Proyecto de Grado

Ingeniería en Computación

2019

Aplicación de IoT con diversas tecnologías inalámbricas

Autor:
Federico Detta

Docente Supervisor:
Matías Richart

Docente Supervisor:
Eduardo Grampin

29 de febrero de 2020

Resumen

El objetivo general del proyecto es el desarrollo completo (de punta a punta) de una aplicación sencilla de IoT (Internet of Things). Con esto se busca generar conocimiento en el uso de distintas tecnologías inalámbricas para la comunicación entre dispositivos de IoT así como en la recolección, almacenamiento y acceso a los datos. Se propone como caso de uso implementar una aplicación que, mediante el uso de sensores, permita contar la cantidad de individuos con dispositivos inalámbricos que se encuentran en un área determinada y disponibilizar estos datos.

Índice

1. Introducción **4**
1.1. Caso de uso . . . 4
1.2. Estado del Arte . . . 4
1.3. Trabajos relacionados . . . 5
1.4. Arquitectura definida . . . 6

2. Conteo de dispositivos Wi-Fi **8**
2.1. Arquitectura 802.11 . . . 8
2.2. Trama 802.11 . . . 8
2.3. Modo promiscuo de interfaz inalámbrica . . . 10
2.4. Anonimización de dirección MAC . . . 10
2.4.1. Android . . . 11
2.4.2. iOS . . . 11
2.5. Consideraciones Éticas . . . 11

3. Ecosistema LoRaWAN **13**
3.1. Protocolo LoRa . . . 13
3.1.1. Ancho de Banda . . . 14
3.1.2. Modulación CSS . . . 14
3.1.3. Spreading Factor . . . 15
3.1.4. Link Budget . . . 15
3.1.5. Duty Cycle . . . 16
3.2. LoRaWAN . . . 16

4. Servidor de LoRaWAN **21**
4.1. The Things Network (TTN) . . . 21
4.1.1. Integración de TTN . . . 22
4.2. LoraServer . . . 22

5. Gateway LoRaWAN **24**
5.1. The Things Gateway . . . 24
5.2. Gateway Sparkfun SPX-14893 . . . 26

6. Nodo de sensado **28**
6.1. Lua-RTOS . . . 28

7. Implementación contador de dispositivos **30**
7.1. Implementación de la función radar en Lua-RTOS . . . 31
7.1.1. Capa 3 . . . 31
7.1.2. Capa 2 . . . 32
7.1.3. Capa 1 . . . 33
7.2. Comunicación con infraestructura . . . 33
7.2.1. Configuración LoRa en LuaRTOS . . . 34
7.3. Ejecutar contador de dispositivos . . . 35
7.4. Concurrencia . . . 36

8. Visualización de los Datos **38**
8.1. Integración HTTP con Lora Server . . . 38
8.2. Persistencia . . . 38
8.3. Interfaz con el usuario . . . 39
8.3.1. Información en tiempo real . . . 39
8.3.2. Estadísticas . . . 40
8.3.3. Búsqueda por MAC . . . 41
8.3.4. Traza . . . 42
8.3.5. Mi Traza . . . 43
8.3.6. Fabricantes . . . 43

9. Evaluación de solución **45**
9.1. Prueba de estrés . . . 45
9.2. Prueba funcionamiento y validación . . . 46
9.3. Ingenieria deMuestra 2019 . . . 48

10. Conclusiones **53**

11. Trabajos Futuros **54**

12. Bibliografía utilizada **55**

13. Apéndices **58**
13.1. Instalación postgreSQL . . . 58
13.2. Instalación Apache . . . 58
13.3. Instalación PHP . . . 58
13.4. LoraServer . . . 58
13.4.1. LoraServer Bridge . . . 59
13.4.2. Instalación de LoraServer . . . 59
13.4.3. LoraServer App Server . . . 60
13.4.4. Configurar LoraServer App . . . 61
13.5. Instalación de Lua-RTOS en Sparkfun . . . 61
13.6. Activación de The Things Gateway . . . 63
13.7. Parametrización Gateway SPX-14893 . . . 63
13.8. Puntos de control . . . 64

1. Introducción

1.1. Caso de uso

Se definió como caso de uso implementar una aplicación que, mediante el uso de sensores, permita contar la cantidad de dispositivos inalámbricos (por ejemplo celulares) que se encuentran en un área determinada y disponibilizar estos datos. Los sensores se tratan de dispositivos que actúan como monitores Wi-Fi de la actividad inalámbrica que sucede a su alcance.

Para ello se utilizan las interfaces Wi-Fi de los sensores a fin de contar pasivamente la cantidad de dispositivos en el área. Esto se realiza analizando los paquetes que forman parte de la comunicación entre los demás dispositivos inalámbricos y los Access Point a su alcance. Esta información será enviada mediante el protocolo LoRaWAN a un servidor, a fin de generar una base de datos para su análisis y consulta.

Como aplicaciones del caso de uso definido se puede analizar la sobrecarga de un servicio de atención al cliente, de una cafetería por ejemplo, para identificar el mejor momento para ir a comprar y esperar lo mínimo posible.

También es útil a nivel de infraestructura, y dado que los nodos son alimentados a batería, pueden ser utilizados para detectar la falta de disponibilidad de las redes Wi-Fi de la zona, o para generar un mapa de calor y mejorar la calidad del servicio.

Esta solución puede ser utilizada a su vez para seguridad física, ya que si un sensor en un área restringida detecta un dispositivo desconocido, podría tratarse potencialmente de un intruso.

Para realizar el corriente trabajo fue necesario investigar y entender el funcionamiento de los protocolos Wi-Fi, LoRa y LoRaWAN, así como del sistema embebido Lua-RTOS, por lo que este trabajo no solo contribuye a la resolución del caso de uso definido, sino también con el estudio de estas nuevas tecnologías.

En el Capítulo 2 se estudia el protocolo Wi-Fi y el método definido para identificar al dispositivo en la comunicación inalámbrica entre éste y la infraestructura. En el Capítulo 3 se analizan los protocolos LoRa y LoRaWAN, para luego en el Capítulo 4 presentar un servidor de LoRaWAN comunitario y otro privado, con los beneficios y desventajas que tiene cada uno. En el Capítulo 5 se analiza un gateway LoRaWAN multicanal propietario y se explica cómo configurar uno de un canal. En el Capítulo 6 se explica como realizar la instalación y configuración necesaria para el funcionamiento del nodo de sensado. En el Capítulo 7 se explica cómo implementar el contador propuesto en el nodo. En el Capítulo 8 se presenta la capa de aplicación propuesta y su integración a el resto de la solución. El Capítulo 9 refiere a las pruebas realizadas. Por último en el Capítulo 10 se presentan las conclusiones y en el Capítulo 11 los trabajos futuros identificados.

1.2. Estado del Arte

El término Internet de las Cosas (IoT por sus siglas en inglés), hace referencia a sistemas de dispositivos físicos que transfieren datos a través de redes sin la intervención humana.

Gartner pronostica que el mercado de IoT crecerá a 5.8 mil millones de dispositivos en 2020 sólo para aplicaciones de empresas y automotores [1]. Para dar solución a la variedad

de demandas que requieren estos productos, se vienen desarrollando distintas tecnologías de comunicación inalámbrica dependiendo de la red que es necesaria formar entre ellos.

Las redes de comunicación para estos dispositivos se pueden dividir en 3 tipos. Las redes *Local Area Network* (LAN) son utilizadas para comunicaciones de corto alcance, como por ejemplo para comunicar dispositivos personales. En este caso se cuenta con protocolos inalámbricos ampliamente adoptados como Wi-Fi y *Bluetooth Low Energy* (BLE).

Las redes con tecnología celular son ideales para aplicaciones que necesitan alto ancho de banda y gran alcance, pero se tratan de tecnologías propietarias. Un ejemplo de aplicación de estas redes puede ser la videollamada entre dispositivos móviles.

Por otro lado, las redes *Low Power Wide Area Network* (LPWAN) ofrecen larga duración de batería, pudiendo enviar inalámbricamente pequeñas cantidades de datos a largas distancias. Podría ser un ejemplo de estas redes sensores a batería que miden la temperatura del ambiente y la envían a un servidor a varios kilómetros de distancia, sin necesidad de recargar las baterías por meses.

En la Figura 1, presentada por LoRa Alliance como parte de su introducción a la tecnología LoRaWAN [2], se muestran los principales beneficios y desventajas de cada tecnología, así como los protocolos más conocidos.

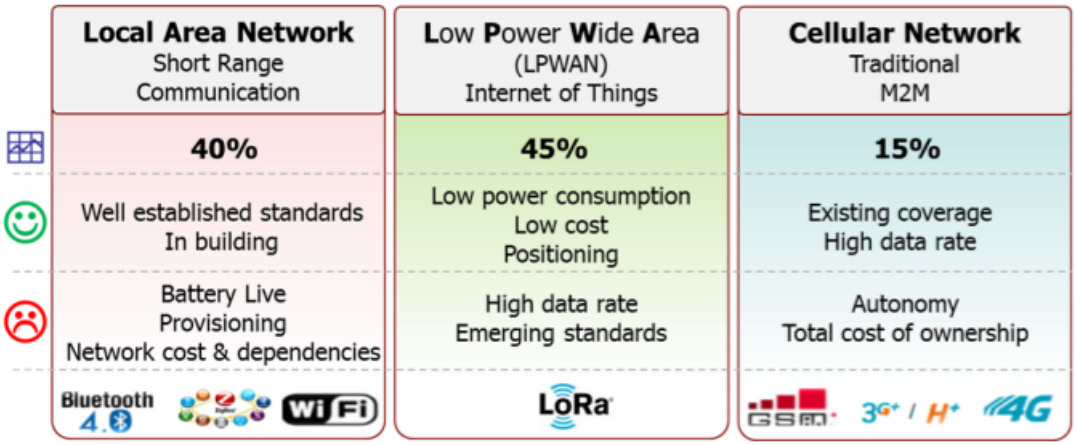

Figura 1: Protocolos de comunicación para dispositivos IoT.

Este trabajo se centrará en las redes LPWAN, particularmente en los protocolos LoRa para la capa física y LoRaWAN para las capas superiores. Se decide la utilización de estas tecnologías ya que será necesario utilizar sensores distribuidos, y con estas tecnologías no será necesario contratar servicios privados como con las redes celulares. La principal desventaja de esta tecnología ante redes como NB-IoT, es la alta latencia, o sea el tiempo que puede demorar en enviarse un paquete en la red. Esta situación se da sobre todo para los mensajes que debe recibir el sensor, pero ésta no es una propiedad necesaria para el caso de uso elegido.

1.3. Trabajos relacionados

La identificación de dispositivos se puede realizar en base a la recolección pasiva de las tramas Wi-Fi de tipo *probe request* [3]. En este trabajo, extraen de estas tramas: la dirección *MAC* que

identifica de forma única al dispositivo, el indicador de fuerza de señal recibida *RSSI*, que es una escala de referencia para medir el nivel de potencia de las señales, y el registro de tiempo *timestamp*, a fin de generar una base de datos en cada nodo de sensado. Los sensores se tratan de dispositivos Raspberry Pi 3, cada uno con una tarjeta de red Wi-Fi externa. Dichas tramas son las que envían los dispositivos para detectar las redes inalámbricas a su alcance. Con dicha información, los autores del paper plantean formas de ubicar a los usuarios, detectar perfiles de usuario, y por último clasificar dispositivos entre celulares y notebook. Particularmente en este trabajo se presenta como generar distribuciones en el tiempo a fin de utilizarlas como datos de entrenamiento para algoritmos de aprendizaje supervisado. El aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento, para que a partir de ellos, contar con una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida.

Respecto a este trabajo, si bien se considera su costo bajo (U$S50 por dispositivo), la solución que será planteada se implementa por cerca de la mitad de costo, permitiendo a su vez el análisis en tiempo real sobre una base de datos centralizada. A su vez, se considerarán no solo las tramas de probe request, ya que éstas son esporádicas.

Otro estudio que se basa en la recolección pasiva de las tramas *probe request* [4], agrega la comunicación a Internet de los dispositivos de sensado (agregando una segunda tarjeta Wi-Fi USB), y un servidor para su análisis. Plantea el uso de dos algoritmos, uno para optimizar la información que se envía a la infraestructura a partir de analizar si el dispositivo se mueve, y otro para monitorear el flujo de las personas. Esto lo hace generando un vector de trayectoria para cada dispositivo identificado por su MAC, y cada componente de dicho vector es el dispositivo de sensado que lo detectó con el mayor RSSI. Esta solución tiene como limitante la necesidad de tener que crear una red inalámbrica con Access Points (AP) en las cercanías de cada nodo, y que el costo aumenta respecto a la solución anterior.

Otros estudios [5], estudian la movilidad de los dispositivos a partir de la información que provee la propia infraestructura de una organización. Particularmente en ese trabajo, con la información proveniente de la controladora Radius se generó un grafo bipartito de dispositivos y AP, a fin de aplicar análisis de redes y obtener información del comportamiento social de las personas que utilizan los dispositivos.

La técnica para poder detectar y capturar comunicaciones inalámbricas es la denominada *Sniffing*. Existe una amplia variedad de implementaciones para Wi-Fi, sin embargo para la tecnología que se seleccionó para los nodos (Lua-RTOS), no se encuentra disponible ningún código ni documentación.

A su vez, respecto a cómo identificar quién es el dispositivo en todos los tipos de trama de la comunicación Wi-Fi existe documentación teórica, como son el desarrollo y análisis del propio protocolo, pero no se encontraron ejemplos de implementación.

1.4. Arquitectura definida

Para el caso de uso se definió la arquitectura que se ilustra en la Figura 2. Los dispositivos inalámbricos son detectados por los nodos (cuadros verdes), tal como se explicará posteriormente en los Capítulos 2, y 6. Esta información será enviada mediante LoRa al gateway (cuadro naranja), configurándolo como se explica en el Capítulo 5.2.

Como lo que se desea es detectar dispositivos Wi-Fi y enviar el dato a una infraestructura privada, se utilizaron las placas Sparkfun también como gateway. Si bien se tiene la limitante de ser un gateway de un canal, como el nodo no va a recibir información de la infraestructura, ésta no será una limitante para el proyecto.

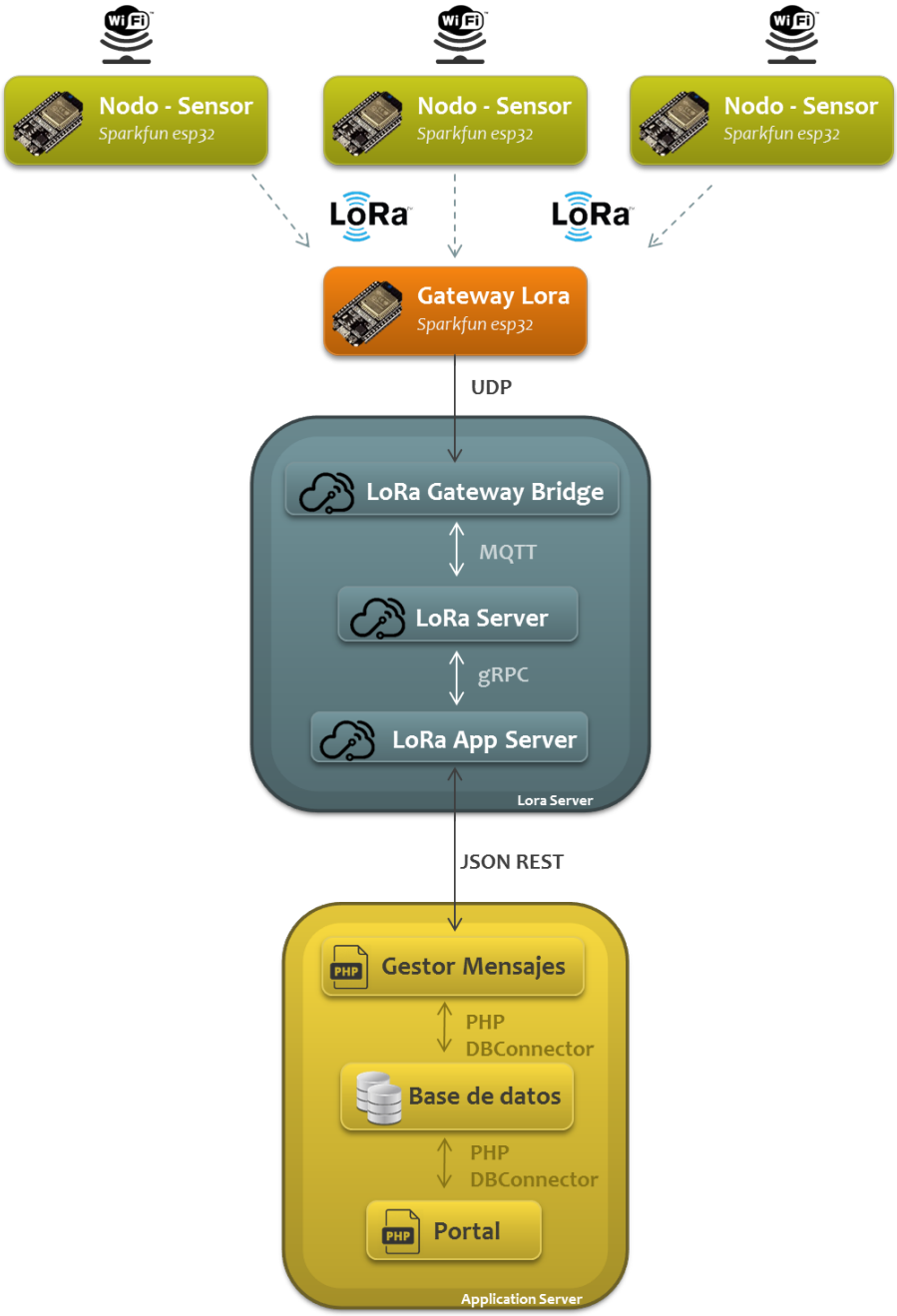

Figura 2: Arquitectura de la solución.

Luego la información es reenviada a una solución de Lora Server (cuadro gris), explicada en el Capítulo 4.2, para integrarla con una aplicación web (cuadro amarillo), detallada en el Capítulo 8.

2. Conteo de dispositivos Wi-Fi

Para contar la cantidad de dispositivos que detecta un sensor se utilizarán características propias del protocolo Wi-Fi, por ello es que se resumen a continuación las que fueron utilizadas en el corriente trabajo.

2.1. Arquitectura 802.11

Tal como se ilustra en la Figura 3, los dispositivos inalámbricos y el Access Point (AP), que presta el servicio de infraestructura, se agrupan en unidades denominadas *Basic Service Set* (BSS). En la Figura 3 hay 2 BSS, ambos conectados a un router que brindará la posibilidad de comunicación entre los dispositivos de los 2 BSS y también la conexión a Internet [6].

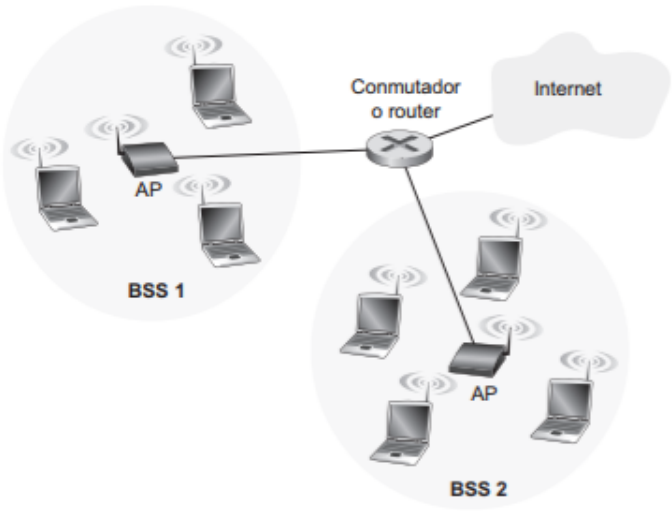

Figura 3: Arquitectura 802.11.

Tanto los dispositivos como los AP tienen direcciones MAC de 6 bytes para la interfaz de red 802.11, siendo la asignación de dichas direcciones administradas por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE sus siglas en inglés) y globalmente únicas. Se tratan teóricamente de direcciones únicas, ya que en todos los sistemas operativos hay métodos que permiten a las tarjetas de red identificarse con direcciones MAC distintas de la real.

Los dispositivos inalámbricos tienen dos formas de descubrir los AP que se encuentran al alcance de su antena, que se denominan exploración pasiva y exploración activa. Los AP transmiten cada determinado lapso un mensaje denominado *beacon*, que contiene información sobre las redes que disponibiliza. La exploración pasiva se trata de escuchar estos mensajes. Por otro lado, en la exploración activa el dispositivo emite una trama denominada *probe*, con la finalidad que los AP que están al alcance de él le respondan al recibir la trama.

2.2. Trama 802.11

En la Figura 4 se presentan los campos de la trama 802.11, expandiendo los más importantes que integran el campo de control de trama. A continuación se explicarán los campos que serán utilizados para este trabajo.

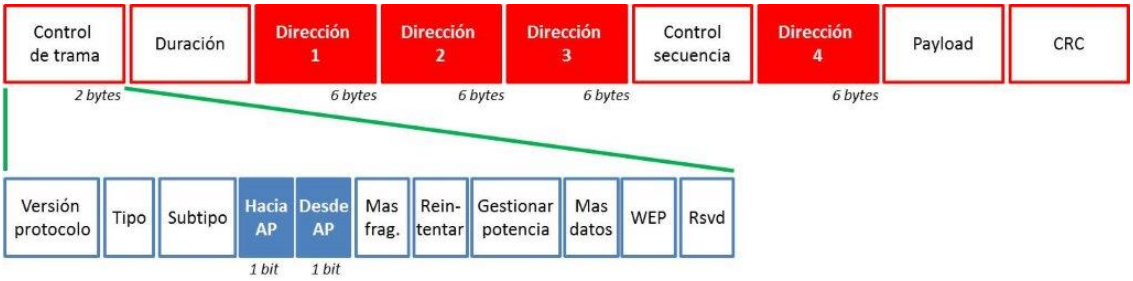

Figura 4: Campos utilizados de trama 802.11.

Campos de dirección MAC

La trama 802.11 tiene 4 direcciones MAC:

- La **Dirección 1** contiene la dirección del destinatario de la trama.
- La **Dirección 2** refiere a la dirección del dispositivo que transmite de la trama.
- La **Dirección 3** contiene la dirección MAC de la interfaz del router a la que está conectada la BSS de la subred del dispositivo (BSSID).
- La **Dirección 4** se utiliza cuando los AP se envían tramas entre sí en modo ad hoc, es decir cuando las envían directamente sin pasar por un router.

Campos de flujo de mensajes

Los campos *hacia AP* y *desde AP* se utilizan para conocer la dirección del envío de la trama. El significado cambia si se está utilizando el modo de infraestructura o ad hoc, y en este caso si quien envía es un dispositivo o un AP. A continuación se detalla el significado de los campos dirección según el valor de estos campos de dirección.

- haciaAP y desdeAP son 0: en este caso se trata de una transmisión entre dispositivos en una misma BSS
  - Dirección 1: MAC destino
  - Dirección 2: MAC origen
  - Dirección 3: BSSID
- haciaAP es 1 y desdeAP es 0: trama que se envía a la infraestructura
  - Dirección 1: BSSID
  - Dirección 2: MAC origen
  - Dirección 3: MAC destino
- haciaAP es 0 y desdeAP es 1: trama que se recibe a la infraestructura
  - Dirección 1: MAC destino
  - Dirección 2: BSSID
  - Dirección 3: MAC origen

- haciaAP y desdeAP son 1: trama enviada a otro dispositivo en otro BSS, siendo WDS la transmisión entre APs
    - Dirección 1: MAC AP receptor
    - Dirección 2: MAC AP transmisor
    - Dirección 3: MAC destino
    - Dirección 4: MAC origen

En la Figura 5 se presenta una trama 802.11 enviada desde un celular a un AP. Se puede apreciar que la bandera haciaAP es 1 y desdeAP es 0, y las 3 direcciones MAC. Esta captura fue realizada mediante el software Wireshark.

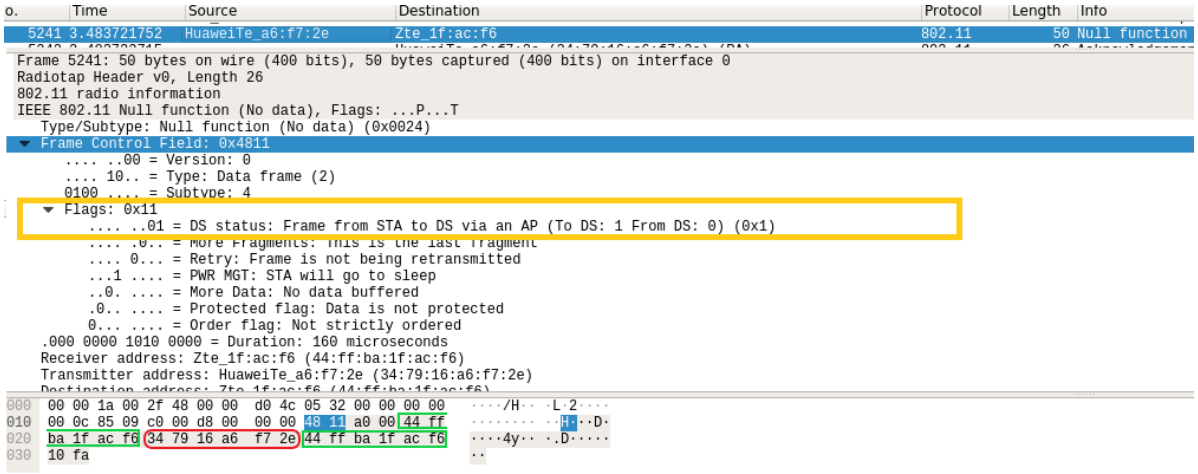

Figura 5: Captura de trama 802.11 enviada desde un dispositivo inalámbrico a un AP.

2.3. Modo promiscuo de interfaz inalámbrica

Se trata de un modo que poseen las interfaces de red que permite procesar no solo las tramas para los cuales la interfaz es la destinataria, sino todas las tramas con otro destinatario que recibe por estar en la misma red, y normalmente desecharía.

No todos los dispositivos permiten colocarse en este modo, y muchos sistemas operativos requieren permisos especiales para habilitarlo.

2.4. Anonimización de dirección MAC

A nivel de seguridad de dispositivos inalámbricos se está trabajando para poder anoniminizar lo máximo posible la actividad de los dispositivos, a fin de evitar de que pueda ser creado un historial de actividad de un dispositivo sólo por conocer su dirección MAC. Es por ello que los proveedores están trabajando en la anonimización de la dirección MAC.

2.4.1. Android

A partir de Android 8.0, los dispositivos Android usan direcciones MAC aleatorias para la exploración activa de redes. En Android 9, puede habilitar una opción de desarrollador (está deshabilitada de manera predeterminada) para que el dispositivo use una dirección MAC aleatoria cuando se conecte a una red Wi-Fi [7].

En Android 10, la anonimización de la MAC está habilitada de manera predeterminada para el modo cliente, SoftAp y Wi-Fi Direct.

2.4.2. iOS

Desde iOS 8, se previó también que se anonimice la dirección MAC para la exploración activa de redes inalámbricas.[8].

2.5. Consideraciones Éticas

Se programaron los sensores para que trabajen en dos modos, en uno sólo identifica direcciones MAC distintas para contabilizarlas y persistir únicamente el total, y en el otro modo persiste la lista completa de direcciones. Esta sección hace referencia a cuando se decide recolectar la lista de direcciones MAC.

Para esta recolección, se colocan los sensores en estado promiscuo a fin de recolectar pasivamente información. Vale aclarar que la información que se obtendrá son las direcciones MAC de las tramas, no siendo estos campos cifrados. Si bien el resto de los campos se descartan, vale aclarar igual que el único campo que se puede encontrar cifrado es el payload.

En ningún momento se identifica personas sino dispositivos, y tampoco se tendría forma sólo con esta información, de mapear dispositivos a personas. Información de tipo personal podría encontrarse en las capas superiores del modelo OSI, no en las tramas 802.11.

Se pide utilizar el modo de recolección de MAC solo en caso de estar seguro de cumplir con la normativa vigente, en caso contrario esta solución solo deberá ser utilizada con la configuración para recolectar totales y no direcciones. De esta forma la dirección MAC no será almacenada.

Se quiere aclarar que la finalidad de este trabajo no es generar daño de ninguna manera, sino generar conocimiento sobre las distintas tecnologías y como aplicarlas para un caso de uso.

A continuación se presentan dos reglamentaciones que podrían impactar en la recolección de estos datos: Ley 18.331 de Uruguay y el Reglamento General de Protección de Datos (GDPR).

Ley 18.331

La Ley 18.331 - Ley de protección de Datos Personales - acción de Habeas Data - promulgada en el año 2008 prevee las precauciones que deben tomarse para mantener y manipular datos personales. Allí se define dato personal como “información de cualquier tipo referida a personas físicas o jurídicas determinadas o determinables.” [9].

GDPR

El Reglamento General de Protección de Datos (GDPR por sus siglas en inglés) es el reglamento europeo relativo a la protección de las personas físicas en lo que respecta al tratamiento

de sus datos personales y a la libre circulación de estos datos. Entró en vigor el 25 de mayo de 2016 y fue de aplicación el 25 de mayo de 2018 [10].

Si bien es una normativa a nivel de la Unión Europea, este reglamento es extendido para empresas extranjeras que operen con residentes europeos. A su vez, se está utilizando como marco de referencia en todas partes del mundo, por lo que se decidió hacer referencia a este reglamento.

Tal como se expresa anteriormente, una dirección MAC no se trata de un dato de una persona determinable, por lo que no habría incumplimiento de la Ley. Respecto al reglamento europeo, este es más específico y taxativo que la Ley uruguaya. De hecho en una lista de ejemplos de que es procesar información personal colocan la dirección MAC como un dato personal [11]. Según varios blogs esto se debe a que definen la MAC como un dato "linkeable" a una persona, y si bien es necesario sumar otra fuente de información para identificar a la persona, etiquetó esta información como dato personal. Por lo que en estos casos se recomienda leer completamente esta reglamentación a fin de verificar que es posible cumplirla, y sino utilizar esta solución en el modo de recolección de total de dispositivos.

3. Ecosistema LoRaWAN

3.1. Protocolo LoRa

LoRa es un método de comunicación inalámbrica por radio frecuencia. Trabaja en 3 rangos de frecuencias ISM (Industrial Scientific Medical): 915MHz para América, 868MHz para Europa y 433MHz para Asia. Son bandas menores a las frecuencias de GHz en las que no se necesitan licencias para emitir. Como desventaja es que se tendrá interferencia, lo que podría generar que el mensaje no sea recibido.

Se trata de un protocolo LPWAN para la transmisión entre dispositivos distribuidos. Hay 2 tipos de dispositivos en la comunicación, por un lado está el dispositivo que genera un mensaje y lo transmite, en general denominado nodo o sensor, y por otro el gateway que se encuentra conectado a la infraestructura de red (generalmente mediante IP) y es el encargado de recibir el mensaje y reenviarlo a la infraestructura. Es un protocolo de capa física propietario de la empresa Semtech.

Respecto a la característica de *Low Power*, se diseñó este protocolo para trabajar con dispositivos que funcionen a batería, con la idea de que la carga dure años. Se trata de un protocolo *Wide Area* ya que se prevee que su alcance pueda llagar a kilómetros de distancia, y no de unos metros como las redes Wi-Fi.

LoRa es ideal para sensores de bajo consumo distribuidos lejos del gateway (hasta kilómetros), para sensar datos puntuales que no varíen mucho en el tiempo. A su vez, es un protocolo pensado para recolectar información, por lo que favorece el envío de información desde el sensor al gateway sobre la comunicación inversa.

En la Figura 6 se presenta un cuadro comparativo de LoRa con distintos protocolos de comunicación inalámbrica respecto a rango de alcance y ancho de banda. [14]

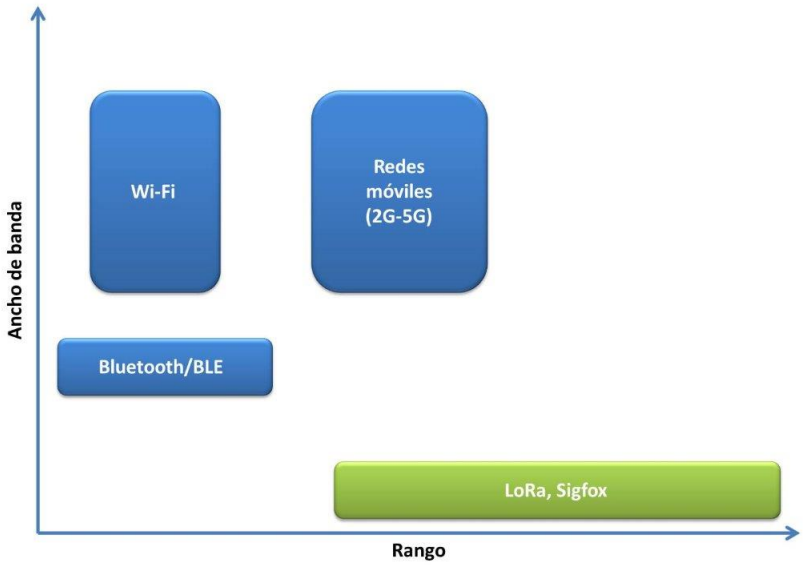

Figura 6: Rango y ancho de banda según tecnología.

3.1.1. Ancho de Banda

Para poder cumplir con estas dos características descriptas, es decir cubrir áreas amplias pero pudiendo utilizar baterías que duren, es necesario afectar el *Bandwidth* (ancho de banda).

Llevado al espectro de transmisión de los dispositivos, el bandwidth es el rango de frecuencias alrededor de una frecuencia seleccionada que será utilizado en una transmisión. Por ejemplo para el rango de América de LoRaWAN, los bandwidth fijados para mensajes de subida son de 125kHz y 500kHz, y para los mensajes de bajada 500kHz. A menor bandwidth, mayor duración de batería y menor posibilidad de interferencia, pero menor alcance y menor información que puede enviarse (a 500KHz se puede enviar 4 veces más bytes por transmisión que a 125kHz).

3.1.2. Modulación CSS

Modulación se refiere a cómo codificar información para transmitirla por radio. La modulación LoRa se basa en la tecnología de espectro extendido Chirp , denominada Chirp Spread Spectrum. Los *Spread Spectrum* son métodos por los cuales una señal es deliberadamente propagada en un rango de frecuencias.

La modulación de frecuencias Chirp utiliza amplitud fija. Puede utilizar todo el espectro de frecuencia asignado para transmitir señales, produciendo una señal que recorre el canal. Un *chirp*, generalmente llamado señal de barrido, puede aumentar (*up chirp*) o disminuir (*down chirp*) en el tiempo.

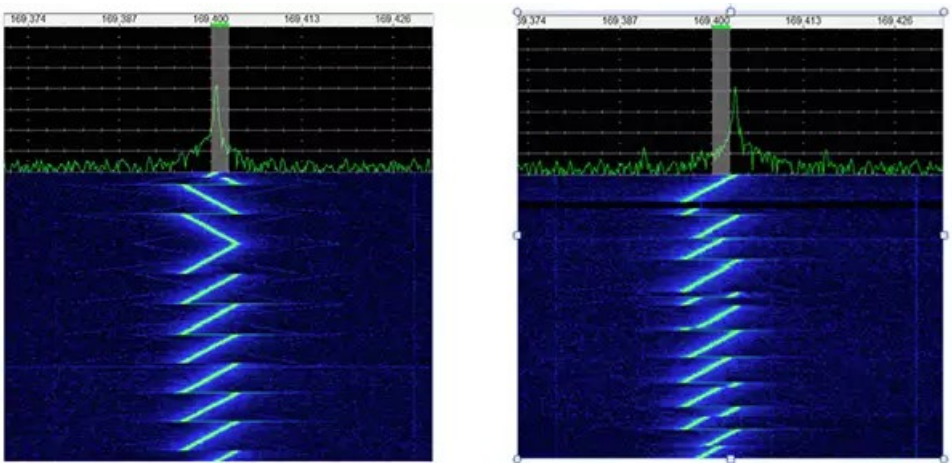

Figura 7: Espectrograma mensaje LoRa [17].

Tan como se ve en la Figura 7 (izquierda), un mensaje comienza con un preámbulo que se trata de una serie de upchirps seguidos de 2 downchirps. Desde ese momento se envía el mensaje codificado, tal como se aprecia en Figura 7 (derecha).

Los chirps se desplazan cíclicamente, y son los saltos de frecuencia los que determinan cómo se codifican los datos en los chirps.

3.1.3. Spreading Factor

El factor de propagación *Spreading Factor* es, a grandes rasgos, la duración del chirp. LoRa opera con Spreanding Factor de 7 a 12. SF7 es el tiempo más corto en el aire, SF12 será el más largo.

Un *símbolo* representa uno o más bits de datos. El Spreading Factor es la cantidad de bits en los que se puede codificar un símbolo. Por ejemplo si se quiere representar el número 127 con SF 7, el símbolo será "1111111". Este símbolo luego será codificado en un *up-chirp* para enviarlo por radio.

Hacer que el *Spreading Factor* sea un paso más bajo, permite enviar el doble de bytes al mismo tiempo. Pero disminuirlo dificulta que el gateway reciba una transmisión, ya que será más sensible al ruido.

3.1.4. Link Budget

Para conocer la calidad de una transmisión de radio se calcula el *Link Budget*. Es la suma de 4 medidas: la potencia de la transmisión, la sensibilidad del receptor, la ganancia de la antena, y la pérdida de señal en el camino [15]. A continuación se explicará brevemente a que se refiere cada medida, y se enfatizará en negrita los parámetros de LoRa que influyen en cada una.

La sensibilidad del receptor depende del **bandwidth**, del factor de ruido *Noise Factor* (NF), y del factor *Signal Noise Ratio* (SNR), que indica qué tan fuerte tiene que ser la señal respecto al ruido para poder ser demodulada. El SNR depende del **Spreading Factor** (SF), cuanto mayor el SF menor cantidad de dB de SNR se necesitan para decodificar la información. Esto genera un *Data Rate* menor, y menor duración de la batería pero se consigue un mayor rango. En la banda de América, el Data Rate puede llegar a 22kpbs.

Como se puede observar en la Figura 8, en un camino sin interferencias, la potencia de la señal va siendo modificada desde el transmisor al receptor por distintos factores. En particular la pérdida de señal en el aire, es un valor teórico ya que en el mundo real hay muchos obstáculos, y se calcula a partir de la distancia entre el emisor y el receptor y la **frecuencia**. La potencia que recibe el receptor debe estar por encima de la sensibilidad del receptor para poder demodular la señal.

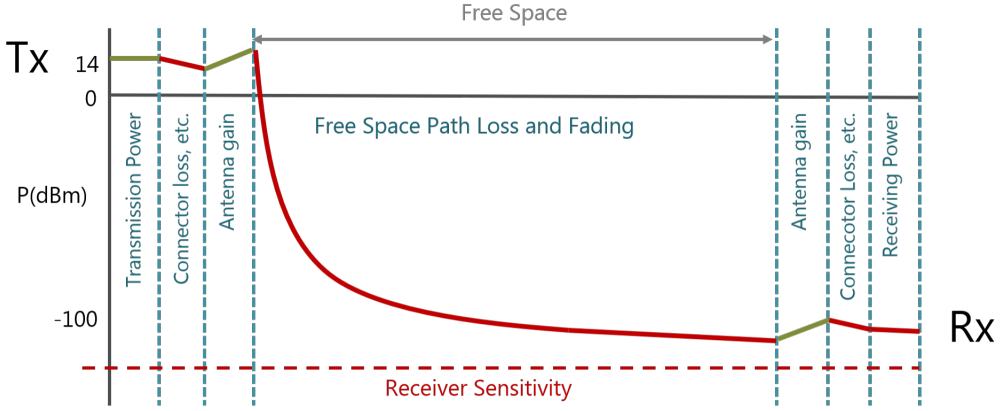

Figura 8: Transmisión en un camino sin interferencias.

Según consta en la página de Semtech, el Link Budget máximo de sus dispositivos es de 154dB. Verificando en una página de calculadora de Link Budget [20], en condiciones ideales, podría transmitirse mediante LoRa a 1300kms. En la práctica, según publica The Things Network en su sitio [16], el nuevo record de distancia entre un dispositivo y gateway es de 766Kms.

La potencia máxima usada para transmitir en la banda de América es de 20dBm. Cuanto mayor la potencia, mayor alcance tendrá, pero menos durará la batería.

3.1.5. Duty Cycle

Se define el *Duty Cycle* como la fracción de tiempo que un dispositivo está ocupado transmitiendo, y para ello hay que considerar el tiempo que el dispositivo estará transmitiendo en cada uno de los canales.

Por ejemplo, si un dispositivo está transmitiendo en un canal por 3 unidades de tiempo cada 10, tendrá un Duty Cycle del 30 %. Pero si está transmitiendo 3 unidades de tiempo cada 10, en 3 canales distintos como se observa en la Figura 9, tendrá un Duty Cycle de 90 %.

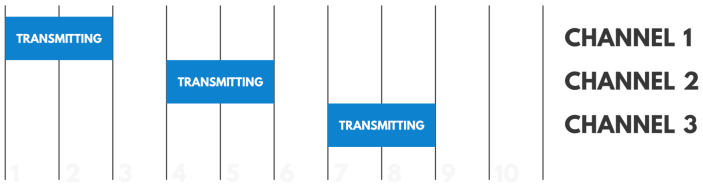

Figura 9: Duty Cycle multicanal.

Para la banda de LoRaWAN definida para Uruguay dicha fracción no está limitada, aunque la FCC reguló un tiempo de permanencia máximo de 400 ms por canal. Esto limita el tamaño máximo de payload que puede enviarse para cada Spreading Factor, que se discutirá en el próximo capítulo. Por otro lado, si se considerara la misma solución en la banda europea, hay que tener en cuenta que el duty cycle no puede superar el 1 % en la mayoría de los canales [21].

Más allá de las regulaciones que existan en cada región, el *Duty Cycle* es un parámetro a tener en cuenta para calcular la cantidad máxima de dispositivos que puede atender un gateway, ya que por ejemplo un gateway de 8 canales podrá atender a 800 dispositivos si éstos demoran 1 unidad de tiempo cada 100 en enviar información (1 % de duty cycle).

3.2. LoRaWAN

LoRaWAN es un protocolo de red abierto que usa el protocolo propietario LoRa para la capa física. Fue diseñado para conectar inalámbricamente a Internet dispositivos operados a batería.

En 2015 se creó una asociación abierta sin fines de lucro: *LoRa Alliance*, dedicada a la estandarización de redes LPWAN y la promoción global del estándar abierto LoRaWAN. La misión de LoRa Alliance, que ya posee más de 500 miembros, es apoyar y promover la adopción global del estándar LoRaWAN asegurando la interoperabilidad de todos los productos y tecnologías LoRaWAN, permitiendo que Internet de las Cosas ofrezca un futuro sostenible [12].

Esta estandarización es importante para conectar a una red múltiples dispositivos IoT, construidos por distintos proveedores, y poder conectar esta red IoT a Internet. Cabe destacar que estas redes pueden escalar a muchos más dispositivos que una red TCP/IP.

LoRaWAN define la arquitectura de la red, la estructura de los paquetes de datos, como los paquetes son procesados en el servidor, cómo debe armarse el paquete, y cómo debe cifrarse.

Tal como se muestra en la Figura 10, LoRaWAN tiene 3 capas: una de aplicación (capas color verde), una de LoRa MAC (capas color celeste), y una capa física (capa naranja y gris). La capa MAC construye una trama MAC utilizando el payload MAC que contiene un encabezado de trama (con las direcciones de origen y de destino más un contador de trama), un puerto de trama y el payload de trama (que contiene los datos de la aplicación). El puerto de trama se usa para determinar si la trama contiene sólo comandos MAC (cuando se establece en 0) o datos específicos de la aplicación. Finalmente, la capa PHY usa la trama MAC como payload PHY y construye la trama PHY, después de insertar el preámbulo, el encabezado PHY y el CRC.

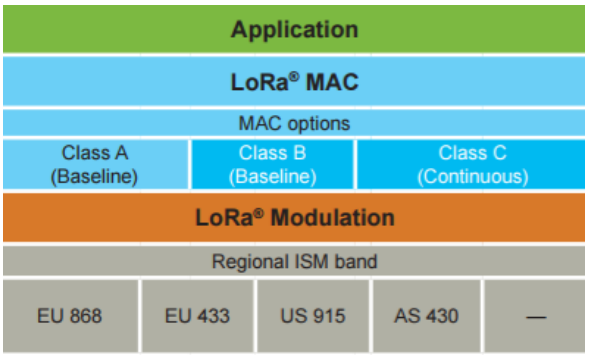

Figura 10: Stack LoRaWAN [2].

Según la región en la que se encuentren los dispositivos, el protocolo indica los valores de los parámetros. Para el caso de Uruguay, si bien se encuentra en la Región 2 de la banda ISM (902MHz a 928MHz), no es posible utilizarla completamente. Dado que las frecuencias entre 911MHz y 915MHz se encuentran asignadas a ANTEL para sistemas IMT (Telecomunicaciones Móviles Internacionales) [13], se utilizan los parámetros fijados para la banda de Australia, que es la AU915-928MHz.

Canales para transmisión

Cada rango de frecuencias se divide en canales estipulados por protocolo, por lo que un canal no es otra cosa que una frecuencia específica. Particularmente para la banda AU915, se establecen 64+8+8 canales (Figura 11). Los primeros 64 se tratan de canales de subida de 125KHz, en incrementos de 200KHz (de 915.2MHz a 927.8MHz). Los siguientes 8 refieren a canales de subida de 500KHz en incrementos de 1.6MHz (de 915.9MHz a 927.1MHz). Los otros 8 son los canales de bajada de 500KHz en incrementos de 600KHz (de 923.3MHz a 927.5MHz).

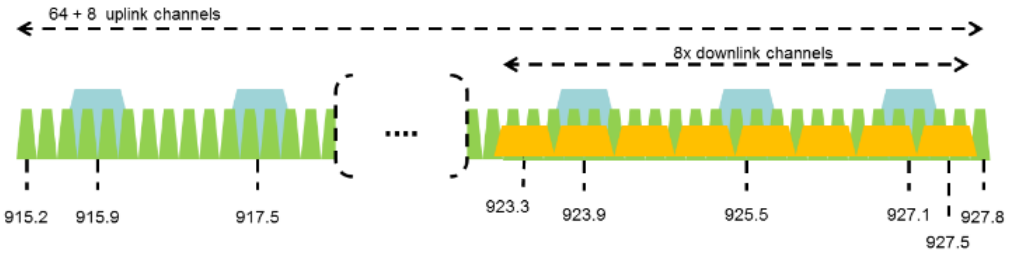

Figura 11: Canales banda AU915-928 [19].

Corrección de errores

LoRaWAN maneja *coding rate*, que refiere a cuántos bits de corrección de errores se agregan cada 4 bits de datos (4/5, 4/6, 4/7, 4/8). Contar con más bits de corrección permite mandar mensajes más lejos, porque el mensaje llega con errores pero el receptor puede corregir mayor cantidad de los mismos, aunque agota más rápido la batería. El protocolo sugiere utilizar 4/5.

Data Rate

La velocidad de datos depende del ancho de banda utilizado y el Spreading Factor.

Según define el estándar [19], para la región AU915-928 están definidos los DR que figuran en el Cuadro 1, así como el máximo payload para cumplir con el tiempo máximo de transmisión de 400ms. Los DR del 8 al 13 están reservados para futuras aplicaciones.

Cuadro 1: Tabla Data rate.

Data Rate	Configuración	bits/sec	Max Payload (Bytes)
DR0	SF12/125kHz	250	N/A
DR1	SF11/125kHz	440	N/A
DR2	SF10/125kHz	980	19
DR3	SF9/125kHz	1760	61
DR4	SF8/125kHz	3125	133
DR5	SF7/125kHz	5470	250
DR6	SF8/500kHz	12500	250
DR7	RFU		
DR8	SF12/500kHz	980	41
DR9	SF11/500kHz	1760	117
DR10	SF10/500kHz	3900	230
DR11	SF9/500kHz	7000	230
DR12	SF8/500kHz	12500	230
DR13	SF7/500kHz	21900	230
DR14	RFU		
DR15	Definida para LinkADRReq		

Seguridad - Claves de Cifrado

Es importante para cualquier red LPWAN incorporar seguridad. LoRaWAN utiliza dos capas de seguridad: una para la red y otra para la aplicación. En la Figura 12, que se encuentra en el sitio de Lora Alliance [18], se puede observar dónde se cifra y descifra cada capa.

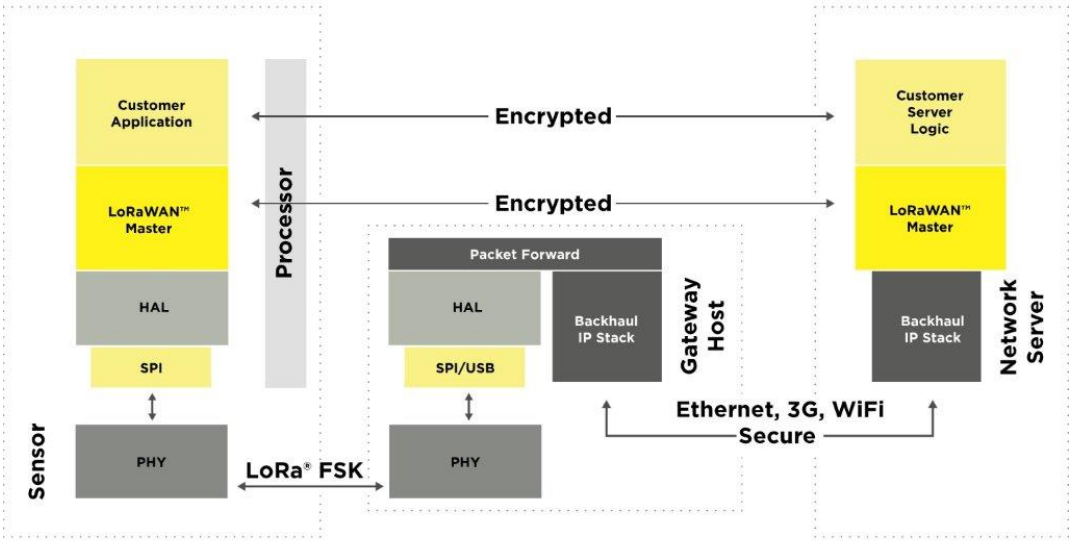

Figura 12: Seguridad LoRaWAN.

La seguridad de la red garantiza la autenticidad del nodo en la red mientras que la capa de seguridad de la aplicación garantiza que el operador de red no tenga acceso a datos de la aplicación del usuario final. El cifrado AES se utiliza con el intercambio de claves utilizando un identificador IEEE EUI64.

La clave utilizada para la seguridad a nivel de sesión de red (*Network Session Key*) es única de 128 bits compartida entre el dispositivo final y el servidor de red. Esta clave es única para nodo y compartida entre el nodo y el servidor.

La clave seguridad de la aplicación es única de 128 bits (*Aplication Session Key*) compartida de extremo a extremo a nivel de aplicación. También es única por nodo y es la utilizada para cifrar y descifrar los payloads de aplicación.

Antes de iniciar la comunicación con un nuevo nodo en una red LoRaWAN es necesario activarlo mediante uno de los 2 métodos disponibles: activación por personalización (*ABP - Activation by Personalization*), o activación por aire (*OTAA - Over-the-Air Activation*).

Si se elige ABP, hay que escribir las 2 claves de sesión tanto en el nodo como en el servidor.

Para OTAA en lugar de tener que configurar 2 claves, se manejan 3: un identificador de 64 bits **DevEUI** , **AppEUI** de 64 bits y **AppKey** de 128 bits, utilizadas para obtener las claves necesarias para el cifrado al momento de la activación. Como las claves de sesión se negocian dinámicamente, OTAA permite cambiar la clave de los dispositivos si es necesario.

Clases de Dispositivos LoRa

Se definen 3 clases de dispositivos, dependiendo de las aplicaciones y requisitos, ponderando entre latencia de comunicación para los mensajes de bajada del nodo y la duración de la batería.

Los dispositivos de **Clase A**, son sensores a batería, por lo que se busca que sean los más eficientes respecto al consumo de energía. En estos dispositivos la comunicación siempre es iniciada por el nodo, en cualquier momento, y es completamente asíncrona dado que los mensajes de descarga *Downlink* solo pueden recibirse luego de una transmisión. Luego de cada transmisión se abren 2 ventanas cortas de tiempo para recibir mensajes que se encuentran almacenados en el servidor para ese nodo, con una variación de tiempo basada en la configuración más un tiempo aleatorio (tipo de protocolo ALOHA). Esta clase debe ser compatible por todos los dispositivos.

Los dispositivos de **Clase B**, son actuadores a batería donde se busca eficiencia de energía con un control de latencia en los *Downlink*. La comunicación se segmenta y se va sincronizando con un *Beacon*. Adicionalmente a las ventanas de recepción en un dispositivo de Clase A, los de Clase B abren ventanas de recepción adicionales a horas programadas desde la recepción del Beacon. De esta forma se sincroniza con el gateway y ambos sabrán cuando el dispositivo estará escuchando, siendo la latencia programable hasta 128 segundos.

Por último, los dispositivos de **Clase C**, son actuadores bidireccionales conectados continuamente a la corriente, por lo que pueden comunicarse en todo momento, no teniendo latencia para los mensajes de *Downlink*. Estos dispositivos tienen ventanas de recepción abiertas casi continuamente, solo se cierran cuando se transmiten mensajes (semiduplex).

Para los dispositivos alimentados por batería, es posible el cambio de modo temporal entre las clases A y C, y es útil para tareas intermitentes como las actualizaciones inalámbricas de firmware.

4. Servidor de LoRaWAN

Para administrar una red LoRaWAN es necesario un servidor. Sus principales funciones son administrar los gateway, mantener la seguridad en la comunicación con los nodos, y procesar los datos que recibe (eliminando mensajes duplicados y reenviando la información a aplicaciones). En este Capítulo se estudian dos proyectos de código abierto, el primero se trata de un servidor en la nube con un enfoque comunitario, y el segundo de un servidor local.

4.1. The Things Network (TTN)

The Things Network, integrante de Lora Alliance, provee una infraestructura global de LoRaWAN que permite conectar nodos a aplicaciones. Se basan en un enfoque de comunidad, donde personas comparten sus gateway a la plataforma y cualquiera que se registre puede utilizarlos para enviar su información. También cuentan con sensores y gateways bajo su nombre.

En la página disponibilizan un mapa que ubica a los gateways disponibles. A la fecha están disponibles 8277 gateways, pero en Uruguay solo hay 2, un en Fray Bentos y otro en Montevideo según se puede ver en la Figura 13

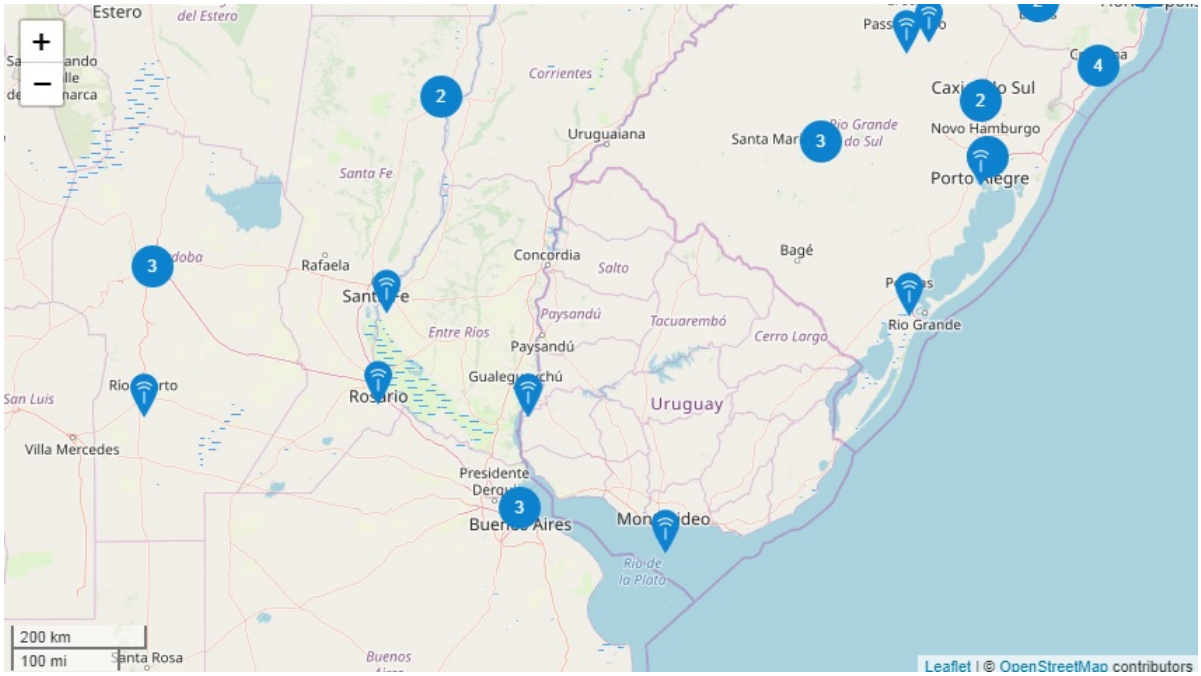

Figura 13: Gateways TTN Uruguay.

Tanto los diseños del gateway y del nodo de TTN, como el código sobre el que se ha escrito todo el backend de TTN ha sido disponibilizado como código abierto en su cuenta de GitHub, manteniendo el espíritu abierto y de comunidad con el que empezaron en 2015.

La principal limitante de esta infraestructura es su política de *Fair Policy*. En esta se establece que el tiempo de envío de los mensaje de tipo uplink no puede superar los 30 segundos por día por nodo, y los mensajes downlink no pueden ser más de 10 por día por nodo [22]. Según se expresa en el sitio de TTN, los 30 segundos equivalen a 20 mensajes en SF12 o 500 mensajes en SF7, para 10 bytes de payload.

Esta es una limitante de esta solución "comunitaria". Para el caso de querer escalar y utilizar infraestructura de forma privada, también poseen el proyecto The Things Industries.

4.1.1. Integración de TTN

TTN ofrece integración con otros sistemas por HTTP y MQTT (MQ Telemetry Transport se trata de un protocolo para el envío de mensajes mediante publicación/suscripción). También ofrece APIs en distintos lenguajes como: Go, Java, Node-RED y Node.js, con las cuales se puede construir una aplicación end-to-end.

Se realizaron pruebas utilizando la integración por HTTP a un endpoint de requestbin.com, de forma de no necesitar infraestructura propia y evaluar la solución. De esa forma se pueden observar en el endpoint los mensajes en formato JSON generados por TTN a partir del mensaje del nodo.

Entre los objetos del JSON se envía la dirección a la que se puede enviar los downlinks a la aplicación de TTN. Se probó enviando con el comando curl un mensaje por POST con los datos "dev_id" y "payload_raw", recibiendo el mensaje en el nodo.

4.2. LoraServer

Se seleccionó LoraServer como servidor LoRaWAN, de forma de contar con la infraestructura bajo control, con una solución de código abierto.

El proyecto LoRa Server proporciona componentes para construir redes LoRaWAN. Los 3 componentes centrales son: *LoRa Gateway Bridge*, *LoRa Server* y *LoRa App Server*.

Los componentes pueden intercambiarse para personalización o para integrar LoRa Server en las infraestructuras existentes. Todos los componentes están registrados bajo la licencia MIT y pueden usarse con fines comerciales.

LoRa Gateway Bridge

Es un servicio que convierte los paquetes que recibe del gateway ("packet-forward UDP") en paquetes procesables por el servidor (en formato JSON). Luego los envía como mensajes a través de MQTT al servidor. Este componente tiene como ventajas principales que el servidor sólo necesita conocer el tópico de MQTT para mandar downlinks, y que se puede generar una conexión segura con los gateways LoRa (usando MQTT sobre TLS)

LoRa Server

Es el servidor de red LoRaWAN. La responsabilidad del componente del servidor de red es el manejo de los uplinks recibidos por los gateways, la eliminación de aquellos recibidos por más de un gateway, y el envío de la confirmación de recepción (ACK). También es el responsable del manejo de la capa LoRaWAN MAC, realizando los chequeos de seguridad y gestionando los *Adaptive Data Rate* (ADR). Por último es el responsable de la programación del envío de los downlink a los nodos, decidiendo a qué gateway le enviará el mensaje para que se lo reenvie al nodo.

LoRa App Server

Es un servidor de aplicaciones LoRaWAN. Es responsable del inventario de los dispositivos de una infraestructura LoRaWAN, de agregar dispositivos y del manejo y cifrado de los payloads de las aplicaciones.

Ofrece una interfaz web donde se pueden gestionar los parámetros del sistema: usuarios, organizaciones, aplicaciones y dispositivos. Para la integración con servicios externos, ofrece una API RESTful y gRPC. Los datos del dispositivo pueden enviarse y/o recibirse a través de MQTT, HTTP y escribirse directamente en una base de datos de series temporales InfluxDB.

En la Figura 14 se presenta la arquitectura de estas componentes en una red LoRaWAN. Tal como lo presenta la figura, el componente de bridge puede estar instalado junto al gateway (si éste tiene un sistema operativo compatible), o se puede instalar un bridge independiente a los gateways (arriba a la derecha de la figura). Tal como se presentó en el Capítulo 1.4, para el corriente trabajo se instaló un servidor con los 3 componentes dentro de él. Si bien se siguieron las guías del proveedor, se detalla el procedimiento en el Apéndice 13.4.

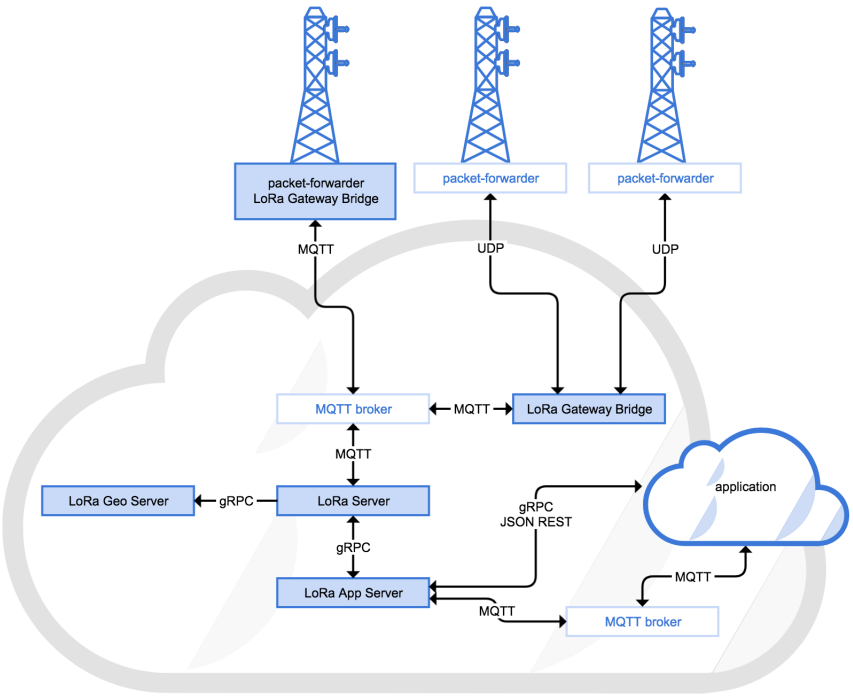

Figura 14: Arquitectura LoraServer.

5. Gateway LoRaWAN

Los gateway LoRaWAN se encargan de recibir los paquetes LoRa enviados por los sensores y reenviarlos como paquetes UDP si se utiliza un Lora Gateway Bridge o TCP si se utiliza MQTT, para que éste sea enviado finalmente al Lora Server.

A grandes rasgos se pueden identificar 2 tipos de gateway, los de un único canal, y los multicanal. Esto refiere a la cantidad de canales que estará utilizando el gateway a la vez. Por ejemplo en la banda AU915-928, con un gateway de 8 canales se pueden recibir los uplink en cualquiera de los canales definidos por protocolo para estos mensajes, tal como se explicó en la Sección 3.2. Por otro lado, los gateway de un único canal no cumplen totalmente lo estipulado por protocolo ya que no pueden recibir mensajes en todos los canales a la vez y poseen una utilidad más reducida, pero son más económicos. A continuación se presenta el análisis realizado para un gateway multicanal y para un gateway de un canal único, siendo este último el elegido para el despliegue del caso de uso.

5.1. The Things Gateway

Se probó el gateway ofrecido por TTN, que se trata de un gateway LoRaWAN de 8 canales configurado para integrarse directamente a la infraestructura de TTN.

Figura 15: The Things Gateway.

Si bien según el proveedor puede cubrir hasta 10.000 nodos en 10Kms, lo que se probó fue la integración con el resto de la infraestructura. Esto es la recepción de los uplinks de los nodos, y el envío de los downlinks a los nodos generados desde una aplicación.

Se pudo configurar rápida y satisfactoriamente el gateway siguiendo los pasos descriptos en el Capítulo 13.6. Cabe mencionar la solución a 2 inconvenientes que se tuvieron:

Cambiar de red Wi-Fi el gateway

Si se desea cambiar la red Wi-Fi a la que se conecta el gateway, esta configuración no se realiza en la consola de TTN sino que es necesario acceder a http://things-gateway.local/ desde la misma red.

Falla general en gateway

Durante las pruebas el gateway dejó de funcionar, presumiblemente luego de una actualización automática, ya que tenía esta opción configurada en la consola. Tal como se puede ver

en la Figura 16 ninguno de los led azules de estado prendían, así como tampoco el led interno verde.

Este inconveniente se solucionó actualizando el firmware manualmente. Para ello hay que formatear una memoria microSD en FAT32. Luego hay que crear una carpeta con nombre *update* y copiar los archivos *firmware.hex* y *checksums* de https://github.com/TheThingsProducts/gateway/tree/develop/firmware#installation.

Figura 16: Falla en gateway TTN.

Luego de desenchufar la corriente del gateway, insertar la memoria y volver a conectarlo, el firmware se actualizó y volvió a prender todos los leds tal como se observa en la Figura 17.

Figura 17: Gateway funcionando.

5.2. Gateway Sparkfun SPX-14893

Se utilizó una placa Sparkfun SPX-14893 para configurarla como gateway. Se trata de una placa con un microcontrolador programable ESP32, con interfaces Wi-Fi y Bluetooth, y un módulo LoRa RFM95W.

Si bien se tiene la limitante de ser un gateway de canal único, es decir utilizar una única frecuencia de transmisión a la vez, permite configurarlo para que reenvie los paquetes a un servidor privado, distinto al de TTN. Al solo estar recibiendo en un canal, un gateway con estas características no cumple totalmente con el estándar LoRaWAN.

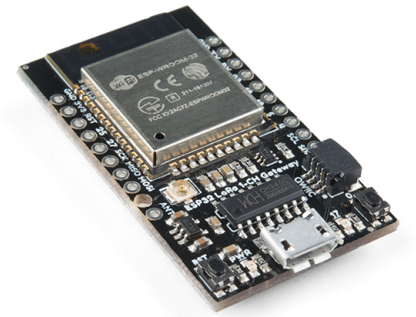

Figura 18: Sparkfun SPX-14893.

A dicha placa se le instaló el software de gateway sugerido por el proveedor en su sitio [24]. Según informan dicho software no tiene implementada el envío de downlinks a los nodos, ni soporte para dispositivos de clase B o C, pero para las funcionalidades previstas para el trabajo actual se consideró aceptable.

Al gateway de un canal se le configuraron 2 modos que pueden potenciar su utilidad. Se tratan de los modos CAD (Channel Activity Detection) y HOP.

- *CAD* es una función en el chip de LoRa que permite detectar el Spreading Factor de los mensajes recibidos, a partir de analizar sus cabezales. De esta forma se pueden recibir mensajes de cualquier Spreading Factor en un mismo canal, disminuyendo las colisiones. Como contra se tiene menor alcance, ya que el chip utiliza parámetros de RSSI para detectar si recibió una señal o ruido. Para usar esta función hay que usar más pins "DIO" en el chip, y habilitar "DIO1" para detectar este estado.
- *HOP* refiere, como su traducción del inglés, a "saltar" de canal. Para este modo se le agrega al modo CAD, la característica que el transceiver LoRa salte de un canal a otro. Esto emula en principio un gateway multicanal, con la desventaja que se perderán todos los paquetes enviados en los otros canales. Como el nodo no confirma si el gateway recepcionó el paquete, se desconocería qué información es la que se pierde.

Respecto a la conexión a Internet, si el LoRa Bridge Server se encuentra en la misma red que el gateway, la única funcionalidad que requerirá esta conexión será la sincronización de la

hora. Actualmente se tiene como problema o bug de la versión actual del software que si al prender el gateway no tiene conexión a Internet, éste no inicia. El dato de fecha y hora se usa para indicar cuando el gateway recibió un paquete de un nodo. Si este dato no es relevante o si se consiera aceptable el dato de la hora que recibe el paquete el LoRa Server o la aplicación, puede omitirse las líneas de sincronización de hora para poder aislar el gateway de Internet, o para no tener que crear excepciones en el firewall corporativo si se cuenta con un política de seguridad estricta.

Para parametrizar el software del gateway hay que modificar 2 archivos, tal como está explicado en el Apéndice 13.7. En ellos hay que ajustar la banda de frecuencias que se utilizarán, el Spreading Factor, datos del funcionamiento del gateway, la dirección del servidor a la que se enviarán los paquetes, y las credenciales de la red inalámbrica.

Con esta parametrización realizada, se sube el programa al dispositivo y se puede acceder mediante el navegador web a una interfaz de usuario. En ella se puede conocer el estado del gateway y modificar alguno de sus parámetros.

Al utilizar un único canal, la autenticación de los dispositivos se debe realizar únicamente mediante ABP, lo que disminuye la seguridad respecto a otro gateway que permita utilizar OTAA, ya que con este último las claves de sesión son generadas dinámicamente.

A su vez, también se pudo probar satisfactoriamente la placa MakerFocus ESP32 Development Board v2 como gateway LoRa de un canal.

6. Nodo de sensado

Los denominados "nodos" consisten de al menos 2 dispositivos: un microprocesador y un módulo de comunicación (en este caso en particular uno que transmite en LoRa). Esto es porque en general se tiene adicionalmente un dispositivo que está sensando algo en el ambiente, el cual se conecta al microcontrolador para que haga algún procesamiento (en general filtrado de señal), para que luego el módulo de comunicación lo envíe.

Para el proyecto se utilizó la placa Sparkfun SPX-14893 también como nodo, dado que se tenía disponibilidad de estas placas.

Se decidió aprovechar un proyecto disponible [25] para instalarle al nodo un sistema operativo y programarlo de forma de poder utilizar el lenguaje de scripting *Lua*. Esto tiene como ventaja la facilidad para prototipar y configurar inicialmente la solución, ya que la parametrización se hace sin tener que volver a grabar (flashear) el sistema operativo en la placa. Otra ventaja es que esta parametrización puede realizarla alguien que no sepa programar en bajo nivel, ya que él opera con funciones siendo agnóstico de cómo se resuelve a nivel de la placa. A su vez, se pretende con este trabajo, generar conocimiento para que se pueda replicar la programación en capas para otras funcionalidades de éstas placas.

6.1. Lua-RTOS

Se trata de un sistema operativo de tipo "real-time" desarrollado por WhitecatBoard, basado en Lua. Se diseñó para ejecutarse en sistemas integrados con requisitos mínimos de memoria flash y RAM. Lua-RTOS es el núcleo del ecosistema Whitecat, y actualmente está disponible para las plataformas ESP32, ESP8266 y PIC32MZ [26].

Whitecat es un ecosistema que está siendo desarrollado en el laboratorio Citilab por un equipo de ingenieros, educadores y diseñadores, creado para implementar casos de uso IoT reales de una manera fácil. Cubre todos los aspectos para este tipo de soluciones: hardware y software para el nodo, el gateway y la nube.

Lua-RTOS tiene un diseño de tres capas tal como se muestra en la Figura 19.

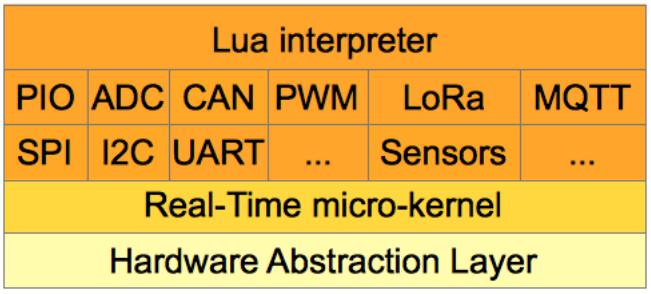

Figura 19: Capas Lua RTOS.

1. En la capa superior hay un intérprete Lua, que WhiteCat le agregó los módulos que consideraron relevantes para acceder al hardware y a los servicios de middleware (Lua Threads, LoRa WAN,MQTT, etc).

2. En la capa del medio hay un "real-time micro-kernel" con tecnología FreeRTOS. Se trata del mínimo software necesario para implementar un sistema operativo (administración de memoria, comunicación, etc).

3. La más baja es la capa de abstracción del hardware, que puede comunicarse directamente con el hardware de la plataforma. Cuando se trata de migrar Lua-RTOS a otras plataformas, los programadores necesitan escribir código solo para la capa inferior, porque las capas superior e intermedia pueden ser las mismas en todas las plataformas.

Para el corriente proyecto se utiliza Lua-RTOS para ESP32, disponible como un componente del framework oficial de su fabricante Espressif para dicho chip, denominado *ESP-IDF*.

Una característica de Lua a destacar es que no utiliza variables globales, sino que maneja una estructura dinámica donde se va almacenando el estado, y el puntero a esta estructura debe ser pasado como argumento a todas las funciones de Lua. Por lo que las variables que se desean pasar a una función deben apilarse a esta estructura antes de llamarla, y ser consumidas luego dentro la función.

En el Capítulo 13.5 del Apéndice, se explica cómo instalar este sistema operativo para la placa Sparkfun.

7. Implementación contador de dispositivos

Para contar los dispositivos en el medio es necesario contar con la lógica para este fin en los nodos. Dado que en Lua-RTOS no existía una función que permitiera identificar dispositivos inalámbricos, fue necesario estudiar y entender la implementación general de las capas, desde Lua hasta el driver de Espressif. En la Figura 20 se presenta un diseño resumido de las 3 capas que serán necesarias para el contador, que luego serán detalladas cada una en una sección.

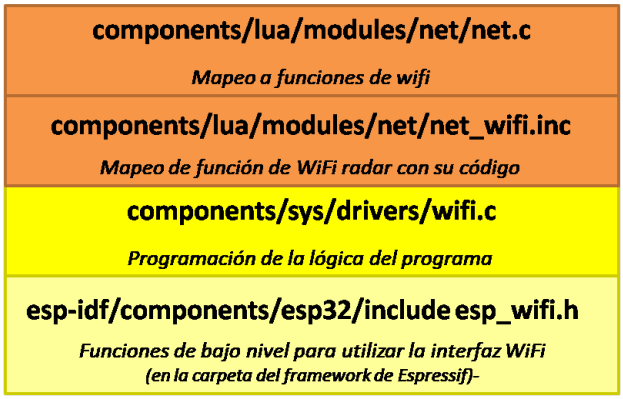

Figura 20: Capas Lua RTOS para función radar.

Se agregará la función *radar* a las de la biblioteca de Wi-Fi de Whitecatboard, haciendo lo que se denomina *binding* entre esta función de Lua y una programada en la capa inferior. De esta forma por ejemplo ejecutando *net.wf.radar(false,3,-60,true)*, devolverá la cantidad de dispositivos conectados a algún AP que pudo detectar con un RSSI mínimo de -60, recorriendo 3 segundos por canal.

Concretamente, para llamar a la función radar, hay que colocar el prefijo de la biblioteca de funciones de Wi-Fi, que es: *net.wf.* [29], y la sintaxis de la función es:
net.wf.radar(isTable, secPerChan, minRssi, isData) donde:

- *isTable* es un booleano que indica si la función devuelve la tabla de direcciones MAC y el RSSI correspondiente, o la cantidad
- *secPerChan* es la cantidad de segundos que se escaneará cada canal
- *minRssi* es el RSSI mínimo de los dispositivos
- *isData* escanea solo paquetes de clientes conectados o todos los paquetes

Cabe aclarar que si isData se fija en false, no solo se detectarán los dispositivos sin conectarse a una red inalámbrica, sino también se detectarán los AP como falsos positivos, ya que envían tramas de administración a otros dispositivos en la misma BSS, es decir con las banderas desdeAP y haciaAP en 0.

En el corriente Capítulo se presenta el código necesario para la función que cuenta dispositivos, y la configuración para el envío mediante LoRaWAN de los datos desde el nodo.

7.1. Implementación de la función radar en Lua-RTOS

A continuación se presenta la programación necesaria para cada capa en Lua-RTOS del contador de dispositivos.

El código desarrollado para esta función está disponible como un *fork* del proyecto original en https://github.com/fdetta/Lua-RTOS-ESP32.

7.1.1. Capa 3

En la carpeta *components/lua/modules* están los archivos de cada módulo del sistema operativo, divididos en carpetas. En particular en la carpeta *net* se encuentra el archivo net.c, que define que el prefijo *wf* se mapea con el registro *wifi_map*. Esto se puede observar dentro de la definición del registro "static const LUA_REG_TYPE net_map[]" , en la línea
"{ LSTRKEY('wf'), LROVAL (wifi_map) },".

El registro *wifi_map* se encuentra en el archivo *net_wifi.inc* alojado en la misma carpeta. En este registro se definen todas las funciones que estarán disponibles para el módulo, y su mapeo a la función dentro del mismo archivo.

Para el corriente trabajo se agrega la siguiente linea
" { LSTRKEY('radar'), LFUNCVAL(lwifi_radar) },", tal como se puede observar en la Figura 21.

```
static const LUA_REG_TYPE wifi_map[] = {
    { LSTRKEY( "setup"      ),     LFUNCVAL( lwifi_setup           ) },
    { LSTRKEY( "scan"       ),     LFUNCVAL( lwifi_scan            ) },
    { LSTRKEY( "start"      ),     LFUNCVAL( lwifi_start           ) },
    { LSTRKEY( "stop"       ),     LFUNCVAL( lwifi_stop            ) },
    { LSTRKEY( "stat"       ),     LFUNCVAL( lwifi_stat            ) },
    { LSTRKEY( "radar"      ),     LFUNCVAL( lwifi_radar           ) },
#if CONFIG_ESP32_WIFI_NVS_ENABLED
    // wps ssid+auth can only be stored to nvs
    // so if nvs is disabled, wps would have to be repeated on every boot
    { LSTRKEY( "startwps"  ),     LFUNCVAL( lwifi_wps          ) },
#endif
    { LSTRKEY( "startsc"   ),     LFUNCVAL( lwifi_smartconfig  ) },
    { LSTRKEY( "auth"      ),     LROVAL  ( wifi_auth_map      ) },
    { LSTRKEY( "mode"      ),     LROVAL  ( wifi_mode_map      ) },
    { LSTRKEY( "powersave" ),     LROVAL  ( wifi_powersave_map ) },
    { LSTRKEY( "wpstype"   ),     LROVAL  ( wifi_wpstype_map   ) },
    { LSTRKEY( "timecheck" ),     LROVAL  ( wifi_timecheck_map ) },

    DRIVER_REGISTER_LUA_ERRORS(wifi)
    { LNILKEY, LNILVAL }
};
```

Figura 21: Mapeo agregado en components/lua/modules/net/net_wifi.inc para la función radar.

Luego se define la función *static int lwifi_radar(lua_State* L)*. Allí primero se recuperan los parámetros del stack. Cabe mencionar que se considera mejor acceder a la pila mediante números negativos (desde el tope del stack), y no desde la base como utiliza Whitecat para el

resto de las funciones de la biblioteca, ya que podría suceder que la pila sea más grande y tenga otros parámetros debajo. Así, se implementan las actividades de pop y push de manera más adecuada para colas LIFO (Last in First Out).

Luego se llama a la función wifi_radar(&rssi, &maclist,&count,secs,onlyData))) de capa 2. Queda bloqueado hasta finalizado el escaneo realizado en la capa inferior, para generar la salida dependiendo si se seleccionó devolver el resultado en una tabla o la cantidad.

7.1.2. Capa 2

La funciones de capa 2 de la Wi-Fi se encuentran en el archivo *components/sys/drivers/wifi.c*. Se agregó la función *wifi_radar* que será la que llama la capa 3. La sintaxis es:
driver_error_t *wifi_radar(uint8_t *minsen, mac_t* *list, uint16_t * count,int secs,u8_t onlyData)

- *minsen* es el mínimo RSSI a sensar
- **list* es el puntero a la lista de MAC detectadas
- *count* es la cantidad de dispositivos detectados
- *secs* es la cantidad de segundos que escuchará cada canal
- *onlyData* es un booleano que indica si solo se tendrán en cuenta los paquetes de clientes conectados, o todos los paquetes.

En la función primero se inicia la memoria flash mediante la función de esp-idf *nvs_flash_init()* [30], y la interfaz Wi-Fi utilizando el mismo código que el de la función *scan* de la misma biblioteca.

Luego se usan las siguientes funciones de esp-idf para colocar la interfaz Wi-Fi en modo promisuco, llamándolas individualmente dentro de la función *ESP_ERROR_CHECK* para capturar errores:

- *esp_wifi_set_promiscuous_filter(&filter)* : fija el filtro de paquetes que estará aceptando. El filtro será todos o solo el de datos según el valor de *onlyData*.
- *esp_wifi_set_promiscuous_rx_cb(wifi_promiscuous_cb)*: en esta línea se define la función que será llamada al recibir un paquete. En este caso se trata de *wifi_promiscuous_cb*.
- *esp_wifi_set_promiscuous(true)*: en esta linea se pone la intefaz en modo promiscuo.

Luego se recorren los 13 canales de la siguiente forma. Se utiliza la función *esp_wifi_set_channel* para establecer el canal, para luego esperar los segundos definidos para cambiar de canal, mediante la función *vTaskDelay*.

Finalmente, se reserva la memoria continua necesaria para armar el array de MAC y RSSI, a fin de poder pasárselo a la capa superior. También se borran las estructuras auxiliares y se deshabilita el modo promiscuo de la interfaz Wi-Fi.

Función de callback *wifi_promiscuous_cb*

Cada vez que se recibe un paquete se ejecuta la función: *void wifi_promiscuous_cb(void *buf, wifi_promiscuous_pkt_type_t type)*, que tiene como primer parámetro la información recibida, y como segundo el tipo (que puede ser wifi_promiscuous_pkt_t or wifi_pkt_rx_ctrl_t).

A su vez, el wifi_promiscuous_pkt_t está compuesta de 2 estructuras

- El cabezal *rx_ctrl*, de tipo wifi_pkt_rx_ctrl_t. De aquí se obtiene el RSSI, ya que nos interesa la potencia del mensaje recibido y no la potencia entre el dispositivo y el AP.
- El *payload*, puntualmente el argumento es el puntero al primer elemento. El largo de la estructura está en el parámetro rx_ctrl.sig_len. Esta estructura es la trama 802.11, con los campos tal como se detallan en el Capítulo 2.2.

Con esta información, según el valor de las banderas haciaAP y desdeAP se extrae del paquete la dirección del dispositivo y se agrega la MAC a la lista.

Manejo dinámico de memoria en el armado de la lista

Es práctica común de los sistemas embebidos reservar memoria fija para el armado de estructuras, dado que los dispositivos tienen memoria muy limitada. Esto asegura que el sistema se comporte de forma esperada y no existan desbordamientos de memoria.

Sin embargo, en este trabajo se optó por estudiar el armado de listas con memoria dinámica, dado que el dimensionamiento de la lista puede variar mucho dependiendo de dónde se despliegue la solución. Este planteo se basa en querer aprovechar lo máximo posible Lua, dado que cuanto menos memoria utilice la función, más operaciones permitirá realizar posteriormente.

El mismo proyecto de WhitecatBoard cuenta con una biblioteca para utilizar listas encadenadas, definida en el archivo *LinkedList.h*. Para este trabajo se utilizaron 3 funciones:

- *ListInitialize()*: para inicializar la lista
- *ListAppend(maclist, (void*)mac, strlen(mac))*: para agregar el elemento mac a la lista maclist, utilizada en la función que se dispara al recibir un paquete.
- *ListFindItem(maclist,mac,macCompare)*: para buscar el elemento mac en la lista maclist, comparando según la función macCompare. macCompare compara solo los primeros 12 bytes, ya que en la cadena se envía también el RSSI.
- *ListFree(maclist)*: para borrar la lista y liberar la memoria.

Tal como se detallará en el Capítulo 9.1, se realizaron pruebas para determinar el tamaño máximo de lista que se puede generar utilizando listas encadenadas. Como no se tuvieron problemas con menos de 1200 elementos, se consideró que la solución era aceptable para este caso de uso.

7.1.3. Capa 1

La capa 1 se considera la programación del framework de Espressif para el manejo del hardware. La biblioteca de Wi-Fi se encuentra en *esp-idf/components/esp32/include esp_wifi.h*, pero dado que hay amplia documentación en la web del fabricante, bajo el título de *API Reference*, no se recomienda consultar el código directamente salvo que sea necesario realizar cambios.

7.2. Comunicación con infraestructura

Se configuró el nodo para transmitir en la interfaz LoRa por un canal único, dado que el gateway seleccionado puede recibir únicamente en una frecuencia. A continuación se detalla la configuración para los parámetros en LuaRTOS, y cómo enviar la información tanto a un servidor LoRaWAN privado (LoRa Server) como a uno comunitario (TTN).

7.2.1. Configuración LoRa en LuaRTOS

Este sistema operativo no permite fijar el canal de transmisión de LoRa a nivel de Lua, por lo que los parámetros deben fijarse en la configuración y no pueden modificarse luego de compilado y flasheado el sistema operativo. Éstos se encuentran en el archivo *components/lora/node/lmic/lorabase.h*. En la Figura 22 se puede observar como se definen las frecuencias tal como lo establece Lora Alliance para los parámetros regionales. Se definen 3 parámetros que se tratan de las frecuencias base, y 3 adicionales denominados "step", para ir recorriendo cada uno de los canales.

```
113  #elif defined(CFG_us915)  // ========================================
114
115  enum _dr_us915_t { DR_SF10=0, DR_SF9, DR_SF8, DR_SF7, DR_SF8C, DR_NONE,
116                     // Devices behind a router:
117                     DR_SF12CR=8, DR_SF11CR, DR_SF10CR, DR_SF9CR, DR_SF8CR, DR_SF7CR };
118  enum { DR_DFLTMIN = DR_SF8C };
119  enum { DR_PAGE = DR_PAGE_US915 };
120
121  // Default frequency plan for US 915MHz
122  enum { US915_125kHz_UPFBASE = 902300000,
123         US915_125kHz_UPFSTEP =    200000,
124         US915_500kHz_UPFBASE = 903000000,
125         US915_500kHz_UPFSTEP =   1600000,
126         US915_500kHz_DNFBASE = 923300000,
127         US915_500kHz_DNFSTEP =    600000
128  };
129  enum { US915_FREQ_MIN = 902000000,
130         US915_FREQ_MAX = 928000000 };
```

Figura 22: Configuración de plan de frecuencias en components/lora/node/lmic/lorabase.h.

Pero para este caso de uso, y para asegurarse que toda la transmisión se realice en la frecuencia elegida, se setearon las bases y los límites de la banda en la misma frecuencia, y los saltos en 0.

Configuración del nodo en Lora Server

Para agregar el nodo al Lora Server hay que seguir los siguientes pasos:

1. En la intefaz web de Lora Server, acceder a la aplicación, para agregar un nuevo dispositivo
2. Lo primero que va a solicitar el es DevEUI, en ese campo hay que colocar el resultado de ejecutar "lora.getDevEui()" en la consola de Lua del nodo.
3. Colocar la versión de MAC en 1.0.2.
4. Las claves Device Address, Network Session Key y App Session Key se pueden generar aleatoriamente para luego agregarlas al script de Lua en el nodo.

Configuración del nodo en TTN

En el caso de querer agregar el nodo a TTN, el procedimiento es el siguiente:

1. En la consola de TTN, ir a la aplicación para agregar un nuevo dispositivo
2. Colocar un nombre como identificador, y en Device EUI el resultado de ejecutar "lora.getDevEui()" en la consola de Lua del nodo.
3. En settings, pasar el activation method a ABP y grabar.
4. Las claves Device Address, Network Session Key y App Session Key se pueden generar aleatoriamente para luego agregarlas al script de Lua en el nodo.

Script Lua para conexión

A continuación se muestran el script necesario para poder mandar el texto "hola" mediante LoRaWAN en Lua.

```
lora.attach(lora.BAND915)
lora.setAppEui(appeui)
lora.setDevAddr(devaddr)
lora.setNwksKey(nwkskey)
lora.setAppsKey(appskey)
lora.setDr(0)
lora.setAdr(false)
lora.setReTx(0)
lora.tx(false,1,pack.pack("hola"))
```

En este script, el comando lora.attach(band) fija la banda de frecuencias. Los próximos 4 comandos fijan las credenciales de la comunicación LoRaWAN, a partir de las variables appeui,devaddr,nwkskey y appskey definidas previamente. Mediante el comando lora.setDr(0) define que el máximo data rate a ser utilizado en la próxima transmisión sea 0, y mediante el comando lora.setAdr(false) se deshabilita la función de *Adaptive Data Rate*. Luego se fija con el comando lora.setReTX(0) que no se retransmitan mensajes sin confirmación. Finalmente el comando lora.tx que es el utilizado para transmitir tiene 3 parámetros, el primero es un booleano que indica si se quiere la confirmación del paquete, el segundo es el puerto (que en este caso según la wiki de Whitecat debe fijarse en 1), y el tercero es el mensaje, que debe ser el mensaje codificado en hexadecimal y por ello se utiliza también la función pack.pack(string).

7.3. Ejecutar contador de dispositivos

Según la wiki de Whitecat, al iniciar el dispositivo se ejecutan los archivos system.lua y autorun.lua. Se recomienda usar el primero para configuración, y el segundo para tareas que deben ejecutarse posteriormente.

system.lua

El archivo system.lua se programó de la siguiente manera:

```
dofile(abp.lua)
```

Es decir, ejecuta en su única línea otro script para configurar los parámetros de la transmisión de LoraWAN. El script abp.lua es el ejemplo *Script Lua para conexión*, pero sin la última línea que transmite el dato.

autorun.lua

En este script, se deja a Lua en un loop infinito. Se ejecuta el archivo que escanea dispositivos y los envía, para luego apagarse determinada cantidad de segundos antes de volver a escanear. Se utiliza la función tmr.delay de forma que sea el propio sistema operativo el que maneje la espera y pueda optimizar la batería.

```
while (true)
do
  dofile("scan.lua")
  tmr.delay(15)
end
```

El script scan.lua ejecuta la función radar para obtener la lista de MAC detectadas con su RSSI. Luego se recorre la lista y manda al servidor cada elemento en un mensaje LoRaWAN. Eso se realiza mediante el código:

```
t = net.wf.radar(booltable,secs,rssimin,onlyData)
for key,value in pairs(t) do lora.tx(false,1,pack.pack(value)) end
```

Cabe destacar que en entornos con muchos dispositivos, el envío de los elementos de la lista mediante este script puede ser una limitante, dado que el nodo demora aproximadamente 4 segundos entre el envío de mensajes consecutivos.

7.4. Concurrencia

Como las interfaces Wi-Fi y LoRa de la placa del nodo son independientes, se consideró que era óptimo estar escaneando y enviando la información a la vez. Para ello se programó el script scan.lua utilizando la biblioteca Thread y con una cola con semáforo. De esta forma un hilo escanea los dispositivos Wi-Fi y los agrega a la cola, y otro hilo consulta la cola y envía los elementos nuevos mediante LoRaWAN. El código es el siguiente:

```
lista={}
mtx=thread.createmutex()

thread.start(function()
  while (true) do
    t=net.wf.radar(true,2,-70,false)
    for key,value in pairs(t) do
      mtx:lock()
      table.insert(lista,value)
      mtx:unlock()
    end
  end
end)

thread.start(function()
  while (true) do
    if #lista=0 then
      thread.sleep(10)
    end
```

```
    for key,value in pairs(lista) do
      mtx:lock()
      lora.tx(false,1,pack.pack(value))
      table.remove(lista,key)
      mtx:unlock()
    end
  end
end)
```

Sin embargo, sucedió en más de una oportunidad que el hilo de transmisión LoRa se bloqueó en el envío de mensajes (previo a consultar por mensajes que pueda recibir) y ello hizo que se bloqueara el nodo completamente.

Si bien el manejo de hilos concurrentes era la solución más recomendable, "debugguear" las funciones de la transmisión de LoRa para ubicar este problema intermitente iba a demandar un tiempo considerable. A lo que no afectaba al alcance de este trabajo, se descartó esta implementación.

8. Visualización de los Datos

En esta sección se explica la solución de capa de aplicación que se realizó para la interacción con el usuario. La interfaz con el usuario será un sitio web que muestra información tanto en tiempo real como estadísticas históricas.

8.1. Integración HTTP con Lora Server

Se realizó la integración HTTP para poder exportar los datos obtenidos por los nodos a una aplicación que pueda almacenar los datos para su consulta. Se decidió usar PHP para las páginas web, y postgreSQL como motor de base de datos por familiaridad en su uso.

Para ello en la consola de Lora Server, hay que acceder a la aplicación y dentro de la sección *Integrations* de la aplicación se crea una nueva integración HTTP. En este caso se ingresa la ruta a la página PHP que atanderá los nuevos mensajes en *Uplink data URL*.

En la Figura 23 se muestra el paquete HTTP enviado por el Lora Server como parte de la integración HTTP. Como se puede ver el Lora Server agrega un cabezal con información de la aplicación y coloca en el campo *rxInfo* el mensaje enviado por el gateway.

Figura 23: Mensaje de integración HTTP.

El campo de datos *data*, es enviado en base 64, por lo que se decodifica previo a su almacenamiento. Se decidió almacenar el dato con la hora que la recibe la aplicación y no la que envía el gateway, considerando lo expresado en el Capítulo 5.2.

8.2. Persistencia

Se crearon 3 tablas: *log* que almacena la información recibida por los nodos en caso que se decida enviar la MAC, *devicecount* en caso que sólo se envíe el total de dispositos, y *devicename* que almacena nombres personalizados para los identificadores de los nodos.

La tabla *log* se crea con los siguientes parámetros, de la siguiente manera:

```
CREATE TABLE log
(
   mac character varying(23),
   sensor character varying(50),
   rssi integer,
```

```
    seenat timestamp without time zone DEFAULT CURRENT TIMESTAMP
);
```

En el caso de *devicecount* se crea con los siguientes parámetros:

```
CREATE TABLE devicecount
(
    quantity integer,
    seenat timestamp without time zone DEFAULT CURRENT TIMESTAMP
);
```

Por otro lado, la tabla devicename se creó con la finalidad de mostrar nombres más entendibles por los usuarios, dicha tabla tiene se creó con la siguiente estructura:

```
CREATE TABLE devicename
(
    sensor character varying(50),
    name  character varying(25)
);
```

8.3. Interfaz con el usuario

Se disponibiliza un sitio web con 6 consultas, a partir de la información almacenada en la base de datos. Cada consulta será desarrollada en un capítulo a continuación. Todo el desarrollo se realizó considerando que los nodos mandan la lista de dispositivos detectados.

Para hacerlo visualmente más amigable se utilizó Bootstrap 3, y la API que ofrece Google [31] para realizar gráficas. Éstas consultas, conjuntamente a las de la API de los fabricantes de dispositivos Wi-Fi son las únicas integraciones que necesitan Internet.

Vale aclarar que la información que se envía a Google son cantidades y/o fechas, y los datos utilizados para este trabajo fueron mayormente de prueba, pero si se quiere tener un mayor nivel de seguridad de la información se sugiere utilizar alguna aplicación local para presentar la información.

8.3.1. Información en tiempo real

En esta consulta se muestra para cada nodo, la cantidad de dispositivos que está sensando, tal como se muestra en la Figura 24. Para fines visuales, se le agregó un mapa de fondo así como la ubicación geográfica de cada sensor en él.

Consulta SQL

Para la consulta se tuvieron 2 consideraciones: contar cada dispositivo Wi-Fi sólo en el sensor que lo detectó con potencia más fuerte, y establecer una ventana de tiempo que se considera "Tiempo real". Esto último es porque los sensores mandan información espaciadamente, y los tiempos de transmisión varían según el tamaño de la lista.

La consulta SQL para este caso, considerando $1 la cantidad de minutos de ventana y $2 el identificador del sensor que queremos contar se muestra a continuación:

Figura 24: Consulta dispositivos Wi-Fi en tiempo real.

```
SELECT count(distinct(t.mac)) as total from (
SELECT t1.mac,t1.sensor,t1.rssi,t1.seenat FROM public.log as t1
  INNER JOIN
  (SELECT mac,min(rssi) as maximo
  FROM public.log
  WHERE now()::timestamp-seenat < $1 * interval '1' minute
  GROUP BY mac) as t2
  on t1.mac=t2.mac and t1.rssi=t2.maximo
  ) as t WHERE now()::timestamp-seenat < $1 * interval '1' minute
  and sensor=$2;
```

8.3.2. Estadísticas

En esta consulta se muestra en una única gráfica la cantidad de dispositivos sensados por hora por cada nodo, para un día seleccionado. Como ejemplo se muestra la Figura 25.

Consulta SQL

Los parámetros de esta consulta serán $1 el identificador del sensor y $2 la fecha.

```
select date_trunc('hour',t1.datetr) as datetrh ,max(macs) as macs from (
select distinct(date_trunc('hour',seenat)) as datetr
from log group by datetr) as t1
left join
(SELECT date_trunc('hour',seenat) as datetr,
count(distinct(mac)) as macs from log where sensor=$1 group by datetr)
as t2 on t1.datetr=t2.datetr where DATE(t1.datetr)=$2
group by date_trunc('hour',t1.datetr) order by date_trunc('hour',t1.datetr);
```

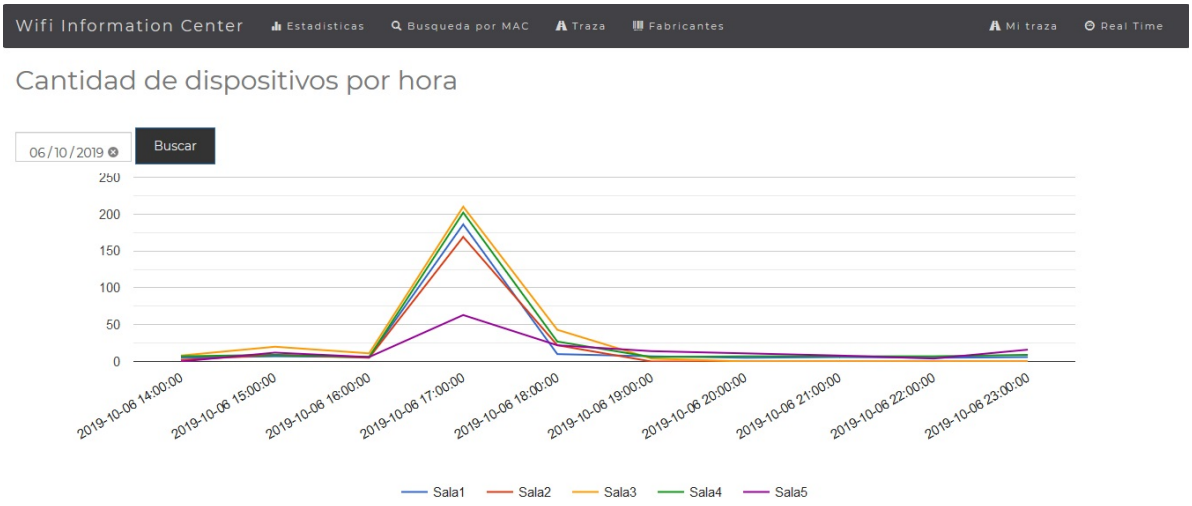

Figura 25: Consulta estadísticas por nodo.

8.3.3. Búsqueda por MAC

En esta consulta se pueden buscar los registros tal como se almacenaron en la tabla *log* (Figura 26).

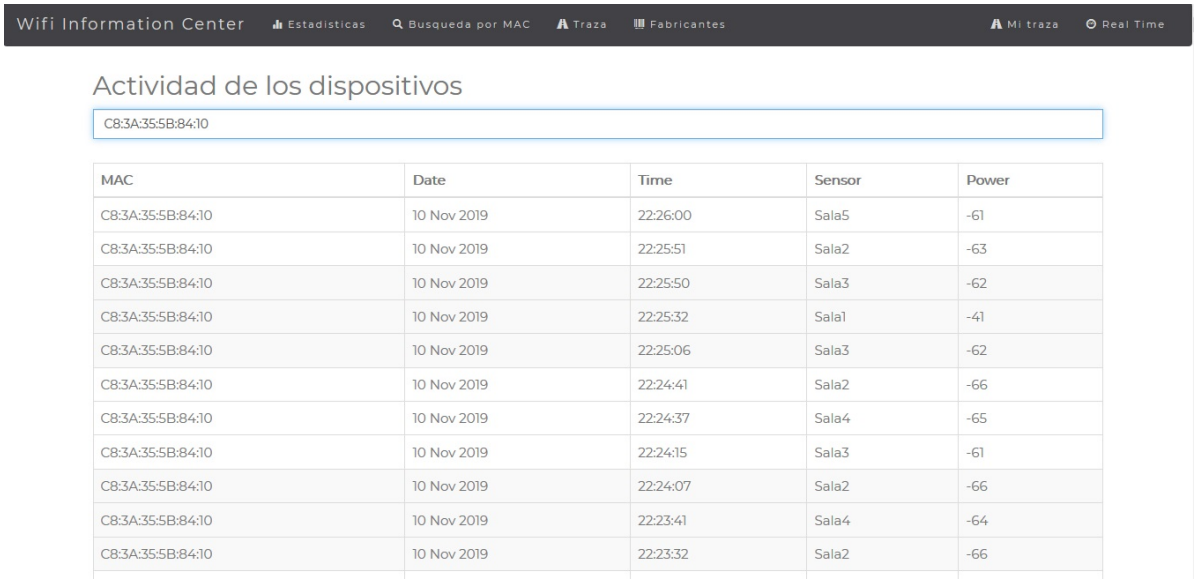

Figura 26: Consulta búsqueda de MAC.

Consulta SQL

La consulta SQL es:

```
SELECT log.mac,to_char(log.seenat, 'HH24:MI:SS') as seenathour,
to_char(log.seenat,'DD Mon YYYY') as seenatday, devicename.name, log.rssi
FROM log left join devicename on log.sensor = devicename.sensor
order by log.seenat desc;
```

8.3.4. Traza

Esta consulta permite armar una traza para una dirección MAC, suponiendo que el dispositivo Wi-Fi se encontraba en la posición del sensor el tiempo entre ese registro y el próximo en otro sensor. En la Figura 27 se muestra la consulta.

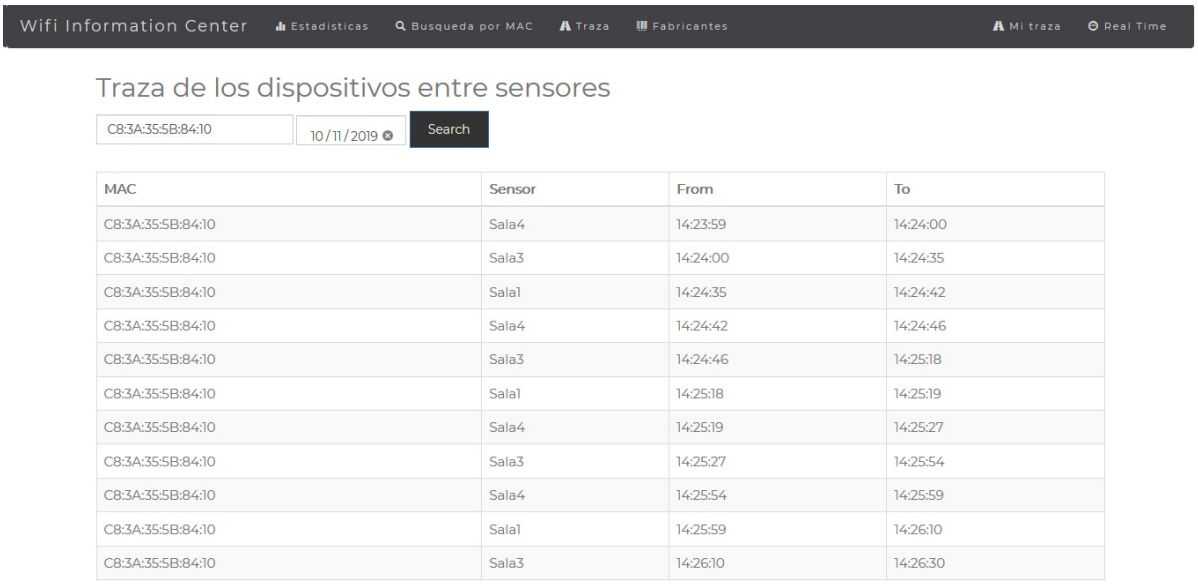

Figura 27: Consulta traza de un dispositivo.

Consulta SQL

Los parámetros en esta consulta son 3:

- $1 es la dirección MAC
- $2 es el RSSI mínimo que se considerará. A modo de simplificar la consulta se agregó este parámetro para descartar señales débiles que podría estar tomando 1 sensor lejano.
- $3 es la fecha de la que se desea la traza

La consulta SQL es entonces:

```
SELECT log.mac, devicename.name, to_char(log.seenat, 'HH24:MI:SS')
as seenath,
to_char(LEAD(log.seenat) over (order by log.mac, log.seenat), 'HH24:MI:SS')
as next, log.seenat,
LEAD(log.seenat) over (order by log.mac,log.seenat) as nextdate
FROM log left join devicename on log.sensor = devicename.sensor
WHERE mac = $1 and rssi>$2 and DATE(seenat) = DATE($3)
ORDER BY log.mac, log.seenat ;
```

8.3.5. Mi Traza

Esta consulta muestra la traza de un dispositivo Wi-Fi, pero en lugar de armarla según campos de selección, despliega la traza realizada ese día por el dispositivo más cercano a uno de los nodos. La idea detrás de esta consulta es que si una persona acerca su dispositivo a ese nodo, puede verificar la traza que acaba de realizar.

Consulta SQL

En este caso se realizan 2 consultas. Primero para detectar el dispositivo más cercano se toman parámetros $1 el sensor considerado para la detección, y $2 la cantidad de minutos de la ventana. La primer consulta queda entonces:

```
SELECT log.mac FROM log
WHERE log.sensor =$1 and now()::timestamp-log.seenat \< $2 * interval '1' minute
ORDER BY log.rssi desc LIMIT 1;
```

Con este dato como $1, y $2 el rssi mínimo, se ejecuta la siguiente consulta:

```
SELECT log.mac, devicename.name, to_char(log.seenat, 'HH24:MI:SS')
as seenath,
to_char(LEAD(log.seenat) over (order by log.mac, log.seenat),'HH24:MI:SS')
as next, log.seenat,
LEAD(log.seenat) over (order by log.mac, log.seenat) as nextdate
FROM log LEFT JOIN devicename on log.sensor = devicename.sensor
WHERE mac = $1 and rssi>$2 and DATE(seenat)=DATE(now()::timestamp)
ORDER BY log.mac, log.seenat ;
```

8.3.6. Fabricantes

Aprovechando que los 6 primeros bytes de la dirección MAC indican el fabricante del dispositivo, se generó esta consulta a fin de conocer y cuantificar las distintas tecnologías utilizadas en los dispositivos Wi-Fi (Figura 28).

Figura 28: Consulta fabricantes dispositivos Wi-Fi.

Consulta SQL

En este caso la consulta obtiene los distintos fabricantes de dispositivos conectados en los últimos $1 minutos.

```
SELECT LEFT(mac,8) as vendor,count(DISTINCT(mac)) as macs
FROM log WHERE now()::timestamp-seenat \< \$1 * interval '1' minute
GROUP BY LEFT(mac,8) ORDER BY macs DESC;
```

Con cada salida se consulta una API disponible online [28], y se arma la gráfica de tipo torta de Google. Cabe mencionar que se podría descargar la lista completa de fabricantes con sus prefijos, y realizar una consulta SQL local en lugar de consultar una API externa.

9. Evaluación de solución

A fin de evaluar la solución propuesta se realizaron 3 pruebas. En la Sección 9.1 se detalla la prueba realizada respecto a la cantidad máxima de dispositivos que pueden sensar los nodos. En la Sección 9.2 se hizo una prueba controlada para evaluar el comportamiento de la solución completa, y en la Sección 9.3 se evaluó la solución en una situación real.

9.1. Prueba de estrés

Como se utilizó memoria dinámica para la generación de la lista de direcciones MAC se realizó una prueba de estrés para evaluar cuál es el máximo de dispositivos que puede contar un nodo.

Para afectar lo mínimo posible el código del nodo y poder obtener la medida más precisa posible sobre este valor, se programó en la función de *callback* (descripta en el Capítulo 7.1.2) que cada vez que se reciba un paquete, se agregue varias veces la dirección MAC de origen a la lista, mediante el comando *ListAppend*.

Como se muestra en la Figura 29 el comando radar funciona bien con 1156 elementos, finalizando y dejando en memoria la tabla. Luego, con 1370 elementos como se observa en la Figura 30, genera un error de *StoreProhibited*.

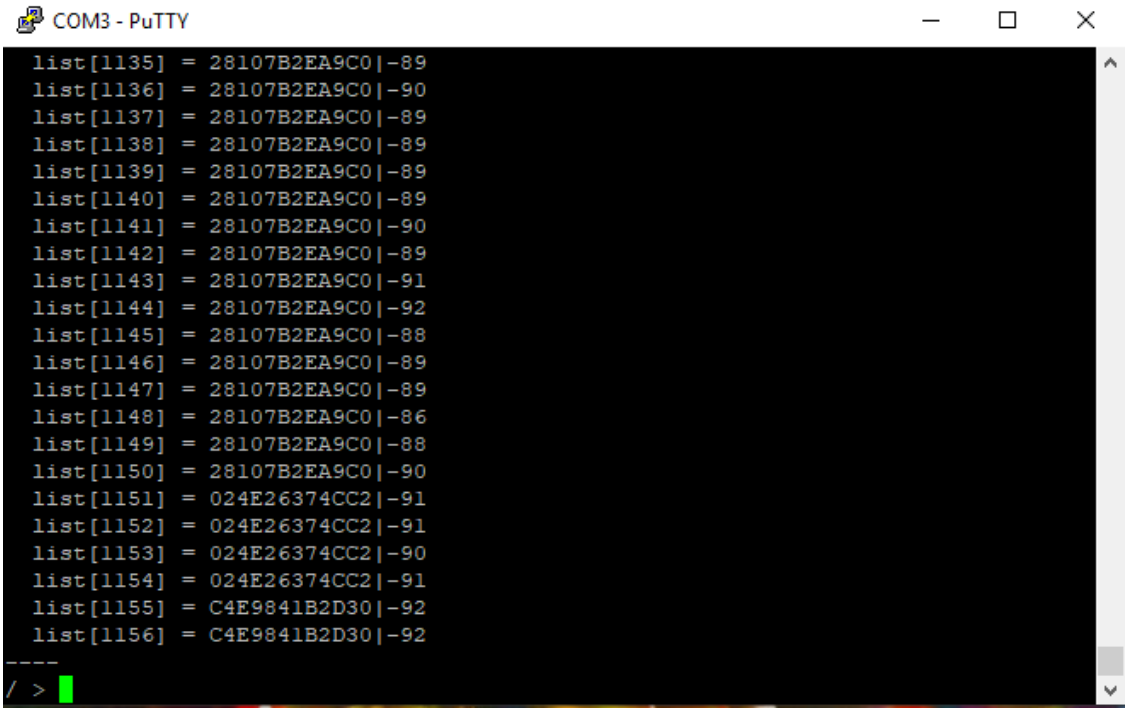

Figura 29: Test de estrés - Funcionamiento ok para una lista de 1156 elementos.

Por lo que si se utiliza el nodo configurado como se definió en los capítulos anteriores y para este fin, esta solución puede sensar hasta 1200 elementos sin problemas de memoria.

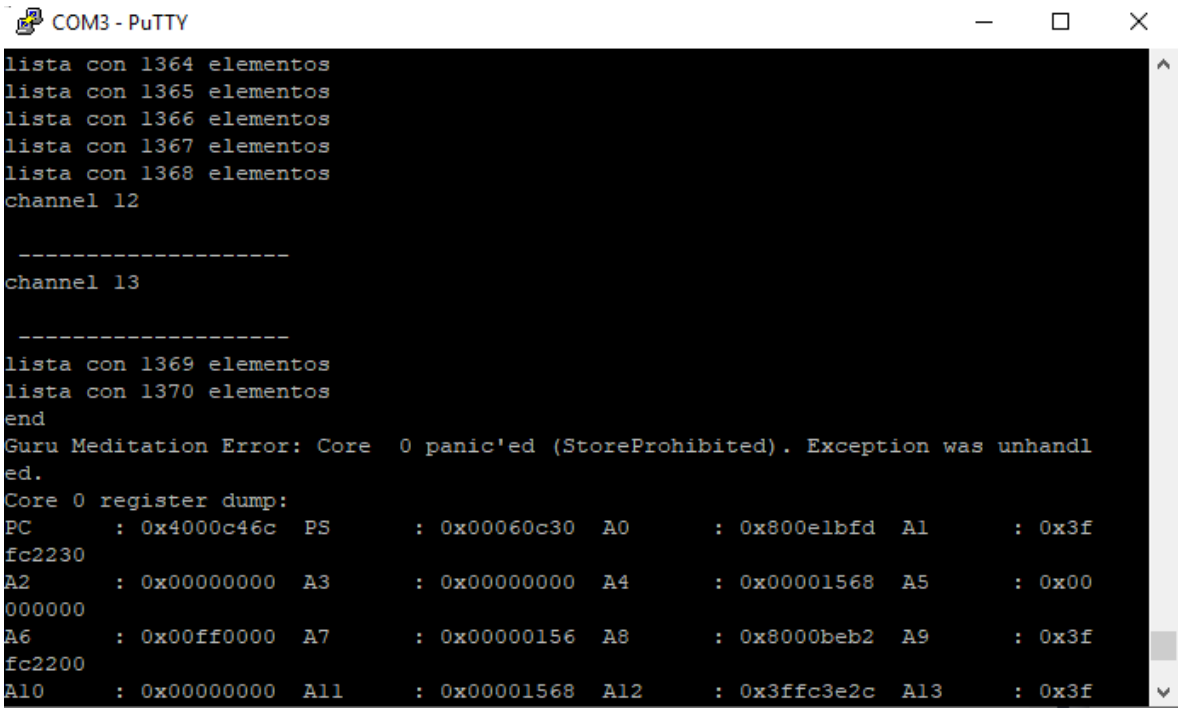

Figura 30: Test de estrés - Error para una lista de 1370 elementos.

9.2. Prueba funcionamiento y validación

Se realizó una prueba de funcionamiento y validación de la solución de forma controlada. Se buscará emular las casuísticas más habituales y evaluarán los resultados. A nivel de la solución, se desplegaron y configuraron 4 nodos en habitaciones contiguas a aproximadamente 10 metros uno del otro.

Para estas pruebas se utilizaron 4 celulares de distintas marcas y versiones de Android, ellos fueron:

- Sony Xperia - Android 4.1.2
- LG - Android 5
- Xiaomi - Android 7.0 con MIUI 11.0.2.0
- Huawei - Android 10 con EMUI 10.0.0

Como observación surge que los AP son detectados como dispositivos cuando se escuchan todas las tramas. Esto se puede validar ejecutando la función radar con *isData* en ambos valores. También el gateway LoRa es detectado como dispositivo si se encuentra al alcance de un nodo, ya que éste envía los paquetes al Bridge mediante Wi-Fi.

Prueba alcance

Se utilizaron los dispositivos para obtener una correspondencia entre el RSSI y la distancia del nodo al dispositivo. Se pudo observar que el RSSI varió entre -24 y -46 cuando los dispositivos estuvieron a menos de 20 cms de la antena, y entre -56 y -68 tanto para cuando los dispositivos estaban a 3 metros en la misma habitación, como a 1 metro del otro lado de la pared.

Para las pruebas siguientes, se fijó el RSSI mínimo en -50, a modo de poder evaluar la actividad de los celulares, sin que se vean alterados por dispositivos que pudieran encontrarse cerca.

Detección dispositivos sin conexión a red

Se probó detectar los 4 dispositivos con la interfaz Wi-Fi activada, pero sin que ellos estuvieran conectados a una red. Para ello se configuró para que acepte todas las tramas, colocando la variable de radar *isData* en false.

Los dispositivos con Android 4.1.2 y 5 fueron detectados con su MAC, pero los otros 2 dispositivos utilizaron anonimización de su dirección. Ambos dispositivos las generaron con el prefijo "DA:A1:19".

Al colocar el tiempo de escucha por canal en 1 segundo, el dispositivo Xiaomi generó entre 1 y 2 direcciones aleatorias por escaneo, pero al cambiarlo a 3 segundos aumentó entre 3 y 4 direcciones aleatorias. El dispositivo Huawei generó 1 sola MAC aleatoria en ambos casos.

Detección dispositivos con conexión a red

Para esta prueba se conectan los 4 celulares a una red Wi-Fi, colocándose al alcance de uno de los nodos, para luego ser movidos entre los otros.

Al aceptar todos los tipos de trama, la observación más importante es que los dispositivos con Android 7 y 10 generan adicionalmente 1 dirección MAC aleatoria, esporádicamente, con el prefijo "DA:A1:19". Se detectó que para el caso del dispositivo Huawei dicha actividad es la asociada a los servicios de *Búsqueda de redes Wi-Fi*, que es un servicio de Android para mejorar la precisión del posicionamiento permitiendo que las aplicaciones y los servicios busquen redes Wi-Fi en cualquier momento, incluso cuando la red Wi-Fi está deshabilitada. Esto causa que los contadores de dispositivos de la aplicación detecten cada determinado tiempo un dispositivo más de los que efectivamente existen. Fuera de esos momentos, la aplicación detectó correctamente la cantidad de dispositivos. En una hora, se detectaron 11 MAC aleatorias generadas por estos 2 dispositivos (Figura 31).

Respecto a las trazas, las mismas funcionaron correctamente para los dispositivos conectados. En la Figura 32 se muestra la consulta "mi traza" del dispositivo que recorrió todos los nodos en orden.

Luego se cambió el script de Lua de los nodos para que sólo detecten tramas de tipo datos. En más de 3 horas de prueba continua, se detectó en todo momento únicamente los 4 celulares con su MAC.

En resumen, si se selecciona detectar sólo dispositivos conectados a una red Wi-Fi, se detectarán correctamente estos dispositivos, pero hay que tener en consideración que algunos de estos dispositivos pueden generar paquetes con direcciones MAC aleatorias, y por otro lado no se detectarán dispositivos con la interfaz Wi-Fi activada sin estar conectados a una red. Por lo que se tendrá una buena medida de los dispositivos conectados a alguna red, pero no del total de dispositivos en un área.

Si se desea tener una idea más aproximada del total del dispositivos en un área, detectar también dispositivos no conectados es la mejor opción, pero como contra se detectarán falsos positivos a raíz de la anonimización de direcciones MAC, así como la contabilización de los AP como dispositivos.

Actividad de los dispositivos

da:a1:1

MAC	Date	Time	Sensor	Power
DA:A1:19:27:B7:8A	08 Dec 2019	02:47:22	Sala4	-28
DA:A1:19:79:79:79	08 Dec 2019	02:47:01	Sala1	-23
DA:A1:19:E0:38:02	08 Dec 2019	02:45:04	Sala1	-39
DA:A1:19:13:F7:07	08 Dec 2019	02:44:24	Sala4	-27
DA:A1:19:64:64:64	08 Dec 2019	02:32:25	Sala1	-23
DA:A1:19:AF:AF:AF	08 Dec 2019	02:22:27	Sala1	-22
DA:A1:19:1C:1C:1C	08 Dec 2019	01:58:44	Sala1	-49
DA:A1:19:A0:CD:11	08 Dec 2019	01:55:05	Sala1	-34
DA:A1:19:07:65:81	08 Dec 2019	01:53:55	Sala1	-31
DA:A1:19:CF:CF:CF	08 Dec 2019	01:53:50	Sala1	-40
DA:A1:19:37:A6:E6	08 Dec 2019	01:53:43	Sala1	-31

Figura 31: MAC aleatorias generadas por dispositivos conectados a una red Wi-Fi.

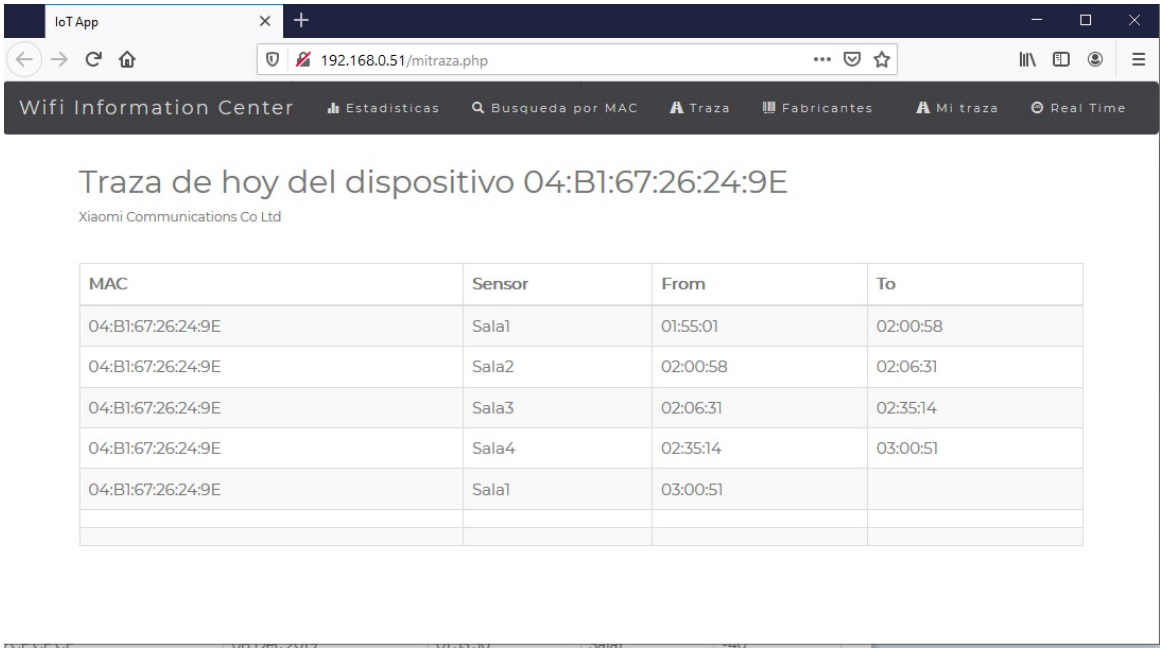

Figura 32: Traza dispositivo conectado.

9.3. Ingenieria deMuestra 2019

Del 10 al 12 de octubre del 2019 se organizó en Facultad de Ingeniería este evento organizado por la Facultad de Ingeniería de la Universidad de la República y su Fundación Julio Ricaldoni. Es un evento con la finalidad de presentar proyectos, investigaciones y emprendimientos de estudiantes y docentes, para exhibir las actividades que se realizan en la institución [32].

Se inscribió este proyecto como parte de los proyectos de grado del Instituto de Computación,

aprovechando este evento para probar la solución en un caso real. Se trató del primer proyecto del Instituto utilizando éstas tecnologías.

Figura 33: Instalación de nodos en Ingeniería deMuestra 2019.

Se colocaron 4 nodos con baterías de tipo "powerbank" distribuidos en las distintas salas donde se presentaban los proyectos, tal como se muestra en la Figura 33. Se trataban de baterías 18650 de 3000mAh, 3.7V. En el stand se colocó 1 nodo más, junto al gateway y el resto de la infraestructura instalada en una máquina virtual en un PC (Figura 34). El gateway se conectó a un "powerbank" de 20.000 mAh. Los nodos identificados como *A01* y *B01* se encontraban en distintas ubicaciones a 50 metros del gateway, ambos con 3 paredes en su trayecto y varios Stands con gente circulando. Por otro lado los nodos *Hall* y *Carpa* se encontraban aproximadamente a 20 metros del gateway, detras de una pared y también con Stands y gente circulando. Cabe señalar que contar con una solución similar pero utilizando una tecnología de comunicación como Wi-Fi, hubiera requerido más infraestructura, demandando mayores tiempos de instalación y costos.

Pruebas Realizadas

Si bien LoRaWAN es un protocolo pensado para el envío de poca información durante períodos espaciados de tiempo, maximizando así la duración de la batería, se configuraron los nodos para estar sensando de forma continua durante las 4 horas diarias que duraba el evento, a modo de estresar la solución y evaluar los resultados. Se pudo apreciar que cuando se detectaban muchos dispositivos, el tiempo que demoraba el nodo en enviar el mensaje y esperar las respuestas (aproximadamente 4 segundos por mensaje) hacía que el sensado se realizara de forma muy espaciada, degradando la calidad del dato de cantidad de dispositivos.

Figura 34: Stand en Ingeniería deMuestra 2019.

Por lo que la máxima cantidad de mensajes que puede enviar el nodo mediante LoRaWAN en una hora es de 900 aproximadamente, y al configurar el envío de la lista completa de direcciones MAC, éste será el máximo de dispositivos por hora que detectará la solución.

Como conclusión general se puede indicar que la solución se comportó de forma estable, durando todas las baterías las 4 horas diarias del evento. El gateway, que era el que se encontraba con 1 batería más potente, no llegó a descargar la mitad de la carga al finalizar los 3 días del evento.

Analizando los datos generados en el evento, se detectaron en total 11.385 dispositivos. Esto se debe en gran medida por la anonimización de la MAC de los dispositivos no conectados a una red Wi-Fi, tal como se explicó en el Capítulo 2.4. De hecho, 4.113 direcciones MAC tienen el prefijo “DA:A1:19”, registrado por el fabricante “Google, Inc.”.

Si se quitan los dispositivos con este prefijo y los que no tenian registrado el fabricante en el sitio consultado [28], se obtienen 2.021 dispositivos pertenecientes a 145 fabricantes. En el Cuadro 2 se presentan los 10 fabricantes con más dispositivos.

Al analizar el segundo fabricante, se observan que 180 dispositivos tienen el prefijo “D4:63:C6” y fueron detectados sólo por un sensor durante los 3 días del evento. Por lo que se supone que se trata de alguno de los dispositivos utilizados por alguno de los proyectos presentados en el evento, y probablemente anonimice su dirección MAC.

Fabricante	Cantidad
Samsung Electronics Co.,Ltd	482
Motorola Mobility LLC, a Lenovo Company	360
Apple, Inc.	272
Xiaomi Communications Co Ltd	260
HUAWEI TECHNOLOGIES CO.,LTD	135
Intel Corporate	61
SAMSUNG ELECTRO-MECHANICS(THAILAND)	42
Murata Manufacturing Co., Ltd.	41
LG Electronics (Mobile Communications)	39
Hon Hai Precision Ind. Co.,Ltd.	24

Cuadro 2: Fabricantes de dispositivos inalámbricos.

Con la información recolectada, descartando estos casos, se ejecuta la consulta de dispositivos por hora en la aplicación para obtener una medida más fiable de la cantidad de dispositivos de individuos que visitaron el evento.

Tal como se puede observar en la Figura 35 el primer día tuvo como máximo 367 dispositivos entre las 19:00 y las 20:00, en el nodo ubicado en el B01.

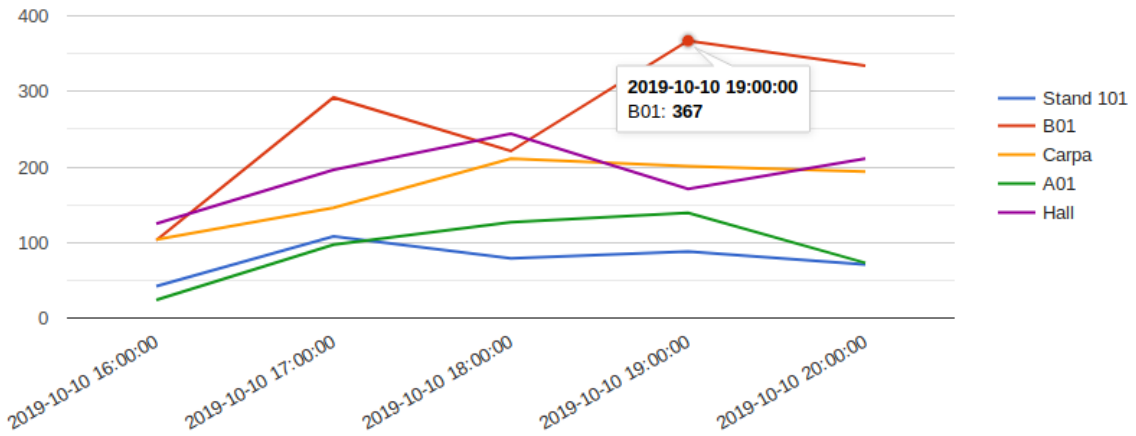

Figura 35: Dispositivos por hora 10/10/2019.

En el segundo día del evento, el máximo de dispositivos fue 317 entre las 18:00 y las 19:00, también en el B01.

El último día del evento tuvo como máximo 345 dispositivos también entre las 18:00 y 19:00 en el B01.

Respecto a la devolución de la gente que visitó el stand, la conclusión más relevante que se puede destacar es el poco conocimiento que existe respecto a la facilidad con la que se puede obtener información desde los celulares, solo por tener la interfaz de Wi-Fi activada.

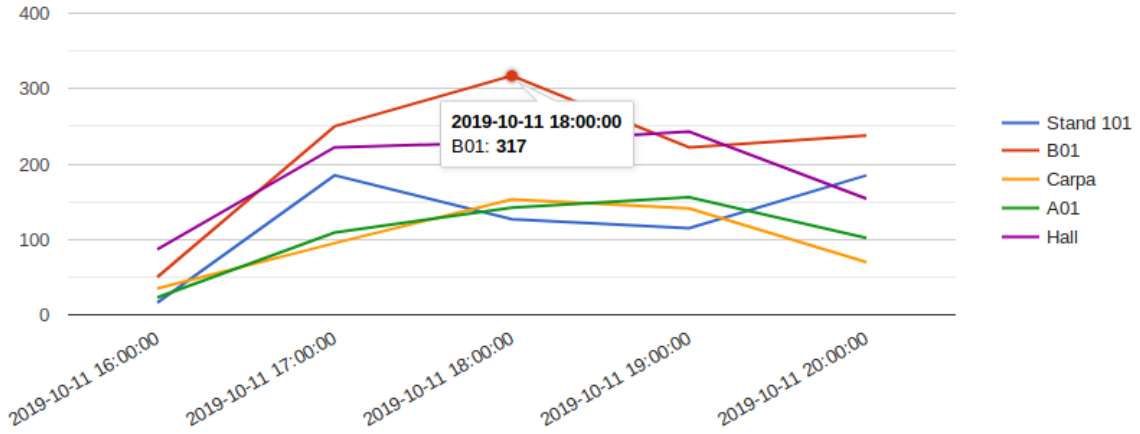

Figura 36: Dispositivos por hora 11/10/2019.

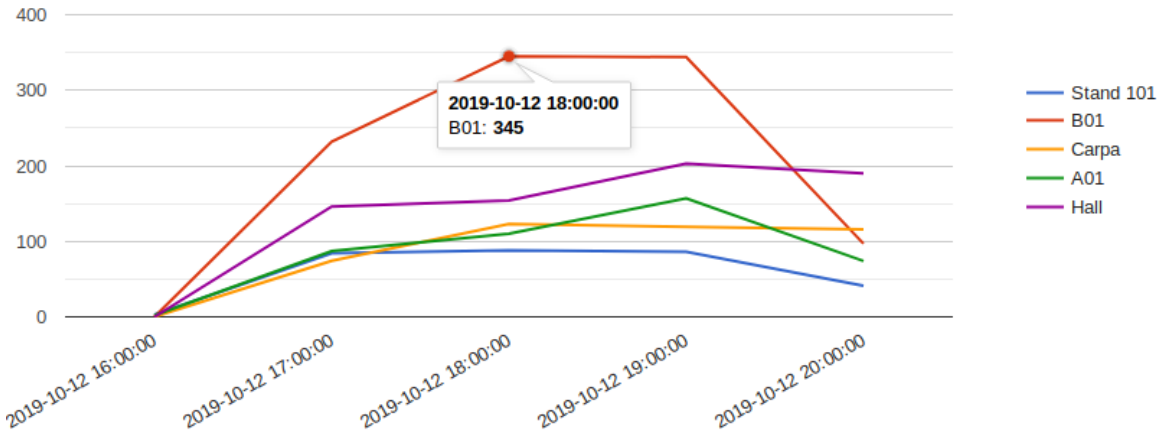

Figura 37: Dispositivos por hora 12/10/2019.

10. Conclusiones

Se logró implementar satisfactoriamente una aplicación IoT de punta a punta, controlando completamente la infraestructura. Es decir, se trabajó en todos los aspectos de la solución de IoT: se implementó en los nodos un sensado de dispositivos inalámbricos mediante la modificación del driver de la interfaz Wi-Fi para permitir la captura de paquetes, se procesó en el mismo nodo los datos sensados, se envió la información mediante LoRaWAN a una infraestructura privada, donde se procesó y desplegó la información en un servidor central. Se trata a su vez de una solución de bajo costo, al ser realizada únicamente con software de código abierto y un único modelo de placa tanto para el nodo como para el gateway.

También se pudo desplegar y presentar el proyecto en el evento Ingeniería deMuestra, cumpliendo satisfactoriamente con los objetivos propuestos.

La detección de dispositivos propuesta se basa en analizar todas las tramas 802.11 y no sólo a las de tipo *probe request* que son las relacionadas al escaneo activo de redes inalámbricas, ya que éstas son esporádicas. Para dicho fin, se colocan las interfaces Wi-Fi de los nodos en modo promiscuo para capturar las tramas, y se identifican las direcciones MAC de los dispositivos inalámbricos a partir de las banderas haciaAP y desdeAP.

El método detecta correctamente los dispositivos al alcance de los nodos sólo si se consideran los dispositivos conectados a una red Wi-Fi, conjuntamente con su traza. Aunque hay que tener en consideración descartar algunos falsos positivos con MAC aleatoria que generaron dichos dispositivos, a raíz de servicios como el de posicionamiento a partir de redes Wi-Fi.

Si se quiere tener una idea más aproximada de la cantidad de dispositivos que existen en un área, se puede colocar la solución para que detecte también los dispositivos con la interfaz Wi-Fi activada pero que no estén conectados a una red. Esto trae 2 situaciones que hay que tener en cuenta. Por un lado, dado que los dispositivos más modernos anonimizan la dirección MAC, es recomendable colocar el parámetro de segundos de escucha por canal en el mínimo posible, para que varíe lo mínimo posible el número de dispositivos detectados respecto al real. Pero esto podría generar no detectar algún dispositivo conectado que en esa ventana de tiempo no envió ningún dato a la infraestructura. También existirán esporádicamente falsos positivos, correspondientes a MAC aleatorias generadas por algún dispositivo conectado.

Si se quiere contar dispositivos de una infraestructura conocida, se puede optimizar la solución reduciendo la casuística de valores de estas banderas, fijando el valor del canal Wi-Fi, y aumentando los segundos de escucha por canal.

LoRaWAN es un protocolo eficaz para el envío de poca información desde nodos alimentados a batería a largas distancias, sensando datos cada lapsos de tiempo espaciados.

El alcance de la señal entre el nodo y el gateway y el ancho de banda de la comunicación afectan la duración de la batería. Por lo que dependiendo del caso de uso hay que fijar los parámetros *bandwidth*, *Spreading Factor* y *Data Rate* teniendo en cuenta cuál se desea priorizar.

Si se cuenta con una aplicación en la cual la arquitectura prevee que no será necesario la comunicación hacia el nodo, un gateway de 1 canal será una solución efectiva y económica. Vale aclarar que otra característica que se afecta con 1 gateway de 1 canal es la seguridad a nivel de claves de cifrado, ya que no se puede utilizar OTAA.

Las placas Sparkfun SPX-14893 demostraron ser versátiles en su uso. Funcionaron correctamente para utilizarlas como nodo y como gateway. En particular el microcontrolador ESP32 cuenta con un framework que facilita la programación al contar con una amplia documentación en su sitio.

Flashear el nodo con un sistema operativo RTOS le da nuevas posibilidades a lo que programación refiere. Se considera una buena opción principalmente cuando se desea prototipar (tanto para probar soluciones como para agregar características), o cuando se tiene un equipo de programadores que no todos son expertos en lenguajes de programación de bajo nivel. Si se cuenta con una aplicación crítica, se recomienda flashear el nodo con el programa específico, para evitar interrupciones de otros procesos, y para minimizar el consumo de energía y memoria.

En particular LuaRTOS demostró potencialidad, pudiéndole agregar satisfactoriamente una función a su biblioteca de Wi-Fi. Pero por otro lado, la falta de documentación de calidad y la dificultad propia de este tipo de sistemas en debugguear hace que la curva de aprendizaje de este sistema sea grande. En este sentido, se tuvo problemas con la implementación de hilos en la solución.

LoraServer es una solución recomendable para implementar la infraestructura LoRaWAN de forma privada. Consume pocos recursos, por lo que es posible instalarla incluso en dispositivos pequeños, es sencilla de usar y tiene la posibilidad de integrarla a otras aplicaciones. Al estar dividida en 3 módulos es posible optimizar la solución dependiendo de la infraestructura con la que se cuenta.

Por otro lado The Things Network, que es una plataforma con un enfoque comunitario, es recomendable cuando no se cuenta con infraestructura propia y existen otros gateway al alcance de los nodos, y se cumple con lo fijado en la *Fair Policy*.

11. Trabajos Futuros

Como trabajos futuros se pueden sugerir los siguientes tópicos.

Se puede extender LuaRTOS para que escanee los beacons de Bluetooth en lugar de tramas Wi-Fi, ya que estos dispositivos (como son pulseras o celulares) emiten permanentemente este tipo de paquetes.

Respecto a la detección de dispositivos mediante tramas 802.11, se puede realizar una mayor discriminación y estudio de las tramas respecto a su tipo y subtipo. Estos campos permitirían obtener información más detallada del tipo de actividad de los dispositivos en las redes.

Otro enfoque de posibles trabajos futuros puede ser el de análisis de datos. Por ejemplo la geolocalización de los dispositivos en espacios interiores con mayor precisión, a partir de la triangulación de las señales detectadas por los nodos.

12. Bibliografía utilizada

Referencias

[1] Gartner, Inc. Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020 (2019) <https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io>(verificado el 21/01/2020)

[2] LoRa Alliance: A technical overview of LoRa® and LoRaWAN™ (2015) <https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf >(verificado el 21/01/2020)

[3] Redondi, Alessandro E.C., Cesana, Matteo: Building up knowledge through passive WiFi probes <https://re.public.polimi.it/retrieve/handle/11311/1043865/257861/1-s2.0-S0140366417302773-main.pdf >(verificado el 21/01/2020)

[4] Kai Li,Chau Yuen, Salil S. Kanhere, Kun Hu, Wei Zhang, Fan Jiang, Xiang Liu: An Experimental Study for Tracking Crowd in Smart Cities <https://www.researchgate.net/publication/329053502_An_Experimental_Study_for_Tracking_Crowd_in_Smart_Cities >

[5] Detta, Federico: Aplicación del análisis de redes al modelado de relaciones personales extraídas de trazas de sistemas de comunicaciones. (2019 - Actividad Integradora Licenciatura en Computación - FING)

[6] Kurose, James F., Ross, Keith W.: Redes de Computadoras: Un enfoque descendente, Capítulo 6.3 (2010)

[7] Privacy: MAC Randomization — Android Open Source Project <https://source.android.com/devices/tech/connect/wifi-mac-randomization>(verificado el 21/01/2020)

[8] About the security content of iOS 8 - Apple Support <https://support.apple.com/en-us/HT201395>(verificado el 21/01/2020)

[9] Ley 18.331 <https://parlamento.gub.uy/documentosyleyes/leyes/ley/18331>(verificado el 21/01/2020)

[10] European Commission - Data protection in the EU<https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en >(verificado el 21/01/2020)

[11] European Commission - What constitutes data processing? <https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-constitutes-data-processing_en>(verificado el 21/01/2020)

[12] Semtech: LoRa Alliance — Ecosystem — Semtech LoRa Technology — Semtech <https://www.semtech.com/lora/ecosystem/lora-alliance>(verificado el 21/01/2020)

[13] URSEC: Asignación de frecuencia para IMT <https://www.gub.uy/unidad-reguladora-servicios-comunicaciones/datos-y-estadisticas/datos/asignacion-frecuencia-para-imt >(verificado el 16/02/2020)

[14] Why LoRa? — Semtech LoRa Technology — Semtech <https://www.semtech.com/lora/why-lora>(verificado el 21/01/2020)

[15] Semtech Corporation: Lora Modulation Basics (2015) <http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf>(verificado el 21/01/2020)

[16] LoRaWAN® distance world record broken, twice. 766 km (476 miles) using 25mW transmission power <https://www.thethingsnetwork.org/article/lorawan-distance-world-record>(verificado el 21/01/2020)

[17] DigiKey: LoRaWAN Part 1: How to Get 15 km Wireless <https://www.digikey.com/en/articles/techzone/2016/nov/lorawan-part-1-15-km-wireless-10-year-battery-life-iot >(verificado el 21/01/2020)

[18] About LoRaWAN® — LoRa Alliance® <https://lora-alliance.org/about-lorawan >(verificado el 21/01/2020)

[19] Lora Alliance: LoRaWAN 1.1 Regional Parameters <https://lora-alliance.org/resource-hub/lorawanr-regional-parameters-v11rb >(verificado el 21/01/2020)

[20] Radio link budget calculator: Wi-fi antennas Jirous.com <http://en.jirous.com/calculation-wifi >(verificado el 21/01/2020)

[21] ETSI: EN300.220, capítulo 7.2.3 (2012) <https://www.etsi.org/deliver/etsi_en/300200_300299/30022001/02.04.01_40/en_30022001v020401o.pdf >(verificado el 21/01/2020)

[22] Duty Cycle — The Things Network <https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html >(verificado el 21/01/2020)

[23] Architecture - LoRa Server, open-source LoRaWAN network-server <https://www.loraserver.io/overview/architecture/>(verificado el 21/01/2020)

[24] ESP32 LoRa 1-Channel Gateway - SPX-14893 - SparkFun Electronics <https://www.sparkfun.com/products/retired/14893 >(verificado el 21/01/2020)

[25] G. Tejera, G. Amorin, A. Sere, N. Capricho, P. Margenat and J. Visca, Robotito: programming robots from preschool to undergraduate school level,"2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2019, pp. 296-301.

[26] GitHub - whitecatboard/Lua-RTOS-ESP32: Lua RTOS for ESP32 <https://github.com/whitecatboard/Lua-RTOS-ESP32 >(verificado el 21/01/2020)

[27] Kurose, James F., Ross, Keith W.: Redes de Computadoras: Un enfoque descendente, Capítulo 5.4.1 (2010)

[28] Nivel Technologies Ltd.: "The Simplest MAC Vendor Lookup API", *MACVendors.com* (2019) <https://macvendors.com/api>(verificado el 21/01/2020)

[29] Net module · whitecatboard/Lua-RTOS-ESP32 Wiki · GitHub <https://github.com/whitecatboard/Lua-RTOS-ESP32/wiki/Net-module#wi-fi>(verificado el 21/01/2020)

[30] Non-volatile storage library — ESP-IDF Programming Guide v4.1-dev-802-ga45e99853 documentation <https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/storage/nvs_flash.html>(verificado el 21/01/2020)

[31] Google Developers <https://developers.google.com/chart>(verificado el 21/01/2020)

[32] Ingeniería deMuestra (2019) <http://idm.fing.edu.uy/>(verificado el 21/01/2020)

[33] Single Channel LoRaWAN Gateway <https://github.com/things4u/ESP-1ch-Gateway-v5.0/>(verificado el 21/01/2020)

13. Apéndices

Se presenta a continuación un conjunto de guías de instalación para las distintas soluciones. El sistema operativo utilizado es Linux Ubuntu 18.04.

13.1. Instalación postgreSQL

Para instalar postgreSQL hay que ejecutar los siguientes comandos:

```
sudo apt-get update
sudo apt-get install postgresql postgresql-contrib
apt install pgadmin3
```

PgAdmin es un programa que permite navegar y consultar las distintas tablas de las bases de datos.

13.2. Instalación Apache

Para instalar apache hay que ejecutar:

```
sudo apt install apache2
```

Luego hay que habilitar el firewall para que permita la comunicación con el puerto, el puerto por defecto es el 80. Para agregar esta configuración hay que ejecutar:

```
sudo ufw app list
```

Allí se pueden ver los perfiles, y para solo autorizar el puerto 80 hay que ejecutar:

```
sudo ufw allow 'Apache'
```

Los archivos que serán publicados están en el carpeta */var/www/html*, todos los archivos de configuración de Apache están en la carpeta */etc/apache2*

13.3. Instalación PHP

Para instalar PHP hay que ejecutar:

```
sudo apt-get install php libapache2-mod-php
```

También hay que instalar el conector de PHP para ejecutar consultas en la base de datos de postgresql

```
sudo apt-get install php-pgsql
```

Luego hay que reiniciar el servicio, mediante el comando:

```
sudo systemctl restart apache2
```

13.4. LoraServer

Si bien para instalar LoraServer se siguieron los pasos sugeridos en la página del proveedor, se detallan a continuación junto a la parametrización realizada. En la configuración *dsn* es el acrónimo de "data source name" y *hal* de "hardware abstraction layer".

Primero hay que instalar el broker de MQTT mosquitto, que será utilizado por el bridge y el server, ejecutando el comando:

```
sudo apt-get install mosquitto
```

13.4.1. LoraServer Bridge

Primero hay que agregar el repositorio:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
sudo echo "deb https://artifacts.loraserver.io/packages/3.x/deb stable main" |
sudo tee /etc/apt/sources.list.d/loraserver.list
sudo apt-get update
```

Luego se instala el bridge:

```
sudo apt-get install lora-gateway-bridge
```

Para iniciar el servicio se ejecuta:

```
sudo systemctl start lora-gateway-bridge
```

El archivo de configuración se encuentra en */etc/lora-gateway-bridge/lora-gateway-bridge.toml*. Si bien este archivo no se cambió, allí puede modificarse si se desea el puerto en el que escuchará el bridge y la configuración de la integración MQTT.

13.4.2. Instalación de LoraServer

Primero hay que instalar Redis, donde el Server guarda todos los datos no persistentes. Para ello hay que ejecutar:

```
sudo apt-get install redis-server
```

En PostgreSQL hay que crear una base de datos para el Server, para ello hay que iniciarlo como usuario *postgres* y crear en este caso el usuario *loraserver_ns* con contraseña *dbpassword*:

```
sudo -u postgres psql
create role loraserver_ns with login password 'dbpassword';
create database loraserver_ns with owner loraserver_ns;
```

Para verificar que el usuario y base de datos se creó correctamente se puede ejecutar:

```
psql -h localhost -U loraserver_ns -W loraserver_ns
```

Luego hay que agregar el repositorio de loraserver:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
sudo echo "deb https://artifacts.loraserver.io/packages/3.x/deb stable main" |
sudo tee /etc/apt/sources.list.d/loraserver.list
sudo apt-get update
```

En este momento, se puede instalar LoraServer mediante:

```
sudo apt-get install loraserver
```

Luego hay que modificar el archivo de configuración, alojado en */etc/loraserver/loraserver.toml*.

- *dsn* colocar la conexión a la base de datos, con el usuario y contraseña creados: dsn="postgres://loraserver_ns:dbpassword@localhost/loraserver_ns?sslmode=disable"
- *network_server.band*: Colocarla en *US_902_928* , y comentar los extra channels, son xa la banda EU.
- *timezone*: se puede dejar en "Local" para que tome la zona horaria del sistema, o se puede colocar la zona horaria donde se utilizará la aplicación.

Para reiniciar el servicio se ejecuta:

```
sudo systemctl restart loraserver
```

13.4.3. LoraServer App Server

Primero hay que crear la base de datos y el usuario y contraseña para este servicio:

```
sudo -u postgres psql
create role loraserver_as with login password 'dbpassword';
create database loraserver_as with owner loraserver_as;
```

Luego hay que habilitar las extensiones pg_trgm y hstore para esta base de datos:

```
loraserver_as
create extension pg_trgm;
create extension hstore;
```

Para verificar que el usuario y la base de datos se creó correctamente se puede ejecutar:

```
psql -h localhost -U loraserver_as -W loraserver_as
```

En este momento se agrega el repositorio del App Server y se instala:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
sudo echo "deb https://artifacts.loraserver.io/packages/3.x/deb
stable main" | sudo tee /etc/apt/sources.list.d/loraserver.list
sudo apt-get update
sudo apt-get install lora-app-server
```

Para iniciarlo se ejecuta:

```
sudo systemctl start lora-app-server
```

El archivo de configuración es */etc/lora-app-server/lora-app-server.toml*, allí hay que cambiar:

- *dsn* modificarlo con la base de datos y usuario creado: "postgres://loraserver_as:dbpassword@localhost/loraserver_as?sslmode=disable"
- *jwt_secret*: se coloca el código generado de ejecutar "openssl rand -base64 32"

13.4.4. Configurar LoraServer App

A nivel parametrización del App Server, se hicieron los siguientes pasos:

En *Network-servers*: se agrega un nombre y la direccion del LoRa Server, en este caso "localhost:8000".

En *Gateway-profiles*: se agrega un perfil indicando los canales que se estaran escuchando para la banda de la región, y en network server el creado en el paso anterior.

En *Organizations*: hay que agregar un nombre para la organización

Luego hay que configurar la sección debajo de estas opciones, eligiendo la organización actual recién creada.

En *Service-profiles* agregar un perfil, en este caso se le agrego la metadata del gateway, el Device request frequency, y los datarate se dejaron en 0.

En *Device-profiles* se agrega un perfil para los dispositivos, en este caso se configuró que tenga version de LoRaWAN MAC 1.0.1, de clase A, y en la pestania join se fijo la frecuencia para la recepcion y envio.

En *Gateways*: Se agrega el gateway con id, la forma más sencilla de obtenerlo es accediendo a la interfaz web de él

En *Applications*: Se crea una aplicación, y se le agregan los dispositivos. El EUi se obtiene de los dispositivos mediante el comando "lora.getDevEui()". Es importante clickear en "Disable Frame couter", para no validar la correlación de mensajes, ya que al resetearse el nodo el contador vuelve a 0, y sino no se reciben los nuevos mensajes. Luego en *Activation* el DevAddr se puede sacar del nodo, y el resto se puede generar aleatoriamente o elegirlos. Al finalizar hay que clickear en "reativate".

En *integrations*, se agrega una integración HTTP, colocando en uplink la página que recibirá los mensajes (en este caso http://server/insertcount.php).

13.5. Instalación de Lua-RTOS en Sparkfun

Es recomendable antes de iniciar los procedimientos contar con el permiso de escritura sobre el puerto USB en Linux.Para ello hay que ejecutar el siguiente comando:

```
sudo usermod -a -G dialout \$USER
```

ESP32 toolchain

Hay que instalar los siguientes paquetes:

```
sudo apt-get install git wget flex bison gperf python python-pip
python-setuptools python-serial python-click python-cryptography
python-future python-pyparsing python-pyelftools cmake
ninja-build ccache
```

Framework esp-idf

Hay que clonar el repositorio de git:

```
git clone --recursive https://github.com/espressif/esp-idf.git
```

Lua-RTOS

Luego hay que clonar el repositorio de Whitecat

```
git clone --recursive https://github.com/whitecatboard/Lua-RTOS-ESP32
```

Agregar placa Sparkfun

Hay que agregar la placa Sparkfun para poder flashear el sistema operativo. Estos pasos se basan en https://github.com/whitecatboard/Lua-RTOS-ESP32/wiki/How-to-add-a-new-board.

Primero se obtiene el parámetro vid:pid ejecutando:

```
python -m serial.tools.list_ports -v
```

Luego, considerando el vid:pid con valor "1A86:7523", hay que agregar al archivo boards/supported_boards.json la siguiente entrada:

```
{
        "description": "Sparkfun",
        "manufacturer": "Generic",
        "image": "",
        "usb_vid_pid": "1A86:7523",
        "usb_port_exp": "",
        "firmwares": [
              {
                    "id": "sparkfun",
                    "description": "Sparkfun",
                    "filesystem": "default"
              }
        ]
}
```

Luego hay que copiar el archivo *sdkconfig* a la carpeta boards y ejecutar los siguientes comandos para preparar el firmware para la nueva placa:

```
make clean
python boards/gen_info.py
```

En este punto hay que ejecutar "make -j5" y elegir la nueva placa. Luego ejecutar "make menuconfig" y configurar la nueva placa.

Flash Sparkfun

Por último hay que flashear la placa con el sistema operativo:

```
make flash
```

Configuración environment

Hay que cambiar en el archivo env los path de los repositorios descargados:

```
export PATH=\$PATH:~/esp/xtensa-esp32-elf/bin
export IDF_PATH=~/esp/esp-idf
```

Conexión serial a la placa

Para conectarse a la placa hay que ejecutar en la carpeta de LuaRTOS:

```
source ./env
```

Y luego para conectarse por consola:

```
picocom  --baud 115200 /dev/ttyUSB0
```

13.6. Activación de The Things Gateway

Para registrar este gateway hay que acceder a "https://activate.thethingsnetwork.org/" y clickear en *start*. Luego hay que ingresar un id y en el *freq plan* seleccionar "united states 915hmz".

En ese momento, sin cerrar el navegador, hay que conectarse a la red Wifi con nombre de la forma "Things-Gateway-XXXX", con clave "thethings", y completar la navegación.

13.7. Parametrización Gateway SPX-14893

Para configurar este gateway hay que modificar los siguientes archivos del software recomendado por Sparkfun para esta función [33].

- En el archivo **esp-sc-gway.h** hay que modificar las siguientes líneas:
  - **#define _LFREQ 868** modificar la banda, en este caso 915
  - **#define _SPREADING SF9** colocar el spreading factor seleccionado para transmitir desde los nodos
  - **#define _CAD 1** si no se va a utilizar este modo, o sea si todos los nodos transmitirán en el mismo SF, colocarlo en 0.
  - **#define _PIN_OUT 1** cambiarlo a 4 para la placa sparkfun.
  - **#define _TTNSERVER "router.eu.thethings.network"** cambiar la dirección por la dirección o IP del Bridge de LoRa. También se puede modificar el puerto mediante el parámetro _TTNPORT en la línea anterior
  - **// Gateway Ident definitions** bajo este título se puede modificar información relativa a la descripción que queremos darle a nuestro gateway
  - **#define NTP_TIMEZONES 2** acá hay que modificar la diferencia con UTC de la zona donde se encuentra el gateway, para el caso de Uruguay ese valor es -3. En la línea anterior es posible cambiar el servidor al que sincroniza la hora.
  - **wpas wpa[] = {{ "" , "" },{ "fire", "water"}, { "ape", "beer"}};** aca dejar {"" ,""} tanto como primer elemento como tercero, y poner como segundo {"nombre" ,"contraseña"} siendo nombre el nombre de la red inalámbrica y contraseña su contraseña.
- En el archivo loraModem.h:

- **uint32_t freq = freqs[0];** En esta línea se elije cuál de las frecuencias se utilizará de los array creados en las lineas superiores, según la banda. Si la frencuencia no se encuentra en el array se puede modificar el mismo y también el índice seleccionado de freqs[]. En el caso de este trabajo se dejaron 3 frecuencias para poder probar la funcionalidad de HOP, ya que ese es el mínimo número de frecuencias para ser fully LoRa Compliant, según la documentación del desarrollador del software.

13.8. Puntos de control

Dada la cantidad de posibles puntos de falla se decidió recopilar en una sección cómo es posible "debugguear" las comunicaciones desde el gateway a la aplicación, principalmente para cuando algo no está funcionando como debería.

Gateway

Se puede acceder a la consola web del gateway en la IP del gateway, puerto 80. Allí se puede consultar información de los paquetes recibidos tal como se muestra en la Figura 38, así como configurar algunos datos. Existe un modo experto y la posibilidad de consultar el log, donde se pueden ver todos los mensajes que procesó el gateway, tal como se muestra en la Figura 39.

ESP Gateway Config

Version: V5.3.3.H; 180825a
ESP alive since Friday 6-12-2019 05:06:34, Uptime: 0-00:09:31
Current time Friday 6-12-2019 05:15:51

Documentation | Expert Mode | Log Files

Package Statistics

Counter	C 0	C 1	C 2	Pkgs	Pkgs/hr
Packages Downlink				0	
Packages Uplink Total				55	355
Packages Uplink OK				52	
SF7 rcvd	0	0	0	0	0 %
SF8 rcvd	0	0	0	0	0 %
SF9 rcvd	0	0	0	0	0 %
SF10 rcvd	52	0	0	52	94 %
SF11 rcvd	0	0	0	0	0 %
SF12 rcvd	0	0	0	0	0 %

Message History

Time	Node	C	Freq	SF	pRSSI
Friday 6-12-2019 05:15:47	26 02 1d a2	0	923300000	10	-57
Friday 6-12-2019 05:15:43	26 02 1d a2	0	923300000	10	-55
Friday 6-12-2019 05:15:03	26 02 1d 6b	0	923300000	10	-35
Friday 6-12-2019 05:15:03	26 02 1d a1	0	923300000	10	-69
Friday 6-12-2019 05:15:00	26 02 1d 6b	0	923300000	10	-34
Friday 6-12-2019 05:14:55	26 02 1d 6b	0	923300000	10	-35

Figura 38: Interfaz web gateway ESP32.

```
{"rxpk":
[{"tmst":100574090,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":9,"rssi":-77,"size":32,"data":"QGsdAiYACgABHGkHH1dMgoh+EkkE5ESV6N3WmWavTOE="}]}
{"rxpk":
[{"tmst":105880339,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":12,"rssi":-56,"size":32,"data":"QGsdAiZAGQABVw4t5ykUaVL/1/tCdE45d9VzkMw1MMQ="}]}
{"rxpk":
[{"tmst":107233475,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":13,"rssi":-35,"size":32,"data":"QGsdAiYACQABh93SfopTbHPLBUjIYx6fMtrSzoW6UlA="}]}
{"rxpk":
[{"tmst":107746237,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":10,"rssi":-76,"size":32,"data":"QGsdAiYADAABwCZmzgQLZ0cSGBK21CWdaw0yTLx75G4="}]}
{"rxpk":
[{"tmst":110203844,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":13,"rssi":-56,"size":32,"data":"QGsdAiZAGgAB16j5p7N8K8U0pSd27WTAIPj2cPlN4/g="}]}
{"rxpk":
[{"tmst":114703542,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":13,"rssi":-56,"size":32,"data":"QGsdAiZAGwAB3AxYB6yM83BgZglreTjzFH+ANGWbPDU="}]}
{"rxpk":
[{"tmst":173528953,"chan":0,"rfch":0,"freq":923.299987,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":14,"rssi":-35,"size":32,"data":"QGsdAiYADAABwCZqxnd5EDMebWGy1lKdaw02TEg5jO8="}]}
```


Figura 39: Log Gateway ESP32.

Comunicación Gateway - Lora Gateway Bridge

Para ver que paquetes está enviando el gateway, se puede utilizar un analizador como Wireshark para capturar los paquetes udp al puerto 1700 de la IP del bridge.

Lora Gateway Bridge

Para ver el log de mensajes del Gateway Bridge y conocer si se están publicando los mensajes MQTT correctamente, así como el estado del servicio en general, se puede ejecutar el siguiente comando: “journalctl -u lora-gateway-bridge -f -n 50”

Lora Server

Para el caso del log del Lora Server el comando es: “journalctl -u loraserver -f -n 50”

Lora App Server

Y finalmente para el App Server: “journalctl -u lora-app-server -f -n 50”

Integración HTTP Server - Aplicación

Esta comunicación se puede analizar mediante Wireshark al ver los POST de la comunicación HTTP entre la IP del server y la de la aplicación. Allí se puede ver el mensaje JSON tal como el de la Figura 23

Aplicación web

Para ver el log tanto del gestor de mensajes como los de las páginas web, se pueden analizar los logs de apache, que se encuentra en */var/log/apache2/error.log*. En la misma carpeta también se encuentra el archivo *access.log*, que tiene el registro de los accesos satisfactorios a las páginas.