

# **Universidad Católica del Uruguay**

## **Facultad de Ingeniería y Tecnologías**



Universidad  
Católica del  
Uruguay

## **Implementación y estudio de una red LoRaWAN**

**Juan Collazo – Mauro González–**

**Andrés Muracciole**

*Memoria de grado presentada a la Facultad de Ingeniería y  
Tecnologías como requisito parcial para la obtención del grado de  
Ingeniero en Electrónica y Telecomunicaciones*

Tutor: Joel Gak

Montevideo, Setiembre 2018



# Agradecimientos

En esta instancia nos gustaría agradecer a todas aquellas personas que directa o indirectamente formaron parte de este trabajo. Por compartir su experiencia con nosotros y ayudarnos en aquellos momentos donde no obteníamos los resultados esperados.

También agradecer a nuestro tutor Joel Gak por confiar en nosotros y aceptar nuestra solicitud de formar parte de este proyecto y en nombre de él, hacer extensivo el agradecimiento a la Universidad Católica del Uruguay y a su cuerpo docente por acompañarnos a lo largo de este camino.

Por último, no nos podemos olvidar de aquellas personas ajenas a la universidad que también estuvieron a nuestro lado como familiares y amigos. Por su invaluable apoyo y sus constates palabras de aliento.

A todos ellos, ¡muchas gracias!

Juan, Mauro y Andrés

# Índice

Agradecimientos .....	3
Índice .....	4
Índice de Figuras .....	7
Glosario.....	10
<b>Capítulo 1.....</b>	<b>13</b>
1.1 Resumen .....	13
1.2 Problema a abordar.....	14
1.3 Estado del arte .....	14
1.4 Justificación del proyecto .....	15
<b>Capítulo 2.....</b>	<b>17</b>
2.1 Sigfox .....	18
2.2 Narrow Band .....	19
<b>Capítulo 3.....</b>	<b>22</b>
3.1 LoRa Alliance .....	22
3.2 LoRa .....	23
3.3 LoRaWAN .....	25
3.3.1 Seguridad .....	25
3.3.1.1 Network Session Key (NwkSKey).....	26
3.3.1.2 Application Session Key (AppSKey).....	26
3.3.1.3 Métodos de Activación .....	28
3.3.1.3.1 ABP .....	29
3.3.1.3.2 OTAA .....	29
3.3.2 Clases .....	31
3.3.2.1 Clase A.....	31
3.3.2.2 – Clase B .....	34
3.3.2.3 – Clase C .....	35
3.3.3 Frecuencias .....	36
3.3.4 ADR (Adaptive Data Rate).....	38
3.3.5 Duty Cycle .....	38
3.3.6 Topología de Red .....	39
3.4 TTN.....	40
3.4.1 Registro de equipos en TTN .....	43
3.4.1.1 Aplicación de Número Aleatorio.....	43

3.4.1.2 Aplicación de Conteo .....	43
3.4.1.3 Aplicación de Palabra Fija .....	44
3.4.1.4 Aplicación de Humedad y Temperatura .....	44
3.4.1.5 Aplicación de Luz.....	45
<b>Capítulo 4.....</b>	<b>46</b>
<b>4.1 Software.....</b>	<b>46</b>
<b>4.2 Hardware .....</b>	<b>47</b>
4.2.1 Placa de prototipo.....	47
4.2.2 Placa final.....	51
4.3.2.1 Modulo de procesamiento.....	52
4.3.2.2 Módulo de censado .....	54
<b>Capítulo 5.....</b>	<b>57</b>
<b>5.1 Introducción .....</b>	<b>57</b>
<b>5.2 Nuestro Gateway.....</b>	<b>58</b>
<b>5.3 Configuraciones.....</b>	<b>61</b>
<b>Capítulo 6.....</b>	<b>66</b>
<b>6.1 Pruebas de alcance .....</b>	<b>66</b>
6.1.1 CON línea de vista .....	66
6.1.1.1 Prueba con gateway en movimiento .....	66
6.1.1.2 Pruebas con nodo en movimiento .....	67
6.1.2 SIN línea de vista.....	70
<b>6.2 Pruebas limitante de paquetes .....</b>	<b>71</b>
<b>6.3 Prueba de stress .....</b>	<b>72</b>
<b>6.4 Prueba de latencia.....</b>	<b>74</b>
<b>6.5 Pruebas de consumo .....</b>	<b>74</b>
6.5.1 Estudio Teórico .....	74
6.5.2 Estudio Empírico: .....	75
6.5.3 Realización de las Medidas: .....	77
<b>6.6 Estudio de Costos .....</b>	<b>80</b>
6.6.1 Gateway .....	80
6.6.2 Nodos.....	81
<b>Capítulo 7.....</b>	<b>82</b>
<b>Anexo A .....</b>	<b>86</b>
<b>Anexo B.....</b>	<b>89</b>
<b>Anexo C.....</b>	<b>93</b>
ALOHA Puro: .....	93
ALOHA Ranurado: .....	94
<b>Anexo D .....</b>	<b>95</b>
<b>Anexo E .....</b>	<b>98</b>
<b>Anexo F .....</b>	<b>99</b>
<b>F.1 Secciones del código .....</b>	<b>99</b>
F.1.1 Sección Debug.....	99
F.1.2 Sección claves e iniciación de variables .....	100
F.1.3 Sección Intervalo de TX .....	100
F.1.4 Sección Eventos.....	100
F.1.5 Sección Función Enviar.....	101
F.1.6 Sección Setup .....	101

F.1.7 Sección Loop .....	101
F.1.8 Sección Funciones .....	102
<b>F.2 Librerías.....</b>	<b>102</b>
F.2.1 Librerías de Sensores .....	102
F.2.2 Librerías para comunicación serial.....	102
F.2.3 Librería LoRaWAN .....	103
F.2.3.1 RegPaConfig .....	103
F.2.3.2 RegLna.....	104
<b>Referencias .....</b>	<b>105</b>

# Índice de Figuras

Fig. 2.1 – Cobertura global de red Sigfox [x].....	18
Fig. 2.2 – Diagrama de red Sigfox [x].....	19
Fig. 2.3 – Ejemplo de sub-bandas NB-IoT dentro de la frecuencia celular [xi] .....	19
Fig. 2.4 – Despliegue de redes NB en los diferentes países [xii] .....	20
Tabla 2.1 - Cuadro comparativo de NB-IoT, Sigfox y LoRaWAN .....	21
Tabla 3.1 - Tabla que relaciona SF con velocidad de transmisión, distancia y el Time on Air [...] .....	23
Tabla 3.2 – Tabla que relaciona el ancho de banda y el SF con el Data Rate [xiv] ....	24
Fig. 3.1 – Relación entre distancia, SF, ancho de banda, velocidad y sensibilidad [xv] .....	24
Fig. 3.2 - Conformación de las claves de red y aplicación [xvi].....	27
Fig. 3.3 – Encriptación con NwkSKey y AppSKey [xvi].....	27
Fig. 3.4 – Diagrama de encriptación competencia del Payload [xvi] .....	28
Fig. 3.5 – Secuencia de pasos con activación ABP [xvi] .....	29
Fig. 3.6 - Mensaje de join-request del nodo hacia el servidor [xvii] .....	29
Fig. 3.7 – Mensaje de join-accept [xvii] .....	30
Fig. 3.8 – Secuencia de pasos con activación OTAA [xvi] .....	30
Fig. 3.9 – Tipos se mensajes rejoin-request [xvii] .....	31
Fig. 3.10 – Relación Batería en función de la Latencia para cada clase [v] .....	32
Fig. 3.11 – Funcionamiento de LoRaWAN clase A [v].....	32
Fig. 3.12 – Estructura de paquete de uplink LoRaWAN [xvii] .....	33
Fig. 3.13 – Posibles estructuras del campo PHYPayload [xvii].....	33
Fig. 3.14 – Ensanchado de ventanas de recepción ante pérdida de beacons [v] .....	35
Fig. 3.15 – Funcionamiento de nodos clase C [v] .....	36
Tabla. 3.3 – Tabla comparativa entre EU867 – 869 y US 902 – 928 [xvii] .....	37
Fig. 3.16 – Bandas de frecuencia de US 902-928 [xvii] .....	37
Fig. 3.17 – Topología LoRaWAN [xv] .....	40
Fig. 3.18 – Distribución de nodos LoRa[xxii].....	41
Fig. 3.19 – Diagrama de TTN [xxii].....	42
Fig. 3.20– Función de decodificación de aplicación Número Aleatorio .....	43
Fig. 3.21 – Función de decodificación de aplicación Conteo .....	44
Fig. 3.22 – Función de decodificación de aplicación Palabra Fija .....	44
Fig. 3.23 – Función de decodificación de aplicación Humedad y Temperatura.....	45

Fig. 3.24 – Función de decodificación de aplicación Luz .....	45
Tabla 4.1 – Características del micro controlador .....	48
Fig. 4.1 – Micro controlador Arduino Pro Mini 3,3V – 8MHz .....	48
Fig. 4.2 – Módulo de comunicación LoRa RFM95W .....	49
Fig. 4.3 – Antena Helicoidal 915Mhz .....	49
Fig. 4.4 – Sensor de humedad y temperatura DHT-11 .....	49
Fig. 4.5 – Sensor de luz BH1750 .....	50
Fig. 4.6 – Sensor de presión BMP280 .....	50
Fig. 4.7 – Esquemático de la placa de prototipo realizado en Eagle .....	50
Fig. 4.8 – Disposición de los componentes de la placa de prototipo realizado en Eagle .....	51
Fig. 4.9 – Placa de prototipo compuesta por un microcontrolador, un sensor de presión y el módulo de comunicación LoRa .....	51
Tabla 4.2 – Cuadro comparativo de diferentes posibles microcontroladores .....	52
Fig. 4.10– Esquemático del módulo de procesamiento de la placa final realizado en Eagle .....	53
Fig. 4.11 – Disposición de los componentes del módulo de procesamiento de placa final realizado en Eagle.....	53
Fig. 4.12 – Módulo de procesamiento.....	54
Fig. 4.13 – Esquemático del módulo de censado de la placa final realizado en Eagle .....	54
Fig. 4.14 – Disposición de los componentes del módulo de censado de placa final realizado en Eagle .....	55
Fig. 4.15 – Módulo de censado .....	55
Fig. 4.16 – Endpoint final .....	56
Fig. 5.1 - Developer KIT.....	59
Tabla 5.1 – Parámetros del módulo RAK831 .....	60
Fig. 5.2 - Gateway compuesto por módulo RAK831 + Host [] .....	61
Tabla 5.2 - Conexión entre RAK831 y Raspberry [xxxvi] .....	62
Fig. 5.3 - Conexión del Gateway, Raspberry PI a la izquierda y RAK831 a la derecha.....	62
Fig. 5.4 - Configuración del gateway en servidor TTN.....	64
Fig. 5.5 - Conexión del gateway, Raspberry PI y RAK831 .....	64
Figs. 6.1 y 6.2 – Instalación de antena para pruebas con gateway en movimiento ..	67
Figs. 6.3 y 6.4 – Antenas omnidireccionales Cisco 3G-ANTM-OUT-OMy Cisco 4G-LTE-ANTM-D respectivamente .....	67
Fig. 6.5 – Relevamiento realizado con TTNMapper .....	68
Fig. 6.6 – Nivel de señal en función de la posición del nodo .....	68
Fig. 6.7 – Criterio de colores de TTNMapper para nivel de recepción .....	69
Fig. 6.8 – Posición de nodo y gateway registrando comunicación entre sí.....	69
Tabla. 6.1 – Información de paquete recibido a 8km. del gateway.....	70
Fig. 6.9 – Antena omnidireccional marca RAK de 6dBi .....	70
Fig. 6.10 – Distancia máxima alcanzada sin línea de vista .....	71
Fig. 6.11 – Paquete 5636 del nodo “Número Aleatorio” en el servidor de red .....	72
Fig. 6.12 – Paquete 5636 del nodo “Número Aleatorio” en el servidor de aplicaciones .....	72
Fig. 6.13– Nodos programados y funcionando para la prueba de stress .....	73
Tabla 6.2 – Cuadro asociado a la disponibilidad de la red .....	73
Tabla 6.3 - Consumo de los componentes en diferentes modos de funcionamiento	75
Fig. 6.14 – Circuito auxiliar .....	76
Tabla 6.4 – Resistencias utilizadas en los diferentes modos de funcionamiento.....	76
Fig. 6.15 – Circuito implementado con la placa Electronics Explorer para medir el consumo del nodo.....	77

<b>Fig. 6.16 – Medidas de consumo del hardware con leds y LDO.....</b>	<b>77</b>
<b>Fig. 6.17 – Medidas de consumo del hardware con LDO y sin leds .....</b>	<b>78</b>
<b>Fig. 6.18 – Medidas de consumo del hardware sin LDO ni leds.....</b>	<b>79</b>
<b>Tabla 6.5 – Cuadro comparativo con los datos teóricos y prácticos de consumo .....</b>	<b>79</b>
<b>Fig. 6.19 – Gráfico de duración en años en función de la frecuencia de envíos .....</b>	<b>80</b>
<b>Tabla 6.6 – Detalle de gastos del gateway .....</b>	<b>81</b>
<b>Tabla 6.7 – Detalle de gastos de un endpoint .....</b>	<b>81</b>
<b>Fig. A.1 – DSSS del lado del emisor [xxiv].....</b>	<b>86</b>
<b>Fig. A.2 – DSSS del lado del receptor [xxiv] .....</b>	<b>87</b>
<b>Tabla. A.1 – Relación entre SF y ganancia [xxiv].....</b>	<b>87</b>
<b>Fig. B.1 - Creación de la matriz de estado final o cifrado a partir del mensaje original y las sub-claves.....</b>	<b>89</b>
<b>Fig. B.2 - Ejemplo de clave principal.....</b>	<b>90</b>
<b>Fig. B.3 - Función RotWord .....</b>	<b>90</b>
<b>Fig. B.4 - Tabla de sustitución .....</b>	<b>90</b>
.....	<b>90</b>
<b>Fig. B.5 - En la figura de la izquierda de hace el “SubBytes” y luego a la derecha se aplica el XOR .....</b>	<b>90</b>
<b>Fig. B.6 - Ultimo paso .....</b>	<b>91</b>
<b>Fig. B.7 - Creación de la matriz de estado .....</b>	<b>91</b>
<b>Fig. B.8 - Operación AddRoundKey .....</b>	<b>91</b>
<b>Fig. B.9 - Función SwifRows .....</b>	<b>91</b>
<b>Fig. B.10 - Función MixColumns .....</b>	<b>92</b>
<b>Fig. F.1 – Instancia donde se define el tipo de sincronización a utilizar. Para este caso está habilitada ABP .....</b>	<b>100</b>
<b>Fig. F.2 – Diagrama de bloques del módulo SX1276 [xxiv].....</b>	<b>103</b>
<b>Fig. F.3 – Registro RegPaConfig [xxvi] .....</b>	<b>104</b>
<b>Fig. F.4 – Registro RegLna [xxvi].....</b>	<b>104</b>

# Glosario

AppEUI: Identificador estático y único de la aplicación de 64 bits. Se expresa en 8 caracteres hexadecimales.

AppKey: Llave de encriptación compuesta por la NwkSKey y AppSKey de 128 bits de largo expresada en 16 caracteres hexadecimales.

AppSKey: Clave para encriptación y verificación de integridad a nivel de aplicación. Encripta únicamente el payload del paquete. Tiene un largo de 128 bits.

DevAddr: Identificador dinámico lógico del dispositivo en la red formado por 32 bits.

DevEUI: Identificador estático y único del dispositivo físico de 64 bits. Se expresa en 8 caracteres hexadecimales.

Downlink: Paquete entrante o recibido.

Endpoint o nodo: Dispositivo remoto que suele estar conectado a un sensor y envía y recibe información hacia y desde el *gateway* respectivamente usando modulación LoRa. Poseen una dirección lógica que los diferencia de los demás (DevAddr) y así forman parte de la red LoRaWAN.

Gateway: También llamados estación base o concentrador. En el dispositivo que tiene la capacidad de procesar paquetes modulados con LoRa y enviarlos a internet y viceversa. Es el nexo entre los nodos y los servidores de red y aplicaciones.

Gateway ID: Identificador único del *gateway*.

LoRa: Long Range. Tipo de modulación RF implementada por Semtech de baja potencia y ancho de banda para uso de IoT.

LPWAN: Es la abreviación de Low Power WAN. Es una familia de tecnologías enfocadas en el bajo uso de potencia para la transmisión de información.

NwkSKey: Clave para encriptación y verificación de integridad de los datos a nivel de red. Encripta el paquete completo incluido payload y encabezados con dirección de origen y destino. El largo de la misma es de 128 bits expresados en hexadecimal.

Roaming: Capacidad que tiene la red en determinar cuál *gateway* es el más propicio para comunicarse con un nodo cuando este se encuentra en movimiento. Esta decisión es tomada por el servidor de red a partir de los niveles de señal recibidos por las estaciones más cercanas.

Servidor de Aplicaciones: Es aquel capaz de poder descifrar el payload de un mensaje e interpretar el valor en bruto que se envió desde el nodo.

Servidor de Red: Aquel capaz de poder descifrar parte del payload y mediante el *AppEUI* determinar a qué servidor de aplicaciones enviar el paquete. No es capaz de leer el dato en sí ya que desconoce de la llave de aplicación.

SNR: Signal-to-noise ratio o relación señal/ruido es la porción entre la potencia de transmisión de la señal y la potencia de ruido interferente. Se mide en dB.

Spreading Factor (SF): Número de chips en el que se divide un símbolo que se quiere transmitir.

Time on Air: Tiempo que tarda en llegar a destino un paquete de uplink. Este depende de varios factores como el SF, el ancho de banda y el largo del mismo entre otros.

Uplink: Paquete saliente o enviado.



# Capítulo 1

## Introducción

### 1.1 Resumen

Este proyecto consta del estudio e implementación una red de largo alcance utilizando la tecnología LoRa<sup>1</sup>. A este tipo de red se la conoce como LoRaWAN (Long Range Wide Area Network), la cual debido a sus características es ideal para la conexión de equipos y sensores. Hoy en día esta clase de redes están fuertemente vinculadas a lo que se conoce como redes para aplicaciones de IoT<sup>2</sup> (Internet of Things).

En primera instancia se estudiará en profundidad el funcionamiento de la tecnología y se la comparará con otros protocolos de red utilizados en IoT. Luego, mediante procedimientos teóricos y los conocimientos adquiridos se decidirá cuáles son los componentes adecuados para montar una red de este tipo. De esta forma se explicarán los principales componentes que forman una red LoRaWAN y detalles de cada uno. En tercer lugar, se llevarán a cabo una serie de pruebas con el fin de poder ver el comportamiento de esta tecnología transmitiendo a línea de vista o dentro de ciudad, así como también pruebas de consumo eléctrico para comparar los datos empíricos con los teóricos. Por último, luego de realizadas las pruebas se hará una proyección de las capacidades máximas que podría llegar a alcanzar una red con un único dispositivo central.

---

<sup>1</sup> Long Range: Modulación para transmisión de paquetes a largas distancias y baja potencia

<sup>2</sup> IoT: Equipos interconectados que intercambian información en procura de conectar las cosas entre si y hacia internet

## 1.2 Problema a abordar

El problema principal a tratar será poder interconectar varios dispositivos finales a un mismo *gateway* de forma de poder enviar y recibir información de manera eficaz, confiable y barata.

El alcance del proyecto en primer lugar será poder conocer y entender en profundidad el funcionamiento de este protocolo. Esto será necesario ya que, al ser una tecnología relativamente nueva, ningún integrante del grupo cuenta con conocimientos previos de la misma. A pesar de ello, se eligió LoRaWAN debido a su capacidad de transmitir información a largas distancias y a costos reducidos de despliegue como se verá más adelante. A su vez, gracias a esta investigación, aparecerá otro problema a abordar que será el estudio de los diferentes hardware que existen en el mercado, de forma de poder decidir el que mejor se adapte a nuestras necesidades. A partir de experiencias previas con redes TCP-IP inalámbricas, como WiFi, se conoce que, al implementar redes con muchos dispositivos finales que se encuentren intercambiando información entre sí, éstas se vuelven cada vez más complejas. Por ello es que se cree que esto también será un problema importante a tratar; principalmente cuando se realice una proyección a gran escala.

En cuanto al alcance, se espera poder lograr comunicar varios dispositivos finales a una estación base. Estas comunicaciones deberán tener la particularidad de ser capaces de enviar y recibir paquetes hacia uno o múltiples equipos de ser necesario, de forma de poder atacar y mitigar los problemas de distorsión e interferencia.

Palabras claves:

LoRaWAN; LoRa; IoT; TTN; 915MHz

## 1.3 Estado del arte

LoRa es una tecnología de comunicación inalámbrica reciente, creada por la empresa Semtech, que se comenzó a utilizar en el transcurso del año 2012. Dado a su gran potencial, varias empresas como Cisco, IBM, HP, Microchip y Multitech se interesaron en el tema y comenzaron a fabricar y diseñar diferentes dispositivos capaces de medir, procesar, recibir y transmitir información utilizando esta modulación [i].

A pesar de su corta vida, hoy en día existen varios proyectos a nivel mundial que se apoyan sobre este protocolo y el uso de este crece de forma exponencial [ii]. No es raro escuchar que a futuro todo estará conectado entre sí y que, si bien existen varias tecnologías aplicables a ello, una de las que predominará en el mercado será LoRa sin lugar a duda ya que fue pensada principalmente para aplicaciones de IoT; a diferencia de otras que son adaptaciones.

En la actualidad existen soluciones similares a las que se plantean en este proyecto; sobre todo en Europa, parte de Asia y Norteamérica, las cuales buscan

interconectar de forma inalámbrica, diferentes equipos que se encuentren en lugares remotos o a grandes distancias entre sí. A forma de ejemplo, uno de estos fue el llevado a cabo en Corea por las empresas Samsung y SK Telecom, las cuales desplegaron una red LoRaWAN en la ciudad de Daegu con la finalidad de introducir al país en las IoT. A parte de poder interconectar diferentes dispositivos personales, esta red está siendo implementada para ser capaz de obtener información del tráfico y del clima utilizando sensores en las calles de la ciudad. La información es enviada a nodos centrales que son capaces de interpretar y analizar los datos en servidores especializados [iii].

A menor escala que el caso anterior, existen también otros trabajos llevados adelante en diferentes universidades europeas como es el caso de la Universidad de Huelva. Aquí se instaló un *Gateway* en el Campus de El Carmen, de forma de permitir aplicaciones IoT dentro del mismo y así medir y estudiar el alcance de la tecnología para futuros proyectos [iv]. El principal impulsor de esto es LoRa Alliance [v], a quienes se le atribuye el haber diseñado el protocolo LoRaWAN. Esta es una alianza sin fines de lucro conformada por diversas empresas y personas distribuidas por el mundo con una misma filosofía; “*el internet de las cosas es ahora*”. El objetivo es poder intercambiar experiencias y conocimientos para así mejorar el protocolo, garantizando interoperabilidad y flexibilidad técnica.

Si se mira hacia Uruguay y América del Sur se puede notar que estamos atrasados en relación con grandes potencias como Estados Unidos, Europa, Asia o Australia, en las cuales ya hay redes desplegadas. Sin embargo, se ve que se está en crecimiento ya que, según Lora Alliance en julio del presente año, ya se cuenta con redes LoRaWAN implementadas en nuestro país, pero privadas y no muy extendidas<sup>3</sup>. Por otro lado, en este 2018 desembarcó en Uruguay una empresa argentina llamada YEAP! [vi], la cual tiene como objetivo desplegar una red LoRaWAN por todo el país. En una primera instancia pretenden tener instalados unos 20 concentradores en Montevideo para luego llegar al número de 1200 a lo largo y ancho del país para finales de 2019.

## 1.4 Justificación del proyecto

Si bien ya existen soluciones similares como por ejemplo la red que posee TeLlot, muchas de ellas se encuentran aún en etapa de prueba y desarrollo; de éstas últimas implementadas principalmente por desarrolladores particulares para redes propias. Sumado a ello, la mayoría de estos trabajos se están realizando en otros países, por lo que el principal diferenciador se enfoca en no solo continuar con las pruebas de funcionamiento, sino también poder adaptarla a las necesidades del mercado local.

En la universidad existen otros trabajos [vii] [viii] de tesis de grado y proyecto de maestría en los que han estudiado la modulación LoRa, pero sin llegar a implementar una red LoRaWAN. En estos trabajos las comunicaciones que se alcanzaron a realizar

---

<sup>3</sup> La empresa TeLlot tiene implementada una red que cubre gran parte de la costa de Montevideo ([www.teliot.io](http://www.teliot.io))

fueron punto a punto entre dos nodos. No se tuvo la necesidad de utilizar un servidor central de red ni un gateway como concentrador y conector de las redes.

Otro aspecto que resaltar de este trabajo es que las principales redes LoRaWAN que existen hoy en día están siendo montadas por grandes empresas o universidades como se mencionó anteriormente. Sin embargo, aquí se propone mostrar que es posible implementar una red propia a costos razonables sin la necesidad de tener que invertir en contratar los servicios de grandes multinacionales. De todas formas, el principal punto por el cual es interesante la implementación de una red así es que, desde hace ya unos años, las empresas tecnológicas están enfocadas en el objetivo de poder comunicar todas las cosas entre sí (IoT) y se habla que uno de los principales protocolos para llevar esto a cabo esto será el utilizado en este trabajo.

# Capítulo 2

## Otras tecnologías

Para comprender la dimensión de una solución que se puede obtener utilizando LoRaWAN, es necesario compararla con otras tecnologías similares para conocer sus prestaciones y limitantes. Por dicho motivo es que se tomaron como ejemplo dos de las más utilizadas, como son Sigfox y Narrow Band (NB-IoT) ya que ambas pertenecen a la familia de las LPWAN. Una vez tratados estos temas, se procederá a mencionar en detalle todo lo que respecta a LoRaWAN.

Internet de las cosas (también llamado IdC o IoT por sus siglas en inglés) refiere a la interconexión de los objetos cotidianos a internet como por ejemplo autos, sensores, electrodomésticos, etc. Si bien muchos de estos suelen estar en lugares donde hay energía eléctrica, otros no; por lo cual un tema no menor en IoT es el consumo eléctrico de los componentes. Sumado a ello, la transmisión de información debe de ser de forma inalámbrica ya que es impensable realizar cableado para cientos, miles o millones de componentes, ya sea por tema de costos como también de escalabilidad.

El avance de las tecnologías en este sentido ayuda a poder obtener información en tiempo real de diversos objetos y a partir de ello generar una base de datos que permita predecir un comportamiento o prevenir diferentes problemas. Los mercados donde se puede introducir IoT son muy variados y van desde ciudades inteligentes donde ayuden a generar ahorros energéticos; hasta aplicaciones hogareñas o industriales que permitan predecir la falla de una máquina a partir de sensores que se encuentran monitoreando los niveles de carga y consumo.

El crecimiento de los objetos conectados a internet en los últimos 10 años ha crecido de forma exponencial al punto que se estima que en el 2020 habrá unos 50 mil millones de dispositivos interconectados entre sí [ix]. Esta expansión debe ir acompañada de nuevas tecnologías capaces de facilitar esta conexión lo más dinámica y

eficaz posible. Es por ello que comienzan a surgir protocolos propios para IoT como los que se mencionan en las secciones 2.1, 2.2 y 3.

## 2.1 Sigfox

Sigfox [x] es una compañía francesa creada en 2009 que surge con el propósito de buscar la forma de conectar dispositivos de forma inalámbrica, sencilla, a bajo costo energético y de largo alcance. Por esto último es que forma parte de las llamadas LPWAN. Utiliza frecuencias sub giga-Hertz, en las bandas ISM (Industrial, Scientific and Medical) y actualmente se encuentra desplegada en más de 40 países, cubriendo un área de 4 millones de kilómetros cuadrados y dándole conectividad a más de 900 millones de usuarios [x], en la Figura 2.1 se presenta un mapa de cobertura.

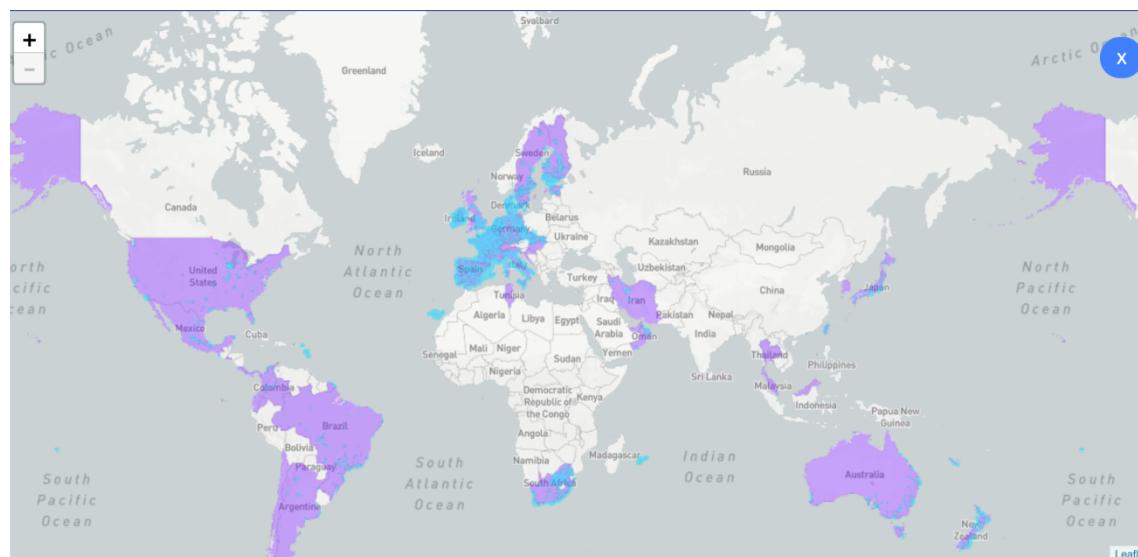


Fig. 2.1 – Cobertura global de red Sigfox [x].

Esta red tiene la particularidad de que es montada y desplegada por la empresa misma, por lo cual el usuario solo se encarga de adquirir los nodos finales. Si bien en cierto punto es una ventaja no tener que implementar y mantener la red uno mismo, se está supeditado a que el proveedor ofrezca cobertura en su país, así como también, en caso de querer utilizar dicha tecnología, a pesar de funcionar en frecuencias ISM, será necesario contratar los servicios de Sigfox.

La topología de la misma se asemeja a la de una red celular donde hay una serie de antenas distribuidas en la ciudad e interconectadas entre sí y a internet. Estas se encargan de recibir y transmitir la información a los nodos finales (propiedad del usuario) y también a un servidor Sigfox en la nube como se puede apreciar en la figura 2.2. A este último es donde el usuario, previamente registrado, puede acceder para recabar los datos obtenidos [x].

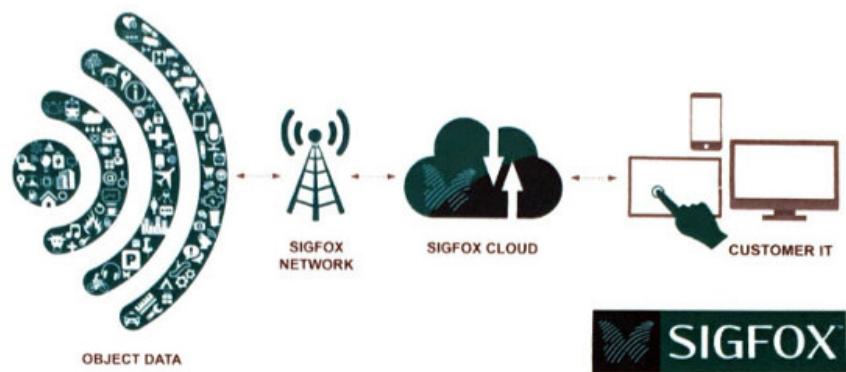


Fig. 2.2 – Diagrama de red Sigfox [x]

La distancia de transmisión, se estima que en áreas rurales se pueden alcanzar entre 30 y 50 kilómetros, mientras que en sitios más densos ésta baja a entre 3 y 10 kilómetros aproximadamente.

Por último, es importante mencionar la limitante que presenta dicha tecnología. Si bien se suele utilizar en aplicaciones donde no se requiere enviar una gran cantidad de paquetes, Sigfox tiene como política que cada usuario puede enviar hasta un máximo de 140 mensajes por día por nodo de unos 12 bytes de largo.

## 2.2 Narrow Band

Narrow Band o NB-IoT [xi] es otro de los protocolos utilizados para comunicar dispositivos entre sí o a internet. Este consiste en tomar parte del espectro utilizado para telefonía celular como se ve en la figura 2.3 y reservarlo para aplicaciones de IoT. La selección de estas frecuencias no es aleatoria, sino que se toman porciones pequeñas del espectro conocidas debido a su buena penetración y bajo ruido lo cual permite lograr una mayor sensibilidad y mayor huella de cobertura.

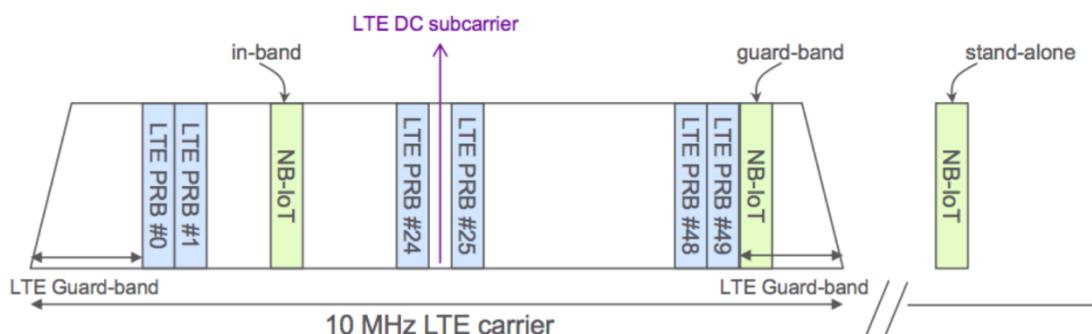


Fig. 2.3 – Ejemplo de sub-bandas NB-IoT dentro de la frecuencia celular [xi]

Una ventaja que presenta esta tecnología respecto a Sigfox es que utiliza la red celular la cual ya se encuentra desplegada. Al igual que sucede con Sigfox, el usuario solo debe preocuparse por adquirir del nodo ya que la red pertenece al proveedor de

servicios. Con Narrow Band es posible alcanzar velocidades de transmisión de 9600 bps con un canal de 12,5 KHz de ancho<sup>4</sup>, pero con algunas técnicas modernas de modulación de señales se registraron velocidades de hasta 19200 bps. Esto es gracias a un procedimiento que consiste en enviar solo aquella parte de la señal que sufre variaciones [xi].

NB-IoT está destinada a cubrir esa porción del mercado donde se precise enviar mayor cantidad de información para la cual ni Sigfox ni LoRaWAN están preparadas. Si bien en algunos puntos son competencia, a su vez cada una de ellas tiene una porción del espectro donde se destaca, por lo cual pueden coexistir simultáneamente sin inconvenientes. El mayor diferencial de esta tecnología respecto a las demás es su baja latencia. Esto la hace propicia para aquellas aplicaciones donde se requiera que, ante algún incidente, el dispositivo pueda realizar una acción inmediata [xii].

Si bien las empresas telefónicas se encuentran realizando algunas implementaciones en este rubro; hasta el momento no hay aplicaciones concretas que utilicen NB-IoT en nuestro país, ya que como se puede ver en la imagen 2.4, todavía no hay una red de ese tipo en nuestro país. Con la liberación y venta de la banda 700 MHz de LTE se cree que comenzarán a aparecer las primeras aplicaciones. Sin embargo, habrá que esperar a que se lancen al mercado los primeros dispositivos; ya que a diferencia de Sigfox o LoRaWAN, la frecuencia utilizada no pertenece a la parte del espectro libre, por lo cual no se puede traficar sin la previa autorización del ente regulador.

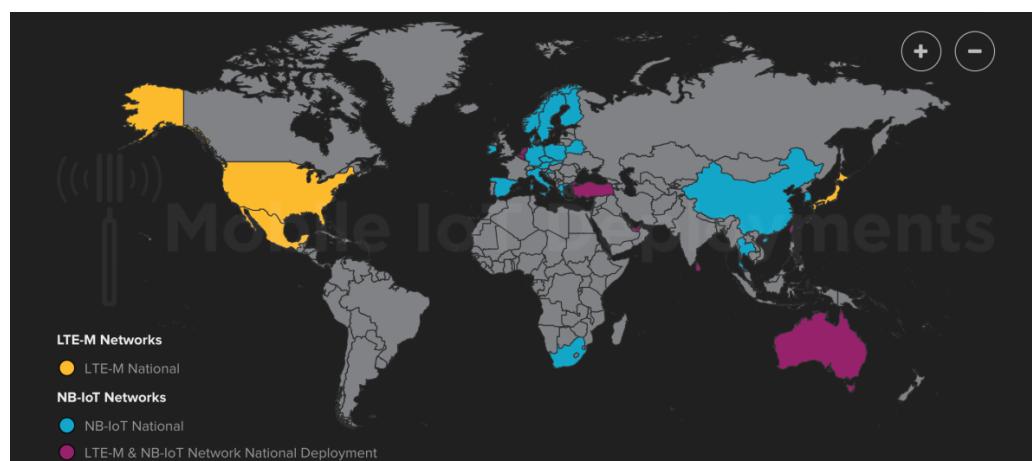


Fig. 2.4 – Despliegue de redes NB en los diferentes países [xii]

En la tabla 2.1 se presenta un resumen comparativo, dando pie a la tecnología central de este trabajo. En la tabla se especifican algunas características centrales de estos dos protocolos de comunicación aplicados a IoT y se los compara con LoRaWAN.

<sup>4</sup> El ancho de banda reservado para NB es de 180 KHz

Tecnología	NB-IoT	Sigfox	LoRaWAN
<b>Frecuencia (MHz)</b>	Guarda de bandas del espectro LTE	Bandas sub giga-hertz	Bandas sub giga-hertz
<b>Ancho de banda DownLink</b>	180 KHz (sub-bandas de 15 KHz)	125 KHz, 500 KHz	Configurable
<b>Ancho de banda UpLink</b>	Single tone o multitone (180 KHz)	125 KHz, 250 KHz, 500 KHz	200 KHz (Canales de 100 Hz en UL y de 600Hz en DL)
<b>Acceso múltiple DL</b>	OFDMA	Chirp Spread Spectrum (CSS)	UNB/FHSS
<b>Acceso múltiple UL</b>	Single tone (FDMA) o multitone (SC-FDMA)	Chirp Spread Spectrum (CSS)	UNB/FHSS
<b>Modulación DL</b>	BPSK, QPSK, opcionalmente 16QAM	LoRa, (G)FSK	GFSK
<b>Modulación UL</b>	BPSK, QPSK, 8PSK, opcionalmente 16QAM	LoRa, (G)FSK	DBPSK
<b>Pico de data rate</b>	DL: 250 kbps  UP single tone: 20 kbps  UP multitonned: 250 kbps	0,3 kbps – 50 kbps	100 bps en subida y 600 bps en bajada
<b>Sensibilidad</b>	-164 dB	-150 - 157 dB	-146 -162 dB

Tabla 2.1 - Cuadro comparativo de NB-IoT, Sigfox y LoRaWAN

# **Capítulo 3**

## **LoRaWAN**

### **3.1 LoRa Alliance**

En marzo de 2015, un grupo de empresas de electrónica y telecomunicaciones crean LoRa Alliance [v] con el fin de poder incentivar a los investigadores a realizar desarrollos utilizando modulación LoRa. Esto motivó a que personas interesadas en el rubro de las IoT comenzaran a familiarizarse con esta tecnología que hasta el momento era muy nueva.

A parte de estimular a los desarrolladores, LoRa Alliance tiene como propósito generar conciencia de la importancia que tendrá en un futuro las aplicaciones de IoT. Por ello invierten dinero en busca de poder lanzar dispositivos que sean cada vez más potentes y económicos para que cualquier interesado pueda acceder a ellos. A su vez, debido a la gran variedad de tecnologías que existen para aplicaciones IoT, las empresas que lo conforman buscan poder afianzar LoRaWAN como de las principales en el rubro del Internet of Things. En la actualidad, la alianza está formada por más de 140 empresas entre las que se destacan Semtech (<https://www.semtech.com/>), Cisco (<https://www.cisco.com/>), IBM (<https://www.ibm.com/uy-es/>) y Microchip (<https://www.microchip.com/>). Todas ellas están unidas bajo el pensamiento de que *“la era del internet de las cosas es ahora”* y al formar parte de ella, éstas se comprometen a compartir sus conocimientos y garantizar la interoperabilidad para que el acceso sea universal.

Uno de los mayores logros que ha tenido es haber podido unificar LoRa con las redes LPWAN. Si bien se sabe que las empresas telefónicas (mediante el uso de NB-IoT) tendrán un rol importante en la conexión de las cosas a internet, se estima que estas podrán cubrir entre un 10% y un 15% del mercado [v]. Para lo demás será necesario otro

tipo de solución ya que hay casos donde la cobertura celular no llega; así como también los costos que implica utilizar Narrow Band para ciertas aplicaciones no lo ameritan. Es para estos casos en los cuales LoRa Alliance trabaja.

## 3.2 LoRa

LoRa es un tipo de modulación de señales de radiofrecuencia que fue desarrollada por Semtech para aplicaciones IoT y que por la forma que tiene de procesar las señales, permite alcanzar grandes distancias con baja potencia. Debido a esto último, se dice que LoRa pertenece a la familia de las LPWAN.

La base de LoRa está fundada en modulación chirp, la cual consiste en deformar los pulsos en señales más anchas de período fijo (ver anexo A). De esta forma no es necesario que los filtros a utilizar sean tan angostos y precisos y, por consiguiente, la demodulación es más barata y fácil ya que la sensibilidad aumenta, lo cual lleva por consecuencia mayor alcance. Este tipo de modulación está fuertemente vinculada al concepto de *spreading factor (SF)*, el cual es un valor que refiere a cuán grande es el ensanchado de las señales [xiii]. En la tabla 3.1 se puede apreciar cómo cambia el valor de otros parámetros al variar el SF. A mayor SF, mayor será el alcance, pero menor la cantidad de información que se podrá enviar debido a la interferencia entre símbolos.

Spreading Factor	Bitrate	Range	Time on Air (ms)
SF7	5470 bps	2 km	56 ms
SF8	3125 bps	4 km	100 ms
SF9	1760 bps	6 km	200 ms
SF10	980 bps	8 km	370 ms
SF11	440 bps	11 km	740 ms
SF12	290 bps	14 km	1400 ms

Tabla 3.1 - Tabla que relaciona SF con velocidad de transmisión, distancia y el Time on Air [xiv]

El bitrate (velocidad de transmisión) se puede calcular como indica la ecuación 3.1.

$$Br = SF \cdot \frac{BW}{2^{SF}} \cdot CR \quad [bits/s] \quad (3.1)$$

El coding rate (CR) es un dato utilizado para la redundancia. Se expresa en k/n donde por cada "k" bits de información, se generan "n" bits de datos de forma que n-k son redundantes. Este puede tomar los siguientes valores: "4/5", "4/6", "4/7" o "4/8".

Data Rate	Configuration	Bits/s	Max Payload
<b>DR0</b>	SF10/125kHz	980	19
<b>DR1</b>	SF9/125kHz	1760	61
<b>DR2</b>	SF8/125kHz	3125	133
<b>DR3</b>	SF7/125kHz	5470	250
<b>DR4</b>	SF8/500kHz	12500	250
<b>DR8</b>	SF12/500kHz	980	41
<b>DR9</b>	SF11/500kHz	1760	117
<b>DR10</b>	SF10/500kHz	3900	230
<b>DR11</b>	SF9/500kHz	7000	230
<b>DR12</b>	SF8/500kHz	12500	230
<b>DR13</b>	SF7/500kHz	21900	230

Tabla 3.2 – Tabla que relaciona el ancho de banda y el SF con el Data Rate [xiv]

LoRaWAN utiliza SF entre 7 y 12 aunque empíricamente se ha demostrado que no es muy eficiente un spreading factor mayor a 10 ya que el consumo de batería es bastante mayor y su rendimiento no es considerablemente mejor a un SF de 10[xv]. Las relaciones entre todos estos conceptos anteriormente planteados se pueden apreciar en la tabla 3.2 y figura 3.1. La configuración del mismo puede hacerse tanto de forma manual como automática mediante el uso de ADR<sup>5</sup>. Se recomienda de ser posible utilizar esta opción debido a que es más escalable que configurándolo estáticamente y genera un mejor rendimiento de la batería ya que va variando este valor en función de la distancia a la cual se encuentre el nodo del *gateway*. Se profundizará más sobre este tema en la sección 3.3.4.

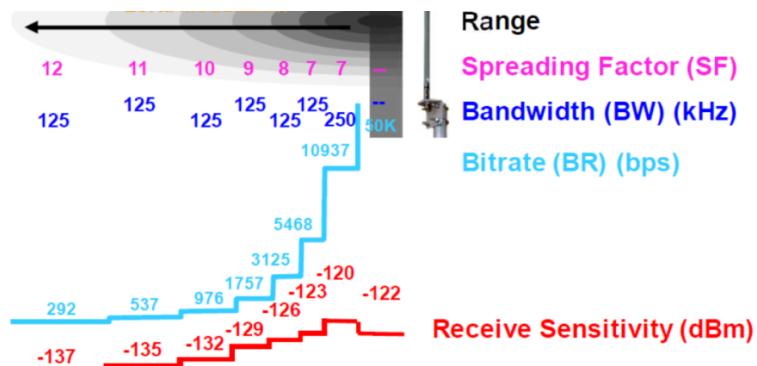


Fig. 3.1 – Relación entre distancia, SF, ancho de banda, velocidad y sensibilidad [xv]

<sup>5</sup> ADR: Adaptive Data Rate

Los dispositivos que trabajan con LoRa tienen la particularidad de ser equipos utilizados para IoT, de forma que, si se ve en una línea del tiempo las señales que envía y recibe un *endpoint*, se pueden notar comportamientos impulsivos de corta duración y espaciados en el tiempo. La transmisión de datos no presenta una conducta continua como una secuencia de paquetes TCP.

Al ser simplemente un tipo de modulación, solo es necesario un emisor y un receptor que sean capaces de modular y demodular estas señales para poder realizar la comunicación. Sin embargo, esta no será segura ya que otro equipo configurado en esa frecuencia también podrá interpretar el paquete. Sumado a ello, tampoco será una red en sí misma ya que no hay forma de direccionar paquetes para algunos nodos en especial. Si se lo desea asociar a las capas del modelo OSI, se puede decir que LoRa conforma la capa física y de enlace. Sin embargo, hasta el momento no aparece el concepto de “direcciones” para diferenciar los nodos o aplicaciones, por lo cual sólo con LoRa no es posible desplegar una red. Aquí es donde es necesario agregar el protocolo LoRaWAN el cual corre sobre LoRa y se detallará en el punto 3.3.

## 3.3 LoRaWAN

### 3.3.1 Seguridad

Al ser una red inalámbrica, es sumamente importante contar con un buen sistema de encriptación y cifrado de los datos ya que estos al viajar por el aire se vuelven muy vulnerables y fáciles de interceptar. Si bien cabría pensar que al no ser un protocolo que pueda transmitir mucha información las aplicaciones que corren sobre ella no son de gran importancia; de todas formas, es necesario cuidar la integridad y seguridad de las mismas ya que en caso de que un “man in the middle” obtenga los datos, puede generar un gran problema que tenga importantes repercusiones económicas. Para poner un ejemplo, supongamos que mediante este protocolo se envía un comando capaz de abrir o cerrar un electroimán que hace de cerradura de una puerta. Si se puede manipular el dato, será muy fácil para un ladrón enviar una señal corrupta capaz de abrir la puerta y así entrar fácilmente sin necesidad de forzar la cerradura. Para ello LoRaWAN incluye seguridad tanto a nivel de red como de aplicación. El tipo de encriptación que utiliza es AES 128<sup>6</sup> y las diferentes llaves de cifrado que maneja son las que se detallarán en las secciones 3.3.1.1 y 3.3.1.2. Este tipo de encriptación fue adoptado debido a su alta robustez. Surgió en 1997 ya que el anterior método (DES<sup>7</sup>) comenzaba a tener los primeros casos de vulnerabilidad. Está basado en el método de Rijndael, en honor a sus investigadores Vincent Rijmen y John Daemen y hasta el momento no hay registro de que haya sido vulnerado. El algoritmo se basa en claves simétricas que surgen a partir de una inicial y es capaz de procesar bloques de texto plano de 128 bits con claves estándar también de 128 bits.

---

<sup>6</sup> Ver el Anexo B para mayor información de dicho método.

<sup>7</sup> DES: Data Encryption Standard

En el Anexo B se puede apreciar con mayor detalle los pasos llevados a cabo por este algoritmo para cifrar cada paquete, así como también lo que realiza el destinatario para descifrar los mismos. Al ser AES 128, las claves son justamente de 128 bits, por lo cual, en caso de querer vulnerar este método mediante fuerza bruta se requerirá probar una por una en un total de  $3,4 \times 10^{38}$  posibilidades distintas.

### 3.3.1.1 Network Session Key (NwkSKey)

La llave NwkSKey se utiliza para la interacción entre el nodo final y la red. Su funcionalidad es encriptar la información desde el nodo hasta el servidor de red y llevar a cabo una validación del mensaje (MIC<sup>8</sup>) de forma de corroborar la integridad de este (es similar a un checksum).

En consecuencia, el servidor de red es capaz de poder desencriptar la información a partir de esta clave y además de realizar chequeos de integridad, permite leer los encabezados, el origen y la aplicación de destino. Sin embargo, no es capaz de interpretar el payload en sí mismo ya que este viene encriptado también con la AppSKey.

### 3.3.1.2 Application Session Key (AppSKey)

La clave AppSKey relaciona el nodo final con el servidor de aplicaciones y se encarga de encriptar y desencriptar el payload, por lo tanto, los paquetes que viajan por la red solo pueden ser interpretados por el usuario final. Se puede asociar como una encriptación punto a punto a nivel de aplicación.

Como se puede apreciar en la figura 3.3, en primer lugar, se recopila la información a enviar (por ejemplo, obtenida de un sensor) y se cifra con la AppSKey. Luego se le agregan los encabezados, dirección de origen y destino y se vuelve a encriptar, pero ahora con la NwkSKey. Por último, se lo modula en LoRa y se envía al gateway inalámicamente. Una vez que el paquete llega a destino, primero se descifra con la clave de red y se valida la integridad. En segundo lugar, se lee la dirección del nodo que envió la información y el destino de este ya que es a este último que se le envía el payload (encriptado aún por la AppSKey). Por último, el servidor de aplicaciones lee la información y despliega la información. La forma en que es desplegada la información varía según el servidor de red que sea utilizado.

A su vez está la AppKey la cual surge de las dos anteriores como muestra la figura 3.2. Esta es la que se suele utilizar cuando la activación es mediante el método OTAA (ver punto 3.3.1.3.2) y solo es conocida por el dispositivo y la aplicación.

---

<sup>8</sup> MIC: Message Integrity Code



Fig. 3.2 - Conformación de las claves de red y aplicación [xvi]

Si bien las claves se podrían crear de forma manual o por algún método de creación de llaves; TTN (The Things Network) simplifica este trabajo al usuario y las proporciona una vez que se registra un nuevo nodo en el servidor.

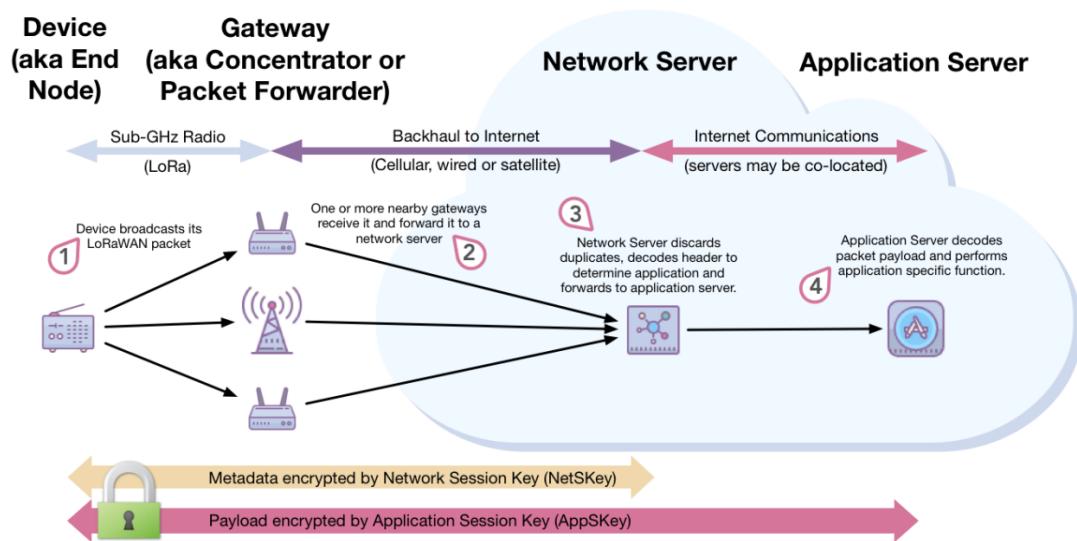


Fig. 3.3 – Encriptación con NwkSKey y AppSKey [xvi]

Como se ve en la figura 3.3, la clave de sesión de aplicación es la que se usa para la encriptación del payload punto a punto y para ello se utiliza el método AES 128.

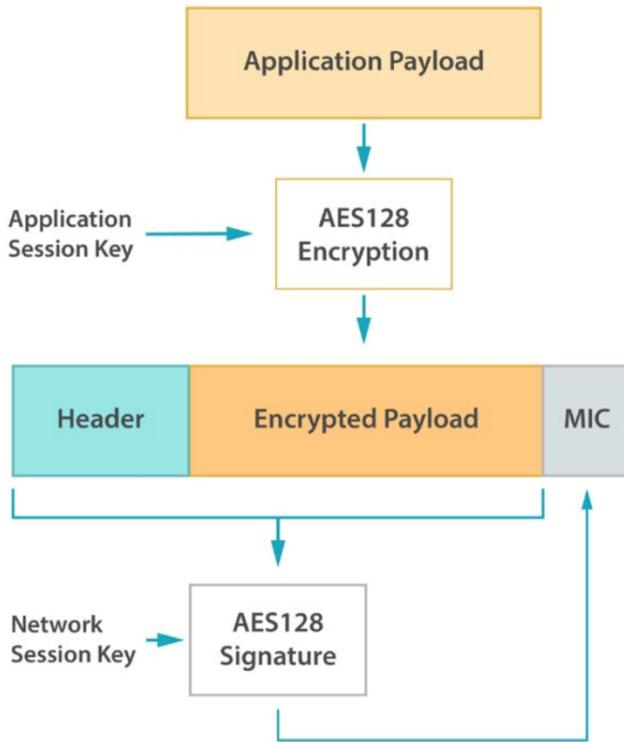


Fig. 3.4 – Diagrama de encriptación competa del Payload [xvi]

Ahora que se sabe la forma de encriptado surge la pregunta ***¿cómo conoce el dispositivo cuáles son las llaves?***

### 3.3.1.3 Métodos de Activación

En la práctica, las llaves pueden obtenerse debido a 3 factores [xvi]:

- 1) Generarse mediante un proceso aleatorio.
- 2) Configurarse manualmente de ambos lados y guardarla en la memoria EEPROM de los dispositivos.
- 3) Negociarla sobre otra plataforma (bluetooth, SMS, etc.)

El punto 3 escapa de lo que es LoRaWAN ya que la negociación se realiza por fuera y en cierto punto está relacionado con la segunda opción debido a que una vez que se determina por medio de otra plataforma; luego se configuran manualmente en la memoria de los equipos. En caso de que se elija esta manera de activación, los dispositivos se deben programar en modo ABP (ver punto 3.3.1.3.1), mientras que si lo hacen como indica el punto 1 la activación se realiza usando OTAA. Ambos conceptos se detallan en las secciones 3.3.1.3.1 y 3.3.2.3.2 respectivamente.

Es necesario aclarar que una vez que se determina una llave, esta será siempre la misma a menos que manualmente se desee cambiarla o que se negocien otras debido a que se cerró la sesión. LoRaWAN no soporta ningún protocolo capaz de modificarla llave cada un determinado tiempo.

### 3.3.1.3.1 ABP

Activation By Personalization (ABP) es un método por el cual las claves (*DevAddr*, *NwkSKey* y *AppSKey*) se configuran de antemano tanto en el concentrador como en el *endpoint*. De esta forma no requieren intercambiar mensajes para la negociación de estas, sino que simplemente una vez que se encienden los equipos, estos ya pueden comenzar a traficar. La figura 3.5 muestra como son los pasos para activación por ABP. La ventaja aquí es que la activación es inmediata, sin embargo, tiene una clara desventaja y es que, en caso de que un tercero obtenga las llaves, podrá interceptar los mensajes y descifrarlos.

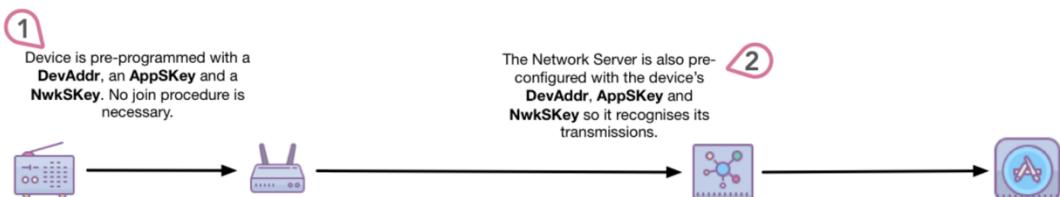


Fig. 3.5 – Secuencia de pasos con activación ABP [xvi]

### 3.3.1.3.2 OTAA

Este método es más complejo que el anterior y en la práctica es el que se recomienda utilizar ya que permite una mayor escalabilidad debido a que se realiza una negociación automática entre los dispositivos y la red [xvi]. En este caso a cada nodo se le configura un *DevEUI*, un *AppEUI*<sup>9</sup> y un *AppKey* único de 128 bits el cual es utilizado para generar las llaves de sesión (*NwkSKey* y *AppSKey*). Una vez que se enciende el dispositivo, este envía un mensaje de *request* el cual no viaja encriptado. Dicho mensaje lo conforma el *AppEUI* y el *DevEUI* de 64 bits cada uno y el *DevNonce* (un número de 16 bits el cual comienza en 0 y va incrementando a medida que se mandan peticiones de *requests*) como muestra la figura 3.6. Estos 3 valores son firmados por 4 bytes (32 bits) de *MIC* a partir de un cálculo matemático.

En caso de que la sincronización no se realice ante el primer mensaje de *request*, este deberá seguir insistiendo hasta lograrlo. Lo importante aquí es que a medida que envía las peticiones, debe ir incrementando en uno el valor del *DevNonce*. De esta forma el *gateway* va guardando el número de peticiones (*DevNonce*) que recibe y en caso de que este valor sea menor al último registrado, lo descarta. Esto lo hace para prevenir que ataques de terceros que mediante el envío de *join-request* previamente grabados, logren desconectar el nodo de la red.

Size (bytes)	8	8	2
Join-request	JoinEUI	DevEUI	DevNonce

Fig. 3.6 - Mensaje de join-request del nodo hacia el servidor [xvii]

<sup>9</sup> En algunas biografías se lo puede encontrar también con el nombre de JoinEUI.

Una vez que un paquete llega a la estación base, esta solo oficia de pasarela hacia el servidor de red. No realiza ninguna verificación, sino que es el servidor quien debe recalcular el MIC con la *AppKey* y en caso de que concuerde le responde al nodo con un mensaje de *join-accept*; de lo contrario lo descarta. El *join-accept* está formado por el *JoinNonce*, el identificador de red *NetID*, la dirección del dispositivo, un campo de 1 byte llamado *DLSsettings* que contiene parámetros de bajada, el *RxDelay* que sirve para determinar el tiempo entre Tx y Rx y un campo opcional de hasta 16 bytes llamado *CFList* que está formado por parámetros de la red.

Size (bytes)	3	3	4	1	1	(16) Optional
Join-accept	JoinNonce	Home_NetID	DevAddr	DLSsettings	RxDelay	CFList

Fig. 3.7 – Mensaje de join-accept [xvii]

El *JoinNonce* es un valor que surge a partir de un contador. Este es utilizado por el *endpoint* para calcular la *AppSKey* y la *NwkSKey* mediante un algoritmo propio de la tecnología. El nodo solo acepta este mensaje de *accept* si y solo si el campo de MIC es correcto y el valor de *nonce* es mayor al último que tiene registro. De acá en más la comunicación pasa a ser encriptada [xvii].

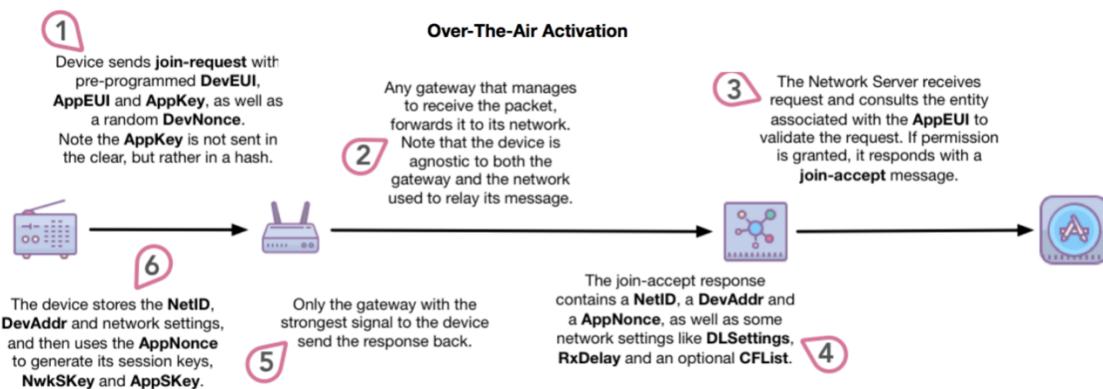


Fig. 3.8 – Secuencia de pasos con activación OTAA [xvi]

Siguiendo con los mensajes del tipo *join*, existen los llamados *rejoin messages* los cuales se intercambian de forma periódica entre el nodo y el *gateway* una vez que la comunicación ya se encuentra establecida. Uno de sus usos es para activar o modificar el *DevAddr* y de esta forma aportarle una mayor seguridad en la red. Por otra parte, sirve para restablecer los parámetros de recepción del dispositivo en caso de que haya habido una desincronización. Esta solicitud de renegociación siempre la inicia el nodo final mediante un mensaje de *rejoin-request*[xvii]. Existen 3 tipos de mensajes de *rejoin-request* los cuales hacen referencia a diferentes cosas. Sus estructuras son como se muestra en la figura 3.9.

Size (bytes)	1	3	8	2
Rejoin-request	Rejoin Type = 0 or 2	NetID	DevEUI	RJcount0
Size (bytes)	1	8	8	2
Rejoin-request	ReJoin Type = 1	JoinEUI	DevEUI	RJcount1

Fig. 3.9 – Tipos se mensajes rejoin-request [xvii]

El de **tipo 0**[xvii] se utiliza para resetear todos los parámetros de radio como el *DevAddr*, los contadores y otra serie de datos necesarios para la comunicación. Estos paquetes son encaminados al servidor de red. Los de **tipo 1**[xvii] son iguales a los *join-request* utilizados para establecer la conexión, pero tienen la particularidad de poder transmitirse sin la necesidad de haber perdido previamente la comunicación. Su utilidad es restaurar una sesión perdida en caso de que el servidor de red haya olvidado las credenciales.

Por último, los de **tipo 2**[xvii] tienen la misma funcionalidad que los de tipo 0 excepto que no resetea todos los parámetros de radio, sino que solo algunos específicos como el *DevAddr* y los contadores. Lo demás permanece incambiado.

Cabe destacar que si bien a cualquiera de los 3 tipos se les genera un MIC para que sea validado por el servidor de red; estos no viajan encriptados. Esto se debe a que en ellos no viaja ninguna información sensible en caso de ser interceptados, pero si es necesario verificar la integridad de este.

### 3.3.2 Clases

Este es uno de los puntos más importantes en cuanto a lo que es el protocolo de comunicación LoRaWAN ya que dependiendo de la clase elegida, es como se va a comportar el nodo. Los tipos de clases existentes son tres (A, B o C) y si bien difieren en varias cosas, la más notoria es en cuanto al consumo de energía que presentan cada una de ellas. En la gran mayoría de los casos se suele utilizar la clase A y de no especificarlo se asume que es esta la aplicada. Sin embargo, es necesario también conocer las clases B y C, las cuales surgieron como una mejora dentro de las redes LoRaWAN y que para algunos casos específicos pueden ser útiles.

Por último, a la hora de decidir por cuál optar, hay que cerciorarse que los *gateways* a los cuales se van a conectar los *endpoints* también acepten esta configuración ya que los equipos de gama baja y/o media suelen ser compatibles solo para los de clase A.

#### 3.3.2.1 Clase A

Esta clase es la que se suele utilizar debido a que, dado las características que presentan las aplicaciones de IoT para las cuales se requiere LoRa, es la que mejor se adapta a ellas.

En primer lugar, el tipo de comunicación que se puede implementar es bidireccional, por lo tanto, permite no solo enviar información del nodo al servidor, sino que también es posible escuchar paquetes y a partir de ello ejecutar una tarea. Si bien en estos casos la mayoría de los paquetes suelen ser enviados por el nodo, es importante que estos sean capaces también de recibir, pensando sobretodo en la necesidad de poder realizar ejecuciones de tareas a distancia.

En cuanto al consumo, dicha clase es la más óptima en caso de que el nodo funcione a batería ya que solamente abre las ventanas de escucha una vez que en nodo haya transmitido previamente un mensaje. El resto del tiempo permanece en modo sleep de forma que en este instante el consumo es mínimo. El grafico de la figura 3.10 muestra como la clase A es la mejor del punto de vista de energía.

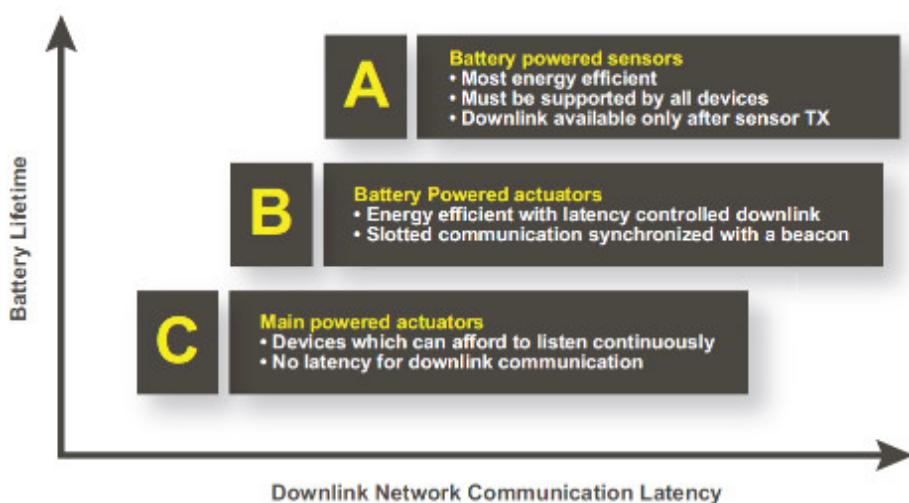


Fig. 3.10 – Relación Batería en función de la Latencia para cada clase [v]

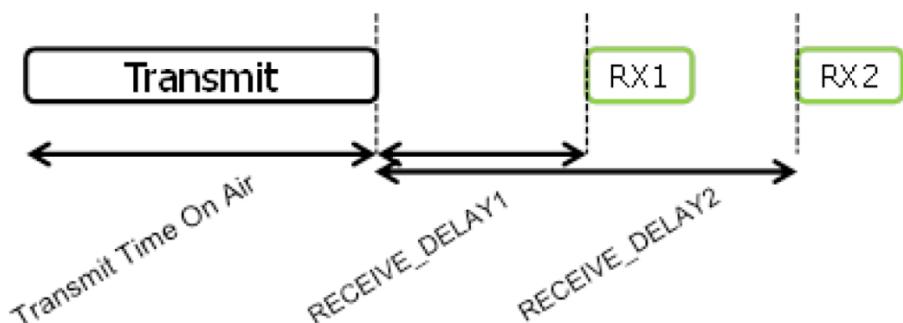


Fig. 3.11 – Funcionamiento de LoRaWAN clase A [v]

Como se puede apreciar en la figura 3.11, la secuencia se compone de una ventana de transmisión, seguida de dos de recepción. En la de transmisión, el tiempo entre una y la siguiente se configura en el nodo (basado en el protocolo ALOHA<sup>10</sup>). Luego de que se envía un paquete, abre dos pequeñas ventanas de recepción. La primera de ellas (RX1) sirve para recibir los datos que le llega desde el servidor. La segunda (RX2) se suele utilizar para complementar a RX1 en caso de que no haya dado el tiempo de recibir toda la información. Si con la primera ventana se trasmittió el mensaje, RX2 no se abre. De esta forma solo puede recibir hasta un máximo de un

<sup>10</sup> Ver anexo C

mensaje por cada uno previamente enviado [xviii]. A su vez, entre la transmisión y la recepción se esperan una serie de delays cuyos tiempos están dados por el campo “RX Delay” de los mensajes de *join-accept* (ver fig. 3.7) y suelen estar en el entorno de los pocos milisegundos con un corrimiento de +/- 20 milisegundos.

La estructura de un paquete enviado por un nodo LoRaWAN clase A es como se muestra en la figura 3.12. El *preamble* cumple la función de especificar donde comienza un nuevo paquete. Seguido a este se encuentran el *PHDR* y *PHDR\_CRC* los cual son un encabezado físico y la redundancia y verificación del anterior respectivamente. Por último, le sigue el *payload* y en caso de que el paquete sea uno enviado por el nodo, se le agrega un campo de *CRC*. Para aquellos que recibe de parte del *gateway*, este último no existe y esto se debe a que se busca de que la trama sea lo más corta posible y así generar un menor impacto en el duty cycle [xvii].

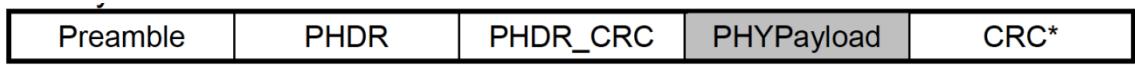


Fig. 3.12 – Estructura de paquete de uplink LoRaWAN [xvii]

Si se va más en detalle, el campo de *payload* (de color gris en la figura 3.12) puede ser de tres tipos, dependiendo si contiene la información propia a mandar, si es un mensaje de *request* o si es del tipo *accept*.

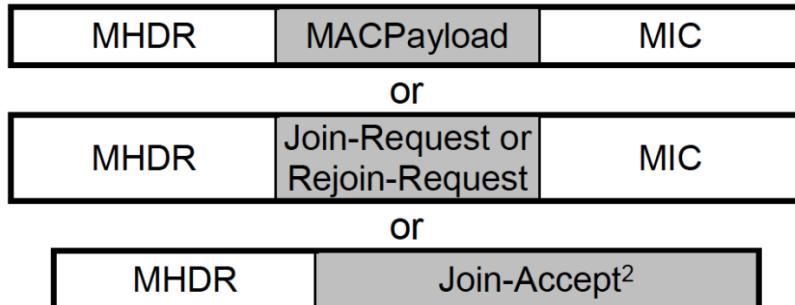


Fig. 3.13 – Posibles estructuras del campo PHYPayload [xvii]

De los tres casos de la figura 3.13, el más interesante es el primero ya que es en el cual que suele transmitir la data. Este tiene un largo variable en función de la cantidad de datos a transmitir. Viendo la estructura del primer caso, el largo del *MHDR* es de 1 byte mientras que el *MIC* es de 4. En cuanto al *MACPayload*, puede tener un tamaño que oscila entre los 7 y los X bytes<sup>11</sup>, siendo X a su vez también variable dependiendo de la región en la que se encuentre funcionando el nodo. Para nuestro caso (US915) el máximo es de 51 bytes [xvii].

En caso de precisar enviar un dato más largo de 51 bytes, se puede realizar un cambio a nivel de librería que permita extender el máximo permitido. De todas formas, esta práctica no se recomienda ya que sería romper con el standard de LoRaWAN y ponerlo a trabajar bajo una exigencia mayor al dispositivo, lo cual implica un aumento en el consumo de su batería y menor vida útil.

---

<sup>11</sup> El mínimo de 7 se debe a que dentro del campo de *MACPayload* no solo se encuentra la data sino también otra información de control como DevAddr y una serie de contadores que entre ellos requieren de 7 bytes.

Por último, la forma que tiene el servidor de distinguir que tipo de mensaje le llegó de los tres posibles de la figura 3.13, es inspeccionando un campo llamado *MType* el cual se encuentra dentro del *MHDR* (*primeros 3 bits*). Los más importantes aquí a tener en cuenta son “000” para un *join-request*, “001” para un *join-accept* y “110” en caso de ser un *rejoin-request*.

### 3.3.2.2 – Clase B

Como se mencionó anteriormente, la clase B es poco utilizada debido a que la gran mayoría de los *gateways* no la soportan. El motivo de esto es porque, en primer lugar, se requiere de una mayor precisión y procesamiento de información ya que no solo deben ser capaces de recibir y transmitir información, sino que también precisan mantener una sincronización de la red mediante paquetes de beacon. En segundo lugar, la clase B surge como una mejora de la anterior; por lo cual este método es más reciente que los del tipo A.

La clase B surge con el propósito de ser utilizados en aplicaciones que requieran que el nodo se encuentre en movimiento y para que esto funcione, ellos deben poder hacer roaming entre varios *gateways* para que la comunicación no se interrumpa. Sin embargo, no es el nodo quien posee la inteligencia de decidir a cuál base conectarse, sino que es el servidor de red el que determina cuál *gateway* es el propicio para establecer la comunicación con el *endpoint*. Esto lo realiza a partir de los niveles de señal que obtiene de los mensajes de *uplink* y *downlink* intercambiados y si el *endpoint* recibe un beacon que difiere con lo que él tiene configurado; este debe mandar un mensaje de *uplink* al servidor para que actualice la información.

La gran diferencia respecto a los de tipo A es que aumenta la cantidad de ventanas de recepción. Sumado a esto, mientras que para la clase A el *gateway* no tiene forma de saber de antemano cuándo recibirá un mensaje por parte del *endpoint*; en los de clase B sí se conoce con exactitud cuándo sucederá esto debido a que el *gateway* envía periódicamente al nodo una serie de mensajes (beacon) que mantiene sincronizados ambos relojes. Estos mensajes a su vez permiten interactuar con el nodo de forma de avisarle a este que abra nueva ventana de recepción (llamada “ping slot”) de forma programada y de largo conocido. Esto es útil cuando se quiere enviar un comando al dispositivo desde el servidor de aplicaciones. Otra diferencia no menor respecto a la clase A, es que en este modo se pueden enviar no solo mensajes unicast, sino también multicast lo cual facilita la comunicación en caso de querer controlar varios dispositivos con un mismo comando.

Todo nodo que se enciende por primera vez comienza traficando información en modo clase A por defecto y luego negocia con la estación base si puede pasarse a clase B. Una vez enviada la solicitud, el *gateway* puede responder afirmativamente mediante un *beacon\_locked* o de forma negativa mediante un *beacon\_not\_found*. En caso de recibir el primero, este comienza a recibir beacons por parte del *gateway*, así como también el nodo se compromete a enviar periódicamente mensajes de control hacia el servidor para que, a partir del nivel de señal de la misma, determinar por cuál puerta de enlace se debe enviar los mensajes de señalización en caso de que el nodo se traslade. Si no se detectan intercambios de dichos mensajes por un tiempo determinado, estos

dejarán de estar sincronizados y por lo tanto volverá automáticamente a trabajar como un nodo clase A. Una vez que se restablece la comunicación y se logran sincronizar nuevamente, vuelve a la clase B[xvii].

La figura 3.14 muestra como es este funcionamiento. Cuando el *endpoint* deja de recibir el primer beacon, este entra en modo “beacon-less” lo cual consiste en ir expandiendo su ventana de recepción para aumentar así la posibilidad de recibir uno nuevamente. En caso retomar la recepción se sincronizan los dispositivos, de lo contrario permanece probando durante un máximo de 2 horas. Una vez cumplido este plazo y no lograr recibir otro beacon, es aquí donde se pasa automáticamente a clase A.

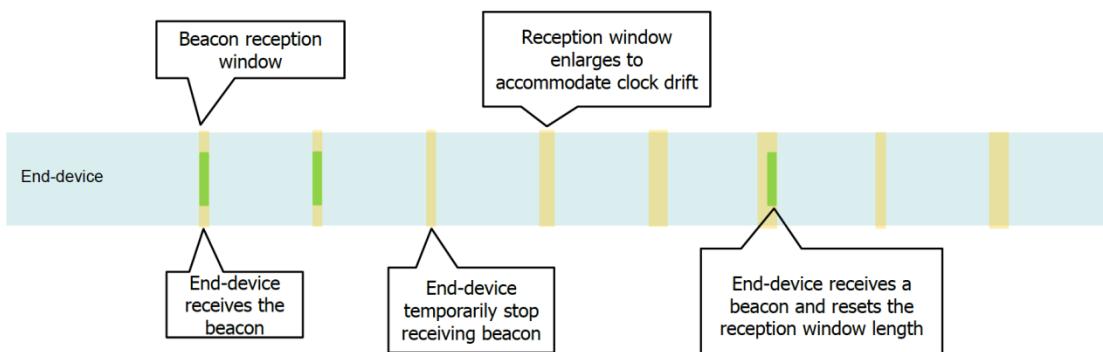


Fig. 3.14 – Ensanchado de ventanas de recepción ante pérdida de beacons [v]

Debido a la mayor cantidad de ventanas de transmisión y recepción que se utilizan en estos tipos de nodos y los constantes mensajes necesarios para mantener a estos sincronizados, es que el consumo de batería es mayor a los de la clase anterior. Sin embargo, todavía es admisible pensar en alimentar dichos módulos con baterías.

### 3.3.2.3 – Clase C

Al igual que para el caso de los de clase B, existen muy pocos *endpoints* que estén configurados en clase C. Los motivos son los mismos que para los del tipo B, sumado a que en este caso el consumo de batería es mucho más elevado ya que el 100% del tiempo se encuentra o escuchando o enviando mensajes. Por ello es por lo que solo se recomienda utilizar esta clase si los nodos se encuentren conectados directamente a la red eléctrica y no por medio de baterías. Como se puede ver en la figura 3.15, estos nunca entran en modo sleep ya que cuando no se encuentran enviando información, permanecen activos a la espera de recibir algún posible paquete.

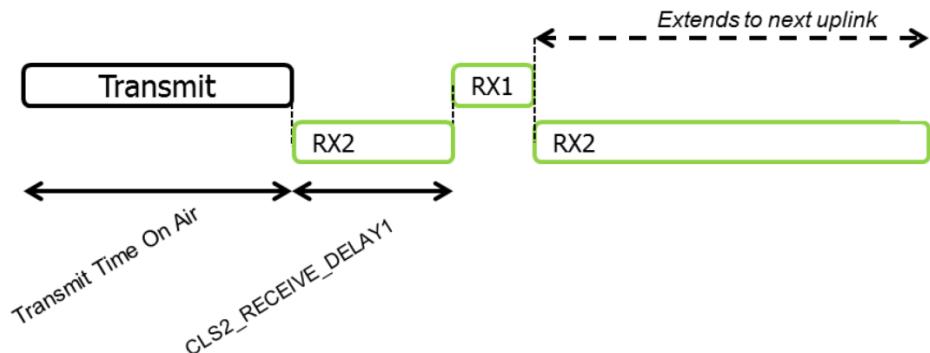


Fig. 3.15 – Funcionamiento de nodos clase C [v]

Si se compara con los nodos de clase A, se puede ver que el comportamiento de las ventanas de transmisión y recepción son muy similares, excepto que en este caso no se dejan delays entre ellas y la última RX2 permanece activa hasta el próximo mensaje de *uplink*.

### 3.3.3 Frecuencias

Como se vio en la tabla 2.1, LoRaWAN utiliza frecuencias por debajo del giga-hertz (sub giga-hertz) lo cual permite alcanzar largas distancias y tener buen poder de penetración de la señal ante objetos interferentes que se encuentran entre el emisor y el receptor.

Si bien utiliza parte del espectro libre para la transmisión y recepción de los paquetes, existen tres grandes bandas de frecuencias donde se suele utilizar esta tecnología. Estas son 433 MHz, 868 MHz y 915 MHz [xix]. La elección de cuál utilizar la hace el usuario en función del país en el cual vaya a implementar la red. Si bien al ser bandas ISM no hay una regulación de cual usar, se recomienda fuertemente la utilización de aquella que corresponda a la región [v]. Para este trabajo se respetó dicha recomendación y siguiendo con el standard propuesto se eligió valerse de la banda de 915 MHz que es la misma que se utiliza en Estados Unidos y gran parte de los países de Sudamérica. El nombre técnico es US 902-928 donde “US” refiere a United States mientras que “902-928” es en ancho del espectro centrado en 915 MHz. Ejemplos de otras son la EU863-870, EU433, CN470-510, CN779-787, AU915-928, AS920-923, AS923-925, KR920-923 e IN865-867 [xiv]. Todas estas son las que existen hasta la actualidad.

En la tabla 3.3 se muestra una tabla comparativa entre Europa y Norte América de forma de ver que, dependiendo de la banda, no solo cambia el rango de frecuencias donde se trafica, sino que también otros parámetros como el SF, las potencias, el ancho y cantidad de canales disponibles.

	Europe	North America
<b>Frequency Band</b>	876 – 869 MHz	902 – 928 MHz
<b>Channels</b>	10	64+8+8
<b>Channel BW Up</b>	125/250 kHz	125/250 kHz
<b>Channel BW Dn</b>	125 kHz	+20 500 kHz
<b>Tx Power Up</b>	+14 dBm	+20 dBm typ (+30 dBm allowed)
<b>Tx Power Dn</b>	+ 14 dBm	+27 dBm
<b>SF Up</b>	7-12	7-10
<b>Data Rate</b>	250bps-50kbps	980bps-21.9kbps
<b>Link Budget Up</b>	155 dB	154 dB
<b>Link Budget Dn</b>	155 dB	157 dB

Tabla. 3.3 – Tabla comparativa entre EU867 – 869 y US 902 – 928 [xvii]

Dicho esto, una banda está formada por una serie de canales de *uplink* y *downlink*, los cuales se utilizan para poder transmitir y recibir información desde varios dispositivos de forma simultánea. Un mismo equipo no trafica información siempre por el mismo canal, sino que va variando de uno en otro para no acaparar el espectro.

En lo que refiera a la banda US 902-928, esta se encuentra conformada por 64 canales de *uplink* de 125 KHz de ancho cada uno (numerados del 0 al 63), comprendidos entre 902,3 y 914,9 MHz y separados entre sí por 200 KHz de ancho como muestra la figura 3.16. Adicionalmente hay 8 canales más de *uplink* de 500KHz entre 903 MHz y 914,2 MHz (numerados del 64 al 71) separados por 1,6 MHz los cuales se utilizan para aplicaciones específicas. En cuanto a los de *downlink*, también existen 8 pero en este caso son de 500 KHz de ancho y van entre los 923,3 MHz y 927,5 MHz[xix].

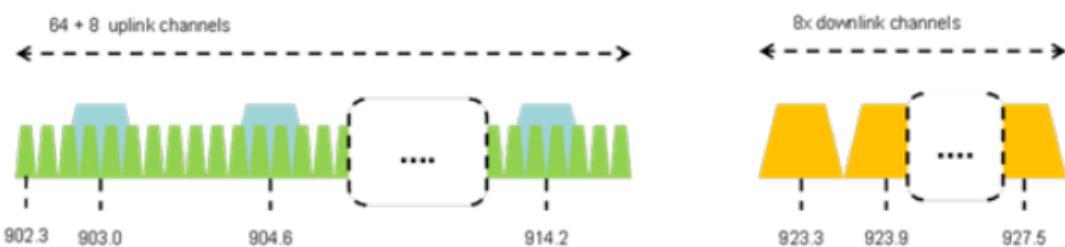


Fig. 3.16 – Bandas de frecuencia de US 902-928 [xvii]

En el anexo D se puede apreciar la tabla completa de frecuencias con sus respectivos canales y ancho de banda. Si bien hay muchos más canales de subida que de bajada, se suelen utilizar únicamente 8 para *uplink* y otros 8 para *downlink*. Como de

bajada solamente hay 8, estos son los que se utilizan; mientras que, para la subida, el usuario debe seleccionar qué sub-banda de canales desea usar. En nuestro caso hemos optado por la número uno<sup>12</sup> la cual comprende frecuencias entre 903,9 MHz y 905,3 MHz y que en el anexo D está marcada de color celeste.

Por último, es bueno recalcar que LoRa no trabaja en una frecuencia fija, sino que lo hace en un rango, por lo cual, en caso de que exista una interferencia en alguna de ellas, el sistema podrá seguir funcionando sin problemas ya que cuenta con el resto para poder comunicarse sin inconvenientes. A su vez, al haber 8 canales (o sub-bandas) disponibles para *uplink*, el usuario puede utilizar la que le resulte más óptima.

### 3.3.4 ADR (Adaptive Data Rate)

ADR es una funcionalidad que se utilizó en dicho trabajo para hacer de los nodos un dispositivo lo más eficiente posible ya que permite optimizar el data rate, el airtime y el consumo de batería.

Para entender este concepto se pone como ejemplo un nodo que se encarga de monitorear un sitio en un estacionamiento. Este sensor se encuentra fijo en el suelo por lo cual a priori el nivel de señal no debería variar con el tiempo. Sin embargo, cuando un auto se estaciona allí, este se interpone entre el emisor y el receptor, por lo cual, la señal se ve alterada. En este caso el *endpoint* debe ser capaz de interpretar esto y modificar los parámetros de la comunicación para que sea lo más eficiente posible. Este comportamiento se realiza a nivel de librerías y no en el código del nodo en sí. Con esta función activada, el dispositivo presenta un mejor rendimiento ya que a partir de los últimos 20 mensajes de *uplink*, este es capaz de detectar algún cambio en el nivel de señal e ir tomando decisiones que influyen positivamente en el data rate. Para ello mide el nivel de señal-ruido y lo compara con la SNR requerida para demodular a dicha velocidad. A partir de esta diferencia (margen) toma la decisión de aumentar el data rate o disminuir la potencia de transmisión según corresponda.

A modo de ejemplo, si la red recibe un mensaje con un SF de 12, una relación SNR de 5 y un margen de 25 dB<sup>13</sup>, el nodo será capaz de detectar que se está haciendo un uso excesivo de la potencia de transmisión y tomará medidas para optimizar esto. Estas consisten en, por ejemplo, disminuir el valor del spreading factor. Al hacer esto todavía se seguirá teniendo un buen margen para una comunicación óptima, pero consumiendo menos energía. Para estos casos es el nodo quien tiene la inteligencia de interpretar dicho comportamiento y tomar la decisión de adaptar el data rate.

### 3.3.5 Duty Cycle

El duty cycle hace referencia a la fracción de tiempo durante el cual el nodo está ocupado; es decir, está enviando o recibiendo información y, por consiguiente, se encuentra ocupando un canal. Para un buen uso de la red, este tiempo debe estar regulado y limitado por algún agente externo de forma que un nodo no pueda acaparar el canal durante mucho tiempo generando así congestión en la red. A modo de ejemplo,

---

<sup>12</sup> Las sub-bandas agrupan de a 8 canales y van numeradas de 0 a 7

<sup>13</sup> 10 dB de margen es lo recomendado

en Europa existe el standard ETSI EN300.220 el cual se encarga de controlar esto. Este valor depende de muchos factores entre los cuales se destaca la región donde se encuentre funcionando el nodo. Para el caso de US-915 el duty cycle es de 1%, lo cual quiere decir que, si un nodo acaparó X milisegundos de un canal para traficar información por él, luego de debe esperar 99 veces X para volver a utilizarlo. La ecuación 3.2 muestra de forma más simple cómo se calcula el tiempo de indisponibilidad del canal [xx].

$$Toff = \left( \frac{\text{Time on air}}{\text{Duty Cycle}} \right) - \text{Time on Air} \quad (3.2)$$

### 3.3.6 Topología de Red

Si pensamos en LoRa esta puede ser tanto punto-punto, punto-multipunto, como también presentar una topología mesh donde todos los dispositivos se comunican con todos. De esta forma no requieren de ningún equipo intermedio para hablar entre sí. Sin embargo, en LoRaWAN esto es distinto. Aquí está presente la figura del *gateway* que cumple el rol de intermediario y por el cual pasan todos los paquetes. Es por tal motivo que el tipo de topología de las redes LoRaWAN es en **estrella** donde el centro de la misma es justamente el *gateway*. A su vez cada centro de estrella se puede comunicar con otra mediante algún protocolo de comunicación TCP-IP, aunque lo más común de encontrar es que cada una de ellas esté directamente conectada a internet.

No existe un número determinado de la cantidad de dispositivos que puede soportar un mismo *gateway* ya que depende de varios factores como por ejemplo el tiempo en el aire de cada paquete que recibe, el tiempo entre que se envía un mensaje y el siguiente, la cantidad de canales que puede manejar (por lo general suelen ser 8 de subida y 8 de bajada, aunque hay equipos más potentes que permiten hasta 16 de *uplink*), el largo del payload de los mensajes y la cantidad de canales que puede escuchar en simultáneo, entre otros factores.

Para entender cómo calcular el máximo número de nodos que puede soportar un *gateway*, la manera más ilustrativa es mediante un ejemplo. Se supone para este caso que todos los *endpoints* de la red se comportan de la misma manera, los cuales envían 10 bytes de información espaciados cada 5 minutos con un SF de 8 de forma que el time on air de cada uno de ellos es de 114 milisegundos. Por lo tanto,  $\frac{(24 \times 60)}{5} = 288$  mensajes por día, lo que equivale a 33 segundos usando el canal. Si se supone que el duty cycle es de 5%, mediante la ecuación 3.2 se obtiene que un mismo canal puede utilizarse cada 660 segundos. Como el *gateway* utiliza 8 canales para recibir paquetes y puede escuchar en ellos a la vez, de la ecuación 3.3 se desprende que puede soportar hasta un máximo de 1047 dispositivos conectados a él.

$$\left( \frac{24 \times 60 \times 60}{660} \right) \cdot 8 = 1047 \quad (3.3)$$

Sin embargo, este valor puede variar dependiendo de dónde se implemente, así como también el equipo utilizado ya que dependiendo del equipo puede escuchar hasta

en 16 canales a la vez, como también uno por turno [xxi]. Dicho esto, se pasa a detallar el resto de la red, más allá de la topología estrella mencionada anteriormente.

Los *endpoints* envían la información al concentrador de forma inalámbrica modulada en LoRa con un único salto de distancia entre ellos y luego éste se comunica vía TCP-IP a los servidores de red y aplicación correspondientes. Para el caso de los mensajes *downlink*, el proceso es el inverso, ya que los paquetes salen con destino hacia el/los nodos desde el servidor, pasando previamente por el *gateway* central. La figura 3.17 muestra como es la topología típica de una red LoRaWAN.

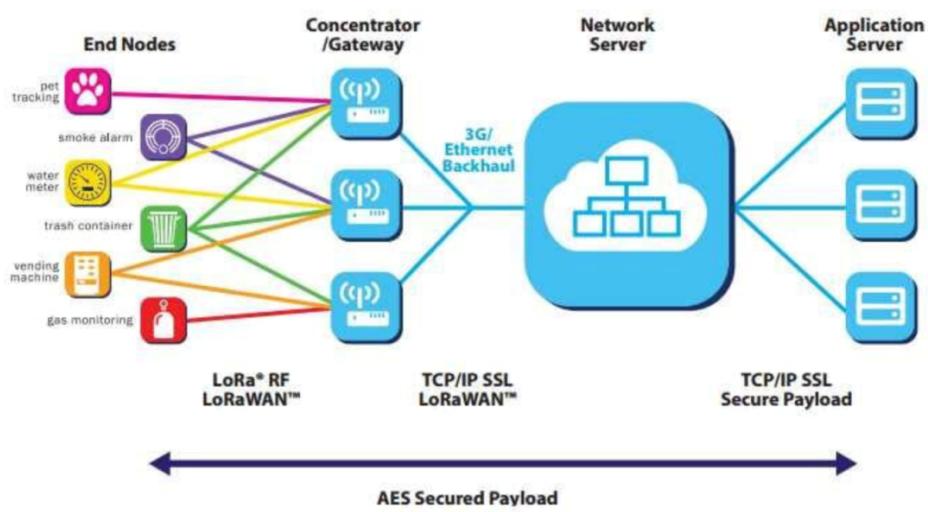


Fig. 3.17 – Topología LoRaWAN [xv]

Una red puede ser implementada de tres tipos:

Basada en el operador: Modelo parecido al de un proveedor de telefonía celular donde este realiza el despliegue de la red y luego les vende a los interesados la posibilidad de conectarse y traficar por ella.

Privada: Aquí una empresa o un particular monta su propia red privada y tiene el control y uso de esta. En estos casos se suelen utilizar equipamientos más económicos que en el modo anterior.

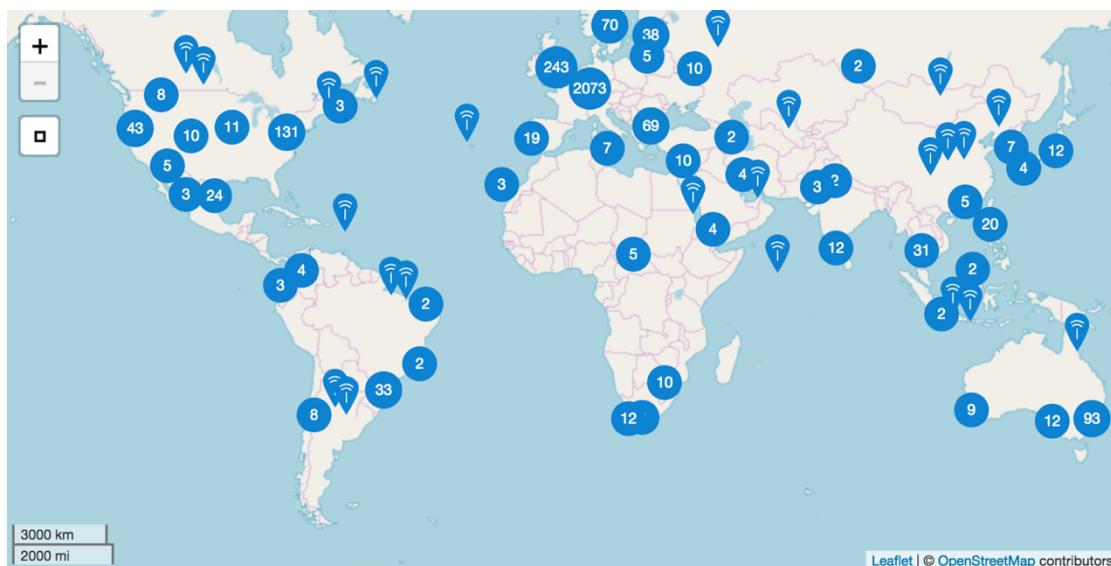
Híbrido: Debido a su uso libre del espectro, LoRaWAN es de las pocas tecnologías que permiten este modo. Consiste en que el operador proporciona la cobertura exterior, pero ante la solicitud de las empresas o particulares interesados, se les brinda una mayor cobertura para cubrir sus necesidades. Esto se debe a que mediante las claves de red y aplicación se permite el pasaje de paquetes propios por medio de puertas de enlace de terceros, de forma que se adopta un uso compartido de la red.

## 3.4 TTN

TTN o The Things Network (<https://www.thethingsnetwork.org>) es una plataforma gratuita formada por más de 46.000 miembros cuya finalidad es brindar a los

usuarios un servidor de red y de aplicaciones [xxii]. Con él, se puede tener un registro de los mensajes que envía o recibe el *gateway*, así como también poder interpretar el payload enviado por el nodo en una primera instancia. A su vez, brinda la posibilidad de interactuar con servidores externos propios, base de datos, IFTTT (If This, Then That) u otras integraciones que pueden ser beneficiosas para llevar un control detallado de la red.

Uno de los principales objetivos de sus creadores fue desplegar una red LoRaWAN mundial con la ayuda de los propios usuarios, los cuales “cuelgan” su *gateway* a la red y estos pasan a ser de público conocimiento. Esto, lo único que implica es que un nodo propio puede tráfico (en caso de tener las llaves necesarias) a través del concentrador de un tercero, pero como los paquetes viajan cifrados, la comunicación sigue siendo segura ya que solo puede ser leída por su dueño quien conoce sus credenciales. Hasta agosto de 2018 se llevan documentados más de 4300 *gateways* en la red TTN a lo largo del planeta ubicados principalmente en Europa y Norteamérica [xxii]. Este despliegue se muestra en la figura 3.18.



Como se puede apreciar en la figura 3.19, TTN permite no solo funcionar como servidor de red y de aplicaciones, sino que también facilita la integración a otras plataformas (Amazon Web Services, Microsoft Azure, IFTTT, entre otros) mediante protocolos estándar muy utilizados en la actualidad o mismo exportar la información a una API ubicada en un servidor propio por medio de http [xxiii].

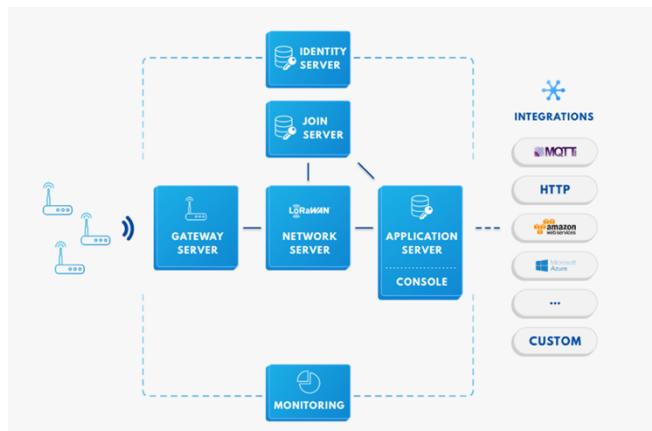


Fig. 3.19 – Diagrama de TTN [xxii]

En la actualidad, The Things Network ya es parte importante de LoRaWAN debido a su gran popularidad y constante evolución llevada a cabo por sus propios desarrolladores. Cuenta con un foro (<https://www.thethingsnetwork.org/forum>), el cual es sumamente activo donde diariamente se suben consultas y son respondidas por los mismos usuarios. Por último, cabe destacar el importante vínculo que existe entre ésta y el repositorio GitHub ya que, al ser ambas plataformas open source, es muy común encontrar códigos que se vinculan con TTN.

Para este proyecto fue necesario utilizar un servidor de red y de aplicaciones y se optó por TTN debido a que si bien tiene ciertas limitaciones (las cuales se verificarán mediante pruebas empíricas en el capítulo 6); estas no afectaban el desarrollo del trabajo y por sobre todas las cosas es muy sencilla de utilizar y de uso gratuito. En caso de querer utilizar otro servidor, existen varios en el mercado que cumplen este rol. Un ejemplo de esto son aquellos que trabajan sobre la nube como AWS (<https://aws.amazon.com/>) o Microsoft Azure for IoT (<https://azure.microsoft.com/>), pero para su utilización se precisan licencias las cuales, si bien no son excesivamente caras, este gasto escapaba del presupuesto previsto para el proyecto. De todas formas, se cree que en caso de realizar una implementación de mayores dimensiones o a nivel profesional, cualquiera de estas sería una buena opción a tener en cuenta.

Por otro lado, están los servidores propios de las marcas de los gateways como Cisco IoT Cloud Connect (<https://www.cisco.com/c/en/us/solutions/service-provider/iot-cloud-connect/index.html>) o el servidor de Kerlink (<https://www.kerlink.com/>), los cuales también están implementados en la nube, son pagos y se recomienda usarlos solo en caso de contar con un equipo de la marca.

Por último, se encuentran disponibles una serie de servidores gratuitos que permiten ser usados por cualquier usuario, pero no son tan completos como los anteriores, sino que tienen ciertas restricciones. En esta lista se encuentra el Kaa (<https://www.kaaproject.org/>). Este fue uno de los que se tuvo en cuenta ya que cumple con los requisitos básicos para lo que pretende alcanzar este proyecto. Sin embargo se vio que la versión de prueba gratuita solo permitía crear una única aplicación. En caso de querer tener más era necesario adquirir licencias que iban entre los 10 y 50 dólares mensuales. Este fue el factor por el cual se desistió de su uso y se optó por TTN, ya que

no solo es gratuita y fácil de utilizar; sino que también no cuenta con limitantes en cuanto a la cantidad de nodos y aplicaciones que se le pueden utilizar.

Una vez definido TTN como servidor a utilizar en este proyecto, es necesario mencionar algunas particularidades del mismo. En primer lugar, se habla de que TTN presenta una limitante en cuanto a la cantidad de paquetes de *downlink* que se pueden enviar (10mensajes nodo por día). Por otro lado, si hablamos de los mensajes de *uplink* el servidor permite hasta 30 segundos de tiempo en el aire por nodo por día[xx]. Sabiendo el tiempo en el aire que demora cada paquete (en el entorno de los ms), es posible determinar cuántos paquetes se pueden transmitir en ambos sentidos por día y así calcular en el nodo el tiempo entre envíos para no superar esta limitante diaria.

### 3.4.1 Registro de equipos en TTN

Una vez que se eligió el servidor a utilizar, se deben registrar los equipos en él, ya sean el o los *gateways* como también los nodos con sus respectivas aplicaciones. Esta tarea es muy importante no solo para que el servidor pueda interpretar los mensajes que recibe de ellos, sino que también para brindar al usuario las claves necesarias a aplicar en el *endpoint*.

Para este trabajo se realizaron 5 aplicaciones diferentes cuyas funciones para decodificar el payload de los mensajes recibidos en el servidor fueron programadas en lenguaje Java Script como se indican en las siguientes secciones.

#### 3.4.1.1 Aplicación de Número Aleatorio

La finalidad de esta es generar un número aleatorio entre 0 y 255 y enviarlo al *gateway*. La función que se programó en el servidor de aplicaciones para interpretar el payload es la que se muestra en la figura 3.20.

```
function Decoder(bytes, port) {  
  var data00 = bytes[0];  
  
  resultado = data00  
  return {  
    Dato_Recibido: resultado,  
  };  
}
```

Fig. 3.20– Función de decodificación de aplicación Número Aleatorio

#### 3.4.1.2 Aplicación de Conteo

Dicha aplicación se encarga de generar y enviar un número de forma ordenada y creciente de 1 hasta 65535. Fue pensada principalmente para realizar pruebas de pérdidas de paquetes ya que leyendo el log que generaron los mensajes recibidos, es fácil ver si algún mensaje no llegó a destino. A diferencia de la de “Número Aleatorio”,

este debe ser formado por dos bytes y la función de decodificación siguientes la que se muestra en la figura 3.21.

```
function Decoder(bytes, port) {
    var data00 = (bytes[0] & 0x01)*256;
    var data01 = (bytes[0] & 0x02)*256;
    var data02 = (bytes[0] & 0x04)*256;
    var data03 = (bytes[0] & 0x08)*256;
    var data04 = (bytes[0] & 0x10)*256;
    var data05 = (bytes[0] & 0x20)*256;
    var data06 = (bytes[0] & 0x40)*256;
    var data07 = (bytes[0] & 0x80)*256;

    var data8 = (bytes[1] & 0x01);
    var data9 = (bytes[1] & 0x02);
    var data10 = (bytes[1] & 0x04);
    var data11 = (bytes[1] & 0x08);
    var data12 = (bytes[1] & 0x10);
    var data13 = (bytes[1] & 0x20);
    var data14 = (bytes[1] & 0x40);
    var data15 = (bytes[1] & 0x80);

    resultado = data00 +data01 +data02 +data03 +data04 +data05 +data06 +data07 +data8 +data9 +data10 +data11 +data12 +data13 +data14 +data15;

    return {
        Dato_Recibido: resultado,
    };
}
```

Fig. 3.21 – Función de decodificación de aplicación Conteo

### 3.4.1.3 Aplicación de Palabra Fija

La aplicación de palabra fija permite al usuario poder enviar un texto fijo y no podrá ser modificada a menos que se le cargue una nueva configuración. La condición que se debe respetar es que la frase a mandar no supere los 51 caracteres ya que esto es lo máximo que permite la librería y el protocolo de comunicación para este caso. Con esto se muestra que es posible enviar información más allá de un simple número de unos pocos bytes y su función para decodificar el payload es la que se muestra en la figura 3.22.

```
function Decoder(bytes, port) {
    data="";
    for (var i = 0; i < bytes.length; i++)
        var data = data + String.fromCharCode(bytes[i]);

    resultado = data
    return {
        Dato_Recibido: resultado,
    };
}
```

Fig. 3.22 – Función de decodificación de aplicación Palabra Fija

### 3.4.1.4 Aplicación de Humedad y Temperatura

Esta aplicación, como lo indica su nombre es la responsable de poder interpretar el mensaje que manda un nodo conectado al sensor de humedad y temperatura DHT11. Al igual que para el caso de la aplicación de conteo, el payload lo forman 2 bytes. En el primero viaja la humedad porcentual del ambiente y en el segundo la temperatura. La figura 3.23 muestra el código que se utilizó para implementar la función.

```

function Decoder(bytes, port) {

    var data00 = bytes[0];
    var data01 = bytes[1];
    resultado1 = data00
    resultado2 = data01
    return {
        Humedad: resultado1 + " %",
        Temperatura: resultado2 + " °C"
    };
}

```

Fig. 3.23 – Función de decodificación de aplicación Humedad y Temperatura

### 3.4.1.5 Aplicación de Luz

Por último, se encuentra la aplicación capaz de poder interpretar el mensaje enviado desde un *endpoint* conectado a un sensor que mide la cantidad de luxes que hay en un ambiente. La escala de este varía entre 0 y 65535 y su función de decodificación es la de la figura número 3.24.

```

function Decoder(bytes, port) {
    var data00 = (bytes[0] & 0x01)*256;
    var data01 = (bytes[0] & 0x02)*256;
    var data02 = (bytes[0] & 0x04)*256;
    var data03 = (bytes[0] & 0x08)*256;
    var data04 = (bytes[0] & 0x10)*256;
    var data05 = (bytes[0] & 0x20)*256;
    var data06 = (bytes[0] & 0x40)*256;
    var data07 = (bytes[0] & 0x80)*256;

    var data10 = (bytes[1] & 0x01);
    var data11 = (bytes[1] & 0x02);
    var data12 = (bytes[1] & 0x04);
    var data13 = (bytes[1] & 0x08);
    var data14 = (bytes[1] & 0x10);
    var data15 = (bytes[1] & 0x20);
    var data16 = (bytes[1] & 0x40);
    var data17 = (bytes[1] & 0x80);

    resultado = data00 +data01 +data02 +data03 +data04 +data05 +data06 +data07 +data10 +data11 +data12 +data13 +data14 +data15 +data16 +data17;

    return {
        Dato_Recibido: resultado,
    };
}

```

Fig. 3.24 – Función de decodificación de aplicación Luz

# Capítulo 4

## Endpoints

Estos son los dispositivos finales los cuales fueron programados como clase A para realizar una comunicación bidireccional con el *gateway* mediante protocolo LoRaWAN. A continuación, se detallará cómo se hizo el diseño e implementación del hardware y software de estos.

### 4.1 Software

La programación de los dispositivos finales se llevó a cabo en lenguaje C++ sobre la plataforma de Arduino 1.8.4. Este se puede descargar desde el siguiente link: <https://drive.google.com/open?id=1tindiriRII2K4bv5Viim8dfQDs5qXicx>

Las librerías utilizadas fueron:

- *SPI.h*: Capaz de gestionar la comunicación SPI para la compilación y programación de la memoria no volátil del microprocesador.
- *Lmic.h*: Librería para modulación LoRa e implementación LoRaWAN clases A y B la cual fue creada por IBM y que en la actualidad es abierta a los programadores y se encuentra bajo la gestión del usuario “matthijskooijman” del repositorio GitHub.
- *BH1750FVI.h*: Contiene las funciones para el sensor de luz BH1750.
- *DHT.h*: Contiene las funciones necesarias para el sensor de humedad y temperatura DHT11.

Si bien en un principio se decidió que el único propósito de estos dispositivos sea enviar un paquete similar a un ping, se vio que sería más enriquecedor programarlos para que sean capaces de tomar información de algún sensor y transmitirla al *gateway*. Así entonces se buscó crear un software fácilmente escalable en caso de que en un futuro se deseé enviar otro tipo de dato o conectar el nodo a otro sensor compatible con el microcontrolador. Siguiendo con esta idea, se tomó la decisión de realizar un solo archivo.ino<sup>14</sup> el cual contempla diferentes tipos de información a enviar (dependiendo del sensor) y no un archivo por cada uno. De esta forma es como si hubiera 5 códigos dentro de uno solo donde seleccionando una única línea de código se elige cual cargar. Los posibles programas son los mencionados en los puntos 3.4.1.

## 4.2 Hardware

### 4.2.1 Placa de prototipo

En la primera etapa de diseño de hardware, se decidió producir un prototipo genérico compatible con distintos sensores y capaz de enviar la información obtenida de éstos o generada por otros medios, a través de un módulo de comunicación LoRa RFM95W.

Como módulo de procesamiento se utilizó un Arduino Pro Mini como el de la figura 4.1, dado que sus prestaciones permiten hacer todas las pruebas necesarias. Además, existe una gran comunidad trabajando con el mismo (<https://forum.arduino.cc/index.php?board=32.0>). Este microcontrolador está basado en el microprocesador ATmega328 [xxiv] de la marca Atmel, el cual cuenta con 14 entradas/salidas digitales, 6 entradas analógicas, resonador, botón de reset e indicadores leds. Funciona a 3,3V Y 8MHZ.A su vez, cuenta con 5 modos de bajo consumo, además de características tales como su muestran en la tabla 4.1:

---

<sup>14</sup> Los archivos .ino son la extensión soportada por los Arduino

<b>Memoria Flash</b>	32Kbytes
<b>Memoria SRAM</b>	2Kbytes
<b>Cantidad de pines</b>	28
<b>Frecuencia máxima de operación</b>	20 MHz
<b>CPU</b>	8-bit AVIR
<b>Pines máximos de E/S</b>	23
<b>Interrupciones Internas</b>	24
<b>SPI</b>	1
<b>UART</b>	1
<b>Canales ADC</b>	8
<b>Resolución de ADC</b>	10
<b>Eeprom</b>	1K
<b>Canales PWM</b>	6
<b>Voltaje de Operación</b>	1,8 – 5,5 V
<b>Timers</b>	3

Tabla 4.1 – Características del micro controlador

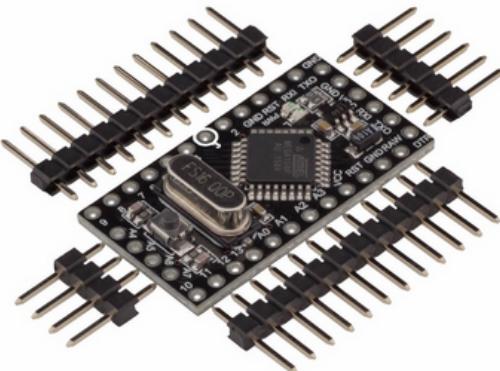


Fig. 4.1 – Micro controlador Arduino Pro Mini 3,3V – 8MHz

El otro elemento principal del sistema es el módulo de comunicación LoRa, figura 4.2, del cual se utilizará una versión basada en el chip RFM95W. Este módulo transmite a una frecuencia de 915 MHz, permite modulaciones LoRa, FSK, GFSK, OOK. Tiene un puerto de comunicación SPI, la potencia de transmisión es de 20 dB a una velocidad de 200 Kbps y asegura una distancia de envíos mayor a 8000m. El fabricante muestra consumos de 120 mA en transmisión, 10.3 mA en recepción y 0.2 uA en standby alimentándose con un voltaje entre 1,8 y 3.7V DC [xxv].

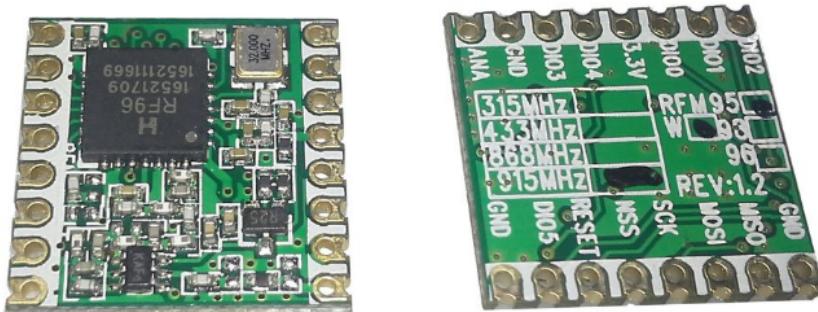


Fig. 4.2 – Módulo de comunicación LoRa RFM95W

El módulo anteriormente mencionado se utiliza asociado a una antena helicoidal, como la de la figura 4.3, específica para la frecuencia 915Mhz de 2.5 dBi [xxvi].



Fig. 4.3 – Antena Helicoidal 915Mhz

También se precisaron distintos sensores, como el DTH-11 de la figura 4.4. Este brinda datos de temperatura y humedad ambiente de forma digital a través de una salida de bus único y se alimenta con un rango de 3.3 a 5.5V DC [xxvii].

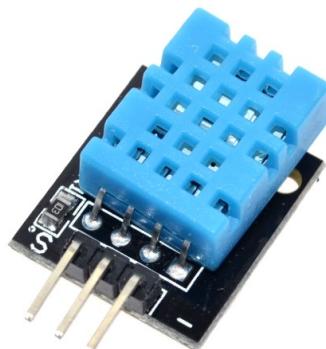


Fig. 4.4 – Sensor de humedad y temperatura DHT-11

Otro de los sensores elegidos fue el de intensidad de luz BH1750 que se muestra en figura 4.5 el cual tiene un rango de alimentación de 3 a 5 VDC, con modo de bajo consumo de energía. Cuenta con salida digital directa y su respuesta espectral es similar a la del ojo humano [xxviii].

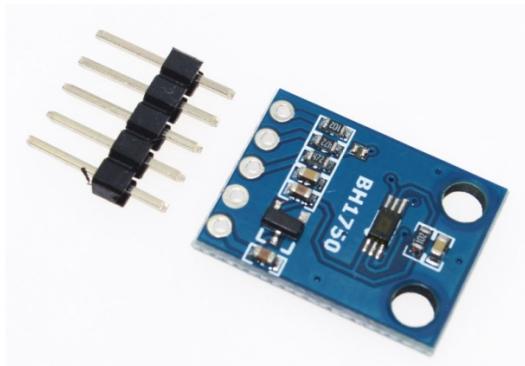


Fig. 4.5 – Sensor de luz BH1750

Para medir la presión barométrica se utilizó el BMP280, de la figura 4.11, el cual es un sensor alta precisión que utiliza el protocolo de comunicación IIC/SPI y tiene modo de bajo consumo [xxix].

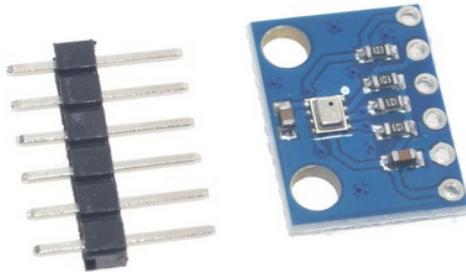


Fig. 4.6 – Sensor de presión BMP280

Todos estos equipos se conectaron en una PCB diseñada a medida a través del software CAD Eagle de Autodesk. En la misma, primero se diseñó el esquemático del circuito y luego la disposición de la placa como se muestra en las figuras 4.7 y 4.8 respectivamente.

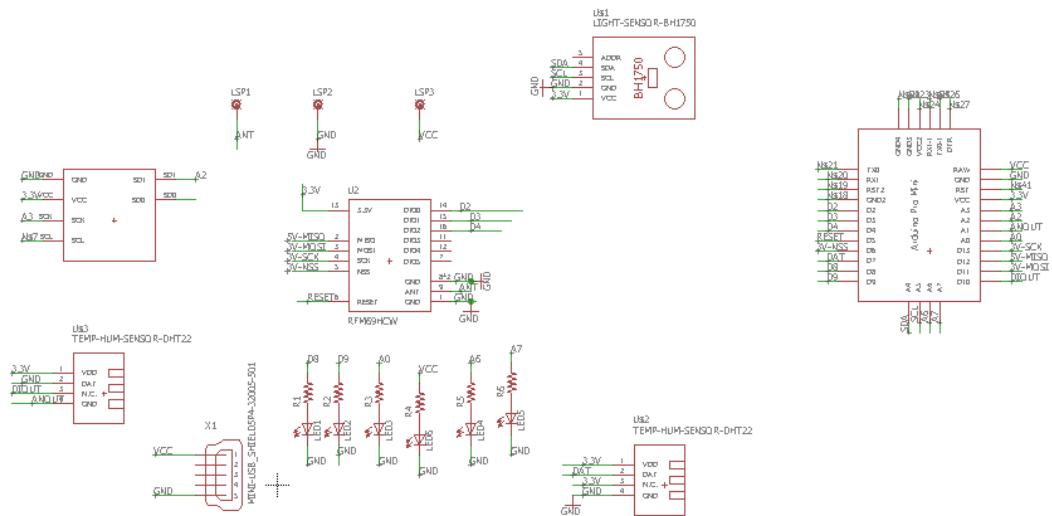


Fig. 4.7 – Esquemático de la placa de prototipo realizado en Eagle

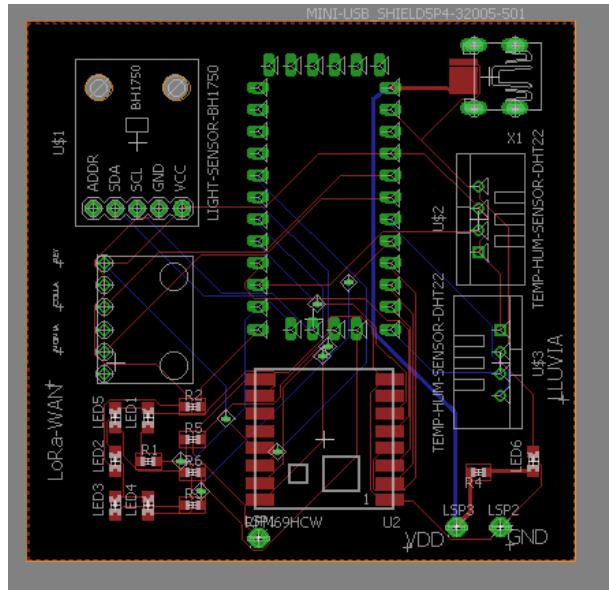


Fig. 4.8 – Disposición de los componentes de la placa de prototipo realizado en Eagle

La imagen 4.8 muestra el diseño de la PCB. Las conexiones a tierra se hicieron a través de planos a tierra, las pistas que se colocaron fueron de 6 mil que es el valor por defecto del software y se tuvo en cuenta que el grosor de las de alimentación principales fueran el doble que las anteriores (12 mil). Este valor se estimó tomando como supuesto que estas pistas deben soportar la corriente de todo el circuito.

Luego de tener el diseño, se envió al fabricante chino PCBWAY, el cual, al cabo de 5 días, entregó las 10 placas para ser enviadas a Uruguay. A las mismas se les soldaron los componentes básicos, LEDS y uno de los sensores, obteniendo finalmente un prototipo que se muestra en la figura 4.9.



Fig. 4.9 – Placa de prototipo compuesta por un microcontrolador, un sensor de presión y el módulo de comunicación LoRa

## 4.2.2 Placa final

Al momento de diseñar un segundo dispositivo se analizaron los componentes a utilizar. En primer lugar, se buscaron posibles competencias del microcontrolador ya que es uno de los componentes más importantes y caros del circuito, y los periféricos dependen de él. Por tal motivo se buscaron unidades que cumplan con las siguientes

características: conectividad SPI, UART/USART, bajo costo, manejo de modo sleep con bajo consumo y varias entradas salidas digitales y analógicas.

En la tabla 4.2 se muestran 3 posibles microcontroladores y se los compara con el Arduino Pro Mini utilizado.

Micro controlador	Velocidad	Alimentación	Tamaño del núcleo	Memoria	Costo	Consumo
MCR908JL3ECDWER-ND	8MHz	2,2 – 3,3 V	8 bits	4 Kb	\$ 2,8	N/E
PIC18F25K40	64MHz	1,8 – 3,6 V	8 bits	32 Kb	\$1,33	900 nA
Atmega328p	20MHz	1,8 – 5,5 V	8 bits	32 Kb	\$1,28	0,75 uA
Arduino Pro Mini con Atmega328p	20MHz	1,8 – 5,5 V	8 bits	32 Kb	\$1,83	3,28 mA

Tabla 4.2 – Cuadro comparativo de diferentes posibles microcontroladores

Luego del estudio de varios componentes y la buena performance del prototipo, realizado en el apartado anterior, se decidió mantener la mayoría de los componentes utilizados, incluyendo el módulo micro controlador Arduino. Por otro lado, la diferencia de costos no era tan significativa como para ameritar realizar los cambios, sumado a que Arduino permite una conexión de periféricos más práctica. A pesar que brindaba menor consumo y mayor velocidad de operación, no se optó por utilizar el PIC debido a que podía generar problemas de compatibilidad con el resto del sistema y no se contaba con el tiempo necesario para afrontar ese problema.

Para esta nueva placa se realizaron varias modificaciones para permitir escalabilidad y mejorar aún más el comportamiento del equipo. Una de las principales fue la división del *endpoint* en módulos. Para este caso se diseñaron 2 módulos los cuales se conectan de forma superpuesta. Mientras que en uno se encuentran los componentes que realizan el procesamiento de la señal (llamado módulo de procesamiento), el otro es el que se encarga de interconectar los periféricos al anterior (módulo de censado). Esta nueva disposición de los componentes permite una mayor escalabilidad del *endpoint* ya sea para modificar un periférico dañado o para conectar otro sensor en el futuro, que en la primera instancia no se tuvo en cuenta.

### 4.3.2.1 Modulo de procesamiento

El módulo de procesamiento cuenta con el microcontrolador, el módulo de comunicación LoRa RFM95W, un conector SMA hembra interior para la antena y conexiones varias (alimentación, interconexiones, leds y resistencias). El conector SMA permite adaptarle una antena de mayor ganancia que la utilizada para el prototipo. Para el dimensionado de las pistas se mantuvo el mismo criterio que para el prototipo 1 pero la pista de la antena en este caso es de 20 mil de ancho y del menor largo posible. Estos valores se tomaron de forma arbitraria con la intención de que la señal tenga buena

calidad. En las figuras 4.10 y 4.11 se muestran los esquemáticos del circuito para esto módulo mientras que en la figura 4.12 se ve el módulo armado.

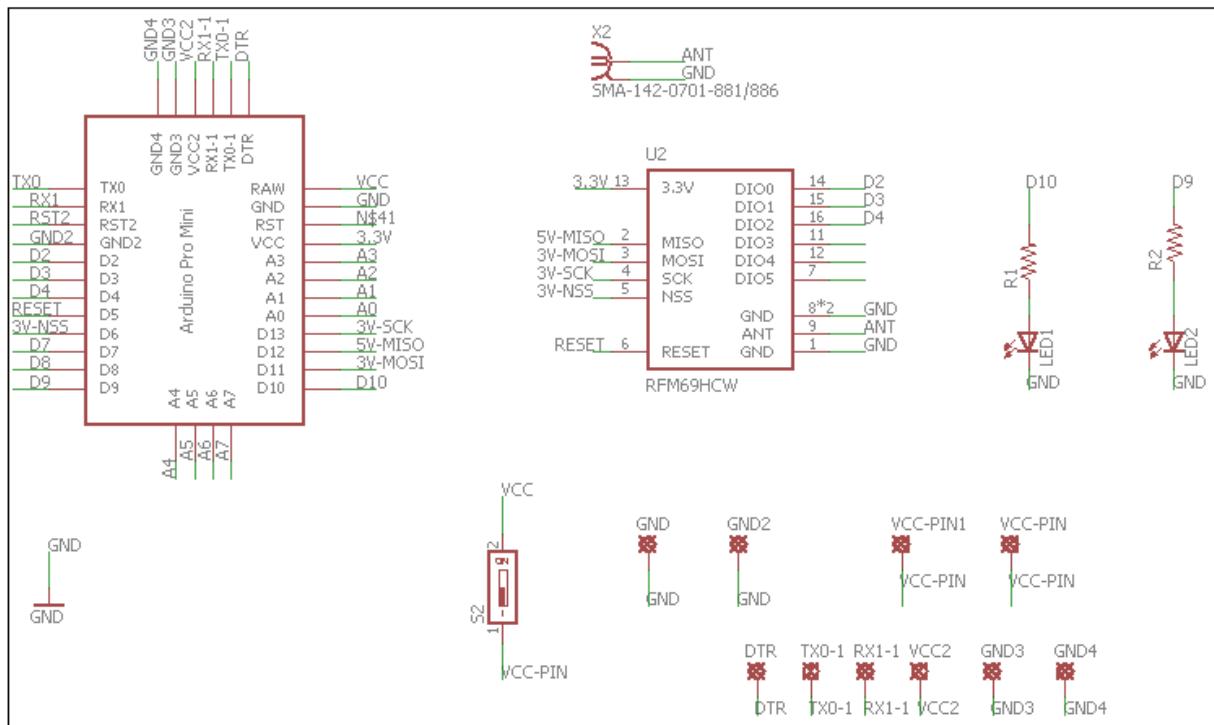


Fig. 4.10– Esquemático del módulo de procesamiento de la placa final realizado en Eagle

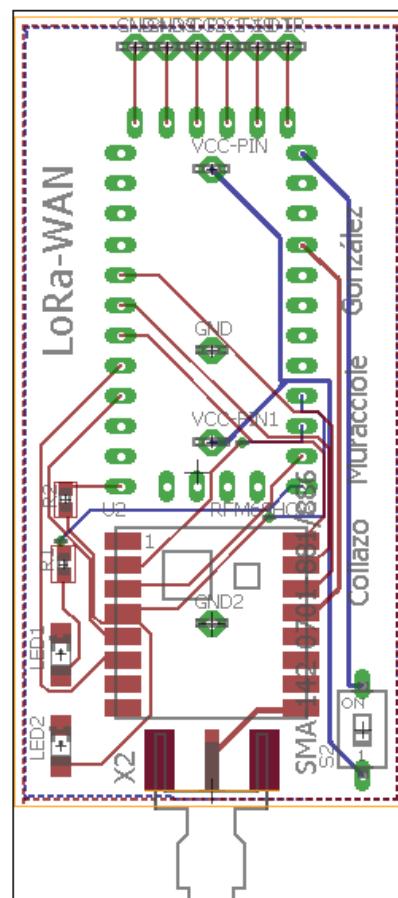


Fig. 4.11 – Disposición de los componentes del módulo de procesamiento de placa final realizado en Eagle



Fig. 4.12 – Módulo de procesamiento

#### 4.3.2.2 Módulo de censado

Este módulo concentra todos los sensores utilizados en el prototipo en una placa de tamaño muy reducido, la cual se conecta con el módulo de procesamiento. Permite la conexión de un solo sensor a la vez, ya que se diseñó de forma tal que se comparten pistas para utilizar la menor cantidad de pines en el módulo de procesamiento. Se hizo de esta manera para dejar pines libres para futuras aplicaciones.

La alimentación se brinda a través de la conexión con el módulo de procesamiento. En las figuras 4.13 y 4.14 se pueden ver los esquemáticos del circuito para implementar este módulo. En la figura 4.15 se muestra la placa impresa sin sensores conectados.

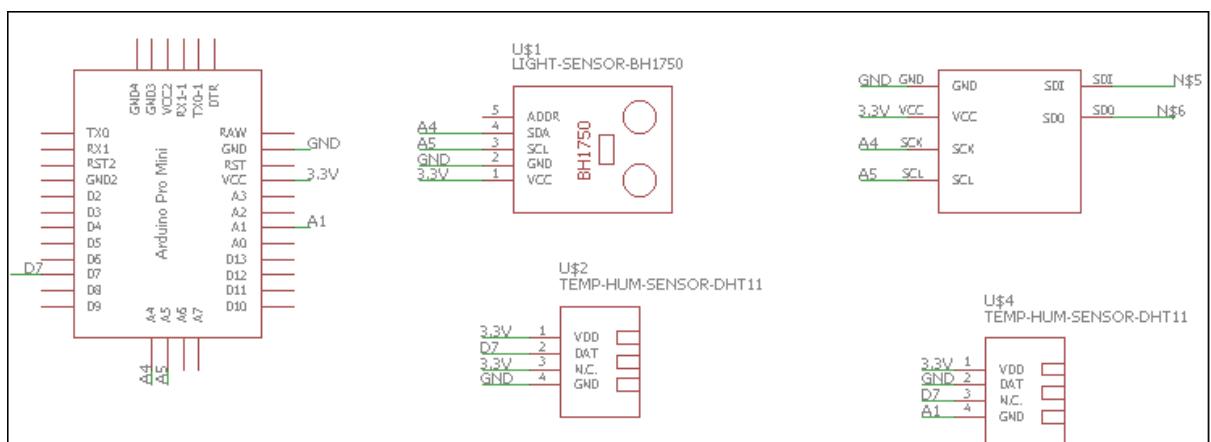


Fig. 4.13 – Esquemático del módulo de censado de la placa final realizado en Eagle

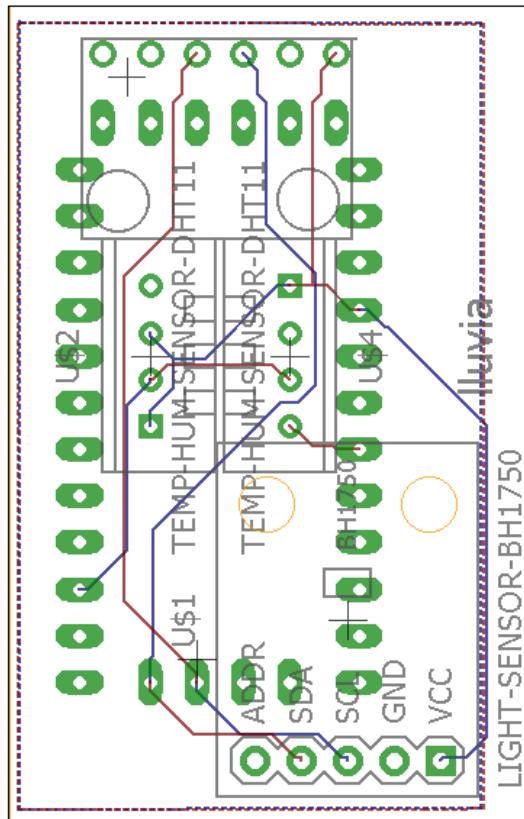


Fig. 4.14 – Disposición de los componentes del módulo de censado de placa final realizado en Eagle

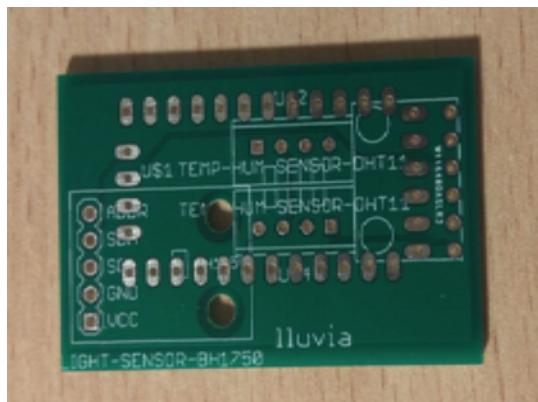


Fig. 4.15 – Módulo de censado

En cuanto a la alimentación del *endpoint*, éste fue diseñado para poder ser utilizado con 2 pilas de AA de 3V. Esto se debe a que es un modelo de batería que se encuentra muy fácilmente en el mercado y la gran mayoría (alcalinas, zinc-carbón, NiMh) soportan los picos de corriente que se necesitan realizar en envío de información. Los dispositivos fueron pensados para ser utilizados en aplicaciones que pueden carecer de conexión a energía eléctrica por cable, por lo cual es importante que puedan alimentarse por alguna fuente externa como baterías fácilmente intercambiables y de bajo costo.

Finalmente, en la figura 4.16 se puede ver cómo es el nodo final compuesto por el módulo de procesamiento y el módulo de censado. En lo que respecta a las baterías, estas no aparecen en la foto ya que por un problema en el diseño de la placa quedaron del mismo lado que el módulo de censado. De todas formas, mediante un FTDI se puede alimentar el micro con la ayuda de un cable USB



Fig. 4.16 – Endpoint final

# Capítulo 5

## Gateway

### 5.1 Introducción

Aquí se explica qué es un *gateway* y qué función cumple en una red LoRaWAN.

Si se hace la traducción, *gateway* es “puerta de enlace” o “ruta de salida”. En una red IP tradicional, un *gateway* es utilizado para unir distintas redes o, por ejemplo, unir una red privada con internet. Aquí la función es solo separar un dominio de red de otro de modo de proteger y limitar la red interna.

Otra función que cumplen este tipo de equipos es la de hacer de traductor entre dos partes que se comunican con diferentes protocolos. Para poner un ejemplo conocido en el ambiente de las telecomunicaciones, se puede mencionar los *gateway* utilizados en telefonía, en donde se tiene que de un lado se comunica utilizando protocolo SIP y la salida se conecta a la red PSTN (Red Telefónica Conmutada). No se va a entrar en detalles de cómo funcionan estos dispositivos en las redes mencionadas, simplemente se mencionan estos ejemplos para mostrar el concepto de *gateway*. Para el caso de LoRaWAN el concepto es el mismo.

Entrando en el funcionamiento que nos interesa, en las redes LoRaWAN un *gateway* es un dispositivo que forma parte de la arquitectura de red. Se encarga de recibir los datos transmitidos por un *endpoint* y los reenvía, según sea su configuración, a un servidor centralizado. En el común de los casos, dichos equipos reciben los paquetes de los nodos utilizando el protocolo LoRa, los traducen y los envían al servidor central por medio de una red IP tradicional. Este servidor puede estar en una red ethernet con el *gateway* o puede estar en un lugar remoto y la conexión se hace por medio de internet.

Al ser LoRa un protocolo RF, varios *gateway* pueden recibir un paquete de un mismo nodo. Varias puertas de enlace se conectan al servidor de red central por medio de conexiones IP estándar, formando así una red en estrella. Las comunicaciones entre los dispositivos finales y el servidor central son generalmente unidireccionales o bidireccionales, pero el estándar también soporta multidifusión. Esto es, que se puede enviar un mensaje desde el servidor central a varios nodos finales a la vez solo si es de clase B o C como se mencionó anteriormente en este documento. En la figura Fig. 3.17 se puede ver como los *gateways* se encuentran entre medio del servidor central y los nodos finales.

## 5.2 Nuestro Gateway

En esta sección se muestra el hardware utilizado para implementar el *gateway* de red LoRaWAN. Como se explicó en la sección 5.1 el *gateway* es el encargado de comunicar la red LoRaWAN con la red IP. En nuestro caso se utilizó el equipo RAK831 de la marca RAK.

Durante la investigación previa a la compra se encontró, buscando en foros como el de TTN o mismo en la página del fabricante (<http://www.rakwireless.com/en>), que este módulo está entre los dispositivos avalados por LoRa Alliance y es compatible con las implementaciones que se pueden realizar utilizando el servidor de TTN.

En la siguiente lista se mencionan otros equipos que se investigaron:

- RAK 831 [xxx]
- The Things Gateway [xxxi]
- Wirnet Station Gateway – Kerlink [xxxii]
- Cisco Wireless Gateway [xxxiii]
- MultiConnect® Conduit [xxxiv]

Comparado con otras marcas, de Gateway de LoRaWAN, se puede ver en sus hojas de especificaciones las diferencias técnicas que hay entre los diferentes productos. En su mayoría son equipos con más recursos que el elegido, pero para fines de este proyecto lo central era lograr establecer la red sin grandes requerimientos de flujo de datos y carga de los equipos. Una de las principales diferencias es que RAK 831 no viene en modo ‘All in one’, es decir, el equipo necesita de otro equipo externo para su funcionamiento. El resto de los Gateway comparados tienen ya integrado un firmware, o sistema operativo, el cual permite que se configure directamente sobre el equipo. Este último punto hace la diferencia sobre el precio del equipo y es uno de los factores principales por el cual se optó por RAK 831. El gateway elegido está por debajo de los 150 dólares mientras que los gateway comparados están bastante por encima de esta cifra. No se indica referencia sobre los precios ya que estos varían constantemente según la disponibilidad y las actualizaciones de equipos. Los precios buscados fueron realizados en los sitios web Digi-Key, AliExpress, iot-shops y thethingsnetwork.

En la figura 5.1 se muestra el “Developer KIT” comprado inicialmente para implementar el *gateway*. De derecha a izquierda el KiT se compone por el módulo RAK831, módulo FTDI2232<sup>15</sup>, cables para la conexión de los módulos, antena y cable USB para conexión del FTDI2232 a una computadora.



Fig. 5.1 - Developer KIT

El RAK831 es un dispositivo de RF que se encarga de la comunicación con los nodos finales. Tiene la capacidad de recibir en diferentes canales de frecuencia al mismo tiempo (8 en simultáneo) y de demodular/modular la señal LoRa enviada hacia y desde los nodos. Las características principales del módulo que describe el fabricante son las mostradas en la tabla 5.1

---

<sup>15</sup>FTDI2232: módulo USB-Serial para conexión de RAK831 a una computadora.

Parameter	Description
<b>Operating Voltage</b>	DC 5V
<b>Operating Temperature</b>	-40C to + 85C
<b>Radio Chipset</b>	SX1301
<b>RF Output Power</b>	23dBm
<b>Input Signal</b>	Up to -142,5dBm(@293bps)
<b>Receiver Sensitivity</b>	Up to -142,5dBm(@293bps)
<b>GPS Receiver</b>	Optional
<b>SNR handling</b>	9dB
<b>Control Interface</b>	SPI
<b>Dimension</b>	80mm*50mm*5mm
<b>Firmware</b>	Gateway HAL
<b>Range</b>	Up to 15 Km
<b>Frequency Band</b>	433/868/915 MHz

Tabla 5.1 – Parámetros del módulo RAK831

Se puede ver en la tabla 5.1 que el módulo soporta varias frecuencias. En este caso se utiliza 915 MHz. El módulo está compuesto por el chip SX1301 el cual es el encargado de hacer la modulación y demodulación de la señal LoRa enviada desde y hacia los nodos para luego comunicarla por conexión serial. El chip SX1301 es de la empresa Semtech, creadora del protocolo LoRa. Otro aspecto para destacar, de la tabla 5.1 es que asegura un alcance de hasta 15 kilómetros. Este parámetro no se especifica en qué condiciones se obtiene ni con qué antena o frecuencia, pero se verá luego en la sección de pruebas el alcance real obtenido.

Como se puede ver, este viene con el módulo FTDI2232 el cual oficia de conversor serial-USB. Este es necesario ya que la placa RAK831 por sí sola no cuenta con una interfaz física para conectarla a la PC y realizar las configuraciones correspondientes.

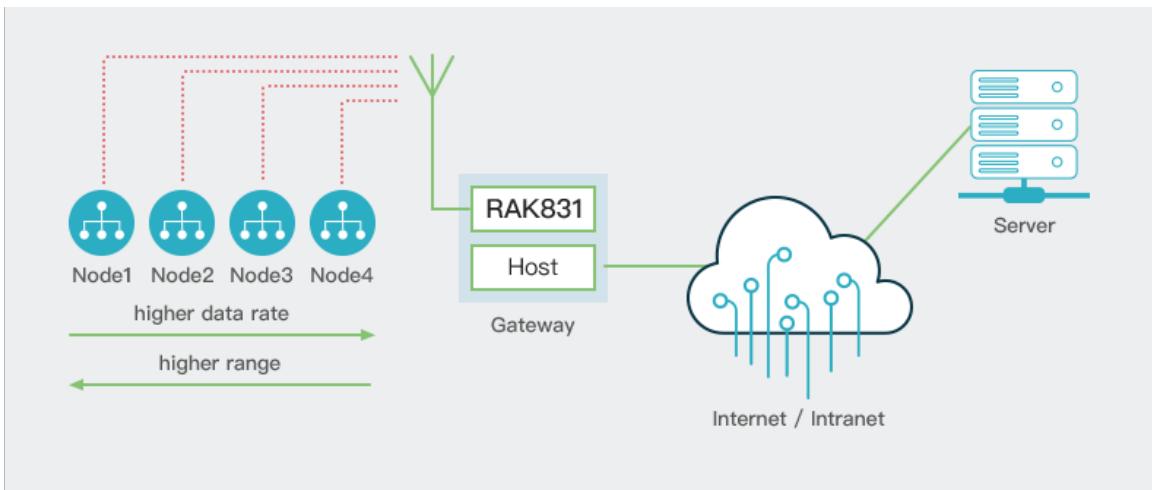


Fig. 5.2 - Gateway compuesto por módulo RAK831 + Host [xxxv]

En la figura 5.2se puede notar como es el diagrama de la red (del cual se ha hablado en secciones anteriores), donde ahora el *gateway* está formado por el módulo RAK831 y un host. En primera instancia para la implementación de este último se utilizó una computadora en la cual se instaló el sistema operativo Ubuntu. Siguiendo la guía del fabricante para la configuración del *gateway* y mediante la conexión USB-Serial utilizando el módulo FTDI2232, se lograron las primeras comunicaciones entre un nodo y el concentrador.

Esta solución fue útil para realizar las primeras pruebas, en donde se logró ver el payload enviado por el nodo en la consola de la computadora, pero no era la mejor solución para tener el *gateway* en distintos puntos ya que había que tenerlo conectado a una computadora, así como tampoco era práctica la configuración para hacer la conexión hacia el servidor TTN. Por tales motivos es que se optó por implementar otra solución.

Luego de investigar, en la página del fabricante del *gateway* y en el sitio oficial de TTN, se encontró que era posible utilizar un Raspberry PI que oficie de host. Este módulo es una minicomputadora en el cual se le puede instalar un sistema operativo y utilizar sus pines para hacer la conexión serial con el RAK831. Además, cuenta con un puerto ethernet y conexión WiFi lo cual fue utilizado para hacer la conexión con el servidor de TTN. Con esto se elimina la necesidad del módulo FTDI2232 y de una computadora convencional.

## 5.3 Configuraciones

En esta sección en primer lugar se mencionarán las conexiones físicas necesarias para implementar el *gateway*. Luego se explicarán las configuraciones realizadas en el Raspberry tanto para setear el RAK831 como la conexión con TTN. También se mostrará la configuración en el servidor.

Las configuraciones realizadas fueron en base a la documentación de los fabricantes RAK y Raspberry [xxxvi], en donde se explican los diferentes pasos para

armar el equipo y su configuración. Aquí se mostrarán las configuraciones que se hicieron siguiendo los manuales del fabricante.

Como se habló en la sección 5.2 el *gateway* está implementado por el módulo RAK831 y el host Raspberry PI 3. La conexión de los dispositivos se hace por conexión serial utilizando los pines que tiene cada uno. En la tabla 5.2 se muestra el pinout para la conexión mientras que en la figura 5.3 se puede apreciar una imagen del *gateway* conectado al RPI3.

RAK 831 Pin	Description on silk screen	RPI physical pin
1	+5V	2
3	GND	6
19	RST	22
18	SCK (SPI Clock)	23
17	MISO	21
16	MOSI	19
15	CSN (Chip Select)	24

Tabla 5.2 - Conexión entre RAK831 y Raspberry [xxxvi]



Fig. 5.3 - Conexión del Gateway, Raspberry PI a la izquierda y RAK831 a la derecha

Para hacer la configuración del *gateway* primero fue necesario instalar el sistema operativo Raspbian Lite en el Raspberry PI. Este es un sistema gratuito disponible en la página del fabricante (<https://www.raspberrypi.org>).

Para poner el sistema operativo en la placa, se utilizó el programa Etcher. Lo que se hizo con este es copiar el sistema operativo en la SD del equipo, por lo cual una vez que este se encienda, levantará la imagen de dicho lugar. Teniendo el SO pronto, el Raspberry se comporta como una computadora (se le puede conectar mouse, teclado, monitor, etc.). Esta configuración fue realizada siguiendo las guías y manuales de libre acceso disponibles en la página de Raspberry [xxxvii].

Teniendo el Raspberry pronto se procedió a hacer la configuración del RAK831 y la integración con TTN. En primer lugar, fue necesario habilitar la conexión SPI (interfaz de conexión serial) del Raspberry para poder utilizar las conexiones mostradas en las figuras 5.4 y 5.5.

Para levantar la configuración del RAK831 se utilizó una librería obtenida de GitHub. Esto es posible ya que el Raspberry está conectado a internet (por WiFi o su puerto ethernet) y como Raspbian es un sistema operativo Linux, es posible hacer manejos tales como descargar un repositorio directamente desde un servidor web. Para nuestra implementación se utilizó el repositorio <https://github.com/ch2i/LoraGW-Setup.git> el cual se descarga e instala directo en la placa. Con este instalado se procedió a programar la configuración inicial del RAK831 y la conexión con el servidor TTN. Para ello, primero se creó el *gateway* en el servidor de red de TTN ya que al momento de configurar el *gateway* en el raspberry es necesario completar parámetros para coincidir con lo creado en el servidor.

Similar a la creación de nodos y aplicaciones en el servidor de TTN, explicado en el capítulo 3, se crea el *gateway* en el servidor, solo que en este caso se realiza a nivel de red y no de aplicaciones. A continuación, se detallan los diferentes parámetros que se deben configurar y que son mostrados en la figura 5.4.

- **Gateway ID:** Identificador único de cada *gateway* configurado en el servidor. Este parámetro es el que sirve para identificar en la cuenta de que usuario se deben mostrar los paquetes recibidos por este *gateway*.
- **Owner:** Identifica la cuenta dueña del *gateway*
- **Status:** Muestra si el equipo está conectado al servidor.
- **Frecuency Plan:** Aquí se elige la frecuencia a utilizar. Cuando el *gateway* se conecta a la red, busca en el servidor la frecuencia seteada y configura según este parámetro. Quien hace esta tarea es el Raspberry PI.
- **Router:** TTN posee varios routers disponibles en internet a lo largo del mundo. Este router es utilizado por el Raspberry para enviar los paquetes al servidor TTN.
- **Gateway Key:** Para ver los paquetes que un *gateway* envía al servidor no solo es necesario ser el dueño del *gateway* ID, sino que también es necesario que coincida la contraseña. Esta es configurada en el propio *gateway*.



Fig. 5.4 - Configuración del gateway en servidor TTN

Luego de configurado el servidor TTN, se puede levantar la configuración en el *gateway*. En la figura 5.5 se muestra los parámetros configurados para este. Dicha configuración es realizada en el Raspberry para la correcta conexión con el servidor, por lo cual el RAK831 no conoce cómo es esta, sino que solo recibe la información de la frecuencia de sus canales. Esto se debe a que, en el servidor, el *frequency plan* se configuró como 915MHz, por lo tanto, el Raspberry le pasará al RAK831 los parámetros para que funcione en esas frecuencias. En la configuración inicial del Raspberry, cuando se hace la descarga de repositorios desde GitHub, se obtienen todos los planes de frecuencias soportados por TTN. Según sea configurado el campo de *frequency plan* es el archivo de configuración que el RPI3 utilizará para configurar el RAK831.

```
{
  "gateway_conf": {
    "gateway_ID": "B827EBFFFFEE9EEB3",
    "servers": [
      {
        "server_address": "router.brazil.thethings.network",
        "serv_port_up": 1700,
        "serv_port_down": 1700,
        "serv_enabled": true
      }
    ],
    "ref_latitude": -34.89099390,
    "ref_longitude": -56.13689846,
    "ref_altitude": 34,
    "contact_email": "tesislorawan@gmail.com",
    "description": "Gateway Tesis"
  }
}
```

Fig. 5.5 - Conexión del gateway, Raspberry PI y RAK831

En las figuras 5.6 y 5.7 se observa que el *gateway ID* debe ser el mismo. El “eui-” que se ve en la configuración del servidor es un prefijo que tienen todos los *gateway ID*. Por otro lado, el *gateway key* se configura al momento de correr el configurador en el Raspberry. Con esta configuración ya queda funcionando la integración del *gateway* con el servidor TTN y los paquetes que reciba el *gateway* los enviará al servidor. Cuando a este llega un paquete, identifica si fue por un nodo configurado por la misma cuenta en la sección de aplicaciones. Es aquí cuando muestra el mensaje en la consola de tráfico y realiza las acciones según esté configurado como se explicó en el capítulo 3.4.

Además de la configuración básica se ve en la figura 5.5 que se agregan parámetros de ubicación geográfica. Esto puede ser simplemente para información o también este dato es utilizado por la integración de TTNMapper para detectar donde se encuentra el equipo y a partir de él trazar los puntos en el mapa con su respectivo nivel de cobertura.

# Capítulo 6

## Pruebas y Estudio de Costos

### 6.1 Pruebas de alcance

Se realizaron distintas pruebas para calcular el alcance que se puede obtener con LoRaWAN para así poder dimensionar una red con mayor precisión.

A su vez, mediante la integración de TTNMapper<sup>16</sup> fue posible trazar un mapa de cobertura y computar así el nivel de señal a partir de la posición.

#### 6.1.1 CON línea de vista

##### 6.1.1.1 Prueba con gateway en movimiento

En esta instancia se realizaron una serie de pruebas donde el elemento que se movía era el *gateway* mientras el nodo se mantenía quieto. Estas fueron llevadas a cabo en un tramo recto de 3 kilómetros de la ruta 5 y se pudo constatar que se recibían paquetes a dicha distancia. Sin embargo, no fue posible continuar con las mediciones ya que luego se entraba en un valle y se perdía la línea de vista. Cabe destacar que la diferencia de altura entre el *gateway* y el nodo era de aproximadamente un metro.

El *endpoint* utilizado fue uno de los de prototipo con una antena helicoidal soldada a la placa (ver figura 4.9) y enviando paquetes cada 10 segundos. Por otro lado, el *gateway* fue instalado dentro de un auto con la antena en el techo como se puede apreciar en figura 6.1 y 6.2

---

<sup>16</sup><https://www.ttnmapper.org/>



Figs. 6.1 y 6.2 – Instalación de antena para pruebas con gateway en movimiento

Se realizaron medidas con 2 antenas diferentes (figuras 6.3 y 6.4) y en ambos casos de obtuvieron los mismos resultados, por lo cual, si bien sirve para corroborar el alcance a kilómetros de distancia, no es posible estimar cual es la máxima distancia que se puede lograr. Por este motivo se procedió a buscar un lugar con mayor línea de vista y de esta forma realizar la prueba del punto 6.1.1.2.



Figs. 6.3 y 6.4 – Antenas omnidireccionales Cisco 3G-ANTM-OUT-OMy Cisco 4G-LTE-ANTM-D respectivamente

### 6.1.1.2 Pruebas con nodo en movimiento

En esta instancia se decidió invertir el procedimiento anterior y dejar fijo el *gateway* mientras el que variaba de posición era el *endpoint*. El nodo utilizado fue el mismo que para el punto 6.1.1.1 y configurado de igual forma. Para la parte del

concentrador, solo se utilizó la antena de la figura 6.4 ya que, a partir de la prueba anterior y sus características similares, se vio que no aportaba demasiado realizarlo con ambas antenas

En primer lugar, se instaló la estación base a 10 metros de altura, mientras que el nodo se lo colocó a nivel del suelo dentro de un auto y se salió a recorrer en busca de sitios con línea de vista hacia el *gateway*. Con la ayuda de un celular, al cual se le había instalado la aplicación TTNMapper fue posible obtener un mapa de cobertura (usando el GPS del teléfono) donde se puede apreciar el nivel de señal en función de la posición del nodo.

Hay que recordar que el dispositivo estaba configurado como de clase A lo cual no es lo ideal si este se encuentra en movimiento. De todas formas, se pudieron recabar las mediciones necesarias.



Fig. 6.5 – Relevamiento realizado con TTNMapper



Fig. 6.6 – Nivel de señal en función de la posición del nodo

Viendo la figura 6.6, los puntos que surgieron de este experimento son los que se encuentran sobre la costa. Esta imagen, al igual que la de la figura 6.5, se obtuvo del servidor de TTNMapper el cual es público; por lo tanto, cualquier persona que realice una prueba aparecerán los datos aquí. Esta es el motivo por el cual se pueden apreciar

las mediciones en verde y rojo en la zona de Pocitos, lo cual refiere a una prueba realizada por otro usuario. Es claro ver que cuando el nodo se encontraba dentro del rango de cobertura, el *gateway* comenzaba a recibir tráfico. El criterio de colores es el que se especifica en la figura 6.7.

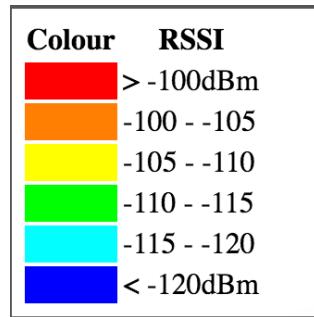


Fig. 6.7 – Criterio de colores de TTNMapper para nivel de recepción

Con esta prueba fue posible recibir un paquete a 8 kilómetros de distancia entre el emisor y el receptor. La ubicación de nodo y Gateway se muestra en la figura 6.8



Fig. 6.8 – Posición de nodo y gateway registrando comunicación entre sí

A dicha distancia se recabó la información del plano de control del paquete (ver Tabla 6.1): del cual se desprenden algunas conclusiones. En primer lugar, el SF de 10 indica que el ADR está bien configurado ya que, a grandes distancias, dicho valor debe ser 10. Por otro lado, cabe destacar que viendo el *rssi* y el *snr* ambos valores se encuentran muy cerca del límite máximo soportado por el módulo para una buena comunicación, por lo cual muestra que no será posible ir mucho más allá de 8 kilómetros. De todas formas, el nivel de señal es óptimo para que el RFM95W pueda interpretar el paquete a dicha distancia sin problemas.

Info	Data
Frecuencia	905,1
SF	10
BW	125
Coding Rate	4/5
Channel	6
rssi	-123
nsr	-15

Tabla. 6.1 – Información de paquete recibido a 8km. del gateway

## 6.1.2 SIN línea de vista

En esta instancia se buscó ver el comportamiento que tenía la red bajo condiciones menos favorables a las del punto 6.1.1 ya que ahora la comunicación entre el nodo y el gateway no era con línea de vista, sino que la prueba se realizó dentro de la ciudad. En primer lugar, se colocó la antena de la figura 6.9 a 30 metros del nivel del suelo, mientras que el nodo de la figura 4.16 se dejó en el techo de un auto en movimiento. Con esto se salió a recorrer para determinar la máxima distancia a la cual el *endpoint* pudiera comunicarse con el *gateway*.



Fig. 6.9 – Antena omnidireccional marca RAK de 6dBi

Una vez realizada esta prueba, la máxima distancia a la cual se registró un tráfico de datos fue a 4 kilómetros como se muestra en la figura 6.10. Esta distancia es la máxima alcanzada, no es que hasta ese punto hubo comunicación continua. En algunos tramos los paquetes eran recibidos por el gateway, pero otros no se alcanzaba a recibir nada.

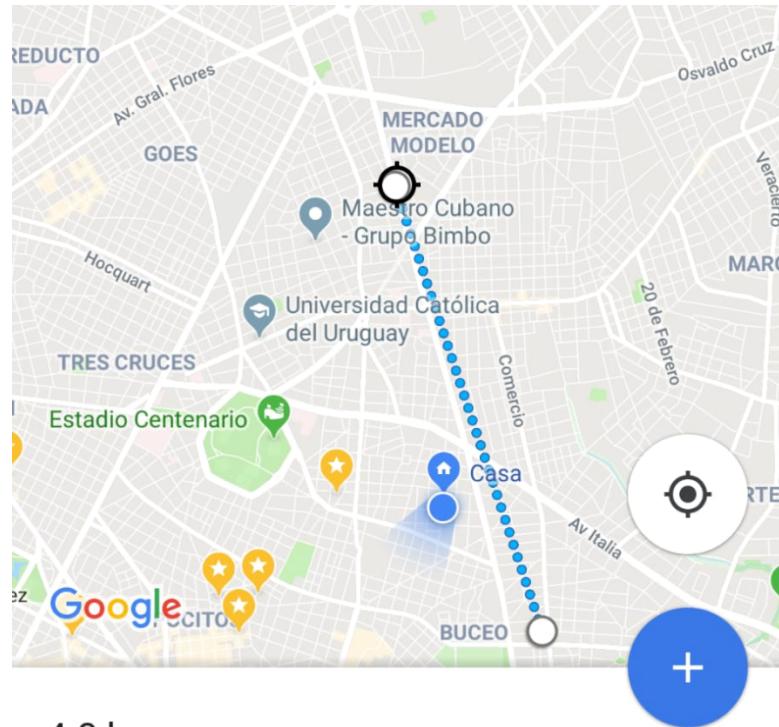


Fig. 6.10 – Distancia máxima alcanzada sin línea de vista

Este ensayo parecía correcto no solo porque el valor de distancia obtenido es menor que en el experimento del punto 6.1.1, lo cual es coherente; sino que también concuerda con la distancia teórica de recepción que se maneja para nodos que se encuentran dentro de la ciudad, la cual se estima entre 2 y 6 kilómetros dependiendo del terreno.

A partir de dicha prueba se pudo también corroborar el comportamiento que presentan los nodos de clase A (clase del nodo utilizado) ya que cuando el auto se encontraba en movimiento, los paquetes no llegaban al gateway, pero una vez que se frenaba, estos comenzaban a llegar.

## 6.2 Pruebas limitante de paquetes

Esta prueba se realizó para constatar la restricción teórica de TTN planteada a la hora de la elección del servidor en cuanto a la máxima cantidad de paquetes que se pueden enviar o recibir por nodo por día.

Como se mencionó en la sección 3.4 de este documento, el servidor limita la cantidad de paquetes de *uplink* por nodo por día en 30 segundos de time on air y en 10

mensajes de *downlink*. Para comprobar esto, se dejó funcionando el *endpoint* de “Número Aleatorio”, de forma que cada 10 segundos generaba un paquete y lo enviaba al *gateway*. Los mensajes tenían un largo de payload de 14 bytes y el tiempo en el aire era de unos 46,3 milisegundos. Se obtiene que luego de 648 mensajes el servidor TTN comenzaría a descartar dichos paquetes ya que a partir de allí se supera el umbral permitido<sup>17</sup>. Sin embargo, el resultado no fue el esperado ya que pasadas las 15 horas de pruebas ininterrumpidas y más de 5600 mensajes de *uplink* (equivalente a 260 segundos de time on air aproximadamente), el servidor continuaba recibiendo mensajes e interpretándolos de forma correcta como se puede ver en las figuras 6.11 y 6.12

time	frequency	mod.	CR	data rate	airtime (ms)	cnt
▲ 18:59:16	904.7	lora	4/5	SF 7 BW 125	46.3	5636 dev addr: 26 03 15 A9 payload size: 14 bytes

Fig. 6.11 – Paquete 5636 del nodo “Número Aleatorio” en el servidor de red

time	counter	port	dev id:	payload:	Dato_Recibido:
▲ 18:59:16	5636	1	<a href="#">sensoresucualeatorio01</a>	30	48

Fig. 6.12 – Paquete 5636 del nodo “Número Aleatorio” en el servidor de aplicaciones

Por lo tanto, con estas pruebas se demostró que la limitante teórica de los 30 segundos de *uplink* por nodo por día no se cumplía. Esto refuerza la idea de utilizar TTN como nuestro servidor. Por otra parte, se realizaron más de 10 envíos de paquetes de *uplink* del *gateway* al nodo y pasado este número, los mensajes seguían llegando al *endpoint*, por lo cual tampoco se registró un corte en este sentido.

## 6.3 Prueba de stress

En este caso se buscó probar el comportamiento del *gateway* ante una constante ráfaga de paquetes entrantes por parte de los nodos y ver así el grado de pérdida de los mismos. Para ello se dejó funcionando durante 24 horas, 4 *endpoints* (ver figura 6.13) en simultáneo a 3 metros de distancia del *gateway*. Cada uno de estos estaba programado para enviar un dato a una frecuencia de 10 segundos.

---

<sup>17</sup> A 10 segundos por paquete, implica que limitaría luego de 1 hora y 48 minutos.

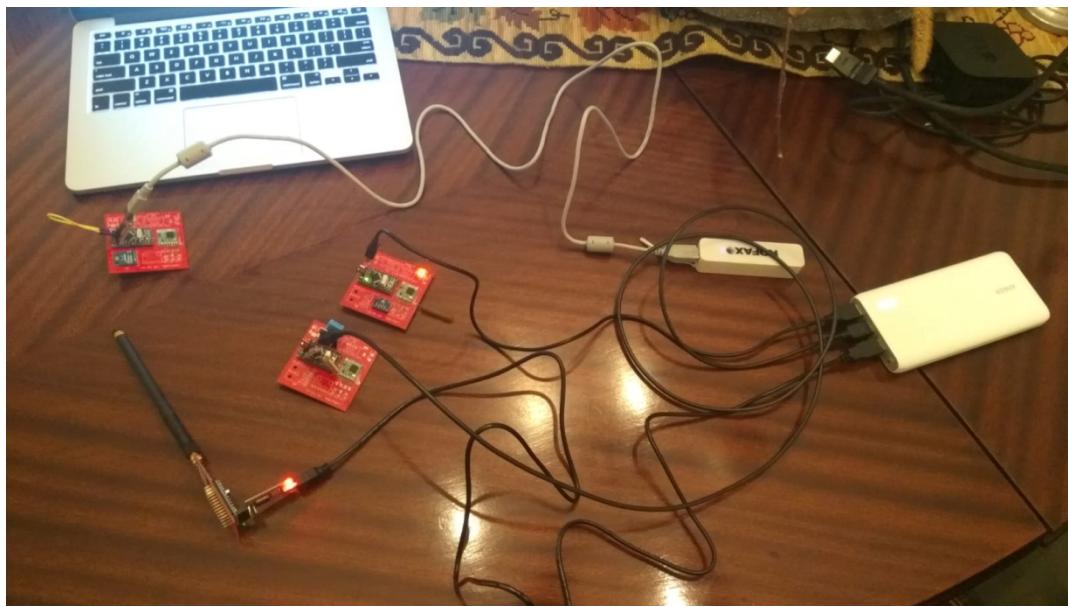


Fig. 6.13– Nodos programados y funcionando para la prueba de stress

De esta forma, una vez que se cumplió el tiempo estipulado, se procedió a calcular la cantidad de paquetes enviados y recibidos en ambos sentidos obteniéndose los resultados de la tabla 6.2

PAQUETES ENVIADOS POR LOS NODOS	26675
PAQUETES RECIBIDOS POR EL GATEWAY	26672
PAQUETES RECIBIDOS POR LOS NODOS	2783
PAQUETES ENVIADOS POR EL GATEWAY	2783

Tabla 6.2 – Cuadro asociado a la disponibilidad de la red

De la tabla 6.1 se desprende que, para el *downlink*, la disponibilidad fue de **99.99%**, mientras que la del *uplink* fue de **100%**.

Estos datos concuerdan con lo esperado ya que si un *Gateway* está diseñado para poder trabajar sin dificultades cuando 1000 nodos se comunican al servidor por medio de este, no habría mayores problemas si solamente 4 están traficando bajo condiciones casi ideales. Por lo cual, si bien es bueno tener en cuenta esa medida, no es representativa si se desea extraer a cientos de nodos enviando información en simultáneo distribuidos a diferentes distancias. Sin embargo, se debe tener en consideración que en la práctica no es usual que los *endpoints* envíen datos cada 10 segundos, sino que por lo general lo suelen hacer en intervalos de tiempo más amplios (minutos u horas).

## 6.4 Prueba de latencia

Para esta medición se dejó encendido un nodo durante 24 horas enviando un paquete cada 10 segundos. Haciendo esto, una vez que se cumpliera el tiempo estipulado se podría comparar la cantidad de paquetes enviados con los que tendría que haber enviado (dato obtenido mediante cálculos teóricos explicados a continuación) y así calcular la latencia promedio. A un día lo conforman 86400 segundos; por lo cual, si el nodo transmite un dato cada 10, al cabo de un día este deberá haber mandado 8640 paquetes. Sin embargo, luego de dejar activo el nodo por dicho tiempo se registraron 7310 paquetes transmitidos, lo cual equivale a haber mandado un paquete cada 11,8 segundos. Es decir que la latencia promedio obtenida por paquete de **1,8 segundos** o, dicho de otro modo, un **18%**.

## 6.5 Pruebas de consumo

En la siguiente sección se midió el consumo del *endpoint* en diferentes situaciones para luego compararlas con los valores teóricos. Con esta prueba se pretende estudiar el comportamiento de la batería y así estimar el tiempo promedio que durará antes de necesitar ser reemplazada.

### 6.5.1 Estudio Teórico

Para obtener un estimativo teórico del consumo de un *endpoint* se analizó el comportamiento de cada componente por separado y bajo diferentes situaciones, ya que el consumo varía bastante entre que envía un dato o se encuentra recibiendo o en modo sleep. Estos valores tomados de los datasheet se reflejan en la tabla 6.3.

Componente	Modo	Consumo (mA)
<b>Microcontrolador (atmega328p)</b>	Sleep	0.005
	Normal	3.3
<b>Oscilador</b>	Normal	0.006
<b>LDO</b>	Normal	0.08
<b>Modulo LoRa</b>	Transmisión	120
	Recepción	10.3
	Sleep	0.0002
<b>LED</b>		0,000330
<b>Total:</b> 123.000330 mA		
<b>Microcontrolador + Oscilador + LDO + Módulo LoRa + LED</b>	Transmisión	123
	Sleep	(Sin definir)
<b>Microcontrolador + Oscilador + LDO + Módulo LoRa</b>	Transmisión	123
	Sleep	0.0858
<b>Microcontrolador + Oscilador + Módulo LoRa</b>	Transmisión	123
	Sleep	0.0058

Tabla 6.3 - Consumo de los componentes en diferentes modos de funcionamiento

## 6.5.2 Estudio Empírico:

Para llevar a cabo las mediciones se utilizó la placa Electronics Explorer de Digilent que ofrece conexión USB con PC, fuentes de alimentación, osciloscopio, generador de señales analógicas/digitales y conexiones tipo protoboard. El software asociado a la placa es el WaveForms, el cual permite observar y guardar medidas, así como también configurar los diferentes parámetros la placa. Además, debido a que se estimaron consumos pequeños del orden de los uA, se utilizó un amplificador de instrumentación como circuito auxiliar el cual está formado por 3 operacionales de precisión y componentes que le permiten una ganancia diferencial de 76 ( $G_{dif} = 76$ ). Ofrece buenas prestaciones para la función que se necesita como: voltaje de offset despreciable, impedancia de entrada muy alta, lo que produce que la ganancia no se vea afectada por la impedancia de la fuente e impedancia de salida muy baja para que no se vea afectada por la carga. En la figura 6.14 se muestra el circuito utilizado.

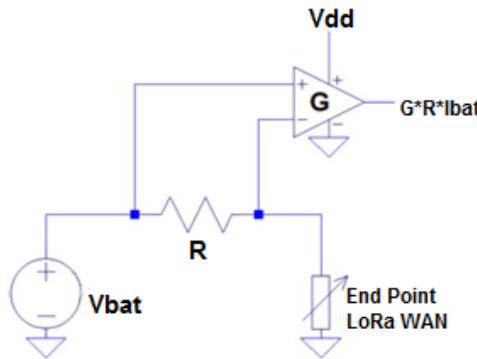


Fig. 6.14 – Circuito auxiliar

El voltaje de salida del circuito viene dado por la ecuación:

$$V_{out} = G_{dif} * (e^+ - e^-) = G_{dif} * R * I_{endpoint} \quad (6.1)$$

Por lo tanto, despejando el valor deseado de  $I_{endpoint}$  se obtiene la ecuación 6.2.

$$I_{endpoint} = \frac{V_{out}}{G_{dif}*R} \quad (6.2)$$

Se alimentó el amplificador con  $V_{dd}=4V$  para evitar saturaciones ya que el voltaje ( $V_{bat}$ ) es de 3,3 voltios. Esto limita la tensión de salida, por lo tanto, se debieron utilizar resistencias adecuadas según el modo de funcionamiento del *endpoint* para qué la medida se dé en ordenes aceptables al circuito y la placa.

Utilizando las corrientes de endpoint de la tabla 6.3 se buscaron valores de  $V_{OUT}$  del orden de los 3.5V y la resistencia dada por:

$$R = \frac{V_{out}}{G_{dif}*I_{endpoint}} \quad (6.3)$$

Dependiendo de esto y de los materiales con los que se contaron, los valores utilizados son los de la tabla 6.4:

Modo	R (Ohm)
Normal transmisión	0.27
Sleep	3917

Tabla 6.4 – Resistencias utilizadas en los diferentes modos de funcionamiento

En la figura 6.15 se observa todo el sistema implementado para realizar las pruebas.

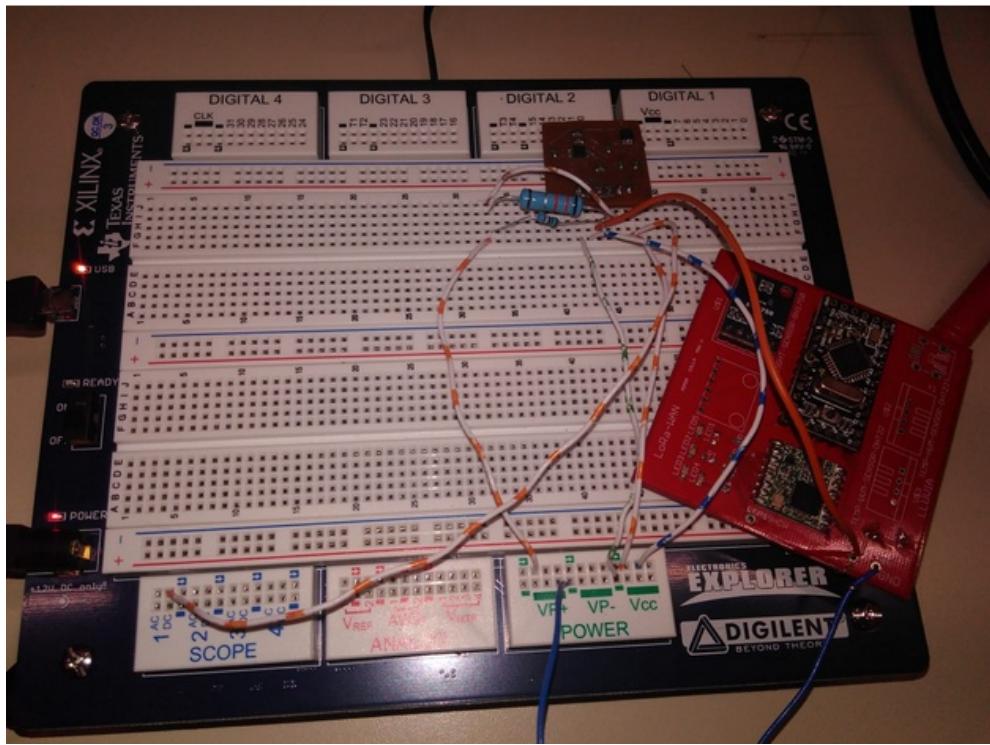


Fig. 6.15 – Circuito implementado con la placa Electronics Explorer para medir el consumo del nodo

### 6.5.3 Realización de las Medidas:

Se tomaron medidas de un *endpoint* en diferentes situaciones de programa y de hardware para comparar con los valores teóricos del dispositivo. Las situaciones bajo las cuales se realizaron las medidas fueron en trasmisión y sleep.

En una primera instancia se realizaron las mediciones al equipo con todos sus componentes; es decir, microprocesador, módulo de comunicación, leds y LDO funcionando sin limitantes.



Fig. 6.16 – Medidas de consumo del hardware con leds y LDO.

De la figura 6.16 se desprende que mientras que el nodo se encuentra en modo sleep este presenta un consumo de 15,75mA. Por otra parte, los picos refieren al

momento en que el equipo realiza una transmisión de datos. En esta instancia el consumo se eleva a 194mA.

En segundo lugar, se procedió a retirarle los leds a la placa, para luego efectuar la misma medida obteniendo las medidas de consumo como indica la figura 6.17.

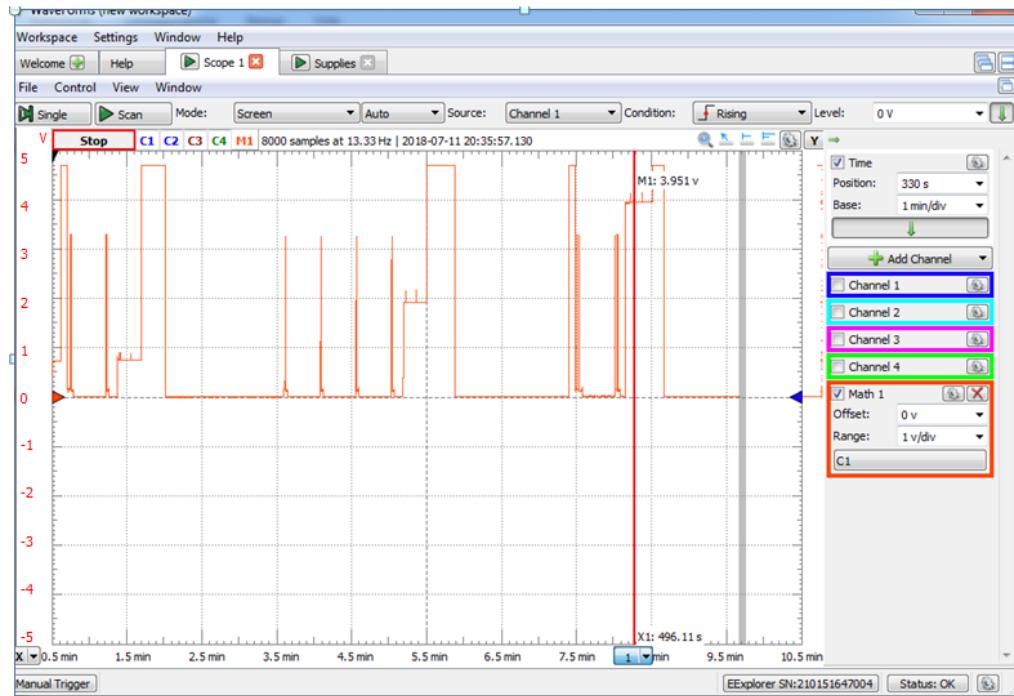


Fig. 6.17 – Medidas de consumo del hardware con LDO y sin leds

En este caso fueron de 85uA en modo sleep y 180mA en transmisión.

Por último, se le quitó el LDO ya que se vio que era uno de los componentes que consumía gran cantidad de corriente en comparación con otros componentes. Por otro lado, bajo las condiciones de uso normal, el microprocesador puede trabajar con la tensión que entregan las baterías sin necesidad de regularla, por lo cual esto no generaría ningún problema a futuro. Sin embargo, al realizar esto es necesario tener precaución de no alimentar el circuito con una batería mayor de 5,5 voltios ya que el integrado puede trabajar en un rango entre 1,8 y 5,5 V sin necesidad de un regulador.

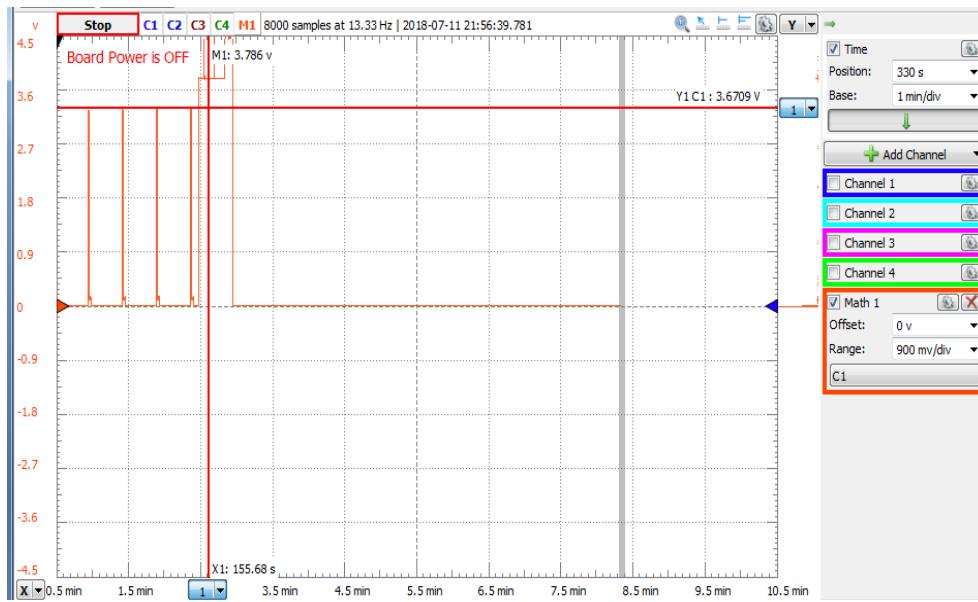


Fig. 6.18 – Medidas de consumo del hardware sin LDO ni leds

Para este último caso las medidas fueron las siguientes: en modo sleep presentó un consumo de 13uA, mientras que al trasmitir los datos este alcanzaba los 180mA.

Por lo tanto, luego de realizadas las pruebas, los resultados prácticos se los compara con los teóricos de la tabla 6.3.

Hardware	Modo	Valor teórico	Valor práctico
Completo	Sleep		15.75 mA
	Transmisión	123 mA	194mA
Sin leds	Sleep	85,8 uA	85uA
	Transmisión	123 mA	180mA
Sin Leds ni LDO	Sleep	5,8 uA	13uA
	Transmisión	123 mA	180mA

Tabla 6.5 – Cuadro comparativo con los datos teóricos y prácticos de consumo

Se puede concluir que los componentes que afectan más drásticamente el consumo son los leds que se encuentran prendidos de forma continua. Además, se observan grandes diferencias entre los valores teóricos y los prácticos. Estos se asocian a que los componentes fueron adquiridos a fabricantes poco confiables.

Sabiendo que el período de transmisión es de 0,225 segundos y conociendo también el comportamiento del *endpoint* y el consumo del mismo bajo distintas circunstancias, es posible estimar el tiempo que estarán en funcionamiento las baterías antes de precisar que ser remplazadas. Suponiendo que el nodo toma una medida y la envía al *gateway* cada 2 horas, una batería de las utilizadas en este trabajo de aproximadamente 3000mAh presentará una duración de 5 años y medio. En el gráfico de

la figura 6.19 se muestra el tiempo en años que dura una batería en función de la cantidad de transmisiones que hace en un día.

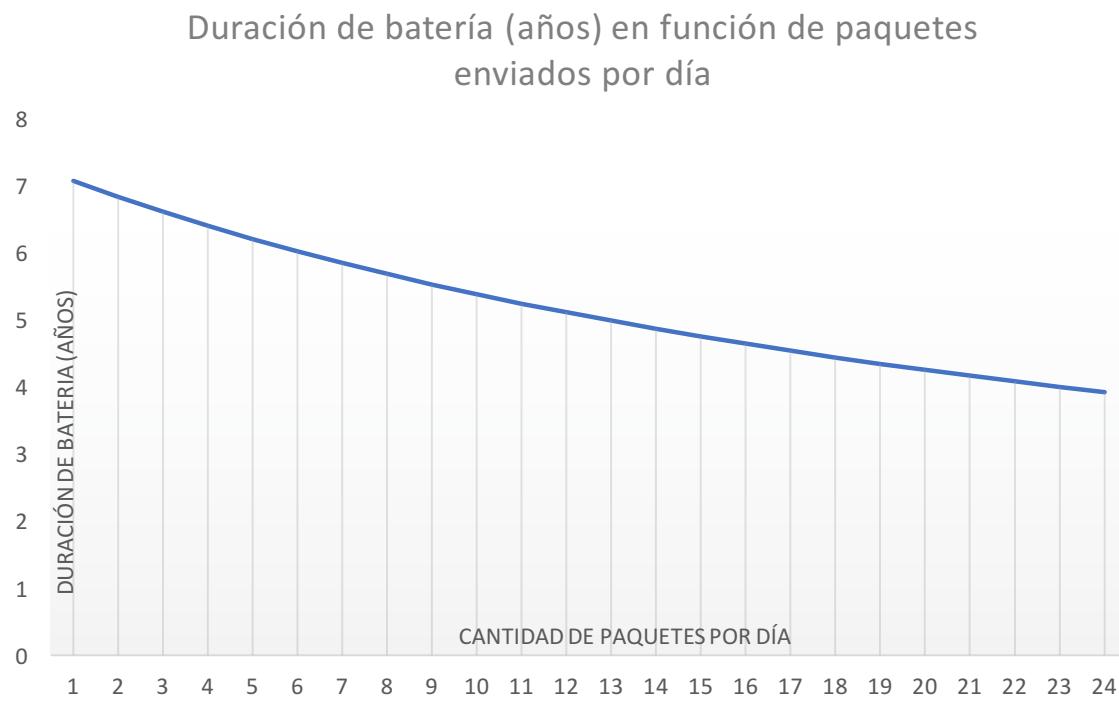


Fig. 6.19 – Gráfico de duración en años en función de la frecuencia de envíos

**Aclaración:** Este estudio se realizó con un nodo clase A, por lo cual pasados esos 0,225 segundos el nodo entraba en modo sleep durante un tiempo prolongado. Para el caso de *endpoints* clase B la duración de la batería será menor como se explicó en el apartado 3.3.2.1.

## 6.6 Estudio de Costos

A continuación, se detallará de forma aproximada la inversión que se requieren hacer para implementar una pequeña red propia.

### 6.6.1 Gateway

Como bien se explicó en apartados anteriores, nuestro equipo está formado no solo por el RAK 831, sino que también es necesario una serie de componentes más que se detallan en la tabla 6.6.

Cabe destacar que los precios que se detallan en las tablas 6.6 y 6.7 refieren a precios en dólares, donde el valor final está dado por el precio de mercado de Estados Unidos, sumado los gastos de traerlos a Uruguay. A su vez, refieren a objetos comprados al por menor, por lo que adquiriéndolos en grandes cantidades estos serán menores.

COMPONENTE	PRECIO
RAK 831	130 dólares
RPI 3	40 dólares
Memoria micro SD 32 GB	10 dólares
Antena 915 MHz	40 dólares
Conecotor SMA / N hembra	6 dólares
Case	5 dólares
<b>TOTAL</b>	<b>231 dólares</b>

Tabla 6.6 – Detalle de gastos del gateway

Para este trabajo se optó por adquirir un *gateway* para desarrolladores de gama media el cual ronda los 250 dólares aproximadamente. De todas formas, es necesario mencionar que se pueden encontrar a la venta equipos más potentes de una mejor calidad cuyos precios pueden superar los 1000 ya que suelen poseer incorporados módulos 3G para facilitar la comunicación con internet, así como también permitir el uso en escenarios outdoor. A su vez, los de mayor costo poseen un procesador más potente que permite que la traducción LoRa-IP sea más rápida

## 6.6.2 Nodos

Para este caso se tomará como ejemplo el costo de los componentes para fabricar un solo dispositivo final.

COMPONENTE	PRECIO
Arduino Pro Mini 8GHz	3 dólares
RFM95W	6 dólares
PCB	3 dólares
Conecotor SMA	1 dólar
Antena 915 MHz	4 dólares
Sensor	2 dólares
<b>TOTAL</b>	<b>19 dólares</b>

Tabla 6.7 – Detalle de gastos de un endpoint

Los costos de la tabla 6.7 están referidos a comprar los componentes al por menor en cantidades menores a 10 de cada uno.

# Capítulo 7

## Conclusiones

Una vez realizados todos los estudios pertinentes a esta tecnología, así como también las pruebas de funcionamiento de la misma; se está en condiciones de concluir los aspectos que se deben tener en cuenta a la hora de planificar y desplegar una red LoRaWAN.

En primer lugar, es necesario resaltar la importancia que tiene en estos días todo lo relacionado a IoT ya que las más importantes empresas de tecnología se encuentran investigando y desarrollando aplicaciones y servicios orientados a conectar las cosas a internet. Esto genera no solo un ahorro a mediano plazo para los usuarios que la utilicen, sino que también le permiten a este no solo automatizar procesos que hasta el momento eran tediosos de realizar, sino que también ayudan a conocer mejor el medio que los rodea (mediante utilización de sensores) y de esta forma tomar acciones más rápido o mismo predecir sucesos. Por otro lado, abre un nuevo nicho de mercado para los ingenieros y desarrolladores interesados en el tema ya que la variedad de cosas a implementar en esta rama de las ciencias es muy amplia. Sumado a ello, el avance de la tecnología en los últimos años ha acompañado de buena manera para que IoT ya no sea algo del futuro, sino que como dice el lema de LoRa Alliance “*el internet de las cosas es ahora*”. El aumento de ancho de banda a nivel mundial, así como las nuevas modulaciones que permiten una mayor velocidad de transferencia hacen posible que IoT sea una realidad.

En rasgos generales, existen variados protocolos de comunicación dentro de la familia de las IoT y la elección de cual utilizar la hace el usuario en función de sus necesidades. Las mencionadas en este trabajo fueron tres (Sigfox, NB-IoT y LoRaWAN) las cuales tienen la particularidad de que todas ellas pertenecen al grupo de las LPWAN. Estas se caracterizan por ser capaces de enviar una decena de bytes a largas distancias consumiendo poca batería. Este último aspecto es muy importante para dispositivos que se encuentran en lugares remotos sin acceso a conexión eléctrica y que la

optimización de las baterías es fundamental. Para estos casos LoRaWAN es una de las tecnologías propicias ya que como se vio en este trabajo, la duración de las mismas ronda los 5 años bajo condiciones promedio. Por otro lado, un aspecto importante que las diferencia de Sigfox o Narrow Band es la posibilidad de crear una red privada sin la necesidad de depender de empresas externas y a costos relativamente baratos. Esto se debe a que LoRaWAN se encuentra respaldada por una alianza llamada LoRa Alliance la cual la conforman importantes empresas del medio e invierten dinero en busca de abaratar los costos de los dispositivos y hacer así más fácil el acceso a ellos. Por otro lado, las frecuencias que utiliza, si bien depende de la región en la que se encuentre el equipo, pertenecen siempre a las bandas ISM (bandas libres), por lo cual no es necesario adquirir permisos para traficar en ellas; cosa que encarecería mucho el despliegue para un usuario particular.

Una vez dicho todo esto y antes de entrar en detalle de cómo debe ser la red planificada, es necesario saber que LoRaWAN es óptima para aquellos casos donde no se requiera enviar paquetes cada pocos minutos, sino que más bien está pensada para unos pocos envíos por día (ideal para sensores). Por otro lado, se recomienda su uso en caso de ser necesario realizar transmisiones a largas distancias y consumiendo poca energía eléctrica. Si se pretende lograr comunicaciones en tiempo real, LoRaWAN no es la más propicia para ello y el porqué de esto se puede ver claramente en las pruebas de latencia y delay del capítulo 6.

Si las necesidades del usuario cumplen con lo mencionado en el párrafo anterior, entonces este protocolo es el adecuado para él.

Una red LoRaWAN está conformada por una serie de nodos, los cuales envían la información al servidor de aplicaciones pasando previamente por un gateway y un servidor de red. Los nodos o *endpoints* son dispositivos formados por un módulo de comunicación que permite la modulación en LoRa, un microcontrolador que se encarga de la comunicación con este y con los sensores pertinentes y una antena capaz de transmitir a la frecuencia deseada. Para el caso de Uruguay este respeta el standard y se suele utilizar US-915. Si bien la comunicación puede ser bidireccional, lo más común es que estos realicen más transmisiones que recepciones de paquetes. Se recomienda leer el apartado que habla de los distintos tipos de clases en caso de que se precise enviar información del *gateway* al *endpoint*.

Luego, los *gateways* (parte más costosa de la red) hacen de puente entre los servidores y los nodos. Son capaces de hablar TCP-IP hacia el servidor de red y LoRa hacia fuera.

Por último, aparecen los servidores de red y aplicaciones los cuales permiten no solo darle integridad y seguridad a la red mediante una serie de claves, sino que también le da la posibilidad al usuario de enviar comandos a él/los nodos, así como también realizar integraciones con servidores propios o base de datos para un mejor manejo de la información de backend. En este trabajo se utilizó TTN por diversos motivos entre los que se destacaba el hecho de ser gratis. En caso de ser necesario se pueden utilizar otros más potentes que permiten un manejo más amplio a nivel de aplicación.

Una vez configurados los equipos que conforman la red y los servidores correspondientes, es necesario planificar la cantidad de antenas que se deben tener a partir del número de *enpoints*, la distancia de estos a las antenas, el tipo de información a enviar y la frecuencia de envíos de los mismos. En este documento se explicó el procedimiento que se debe llevar a cabo para realizar este cálculo. Se recomienda por parte del escritor, adquirir un *gateway* que pueda escuchar y enviar en más de un canal a la vez como el RAK831 ya que esto optimiza aún más la red y disminuye la cantidad de antenas a instalar.

En una primera instancia se había puesto como objetivo el poder diseñar una red para 50 dispositivos, pero a partir de las observaciones obtenidas de las pruebas y el estudio teórico del protocolo, se vio que este número estaba muy por debajo de lo que se suele conectar a un mismo *gateway*.

Suponiendo que los nodos se encuentren a 8 kilómetros de distancia cada uno de la estación central (*time on air* de 350 ms)<sup>18</sup>, envían un mensaje una vez por hora de largo 15 bytes y el *duty cycle* de la red es de 1%; se estima que la antena puede trabajar con 101 nodos en simultaneo. Este dato es el doble de lo que se manejaba, por lo tanto, es claro que para 50 nodos con una única estación central ya es suficiente.

Bajo condiciones dentro de ciudad la distancia entre los nodos y el *gateway* es bastante menor, pero aparecen otros problemas de interferencia por lo cual el tiempo en el aire del paquete si bien será menos a 350 milisegundos, aparecerán otros problemas. De todas formas, LoRaWAN al utilizar frecuencias por debajo del giga-hertz, no es tan vulnerable a interferencias como lo son señales RF en otros rangos superiores de frecuencias. Cabe destacar que en este caso no se tuvo en cuenta la pérdida de paquetes que puede aparecer ya que para ello las medidas del punto 6.3 no son del todo representativas para poder extrapolarlas a cantidades mayores a 4 nodos.

**En este trabajo se demostró que LoRaWAN es una tecnología muy buena para implementar una comunicación bidireccional de forma rápida, barata y eficiente. A su vez se detallaron los pasos a seguir para planificar una red y también los modos en que los nodos pueden/deben trabajar. Sin embargo, no se recomienda su utilización en casos donde el dispositivo se encuentre en movimiento (para eso el *gateway* debe soportar clase B). Por último, tampoco es óptima si se desea montar sobre ella aplicaciones que requieran de transmisión en tiempo real o donde es necesario enviar varios paquetes al día.**

---

<sup>18</sup> Dato aproximado obtenido a partir de medidas empíricas realizadas a 8 km del *gateway*



# Anexo A

## Modulación DSSS

DSSS significa Direct Sequence Spread Spectrum o en español espectro ensanchado por secuencia directa. Esta modulación consiste en buscar una secuencia de expansión por la cual se va a multiplicar la señal de datos de forma de transmitir el producto y en el receptor (el cual conoce la secuencia de expansión) podrá recuperar la información [xxxviii].

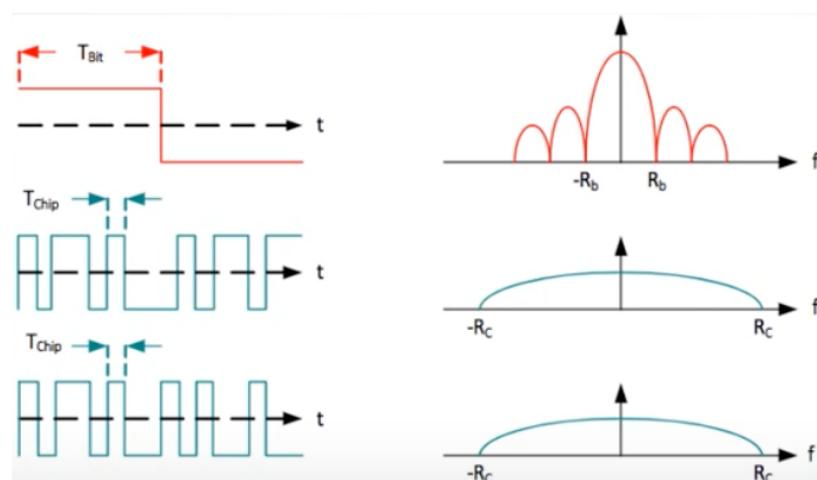


Fig. A.1 – DSSS del lado del emisor [xxiv]

Sea la señal roja de la figura anterior la información a enviar y la segunda la secuencia de código (con una unidad de tiempo denominada *chip*). Estas se multiplican entre sí para obtener la tercera señal modulada. El tiempo de chip debe ser estrictamente menor al tiempo de bit de la señal principal a transmitir.

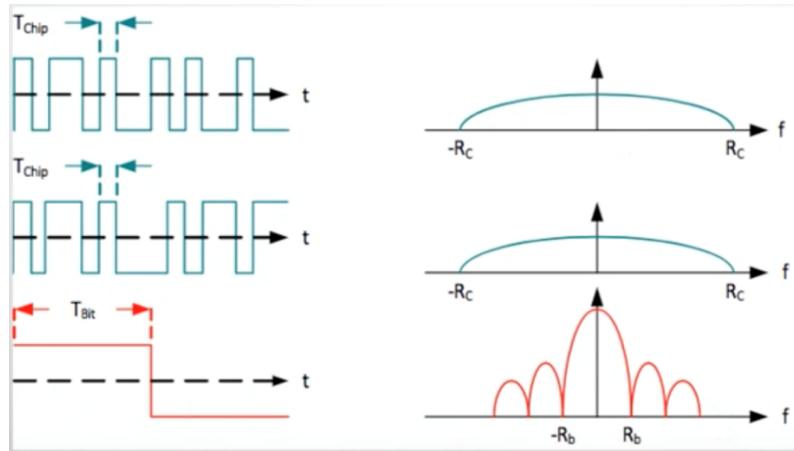


Fig. A.2 – DSSS del lado del receptor [xxiv]

El receptor recibe la señal multiplicada y como este conoce la secuencia de código, realiza el proceso inverso multiplicándola nuevamente por ella. De esta forma recupera la señal principal como se puede apreciar en la figura A.2. Este fenómeno de ampliar el ancho de banda permite tener una ganancia dada por la siguiente ecuación:

$$G = 10 \cdot \log \left( \frac{R_c}{R_b} \right) [dB] \quad (A.1)$$

$$R_b = SF \cdot \frac{BW}{2^{SF}} \cdot \frac{4}{(4 + CR)} \left[ \frac{\text{bits}}{\text{seg}} \right] \quad (A.2)$$

Donde “Rc” es el chip rate (chips por segundo) y “Rb” el bit rate (bits por segundo).

<b>SpreadingFactor (RegModulationCfg)</b>	<b>Spreading Factor (Chips / symbol)</b>	<b>LoRa Demodulator SNR</b>
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Tabla A.1 – Relación entre SF y ganancia [xxiv]

De la tabla A.1 se puede notar que a mayor spread factor, mayor será también la cantidad de chips por símbolo ya que estos se calculan como  $2^{SF}$ . A partir de ello y las ecuaciones A.1 y A.2 se puede ver que a mayor valor de SF, la ganancia será también mayor.

Por otro lado, aparece el concepto de “*chirp*”<sup>19</sup>, el cual es una forma de modulación la cual consiste en variar la frecuencia a modular dependiendo si se tiene un “0” o un “1” para transmitir. Las ventajas de una modulación con Spread Spectrum son:

- 1) Reduce intermodulación.
- 2) En caso de transmitir voz, la modulación es mejor aún
- 3) Aporta integridad a los datos.
- 4) Más susceptible a rebotes y multi caminos
- 5) Aporta seguridad ya que quien intercepte la señal no conoce la frecuencia a la cual se generan los cambios en ella.
- 6) Puede coexistir con otras modulaciones
- 7) Permite mayores distancias
- 8) Es difícil de producir jamming

---

<sup>19</sup> “Chirp” no es lo mismo que “chip”

# Anexo B

## Encriptación AES 128

Para explicarlo hay que entender que se puede dividir en dos. Por un lado, lo que respecta a la creación de las sub-claves, para luego ver cómo a partir de ellas se llega a la matriz de estado final la cual contiene los datos cifrados.

En la figura siguiente se puede apreciar los pasos por los cuales pasan los datos para ser cifrados

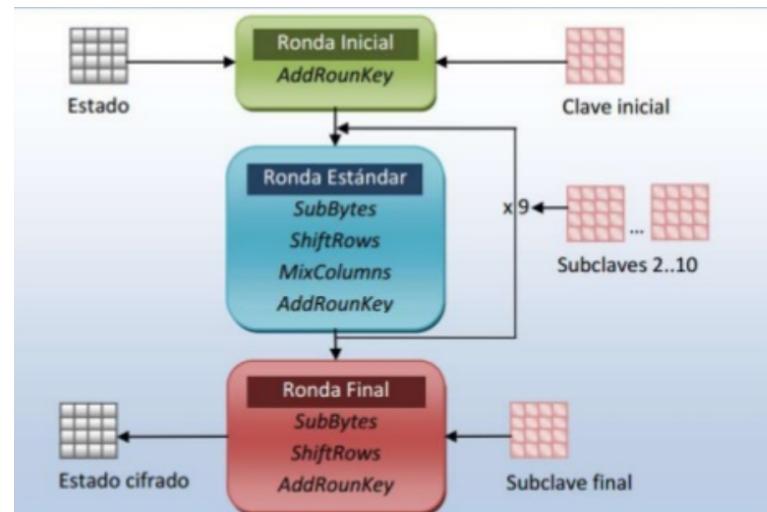


Fig. B.1 - Creación de la matriz de estado final o cifrado a partir del mensaje original y las sub-claves

Para obtener las 10 sub-claves que se generan en el cifrado AES 128 (en caso de que sea AES 192 o 256 la cantidad de estas será mayor), se parte de una clave principal de 128 bits la cual tiene forma de matriz de 4x4 donde cada elemento de ella lo forma una numero hexadecimal de 8 bits cada uno.

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

Fig. B.2 - Ejemplo de clave principal

En primer lugar, se utiliza una operación llamada “RotWord” que rota el primer byte de la columna hacia el último lugar.

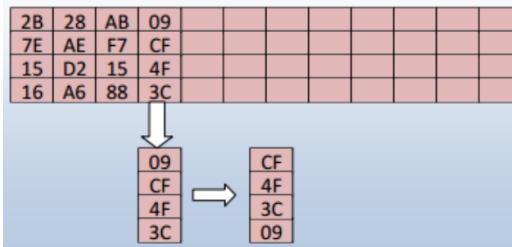


Fig. B.3 - Función RotWord

Luego aplica “SubBytes” (separación) para después hacer la suma XOR entre esta palabra y la que se encuentra 3 posiciones (columnas) más atrás. De esta forma se da origen a la primera palabra de la nueva clave.

		Y																
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
x		0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0		
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15		
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75		
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84		
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf		
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8		
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2		
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73		
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db		
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79		
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08		
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a		
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e		
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df		
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16		

Fig. B.4 - Tabla de sustitución

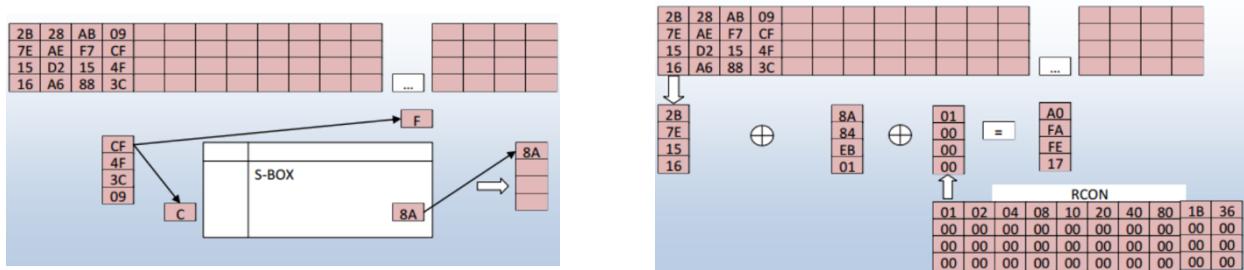


Fig. B.5 - En la figura de la izquierda de hace el “SubBytes” y luego a la derecha se aplica el XOR

A continuación, se repite 3 veces más la suma XOR entre la nueva palabra y la ubicada 3 espacios atrás y así obtener la matriz de 4x4 de la clave. Esto se repite para alcanzar las 10 sub-claves necesarias

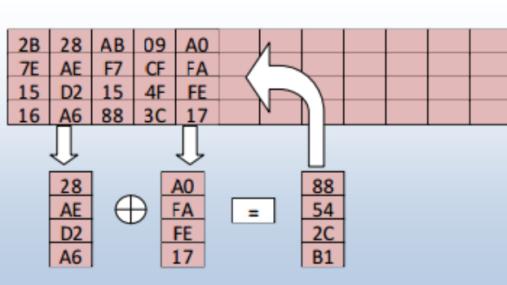


Fig. B.6 - Ultimo paso

Una vez entendido cómo se obtienen las sub-claves, se pasa a detallar cómo se usan cada una de ellas para cifrar el mensaje.

En primer lugar, se toma un bloque de 128 bits del texto plano y se subdivide en bloques de 8 bits cada uno. Estos se pasan a valor hexadecimal y se completa una matriz de 4x4 como se ilustra en la figura 6. A esta la llamaremos matriz de estado.



Fig. B.7 - Creación de la matriz de estado

La primera operación que realiza el algoritmo es la llamada “AddRoundKey” la cual hace la suma OR exclusivo (XOR) entre los bytes del mensaje y los bytes de la clave principal.

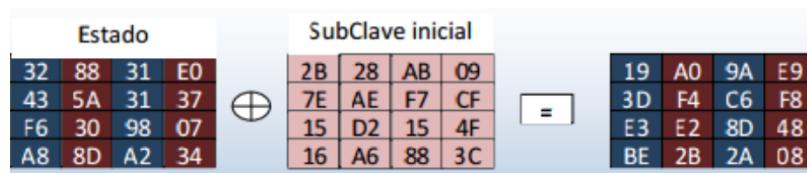


Fig. B.8 - Operación AddRoundKey

Luego, durante las siguientes 9 vueltas se realizarán las operaciones:

- 1) **SubBytes:** Se hace una sustitución de cada valor de la matriz de estado mediante la utilización de una tabla ya vista en la generación de claves
- 2) **ShiftRows:** Permuta filas de estado de forma que la primera no rota, la segunda rota un byte (el de más a la izquierda rota quedando en el último lugar de la fila) la tercera fila rota 2 bytes y la tercera rota 3 bytes.



Fig. B.9 - Función ShiftRows

- 3) **MixColumns**: Multiplica cada columna de la matriz de estado por un polinomio fijo. Este polinomio también es una matriz de 4x4

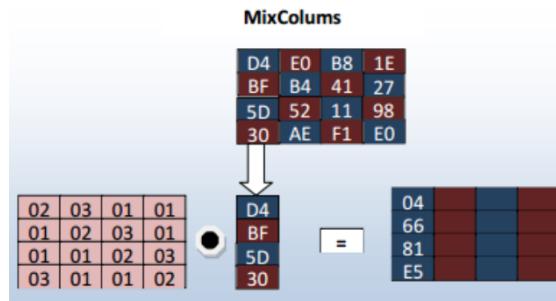


Fig. B.10 - Función MixColumns

- 4) **AddRoundKey**: Realiza la suma XOR de la clave de cada vuelta con los valores de la matriz de estado.

En la última vuelta se realizan las mismas excepto “MixColumns”. De esa forma se obtiene una matriz de estado final que formarán el criptograma resultado final de cifrar el primer bloque de texto plano.

Para el descifrado se hace el proceso inverso con las funciones inversas de forma de obtener el mensaje inicial. Estas funciones son “InvSubBytes”, “InvShiftRows”, “InvMixColumns” e “InvAddRoundKey”.

# Anexo C

## Protocolo ALOHA

ALOHA es un protocolo surgido en la década del 70 para resolver los problemas de acceso múltiple donde usuarios no coordinados compiten por el uso de un solo canal compartido. Existen dos tipos: ALOHA puro y ALOHA ranurado [xxxix].

### ALOHA Puro:

La idea de fondo de dicho protocolo es permitir que los usuarios transmitan cuando tengan datos para enviar y poder resolver de forma eficiente cuando se generan colisiones.

La forma de resolver esto consiste en que una vez que un emisor desea comunicarse con un destinatario, este envía la información y un hub central replica el mensaje a todos los demás nodos incluyendo el emisor. En caso de que se le devuelva el mensaje que el mismo envió, asume que no hubo colisión y por ende el destinatario recibió el paquete. Por el contrario, si no recibe su réplica, interpreta que otro nodo intentó comunicarse al mismo tiempo y se generó una colisión en el medio, por lo cual el hub eliminó ambos mensajes. En caso de suceder esto, los nodos deben esperar un tiempo aleatorio para volver a intentar transmitir la información.

Otra ventaja de recibir constantemente los mensajes de los demás es que estos tienen forma de saber si el medio está ocupado antes de hacer el intento de transmisión. De todas formas, que todos reciban los mensajes de los demás no hace que la red sea insegura ya que estos son descartados cuando no es uno el destinatario. Simplemente sirve para censar el medio.

Una vez que un nodo tomó el canal, este podría adueñárselo tanto tiempo como como precise, generando así gran congestión en la red debido a otros usuarios que no pueden enviar hasta que no se libere. La forma que ALOHA tiene de resolver esto es “partiendo” el paquete en varias tramas pequeñas. De esta forma la probabilidad de saturación disminuye.

## **ALOHA Ranurado:**

ALOHA ranurado o S-ALOHA es una mejora del anterior que permite duplicar la capacidad de un sistema dividiendo el tiempo en intervalos llamados ranuras.

Aquí aparece el concepto de un reloj central, el cual envía información con la señal de reloj a las estaciones periféricas. El funcionamiento es parecido al ALOHA puro, excepto que ahora las estaciones solo pueden enviar los datos al comienzo de cada time slot.

# Anexo D

## Tabla de frecuencias US 902 – 928

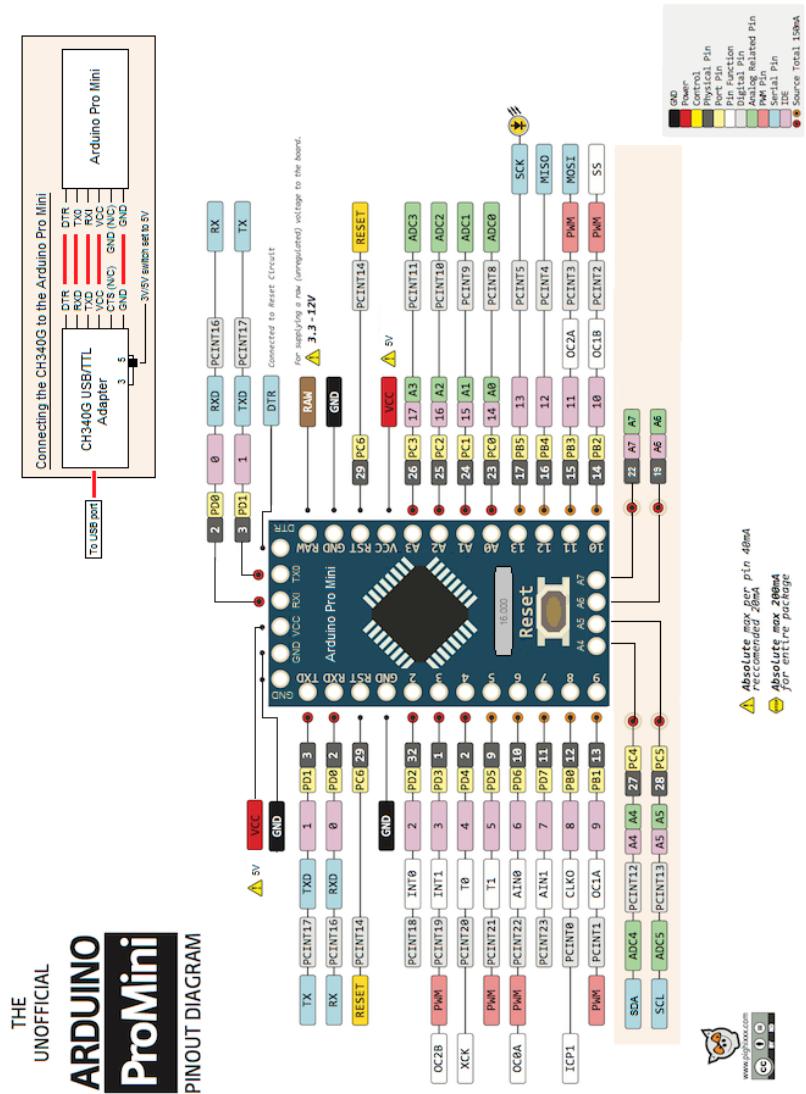
CANAL	ANCHO	DE (MHz)	CENTADO (MHz)	HASTA (MHz)	TIPO	DR
0	125 KHz	902,225	902,3	902,375	UP	DRO - DR3
1	125 KHz	902,425	902,5	902,575	UP	DRO - DR3
2	125 KHz	902,625	902,7	902,775	UP	DRO - DR3
3	125 KHz	902,825	902,9	902,975	UP	DRO - DR3
4	125 KHz	903,025	903,1	903,175	UP	DRO - DR3
5	125 KHz	903,225	903,3	903,375	UP	DRO - DR3
6	125 KHz	903,425	903,5	903,575	UP	DRO - DR3
7	125 KHz	903,625	903,7	903,775	UP	DRO - DR3
8	125 KHz	903,825	903,9	903,975	UP	DRO - DR3
9	125 KHz	904,025	904,1	904,175	UP	DRO - DR3
10	125 KHz	904,225	904,3	904,375	UP	DRO - DR3
11	125 KHz	904,425	904,5	904,575	UP	DRO - DR3
12	125 KHz	904,625	904,7	904,775	UP	DRO - DR3
13	125 KHz	904,825	904,9	904,975	UP	DRO - DR3
14	125 KHz	905,025	905,1	905,175	UP	DRO - DR3
15	125 KHz	905,225	905,3	905,375	UP	DRO - DR3
16	125 KHz	905,425	905,5	905,575	UP	DRO - DR3
17	125 KHz	905,625	905,7	905,775	UP	DRO - DR3
18	125 KHz	905,825	905,9	905,975	UP	DRO - DR3
19	125 KHz	906,025	906,1	906,175	UP	DRO - DR3

20	125 KHz	906,225	906,3	906,375	UP	DRO - DR3
21	125 KHz	906,425	906,5	906,575	UP	DRO - DR3
22	125 KHz	906,625	906,7	906,775	UP	DRO - DR3
23	125 KHz	906,825	906,9	906,975	UP	DRO - DR3
24	125 KHz	907,025	907,1	907,175	UP	DRO - DR3
25	125 KHz	907,225	907,3	907,375	UP	DRO - DR3
26	125 KHz	907,425	907,5	907,575	UP	DRO - DR3
27	125 KHz	907,625	907,7	907,775	UP	DRO - DR3
28	125 KHz	907,825	907,9	907,975	UP	DRO - DR3
29	125 KHz	908,025	908,1	908,175	UP	DRO - DR3
30	125 KHz	908,225	908,3	908,375	UP	DRO - DR3
31	125 KHz	908,425	908,5	908,575	UP	DRO - DR3
32	125 KHz	908,625	908,7	908,775	UP	DRO - DR3
33	125 KHz	908,825	908,9	908,975	UP	DRO - DR3
34	125 KHz	909,025	909,1	909,175	UP	DRO - DR3
35	125 KHz	909,225	909,3	909,375	UP	DRO - DR3
36	125 KHz	909,425	909,5	909,575	UP	DRO - DR3
37	125 KHz	909,625	909,7	909,775	UP	DRO - DR3
38	125 KHz	909,825	909,9	909,975	UP	DRO - DR3
39	125 KHz	910,025	910,1	910,175	UP	DRO - DR3
40	125 KHz	910,225	910,3	910,375	UP	DRO - DR3
41	125 KHz	910,425	910,5	910,575	UP	DRO - DR3
42	125 KHz	910,625	910,7	910,775	UP	DRO - DR3
43	125 KHz	910,825	910,9	910,975	UP	DRO - DR3
44	125 KHz	911,025	911,1	911,175	UP	DRO - DR3
45	125 KHz	911,225	911,3	911,375	UP	DRO - DR3
46	125 KHz	911,425	911,5	911,575	UP	DRO - DR3
47	125 KHz	911,625	911,7	911,775	UP	DRO - DR3
48	125 KHz	911,825	911,9	911,975	UP	DRO - DR3
49	125 KHz	912,025	912,1	912,175	UP	DRO - DR3
50	125 KHz	912,225	912,3	912,375	UP	DRO - DR3
51	125 KHz	912,425	912,5	912,575	UP	DRO - DR3
52	125 KHz	912,625	912,7	912,775	UP	DRO - DR3
53	125 KHz	912,825	912,9	912,975	UP	DRO - DR3
54	125 KHz	913,025	913,1	913,175	UP	DRO - DR3
55	125 KHz	913,225	913,3	913,375	UP	DRO - DR3
56	125 KHz	913,425	913,5	913,575	UP	DRO - DR3
57	125 KHz	913,625	913,7	913,775	UP	DRO - DR3
58	125 KHz	913,825	913,9	913,975	UP	DRO - DR3
59	125 KHz	914,025	914,1	914,175	UP	DRO - DR3
60	125 KHz	914,225	914,3	914,375	UP	DRO - DR3
61	125 KHz	914,425	914,5	914,575	UP	DRO - DR3

62	125 KHz	914,625	914,7	914,775	UP	DR0 - DR3
63	125 KHz	914,825	914,9	914,975	UP	DR0 - DR3
64	500 KHz	902,75	903	903,25	UP	DR4
65	500 KHz	904,35	904,6	904,85	UP	DR4
66	500 KHz	905,95	906,2	906,45	UP	DR4
67	500 KHz	907,55	907,8	908,05	UP	DR4
68	500 KHz	909,15	909,4	909,65	UP	DR4
69	500 KHz	910,75	911	911,25	UP	DR4
70	500 KHz	912,35	912,6	912,85	UP	DR4
71	500 KHz	913,95	914,2	914,45	UP	DR4
0	500 KHz	922,8	923,3	923,8	DOWN	DR8 - DR13
1	500 KHz	923,4	923,9	924,4	DOWN	DR8 - DR13
2	500 KHz	924	924,5	925	DOWN	DR8 - DR13
3	500 KHz	924,6	925,1	925,6	DOWN	DR8 - DR13
4	500 KHz	925,2	925,7	926,2	DOWN	DR8 - DR13
5	500 KHz	925,8	926,3	926,8	DOWN	DR8 - DR13
6	500 KHz	926,4	926,9	927,4	DOWN	DR8 - DR13
7	500 KHz	927	927,5	928	DOWN	DR8 - DR13

# Anexo E

# Datasheet Arduino Pro Mini



# Anexo F

## Secciones del código y configuraciones a nivel de librerías

Debido a que gran parte de este trabajo es la correcta programación de los dispositivos, es importante realizar algunos comentarios sobre el código implementado para no solo entender el trabajo realizado; sino que también poder vincular los conceptos teóricos con los prácticos.

### F.1 Secciones del código

#### F.1.1 Sección Debug

Esta es una de las principales partes del código ya que, a diferencia de las siguientes, es la única donde el usuario debe realizar cambios para definir qué tipo de código desea implementar en el dispositivo.

En dicha sección se le permite al usuario activar las opciones de DEBUG y/o SLEEP. Al utilizar esta primera se irá imprimiendo en la consola una serie de mensajes de forma que le sea fácil al usuario ir entendiendo lo que se va ejecutando y así corroborar su correcto funcionamiento. Por otro lado, la de SLEEP sirve para que el dispositivo

entre en modo ahorro cuando no esté enviando ni recibiendo información. Es muy recomendable tener esta funcionalidad siempre activa.

En esta misma parte del código es donde se configura el método de activación con el cual se desea trabajar (OTAA o ABP). Es estrictamente necesario que solo se deje activada una de las opciones. En caso de que por algún motivo queden activas las dos, el programa no compilará.

```
#define SYNC_AB  
//#define SYNC_OTTA
```

Fig. F.1 – Instancia donde se define el tipo de sincronización a utilizar. Para este caso está habilitada ABP

## F.1.2 Sección claves e iniciación de variables

Aquí es donde se definen las diferentes claves de dispositivo, aplicación y red según corresponda. Estas dependerán del tipo de activación elegido en el punto 4.1.1 y se obtienen del servidor TTN como se explicó anteriormente.

Una vez copiadas en el código, se guardarán en la memoria no volátil del dispositivo. A su vez, se configuran e inicializan las variables donde se guardará el payload a transmitir.

## F.1.3 Sección Intervalo de TX

La funcionalidad de esta sección del código es definir el tiempo en segundos (TX\_INTERVAL) que transcurre entre el envío de un paquete y el siguiente.

En caso de estar definido el modo SLEEP de la sección 4.1.1, este parámetro también cobra importancia ya que define la cantidad de períodos de 8 segundos en que el dispositivo se encuentra en este modo mediante la siguiente ecuación:

$$T_{SLEEP} = \text{Int}(TX\_INTERVAL / 8) \quad (\text{F.1})$$

A modo de ejemplo si TX\_INTERVAL = 60, entonces  $60/8 = 7.5$ , por lo cual estará 7 períodos de 8 segundos en ahorro de energía (56 segundos) ya que toma la parte entera del resultado.

## F.1.4 Sección Eventos

En esta parte del código aparecen los llamados *eventos*. Estos son propios de la librería Imic los cuales ya están definidos en los diferentes archivos que la conforman.

Se le llama evento a todo aquel comportamiento que sufre el dispositivo. Entre los más utilizados se encuentran los de *joined* los cuales toman importancia a la hora de utilizar la sincronización OTAA y son los que indican si ésta se realizó correctamente o hubo fallas. Por otro lado, aparecen los eventos de transmisión y recepción los cuales indican que se realizó una transmisión de datos de forma correcta. Por último, están los relacionados a fallas como timeouts o resets.

De todos los eventos posibles que ofrece la librería lmic, el más relevante es el de transmisión ya que es este el que dispara la función de envío de información.

### F.1.5 Sección Función Enviar

Esta parte el código es uno de los pilares del mismo ya que es acá donde está definida la función capaz de enviar los datos. Esto lo hace mediante el comando “LMIC\_SetTxData2” que se encuentra definida en la librería lmic; al cual se le pasa como parámetros no solo los datos en texto plano a enviar sino también el largo del paquete, el puerto por el cual se enviará y si se desea confirmación de recibido.

### F.1.6 Sección Setup

En la programación en Arduino es obligatorio la función setup para poder ejecutar el archivo. Aquí se definen todas aquellas cosas que son necesarias para poder iniciar el dispositivo y por consiguiente todo aquello que solo se precisa correrlo una sola vez y al principio del mismo como por ejemplo la definición de los pines de entrada/salida, la velocidad de lectura y escritura serial, variables a guardar en la memoria no volátil, etc.

A su vez, en esta sección del código se pueden ver 4 comandos LMIC necesarios para definir ciertos valores para la comunicación LoRa, los cuales se detallan a continuación.

LMIC\_selectSubBand(1): Especifica la sub banda a utilizar. Al marcar 1 se hace referencia a que se usa la segunda banda de 8 canales de transmisión.

LMIC\_setLinkCheckMode(0): 0 este parámetro indica que se deshabilita la validación de los paquetes.

LMIC.dn2Dr = DR\_SF9: Con esta línea se define el valor del Spreading Factor de RX (9) y la ventana de recepción. Estos valores se pueden ver más en detalle en la sección que trata la banda de frecuencias US 902-928.

LMIC\_setDrTxpow(DR\_SF10,14): Se define el valor del Spreading Factor de TX (10) y la potencia en caso de tener apagada la funcionalidad ADR.

### F.1.7 Sección Loop

Este es otro de los módulos más relevantes ya que es aquella instancia que permanece funcionando constantemente de forma cíclica y va llevando el tiempo y la cadena de procesos. Al igual que lo que pasaba con la función “setup”, lo mismo sucede con la “loop” ya que es estrictamente necesario para Arduino que exista una función con este nombre en el programa porque es la que el microcontrolador toma como función principal.

## F.1.8 Sección Funciones

Por último, se encuentra la parte donde se definen una serie de funciones generales y otras tantas propias de cada sensor.

A grandes rasgos se pueden separar en aquellas que son capaces de prender y apagar un led cuando se envía o recibe un paquete; aquellas que permiten leer tanto el paquete antes de ser enviado como también los recibidos y por último una serie de funciones propias de cada sensor que se encargan de ordenar de cierta forma los dato para su posterior transmisión. Estas últimas son las que se replican de forma inversa del lado del servidor de aplicaciones para poder interpretar el dato.

## F.2 Librerías

Para que los sensores funcionen y el módulo de comunicaciones pueda realizar la modulación correspondiente, es necesario incluir una serie de librerías al proyecto .ino las cuales son proporcionadas por los fabricantes.

### F.2.1 Librerías de Sensores

En cuanto a la relacionada al sensor de luz (BH1750FVI) la librería asociada justamente es la “BH1750FVI-master.h” la cual es de acceso libre y se encuentra en GitHub<sup>20</sup>. En segundo lugar, para poder procesar la información que genera el sensor de humedad y temperatura, se utilizó la “DHT.h” la cual es compatible para el DHT22 y DHT11. Al igual que para el caso anterior, ésta es de acceso público desde el repositorio de GitHub<sup>21</sup>.

La particularidad que tienen las librerías en Arduino es que estos archivos son modificables, por lo cual en caso de que el usuario pretenda cambiar un parámetro de ella, lo podrá hacer sin problema. Para estos dos mencionados anteriormente no se realizó ningún cambio, pero si se hizo en otra llamada “Imic-master.h”; los cuales se detallarán en la siguiente sección 4.2.3.

### F.2.2 Librerías para comunicación serial

Para la comunicación del microprocesador con los diferentes dispositivos periféricos, es necesario agregar al código de programación la librería “SPI.h” la cual permite la comunicación síncrona serial entre ellos. En este proyecto, el periférico conectado al Arduino es el módulo de comunicación LoRa, el cual se conecta a los pines MOSI, MISO y SCK. Estos son importantes para llevar a adelante la comunicación serial, donde el sincronismo está dado por el reloj interno del micro y sincroniza el RFM95W mediante la salida SCK (Serial Clock).

---

<sup>20</sup><https://github.com/claws/BH1750/blob/master/BH1750.h>

<sup>21</sup><https://github.com/adafruit/DHT-sensor-library>

## F.2.3 Librería LoRaWAN

Para la librería encargada de la modulación LoRa y lo referente a LoRaWAN, se optó por utilizar la “*lmic-h*”. Esta fue creada y mantenida en sus inicios por IBM para los módulos LoRa SX1272 y SX1276 de Semtech. Actualmente es de uso libre y gratuita para los usuarios que deseen incursionar en LoRa y está gestionada por el usuario *matthijskooijman* de GitHub.

Al ser open source se puede quitar, modificar o agregar parámetros según lo requiera el programador y se encuentra formada por una serie de archivos .c y .h los cuales contienen la programación por defecto del módulo. Con la ayuda del datasheet del módulo se modificaron algunos parámetros de forma de aumentar la potencia de transmisión [xi].

Uno de los bloques a los cuales se les hizo una modificación fue al PA\_BOOST de forma de aumentar la potencia de transmisión.

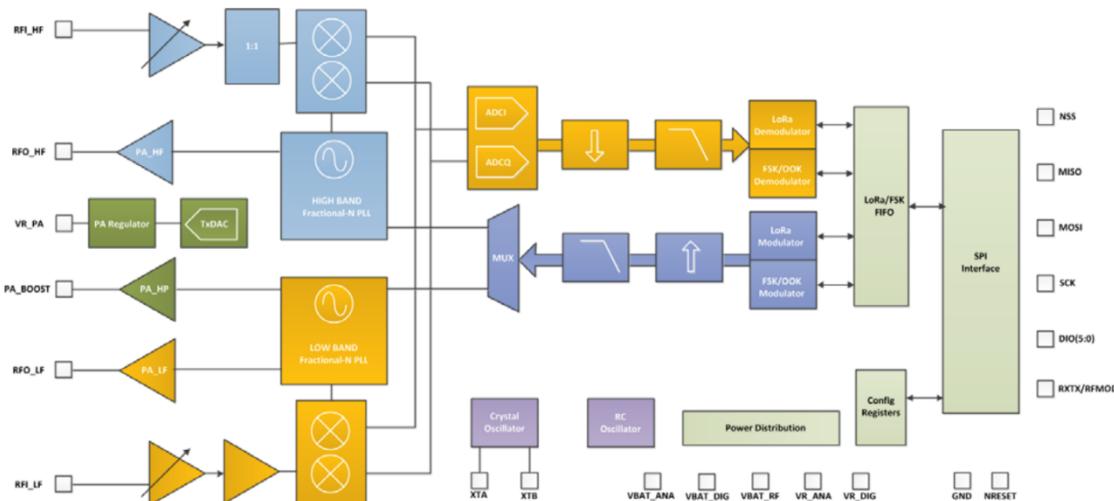


Fig. F.2 – Diagrama de bloques del módulo SX1276 [xxiv]

Este PA\_BOOST está conformado por una serie de registros internos donde los que se encargan de determinar la potencia son el “RegPaConfig” y “RegLna”. Ellos vienen con una configuración básica por defecto, pero en caso de querer aumentar la potencia hay que cambiarlos a otros valores. Para ello nos ayudamos del datasheet correspondiente.

### F.2.3.1 RegPaConfig

Este determina la potencia de salida y se encuentra alojado en el lugar 0x09 de la memoria interna. Como se puede ver en la imagen 4.3, el largo de la misma es de 8 bits, donde cada uno de ellos tiene un significado indicado en la columna de la izquierda.

RegPaConfig (0x09)	7	PaSelect	rw	0x00	Selects PA output pin 0 → RFO pin. Output power is limited to +14 dBm. 1 → PA_BOOST pin. Output power is limited to +20 dBm
	6-4	MaxPower	rw	0x04	Select max output power: Pmax=10.8+0.6*MaxPower [dBm]
	3-0	OutputPower	rw	0x0f	Pout=Pmax-(15-OutputPower) if PaSelect = 0 (RFO pin) Pout=17-(15-OutputPower) if PaSelect = 1 (PA_BOOST pin)

Fig. F.3 – Registro RegPaConfig [xxvi]

Por defecto toma el valor 0x80 (10000000) el cual indica que el boost se encuentra prendido pero la potencia no es máxima. Por lo tanto, este registro fue modificado a 0xFF el cual representa 11111111 e indica que no solo se habilita el boost, sino que también se asigna la mayor potencia de salida permitida.

### F.2.3.2 RegLna

El LNA (Low-Noise Amplifier) es un amplificador electrónico capaz de poder amplificar señales débiles y mejorar el nivel de ruido de ellas para una mejor recepción. Se vio que, modificando este parámetro, la recepción de señales mejoraba. Dicho registro se encuentra en el lugar 0x0C de la memoria y está formado por las variables detalladas en la figura 4.4.

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegLna (0x0C)	7-5	LnaGain	rwx	0x01	LNA gain setting: 000 → not used 001 → G1 = maximum gain 010 → G2 011 → G3 100 → G4 101 → G5 110 → G6 = minimum gain 111 → not used
	4-3	LnaBoostLf	rw	0x00	Low Frequency (RFI_LF) LNA current adjustment 00 → Default LNA current Other → Reserved
	2	reserved	rw	0x00	reserved
	1-0	LnaBoostHf	rw	0x00	High Frequency (RFI_HF) LNA current adjustment 00 → Default LNA current 11 → Boost on, 150% LNA current

Fig. F.4 – Registro RegLna [xxvi]

El valor que se asignó para este trabajo fue de 0x23 (00100011).

# Referencias

---

- [i] Wikipedia (julio de 2018), *Espectro Ensanchado*. [https://es.wikipedia.org/wiki/Espectro\\_ensanchado](https://es.wikipedia.org/wiki/Espectro_ensanchado)
- [ii] LoraAlliance, *WIDE AREA NETWORKS FOR IOT*. [https://docbox.etsi.org/Works hop/2017/201710\\_IoTWEek/WORKSHOP/S04\\_CONNECTING\\_IoT/LoRaAlliance\\_THUBER T.pdf](https://docbox.etsi.org/Works hop/2017/201710_IoTWEek/WORKSHOP/S04_CONNECTING_IoT/LoRaAlliance_THUBER T.pdf)
- [iii] Seleha Raiz (mayo de 2016), *Samsung teams with SK Telecom for “world-first” LoRa network*, <https://www.mobileworldlive.com/featured-content/home-banner/samsung-teams-up-with-sk-telecom-for-iot-network-in-south-korea/>
- [iv] ETSI (mayo de 2017), *Primera Red LoRaWAN de Huelva*. <http://www.uhu.es/etsi/primera-red-lorawan-huelva/>
- [v] LoRa Alliance (2018). <https://www.lora-alliance.org/>
- [vi] YEAP! (2018). <http://www.yeap.com.ar/>
- [vii] Bruno Bellini (diciembre de 2016), *DIEstro – Motion Sensor platform for cattle oestrus detection*. Documento de maestría de la Universidad Católica del Uruguay
- [viii] Mario Ortiz, Martín Marioni, Gisselle Vogel (noviembre de 2016), *eTag – Prototipo E-ink Price Tag*. Memoria de Grado de la Universidad Católica del Uruguay
- [ix] Kirber Pineda (enero de 2018), *¿Qué es IoT?*. <http://smartbasegroup.com/qu e-es-el-iot/>
- [x] Sigfox (n/a). <https://www.sigfox.com/en>
- [xi] Wikipedia (octubre de 2018), *Narrowband IoT*. [https://en.wikipedia.org/wiki/Narrowband\\_IoT#3GPP\\_Narrowband\\_Cellular\\_Standards](https://en.wikipedia.org/wiki/Narrowband_IoT#3GPP_Narrowband_Cellular_Standards)

---

**[xii]** Linkabs (junio de 2018), *NB-IoT vs LoRa vs Sigfox*. <https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox>

**[xiii]** Wikipedia (julio de 2018), *Espectro Ensanchado*. [https://es.wikipedia.org/wiki/Espectro\\_ensanchado](https://es.wikipedia.org/wiki/Espectro_ensanchado)

**[xiv]** TechPlayOn (diciembre de 2017), *LoRa link budget and Sensitivity Calculations Example Explained*. ,<http://www.techplayon.com/lora-link-budget-sensitivity-calculations-example-explained/>

**[xv]** Paul Pickering (junio de 2017), *Desarrollar con LoRa para aplicaciones IoT de baja tasa y largo alcance*. <https://www.digikey.es/es/articles/techzone/2017/jun/develop-lora-for-low-rate-long-range-iot-applications>

**[xvi]** NEWIE VENTURES (n/a), *OTAA or ABP*. [https://static1.squarespace.com/static/560cc2c2e4b01e842d9fac18/t/5a938d38ec212d9451fbecf8/1519619387035/OTAA\\_or\\_AB Pv3](https://static1.squarespace.com/static/560cc2c2e4b01e842d9fac18/t/5a938d38ec212d9451fbecf8/1519619387035/OTAA_or_AB Pv3).

**[xvii]** Lora Alliance (2018), *LoRaWAN™ 1.1 Specifications*. <https://lorawan.org/resource-hub/lorawantm-specification-v11>

**[xviii]** Sabas (octubre de 2017), *Haciendo IoT con LoRa: Capítulo 2 – Tipos y Clases de Nodos*. <https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be>

**[xix]** The Things Network (2018), *LoRaWAN Frequencies Overview*. <https://www.thethingsnetwork.org/docs lorawan/frequency-plans.html>

**[xx]** Arjanvanb (febrero de 2016), *Limitations: Data rate, packet size, 30 seconds uplink and 10 messages downlink per day Fair Access Policy*. <https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day-fair-access-policy/1300>

**[xxi]** Arjanvanb (febrero de 2017), *How many gateways are needed to support 100 nodes sending 10 bytes*. <https://www.thethingsnetwork.org/forum/t/how-many-gateways-are-needed-to-support-1-000-nodes-sending-10-bytes-per-5-minutes/5527/2>

**[xxii]** The Things Network (2018), *Communities*. <https://www.thethingsnetwork.org/community>

**[xxiii]** The Thinks Network (2018), *LEARN*. <https://www.thethingsnetwork.org/docs/>

**[xxiv]** Atmel (noviembre de 2016), *Datasheet ATmega328P*. [http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf)

---

**[xxv]** Semtech (mayo de 2015), *AN1200\_Semtech\_LoRa\_Basics\_v2\_STD*. <https://www.semtech.com/uploads/documents/an1200.22.pdf>

**[xxvi]** Hoperf Electronic (n/a), *RFM95/96/97/98(W)* - *Low Power Long Range Transceiver Module V1.0*. [http://www.hoperf.com/upload/rf/RFM95\\_96\\_97\\_98W.pdf](http://www.hoperf.com/upload/rf/RFM95_96_97_98W.pdf)

**[xxvii]** OSEPP Electronic (2016-2017), *DHT11Humidity & Temperature Sensor*. <https://www.mouser.com/ds/2/268/4001843B-1074150.pdf>

**[xxviii]** ROHM Semiconductor (2011), *BH1750FVI*. <https://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf>

**[xxix]** BOSCH (mayo de 2015), *BMP280 Digital Pressure Sensor*. <https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf>

**[xxx]** Rakwireless (n/a), *RAK831 Lora Gateway Datasheet 1.3* (2017). <http://docs.rakwireless.com/en/LoRa/RAK831-Lora-Gateway/Hardware-Specification/RAK831%20Datasheet%20V1.3.pdf>

**[xxxi]** The Thing Network (2018), *LEARN Specifications*. <https://www.thethingsnetwork.org/docs/gateways/kerlink/specs.html>

**[xxxii]** Kerlink (marzo de 2016), *Wirnet Station 923*. [https://m2mconnectivity.com.au/downloads/Data%20Sheets/Kerlink/DataSheet\\_wirnet\\_station-923.pdf](https://m2mconnectivity.com.au/downloads/Data%20Sheets/Kerlink/DataSheet_wirnet_station-923.pdf)

**[xxxiii]** Cisco (junio de 2018), *Cisco Wireless Gateway for LoRaWAN Data Sheet*. <https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/datasheet-c78-737307.html>

**[xxxiv]** Multitech (setiembre de 2018), *MultiConnect Conduit*. <https://www.multitech.com/documents/publications/data-sheets/86002193.pdf>

**[xxxv]** Neresh Krish (julio de 2017), *Getting started with RAK 831 LoRa Gateway and RP13*. <https://www.hackster.io/naresh-krish/getting-started-with-the-rak-831-lora-gateway-and-rpi3-e3351d>

**[xxxvi]** RAK (2016), *RAk831 + Raspberry Pi3 Quick Start Guide*. <http://s3.cn-north-1.amazonaws.com.cn/docs.rakwireless.com/en/LoRa/RAK831-Lora-Gateway/Application-Notes/RAK831-&-Raspberry-Pi3-Quick-Start-Guide.pdf>

**[xxxvii]** Raspberry (n/a), *Installing Operating System Images*. <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

**[xxxviii]** Semtech (mayo de 2015), *AN1200\_Semtech\_LoRa\_Basics\_v2\_STD*. <https://www.semtech.com/uploads/documents/an1200.22.pdf>

---

[xxxix] Wikipedia (mayo de 2018), *ALOHANet*. <https://es.wikipedia.org/wiki/ALO%20HAnet>

## Datasheets de interés:

[xli] Arduino (2018), *Arduino Pro Mini*. <https://store.arduino.cc/usa/arduino-pro-mini>

[xlii] Microchip (2017), *MIC5205*. <http://ww1.microchip.com/downloads/en/DeviceDoc/20005785A.pdf>

[xliii] Freescale Semiconductor (octubre de 2016), *M68HC08*. <https://www.nxp.com/docs/en/data-sheet/MC68HC908JL3E.pdf>

[xliii] Microchip (2016-2017), *PIC18(L)F24/25K40*. <https://www.mouser.com/ds/2/268/40001843B-1074150.pdf>