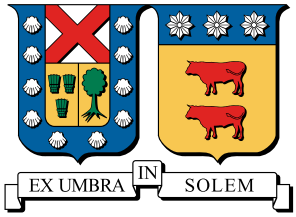
2018

GESTIÓN DE ACTIVOS FIJOS MÓVILES MEDIANTE EL USO DE TECNOLOGÍA INALÁMBRICA

GONZÁLEZ BURNS, EDUARDO FELIPE

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



GESTIÓN DE ACTIVOS FIJOS MÓVILES MEDIANTE EL USO DE TECNOLOGÍA INALÁMBRICA

EDUARDO FELIPE GONZÁLEZ BURNS

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

PROFESOR GUÍA : JAVIER CAÑAS ROBLES
PROFESOR CORREFERENTE : MAURICIO ARAYA LÓPEZ

Enero - 2018

Dedico este trabajo a mi familia, por su apoyo incondicional en todo momento de mi vida ...

Agradecimientos

En primer lugar agradecer a mis padres y hermanos por todo el apoyo que me brindaron durante los largos años de mi desarrollo profesional. Sin ellos no hubiese podido plantearme este desafío personal y no estaría en donde estoy hoy en día.

A Daniela por su amor y apoyo incondicional durante todos estos años, compartiendo conmigo todo el cansancio y sacrificio que esto conlleva.

A mis amigos, tanto de la infancia como aquellos que conocí en la Universidad, por su apoyo, preocupación y compañerismo frente a los desafíos que se nos pusieron en frente.

Al profesor Javier Cañas por aceptar ser mi profesor guía en este trabajo y orientarme durante todo el proceso.

Muchísimas Gracias

Resumen

En este trabajo se presenta la investigación acerca de tecnologías inalámbricas que pueden ayudar a ubicar, controlar y gestionar los activos fijos móviles de una empresa sin hacer uso del Sistema de Posicionamiento Global (*GPS*)[1]. Mediante el uso de tecnología *Bluetooth BLE* se implementó un prototipo funcional que permite obtener la ubicación, aproximada, de un activo basado en la detección su dirección *MAC* y la intensidad de señal recibida. A continuación se desarrolló una *API* que recibe los datos recolectados y alimenta un *Dashboard* en donde se puede encontrar el estado de visibilidad de cada activo junto con el registro de sus últimos movimientos.
Palabras Clave: Gestión de Activos Fijos, GPS, RFID Tags, Bluetooth Low Energy,ZigBee, Node.js, LoopBack.

Abstract

This work shows the investigation of wireless technologies that could help to find, control and manage the mobile fixed assets of a business without using the Global Positioning System. A prototype capable of estimate the location of a fixed asset was implemented using the Bluetooth BLE technology, this was possible analyzing the MAC Address and the signal strength. Following this, an API responsible of receiveing data and send it to a dashboard was implemented. The dashboard shows the actual visibility state of the assets and the logs associated.
Keywords Fixed Assets Management, GPS, RFID Tags, Bluetooth Low Energy, ZigBee, Node.js, LoopBack.

Glosario

- **API:** *Application Programming Interface* Conjunto de funciones y procedimientos diseñados para ser utilizados por otro software como una capa de abstracción.
- **BLE:** *Bluetooth Low Energy* Protocolo inalámbrico Bluetooth de bajo consumo.
- **Backend:** Capa de procesamiento de datos recolectados por el front end.
- **Endpoint:** Punto de entrada a un servicio o proceso. Particuarlmente se le denomiará *endpoint* a la dirección web a la cual se accede a la *API*
- **Framework:** Abstracción de software que ofrece funciones genéricas que permite la implementación de funciones creadas por el usuario.
- **Frontend:** Capa de recolección de datos por parte del usuario, corresponde a la interfaz de usuario.
- **GPS:** *Global Positioning System* Sistema de posicionamiento global mediante el uso de señales satelitales.
- **HTTP:** *Hypertext Transfer Protocol* Protocolo base de la comunicación de la *World Wide Web*.
- **JSON:** *JavaScript Object Notation* Formato de archivo que permite expresar objetos de datos como texto.
- **MAC Address:** Identificador único que poseen las interfaces inalámbricas
- **REST:** *Representational state transfer* Interfaz de comunicación entre sistemas usando prorocolo *HTTP*.
- **RFID:** *Radio Frequency Identification* Sistema de almacenamiento y recuperación de datos utilizados principalmente para identificar un objeto de manera inalámbrica.
- **RPi:** *Raspberry Pi* Serie de computadoras de una sola placa de bajo costo.
- **Timestamp:** Secuencia de caracteres codificados para representar cuándo un evento ocurrió.
- **Vista:** Una pantalla de una aplicación.
- **WPAN:** *Wireless Personal Area Network* Red computacional usada para transmitir datos entre dispositivos digitales de asistencia personal como teléfonos, tables, etc.

Índice General

Índice de Figuras

Introducción

En la actualidad, la adecuada gestión de los activos fijos es fundamental para una empresa. Optimizar el uso de cada uno de sus recursos es una tarea clave para el buen funcionamiento de ésta y sin embargo, no es una tarea sencilla.

A este problema hay que sumarle la gestión y control de los activos fijos que, además, suelen cambiar su posición geográfica constantemente, como lo son las herramientas, maquinaria de trabajo, automóviles, accesorios, etc. Tener un control sobre estos activos resulta particularmente complejo debido a que su uso suele ser muy variado durante su vida útil, por lo que casi necesariamente hay que asignar personal dedicado a monitorear el uso y ubicación de éstos, siendo ésta el área a cubrir por el trabajo de esta memoria.

Las soluciones actuales en el mercado se basan principalmente en el sistema de posicionamiento global (GPS) para monitorear la ubicación del activo fijo móvil con un margen de error de unos cuantos metros. Sin embargo, este sistema tiene ciertos problemas. En primer lugar, la mayoría de los sistemas existentes en el mercado utilizan dispositivos que no poseen fuente de alimentación eléctrica propia, por lo que el activo es quien debe proporcionales la energía para operar. En segundo lugar, la transmisión de los datos obtenidos por el sistema debe ser enviado a través del uso de las antenas de telefonía celular, lo que supone un gasto adicional para la empresa. Finalmente, el tamaño físico de los dispositivos GPS hace que no sea sencillo adherirlos a cualquier tipo de activo fijo móvil, esto sin considerar su elevado costo en el mercado.

La mala gestión de los activos fijos móviles supone para la empresa una serie de gastos innecesarios. Por una parte no se utilizan de manera adecuada, significando que no se optimice la capacidad productiva de éstos, mientras que por otra parte, el bajo control sobre los equipos puede generar problemas como el uso indebido o incluso los robos de éstos.

Este documento pretende explorar sobre las tecnologías existentes, sus usos y aplicaciones, su comparación entre sí y proponer técnicas y métodos para resolver el problema planteado de la mejor forma posible sin recurrir al uso de tecnologías satelitales.

El documento ha sido estructurado en los siguientes capítulos:

Capítulo 1 Definición del Problema: Presenta el problema a tratar, una introducción al contexto en el que se desenvuelve, y los objetivos que se buscan abarcar.

Capítulo 2 Estado del Arte: Se presentan las técnicas existentes para la solución a este problema como tecnologías con el potencial de ser una solución.

Capítulo 3 Diseño y Construcción de la Solución: Se presenta el análisis de las tecnologías existentes y se presenta una propuesta de solución en base a los resultados obtenidos.

Capítulo 4 Validación y Análisis de Resultados: Se muestran los resultados obtenidos mediante la implementación de la solución propuesta.

Capítulo 5 Conclusiones: Finalmente, se entregan las conclusiones obtenidas luego de la realización de este trabajo.

Capítulo 1

Definición del Problema

1.1. Contexto

Es común que las empresas dependan en gran parte de sus activos fijos para maximizar su operación, tales como maquinarias especializadas, vehículos, computadoras, tecnologías personalizadas, etc. Sin embargo es común observar que estas empresas no destinan recursos a la correcta gestión de sus activos ni al control de su buen uso. Es común ver en rubros como la construcción casos en el que los jefes de obras no tienen control sobre la maquinaria que entra y sale de sus bodegas y por tanto son propensos a perder tiempo tratando de ubicar sus activos llegando al caso de presentarse hurtos de equipo valioso.

Esta situación se replica en varios rubros, generando una mala gestión de los activos, afectando la utilidad misma de éstos junto a la reducción de productividad de la empresa.

1.2. Identificación del Problema

Administrar y gestionar activos fijos es una tarea crucial para el correcto funcionamiento de una empresa, la mala gestión de éstos genera costos que podrían ser reducidos o eliminados. Cuando el activo a gestionar es móvil esta tarea se complica aún más.

En la actualidad una de las tecnologías existentes para solventar este problema es la tecnología *GPS*, la cual requiere recibir la señal de al menos 3 satélites para estimar su ubicación y posteriormente debe enviar esos datos hacia algún servidor, generalmente haciendo uso de redes celulares, lo que genera un costo adicional.

El costo de implementar esta tecnología no es menor y además depende de la recepción de la señal satelital y la de las redes celulares, por lo que su funcionamiento es prácticamente nulo en algunas condiciones, como por ejemplo, bajo tierra.

Es necesario encontrar soluciones alternativas que no dependan de estos servicios y que además sean menos costosas para que no solamente las grandes empresas las puedan aplicar.

1.3. Objetivos de la Solución

Objetivo General

Proponer un sistema que permita monitorear la ubicación aproximada de los activos fijos móviles mediante el uso de tecnologías inalámbricas sin hacer uso de tecnología GPS.

Objetivos Específicos

- Analizar las diferentes tecnologías que podrían cumplir el objetivo general, estudiando su *pros* y sus *contras*.
- Proponer técnicas que permitan obtener información útil para la gestión a partir de los resultados obtenidos.
- Crear un prototipo funcional que permita determinar los beneficios de la solución propuesta.

Capítulo 2

Estado del Arte

Tras un breve análisis se determinó que el problema a resolver se divide principalmente en tres etapas.

1. Obtener la ubicación aproximada de un activo fijo.
2. Transmitir los datos recolectados a un servidor central.
3. Implementar el servidor central en cuestión, el cual realizará el procesamiento de los datos obtenidos.

A continuación se presentarán algunas de las tecnologías que podrían ser utilizadas para implementar la solución al problema planteado.

2.1. Tecnologías de Localización

En esta sección se presentarán algunas tecnologías que permiten estimar la ubicación de un activo fijo.

Para poder comparar estas tecnologías se fijaron cinco criterios de evaluación, los cuales se definen de la siguiente manera.

- Autonomía: Corresponde al tiempo durante el cual la tecnología puede operar sin una fuente de corriente eléctrica externa:
  - Alta: Más de 6 meses.
  - Media: Más de 3 meses y menos de 6 meses.
  - Baja: Menos de 3 meses.
- Precisión: Corresponde al nivel de error, en distancia, entre la ubicación real y la ubicación determinada por la tecnología. Se clasifica de la siguiente forma.
  - Alta: Menos de 10 metros de error.
  - Media: Menos de 60 metros y más de 10 metros de error.
  - Baja: Más de 60 metros de error.
- Costo: Corresponde el al costo, aproximado, que tiene implementar la tecnología en un activo fijo. Se clasifica de la siguiente forma.
  - Alto: Sobre los 100 USD.

  - Medio: Sobre los 50 USD y bajo los 100 USD.
  - Bajo: Bajo los 50 USD.
- Ventajas: Principales ventajas de la tecnología en comparación a las otras presentadas.
- Desventajas: Principales desventajas de la tecnología en comparación a las otras presentadas.

GPS

Debido a que la solución a presentar debe ser capaz de cumplir una función similar a la que provee la tecnología *GPS*, es que se comenzará explicando cómo opera esta tecnología y en qué se basa para determinar la ubicación de un objeto.

El sistema de posicionamiento global, *GPS* por sus siglas en inglés, es un sistema que permite determinar en toda la Tierra la posición de un objeto con una precisión de hasta centímetros. Fue desarrollado por el Departamento de Defensa de los Estados Unidos [1] y utiliza 24 satélites para utilizar trilateración [2]

Los satélites en órbita poseen trayectorias sincronizadas de manera tal que cubren toda la superficie de la Tierra. Para determinar la posición, el receptor debe localizar automáticamente como mínimo cuatro satélites de la red, recibiendo una señal indicando la identificación y la hora del reloj de cada uno de ellos. En base a las señales recibidas, el receptor sincroniza el reloj del *GPS* y calcula el tiempo que tardan en llegar las señales al equipo, de manera tal que calcula la distancia al satélite haciendo uso de la técnica de trilateración inversa. Esta técnica consiste en determinar la distancia de cada satélite al punto de medición, una vez conocidas las distancias se puede determinar la posición relativa respecto a los satélites. Además, gracias a las señales enviadas por los satélites, se conoce la posición absoluta de cada uno de ellos, con lo que se obtiene las coordenadas reales del punto de medición [2].

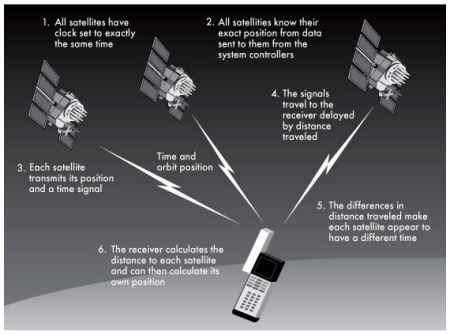


Figura 2.1: How Does GPS Work

[1] United States Department of Defense - `http://www.defense.gov`
[2] How Does GPS Work - `http://www.bestforhunting.com/2011/06/how-does-gps-work/`

La precisión del sistema *GPS* de acceso público se muestra en la siguiente tabla[3].

Precisión en el Plano Horizontal	Precisión de la Altitud
Nivel 50.00 % confianza: 2,5 metros	Nivel 50.00 % confianza: 5,6 metros
Nivel 68.27 % confianza: 3,8 metros	Nivel 68.27 % confianza: 7,4 metros
Nivel 95.45 % confianza: 7,0 metros	Nivel 95.45 % confianza: 14,4 metros
Nivel 99.73 % confianza: 9,8 metros	Nivel 99.73 % confianza: 21,3 metros

Cuadro 2.1: Precisión Experimental de los GPS

Profundizando en el tema de control de flotas mediante el uso de *GPS*, el paper "Real-time tracking management system using GPS, GPRS and Google earth" [3] habla sobre cómo el correcto monitoreo de los vehículos de una empresa puede generar diversos ahorros mejorando la logística de distribución y teniendo un mayor control sobre la ubicación de cada uno de sus activos fijos móviles. El sistema planteado hace uso de un receptor *GPS* conectado a un microcontrolador que posee un transmisor *GPRS* [4]. Cada vez que puede, el transmisor envía la ubicación obtenida por el receptor *GPS* al servidor, quien guarda los datos en una Base de Datos.

Figura 2.2: Transmisión de ubicación al servidor [3]

De esta forma es posible determinar el trayecto que ha recorrido cada vehículo de la flota y así saber si se está haciendo un uso adecuado del activo, si se ha cumplido el kilometraje determinado antes de una mantención o si simplemente todo está en orden. El paper está enfocado a cruzar los datos obtenidos con los datos disponibles en Google Earth [4] y de esta forma mostrar los trayectos recorridos sobre un mapa satelital real.

Para efectos de gestión el margen de error de esta tecnología es aceptable para determinar la ubicación de algunos activos, es por este motivo que el sistema *GPS* es el más popular a la hora de monitorear los vehículos de las empresas. Sin embargo, uno de los mayores inconvenientes de estos sistemas *GPS* es que no cuentan con un sistema de alimentación eléctrica propio y es el activo quien debe suministrarle esta energía, es por esto que estos sistemas están pensados para activos principalmente motorizados.

Además, si bien el receptor es capaz de determinar su ubicación, éste debe utilizar otro tipo de tecnología para enviar estos datos al servidor central o computador encargado de gestionar el activo, lo que supone un gasto adicional para la empresa.

[3]Precision de los GPS - `http://www.elgps.com/documentos/consejos/PrecisionGarmin.html`
[4]Google Earth - `https://www.google.es/intl/es/earth/`

A continuación se presentará la tabla resumen para la tecnología GPS considerando una versión con batería y que transmite los datos a través de una red inalámbrica telefónica.

Autonomía	Baja
Precisión	Alta
Costo	Alto
Ventajas	Ofrece una gran precisión con un error estimado de unos pocos metros.
Desventajas	Los modelos que no dependen de una fuente de corriente eléctrica poseen una autonomía inferior a un mes. Depende de señales satelitales, por lo que no funcionan en lugares con mala recepción.

Bluetooth Low Energy

Bluetooth Low Energy o *BLE* es una tecnología de red personal inalámbrica *WPAN* (por sus siglas en inglés) enfocada en hacer uso de una cantidad considerablemente menor de energía que el protocolo *Bluetooth* clásico manteniendo un rango de comunicación similar. Si bien su creación no fue con el concepto de ser utilizada como una tecnología de localización, es posible considerar la presencia de una señal *bluetooth* como un indicador de que el emisor de dicha señal se encuentra dentro del rango de detección.

Este protocolo es soportado de manera nativa por la mayoría de los sistemas operativos actuales incluyendo *iOS*, *Android*, *Windows Phone*, *macOS*, *Linux*, *Windows 8 y 10*, entre otros. Cabe mencionar que este protocolo no es retro compatible con el protocolo *Bluetooth* clásico.

Siendo capaz de transmitir y recibir datos al igual que su versión tradicional, pero con consumo energético considerablemente menor, *BLE* se convirtió rápidamente en el protocolo preferido para los dispositivos personales que entregan información relevante al usuario, como sistemas de monitoreo cardiaco, sensores de actividad física, sensores de glucosa en la sangre, etc. El hecho de poder transmitir datos relevantes al usuario en un protocolo compatible con tantos sistemas operativos distintos y a un bajo costo lo convirtieron en uno de los protocolos más utilizados para estas tecnologías.

Figura 2.3: Logo de Bluetooth Smart, nombre comercial de esta tecnología

Operaciones GATT

El protocolo *GATT* [5] define una estructura de datos jerárquica que es expuesta a los dispositivos *BLE* a los que se está conectado. Provee una serie de comandos que el cliente puede utilizar para descubrir información acerca del servidor, como su identificador, sus

[5] GATT Overview
https://www.bluetooth.com/specifications/gatt/generic-attributes-overview

servicios disponibles, entre otros. Entre los comandos que se proveen se puede leer datos desde el servidor al cliente y escribir desde el cliente al servidor, lo que permite enviar y recibir datos entre ambas partes. El cliente también puede subscribirse a alguna característica en particular del servidor y éste se la enviará en cuanto se encuentre disponible.

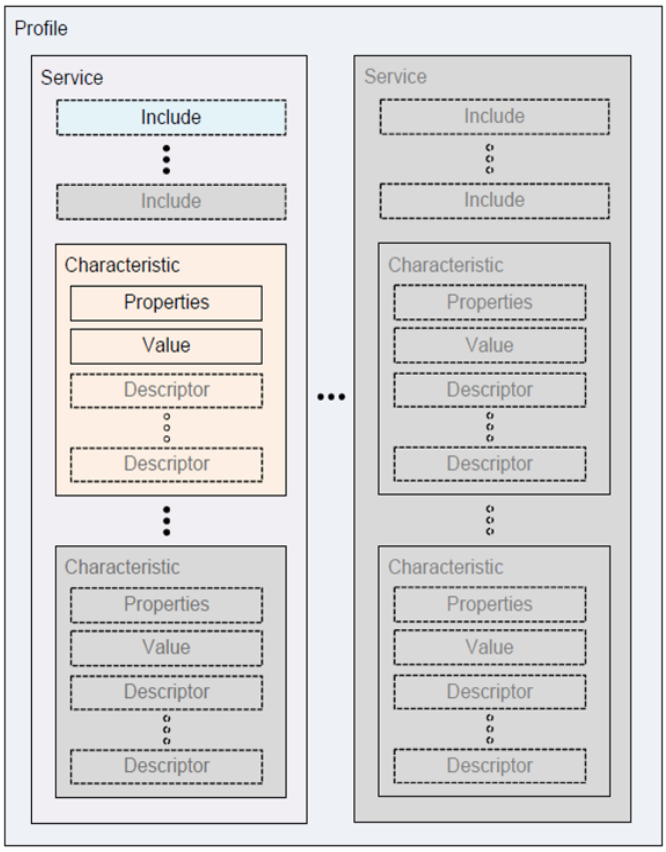


Figura 2.4: Esquema de la jerarquía GATT
Fuente: https://www.bluetooth.com/specifications/gatt/generic-attributes-overview

El nivel superior de la jerarquía es el perfil, el cual está compuesto por uno o más servicios. Un servicio está compuesto por características o referencias a otros servicios. Una característica en un tipo de dato (que se representa por un identificador único global *UUID*), un valor, un set de propiedades que definen las operaciones que la característica soporta y un set de permisos relacionados a la seguridad.

Batería

BLE está diseñado para operar con un bajo consumo energético, esto ha permitido que algunos fabricantes logren que sus dispositivos se alimenten de una batería de 1.000mAh por lapsos de tiempo superiores a los 12 meses. Esto se debe principalmente por la eficiencia que posee el protocolo al transmitir solamente paquetes de pequeño tamaño en comparación a su versión clásica. La versión tradicional de *Bluetooth* podría consumir la energía de una batería de 1.000mAh en cuestión de algunas horas.

Autonomía	Alta
Precisión	Media
Costo	Bajo
Ventajas	Ofrece una gran autonomía y además no depende de señales satelitales como el GPS.
Desventajas	Su precisión se basa únicamente en la intensidad de señal que se reciba.

RFID

La Identificación por Radio-Frecuencia (*RFID* por sus siglas en inglés) es una tecnología que usa campos electromagnéticos para identificar de forma automática etiquetas adheridas a objetos.

Figura 2.5: Símbolo representativo de RFID

Las etiquetas *RFID* se separan en dos grandes grupos. Por un lado están las denominadas etiquetas activas, las cuales poseen una fuente de energía propia y son capaces de transmitir sus datos al lector *RFID* por su propia cuenta. Por otro lado se encuentran las etiquetas pasivas, que corresponden a aquellas etiquetas que no poseen una fuente de energía propia y por tanto dependen de que el campo electromagnético del lector alimente al micro circuito de la etiqueta, permitiendo así enviar sus datos. Estas últimas son las más comunes del mercado, pudiéndose encontrar en las etiquetas de productos de algunas tiendas comerciales, ayudando a prevenir robos, como también es posible encontrarlas en las tarjetas de transporte público, etc.

La mayor diferencia entre un tipo de etiqueta y otra, además de que si poseen fuente de energía propia o no, es el rango de detección que cada una ofrece. Las etiquetas activas poseen un rango de detección mayor que las pasivas debido a que son capaces de enviar su identificación por cuenta propia.

Para el caso de las etiquetas RFID pasivas, la tabla resumen es la siguiente:

Autonomía	Indefinida
Precisión	Alta
Costo	Bajo
Ventajas	Debido a que es el lector el que proporciona la corriente eléctrica necesaria para operar, su autonomía es indefinida. Su costo es muy bajo.
Desventajas	Es el lector el que le proporciona la alimentación eléctrica, por lo que su correcto funcionamiento depende exclusivamente de esta parte.

Para el caso de las etiquetas RFID activas, la tabla resumen es la siguiente:

Autonomía	Alta
Precisión	Media
Costo	Alto
Ventajas	Poseen un aceptable rango de detección y autonomía.
Desventajas	Posee un elevado precio y no todos los modelos permiten cambiarle la batería.

2.2. Tecnologías de Transmisión de Datos

Tras estimar la localización de un activo fijo mediante el uso de alguna de las tecnologías mencionadas en el punto 2.1, el siguiente paso es transmitir estos datos a un servidor central, el cual se encargará de procesarlos.

Para los efectos de esta memoria la tecnología a utilizar será la basada en el protocolo *IPv4*, ya sea mediante el uso de una conexión *ethernet* o *Wi-Fi*. Sin embargo, se entiende que estas conexiones no están siempre disponibles en los lugares donde se desea monitorear los activos, es por esto que a continuación se presentarán algunas tecnologías que podrían, en un eventual caso, utilizarse en reemplazo a las recién mencionadas.

6LoWPAN

6LoWPAN es un acrónimo para Redes Personales de Bajo Poder En *IPv6* en sus siglas en inglés [6].

Este concepto se originó de la idea que "el protocolo de internet debería poder aplicarse incluso a los dispositivos más pequeños" y que los aparatos de bajo poder y capacidad limitada de procesamiento también debiesen poder participar del internet de las cosas. [7]

Para lograr esto, el grupo encargado de desarrollar *6LoWPAN* definió el encapsulamiento y los mecanismos de compresión de los paquetes de protocolo *IPv6* para que sean enviados y recibidos en redes basadas en *IEEE 802.15.4* [8]

[6] 6LoWPAN - `https://http://www.ti.com/wireless-connectivity/6lowpan/overview.html`

[7] What is IoT - `https://www.theguardian.com/technology/2015/may/06/what-is-the-internet-of-things-google`

[8] IEEE 802.15 WPAN™ Task Group 4 - `http://www.ieee802.org/15/pub/TG4.html`

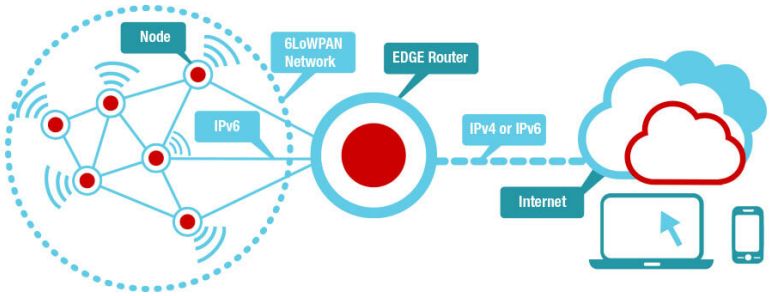


Figura 2.6: Esquema de 6LoWPAN

Los nodos se conectan entre sí creando una red que se auto-repara y que es gestionada por ellos mismos. Generalmente utilizan muy poca memoria y poder de procesamiento lo que permite que se incorporen en microcontroladores de baja potencia.

El router *EDGE* existe para transformar los paquetes comprimidos y optimizados, que utiliza la red *6LoWPAN* a paquetes tradicionales de *IPv4* o *IPv6* utilizables de la forma habitual.

Finalmente un servidor externo u otro punto de conexión puede hacer la gestión y administración de los datos entregados por el router *EDGE*.

Ventajas

Dentro de los beneficios de esta tecnología destacan el uso de protocolos internos conocidos, nodos conectados de punto a punto que no necesitan un punto de acceso para operar (los nodos se comunican entre sí sin la necesidad un intermediario), escalabilidad, auto-reparación y alto grado de conectividad incluso si algunos nodos no se encuentran disponibles momentáneamente.

Además, debido a la modificación y reducción de consumo del protocolo *IPv6*, la tecnología de *6LoWPAN* es bastante útil para ser integrada en pequeños aparatos como detectores de humo, sensores de diversos tipos, detectores de movimiento, etc. Básicamente permite que cualquier dispositivo electrónico, por pequeño y simple que sea, tenga la oportunidad de ser integrado al Internet de las Cosas para enviar sus datos a otra unidad y así tomar las medidas correspondientes. En el caso de los detectores de humo, podrían interconectarse todos los detectores de humo de un edificio y a la vez conectar esta red a un equipo de monitoreo. Cuando alguno de los detectores indique la existencia de humo en sus cercanías, podrá transmitir su identificación y la alarma al router *EDGE*, haciendo uso de la red generada por los otros detectores y de esta forma avisarle al equipo monitor que se deben tomar acciones.

Desventajas

Solamente el router se comunica con el exterior mediante los protocolos *IPv4* y *IPv6*, por lo que no es sencillo conocer la comunicación interna que existe dentro de la red. Además, si por algún motivo, la red no logra conectarse al nodo router, entonces no se realizará ningún tipo de comunicación con el exterior.

ZigBee

ZigBee es una especificación basada en *IEEE 802.15.4* que hace uso de protocolos de comunicación de alto nivel para crear redes con bajo consumo energético y bajas necesidades de ancho de banda[9]. Fue diseñada para proyectos de baja escala que necesitaban conexión inalámbrica y por tanto la distancia de transmisión es reducida. La tecnología fue definida con la intención de ser más simple y menos costosa que otras alternativas como *Bluetooth* o *Wi-Fi*. Su bajo consumo limita la distancia de transmisión desde unos cuantos metros hasta unos 100 metros, aproximadamente, dependiendo del ambiente y los obstáculos. Para solventar este problema, *ZigBee* se interconecta entre nodos retransmitiendo los datos entre sí hasta llegar a un nodo del tipo router según se muestra en el siguiente esquema.

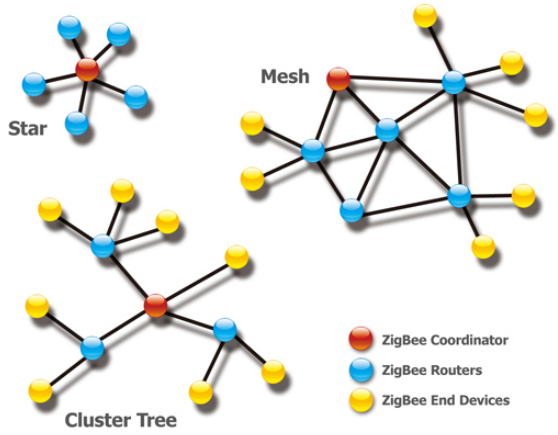


Figura 2.7: Esquema de ZigBee

Independiente del modelo, se debe tener un nodo coordinador encargado de la creación de la red, el control de sus parámetros y su mantención básica. Cuando se usa el esquema estrella el nodo coordinador debe ser el nodo central.

Posteriormente se creó la especificación *ZigBee Smart Energy V2.0* que hace uso del sistema de compresión de *6LoWPAN* mencionado en la sección 2.1.

Ventajas

Es utilizado en una gran cantidad de proyectos y comienza a tomar cada vez más relevancia en proyectos del tipo *IoT*. Una gran cantidad de empresas prefieren hacer uso de esta tecnología sobre una tecnología similar, como es el caso de *6LoWPAN*, como se puede verificar en su sitio web.

Desventajas

El protocolo de *ZigBee* se pensó para aplicaciones que necesitan un bajo consumo de energía y bajo flujo de datos entre nodos, es por esto que no se desempeña bien cuando los nodos se mueven constantemente. Además, *ZigBee* operara en conjunto con otros dispositivos que usen el mismo protocolo, mientras que *6LoWPAN* se comunica al exterior mediante el uso de protocolos *IP*.

[9]Zigbee Alliance - `http://www.zigbee.org/`

WiMAX

Interoperabilidad mundial para acceso por microondas *(WiMAX por sus siglas en inglés)* es una familia de estándares de comunicación inalámbrica que se basa en el estándar *IEEE 802.16*. Se enfoca en redes inalámbricas de largo rango, tanto para conexiones móviles como para conexiones fijas. Cuando se ideó pretendía ser la tecnología líder para proveer internet como alternativa al cable y a las conexiones *DSL*[10]. A pesar de ser más económico que implementar cableado como con *DSL* la industria mundial de telecomunicaciones prefirió invertir en otras alternativas como el *LTE*[11].

Figura 2.8: Logo de WiMAX Forum, Principal organización que implementa esta tecnología

Su funcionamiento se basa en dos partes. Primeramente en estaciones de transmisión instaladas por los proveedores para cubrir cierta área, y los receptores, instalados por los clientes.

Ventajas

Dentro de sus ventajas destacan el bajo costo de implementación, considerando la escala, y su comportamiento flexible. Puede ser instalado de forma más rápida que otras tecnologías debido a que sus torres de transmisión son más pequeñas. También hay que considerar que no sólo se diseñó para conexiones fijas, también es posible usar *WiMAX* en dispositivos móviles y computadores haciendo uso de antenas *USB*. Su velocidad de transmisión puede alcanzar los 40 Mbps para los equipos móviles y hasta un 1 Gbps para las estaciones fijas, cubriendo un área de varios kilómetros.

Desventajas

Debido a que su tecnología se basa señales inalámbricas, mientras más lejos se esté de la fuente, más lenta e inestable se volverá la conexión. Esto significa también que mientras más usuarios estén utilizando el área de cobertura de una estación de *WiMAX*, más afectada se verá la calidad del servicio.

[10]Digital Subscriber Line - `https://www.lifewire.com/digital-subscriber-line-817527`
[11]LTE - `http://www.3gpp.org/technologies/keywords-acronyms/98-lte`

Symphony Link

La empresa Estadounidense *Link Labs* [12] ha desarrollado una tecnología llamada *Symphony Link* que se presenta como una solución inalámbrica para empresas que tienen la necesidad de conectar de forma segura sus dispositivos *IoT* a la nube.

Bajo la premisa de que el rango que proporcionan las redes *Wi-Fi* y *ZigBee* son limitantes y que las redes celulares son muy costosas, y con un alto consumo energético, se diseñó *Symphony Link* para ser una red de bajo consumo energético y con una amplia cobertura. A su vez se denomina escalable y confiable.

Figura 2.9: Logo de LinkLabs, Empresa desarrolladora de Symphony Link

Su funcionamiento se basa en el uso de pequeños módulos de bajo consumo que se instalan en los dispositivos a monitorear que se encargan de realizar toda la comunicación con un *Gateway* conectado a internet. Este comportamiento es similar al caso de las redes ZigBee en donde se tiene un nodo coordinador, solo que en este caso se trata de un equipo especial y no un nodo regular.

Ventajas

Su principal ventaja es su variedad de módulos disponibles, dependiendo de la necesidad del usuario, además de la existencia de una extensa *API* compatible con las tecnologías actuales.

Según su página web, en una red estándar, es posible tener una cobertura de más de dos kilómetros, manteniendo una conexión estable y la integridad de sus datos.

Desventajas

Al ser creado por una empresa privada, el desarrollo de esta tecnología va dictado fuertemente por las decisiones y avances de la empresa, sin dejar mucho espacio al desarrollo comunitario.

[12] LinkLabs - `https://www.link-labs.com/`

2.3. Frameworks

Node.js

Node.js[13] es un *framework* de código abierto y multiplataforma para la ejecución de código JavaScript en el lado del servidor. Construido con el motor JavaScript V8 de Chrome, *Node.js* ha sido uno de los grandes exponentes del paradigma "JavaScript en todos lados", lo que permite compartir código del lado del servidor con el lado del cliente.

Opera en un sólo *thread*, usando un modelo de operaciones de Entradas y Salidas sin bloqueo, lo que permite miles de conexiones concurrentes sin caer en el costo de manejar diferentes *threads*.

Figura 2.10: Logo Node.js, Fuente: www.nodejs.org

LoopBack

LoopBack[14] es un *framework* de *Node.js* de código abierto que permite crear *API REST* de manera rápida haciendo uso de scripts para la creación de métodos y definiciones. Su flexibilidad permite una gran variedad de configuraciones para adaptarse tanto a soluciones existentes como a proyectos nuevos.

Posee kits de desarrollo de software para Android, iOS y Angular lo que facilita la creación de aplicaciones que consuman la *API*.

Figura 2.11: Logo LoopBack, Fuente: https://loopback.io

[13]Node.js - `www.nodejs.org`
[14]LoopBack - `https://loopback.io`

Angular

Angular[15] es un *framework* de código abierto orientado a aplicaciones web desarrollado en *TypeScript*[16]. Actualmente es mantenido por *Google* y tiene como objetivo el crear sitios de una sola página manteniendo el paradigma Modelo Vista Controlador.

Es la evolución a su predecesor *AngularJs*.

Figura 2.12: Logo Angular, Fuente: https://angular.io

2.4. Módulos

Noble

Noble es un módulo multiplataforma para *Node.js* para trabajar con *Bluetooth BLE*.

Depende de paquetes de desarrollo bluetooth para su correcto funcionamiento.

Proporciona métodos para interactuar con dispositivos *Bluetooth BLE* y maneja los eventos del protocolo para interactuar los los servicios de operaciones *GATT*.

Figura 2.13: Logo Noble, Fuente: https://github.com/sandeepmistry/noble

[15] Angular - `https://angular.io`
[16] TypeScript - `https://www.typescriptlang.org/`

2.5. Computadores de Placa Reducida

Raspberry Pi 3

Es un computador de placa reducida, de bajo costo y cuyo sistema operativo oficial es una variante del sistema *Debian*[17] llamado *Raspbian*[18].

Este modelo tiene un consumo energético de 800 [mA] y posee conectividad *bluetooth* 4.1 y *Wi-Fi* incorporada, además de un puerto *Ethernet*.

El hecho que su sistema operativo esté basado en *Debian* permite que la gran mayoría de sus paquetes sean compatibles siempre que se compilen para la arquitectura de computadores *ARM*.

Figura 2.14: Raspberry Pi 3 Modelo B, Fuente: www.raspberrypi.org

[17] Debian - `https://www.debian.org`
[18] Raspbian Stretch Lite - `https://www.raspberrypi.org/downloads/raspbian/`

Capítulo 3

Diseño y Construcción de la Solución

3.1. Tecnologías a Utilizar

Localización

Descartando el sistema *GPS* por los motivos antes mencionados, las dos opciones para seleccionar son *Bluetooth BLE* y *RFID* (en su versión activa o pasiva), pero primero es necesario aclarar cómo estas tecnologías, que no fueron diseñadas como tecnologías de localización, pueden entregar ese tipo de información.

Si bien ninguna de estas tecnologías es capaz de determinar, por sí solas, su ubicación, ambas emiten una señal con su identificación. Para poder obtener una ubicación aproximada del emisor de dichas señales se debe instalar un receptor en una ubicación conocida, y de esta forma se puede deducir que el activo se encuentra a una distancia no mayor a la máxima distancia de detección del receptor.

Esta técnica tiene dos principales problemas:

1. El área de detección, en la práctica, no es circular ya que se ve afectada por los obstáculos que se encuentren en el camino.

2. La no detección de una señal podría generar la especulación de que el activo no se encuentra en un rango cercano al receptor siendo que en realidad éste se encuentra detrás de un obstáculo.

Es en base a estos problemas que el margen de error de la ubicación estimada no es despreciable y no se recomienda usar este tipo de tecnología si se desea obtener una alta precisión de ubicación (unos pocos metros de error).

Al ser ésta la técnica a utilizar, las etiquetas *RFID* pasivas quedan descartadas, puesto que su rango de detección es muy inferior en comparación a su contraparte activa y a la tecnología *bluetooth* si se desea mantener un bajo costo de implementación.

Finalmente se optó por hacer uso de la tecnología *bluetooth* debido a que posee un menor costo de implementación, existe una gran variedad de dispositivos y *frameworks* compatibles y posee una mejor relación costo/autonomía en términos de batería.

Transmisión de Datos

Para efectos de esta memoria la transmisión de datos se realizó de manera directa, obteniendo los datos de los *beacons* por la interfaz *bluetooth* y envíandolos al servidor central mediante la interfaz *Wi-Fi*. No se implementó ninguna de las otras tecnologías mencionadas,

ya que escapa del alcance de este documento, sin embargo es importante comentar bajo qué situaciones es recomendable hacer uso de alguna de estas propuestas.

Debido a que *WiMAX* implica la instalación de grandes antenas para su operación, y el elevado costo que esto significa, es que son muy pocos los casos en los que ésta sería la mejor opción a implementar.

La tecnologías mencionadas se vuelven interesantes cuando el acceso a internet, ya sea por *Wi-Fi*, *Ethernet* o redes celulares, no es una opción. Esto es común, por ejemplo, en ubicaciones subterráneas o muy remotas. Para este tipo de situaciones una alternativa es crear una red interna que permita la retransmisión de datos hasta llegar a un punto que disponga conexión a internet.

Symphony Link promete ser útil en estos casos, pero al pertenecer a una empresa privada podría existir un alza en los costos de implementación.

Una red montada bajo la tecnología *ZigBee* o *6LoWPAN* parecen ser una mejor alternativa debido a la gran compatiblidad de implementación con diversos *frameworks* y a la capacidad de generar redes de forma automática y adaptarse a los cambios de nodos que se puedan realizar.

Recepción y transmisión de datos

Ya que se hará uso de *bluetooth* se optó por utilizar el computador de placa única *Raspberry Pi 3* Modelo B, que ya incorpora una interfaz *bluetooth* además de una interfaz *Wi-Fi* y *Ethernet* para la transmisión de datos. Gracias a que su sistema operativo es una variación de *Debian* es posible hacer utilizar el *framework Node.js* para desarrollar el software que se ejecutará.

Existen más alternativas, como hacer uso de tecnologías como *Arduino*[1], pero esto significa un mayor esfuerzo en implementación y desarrollo de código debido a sus limitadas capacidades.

Implementación de la *API* Central

Para la implementación de la *API* Central se utilizó el *framework LoopBack* debido a que es un *framework* liviano y de ágil desarrollo.

Se pudo haber implementado haciendo uso de otros *frameworks* como *Flask*[2] o *Ruby on Rails*[3], pero se optó por *LoopBack* por que existía experiencia previa con su uso.

3.2. Conocimiento del Dominio

Antes de comenzar a explicar la solución implementada, es necesario dar una breve introducción sobre los términos específicos del problema que se utilizarán.

Clientes: Empresas o personas naturales interesadas en obtener el servicio de monitoreo. Su uso es más administrativo.

Empleados: Usuarios asociados de un cliente.

Beacons: Transmisores bluetooth que serán detectados por los emisores.

Senders: Dispositivo que detecta los beacons cercanos y envía estos datos al servidor. Para esta solución corresponden ser los dispositivos *Raspberry Pi*

Assets: Beacons o Senders.

[1] Arduino - `https://www.arduino.cc/`
[2] Flask - `http://flask.pocoo.org/`
[3] Ruby on Rails - `http://rubyonrails.org/`

3.3. Definición de Requerimientos

Para determinar si la solución estudiada es válida, se presentarán una serie de requerimientos que ésta debe cumplir.

Detección de Dispositivos
Los *Senders* deben ser capaz de detectar *Beacons* que se encuentren en un radio menor a 30 metros.

Diferenciación de Dispositivos
Los *Senders* deben ser capaz de diferenciar un *Beacon* de otro.

Almacenamiento de Datos
Los *Senders* deben ser capaz de almacenar los datos obtenidos respecto a los *Beacons*, tanto su detección, su ausencia y su variación de señal.

Envío de Datos a Servidor Central
Los *Senders* deben ser capaces de enviar los datos recolectado a un servidor central para su posterior análisis.

Visualización de Información
Debe existir una interfaz web que muestre la información de cada *Sender* y cada *Beacon* identificando si se encuentra en un rango visible o no. Se debe mostrar un registro de los últimos movimientos.

Escalabilidad
Se debe poder agregar tanto *Senders* como *Beacons* al proyecto y éstos deben ser incorporados sin hacer una modificación al código.

Intercomunicación
Los datos recolectados por cada *sender* respecto a un *beacon* deben ser únicos y su análisis en conjunto debe poder generar más información que el análisis de datos por separado.

Autonomía
La solución debe ser capaz de operar de forma autónoma sin la necesidad de intervenir los *Sender* ni los *Beacons* en un periodo de al menos 6 meses.

Rentabilidad
El costo de implementación de la solución no debe superar al 10 % del costo de los activos a controlar.

3.4. Implementación de la Solución

Debido a que se optó por implementar la solución que hace uso de la tecnología *Bluetooth Low Energy* se preparó un prototipo funcional haciendo uso de *beacons* marca Ghostyu[4], modelo *Ibc14*, debido a su bajo costo en el mercado y su gran autonomía y rango de emisión (1 año con una batería *CR2450* y 50 metros según el fabricante).

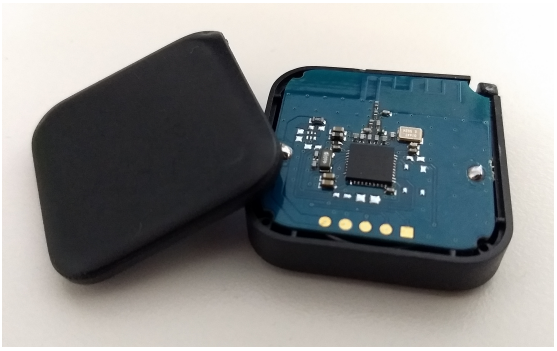


Figura 3.1: Beacon Ghostyu, Fuente: Elaboración Propia

Para el caso de los *senders* se optó por utilizar la computadora de placa reducida *Raspberry Pi 3* Modelo B.

[4]Ghostyu - `http://ghostyu.com`

3.5. Esquema de la solución

Para implementar la solución, se dividó el problema en tres partes. En primer lugar se implementó el software que utilizarán los *senders* para detectar los *beacons* cercanos, en donde estos datos se almacenan de forma local para luego ser enviados a una *API REST* central. La *API* está encargada de recibir todos los datos de todos los *senders* y almacenar esos datos en su propia base de datos distinguiendo a cada *sender* por separado y llevando un registro de sus movimientos. Finalmente se creó un *Dashboard Web* el cual consulta a la *API REST* por los datos respectos a los *senders* y los *beacons*.

A continuación se presenta un esquema que representa la solución implementada.

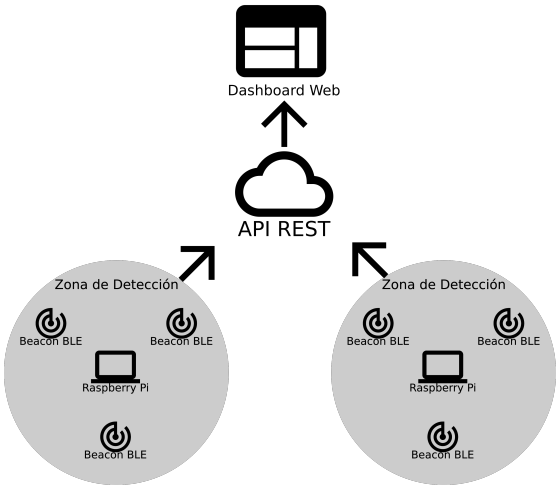


Figura 3.2: Esquema de la solución implementada, Fuente: Elaboración Propia

Software utilizado en *Senders*

Para el caso de los *senders* se optó por hacer uso del framework *Node.js*.

Para la detección de los *beacons* se hizo uso del módulo de *Node.js noble* el cual permite poner a la escucha al *sender* y detectar la dirección física *MAC* de los *beacons* y detectar los cambios de sus *RSSI*, que vendría a ser el equivalente de la intensidad de señal detectada. El módulo en sí ofrece muchos más servicios y operaciones, las cuales no fueron utilizadas en la solución debido a que sólo era necesario determinar la presencia de los *beacons*.

El código está dividido en tres principales componentes, una parte maneja la lógica con la base de datos local, que denominaremos *DB Manager*, que almacena los datos recolectados, una segunda parte administra la interacción con la *API* central, que denominaremos *API Manager*, y finalmente la tercera parte es la encargada de detectar los *beacons* y coordinar las otras dos partes. Esta última es el punto de entrada del programa y es en donde se llaman a las otras dos partes cuando es requerido.

Debido a que la conexión con la *API* central no se puede dar siempre como garantizada (ya sea por problemas de conexión, fallas con la misma, etc) es necesario contar con una base de datos local en el *sender* que permita registrar todos los datos necesarios y que no sea susceptible a este tipo de fallas. Esta base de datos se implementó en *SQLite* [5] debido a que genera base de datos basadas en archivos y resulta muy útil cuando los volúmenes de datos son reducidos.

[5] SQLite - `https://www.sqlite.org`

Se determinó que los datos mínimos que se debían almacenar en esta base de datos, para obtener buenos resultados, se podía almacenar en tres tablas.

- *beacons*: Esta tabla almacena los datos de los *beacons* y su estado actual (visible o no). Sus columnas son:
  - id: Identificador autogenerado por la base de datos para filtrar resultados.
  - address: Identificador único del *beacon*.
  - visible: Indica si el *beacon* se encuentra visible actualmente.
  - rssi: Indica la intensidad de señal actual del *beacon*.
  - no_signal_count: Contador de uso interno para determinar su un *beacon* salió del radio de detección.
  - last_visible_update: Fecha en la que se supo por última vez algo respecto al *beacon*, ya sea una entrada o una salida.
- *logs*: Esta tabla almacena los registros y cambios detectados por el *sender*. Sus columnas son:
  - id: Identificador autogenerado por la base de datos para filtrar resultados.
  - beacon_id: Identificador interno de la base de datos respecto al *beacon* que está generando el registro.
  - rssi: La intensidad de señal del *beacon* que generó el registro en el momento en el que éste se generó.
  - event: Valor que determina el tipo de evento registrado, siendo -1 si se detecta que el *beacon* salió del rango de detección, 0 si se detecta que el *rssi* del *beacon* varió una cantidad determinada respecto al valor que se tenía anteriormente y 1 si se detecta que el *beacon* entró al área de detección.
  - date: Fecha en la cual se ingresó el registro.
- api_synchronize: Esta tabla almacena datos sobre la última sincronización con la *API* central. Sus columnas son:
  - id: Identificador autogenerado por la base de datos para filtrar resultados.
  - update_date: Fecha en la que se realizó una sincronización con la *API* central.
  - last_log_date: Indica la fecha del último *log* que se sincronizó con la *API* central.

El programa comienza llamando al *DB Manager* indicándole que debe preparar la base de datos para uso, una vez se determina que la base de datos local está funcionando de manera correcta comienza la detección constante de *beacons*. Debido a la forma en la funciona *Node.js*, cada vez que el *sender* detecta un *beacon* el programa llama a un "evento' que se encarga de realizar alguna acción correspondiente. Cuando se detecta un *beacon* lo primero que se hace es verificar si el *beacon* detectado corresponde ser uno de los *beacons* que estamos monitoreando, esta verificación se debe realizar debido a que el *sender* no discrimina a priori qué tipo de dispositivo detectar y se mostrará cualquier aparato que emita señal mediante el protocolo *Bluetooth Low Energy*. Si el *beacon* detectado es de nuestro interés pueden suceder dos eventos; si la intensidad de señal con la que se detectó el *beacon* varió una cantidad mayor a un parámetro predefinido, entonces se debe registrar este cambio en la base de datos llamando al *DB Manager*, el segundo caso ocurre cuando el *beacon* detectado se encontraba como no visible en la base

de datos, de ser así, se debe indicar este evento también en la base de datos mediante el *DB Manager* y posteriormente se debe indicar que está visible actualmente.

Hasta ahora el programa detecta y registra correctamente cuando un *beacon* entra en un radio de detección o cuando su intesidad de señal varía sobre una cantidad determinada, sin embargo no se está determinando cuándo un *beacon* sale del área de detección. El problema de esta situación es que la única forma de determinar este caso es verificando constantemente si se detecta o no el *beacon* en cuestión y hasta ahora sólo recibimos datos cuando se detecta. Para solucionar este problema el programa periódicamente suma en 1 un contador en la base de datos que determina cuántos segundos lleva ausente un *beacon* y este contador se resetea cada vez que se detecta el *beacon* en cuestión. De esta forma, y mediante parámetros del programa, se determina cuántos segundos debe estar "desaparecido" el *beacon* para determinar que salió del rango de detección.

Para la sincronización con la *API* central, cada *N* segundos (fijado por parámetros del programa) se envían los datos nuevos a la *API* y se actualiza en la base de datos local cuáles fueron los datos enviados para evitar que se envíen más de una vez. En caso de no existir datos nuevos se envía un mensaje en blanco indicando que no hay mensajes nuevos y así la *API* central sabe que no existen datos nuevos y no se ha dado el caso en el que el *sender* ha perdido la conexión a Internet.

API REST

La funcionalidad de la *API* es la de recibir los datos enviados por los *senders* y almacenarlos en una base de datos accesible desde Internet. La *API* es capaz de reconocer el *sender* que envía los datos mediante un identificador y lo compara con sus registros locales, de esta forma se evita que se reciban datos desde una fuente desconocida. En el modelo completo de la base de datos se puede asociar *senders* a clientes y se pueden crear empleados asociados a clientes, ésto con la función de que cada cliente pueda otorgar permisos a sus empleados para monitorear y vigilar los registros y movimientos de los activos, sin embargo ese caso de uso escapa del contexto de este prototipo funcional y por lo tanto no se implementó de manera lógica, pero sí se definió en el modelo de la base de datos.

La base de datos en la cual la *API* guarda sus registros fue creada haciendo uso del Sistema de Administración de Base de Datos *MySQL* [6], debido a que corresponde a un sistema de base de datos relacional, es gratuito y se tenía experiencia previa haciendo uso de este sistema.

La comunicación entre el *sender* y la *API* central se basa principalmente en un sólo *endpoint*, al cual el *sender* debe realizar todos los envíos de datos.

El *endpoint* especificado espera tres parámetros formados en *JSON*.

- Identifier: Un *string* identificado del *sender* que envía los datos.
- Logs: Los registros detectados por el *sender* y que no han sido informados aún a la *API* en forma de un *arreglo*.
- Visibility: Un *arreglo* que contiene el estado de visibilidad de cada *beacon* según el *sender*.

En base a este llamado periódico la *API* tiene la función de actualizar los datos que tiene en su base de datos con los datos que recién han llegado. Cada vez que se reciben datos éstos quedan registrada con una fecha y hora correspondiente a la fecha y hora actual del sistema y así queda un registro del momento en el que se realizó la sincronización de datos.

Finalmente *LoopBack*, al estar bien definido, entrega los *endpoints* que consumirá el *Dashboard Web* para mostrar la información relevante para el potencial cliente.

[6] MySQL - `https://www.mysql.com`

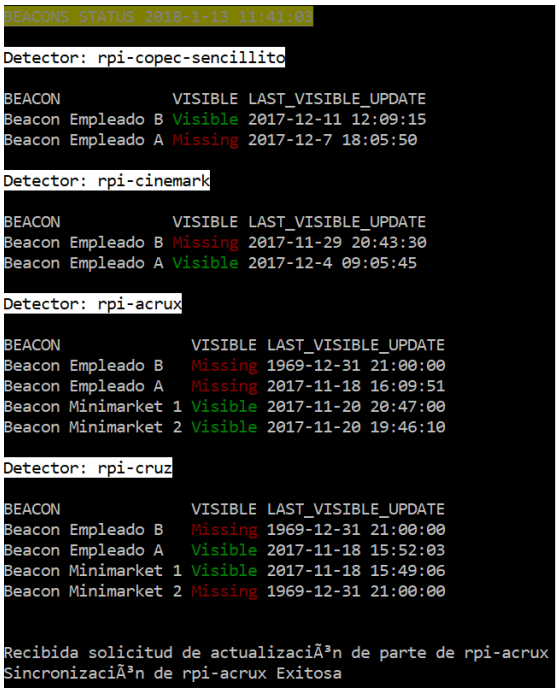


Figura 3.3: Información mostrada por consola en la API *Central, Fuente: Elaboración Propia*

Dashboard Web

El *dashboard* es la interfaz que finalmente verá el cliente. En él se mostrará la información actual de los activos que están asociados a su cuenta y los valores más importantes. Para obtener la información necesaria debe conectarse con la *API* central y recolectar desde allí los datos.

Fue implementado haciendo uso del framework *Angular* ya que permite la fácil integración con clientes *REST* para comunicarse con la *API*.

Para determinar que el prototipo cumple los objetivos se determinó que el *dashboard* debía ser capaz de mostrar la siguiente información:

- Estado actual de cada *beacon* según cada *sender*.
- Registros de los *beacons* filtrados por *sender* o *beacon*

Para lograr que la información se fuese actualizando periódicamente y así dar la ilusión de que se obtenía la información en tiempo real, se fijó un temporizador que cada 5 segundos llamaba a una función encargada de actualizar los datos.

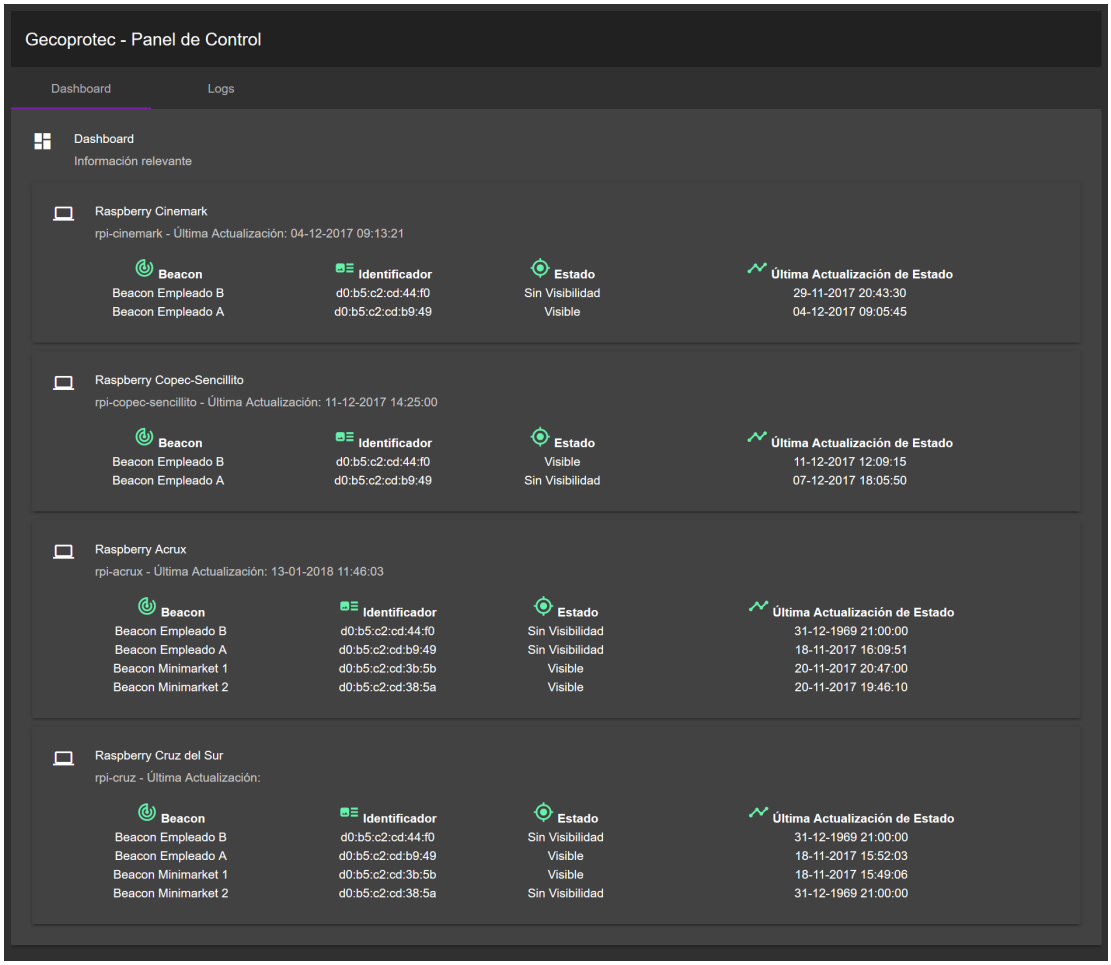
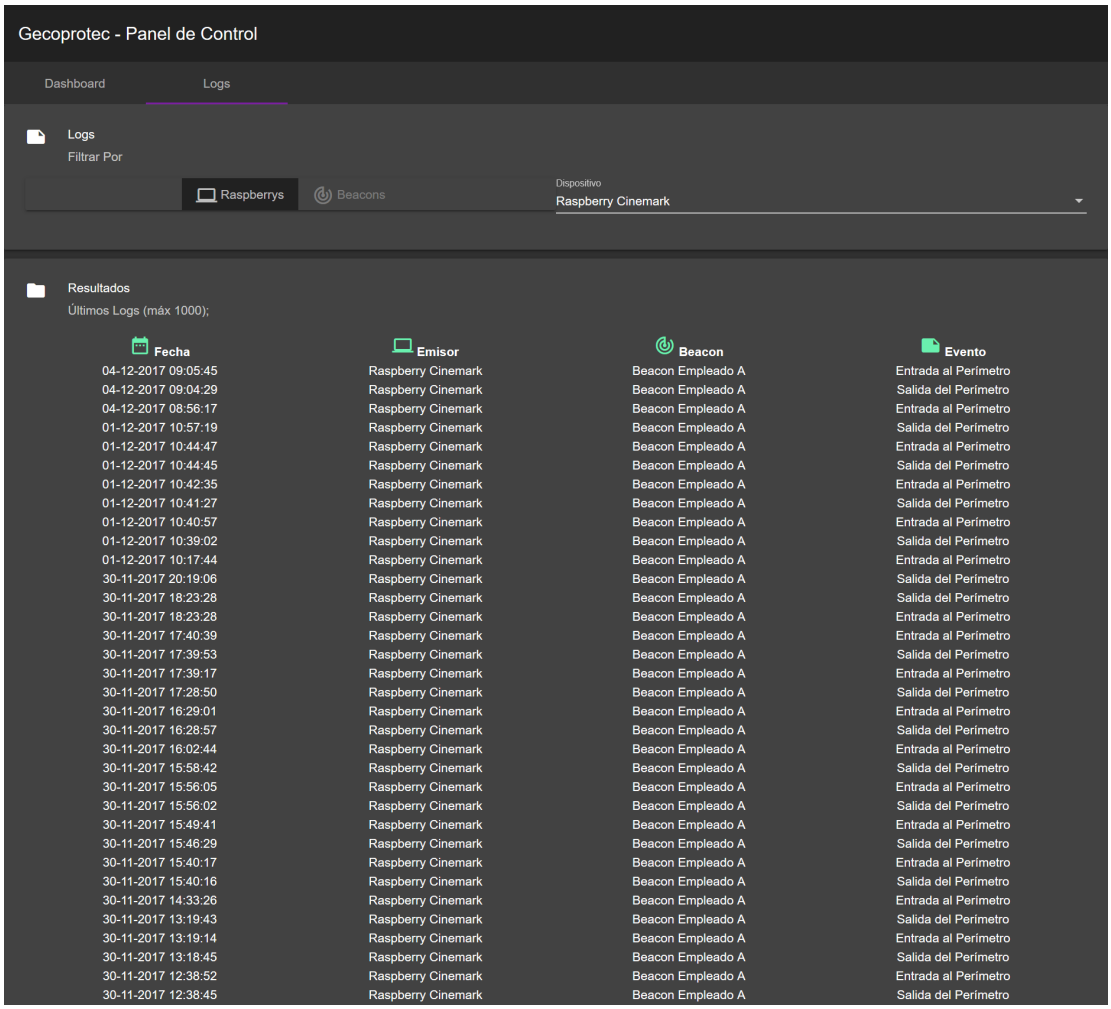


Figura 3.4: Screenshot de Dashboard Web con datos Reales, Fuente: Elaboración Propia



Fecha	Emisor	Beacon	Evento
04-12-2017 09:05:45	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
04-12-2017 09:04:29	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
04-12-2017 08:56:17	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
01-12-2017 10:57:19	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
01-12-2017 10:44:47	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
01-12-2017 10:44:45	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
01-12-2017 10:42:35	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
01-12-2017 10:41:27	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
01-12-2017 10:40:57	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
01-12-2017 10:39:02	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
01-12-2017 10:17:44	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 20:19:06	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 18:23:28	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 18:23:28	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 17:40:39	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 17:39:53	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 17:39:17	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 17:28:50	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 16:29:01	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 16:28:57	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 16:02:44	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 15:58:42	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 15:56:05	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 15:56:02	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 15:49:41	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 15:46:29	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 15:40:17	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 15:40:16	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 14:33:26	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 13:19:43	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 13:19:14	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 13:18:45	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro
30-11-2017 12:38:52	Raspberry Cinemark	Beacon Empleado A	Entrada al Perímetro
30-11-2017 12:38:45	Raspberry Cinemark	Beacon Empleado A	Salida del Perímetro

Figura 3.5: Screenshot de Dashboard Web con datos Reales, Fuente: Elaboración Propia

Capítulo 4

Validación y Análisis de Resultados

4.1. Validación de Resultados

En esta sección se verificará que la solución propuesta cumple con los requerimientos definidos en el Capítulo 3 y en qué grado se cumplen. Además se comentará cómo se podrían mejorar los resultados y cuáles son sus limitantes.

Instalar *beacons* dentro de activos fijos móviles resulta una tarea tediosa en la etapa de desarrollo, pues de realizarse alguna modificación o verificación es necesario desmontar el activo, para extraer el *beacon*, y posteriormente volver a instalarlo. En vista a este problema se optó por una solución alternativa en la que se entregó a dos empleados la empresa *Modyo* [1] un *beacon* a cada uno para que lo portaran constantemente consigo, mientras que en sus respectivas oficinas se instaló un *sender* encargado de monitorear toda actividad de cualquiera de ambos *beacons*. Esta alternativa es apropiada, como prototipo de prueba, puesto que cumple el mismo objetivo y, además, un trabajador presenta mayor movimiento dentro de un sector que un activo fijo móvil promedio, por lo que generará más datos al sistema para su posterior análisis.

El tiempo de recolección de datos fue de 45 días sin interrupciones a excepción de pequeños lapsus en los que se actualizaba el software de los *senders* o de la *API Central*.

La distancia entre ambos *senders* es la suficiente como para que uno no sea capaz de detectar un *beacon* ubicado en la oficina del otro, sin embargo, es posible que un *beacon* sea detectado por ambos *senders* si el empleado se encuentra en un área común ubicada entre ambas oficinas. Este comportamiento es deseado puesto que permite interpolar los datos de más de un *sender* para obtener más información respecto a un *beacon*.

Detección de Dispositivos

Para verificar este requerimiento se activaron 10 *beacons* dentro de un área que se extendía hasta los 30 metros de distancia de un *sender* y se observó si éstos eran detectados. De todas las pruebas que se realizaron, todos los *beacons* fueron detectados junto a otros dispositivos que también hacen uso del protocolo *BLE*, lo que permite confirmar el cumplimiento de este requerimiento.

Diferenciación de Dispositivos

Ya que cada *beacon* emite como identificación su dirección *MAC* es posible identificar con certeza a cada *beacon* como único. Esto se debe al hecho de que a cada fabricante se le asigna un lote de direcciones únicas para que las utilice en sus dispositivos y así evitar que dos aparatos utilicen el mismo identificador.

[1] Modyo - `https://www.modyo.com`

```
Inicio detección de dispositivos
d0:b5:c2:cd:b9:49 -55
d0:b5:c2:cd:44:f0 -22
d0:b5:c2:cd:3b:5b -32
d0:b5:c2:cd:38:5a -21
```

Figura 4.1: Screenshot de Software de Detección, Fuente: Elaboración Propia

Almacenamiento de Datos

Gracias a que los *senders* hacen uso de *SQLite* son capaces de almacenar todos los datos recolectados de manera local y así seguir detectando y registrando los cambios de estado de los *beacons* incluso si no poseen conexión a Internet. Los datos almacenados permiten determinar si el movimiento registrado corresponde a la entrada de un *beacon* al área de detección de un *sender*, una salida de éste o si simplemente hubo una variación en su intensidad de señal.

Table: logs

	id	beacon_id	rssi	event	date
	Filter	Filter	Filter	Filter	Filter
71	71	2	-73	1	2017-11-15 19...
72	72	2	1	-1	2017-11-15 19...
73	73	2	-78	1	2017-11-15 19...
74	74	2	1	-1	2017-11-15 19...
75	75	2	-87	1	2017-11-15 19...
76	76	2	1	-1	2017-11-15 19...
77	77	2	-92	1	2017-11-15 19...
78	78	2	1	-1	2017-11-15 19...
79	79	2	-74	1	2017-11-15 19...
80	80	2	1	-1	2017-11-15 19...
81	81	2	-92	1	2017-11-15 19...
82	82	1	-89	1	2017-11-15 19...
83	83	2	1	-1	2017-11-15 19...
84	84	1	1	-1	2017-11-15 19...
85	85	2	-99	1	2017-11-15 19...

Table: api_synchronize

	id	update_date	last_log_date
	Filter	Filter	Filter
110	110	2017-11-16 12...	2017-11-16 12...
111	111	2017-11-16 12...	2017-11-16 12...
112	112	2017-11-16 12...	2017-11-16 12...
113	113	2017-11-16 12...	2017-11-16 12...
114	114	2017-11-16 12...	2017-11-16 12...
115	115	2017-11-16 12...	2017-11-16 12...
116	116	2017-11-16 12...	2017-11-16 12...
117	117	2017-11-16 12...	2017-11-16 12...
118	118	2017-11-16 12...	2017-11-16 12...
119	119	2017-11-16 12...	2017-11-16 12...
120	120	2017-11-16 12...	2017-11-16 12...
121	121	2017-11-16 12...	2017-11-16 12...
122	122	2017-11-16 12...	2017-11-16 12...
123	123	2017-11-16 12...	2017-11-16 12...
124	124	2017-11-16 12...	2017-11-16 12...
125	125	2017-11-16 12...	2017-11-16 12...
126	126	2017-11-16 12...	2017-11-16 12...
127	127	2017-11-16 12...	2017-11-16 12...

Figura 4.2: Screenshot de datos almacenados en base de datos SQLite, Fuente: Elaboración Propia

Envío de Datos a Servidor Central

Debido a que el software de los *senders* fue desarrollado en *Node.js*, el cual es un lenguaje basado en *javascript*, el envío de datos a la *API* Central resulta muy sencillo. Además, al hacer uso de una base de datos local, se puede guardar cuándo fue la última interacción con el servidor central y qué datos se enviaron, lo que evita enviar los mismos datos más de una vez y reduce el consumo de ancho de banda que requiere la operación.

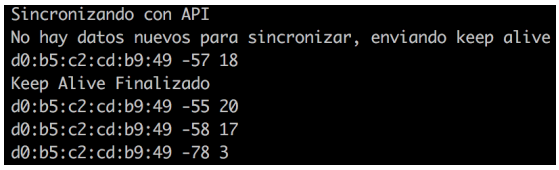


Figura 4.3: Screenshot de Sincronización de datos con API *Central, Fuente: Elaboración Propia*

Visualización de Información

La interfaz web se desarrolló en lo que es el *Dashboard Web* que se presentó en el capítulo anterior. En él se puede observar el estado de visibilidad de cada *beacon* respecto a cada *sender* y se pueden revisar todos los registros que existen en el servidor central que puedan ser de utilidad. Este *dashboard* puede mejorar en varios aspectos, sin embargo, fue diseñado de manera sencilla como prototipo funcional por lo que no se añadieron más funcionalidades.

Escalabilidad

Un factor importante, para determinar si esta solución es viable es conocer el número máximo de *beacons* que se pueden detectar simultáneamente. Ya se confirmó en la prueba de *Detección de Dispositivos* que con 10 dispositivos la solución funciona, pero para determinar cuál es la cantidad máxima de *beacons* que se pueden detectar es necesario conocer cómo funciona el protocolo de conexión de *Bluetooth*. El protocolo de conexión se divide en dos pasos [5], el primer paso corresponde al paso de *Inquiry*, en donde el dispositivo maestro (*beacon*) invita a los otros dispositivos (*senders*) a conectarse. El segundo paso, llamado *Paging*, en donde se comienza la conexión. Es el paso de *Inquiry* el que se utiliza en la solución, ya que jamás se realiza una conexión con el dispositivo, sólo se desea detectar su presencia. El detalle específico de los datos que se envían durante la etapa de *Inquiry* escapa del alcance de este trabajo, pero sí es importante saber la forma en la que se envía. Desde la versión 1.2 de la especificación *bluetooth* se ha implementado la técnica que se conoce como *Adaptive Frequency Hopping* (*AFH*) [2]. Esta técnica consiste en ir cambiando entre los 83.5 canales de transmisión que existen en las bandas de *2.4GHz*, que utiliza *bluetooth*, con la intención de generar la menor cantidad de interferencia o colisión entre un dispositivo y otro. Existen otras técnicas que permiten reducir aún más las colisiones, como por ejemplo detectar la presencia de señales *Wi-Fi* que operan con el mismo tipo de banda *2.4GHz*, descartando esos canales.

Conociendo la forma de operar del protocolo *bluetooth* se puede determinar que un mínimo razonable de *beacons* que un *sender* puede detectar, sin problemas, está por el orden de los 50. En caso de no existir interferencia alguna, y sólo utilizando los canales proporcionados por la misma banda de radio de *2.4GHz*, se puede esperar que al menos 79 *beacons* puedan operar sin interferencia (por lo general no se utilizan los 83 canales disponibles). Tras conocer estos números, y entendiendo que aunque dos *beacons* operen en el mismo canal aún así pueden ser detectados, es que se puede determinar que el *hardware* de esta tecnología no es un limitante para la solución. Además, el mínimo seguro de 50 *beacons* corresponde a la cantidad de *beacons* detectables por un solo *sender*, por lo que incluir más *senders* en otras ubicaciones permite expandir este número.

[2]Adaptive Frequency Hopping for Reduced Interference between Bluetooth® and Wireless LAN - `https://www.design-reuse.com/articles/5715/adaptive-frequency-hopping-for-reduced-interference-between-bluetooth-and-html`

Para el caso contrario, en donde se desea usar muchos *senders* para monitorear un *beacon* tampoco existen problemas, puesto que al ser un escaneo pasivo no se produce interferencia entre los *senders*.

Por el lado del *software* utilizado, y la forma en la que se diseño el modelo de la base de datos tanto en el servidor central como en los *senders*, es posible agregar activos asociados a un cliente de manera muy sencilla, siendo necesario simplemente ingresar los registros en la tabla correspondiente y sincronizar los *senders* mediante un *script* verificador. La solución sólo necesita saber si el activo ingresado es un *sender* o un *beacon* y se debe indicar cuál es su identificador.

id	identifier	placeId	assetId	clientId	description
6	d0:b5:c2:cd:44:f0	NULL	2	1	Beacon Empleado B
7	d0:b5:c2:cd:b9:49	NULL	2	1	Beacon Empleado A
8	rpi-cinemark	NULL	1	1	Raspberry Cinemark
9	rpi-copec-sencillito	NULL	1	1	Raspberry Copec-Sencillito
10	d0:b5:c2:cd:3b:5b	NULL	2	1	Beacon Minimarket 1
11	d0:b5:c2:cd:38:5a	NULL	2	1	Beacon Minimarket 2
12	rpi-acrux	NULL	1	1	Raspberry Acrux
13	rpi-cruz	NULL	1	1	Raspberry Cruz del Sur

Figura 4.4: Screenshot de los activos registrados en el servidor central, Fuente: Elaboración Propia

Intercomunicación

Como cada *sender* y cada *beacon* tiene asociado un identificador único en el sistema y como cada registro guardado en las bases de datos tiene asociado una fecha y hora del suceso, es posible cruzar todos los datos recolectados por todos los *senders* respecto a todos los *beacons* para obtener una imagen más amplia del escenario observado. Teóricamente se podría obtener una ubicación estimada un poco más precisa haciendo uso de una técnica de trilateración, en donde se conoce el área aproximanda de detección de cada *sender* y se determina que si más de un *sender* detecta el mismo *beacon*, entonces ese *beacon* se encuentra dentro del área que se produce por la intersección del área de los *senders* que lo detectan. Una técnica similar se utiliza en el paper "Indoor Localization Method Based on Wi-Fi Trilateration Technique"[6] haciendo uso de las señales *Wi-Fi*.

Autonomía

Si bien el periodo de obtención de datos no fueron los 6 meses que el requerimiento solicitaba, en todo el periodo de prueba no fue necesario realizar ningún tipo de mantención, ni para los *senders*, ni para los *beacons*. El software de detección controló correctamente cualquier tipo de error que se pudiese presentar, evitando su término de ejecución de manera no deseada, y logró mantenerse operativo durante todo este periodo en todos los *senders* en los que se implementó. El servidor central mantuvo operativa a la *API* en todo momento recibiendo constantemente las solicitudes de actualización por parte de los *senders* y atendiendo a los clientes que deseaban ver el *dashboard web*. Por parte de los *beacons*, sus funcionamientos se mantuvieron constantes y no se observó ninguna baja de intensidad de señal durante el periodo de prueba, lo que da a entender que en ningún momento un *beacon* estuvo cerca de agotar su batería. Hay que recordar que según la información del fabricante un *beacon* puede funcionar hasta 12 meses de manera autónoma sin cambiar su batería.

Rentabilidad

Para determinar si la solución es rentable es necesario comparar el costo de implementación frente al impacto que la solución genera. Para determinar los costos de los dispositivos utilizados se realizó un promedio en base a varios proveedores encontrados por Internet, estos datos se presentan resumidos en la siguiente tabla.

Dispositivo	Valor Nacional Promedio (CLP)	Valor Internacional Promedio (USD)
Raspberry Pi Modelo 3B	39.000	50
Beacon Bluetooth	15.000	15

Cuadro 4.1: Tabla comparativa de precios, Fuente: Elaboración Propia

Para estimar el costo de implementación se tomó como ejemplo una empresa constructora de mediano tamaño que cuenta con dos bodegas de almacenamiento de activos y 10 activos fijos móviles a monitorear. Además se consideró el gasto de arriendo de una máquina virtual para implementar la *API* central por un periodo de 1 año, y se asumió que la empresa contaba con conexión a Internet. Para el cálculo del costo se utilizaron los valores nacionales, el resultado se presenta en la siguiente tabla.

Dispositivo	Cantidad	Costo Unitario	Sub-Total
Raspberry Pi Modelo 3B	2	39.000	78.000
Beacon Bluetooth	10	15.000	150.000
Hosting API Central	1	40.000	40.000
Baterías y Varios	1	30.000	30.000
		Total:	**298.000**

Cuadro 4.2: Costo estimado de implementar la solución por 1 año en pesos chilenos Fuente: Elaboración Propia

El resultado obtenido corresponde al costo aproximando que tendría implementar la solución y mantenerla operativa durante el primer año. Sin embargo, gran parte de este costo corresponde a la inversión inicial, ya que durante los siguientes años el único costo existente serían los costos de mantención del *Hosting* y los recambios de batería de los *beacons*.

Si se desease validar el requerimiento en el primer año de implementación, entonces los activos a monitorear deben estar valorizados en un total de al menos $2.980.000 CLP. Sin embargo, a partir segundo año el valor total de los activos debe ser de al menos $700.000 CLP para cumplir el requerimiento.

4.2. Análisis de Resultados

Uno de los objetivos que se desean cumplir es el de proporcionar información que pueda ayudar mejorar la gestión de los activos fijos y así tomar mejores decisiones. Si bien el experimento se realizó monitoreando el movimiento de empleados dentro de una oficina, esto no impidió obtener información interesante que podría ser traspasada al contexto de activos fijos móviles.

La información obtenida tras el análisis de los resultados se resume en la siguiente tabla:

	Empleado A	**Empleado B**
Promedio de minutos que el empleado llega tarde a la oficina.	11	48
Promedio de minutos que el empleado se queda después de la jornada laboral.	60	32
Promedio de minutos que el empleado llega tarde los días Lunes.	22	60
Promedio de minutos que el empleado se retira temprano los días Viernes.	2	116
Promedio de veces que el empleado sale de su oficina al día.	46	26

Cuadro 4.3: Información obtenida a partir de los datos, Fuente: Elaboración Propia

Este tipo de información obtenida a partir de los datos se puede aplicar al contexto de activos fijos móviles calculando, por ejemplo, cuáles son los horarios en los que se registra un mayor movimiento del activo para así programar mantenciones en los horarios de bajo movimiento. Si se hace uso de una bodega para guardar los activos se podría determinar cuáles son los que más se utilizan para posicionarlos lo más cerca de la entrada posible. También es posible monitorear que algún activo no sea utilizado en algún horario en particular, etc.

Se comprueba así que los datos almacenados en la base de datos permiten generar información relevante para los encargados de la gestión de los activos fijos, ayudando así a mejorar la toma de decisiones y mejorar el uso de los recursos disponibles.

Capítulo 5

Conclusiones

5.1. Conclusiones Generales

Tras desarrollar este trabajo se logró recorrer todo el proceso, desde la identificación del problema, hasta la propuesta de la solución final que hace uso de la tecnología *Bluetooth Low Energy* y que permitiría ser una alternativa viable a la actual tecnología *GPS* con todas sus ventajas y desventajas. Durante el desarrollo se pudo apreciar la gran versatilidad que tienen los dispositivos *Raspberry Pi* ya que, con su bajo costo, son capaces de ofrecer la potencia y compatibilidad necesaria para armar sistemas de bajo consumo energético. El hecho de que su sistema operativo principal esté basado en *Debian* permite que proyectos ya existentes, o tecnologías que ya sean conocidas por los desarrollares, sean implementadas directamente en estos equipos sin la necesidad de realizar mayores modificaciones, reduciendo así el tiempo de desarrollo. Para el caso de los *beacons* se observó la gran utilidad que se le puede dar a un componente tan simple como es un transmisor que sólo envía su identificación por ondas electromagnéticas, sin embargo, es esta simpleza la que le permite tener una autonomía tan impresionante como la observada y con tan reducido tamaño.

Respecto a las tecnologías utilizadas cabe destacar la potencia y simpleza que posee *Node.js* para operar haciendo uso del motor *javascript* de *Chrome*. Cada vez se ven más y más proyectos montados en este *framework* debido a su compatibilidad y a lo sencillo que es programar en este lenguaje respecto a otros lenguajes de programación cuya curva de aprendizaje es mucho más difícil y larga. Por su parte *LoopBack* a pesar de ser un *framework* relativamente nuevo posee la potencia suficiente para montar una *API* en poco tiempo y utilizando una cantidad mucho menor de recursos computacionales que otros *frameworks* similares, como es el caso de *Ruby on Rails*.

El hecho de que la solución implementada no haga uso del sistema *GPS* permite que no dependa de la recepción de las señales satelitales para su funcionamiento, sin embargo, esto también significa que la solución sólo se puede aplicar en lugares donde la empresa tenga acceso físico para colocar los *senders* correspondientes. Es por esto que el uso de *beacons* no sería de utilidad si se desease monitorear, por ejemplo, embarcaciones, siendo la tecnología *GPS* la mejor opción.

No hay que pensar que las tecnologías satelitales son excluyentes con el sistema desarrollado, de hecho, sería interesante implementar un *GPS* y un *beacon* a un camión minero, por ejemplo, para poder así monitorearlo tanto en la superficie como en el interior de una mina subterránea. Esto se podría complementar además con las redes *ZigBee* para extender el rango de detección bajo tierra.

Si bien esta memoria se enfocó en cómo monitorear y cómo controlar activos fijos mediante el uso de tecnologías inalámbricas, la implementación de una *API* central y las técnicas de

sincronización de datos pueden ser utilizadas en una variedad de problemas completamente distintos al visto. La sincronización e integridad de datos es un problema recurrente en la informática cuando los datos son recolectados de manera individual por distintos dispositivos para luego ser enviados y sincronizados por un ente central.

Como conclusión final hay que recalcar cómo las tecnologías emergentes pueden ir desplazando a tecnologías que se han establecido en el mercado durante largos periodos de tiempo. El área de la informática es una de las áreas profesionales más volátiles en términos de conocimiento ya que lo que se es capaz de hacer, y lo que no, depende fuertemente de las tecnologías y técnicas actuales. Es por esto que siempre habrá espacio para la innovación y siempre se podrá buscar una nueva solución a un problema tan antiguo como es la gestión correcta de activos en una empresa.

5.2. Cumplimiento de Objetivos

En la sección 1.3 se definieron los objetivos que la solución debía cumplir para ser considerada como válida. En esta sección se revisarán y se comentará en qué fueron cumplidos.

Objetivo General

Tras recolectar más de 3000 registros de movimiento y cerca de 10000 registros de variaciones de intensidad de señal, durante los 45 días que duró el proceso de recolección de datos, se pudo confirmar que es posible conocer la ubicación estimada de un activo fijo (en este caso un empleado) mediante la detección de señales *Bluetooth*. Para verificar la validez de estos resultados se tomó un periodo de tiempo de 5 días hábiles y se comparó los resultados obtenidos respecto a la primera y a la última detección de un empleado durante el día con los registros de la empresa respecto a la hora en que se marcó la entrada y la salida de su jornada laboral. Tras analizar estos resultados se observó que la diferencia entre los registros internos de la empresa y los registros detectados por la solución varían solamente en algunos minutos, esto se justifica en el tiempo que se demora el empleado en llegar a la oficina donde se ubica el *sender*.

Objetivos Específicos

- Análisis de Tecnologías: En el capítulo 2 se hizo un análisis sobre las diversas tecnologías que tenían el potencial de cumplir el objetivo general, estudiando sus ventajas y desventajas. Para la localización del activo se estudiaron las tecnologías basadas en *bluetooth* y en etiquetas *RFID*, además se estudiaron diferentes tecnologías para la transmisión de datos, ampliando la cantidad de opciones a utilizar según las condiciones existentes.
- Técnicas para la Gestión: El sincronizar los datos obtenidos por cada *sender* permitió obtener una imagen más global de la situación, a esto hay que agregarle la implementación del *Dashboard Web* que permite visualizar, en tiempo real, el estado de cada *beacon* respecto a cada *sender* y también permite revisar los registros existentes en la base de datos central. Toda esta información permite al encargado de gestión optimizar el uso de sus recursos y planificar de mejor forma el uso de sus activos.
- Prototipo Funcional: Para comprobar la eficacia de la solución propuesta se creó el prototipo funcional presentado, el cual consiste en el software implementado en los *senders*, la *API* Central y el *Dashboard Web*.

5.3. Mejoras y Recomendaciones Futuras

No cabe duda de que el trabajo presentado en esta memoria puede seguir mejorando en varios aspectos ya vistos, como también puede ir expandiéndose de forma horizontal añadiendo otro tipo de tecnologías para complementar la solución.

Cuando se presentaron las posibles tecnologías a utilizar una de ellas fue la tecnología de las etiquetas *RFID* pasivas, la cual se descartó debido al corto rango de detección que poseía, sin embargo, esta tecnología podría complementar la solución actual utilizándose como identificación del empleado que está haciendo uso del activo fijo monitoreado. Un ejemplo de este uso sería el asignarle una tarjeta *RFID* a un trabajador y solicitarle que la acerque a un sensor para abrir, de manera electrónica, una bodega que contiene activos fijos monitoreados por *beacons bluetooth*. De esta forma se puede realizar una asociación de un empleado con un movimiento de un activo, lo que genera un mayor control y seguridad de los flujos de activos en una empresa.

La comunicación entre un *sender* y la *API* central se implementó de una forma muy sencilla y, a causa de esto, muy insegura también. Como recomendación futura se propone implementar una comunicación encriptada mediante un certificado *SSL* [1] para evitar ataques del tipo *Man In The Middle* [2]. Además, el servidor central no hace ningún tipo de verificación de autenticidad del *sender*, por lo que cualquier solicitud que llegue con una identificación válida será recibida e ingresada a la base de datos. Para solucionar este problema se recomienda hacer uso de *tokens* de acceso que expiren al momento de ser utilizados, de esta forma se dificulta la posibilidad de falsificar la identidad de un *sender*.

[1] What is SSL - `http://info.ssl.com/article.aspx?id=10241`

[2] What is MITM - `https://www.incapsula.com/web-application-security/man-in-the-middle-mitm.html`

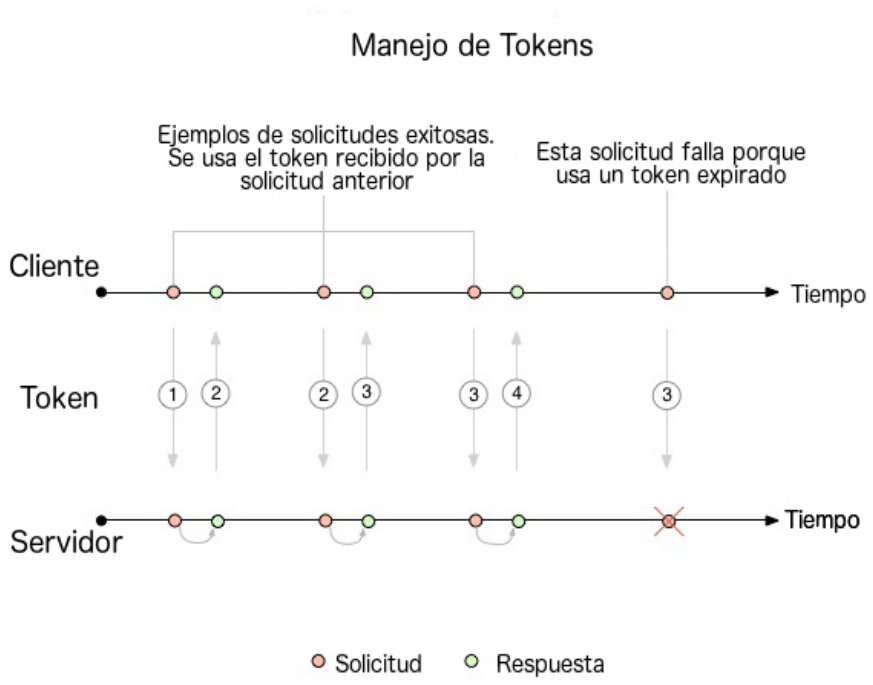


Figura 5.1: Esquema de manejo de Tokens, Fuente: https://github.com/lynndylanhurley/devise_token_auth - Traducción: Elaboración Propia

Bibliografía

[1] Alison K Brown y Mark A Sturza. *GPS tracking system*. US Patent 5,379,224. Ene. de 1995.

[2] A. Núñez-García Del Pozo y Jl Valbuena Duran. *Determinación de movimientos pequeños por procedimientos de trilateración*. 1992.

[3] Noppadol Chadil, Apirak Russameesawang y Phongsak Keeratiwintakorn. «Real-time tracking management system using GPS, GPRS and Google earth». En: *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*. Vol. 1. IEEE. 2008, págs. 393-396.

[4] Regis J Bates. *GPRS: general packet radio service*. 2001.

[5] Goutam Chakraborty, Kshirasagar Naik, Debasish Chakraborty, Norio Shiratori, David Wei. *Analysis of the Bluetooth device discovery protocol*. Springer Science Business Media LLC, 2008.

[6] Maxim Shchekotov. «Indoor localization method based on wi-fi trilateration technique». En: *Proceeding of the 16th conference of fruct association*. 2014, págs. 177-179.

Capítulo 6

Anexos

6.1. Modelo Base de Datos Servidor Central

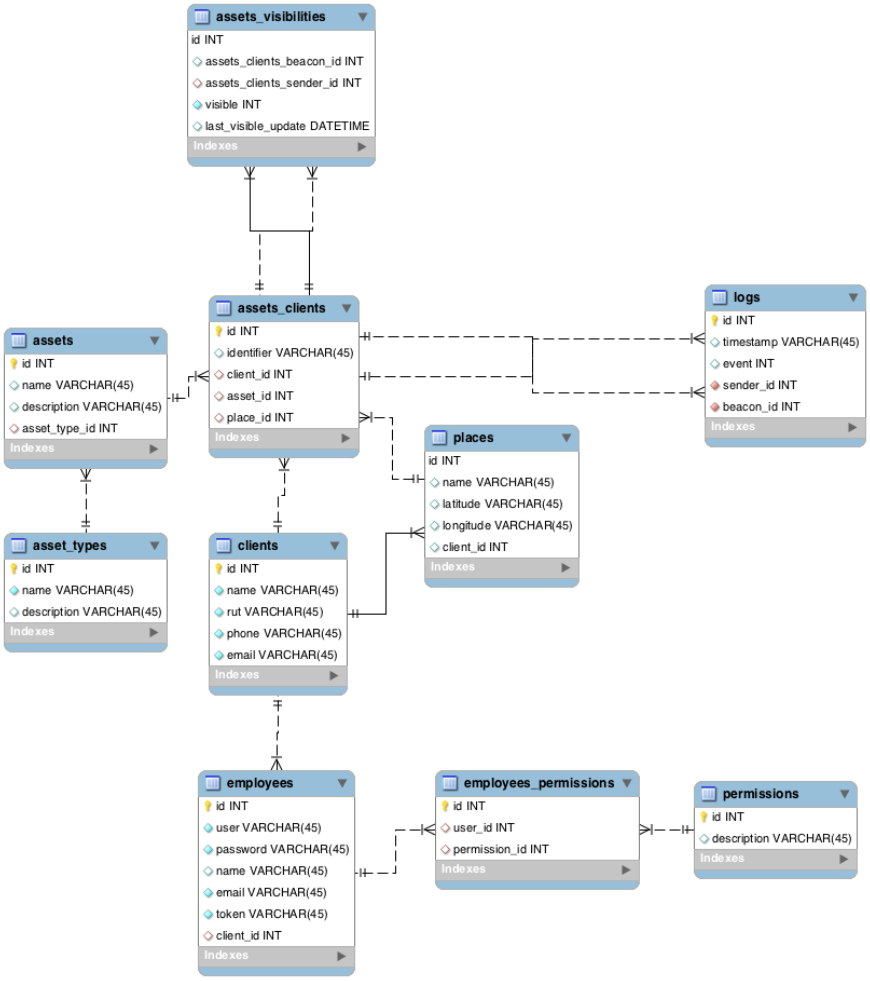


Figura 6.1: Modelo Base de Datos API, Fuente: Elaboración Propia

6.2. Modelo Base de Datos *Senders*

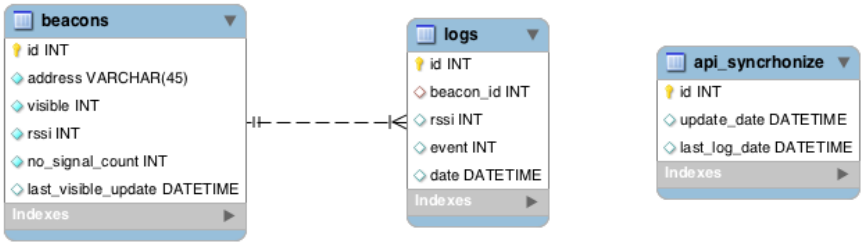


Figura 6.2: Modelo Base de Datos Senders, Fuente: Elaboración Propia