mA. Igualmente se puede visualizar en el gráfico el consumo para 1000 y 2000 mAh, para el caso de que se utilice otro tipo de batería.

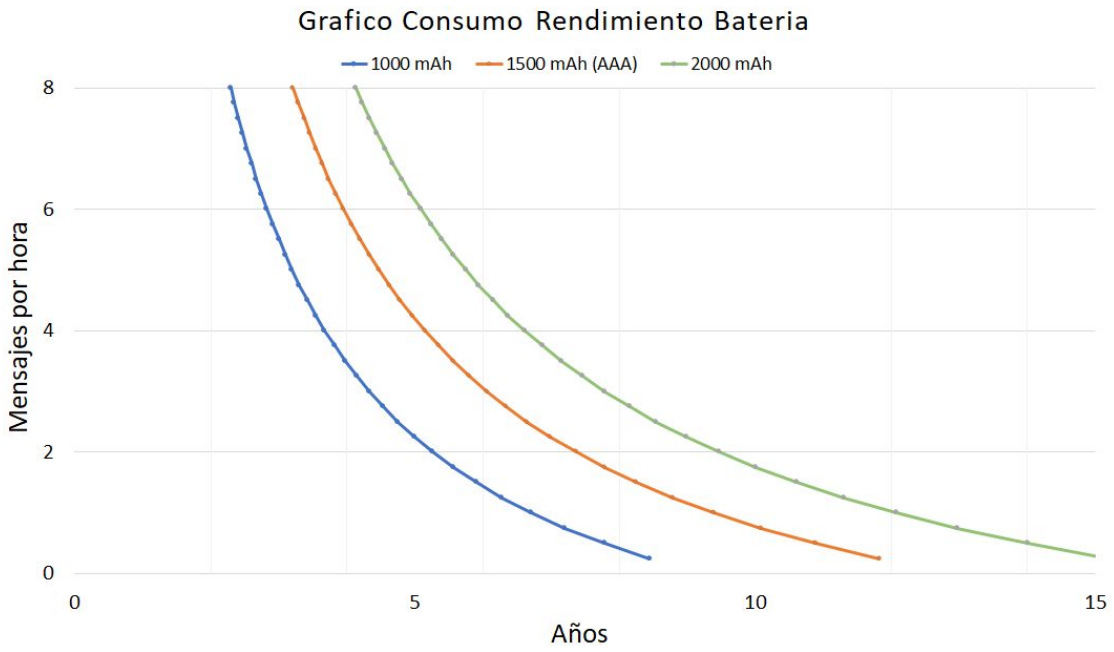

Imagen 84: Vida útil de la batería en función de la frecuencia de mensajes.

En el gráfico anterior se puede observar que enviando un mensaje cada 15 minutos (cuatro mensajes por hora) y con las baterías alcalinas AAA, se puede obtener una duración mayor a cinco años, siendo estos valores aceptables para el objetivo del proyecto.

8. Estimación y costos

En cualquier desarrollo de producto las estimaciones y los costos son determinantes del proyecto, ya que esto define en gran parte si el mismo es viable o no.

En cualquier presupuesto de un proyecto, se tienen diferentes tipos de costos directos, indirectos, costos fijos y costos variables. También para una correcta estimación es importante incluir la mano de obra para poder llevar a cabo el proyecto. Por estos motivos realizar esta tarea correctamente es de gran complejidad, por todas las variables que se presentan. Muchas veces es necesario en las estimaciones recurrir al “juicio de expertos” para que el presupuesto final sea lo más realista posible.

Para este proyecto en particular las estimaciones de costo se realizaron para una producción de 100, 1.000 y 10.000 unidades. Se utilizó como moneda el dólar estadounidense, siendo la misma menos afectada por la inflación.

Por otra parte los costos fueron divididos en diferentes secciones para poder segmentar mejor las variables y luego concluir la viabilidad del producto.

8.1 Costo del producto

En esta parte del presupuesto se definieron los costos directos del desarrollo, es decir los materiales y la mano de obra específica para su realización. Para poder realizar esta estimación se consultó a diferentes personas con experiencia en desarrollos de productos que hicieron más realista y precisa esta tarea.

En el área de encapsulados, con los conocimientos adquiridos de los estudiantes y la ayuda de un experto en diseño de encapsulados se analizó que método utilizar para realizar el encapsulado. Esto va a depender de las cantidades que se estiman fabricar, concluyendo que para una cantidad de 100 unidades es recomendable realizar los encapsulado a través de impresión 3D y a partir de las 1.000 unidades es recomendable y más rentable realizar un molde. El molde permite fabricar una gran cantidad de unidades con una mejor terminación, dándole al producto final competitividad en el mercado.

Para la parte de mano de obra, fue necesaria la ayuda de personas con experiencia en empresas para poder cotizar cuánto tiempo y personal llevaría ejecutar todas las tareas para la realizar el producto. Estas tareas abarcan soldar todos los componentes del PCB, programar el microcontrolador, armar el nodo, entre otras.

8.1.1 Cotización componentes

En esta sección se calcularon los costos de todos los componentes que van en el PCB sin ser el microcontrolador RAK. Para esto se hicieron simulaciones en páginas que habitualmente se utilizan para enviar componentes electrónicos y se elaboró una tabla con los mismo en diferentes escalas 100, 1.000 y 10.000 unidades.

Componentes	Cant.	Descripción	Costo 100 Un	Costo 1000 Un	Costo 10000 Un
Resistencias	10	10 KΩ	5,7	35,48	292,4
Resistencias	3	12 Ω	4,59	20,67	128,7
Botones	2	Botones Reset y Libre	100	920	8100
Interruptor	1	Llave encendido	49,5	374	3278
Transistor	1	Control Bat.	9,77	56,61	377,4
Capacitor	2	10µF	21,02	129,2	1081
Capacitor	4	100nF	15,52	76,63	641,52
Capacitor	1	47uF	6,41	37,22	280,1
Regulador	1	3.3 V	55,6	342,22	2950
Conector Antena	1	SMA	73	720	7100
LEDS	3	RGB	48,12	278,1	2908,8
Sensor Luz	1	BH1750	70	670	6200
Diodo	1	Protección	17,68	98	980
Envío			100	200	500
Total			**571,21**	**3922,65**	**34525,52**
Costo Unitario			**5,7121**	**3,92265**	**3,452552**

Tabla 10: Costos de componentes.

En la tabla se puede observar como los costos de los componentes son menores cuando se aumentan las unidades, siendo importante esta diferencia al pasar de 100 a 1.000 unidades de producción.

Por otra parte, se realizó una cotización a la empresa RAK por una compra de 10.000 unidades, para ver si los costos de este microcontrolador y

transceptor LoRa podrían disminuir. La respuesta de la empresa fue que para esta modalidad de producción, el costo por unidad sería de 5.9 dólares.

8.1.2 Cotización encapsulado

Para realizar el encapsulado se definieron dos formas distintas dependiendo de la modalidad de producción. Para la fabricación de 100 unidades se utilizó impresión 3D, ya que realizar un molde para tan pocas unidades es inviable. Por otra parte, para 1.000 y 10.000 unidades se eligió fabricar un molde. Es importante mencionar que en la fabricación de 1.000 unidades el costo del molde es considerable, por este motivo si se optara por producir esta cantidad habría que analizar particularmente qué método de fabricación utilizar.

En costos la fabricación del encapsulado por impresión 3D es de aproximadamente 5 USD por unidad. Por otra parte, la fabricación de un molde similar cuesta alrededor de 20.000 USD y luego el material para fabricar cada unidad se estimó en 1 USD.

8.1.3 Presupuesto unitario

Luego de tener el costo de los componentes y el encapsulado, se realizó una cotización de los elementos restantes que son el PCB y la mano de obra. Para esto se realizaron diferentes consultas a distintos proveedores para conseguir el mejor precio en todas las modalidades de producción 100, 1.000 y 10.000 unidades.

Para evaluar el costo de mano de obra se analizó cuánto tiempo le podría llevar a un técnico armar un encapsulado completo, llegando a la conclusión que una hora era un tiempo razonable. Evidentemente a gran escala este tiempo podría disminuir, si existiera una cadena de producción automatizada.

Para la cotización del PCB se consultó a la empresa PCBWay cuál sería el costo asociado, para las distintas modalidades de producción.

Por otra parte, luego de obtener el costo total se colocó un factor por riesgos. Este factor se suele utilizar en gestión como medida de prevención ante posibles cambios en la duración o en los costos del proyecto. En este caso en particular, pueden surgir diferentes imprevistos como por ejemplo costos no contemplados de importación, impuestos , envíos, despachantes de aduana, entre otros.

En la siguiente tabla se encuentra detallada la cotización general del producto, llegando a un costo del producto final de 32 USD para una producción de 100 unidades, 48 USD para una producción de 1.000 unidades (fabricando un molde) aunque si se utilizara impresión 3D el costo sería de 26

USD. Por último en la fabricación de 10.000 unidades el costo de fabricación unitario sería de 22 USD.

Item	Descripción	Costo 100 Un	Costo 1.000 Un.	Costo 10.000 Un
Micro RAK 4260	Costo del Chip	880	6500	59000
Componentes	Elementos PCB	571,21	3922,65	34525,52
PCB	Costo	100	500	3000
Encapsulado		500	22000	30000
Costo mano de obra	Ensamble Producto	400	4000	40000
Sub TOTAL		2451,21	36922,65	166525,52
Factor de Riesgo	Aduanas, envios, impuestos	1,3	1,3	1,3
TOTAL		3186,573	47999,445	216483,176
Costo Unitario Total		**31,86573**	**47,999445**	**21,6483176**

Tabla 11: Costos en dólares de componentes.

9. Conclusiones

En esta sección se presentan las conclusiones finales analizando el objetivo planteado inicialmente, los conocimientos adquiridos y las reflexiones personales.

9.1 Generales

El objetivo principal de este proyecto se cumplio, realizando de manera completa un desarrollo de producto. Esto implica diferentes tareas desde el estudio del problema, la planificación del producto, hasta la culminación de un prototipo final.

Este desarrollo tiene una cuota importante de innovación, generando conocimientos de esta tecnología inalámbrica emergente a nivel regional.

Por otra parte, se supieron sobrellevar los diferentes inconvenientes que aparecieron durante el proceso de ejecución del proyecto, siendo la más importante la situación mundial producida por el virus COVID-19. Esto generó atrasos en los envíos internacionales y grandes dificultades, debido a que las reuniones del grupo debieron ser mediante plataformas de videoconferencia.

9.2 Académicas

A nivel académico, se lograron emplear diferentes conocimientos aprendidos a lo largo de la carrera universitaria así como aplicarlos a nuevos desafíos, como lo son el diseño de un PCB, el diseño de un encapsulado, la programación de un microchip, el desarrollo de un servidor web con manejo de base de datos y la implementación de una red LoRaWAN completamente funcional.

También se adquirieron conocimientos en el estudio de viabilidad de un proyecto, para de esta manera realizar un correcto análisis económico del mismo.

Por otra lado, el proyecto motivó e impulsó el área de innovación, buscando diferentes maneras de solucionar los problemas que surgían durante la implementación. Un claro ejemplo de esto fue buscar la manera de poder decidir si la luz estaba encendida o no, investigando y testeando cuál era la manera más efectiva para lograr esto.

9.3 Reflexiones personales

Como primera reflexión grupal creemos que este proyecto nos llevó a aplicar los conocimientos que fuimos adquiriendo durante todo el ciclo universitario, convirtiéndolo en la tarea más importante de la carrera.

Creemos que la Universidad nos brindó las herramientas principales para poder realizar este proyecto con éxito y luego está en los estudiantes superarse para poder realizarlo lo más satisfactoriamente posible.

Particularmente en este proyecto de grado, como grupo sentimos que los objetivos se cumplieron de manera exitosa. Sin embargo, siendo meticulosos creemos que hay varios puntos donde podríamos mejorar, si este proyecto finalmente se lleva a cabo.

Por otra parte, siempre nos motivó poder realizar un desarrollo completo de un producto para culminar nuestra carrera universitaria. Por este motivo la predisposición siempre fue positiva, haciendo que el desarrollo del proyecto sea más accesible.

También, creemos que este proyecto tiene un gran potencial, debido a que con un costo relativamente bajo, se puede realizar un sistema de monitoreo de áreas muy amplias y que no tengan acceso a energía eléctrica. Un claro ejemplo de otra aplicación donde se puede aplicar este desarrollo, es la medición del nivel de agua en los arrozales

En lo académico también adquirimos conocimientos en distintas áreas como son las tecnologías inalámbricas, instalación de componentes, mediciones transitorias de consumo.

9.4 Cronograma de proyecto

En la siguiente imagen se visualiza en color azul el cronograma tentativo inicial y luego en color verde el que realmente se adaptó al proyecto.

En consecuencia a los inconvenientes mencionados de situación de emergencia sanitaria, el proyecto se aplazó aproximadamente seis meses, teniendo una duración total de aproximadamente un año y dos meses.

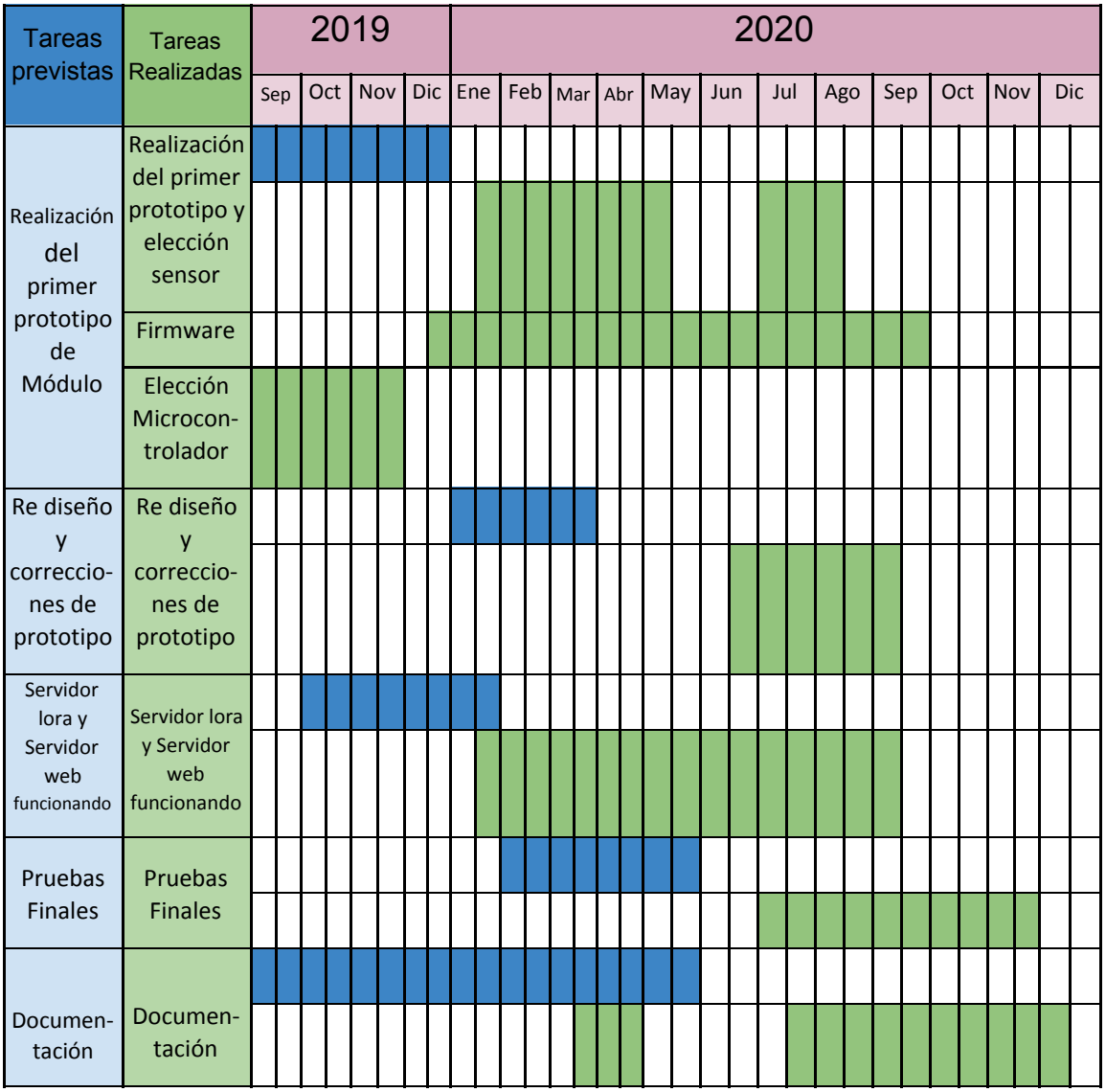

Tabla 12: Cronograma del proyecto.

10. Trabajos a futuro

En este proyecto de grado al ser un desarrollo de producto, hay varios aspectos que obviamente son necesarios mejorar o profundizar si el mismo se llevara a cabo. En esta sección se explicarán algunas áreas donde se cree necesario implementar determinadas mejoras.

10.1 Seguridad y almacenamiento de la información

El primer punto relevante que es necesario perfeccionar es la seguridad de la información, para esto se propone crear un servidor privado que cumpla el rol que actualmente cumple TTN, evitando pasar los datos por internet. A pesar de que los datos están encriptados, este punto puede ser un posible punto de falla de seguridad.

Es preferible manejar los datos a un nivel local dentro del aeropuerto y en caso de querer acceder desde internet hacer un sitio web que tenga la información duplicada o a través de algún tipo de VPN.

Esta modificación permitirá hacer más rápido y robusto el producto, tanto en seguridad como en funcionamiento. De esta forma no se depende de un servidor externo del que no se tiene el control.

10.2 Automatización instalación

Actualmente el microcontrolador es únicamente un sensor de telemetría. Si el proyecto se realizará, un posible trabajo a futuro es la implementación de un modo que te permita automatizar la instalación, de manera que ayude al operario a orientar y calibrar el nodo.

Por otra parte, es importante que este modo permita fácilmente el ingreso del nodo al sistema, por ejemplo a través de un código QR.

10.3 Comunicación con el nodo

Como se mencionó en la sección 5.3.1 Descripción de bloques se decidió hacer unidireccional la comunicación entre nodo y el servidor.

La unidireccionalidad en la comunicación genera inconvenientes al momento de querer interactuar con el sensor. Ya sea porque se quiere cambiar el valor de una variable o reconfigurar el tiempo del modo sleep.

La capacidad del nodo de poder recibir datos podría permitir a futuro evaluar la posibilidad de actualizar el firmware remotamente. Esto requiere de una memoria con suficiente espacio para almacenar ambos códigos al mismo tiempo.

Actualmente se está utilizando un 43.5% de la memoria destinada para el programa, sin embargo al agregar el modo de instalación suponemos que se va a superar el 50% del código dificultando la actualización de firmware remotamente. Esto se podría solucionar con una memoria eeprom de un tamaño de 128 kb en donde se almacenaria temporalmente el firmware.

10.4 Cambios en el circuito

Antes de proceder a hacer una nueva versión de PCB se tendrían que hacer dos cambios en el circuito esquemático, que resultaron en el desarrollo del proyecto.

El primer cambio sería eliminar el circuito auxiliar para poder controlar el sensor de luz BH1750, ya que se comprobó que es posible hacerlo directamente desde una salida digital del microcontrolador.

El cambio en el circuito del BH1750 recién mencionado se puede visualizar en el siguiente esquemático.

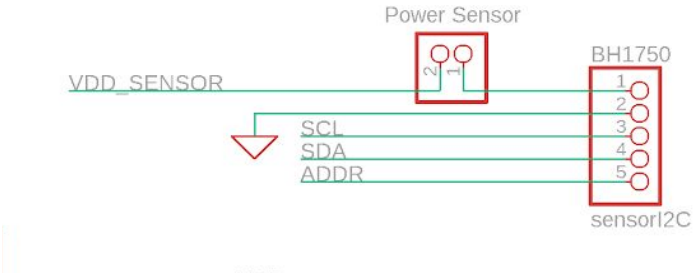

Imagen 85: Circuito esquemático con modificación en circuito de batería y circuito del BH1750

El segundo cambio sería modificar el circuito de medición de batería, como se habló en la sección 5.1.4 Medición de batería agregando un transistor NMOS para poder garantizar el apagado del circuito.

10.5 Rediseñar el encapsulado

En el caso de continuar con el proyecto y llevarlo a cabo, habría que evaluar si es necesario realizar un molde dependiendo de las cantidades que se desean producir.

Por otra parte, se debería mejorar la estanqueidad del producto hasta un IP66 en cualquiera de las dos formas de producción ya sea por impresión 3D o molde, ya que este producto se va a enfrentar durante un tiempo prolongado a diversas condiciones climáticas.

A su vez otro factor importante para los envolventes plásticos es la degradación ante la luz ultravioleta, ya que no todos los plásticos se degradan igual de rápido. Siendo unos plásticos más resistentes a la radiación.

11. Bibliografía

[1] “What is LoraWAN”,

https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf

[2] “LPWAN Technologies Comparison”,

https://behrtechnologies.com/blog/tag/lpwan-comparison/

[3] “SigFox Technology”,

https://www.sigfox.com/en/what-sigfox/technology

[4] “Google Maps”,

https://www.google.es/maps/preview

[5] “Transceptores SX12XX”,

https://www.rfsolutions.co.uk/downloads/1537522406DS_SX1261-2_V1.1_SEMTECH.pdf

[6] “LoPy4”,

https://pycom.io/product/lopy4/

[7] “RFM95/96/97/98(W) - Low Power Long Range Transceiver Module ”,

https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_9W.pdf

[8] “Moteino MEGA”,

https://lowpowerlab.com/shop/product/119

[9] “SAM R34/R35 Low Power LoRa® Sub-GHz SiP Datasheet”,

https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-R34-R35-Low-Power-LoRa-Sub-GHz-SiP-Data-Sheet-DS70005356C.pdf

[10] “RAKWireless Documentation”,

https://docs.rakwireless.com/

[11] “RAK2245 Pi HAT WisLink LPWAN Concentrator”,

https://docs.rakwireless.com/Product-Categories/WisLink/RAK2245-Pi-HAT/

[12] “RHF0M301”,

https://www.robotshop.com/media/files/pdf/915mhz-lora-gateway-raspberry-pi-hat-datasheet1.pdf

[13] “iC880A-SPI LoRa® Concentrator”,

https://wireless-solutions.de/products/lora-solutions-by-imst/radio-modules

[14] “RAK2245 Quick Start Guide”,

https://docs.rakwireless.com/Product-Categories/WisLink/RAK2245-Pi-HAT/Quickstart/

[15] “Datasheet Gy-49”,

https://datasheets.maximintegrated.com/en/ds/MAX44009.pdf

[16] “Datasheet VELM6070”,

https://www.vishay.com/docs/84310/designingveml6070.pdf

[17] “Datasheet TEMT6000”,

https://www.vishay.com/docs/81579/temt6000.pdf

[18] “Datasheet OPT3001”,

https://www.ti.com/lit/ds/symlink/opt3001.pdf?ts=1607292667561&ref_url=https%253A%252F%252Fwww.google.com%252F

[19] “Cálculo del Presupuesto de Potencia”,

https://slideplayer.es/slide/538740/

[20] “RAK 860-930MHz 3dBi Fiberglass Antenna Specification”,

https://downloads.rakwireless.com/Accessories/Antenna/Fiberglass-Antenna/RAK_860-930MHz_3dBi_Fiberglass_Antenna_Specification.pdf

[21] “Datasheet ANT-916-CW-RCS”,

https://linxtechnologies.com/wp/wp-content/uploads/ant-916-cw-rcs.pdf

[22] “IoT Accessories by RAK”,

https://store.rakwireless.com/collections/accessories

[23] “Programador J-Link”,

https://www.segger.com/products/debug-probes/j-link/

“Logo Atemel Studio”,

https://shareappscrack.com/atmel-studio/?token=81045790

[24] ”Fundamentos del I2C”,

https://teslabem.com/nivel-intermedio/fundamentos/

[25] “Sensor BH1750”,

https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf

[26] ”The Things Network”,

https://console.thethingsnetwork.org/

[27] “Luminaria FEA-1200”,

https://www.ti.com/lit/ds/symlink/opt3001.pdf?ts=1607292667561&ref_url=https%253A%252F%252Fwww.google.com%252F

[28] “TTN Mapper”,

ttnmapper.org

[29] “Tabla grados IP”,

https://es.slideshare.net/CristianMendez2/tabla-grados-ip

[30] “Duracell MN2400 Size: AAA (LR03)”,

https://www.duracell.com/wp-content/uploads/2020/02/MN24US1119.pdf

12. Glosario

ADC	Analogic to Digital Converter, convertidor analógico digital.
ADDR	Dirección de esclavo.
AES	Protocolo de encriptación.
ALOHA	Protocolo de identificación.
BLE	Bluetooth low energy.
BOM	Bill of materials, lista de materiales.
Debug	Métodos para encontrar errores y testeo.
Downlink	Mensaje desde el gateway al nodo.
Dropout	Voltaje de caída en regulador de voltaje.
EUI	Número de identificación.
FCC	Comisión Federal de Comunicaciones.
Firmware	Programa orientado al microcontrolador.
Footprint	Diseño de superficie donde se sueldan los componentes.
FSK	Modulación de frecuencia.
FTDI	Conversor de protocolo usb a serial.
Gateway	Puerta de enlace.
Hardware	Parte física de los elementos eléctricos.
Hércules	Herramienta de análisis de puerto serial.
HTTP	Lenguaje de desarrollo de sitio web.
I2C	Protocolo de comunicación maestro esclavo.
IDE Arduino	Entorno de programación de firmware.
IEC	Organismo generador de normas.
IEEE	Organismo generador de normas.
IoT	Internet of things, internet de las cosas.
iPEX	Tipo de conector de la antena.
J-link	Puerto de programación de microcontrolador.

Javascript	Lenguaje de programación para desarrollo en sitios web.
Join	Mensaje para vincularse a la red LoRa.
Jumper	Elemento de circuito que permite abrir o cerrar circuito.
LDO	Regulador lineal de voltaje.
LiPo	Tecnología de batería de Litio polímero.
Litio	Tecnología de batería de Litio.
Modo sleep	Modo en el que un microcontrolador reduce sus funciones y consumo de batería.
MQTT	Protocolo de suscripción de LoRa.
MySQL	Protocolo de gestión de base de datos.
NMOS - PMOS	Transistor, Negative/Positive-channel Metal-Oxide Semiconductor.
Payload	Parte útil del mensaje LoRa.
PCB	Placa de circuito impreso.
PVC	Plástico, Cloruro de polivinilo.
Python	Lenguaje de programación de alto nivel.
QR	Imagen que asocia una imagen con una dirección web.
RAKWireless	Empresa proveedora de tecnología inalámbrica.
RAM	Tipo de memoria volátil.
SCADA	Supervisión, Control y Adquisición de Datos.
SiP	System in Package.
SMA	Tipo de conector de antena.
Transceptor	Adaptador capaz de enviar y recibir mensajes.
Uplink	Mensaje desde el nodo al gateway.
VPN	Red privada virtual.

13. Anexo

13.1 Firmware Nodo

13.1.1 Board_init.c

```
#include <compiler.h>
#include <board.h>
#include <conf_board.h>
#include <port.h>
#include <stdint.h>
#include "asf.h"


#if defined(_GNUC_)
void board_init(void) WEAK_attribute_((alias("system_board_init")));
#elif defined(_ICCARM_)
void board_init(void);
#  pragma weak
board_init=system_board_init #endif

void system_board_init(void)
{
    struct port_config pin_conf;
    port_get_config_defaults(&pin_conf);


    /* Configure LED Blue as outputs, turn them on */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(PIN_PA27, &pin_conf);
    port_pin_set_output_level(PIN_PA27, false);

    /* Configure LED Yellow as outputs, turn them on
    */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(PIN_PB22, &pin_conf);
    port_pin_set_output_level(PIN_PB22, false);

    /* Configure LED Red as outputs, turn them on */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(PIN_PA09, &pin_conf);
    port_pin_set_output_level(PIN_PA09, false);

    /* Configure Enable Bat as outputs, turn them off */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(PIN_PA06, &pin_conf);
    port_pin_set_output_level(PIN_PA06, false);

    /* Configure I2C Enable as outputs, turn them off
    */ pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(PIN_PA14, &pin_conf);
```

```
    port_pin_set_output_level(PIN_PA14, true);


    ///*Configure Button as input, pull up */
    pin_conf.direction = PORT_PIN_DIR_INPUT;
    pin_conf.input_pull = PORT_PIN_PULL_UP;
    pin_conf.powersave = false;
    port_pin_set_config(PIN_PA08, &pin_conf);
#ifdef RFSWITCH_ENABLE
    /* Configure RFSWITCH as output */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(RF_SWITCH_PIN, &pin_conf);
    port_pin_set_output_level(RF_SWITCH_PIN, RF_SWITCH_INACTIVE);
 #endif

 #ifdef TCXO_ENABLE
    /* Configure TXPO PWR as output */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(TCXO_PWR_PIN, &pin_conf);
    port_pin_set_output_level(TCXO_PWR_PIN, TCXO_PWR_INACTIVE);
 #endif
    /* Configure RFSWITCH SKY13373 PWR as output */
    pin_conf.direction = PORT_PIN_DIR_OUTPUT;
    port_pin_set_config(RFSW_PWR_PIN, &pin_conf);
    port_pin_set_output_level(RFSW_PWR_PIN,
    RFSW_PWR_INACTIVE);
```

13.1.2 conf_app.h

```
#ifndef APP_CONFIG_H_
#define APP_CONFIG_H_

/***************************** INCLUDES **********************************/

/***************************** MACROS ************************************/
/* Number of software timers */
#define TOTAL_NUMBER_OF_TIMERS          (25u)


/*Define the Sub band of Channels to be enabled by default for the
application*/ #define SUBBAND 2
#if ((SUBBAND < 1 ) || (SUBBAND > 8 ) )
#error " Invalid Value of
Subband" #endif

/* Activation method constants */
#define OVER_THE_AIR_ACTIVATION         LORAWAN_OTAA
#define ACTIVATION_BY_PERSONALIZATION   LORAWAN_ABP

/* Message Type constants */
#define UNCONFIRMED                     LORAWAN_UNCNF
#define CONFIRMED                       LORAWAN_CNF
```

```
/* Enable one of the activation methods */
#define DEMO_APP_ACTIVATION_TYPE              OVER_THE_AIR_ACTIVATION
//#define DEMO_APP_ACTIVATION_TYPE              ACTIVATION_BY_PERSONALIZATION

/* Select the Type of Transmission - Confirmed(CNF) /
Unconfirmed(UNCNF) */ #define DEMO_APP_TRANSMISSION_TYPE
UNCONFIRMED
//#define DEMO_APP_TRANSMISSION_TYPE           CONFIRMED

/* FPORT Value (1-255) */
#define DEMO_APP_FPORT                         1

/* Device Class - Class of the device (CLASS_A/CLASS_C)
*/#define DEMO_APP_ENDDEVICE_CLASS              CLASS_A
//#define DEMO_APP_ENDDEVICE_CLASS                CLASS_C


/* ABP Join Parameters */

#define DEMO_DEVICE_ADDRESS                     0x2601149D
 #define DEMO_APPLICATION_SESSION_KEY           { 0x58, 0x93, 0x81, 0xD6,
   0x80, 0x97, 0xCB, 0xA1, 0xF1, 0xB6, 0x7D, 0xC6, 0x56, 0x1D, 0x36, 0x45
   }
 #define DEMO_NETWORK_SESSION_KEY               { 0x4F, 0xC8, 0x7C, 0xC5,
   0xF8, 0x71, 0xD0, 0xFB, 0x40, 0x88, 0x25, 0x4D, 0x8D, 0x0B, 0x5A, 0x3A
   }

/* OTAA Join Parameters */

 #define DEMO_DEVICE_EUI                        { 0x00, 0xC0, 0x6B, 0xC3,
   0xCB, 0x93, 0x39, 0xC8 }


 #define DEMO_APPLICATION_EUI                   { 0x70, 0xB3, 0xD4, 0x7E,
   0xD6, 0x02, 0x9E, 0xC1 }
 #define DEMO_APPLICATION_KEY                   { 0x36, 0xB3, 0xB4, 0x4B,
   0xDD, 0x89, 0xCF, 0xA8, 0xDD, 0x94, 0xA6, 0x57, 0x8B, 0x1C, 0x03, 0xC2
   }

/* Multicast Parameters */
#define DEMO_APP_MCAST_ENABLE                   true
#define DEMO_APP_MCAST_GROUP_ADDRESS            0x0037CC56
 #define DEMO_APP_MCAST_APP_SESSION_KEY         {0x2B, 0x4E, 0x15, 0x16,
   0x22, 0xAE, 0xD2, 0xA6, 0x2B, 0x7E, 0x15, 0x16, 0x28, 0xAE, 0xD2,
   0xA6}
 #define DEMO_APP_MCAST_NWK_SESSION_KEY         {0x3C, 0x1F, 0x26, 0x23,
   0x39, 0xBF, 0xE3, 0xB7, 0xBC, 0x18, 0x26, 0x99, 0x1C, 0xD0, 0x50,
   0x4D}

/* This macro defines the application's default sleep duration in
milliseconds */

#define DEMO_CONF_DEFAULT_APP_SLEEP_TIME_MS     5000

#endif /* APP_CONFIG_H_ */
```

13.1.3 main.c

```
/************************ INCLUDES*************************************/
#include "system_low_power.h"
#include "radio_driver_hal.h"
#include "lorawan.h"
#include "sys.h"
#include "system_init.h"
#include "system_assert.h"
#include "aes_engine.h"
#include "enddevice_demo.h"
#include "sio2host.h"
#include "extint.h"
#include "conf_app.h"
#include "sw_timer.h"
#include "nvm.h"
#include "pmm.h"
#include  "conf_pmm.h"
#include "sleep_timer.h"
#include "sleep.h"
#include "Initial_funcion.c"
#include "conf_sio2host.h"

/************************** Macro definition ******************************/
/* Button debounce time
in ms */ #define
APP_DEBOUNCE_TIME

/************************** Global variables ******************************/
bool button_pressed = false;
bool factory_reset
= false; bool
bandSelected =
false; uint32_t
longPress = 0;
uint8_t lTimerId =
0xFF; extern bool
certAppEnabled;
bool deviceResetsForWakeup = false;

/************************** Main ******************************/
int main(void)
{
     inicializaciones();
     mote_de
     mo_init
     ();
     while
     (1)
     {
         SYSTEM_RunTasks();
     }
}
```

13.1.4 enddevice.c

```
/****************************** INCLUDES
************************************/ #include "asf.h"
#include "temp_sensor.h"
#include "conf_app.h"
#include <sercom.h>
#include <pinmux.h>
#include "lorawan.h"
#include "system_task_manager.h"
#include "enddevice.h"
#include "conf_app.h"
#include "sio2host.h"
#include "resources.h"
#include "delay.h"
#include "sw_timer.h"
#include "LED.h"
#include "pmm.h"
#include "radio_driver_hal.h"
#include  "conf_pmm.h"
#include "conf_sio2host.h"
#include "pds_interface.h"
#include "I2C_nodo.c"

#if (EDBG_EUI_READ == 1)
#include
"edbg_eui.h
" #endif
#include
"atomic.h
"
#include
<stdint.h
>
/****************************** MACROS
*************************************/
/************************** GLOBAL VARIABLES
*********************************/ static bool joined = false;
static char
temp_sen_str[25];
static uint8_t
data_len = 0; bool
certAppEnabled =
false;

static volatile uint8_t appTaskFlags = 0x00u;
/* Default Regional band start
delay time */ volatile static
uint8_t count = 5;

extern uint8_t
demoTimerId;
extern uint8_t
```

```
lTimerId;
static AppTaskState_t appTaskState;


uint8_t bandTable[] =
{
0xFF,
#if (NA_BAND == 1)
ISM_NA915,
#endif
0xFF,
0xFF
};

/*ABP Join Parameters */
static uint32_t demoDevAddr = DEMO_DEVICE_ADDRESS;
static uint8_t demoNwksKey[16] = DEMO_NETWORK_SESSION_KEY; static uint8_t
demoAppsKey[16] = DEMO_APPLICATION_SESSION_KEY;
/* OTAA join parameters */
static uint8_t demoDevEui[8] =  DEMO_DEVICE_EUI;
static uint8_t demoAppEui[8] = DEMO_APPLICATION_EUI;
static uint8_t demoAppKey[16] = DEMO_APPLICATION_KEY;

static LorawanSendReq_t lorawanSendReq;

/* Muticast Parameters */
static bool demoMcastEnable = DEMO_APP_MCAST_ENABLE;
static uint32_t  demoMcastDevAddr = DEMO_APP_MCAST_GROUP_ADDRESS;
static uint8_t demoMcastNwksKey[16] = DEMO_APP_MCAST_NWK_SESSION_KEY;
static uint8_t demoMcastAppsKey[16] = DEMO_APP_MCAST_APP_SESSION_KEY;


//enum status_code status_test;
/************************** EXTERN VARIABLES **************************/
extern bool button_pressed;
extern bool factory_reset;
extern bool bandSelected;
extern uint32_t longPress;


static void
sendData(void); static
void sleep(void);
static void appPostTask(AppTaskIds_t id);
static SYSTEM_TaskStatus_t
(*appTaskHandlers[])(void); static
SYSTEM_TaskStatus_t processTask(void);
static void processJoinAndSend(void);
static void appWakeup(uint32_t
sleptDuration); static  void
app_resources_uninit(void); static
void dev_eui_read(void);
```

```
/************************** FUNCTION PROTOTYPES ***********************/
SYSTEM_TaskStatus_t APP_TaskHandler(void);
static float convert_celsius_to_fahrenheit(float cel_val);
/*****************************************************************//*
 \brief     Function that processes the Rx data
 \param[in] data - Rx data payload
 \param[in] dataLen - The number of Rx bytes
 ***************************************************************
*********/ static void demo_handle_evt_rx_data(void *appHandle,
appCbParams_t *appdata);

/***************************** FUNCTIONS *******************************
static SYSTEM_TaskStatus_t (*appTaskHandlers[APP_TASKS_COUNT])(void) = {
/* In the order of descending priority */
processTask
};



/*****************************************************************//**
\brief    Calls appropriate functions based on state variables
 ***************************************************************
*********/ static SYSTEM_TaskStatus_t processTask(void)
{
processJoinAndSend();
return
SYSTEM_TASK_SUCCESS;
}



/*****************************************************************//**
\brief    Sends Join request or Data to the network
 ***************************************************************
*********/ static void processJoinAndSend(void)
{
StackRetStatus_t status = LORAWAN_SUCCESS;
if(!joined)
{
        status = LORAWAN_Join(DEMO_APP_ACTIVATION_TYPE);

        if (LORAWAN_SUCCESS == (StackRetStatus_t)status)
        {
            printf("\nJoin Request Sent\n\r");

        }
        else
        {
            print_stack_status(status); appTaskState = JOIN_SEND_STATE;
            appPostTask(PROCESS_TASK_HANDLER);
        }
}
```

```
if(joined)
{
        port_pin_set_output_level(PIN_PA09, false);// luz roja sleep();

}
else if(!joined)
{
printf("Device not joined to the network\r\n"); delay_ms(30000); //30s
        appTaskState = JOIN_SEND_STATE;
        appPostTask(PROCESS_TASK_HANDLER);
}

 }



 /*****************************************************************//**
 \brief    Initialization the Demo application
 *************************************************************
 *********/ void mote_demo_init(void)
 {
//bool status = false;
/* Initialize ADC temp sensor */
resource_init();
/* Read DEV EUI from EDBG
*/ dev_eui_read();
/* Initialize the LORAWAN Stack */
     LORAWAN_Init(demo_appdata_callback,  demo_joindata_callback);

       //defina el join y el send
printf("\n\n\r*************************************************\n\
r"); printf("\n\rMicrochip LoRaWAN Stack %s\r\n",STACK_VER);
printf("\r\nInit -
Successful\r\n");
LORAWAN_Reset(ISM_NA915);
mote_set_parameters(ISM_NA915);

 }



 /*****************************************************************//*
 \brief Callback function for the ending of Activation procedure
  *************************************************/
  void demo_joindata_callback(bool status)
  //funcion del join
 {
/* This is called every time the join process is finished */

if(true == status)
{
        uint32_t devAddress;
        bool mcastEnabled;
```

```
        joined = true;
        printf("\nJoining Successful\n\r");
        LORAWAN_GetAttr(DEV_ADDR, NULL, &devAddress);
        LORAWAN_GetAttr(MCAST_ENABLE, NULL, &mcastEnabled);

        if (devAddress  != DEMO_APP_MCAST_GROUP_ADDRESS)
        {
            printf("\nDevAddr:  0x%lx\n\r", devAddress);
        }
        else if ((devAddress == DEMO_APP_MCAST_GROUP_ADDRESS) && (true ==
          mcastEnabled))
        {
            printf("\nAddress conflict between Device Address and Multicast
              group address\n\r");
        }
        print_application_config();
}
else
{
        joined = false;
        printf("\nJoining Denied\n\r");
}
printf("\n\r************************************************************\n\r");

appPostTask(PROCESS_TASK_HANDLER);

 }


  /*****************************************************************//*
\brief Function to send data from end device to application serve
/*****************************************************************/

void sendData(void)
 {
uint16_t adc_value;
int status = -1;
uint16_t Value;
char Value_low,Value_high,adc_value_high,adc_value_low;

adc_value = get_sensor_data_PA07();
configure_i2c();
Value = read_modulo_I2C();

printf("apagar modulos");
port_pin_set_output_level(PIN_PA27, false);// luz azul
//port_pin_set_output_level(PIN_PA14, false);// bh1750
port_pin_set_output_level(PIN_PA06, false);// medidor bat

printf("ADC : %d I2C : %d\n\r", adc_value, Value);
Value_high = (char)(Value/256);
Value_low = (char)((Value - Value_high*256));
adc_value_high = (char)(adc_value/256);
```

```
adc_value_low = (char)((adc_value - adc_value_high*256));
snprintf(temp_sen_str,sizeof(temp_sen_str),"@%c%c%c%c
       \n",Value_high,Value_low,adc_value_high,adc_value_low);

data_len = 6;//strlen(temp_sen_str);
lorawanSendReq.buffer = &temp_sen_str;
lorawanSendReq.bufferLength = data_len - 1;
lorawanSendReq.confirmed =
DEMO_APP_TRANSMISSION_TYPE; lorawanSendReq.port
= DEMO_APP_FPORT;
status = LORAWAN_Send(&lorawanSendReq);
if (LORAWAN_SUCCESS == status)
{
         printf("\nTx Data Sent \r\n");
         printf(temp_sen_str);


}
else
{
         print_stack_status(status);
         appTaskState = JOIN_SEND_STATE;
         appPostTask(PROCESS_TASK_HANDLER);
}

 }


 static void appWakeup(uint32_t sleptDuration)
 {

HAL_Radio_resources_init();
sio2host_init();
printf("\r\nsleep_ok %ld ms\r\n", sleptDuration);
printf("encender modulos");
port_pin_set_output_level(PIN_PA27, true);// luz azul
port_pin_set_output_level(PIN_PA09, true);// luz roja
port_pin_set_output_level(PIN_PA06, true);// medidor bat
sendData();
 }



 /*************************************************************************//*
  \brief     Set join parameters function
  \param[in] activation type - notifies the activation type (OTAA/ABP)
  \return    LORAWAN_SUCCESS, if successfully set the
             join parameters LORAWAN_INVALID_PARAMETER,
             otherwise
  ************************************************************************
 */ StackRetStatus_t set_join_parameters(ActivationType_t
 activation_type)
 {
StackRetStatus_t status;
```

```
printf("\n********************Join Parameters********************\n\r");
if(ACTIVATION_BY_PERSONALIZATION == activation_type)
{
         status = LORAWAN_SetAttr (DEV_ADDR,
         &demoDevAddr); if (LORAWAN_SUCCESS ==
         status)
         {
             status = LORAWAN_SetAttr (APPS_KEY, demoAppsKey);
         }

         if (LORAWAN_SUCCESS == status)
         {
             printf("\nAppSessionKey : ");

             print_array((uint8_t *)&demoAppsKey, sizeof(demoAppsKey));
             status = LORAWAN_SetAttr (NWKS_KEY, demoNwksKey)
       }

         if (LORAWAN_SUCCESS == status)
         {
              printf("\nNwkSessionKey : ");

              print_array((uint8_t *)&demoNwksKey, sizeof(demoNwksKey));
          }
          else
          {
                 status = LORAWAN_SetAttr (DEV_EUI, demoDevEui); if
                 (LORAWAN_SUCCESS == status)
                 {
                     printf("\nDevEUI : ");

                     print_array((uint8_t *)&demoDevEui,
                     sizeof(demoDevEui)); status =
                     LORAWAN_SetAttr (APP_EUI, demoAppEui);
                 }

                 if (LORAWAN_SUCCESS == status)
                 {
                     printf("\nAppEUI : ");

                     print_array((uint8_t *)&demoAppEui,
                     sizeof(demoAppEui)); status =
                     LORAWAN_SetAttr (APP_KEY, demoAppKey);
                 }
         }

         if (LORAWAN_SUCCESS == status)
         {
             printf("\nAppKey : ");

             print_array((uint8_t *)&demoAppKey, sizeof(demoAppKey));
         }
}
return status;
```

```
    }
    ***********************************************************************//*
     \brief      Reads the DEV EUI if it is flashed in EDBG MCU
     ************************************************************
    *********/ static void dev_eui_read(void)
    {
    #if (EDBG_EUI_READ == 1)
   uint8_t
   invalidEDBGDevEui[8];
   uint8_t EDBGDevEUI[8];
   edbg_eui_read_eui64((uint8_t *)&EDBGDevEUI);
   memset(&invalidEDBGDevEui, 0xFF, sizeof(invalidEDBGDevEui));
   /* If EDBG doesnot have DEV EUI, the read value will be of all 0xFF,
           Set devEUI in conf_app.h in that case */
   if(0 != memcmp(&EDBGDevEUI, &invalidEDBGDevEui, sizeof(demoDevEui)))
   {
    /* Set EUI addr in EDBG if there */
    memcpy(demoDevEui, EDBGDevEUI, sizeof(demoDevEui));
}
 #endif
 }


 static void sleep(void)
 {
static bool deviceResetsForWakeup = false;
PMM_SleepReq_t sleepReq;
/* Put the application to sleep */
sleepReq.sleepTimeMs = DEMO_CONF_DEFAULT_APP_SLEEP_TIME_MS;
sleepReq.pmmWakeupCallback = appWakeup;
sleepReq.sleep_mode = CONF_PMM_SLEEPMODE_WHEN_IDLE;
if (CONF_PMM_SLEEPMODE_WHEN_IDLE == SLEEP_MODE_STANDBY)
{
        deviceResetsForWakeup = false;
}
if (true == LORAWAN_ReadyToSleep(deviceResetsForWakeup))
{
        app_resources_uninit();
        if (PMM_SLEEP_REQ_DENIED == PMM_Sleep(&sleepReq))
        {
            HAL_Radio_resources_init();
            sio2host_init();
            appTaskState = JOIN_SEND_STATE;
            appPostTask(PROCESS_TASK_HANDLER);
            printf("\r\nsleep_not_ok\r\n");
            }
}
else
{
        printf("\r\nsleep_not_ok\r\n");
        appTaskState = JOIN_SEND_STATE;
        appPostTask(PROCESS_TASK_HANDLER
        );
```

```
}
 }
```

13.1.5 I2C_nodo.c

```
#include
"i2c_common.h"
#include
"i2c_master.h"
#include "i2c_master_interrupt.h"

static void  configure_i2c(void);
static uint16_t
read_modulo_I2C(void);

/*I2C GLOBAL VARIABLES*/
struct i2c_master_module
i2c_master_instance_module; struct
i2c_master_packet master_packet_port;

static void configure_i2c(){
     /*I2C inicializacion*/
     struct i2c_master_config config_i2c_port;
     i2c_master_get_config_defaults(&config_i2c_port);
     config_i2c_port.baud_rate = I2C_MASTER_BAUD_RATE_100KHZ;
     config_i2c_port.buffer_timeout = 65535;
     config_i2c_port.generator_source = GCLK_GENERATOR_0;
     config_i2c_port.inactive_timeout = I2C_MASTER_INACTIVE_TIMEOUT_DISABLED;
     config_i2c_port.pinmux_pad0 = PINMUX_PA16C_SERCOM1_PAD0;
     //SDA PA16 SERCOM1 PAD0 C
     config_i2c_port.pinmux_pad1 = PINMUX_PA17C_SERCOM1_PAD1;//SCL
     config_i2c_port.run_in_standby = false;
     config_i2c_port.scl_low_timeout = false;
     config_i2c_port.sda_scl_rise_time_ns = 215;
     config_i2c_port.start_hold_time = I2C_MASTER_START_HOLD_TIME_300NS_600NS;
     config_i2c_port.unknown_bus_state_timeout = 65535;
     /* Initialize and enable device with config */
     i2c_master_init(&i2c_master_instance_module, EXT1_I2C_MODULE,
     &config_i2c_port); i2c_master_enable(&i2c_master_instance_module);
     /*Fin configuracion*/
}

/*****************************************************************//**
\brief      Reads back the device MAC address stored in User page of EDBG
\param[in]  eui - Device EUI read back from EDBG(8 bytes)
*****************************************************************
***/ static uint16_t read_modulo_I2C(void)
{
```

```
uint32_t timeout = 0;

uint8_t

kit_data[2];

static uint8_t write_buffer[1] = {
0x23,
};

/** Send the request token */
master_packet_port.address = SLAVE_ADDRESS;
master_packet_port.data_length =
sizeof(write_buffer);
master_packet_port.data = write_buffer;
master_packet_port.ten_bit_address = false;
master_packet_port.high_speed = false;
master_packet_port.hs_master_code = 0x0;

 while (i2c_master_write_packet_wait(&i2c_master_instance_module,
   &master_packet_port) !=
STATUS_OK) {
/* Increment timeout counter and check if timed
out. */ if (timeout++ == TIMEOUT) {

printf("\r\n Timeout1");

return 0;
}
}


/** Get the extension boards info */
master_packet_port.data_length    = 2;
master_packet_port.data           = kit_data;
while (i2c_master_read_packet_wait(&i2c_master_instance_module,
&master_packet_port) ! = STATUS_OK) {
/* Increment timeout counter and check if timed out. */
if (timeout++ == TIMEOUT) {
return 0;
}
}
printf("HIgh%x,",master_packet_port.data[0]);
printf("Low%x\n",master_packet_port.data[1]);
return master_packet_port.data[0]*255+master_packet_port.data[1];
}
```

13.2 Software servidor

```
import time
import threading
import ttn
import serial
import MySQLdb
import datetime

#Funcion Interrupcion Serial

def Serial_Interrupt():



    global estado global serial_last
    print('Actualizacion Serial') ard.write(b'\n') time.sleep(1)
    lectura = ard.read(ard.inWaiting()) if (lectura[0] == 64):

        estado = [lectura[1]-48, lectura[2]-48, lectura[3]-48,
        lectura[4]-48, lectura[5]-48]

        serial_last = datetime.datetime.now()

    else :

        estado = [-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]

    threading.Timer(5,Serial_Interrupt).start()



#Funcion Interrupcion TTN

def uplink_callback(msg, client):



  print("Divice from ", msg.dev_id) print(msg)




  valorI2C = msg.payload_fields.lux nroHardware = msg.hardware_serial

  estadoBat = msg.payload_fields.bat

  print("Nro Hardware",nroHardware)

  print("Sensor digital I2C",valorI2C)
```

```
print("Estado Bateria  ", round(estadoBat,2))


# ABRIR CONEXION BASE DE DATOS Y ENVIO DATO

db =
MySQLdb.connect("www.electronicaboost.com.uy","hboost_python","TesisLora
","hboost_Base_de_D cursor = db.cursor()

consulta= "SELECT * FROM `Identificacion` WHERE `Numero_Hardware` LIKE
'" + str(nroHardware) +"' cursor.execute(consulta)

#respuesta base de datos

records = cursor.fetchall()

#print (records)              #respuesta base de datos

ID= records[0][1] umbral = records[0][2]

#RUTINA DE EVALUAR ESTADO DE LA LAMPARA



#verificar conneccion serial

diferencia = (datetime.datetime.now()-serial_last).seconds condicional =
estado[ID-1]

print(estado)

if (diferencia > 30)or(condicional == -1) :

  print("error comunicacion serial o estado no valido")              #
  es capaz de detectar un error no arre

  EstadoLuz = 3 else :

  #determinar estado de la lampara Prendida o apagada
    if(condicional == 0) : print("Apagada")

    EstadoLuz = 0



    if(condicional == 1) : print("Prendida")

    if ((valorI2C) >= umbral) :                  #no sensa falsos
    positivos

        EstadoLuz = 1 else:
```

```
        EstadoLuz = 2                                   #Falla Luz Rota


  sql = "UPDATE `hboost_Base_de_Datos`.`Panel` SET `ID` = " + str(ID) + ",
  `Estado` = " + str(Esta cursor.execute(sql)
     #envia mensaje sql

  db.close()                    # disconnect from server




#VARIABLES GLOBALES

estado = [-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]          #INICIALIZACION ESTADO
BOTONERA

serial_last = datetime.datetime.now()

#SERIAL ADMINISTACION

ard = serial.Serial('COM12',9600,timeout=5) time.sleep(1)
threading.Timer(5,Serial_Interrupt).start() time.sleep(5)

#TTN ADMINISTACION

app_id = "appy"

access_key = "ttn-account-v2.cxsQJSM77baRYFMmuDx6ycmpvKS8KKuLlbkh5AwrqQY"
handler = ttn.HandlerClient(app_id, access_key)
mqtt_client = handler.data()
mqtt_client.set_uplink_callback(uplink_callback) mqtt_client.connect()


input()
#bloquea el codigo evita que salga #CIERRE DE CONECCIONES
mqtt_client.close() ard.close()
```