Desarrollo de un Sistema de Parqueadero Inteligente Mediante una Red LPWAN

Allauca Fajardo, Bryan David

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Lara Cueva, Román Alcides PhD.

11 de agosto del 2020

URKUND

Urkund Analysis Result

Analysed Document:	Tesis_BryanAllauca.pdf (D77308949)
Submitted:	7/30/2020 4:07:00 AM
Submitted By:	bdallauca@espe.edu.ec
Significance:	2 %

Román Lara Cueva
Firmado digitalmente por Román Lara Cueva
Fecha: 2020.07.30 14:46:13 -05'00'

Sources included in the report:

Tesis de Grado QoE-UNIAJC.pdf (D55040931)
https://repositorio.uloyola.es/bitstream/handle/20.500.12412/2223/TFG_MANDRADES.pdf?sequence=1&isAllowed=y

Instances where selected sources appear:

7

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Desarrollo de un Sistema de parqueadero inteligente mediante una red LPWAN**" fue realizado por el señor **Allauca Fajardo Bryan David** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 07 de agosto del 2020

Firma:

Román Lara Cueva

Firmado digitalmente por Román Lara Cueva
Fecha: 2020.08.07 17:40:39 -05'00'

Ing. Lara Cueva, Román Alcides PhD.

C.C 1713988218

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

RESPONSABILIDAD DE AUTORÍA

Yo, **Allauca Fajardo Bryan David**, con cédula de ciudadanía n° 1725044570, declaro que el contenido, ideas y criterios del trabajo de titulación**: Desarrollo de un Sistema de parqueadero inteligente mediante una red LPWAN** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 09 de agosto del 2020

Firma

.....

Allauca Fajardo Bryan David

C.C.: 1725044570

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN DE PUBLICACIÓN

Yo **Allauca Fajardo Bryan David,** con cédula de ciudadanía n° 1725044570, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Desarrollo de un Sistema de parqueadero inteligente mediante una red LPWAN** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 09 de agosto del 2020

Firma

.....

Allauca Fajardo Bryan David

C.C.: 1725044570

DEDICATORIA

El presenta trabajo de investigación lo dedico a todas aquellas personas que encuentran en las telecomunicaciones una forma de apoyar al desarrollo de las comunidades.

A la comunidad universitaria y científica que busca el avance tecnológico del Ecuador por medio de la investigación y el conocimiento.

A mis seres queridos que me ayudaron en este proceso y a todo aquel que quiera superarse, esperando que pueda motivarles a cumplir sus metas y seguir en el de camino de la colaboración para construir un mundo mejor.

Allauca Fajardo Bryan David

AGRADECIMIENTO

Agradezco a mis padres por todo el apoyo que me han dado, a mi madre Rocío por ser mi soporte principal y siempre estar conmigo, a mi padre Agustín por ser ejemplo de superación y guiarme con su conocimiento.

Agradezco a mi hermano mayor Santiago por ser mi modelo a seguir y a mi hermano menor Christian por su apoyo y motivación.

Agradezco a mis seres queridos por el apoyo brindado, a Jessica por ser parte fundamental en mi vida y apoyarme en todo.

Agradezco a mis amigos Marcelo, Gabriel, Jhonny, Stalin, Alejandro y a todos con quienes compartí momentos de alegría y siempre me brindaron su ayuda.

Agradezco a la Universidad de las Fuerzas Armadas ESPE y a sus docentes que compartieron su conocimiento, con los cuales logre conseguir esta meta.

Allauca Fajardo Bryan David

Índice De Contenidos

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El creciente aumento de la población junto con la utilización de vehículos se ve reflejada en un mayor tránsito vehicular y mayor requerimiento de plazas de estacionamiento, por lo que se necesita soluciones enfocadas a la movilidad y gestión eficiente de parqueaderos. El desarrollo del Internet de las Cosas nos permite crear una Smart University para gestionar los recursos de una Universidad y dar una mejor calidad de vida a la comunidad universitaria. El presente proyecto se enfocó en desarrollar e implementar un sistema de parqueadero inteligente mediante el despliegue de sensores ultrasónicos conectados por la red LPWAN Sigfox, los sensores ultrasónicos están en cada plaza de estacionamiento y detectan constantemente si una plaza está libre u ocupada. Estos datos se envían a la nube de Sigfox y son respaldados en una base de datos en la Internet, la cual se conecta a una aplicación móvil y presenta por medio de un mapa interactivo la ubicación y el estado de cada plaza al usuario en tiempo real. Las pruebas fueron realizadas con el despliegue de 4 sensores en los parqueaderos del campus de la ESPE y toma aproximadamente 3 segundos presentar la información en la aplicación móvil. Se concluyó que los usuarios no perciben el retardo y no presentan problemas para ver el estado de la plaza inmediatamente, con lo que se logra una disminución en el tiempo de búsqueda de una plaza libre de hasta 422 segundos y un ahorro en el consumo de combustible que deriva en un ahorro económico anual de hasta 37 dólares en gasolina súper y 33 dólares en gasolina extra.

PALABRAS CLAVE:

- **INTERNET DE LAS COSAS**
- **LPWAN**
- **SIGFOX**
- **RETARDO**

ABSTRACT

The increasing population growth along with the use of vehicles is reflected in greater vehicular traffic and a greater requirement for parking spaces, in this context solutions focused on mobility and efficient management of parking lots are needed. The development of the Internet of Things allows us to create a Smart University in order to manage the resources of a University and give a better quality of life to the university community. The present project focused on developing and implementing an intelligent parking system deploy ultrasonic sensors connected by the LPWAN Sigfox network, the ultrasonic sensors are in each parking space and constantly detect if a space is free or occupied. These data are sent to the Sigfox cloud and are backed up in a database on the Internet, these data will show the user the location and status of each square in real time through a mobile application using an interactive map.. The tests were carried out with the deployment of 4 sensors in the parking lots of the ESPE campus and it takes approximately 3 seconds to present the information in the mobile application. It was concluded that users do not perceive the delay and do not have problems to see the status of the place immediately, thus achieving a decrease in the search time for a free place of up to 422 seconds and a saving in fuel consumption which results in an annual economic saving of up to $ 37 in super gasoline and $ 33 in extra gasoline.

KEY WORDS:

- **INTERNET OF THINGS**
- **LPWAN**
- **SIGFOX**
- **TIME DELAY**

CAPÍTULO 1

DEFINICIÓN DEL PROYECTO

Introducción

En los últimos años se ha visto un incremento exponencial del uso de las Tecnologías de la Información y Comunicación (TIC) para el diseño y desarrollo de soluciones inteligentes, conceptos como el Internet de las Cosas (IoT del inglés *Internet of things*), virtualización y digitalización convergen para la creación de las *Smart,* las cuales engloban el desarrollo de soluciones inteligentes que proporcionan no solo una mejor calidad de vida sino también una mejora en aspectos económicos, sociales y ambientales. Dentro de las *Smart* se encuentra la *Smart University,* que no tiene una definición establecida, pero se considera a aquella Universidad que cumple con los objetivos de los ejes de los *Smart* (Sánchez, 2015).

Dentro del concepto de IoT está la utilización de tecnologías amigables con el medio ambiente y que ocupen bajos recursos por lo que las redes LPWAN (del inglés *Low-Power Wide-Area Network*) entran como recurso principal para el desarrollo de estas soluciones, por ser redes que tienen una cobertura amplia y un bajo consumo de potencia, aunque su velocidad de transmisión de datos es baja comparado con otras redes inalámbricas como *WiFi*, lo cual es suficiente para cubrir las demandas de proyectos IoT (ITU-T, 2012). Existen varias redes LPWAN que se pueden usar y una de ellas es la red Sigfox con tecnología de transmisión UNB (del inglés *Ultra Narrow Band*), que puede alcanzar grandes distancias usando canales estrechos del espectro y trabaja en las bandas de radio industriales, científicas y médicas (ISM. del inglés industrial, scientific, and medical) (Centenaro, Vangelista, Zanella, & Zorzi, 2016), en la actualidad a nivel de la Universidad de las Fuerzas Armadas ESPE la red Sigfox se ha desplegado dando cobertura a todo el campus. De la misma forma su facilidad de interconectar

sensores y que estos puedan mandar datos a la nube hace que esta red se presente como una solución efectiva (Sigfox, Sigfox Pagina Oficial, 2019).

Otro aspecto principal en IoT es que toda la información se suba a la nube, la red de sensores maneja datos los cuales deben estar en la Internet para posteriormente ser manipulados por aplicaciones que procesan y entregan resultados a los usuarios, un punto importante es la conexión en la Internet y disponer de la información desde cualquier parte del mundo. En la actualidad el despliegue de los *Smartphone* tiene un crecimiento exponencial y se estima que en el Ecuador por cada persona existe un *Smartphone* (INEC, 2017). Su uso es cada vez más común y está relacionado a cumplir varias tareas a través de aplicaciones móviles, por esta razón es importante la relación de aplicaciones web o móviles con las soluciones de proyectos IoT.

Uno de los ámbitos en los que se puede aplicar estos conceptos es en los parqueaderos inteligentes, que permite a los usuarios obtener información de las plazas de estacionamiento, así se puede buscar plazas de estacionamiento libres de forma rápida. La propuesta del presente proyecto es brindar esta información con la ayuda de sensores conectados a una red LPWAN y una aplicación móvil con la que los usuarios pueden interactuar, con esto se logra una reducción del tiempo de búsqueda y combustible dando un ahorro económico al usuario.

Antecedentes

Como un enfoque general de soluciones IoT, en (Borondo, 2015) se establecen algunos objetivos que se buscan para este tipo de proyectos como mejorar la calidad de vida y ayudar al medio ambiente, con una metodología opcional que se puede seguir para la implementación que va desde la toma de datos de los sensores hasta la entrega de datos a los interesados, pasando por etapas como la comunicación con la nube y la gestión de datos almacenados en la misma. En la etapa de comunicación con la nube,

en (Pérez, 2015) se habla de un dispositivo que usa la red de Sigfox para conectar sus datos a la Internet, este dispositivo se implementa para monitorizar diversos parámetros como: temperatura, estado de batería, posición o movimiento. Una vez obtenida la información necesaria se envía estos datos al *backend* de Sigfox y se puede visualizar en la página web oficial de la red, por último, se realiza una comparación de funcionamiento con otras tecnologías ya usadas, como es GPRS, para analizar sus ventajas y limitaciones.

La etapa de almacenamiento de los datos es muy importante en este tipo de proyectos para la gestión y administración de la información, existen la opción de usar el *backend* de Sigfox como en (Pérez, 2015) que permite una manera fácil de visualizar la información, pero no permite la manipulación de datos. Y la otra opción de usar una base de datos externa como en (Castillo, 2018), donde se implementa una red de sensores que permitan el control de variables del medioambiente como: temperatura, humedad, luz, entre otros. Estos sensores se conectan a microcontroladores como Arduino y *Raspberry,* los cuales codifican los datos en hexadecimal para enviar a la nube de Sigfox y posteriormente puedan ser usados por una aplicación web que envía peticiones HTTP y almacena los datos en una base local.

Entre las soluciones que engloba IoT para la creación de una *Smart University* están los sistemas de parqueaderos inteligentes, los cuales pueden estar implementados con diferentes funcionalidades y arquitecturas, un ejemplo de esto es el sistema en (Godoy, 2015), el cual está basado en un módulo RFID, Arduino y la tecnología *ZigBee*, su funcionamiento consiste en el uso de una tarjeta RFID que tiene cada usuario, al usar un plaza libre el usuario debe registrar su auto y validar la hora de entrada y salida, el prototipo tiene un sistema con teclado y pantalla led que permite la interacción no solo para el registro sino también para guiar a los nuevos usuarios hacia

una plaza libre, su objetivo es controlar las plazas de estacionamiento y fijar un sistema tarifado para una ciudad, su principal deficiencia es la necesidad de tener al usuario presente en el parqueadero para que pueda interactuar.

Con una arquitectura diferente tenemos el sistema descrito en (Márquez, 2015), diseñado con una red de sensores maestro y esclavo, un sensor que hace de maestro y los demás que funcionan como esclavos, el maestro manda las actualizaciones por medio de la interconexión de *ZigBee* el cual tiene un dispositivo conectado a la Internet que sube toda la información a un servidor para posteriormente ser usada por una aplicación móvil, las actualizaciones de los sensores envían el estado de las plazas de estacionamiento y la aplicación móvil muestra esta información al usuario con la finalidad de disminuir el tiempo de búsqueda de una plaza libre, la utilización de *Zigbee* y la interconexión a la Internet le hace un sistema remoto que se puede acceder desde cualquier parte pero con la limitante que se necesita un transmisor *Zigbee* para cada plaza y su cobertura es baja.

En la capa física se encuentran los sensores que son los encargados de recolectar la información necesaria para el proyecto, existe una gran variedad de sensores que se pueden usar para la verificación del estado de las plazas de parqueo, el proyecto en (Córdoba & Plazas, 2015) por ejemplo usa sensores ultrasónicos que sensan los obstáculos y la distancia a la que se encuentran así verifican si un automóvil está ocupando la plaza y mandan actualizaciones hacia un controlador central por medio de comunicación SPI (del inglés *Serial Peripheral Interface*), el controlador central cuenta con una conexión Ethernet que sirve para enviar datos a la Internet y puedan ser guardados en una base de datos, los sensores ultrasónicos a diferencia de los infrarrojos tienen la ventaja de medir mayores distancias y no ser afectados por la luz solar. Para la visualización de la información la mayoría de los proyectos la realizan por

medio de aplicaciones en *Smartphone*, este sistema y el descrito en (Márquez, 2015) usa una aplicación web y móvil para mostrar la información al usuario, entra esta se encuentra: la capacidad en plazas que tiene el parqueadero, la ubicación de las plazas tanto ocupadas como libres y sus tarifas.

Motivación e Importancia

A medida que aumenta la población urbana en el mundo, aumenta el crecimiento económico y la movilidad, esto se ve reflejado en el creciente número de personas que usan automóviles (Shoup, 2006). Gran parte del embotellamiento es causado por autos buscando un lugar para estacionarse, lo que significa horas perdidas. Si convertimos esas horas en consumo de combustible, emisión de CO2 y el efecto económico, la búsqueda de estacionamiento es muy costosa desde el punto de vista personal, económico y ambiental. Belloche (2015) estudió datos sobre el tráfico y la movilidad, en donde muestra que mientras el índice de congestión es más alto, hay más competidores o usuarios potenciales en el estacionamiento, lo cual da como resultado que los automóviles se muevan más lento. Adicional a esto si la señal de estacionamiento no está clara o no se encuentra en servicio, el tráfico aumentará y la circulación de vehículos se verá afectada en retraso de tiempo. Relacionando con lo anterior, si los usuarios pueden tener la información de disponibilidad de estacionamiento en tiempo real, podrán ajustar su tiempo de viaje sin perderlo recorriendo la ciudad en vano.

Un sistema de parqueadero inteligente puede ayudar a los conductores a encontrar lugares de estacionamiento de manera eficiente a través de las TIC. El desarrollo de estos parqueaderos inteligentes es gracias a los teléfonos inteligentes, estos pueden conectarse a la Internet y buscar información de tráfico, transporte, viajes, restaurantes y alojamiento en cualquier momento y en cualquier lugar. Además, el despliegue de parqueaderos inteligentes beneficia a la ciudad de forma económica,

puesto que el despliegue de una red de sensores inalámbricos es económico comparado con el despliegue tradicional de sensores cableados. Primero, los conductores obtienen una reducción en el tiempo de búsqueda de estacionamiento, disminuyen la contaminación ambiental al consumir menos gasolina y aliviar la congestión del tránsito vehicular a través de la información que proporciona el estacionamiento inteligente. Si los usuarios saben que no hay una plaza libre de estacionamiento cerca de su destino pueden usar el transporte público (Tahon, et al., 2010). Eso aumenta la tasa de uso de las instalaciones de la ciudad y sus ingresos. Segundo, si los conductores pueden encontrar el estacionamiento rápidamente, el tiempo que pasa la plaza de parqueo inactivo es más corto y los ingresos por estacionamiento aumentan. Tercero, una vez que el tráfico es fluido, aumenta la movilidad urbana y aumenta la capacidad de la ciudad, esto atrae a la población a realizar más actividades y se dinamiza la ciudad.

Para lograr el desarrollo de un sistema de parqueadero inteligente se debe contar con el despliegue de una red de sensores que tenga la posibilidad de conectarse con la Internet para permitir el intercambio de información. Una característica importante dentro de este entorno es la ubicuidad, lo que significa, la capacidad que tienen los dispositivos, en este caso sensores, de conectarse en cualquier lugar. Como la ubicuidad está estrechamente relacionada con la cobertura de la red inalámbrica, de aquí nace la importancia de utilizar tecnologías inalámbricas de gran alcance (Cárdenes, 2016). Una propuesta interesante es la utilización de redes LPWAN que tiene como característica principal ser de largo alcance, baja velocidad de transmisión y bajo consumo de energía. En la actualidad, Sigfox es uno de los primeros operadores que han desplegado una red inalámbrica de acceso mundial de tipo LPWAN que está

enfocada en desarrollo de soluciones IoT. La facilidad de interconectar sensores con la red y su bajo costo económico ha permitido su rápida penetración en el mercado.

Es por ello que se hace necesario la creación de un proyecto que tenga como finalidad el desarrollo de un sistema de parqueadero inteligente mediante una red LPWAN, en nuestro caso Sigfox, la cual nos permite cumplir con características del IoT entre las cuales está la interconectividad para dar acceso a la información y la comunicación mundial, y de la mano viene la escalabilidad un concepto que tiene relación con el número de dispositivos interconectados para generar una red de información mundial (Cobos, 2016). La capacidad de un sistema de poder controlar un número mayor de dispositivos es un punto fundamental ya que mientras más dispositivos se sumen a la red mayor será la información generada que puede ser analizada e interpretada para dar resultados más eficientes. En este punto Sigfox entra como actor principal ya que soporta aproximadamente 1.000.000 dispositivos por Gateway (Sigfox, Sigfox Build, 2019).

Alcance

El proyecto en mención busca informar en tiempo real sobre el estado de las plazas de parqueo de los laboratorios del Departamento de Eléctrica, Electrónica y Telecomunicaciones de la Universidad de las Fuerzas Armadas ESPE para que el usuario pueda ir directamente a un lugar que este libre sin perder tiempo buscando por cada plaza, para ello se utilizan 4 sensores ultrasónicos ubicados en cada plaza para poder detectar si la misma está libre u ocupada, estos sensores tienen una conexión alámbrica hacia la placa de desarrollo trinXtra la cual se conecta a la red de Sigfox y envían datos del estado de cada plaza a la nube de Sigfox solo cuando detecten que hubo un cambio de estado, es decir si pasa de estado libre a ocupado o viceversa.

Los datos enviados por los sensores son almacenados en la nube de Sigfox para posteriormente ser respaldados en una base de datos remota gratuita que estará disponible en la Internet y con una capacidad de espacio limitada, una aplicación móvil desarrollada en Android Studio se conecta a la base de datos a través de la Internet y extrae los estados actuales de cada plaza por medio de *Web Service*, esta aplicación móvil dispone de una interfaz gráfica que interactúa con el usuario y cuenta con información del estado de la plaza y la ubicación de la misma por medio del mapa de *Google maps*, de esta manera se visualiza en qué parte del campus el usuario puede estacionarse.

Objetivos

Objetivos General

Desarrollar e implementar un sistema de parqueadero inteligente mediante una red LPWAN que permita reducir el tiempo de búsqueda de una plaza de estacionamiento para mejorar la movilidad.

Específicos

- Comparar información sobre sistemas de parqueaderos inteligentes implementados en el país.
- Desplegar una red de sensores ultrasónicos para monitorizar el estado de las plazas de parqueo en el campus de la Universidad de las Fuerzas Armadas ESPE.
- Desarrollar una base de datos en la Internet para almacenar la información reportada por los sensores.
- Medir los tiempos de transmisión en la red Sigfox.
- Identificar el consumo de energía del sistema.

- Desarrollar una aplicación móvil que permita la interacción con el usuario y muestra el estado de las plazas.
- Disminuir el tiempo de búsqueda de una plaza de estacionamiento libre.

CAPÍTULO 2

MARCO TEÓRICO

Internet of Things (IoT)

IoT puede ser definida de muchas maneras, según la ISO (ILNAS, 2018) "Es una infraestructura de objetos interconectados, personas, sistemas y recursos de información junto con servicios inteligentes que les permiten procesar información del mundo físico y virtual para reaccionar", la ITU (ITU-T, 2012) lo define como "Una infraestructura global para la sociedad de la información, que permite servicios avanzados mediante la interconexión de cosas (físicas y virtuales) basadas en tecnologías de información y comunicación interoperables existentes y en evolución", y en (Guimarães, Benessia, & Curvelo, 2013) lo definen como “Una infraestructura de red global que une: objetos, dispositivos físicos y virtuales identificados de forma exclusiva mediante la explotación de capacidades de detección, comunicación y actuación de datos. En general se podría definir como una red abierta y completa de objetos inteligentes que tienen la capacidad de auto organizarse, compartir información, datos y recursos, reaccionar y actuar ante situaciones y cambios en el entorno como se representa gráficamente en la Figura 1, todo tipo de dispositivos conectados a nivel mundial.

Figura 1

Representación gráfica del IoT

Nota. El gráfico representa como todas las cosas están interconectadas en todo el mundo. Tomado de Somayya, M. (Agosto de 2015). Internet of Things: Smart Things. *International Journal of Future Computer and Communication, 4*(4), 250-253.

IoT es uno de los últimos avances en las TIC, que permite la conectividad global y gestión de sensores, dispositivos, usuarios e información. Se están desarrollando varias acciones que tienen el propósito de agregar casi todos los objetos físicos imaginables al IoT. En este punto los objetos son cada vez más capaces de caracterizarse no solo por un identificador único, sino también por transmitir su ubicación, automatizar las cosas, monitorizar y enviar datos sensibles que permitan tomar decisiones. IoT se refiere a la codificación y la creación de redes de objetos cotidianos y las cosas para hacerlos individualmente legibles por máquina y rastreables en Internet (Somayya, 2015). Actualmente, hay 9 mil millones de dispositivos interconectados y se espera que llegue a 24 mil millones de dispositivos para 2020. El Consejo Nacional de Inteligencia de los Estados Unidos prevé que "para 2025 los nodos de Internet pueden residir en paquetes de alimentos, muebles, documentos en papel y más" (Kosmatos, Tselikas, & Boucouvalas, 2011). IoT describe un mundo donde casi cualquier cosa puede conectarse y comunicarse de una manera inteligente como nunca antes.

Smart Things

Las cosas inteligentes hacen que el mundo que nos rodea sea más inteligente. *Smart Things* son un conjunto de dispositivos que se pueden monitorizar y controlar a través de un dispositivo central y servicios web. La idea de las cosas inteligentes y el IoT se popularizaron recientemente (Sterling, 2005) y permite que los usuarios puedan controlar y automatizar las acciones de hoy directamente por medio de aplicaciones de *Smart Things*. Actualmente los refrigeradores, las lavadoras, los televisores, los zapatos y los teléfonos inteligentes ya están en desarrollo y uso, pero el diseño de la experiencia del usuario para Internet de las cosas sigue siendo un concepto bastante nuevo. En la Figura 2 se representa el concepto de *Smart Things,* en donde se observa una interconexión entre todos los dispositivos. La transición actual de direccionamiento IPv4 a IPv6 proporcionará un número prácticamente ilimitado de direcciones IP públicas capaces de proporcionar acceso bidireccional y simétrico *Machine to Machine* (M2M) a miles de millones de *Smart Things* (Somayya, 2015).

Figura 2

Representación gráfica de Smart Things

Nota. La figura muestra una conexión de las cosas inteligentes y su comunicación. Tomado de Somayya, M. (Agosto de 2015). Internet of Things: Smart Things. *International Journal of Future Computer and Communication, 4*(4), 250-253

Al concepto *Smart Things* le acompañan cuatro capas arquitectónicas lógicas que tienen las siguientes funciones:

1. Se conectan al *Smart Things Hub* o también en algunos casos se puede conectar directamente a la nube.
2. Actúa como un *gateway* para obtener mensajes y eventos con dirección hacia o desde la nube.
3. Proporciona la abstracción y la capa de inteligencia, así como los servicios web que admiten la capa de presentación.
4. Proporciona la capa de presentación para cosas inteligentes en forma de aplicaciones móviles e IDE web (Somayya, 2015).

Dentro de *Smart Things-Cloud* también existen cuatro capas lógicas de arquitectura:

1. Conectividad: la cual es responsable de mantener la conectividad a *Smart Things Hubs* y aplicación *Smart Things Mobile.*
2. Procesamiento de eventos y enrutamiento: esta capa enruta eventos desde hubs o dispositivos a aplicaciones inteligentes que están suscritas a dispositivos o eventos específicos.
3. Aplicación: esta capa proporciona la capa de acceso a datos para cuentas, usuarios y dispositivos, también es responsable de la ejecución de aplicaciones inteligentes.
4. Servicios web: Esta capa proporciona los servicios web o la capa de interfaz de programación de aplicaciones (API, del inglés *Application Programming Interface*) que admite tanto las aplicaciones móviles como desarrolladores de sistema externo utilizando las API de *Smart Things* (Somayya, 2015).

Smart University

El enfoque principal de las *Smart University* está en el área de educación, esto no quiere decir que exista otro enfoque que impulsa el cambio en otros aspectos, entre sus intereses esta la gestión, la seguridad y la protección del medio ambiente. La disponibilidad de tecnología cada vez más actual se refleja en cómo se deben realizar los procesos que tienen gran relevancia en una era digital que cambia constantemente. Esto lleva a la adopción de una variedad de soluciones inteligentes en entornos universitarios para mejorar la calidad de vida y mejorar el rendimiento tanto de docentes, estudiantes y personal administrativo. A parte de esto se enfoca en proporcionar servicios de autoaprendizaje, automotivación y servicios personalizados para que los alumnos pueden asistir a cursos a su propio ritmo y poder acceder al contenido de aprendizaje personalizado (Akhrif & Hmina, 2018).

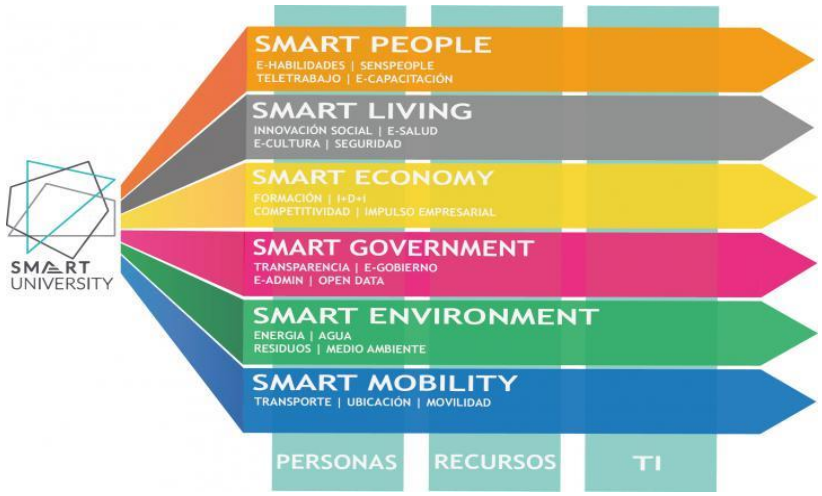
En resumen, una *Smart University* mejora la calidad de vida mediante el uso de Tecnologías de la Información (TI) con un proceso intensivo, global y sostenible que permite interconectar tanto actores como servicios con el fin de dar un beneficio innovativo a la comunidad. Para cumplir con este propósito la Unión Europea (EU) propuso seis pilares fundamentales o ejes que ayudan a crear una política de sostenibilidad, una eficiencia en el consumo de recursos aprovechándolos al máximo y una gestión correcta con propuesta a la innovación (Sánchez, 2015). Estos ejes propuestos por la EU se representan en la Figura 3 y son:

1. *Smart People*: Orientada a la comunidad.
2. *Smart Living*: Orientada al estilo de vida.
3. *Smart Economy*: Orientada a la parte económica.
4. *Smart Government*: Orientada a la administración y gestión gubernamental.
5. *Smart Environment*: Orientada al entorno.

6. *Smart Mobility*: Orientada a la movilidad.

Figura 3

Ejes de una Smart University



Nota. La figura muestra los ejes de una *Smart University* y el campo que cubre cada una en las soluciones. Tomado de Sánchez, B. J. (2015). Transformación Digital de la Universidad (XI). La experiencia en la UA. Universidad de Alicante, Alicante.

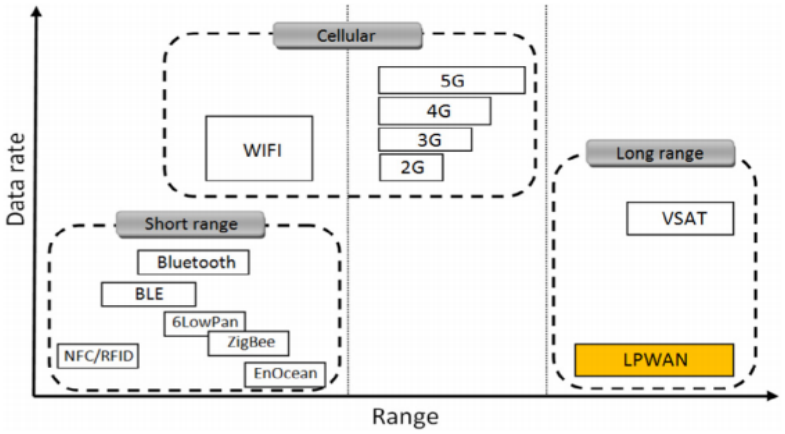
LPWAN

Una *Low-Power Wide-Area Network* es un tipo de red de telecomunicaciones inalámbrica de área amplia, diseñada para crear una red de sensores inalámbricos, esta red puede ser privada o pública dando servicio de infraestructura. LPWAN está ganando cada vez más popularidad a nivel de comunidades de investigación e industriales debido a sus características de comunicación de bajo consumo, largo alcance y bajo costo. Proporciona comunicación de gran alcance: hasta 10–40 km en zonas rurales y 1–5 km en zonas urbanas (Centenaro, Vangelista, Zanella, & Zorzi, 2016), una comparativa de velocidad de datos versus capacidad de alcance entre algunas tecnologías existentes se muestra en la Figura 4. Además, tiene una gran eficiencia desde el punto de vista energético (más de 10 años de vida útil de la batería) y económico. Muchas tecnologías LPWAN en el ancho de banda licenciada y no licenciada han surgido últimamente. Entre

ellos, Sigfox, LoRa y NB-IoT son las tecnologías emergentes líderes en la actualidad que implican muchas diferencias técnicas (Mekkia, Bajic, Chaxel, & Meyer, 2019).

Figura 4

Posicionamiento LPWAN entre las tecnologías de comunicación por radio: velocidad de datos vs capacidad de alcance



Nota. Tomado de Mekkia, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express, 5*(1), 1-7.

Las redes LPWAN se utilizan generalmente para proporcionar una transferencia de baja velocidad de datos entre puntos finales y un servidor central. El flujo de información se realiza principalmente en la dirección del enlace ascendente (desde los puntos finales hasta el servidor central) con acuses de recibo, configuración y actualizaciones del sistema utilizando la conectividad del enlace descendente. Los acuses de recibo se pueden usar para reducir las retransmisiones innecesarias del enlace ascendente, mientras que la configuración permite que la red responda a las condiciones cambiantes. La mayor parte de las aplicaciones LPWAN se centran en dar información sobre objetos estáticos (por ejemplo, electrodomésticos o carros estacionados) o cuasiestáticos (por ejemplo, animales). La mayoría de tecnologías LPWAN han adoptado comúnmente topologías en estrella con una estación base que

admite conexión con una gran cantidad de dispositivos finales distribuidos en un área potencialmente amplia (Real Wireless , 2015).

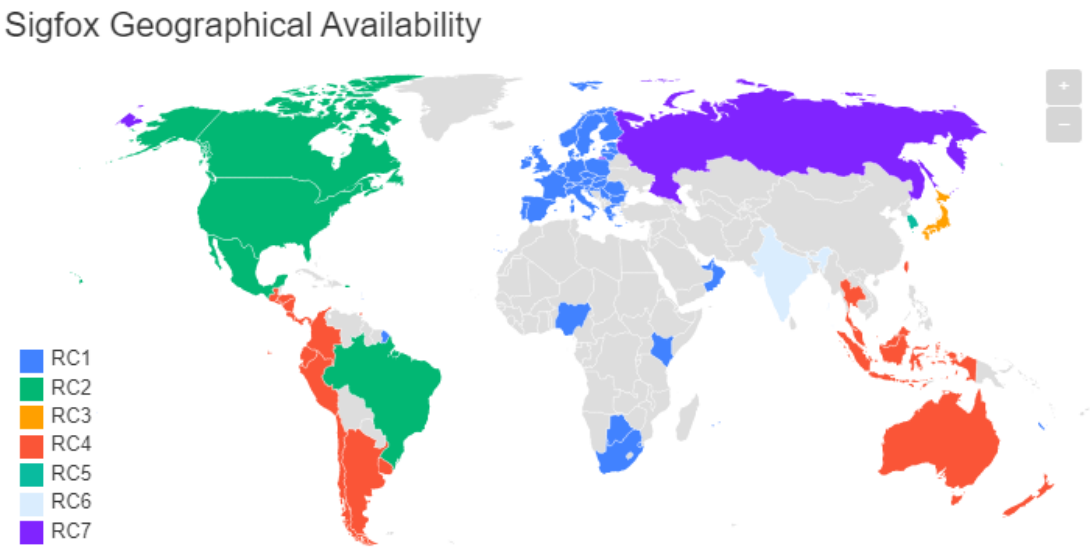
Sigfox

Sigfox es un operador de red a nivel mundial fundado en Francia 2009, construye redes inalámbricas LPWAN para interconectar dispositivos y sensores de baja potencia. Está dedicada para soluciones IoT económicas, fiables, de bajo consumo energético y donde no es necesario una alta velocidad de transmisión de datos. Su red está en despliegue a nivel global con mayor participación en Europa (Sigfox, Sigfox Build, 2019). Entre las características de funcionamiento y tecnologías que usa se encuentra:

- Emplea Codificación de Desplazamiento de Fase Binaria Diferencial (DBPSK).
- Emplea Codificación de Desplazamiento de Frecuencia Gaussiana (GFSK).
- Usa frecuencias en la banda de radio ISM (Europa: 868 MHz y EEUU/América Latina: 902 MHz).
- Red de tipo LPWAN.
- Tecnología UNB.
- Se maneja por 4 zonas como se observa en la Figura 5:
    - RC1 Europa.
    - RC2 USA, México.
    - RC3 Japan.
    - RC4 América Latina.
    - RC5 Sur Korea.
    - RC6 India.
    - RC7 Rusia.

Figura 5

Disponibilidad Sigfox por zonas



Nota. La figura muestra la disponibilidad de la red Sigfox a nivel mundial separado por zonas. Tomado de Sigfox. (2019). *Sigfox Build*. Obtenido de https://build.sigfox.com/sigfox

Protocolo de Sigfox

El protocolo de Sigfox tiene algunas características que se mencionan a continuación:

- Autonomía: Consumo de energía reducido con esto se logra una larga duración de la batería.
- Simplicidad: No existe la señalización o una configuración compleja para que se pueda conectar los diversos dispositivos.
- Eficiencia en costo: Optimización del hardware y del costo de transmisión con paquetes de suscripción bajos.
- Mensajes pequeños: La red permite mensajes con información de datos pequeños (12 bytes).
- Complementario: Puede ser usado como solución de respaldo o complementaria a tecnologías como *WiFi*, Bluetooth o redes móviles.

Sigfox permite el intercambio de mensajes tanto de subida como de bajada con los dispositivos conectados de forma asimétrica, es decir la cantidad de mensajes de subida son más que la cantidad de mensajes de bajada permitidos según los paquetes de suscripción disponibles.

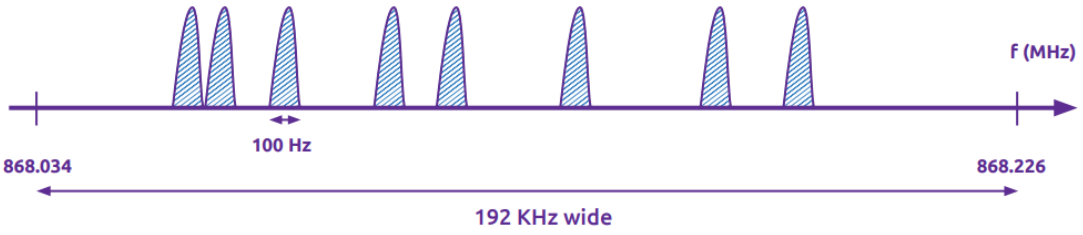
UNB Ultra Narrow Band

La tecnología UNB utiliza canales de RF estrechos para proporcionar una alta sensibilidad del receptor y un rango extendido, pero con velocidades de bits muy bajas, 100 o 600 bps dependiendo de la región de operación. Los sistemas UNB usan un canal de banda estrecha para cada transmisión por separado: *uplink* (UL) o *downlink* (DL) y el espectro que se utiliza permite que transmisiones múltiples UL y DL puedan ocurrir simultáneamente gracias al acceso múltiple por división de frecuencia (FDMA, del inglés *Frecuency Division Multiple Access*). La potencia de transmisión se estipula por dispositivo siendo ocupada por un canal relativamente estrecho. Existen algunos sistemas patentados, entre ellos: Sigfox, *Telensa*, *Nwave* (Real Wireless , 2015).

Las redes Sigfox operan a un enlace ascendente de 100 bps con un límite de carga útil de 12 bytes y utilizan bandas ISM. Además, los dispositivos finales están limitados en la cantidad de transmisiones que pueden hacer por día. Sigfox ha introducido una comunicación bidireccional básica para atacar las demandas del mercado a su oferta unidireccional. Esta tecnología está utilizando 192kHz de la banda pública disponible para el intercambio de mensajes a través del aire. Cada mensaje tiene un ancho de banda de 100Hz como se observa en la Figura 6 y es transferido con una velocidad de datos de 100 o 600 bps, esto depende de la región de operación (Sigfox, Sigfox Build, 2019).

Figura 6

Tecnología Sigfox basada en UNB



Nota. La figura muestra una representación de cómo trabaja UNB en función de la frecuencia. Tomado de Sigfox. (2019). *Sigfox Build*. Obtenido de https://build.sigfox.com/sigfox

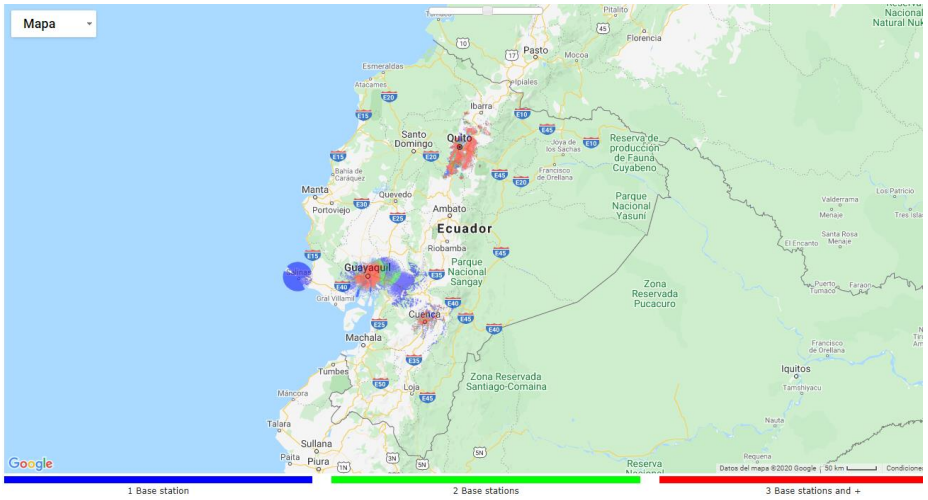
Lo que implica que un mensaje de 12 bytes tarda aproximadamente 2 segundos en ser transmitido. Por otro lado, no existe tráfico de sincronización ni señalización entre el dispositivo y la red con el fin de reducir el consumo energético y ancho de banda. El mensaje será transmitido 3 veces en 3 frecuencias diferentes mediante técnicas de salto de frecuencias (*frequency hopping*). De esta manera se logra conseguir una mayor robustez gracias a la diversidad en tiempo y en frecuencia (Sigfox, Sigfox Build, 2019).

Cobertura

Sigfox al ser una red licenciada procedente de Francia y su despliegue es reciente a nivel de América Latina por lo que no se dispone de cobertura total a nivel de Ecuador, en la Figura 7 se puede observar la cobertura de la red Sigfox a nivel nacional.

Figura 7

Cobertura Sigfox Ecuador

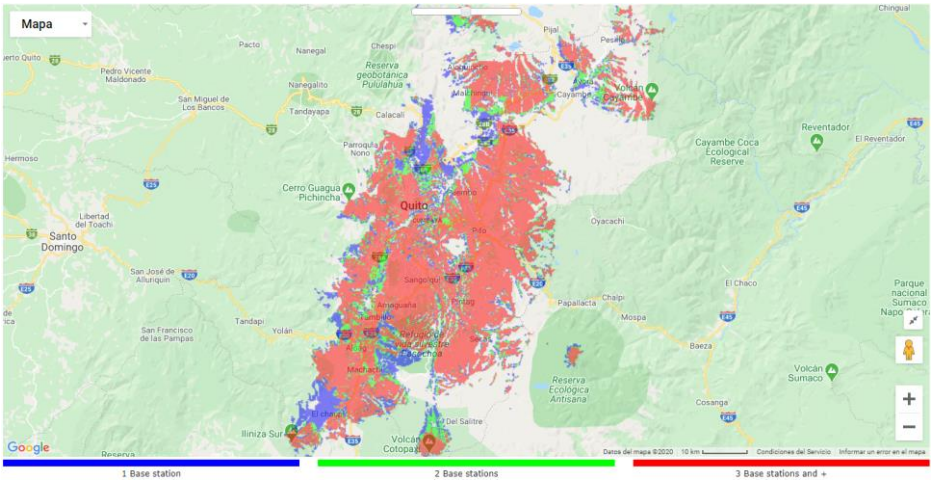


Nota. La figura muestra la cobertura de la red Sigfox a nivel de Ecuador. Tomado de Sigfox. (Febrero de 2020). *Backend Sigfox*. Obtenido de https://backend.sigfox.com/welcome/news

En cuanto a nivel de Pichincha en la Figura 8 se puede observar la cobertura que tiene la red Sigfox, y se aprecia que en Quito y sus alrededores existe una cobertura total.

Figura 8

Cobertura Sigfox Quito



Nota. La figura muestra la cobertura de la red Sigfox a nivel de Pichincha. Tomado de Sigfox. (Febrero de 2020). *Backend Sigfox*. Obtenido de https://backend.sigfox.com/welcome/news

Android

Android es una plataforma de software y un sistema operativo desarrollado para dispositivos móviles y *tablets*, basado en el *kernel* de *Linux* 2.6 y desarrollado por Google la cual compró la empresa en 2005, más tarde *Open Handset Alliance* (OHA) contribuyó con el desarrollo de este sistema operativo y fue presentado en 2007. *Android Open Source Project* (AOSP) rige el ciclo de mantenimiento y desarrollo de Android. Permite a los desarrolladores escribir código administrado en el lenguaje de programación Java, controlando el dispositivo a través de las bibliotecas de Java desarrolladas por Google (Ahamed, 2016).

Android está disponible como código abierto, lo que quiere decir que es una pila de software de código abierto de descarga gratuita para dispositivos móviles que incluye un sistema operativo, middleware y aplicaciones clave basadas en *Linux* y Java. Google lanzó el código de Android como código abierto bajo la Licencia Apache. Android tiene numerosos desarrolladores que escriben aplicaciones en todo el mundo. En primer lugar, los desarrolladores escriben su script en Java y luego descargan las aplicaciones de sitios de terceros o tiendas en línea (Kirthika, Prabhu, & Visalakshi, 2015). Para complementar el entorno operativo, se diseñó e implementó un marco de aplicación específico para *Android*. Por lo tanto, *Android* puede describirse mejor como una pila de solución completa, que incorpora el sistema operativo, los componentes de middleware y las aplicaciones. En Android, el *kernel* modificado de *Linux 2.6* actúa como la capa de abstracción de hardware (HAL) (Ahamed, 2016). Para resumir, el entorno operativo de *Android* se puede etiquetar como:

- Una plataforma abierta para el desarrollo móvil.
- Un diseño de referencia de hardware para dispositivos móviles.
- Un sistema alimentado por un núcleo Linux 2.6 modificado.

- Un entorno de tiempo de ejecución.
- Una aplicación e interfaz de usuario (UI, del inglés *User Interface*) marco de referencia.

Versiones de Android

Android se está actualizando constantemente desde su lanzamiento con nuevas y mejoradas versiones. Estas actualizaciones del sistema operativo base se enfocan en corregir errores y agregar nuevas características de entorno gráfico y operativo más cómodo. En general, cada nueva versión del sistema operativo *Android* se desarrolla bajo un nombre código y un número de versión. Entre las versiones lanzadas hasta la actualidad se encuentran:

1. *Eclair: 2.0 - 2.1.*
2. *Froyo: 2.2.*
3. *Gingerbread: 2.3.3 - 2.3.7.*
4. *Ice Cream Sandwich: 4.0.3 - 4.0.4.*
5. *Jelly Bean: 4.1 - 4.3.*
6. *Kit Kat: 4.4.*
7. *Lollipop: 5.0 - 5.1.*
8. *Marshmallow: 6.0.*
9. *Nougat: 7.0.*
10. *Oreo: 8.0.*
11. *One: 9.0*

Lenguaje de Programación

Para empezar un lenguaje es un sistema de comunicación entre dos personas. Entre los idiomas naturales básicos más comunes está el español, el inglés, etc. Estos idiomas son utilizados para comunicarse entre varias categorías de personas. Ahora los

dispositivos electrónicos no pueden entender el mismo lenguaje de las personas, por lo tanto, existen lenguajes de programación de computadoras especialmente desarrollados para que pueda pasar datos e instrucciones a la computadora y realizar un trabajo específico.

Los lenguajes de programación son lenguajes de notación artificiales desarrollados para ser utilizados en la creación de instrucciones codificadas en la computadora para su posterior ejecución por la misma. Generalmente se componen de una serie de reglas que determinan el significado de las expresiones escritas en el lenguaje. Cada lenguaje de programación es útil con su propio traductor llamado intérprete o compilador (Usman, Akeem, & Gbenga, 2016).

Base de Datos

Las bases de datos y los sistemas de bases de datos se han vuelto parte común en la vida de la sociedad actual, una gran parte de nosotros realizamos varias actividades todos los días que implican de alguna forma interacción con una base de datos. Por ejemplo, al ir a un banco a depositar o retirar fondos, al hacer una reservación en un hotel o al comprar artículos en línea. Incluso al comprar en un supermercado, esto a menudo actualiza automáticamente la base de datos que contiene el inventario de artículos. El avance de la tecnología de medios nos ha permitido almacenar imágenes, audios y video. Las bases de datos y la tecnología de bases de datos tienen un gran impacto en el uso creciente de las computadoras. Las bases de datos desempeñan un papel importante en la mayoría de las áreas donde se emplea computadoras, incluyendo comercio electrónico, negocios, ingeniería, medicina, educación, etc. (Elmasri & Navathe, 2011).

La base de datos es una colección de datos (hechos conocidos que pueden ser grabados y que tienen un significado implícito) relacionados. Un ejemplo simple son los

nombres, números de teléfono y direcciones de las personas que conoce. Una base de datos tiene algunas propiedades las cuales son:

- Una base de datos representa algún aspecto del mundo real, llamado el universo del discurso (UoD, del inglés *Universe of Discourse*).
- Una base de datos es una recopilación de datos lógicamente coherente con algún significado inherente.
- Una base de datos está diseñada, construida y poblada con datos para un propósito específico.

En resumen, una base de datos tiene una fuente de la cual los datos se derivan o se originan, también tiene un porcentaje de interacción con eventos del mundo real y una población que está interesada en el contenido de la misma. En la base de datos pueden ocurrir eventos que hace que la información cambie (Elmasri & Navathe, 2011).

Web Services

Las *webs services* es un sistema de software que utiliza un conjunto de estándares y protocolos diseñado para la interacción M2M que se comunican a través del protocolo HTTP usando serialización XML (del inglés *Extensible Markup Language*). Las *webs services* proporcionan un medio estándar de interoperabilidad entre aplicaciones de software que se ejecutan en diferentes plataformas. Se caracterizan por su gran interoperabilidad y extensibilidad, así como por sus descripciones procesables por máquina, gracias al uso de XML. Los programas que proporcionan servicios simples pueden interactuar entre sí por medio de *web services* para ofrecer servicios sofisticados de valor agregado, entre el uso más común está el intercambio de datos de distintas aplicaciones sin importar en que lenguaje de programación estén desarrolladas (Mustafa, 2016).

CAPÍTULO 3

MATERIALES Y MÉTODOS

Estudio de Mercado del Software

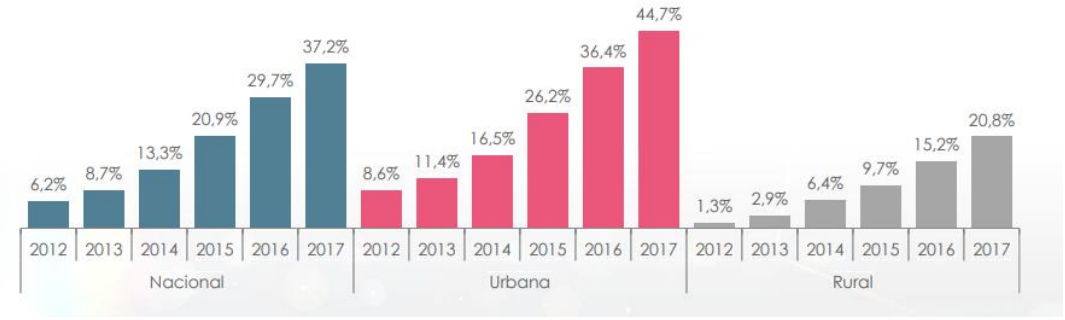
La evolución constante de la tecnología ha permitido el desarrollo de teléfonos inteligentes, los cuales se han convertido en parte fundamental de cada persona, según las cifras en el Ecuador expuestas por el (INEC, 2017), en relación a los teléfonos inteligentes y celulares se observa que:

- 9 de cada 10 hogares poseen al menos un teléfono celular.
- El 58% de las personas de 5 años en adelante tienen por lo menos un celular activo.
- La población con mayor tenencia de celulares está entre los 25 y 44 años con el 82.9%.
- El 32% de la población tiene un teléfono inteligente.
- La población con mayor tenencia de teléfonos inteligentes está entre los 25 y 34 años con el 62.7% seguido de la población entre 16 y 24 años con 57.4%.

Adicional a esto se puede observar en la Figura 9, que el crecimiento de tenencia en cuanto a teléfonos inteligentes es considerable en los últimos 5 años. Por lo que se puede estimar que cada vez hay más teléfonos inteligentes y su uso se vuelve indispensable.

Figura 9

Tendencia de personas que tienen teléfonos inteligentes



Nota. La figura muestra la tendencia de personas que usan teléfonos inteligentes en el Ecuador. Tomado de INEC. (2017). Tecnologías de la Información y Comunicación. Ecuador: TIC.

Al haber una tendencia de crecimiento muy considerable en cuanto al uso de teléfonos inteligentes, es indispensable relación de las soluciones IoT con teléfonos inteligentes, ya que por medio de las aplicaciones móviles desarrolladas los usuarios pueden interactuar con los datos y acceder a los beneficios que dan las soluciones IoT.

Selección del Sistema Operativo

Los dispositivos móviles tienen un gran poder para correr diversas aplicaciones y cumplir varias funciones, pero todo esto se ejecuta sobre un sistema operativo el cual está encargado de administrar los recursos del dispositivo, sincronizar el hardware y organizar los archivos y directorios en los elementos de almacenamiento. Entre los sistemas operativos más usados en los teléfonos inteligentes se encuentran:

- BlackBerry OS
- iOS
- Windows Mobile
- Android

Cada uno de estos tiene sus propias características y la gama de celulares que usan estos sistemas operativos, en (Márquez, 2015) se realizó un análisis de cada uno de ellos con sus características más relevantes, los resultados se muestra en la Tabla 1.

Tabla 1

Comparación de Sistemas Operativos móviles

Características	BlackBerry OS	iOS	Windows Mobile	Android
Software propietario	Si	Si	Si	No
Multitarea	Si	Si	Si	Si
Interacción	Touch Teclado Trackball	Touch	Touch Teclado	Touch
Programado	Java C++	C++	C++	Java C++
Ejecuta Java	Si	No	Si	Si
Kit desarrollo	No	Si	No	Si

Nota. Esta tabla representa las características principales de los diferentes sistemas operativos. Tomado de Márquez, R. M. (2015). *Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos.* Universidad de las Fuerzas Armadas Espe, Sangolquí.

En base a este análisis, se escogió Android como sistema operativo para el desarrollo del sistema de parqueadero inteligente, ya que presenta características importantes para nuestras necesidades. Un punto importante a destacar es que Android se desarrolla mediante código abierto y es gratuito por lo que las aplicaciones se pueden ejecutar sin ningún tipo de licencia o permiso, adicional a esto, permite la creación de entornos multitarea lo que significa que pueden ejecutarse muchas aplicaciones a la vez. Otro punto importante es el lenguaje de programación sobre el que está implementado, Java es un lenguaje de programación de los más antiguos y completos, tiene soporte en muchos sistemas operativos y puede interactuar con otros lenguajes como HTML o PHP que para nuestro sistema es muy importante.

Android Studio

Android Studio es una colaboración entre JetBrains y Google. Android Studio está construido sobre IntelliJ de JetBrain, por lo que su funcionalidad es un superconjunto de IntelliJ. Android Studio es revolucionario porque simplifica el proceso de desarrollo de aplicaciones Android y hace que sea mucho más accesible de lo que había sido anteriormente. Android Studio es ahora el IDE oficial para Android (Google Developer Training Team, 2016).

Android Studio es notablemente consistente en todos los sistemas operativos. De hecho, las interfaces de usuario en Windows y Linux son casi idénticas. Sin embargo, los usuarios de Mac OS encontrarán que algunas de las ubicaciones de sus menús y algunos atajos de teclado son diferentes. Android Studio proporciona herramientas para las fases de prueba y publicación del proceso de desarrollo, y un entorno de desarrollo unificado para crear aplicaciones para todos los dispositivos Android. El entorno de desarrollo incluye plantillas de código con código de muestra para características comunes de la aplicación, amplias herramientas de prueba y marcos, y un sistema de compilación flexible. Existen algunos requerimientos del sistema para poder usar Android Studio (Smyth, 2015), el desarrollo de aplicaciones de Android se puede realizar en cualquiera de los siguientes tipos de sistemas:

- Windows 2003 (32 bits o 64 bits)
- Windows Vista (32 bits o 64 bits)
- Windows 7 (32 bits o 64 bits)
- Windows 8 / Windows 8.1
- Windows 10
- Mac OS X 10.8.5 o posterior (solo sistemas basados en Intel)
- Sistemas Linux con versión 2.11 o posterior de GNU C Library (glibc)

- Mínimo de 2 GB de RAM (recomendable 4 GB)
- 1.5 GB de espacio disponible en disco

Selección de la tecnología LPWAN

La principal característica del sistema desarrollado es la utilización de la tecnología inalámbrica LPWAN, que como se describió antes es una red inalámbrica de largo alcance y bajo consumo. Ahora, existen varias redes LPWAN, cada una con sus respectivas características que pueden llegar a ser ventajas o desventajas dependiendo las necesidades del sistema que se quiera desarrollar. Entre las principales redes LPWAN están:

- Sigfox
- LoRa
- NB-IoT

En la Tabla 2, se muestran las características más importantes para diferenciar una de otra. Todo esto fue el resultado de un análisis completo realizado en (Mekkia, Bajic, Chaxel, & Meyer, 2019), aquí se comparó cada tecnología con el fin de presentar en resumen los puntos técnicos más representativos que pueden ser considerados cuando se está desarrollando una solución.

Tabla 2

Resumen de redes LPWAN: Sigfox, LoRa, NB-IoT

	Sigfox	**LoRaWAN**	**NB-IoT**
Modulación	BPSK	CSS	QPSK
Frecuencia	Banda ISM no licenciada	Banda ISM no licenciada	Frecuencia licencia LTE
Ancho de Banda	100 Hz	250 kHz y 125 kHz	200 kHz
Velocidad de datos máxima	100 bps	50 kbps	200 kbps
Bidireccional	Limitada/Half-duplex	Si/ Half-duplex	Si/ Half-duplex
Máximo mensajes por día	140 (UL), 4 (DL)	Ilimitada	Ilimitada

	Sigfox	LoRaWAN	NB-IoT
Máximo tamaño de payload	12 bytes (UL), 8 bytes (DL)	243 bytes	1600 bytes
Rango	10 km (urbana), 40 km (rural)	5 km (urbana), 20 km (rural)	1 km (urbana), 10 km (rural)
Inmunidad a la Interferencia	Muy alta	Muy alta	Baja
Autenticación y encriptación	No soportada	Si (AES 128b)	Si (Encriptación LTE)
Velocidad de datos adaptativa	No	Si	No
Handover	Los dispositivos finales no se unen a una sola estación base	Los dispositivos finales no se unen a una sola estación base	Los dispositivos finales no se unen a una sola estación base
Localización	Si (RSSI)	Si (TDOA)	No
Permite red privada	No	Si	No
Estandarización	Sigfox en colaboración con ETSI	LoRa-Alliance	3GPP

Nota. Esta tabla representa características principales técnicas de las diferentes redes LPWAN más conocidas. Tomado de Mekkia, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express, 5*(1), 1-7.

En la Tabla 3, muestra los costos relacionados al desarrollo con las diferentes redes LPWAN.

Tabla 3

Costos relacionados a las redes: Sigfox, LoRa, NB-IoT

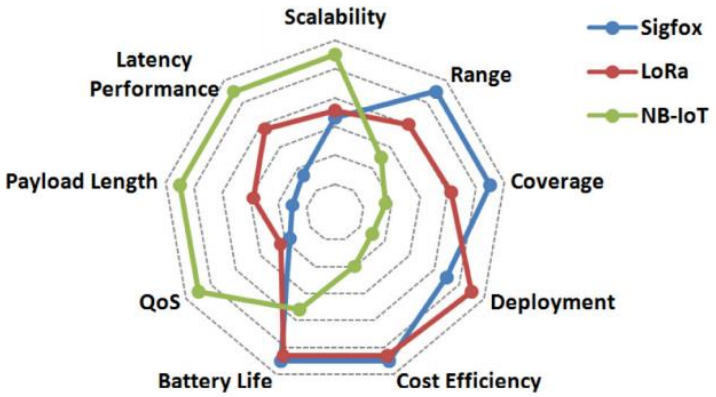
	Costo por espectro	Costo de despliegue	Costo dispositivo final
Sigfox	Gratis	>4000$/estación base	<2$
LoRa	Gratis	>100$/Gateway >1000$/estación base	3-5$
NB-IoT	>500 M$/MHz	>15000$/estación base	>20$

Nota. La tabla representa los costos que implica cada red LPWAN. Tomado de Mekkia, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express, 5*(1), 1-7.

En la Figura 10, muestra una comparación de las ventajas que tiene cada red LPWAN haciendo una relación en términos de IoT.

Figura 10

Ventajas en términos IoT de redes: Sigfox, LoRa, NB-IoT



Nota. La figura muestra una comparacion de las redes LPWAN en los puntos principales del IoT. Tomado de Mekkia, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express, 5*(1), 1-7.

En base al análisis realizado anteriormente, se escoge como tecnología inalámbrica a la red LPWAN Sigfox, ya que presenta ventajas importantes en cuanto al desarrollo de una solución IoT, entre las principales está el costo del dispositivo final y como desarrolladores de soluciones nosotros no cubrimos el costo de despliegue de la red. Otra característica importante que cubre nuestra necesidad es el rango de cobertura que tiene la red y la inmunidad a la interferencia, lo que permite conectar dispositivos en lugares alejados de la estación base. La vida de la batería es un aspecto importante en soluciones IoT y Sigfox nos da un buen desempeño en cuanto a ahorro de energía. Por último, el sistema de parqueadero inteligente es una solución IoT de bajo transferencia de información, es decir que no es necesario una gran velocidad de transmisión porque la información que se maneja es pequeña y los datos que se envían al día son pocos, lo que significa que Sigfox se acopla a nuestras necesidades.

Arduino

Arduino es una plataforma para desarrollar prototipos en código abierto basada en un hardware y software simple y fácil de usar. Consiste en una placa de circuito llamada microcontrolador que puede ser programada por medio de un software llamado Arduino IDE (del inglés, *Integrated Development Environment*) que se utiliza para escribir en lenguaje de programación y cargar el código del IDE en la placa física (Arduino, 2019). Entre las características más importantes están:

- Arduino pueden leer señales de entradas que pueden ser analógicas o digitales provenientes de sensores, procesar esta información y convertir en señales de salida o respuesta ejecutando una acción por ejemplo activar un motor, encender/apagar un LED, activar una sirena, comunicarse con otros dispositivos, etc.
- Las funciones que desempeña Arduino pueden ser controladas por medio de un conjunto de instrucciones enviadas al microcontrolador de la placa a través de Arduino IDE el cual se denomina software de carga.
- Arduino no necesita un programador adicional, esto es una pieza de hardware que se usa para cargar un nuevo código en la placa. En lugar a esto simplemente usa un cable USB conectado al computador.
- IDE Arduino utiliza una versión simplificada del lenguaje de programación C ++, el cual es uno de los más antiguos y conocidos por lo que es fácil de usar.

Tipos de Placas Arduino

Existe una gran variedad de placas Arduino en el mercado, cada una tiene diferentes características técnicas, entre las cuales están: el número de entradas y salidas disponibles, tipos de salidas que soporta, tipo de comunicación con otros dispositivos, arquitectura del procesador, memoria, tamaño entre otros. Por otro lado,

aunque existen varios tipos de placas Arduino todos ellos se programan por el mismo Software, Arduino IDE (Arduino, 2019). En la Tabla 4, se enlistan los Arduino más comunes en el mercado y algunas de sus características.

Tabla 4

Tipos de Arduino

Nombre	Voltaje de operación	Velocidad de reloj	I/O digitales	Entradas analógicas	PWM	UART
Arduino Uno	5V	16MHz	14	6	6	1
Arduino mini	5V	16MHz	14	8	6	1
Arduino Mega	5V	16MHz	54	16	14	4
Arduino Leonardo	5V	16MHz	20	12	7	1
Arduino Ethernet	5V	16MHz	14	8	6	1
Arduino Due	3.3V	64MHz	54	12	12	4
Arduino Pro	3.3V	8MHz	14	6	6	1

Nota. La tabla muestra las placas Arduino más conocidas y sus características técnicas que diferencia una de otra. Tomado de Arduino. (Noviembre de 2019). *Arduino*. Obtenido de Arduino: https://www.arduino.cc/

Kit desarrollo thinXtra

Thinxtra devkit Xkit es un dispositivo de desarrollo y pruebas de prototipos, cuenta con un conjunto completo de características y accesorios que permite configurar fácilmente una solución de IoT. El kit dispone de todo lo necesario para comenzar a usar la red Sigfox disponible a nivel mundial (Thinxtra, 2019). Tiene los siguientes sensores integrados:

- Acelerómetro de 3 ejes.
- Sensor digital de temperatura y presión.
- Interruptor de láminas.
- Sensor de luz.

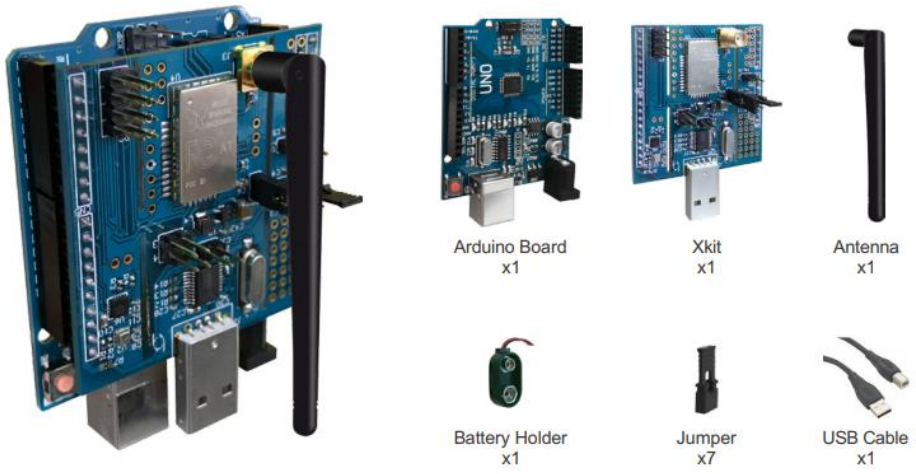
- LED rojo y azul.
- Botón pulsador

También se incluye una parte de hardware para que sea completamente funcional, Figura 11:

- Cable USB.
- Soporte de batería
- Antena
- Clon de placa Arduino Uno R3
- Conectividad de 1 año a la red Sigfox.

Figura 11

Hardware incluido en el Kit thinXtra



Nota. El gráfico muestra el kit de desarrollo con los componentes que fueron proporcionados para desarrollar el sistema. Tomado de Thinxtra. (Noviembre de 2019). *thinxtra Empowering Internet of Things*. Obtenido de https://www.thinxtra.com/

La placa de transmisión Sigfox es compatible con Arduino (o clon), PC, placa STMicro Nucleo y Raspberry Pi. Entre las zonas disponibles para la conectividad a la red Sigfox están: RCZ1 | RCZ2 | RCZ4 (Thinxtra, 2019).

Sensor Ultrasónico

El sensor ultrasónico modelo HCSR04 mostrado en la Figura 12, es compatible con Arduino y otros microcontroladores. Usa un sonar para determinar la distancia de un objeto que se encuentra al frente como lo hacen los murciélagos o los delfines. Ofrece una excelente detección de rango sin contacto con alta precisión y lecturas estables. Tiene un rango aproximado de detección que va desde los 2 cm a 400 cm. Entre sus principales características esta que su funcionamiento no se ve afectado por la luz solar o material negro. Consta de un transmisor ultrasónico y un módulo receptor (Technologies, 2013). Tiene un consumo promedio de 4mA y para calcular la distancia se debe usar la siguiente ecuación:

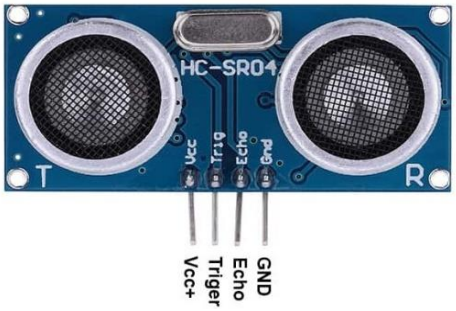
$$distancia = T_{echo} \times \frac{V_{sonido}}{2} \qquad (1)$$

Considerando que la velocidad del sonido es de 340m/s o 29us/cm la ecuación queda de la siguiente manera:

$$distancia = \frac{T_{echo}}{58} \qquad (2)$$

Figura 12

Sensor Ultrasónico HC-SR04



Nota. Tomado de Technologies, C. (2013). Datasheet HCSR04 Ultrasonic Sensor.

Lenguajes de programación seleccionados para el desarrollo

Para la implementación del sistema de parqueadero inteligente se seleccionó Java como lenguaje de programación para desarrollar la aplicación móvil que será ejecutada en los dispositivos Android. PHP es el lenguaje de programación que nos permite crear las *webs services* para realizar el intercambio de datos entre la aplicación móvil y la base de datos, también nos ayuda a ejecutar funciones que permitan guardar y editar la información en la base de datos desde el backend de Sigfox. Por último, C++ es el lenguaje de programación que se usa en Arduino IDE para programar el microcontrolador Arduino.

Java

Java fue desarrollado por un equipo dirigido por James Gosling en Sun Microsystems, más tarde adquirido por Oracle en 2010. Originalmente llamado Oak fue diseñado en 1991 para su uso en chips integrados en dispositivos electrónicos de consumo, en 1995 fue renombrado como Java y rediseñado para desarrollar aplicaciones web. Su rápido aumento y amplia aceptación se remontan a sus características de diseño, particularmente su promesa de que puede escribir un programa una vez y ejecutarlo en cualquier lugar. Como lo indicó su diseñador, Java es simple, orientado a objetos, distribuido, interpretado, robusto, seguro, arquitectura neutral, portátil, de alto rendimiento, multiproceso y dinámico. Java es un lenguaje de programación de funciones generales con todas las funciones que se puede utilizar para desarrollar aplicaciones sólidas de misión crítica. Hoy en día, se emplea no solo para la programación web, sino también para el desarrollo de aplicaciones independientes a través de plataformas en servidores, computadoras de escritorio y dispositivos móviles (Liang, 2015).

Se utilizó para desarrollar el código para comunicarse y controlar el robot rover en Marte. Java inicialmente se volvió atractivo porque los programas Java pueden ejecutarse desde un navegador web. Dichos programas se llaman applets. Los applets emplean una interfaz gráfica moderna con botones, campos de texto, áreas de texto, botones de radio, etc., para interactuar con los usuarios en la Web y procesar sus solicitudes. Los applets hacen que la Web sea receptiva, interactiva y divertida de usar. Los applets están incrustados en un archivo HTML (del inglés *HyperText Markup Language*). Java es muy popular para desarrollar aplicaciones en servidores web. Estas aplicaciones procesan datos, realizan cálculos y generan páginas web dinámicas. Muchos sitios web comerciales se desarrollan utilizando Java en el backend. Java es un lenguaje de programación versátil: puede usarlo para desarrollar aplicaciones para computadoras de escritorio, servidores y dispositivos portátiles pequeños. El software para teléfonos celulares Android se desarrolla utilizando Java (Liang, 2015).

PHP

PHP es un lenguaje simple, pero con gran potencial, fue diseñado para crear y escribir contenido HTML. PHP se puede ejecutar en todos los principales sistemas operativos, desde variantes de Unix, incluidos Linux, FreeBSD, Ubuntu, Debian y Solaris hasta Windows y Mac OS X. También puede ser utilizado con todos los servidores web líderes, incluidos Apache, Microsoft IIS y los servidores Netscape/iPlanet (Tatroe, MacIntyre, & Lerdorf, 2013). Existen 3 formas en las que PHP puede ser usado:

1. Server-side scripting: PHP originalmente se diseñó para crear contenido web dinámico, aunque hoy en día hay varios lenguajes con este propósito PHP sigue siendo el más adecuado para esta tarea. Para poder generar HTML es necesario un analizador PHP y un servidor web. PHP también se ha hecho popular para

generar documentos XML, gráficos, animaciones Flash, archivos PDF y mucho más.

2. Command-line scripting. PHP tiene la habilidad de ejecutar secuencias de comandos desde la línea de comandos, como Perl, awk o el shell de Unix. Se puede usar el command-line scripts para ejecutar tareas de administración del sistema, como copia de seguridad y análisis de registros. En algunos casos incluso se puede crear scripts que ejecuten tareas no visuales.
3. Client-side GUI applications. Usando PHP-GTK se puede escribir aplicaciones GUI multiplataforma completas en PHP.

Una de las características más importantes de PHP es su amplio soporte para bases de datos. PHP admite todas las bases de datos más comunes: MySQL, PostgreSQL, Oracle, Sybase, MS-SQL, DB2 y las bases de datos de estilo NoSQL más recientes como: SQLite y MongoDB. Todo esto hace que crear páginas web con contenido dinámico desde una base de datos sea sencillo (Tatroe, MacIntyre, & Lerdorf, 2013).

C++

C ++ es un lenguaje de programación de alto nivel que permite a un ingeniero de software comunicarse eficientemente con una computadora. Está diseñado como un puente entre el programador y la computadora. La idea es permitir que el programador organice un programa de una manera que pueda entender fácilmente y el compilador luego traduce el programa a lenguaje de máquina. Los datos en una computadora se almacenan como una serie de bytes. C ++ organiza esos bytes en datos útiles. C ++ es un lenguaje altamente flexible y adaptable. Desde su creación en 1980, se ha utilizado para una amplia variedad de programas entre los cuales: firmware para

microcontroladores, sistemas operativos, aplicaciones y programación de gráficos (Oualline, 1995).

Uno de los objetivos principales del lenguaje C ++ es organizar las instrucciones en componentes reutilizables. Se puede escribir programas mucho más rápido si reutiliza la mayor parte de su código de otro lugar. Los grupos de módulos reutilizables se pueden combinar en una biblioteca. Una de las principales innovaciones de C ++ es la idea de combinar datos e instrucciones en una construcción llamada clase u objeto. La programación orientada a objetos le permite agrupar datos con las operaciones que se pueden realizar en esos datos (Oualline, 1995).

CAPÍTULO 4

DESARROLLO Y PRUEBAS

Estado de las Plazas de Parqueo

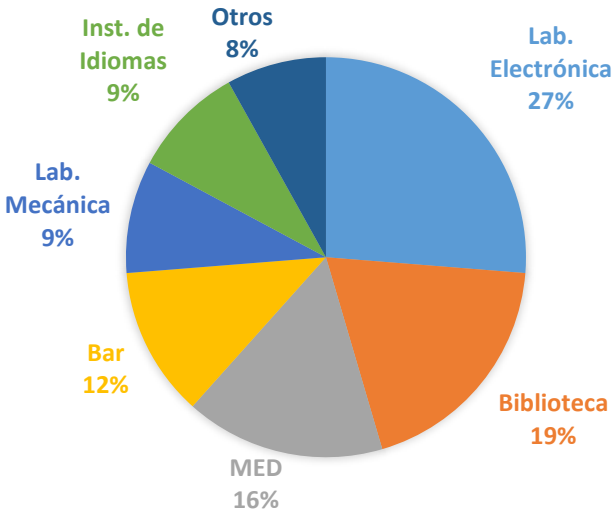
Un estudio realizado en (Márquez, 2015) revela datos importantes acerca del estado de las plazas de parqueo de la Universidad de las Fuerzas Armadas ESPE, en este estudio se realizó una encuesta a la comunidad universitaria con preguntas relacionadas al tiempo de búsqueda de una plaza de estacionamiento, el intervalo de permanencia en las plazas entre otras. Los datos más relevantes usados para el desarrollo del sistema de parqueadero inteligente se muestran a continuación.

Zona de parqueo

La Figura 13 muestra las zonas de parqueo disponibles en la Universidad de las Fuerzas Armadas ESPE y el porcentaje de su utilización.

Figura 13

Zonas de parqueos más utilizados



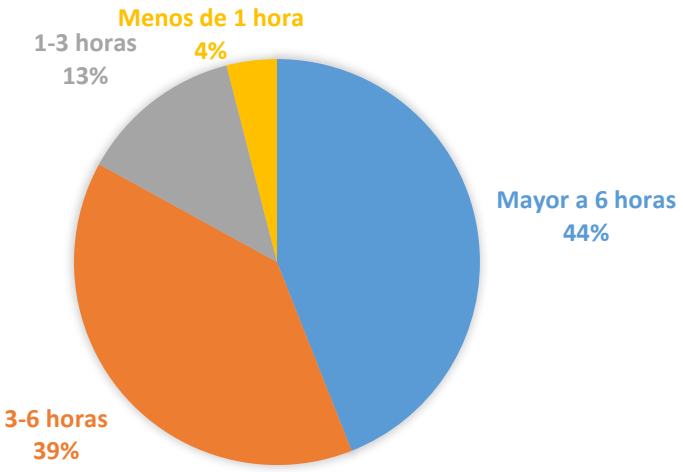
Nota. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí.

Tiempo de permanencia

La Figura 14 muestra los datos sobre el tiempo que utiliza un automóvil la plaza de parqueo o el tiempo que los usuarios dejan sus automóviles parqueados.

Figura 14

Tiempo de permanencia en las plazas de parqueo



Nota. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí.

Estos datos pueden ser traducidos al número de automóviles que usan una misma plaza de estacionamiento en un día, considerando que las horas activas en un día dentro de la jornada universitaria son 15 (desde las 07:00 horas a 22:00 horas) y que una plaza de estacionamiento este siempre ocupada, dan como resultado lo siguiente:

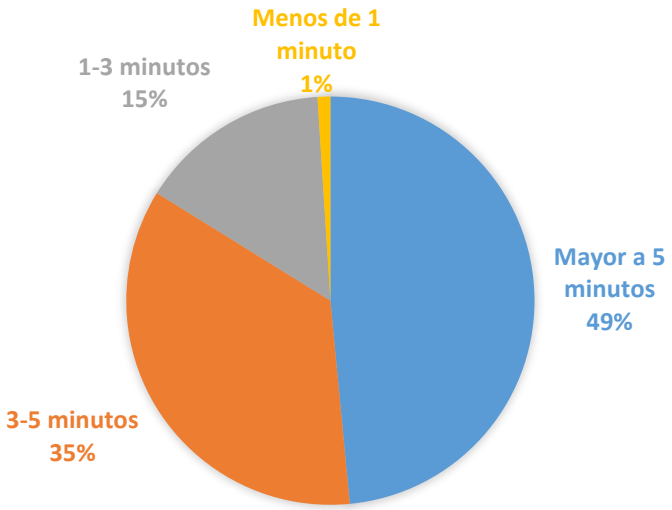
- Menos de 2 automóviles por día 44%.
- De 2 a 5 automóviles por día 39%.
- De 5 a 15 automóviles por día 13%.
- Más de 15 automóviles por día 4%.

Tiempo de búsqueda de una plaza libre

La Figura 15 muestra los datos sobre el tiempo que toma buscar una plaza de estacionamiento libre en el campus universitario, estos datos son importantes ya que uno de los objetivos principales es la reducción de estos tiempos.

Figura 15

Tiempo que toma buscar una plaza de parqueo libre



Nota. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí.

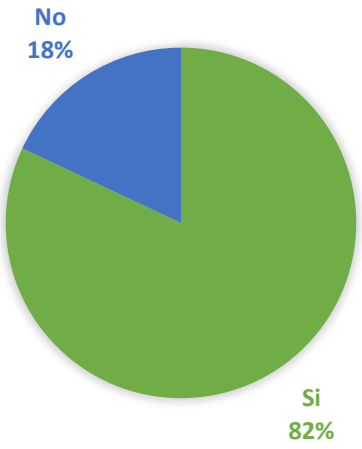
Perspectiva del Usuario

Por último, el estudio arrojo datos sobre la perspectiva que tiene el usuario, es decir que si consideran que existe una pérdida de tiempo al momento de buscar plazas de estacionamiento libre y si piensan que una aplicación móvil que muestre el estado de las plazas de estacionamiento es una solución para reducir la pérdida de tiempo que conlleva buscar plazas libres.

En la Figura 16 se muestra el resultado sobre la perspectiva del usuario al considerar si existe una pérdida de tiempo al momento de buscar una plaza libre, el 82% de los encuestados respondieron que sí y un 18% respondieron que no.

Figura 16

Pérdida de Tiempo

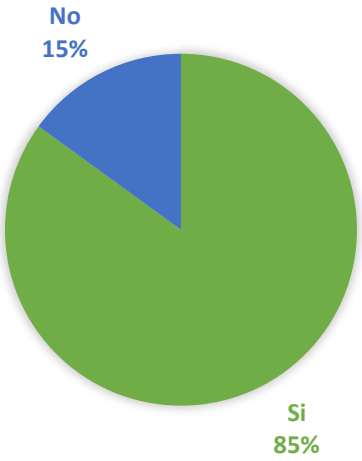


Nota. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí.

En la Figura 17 se observa los datos de la encuesta en cuanto a si los usuarios consideran que una aplicación móvil es una solución para reducir la pérdida de tiempo de búsqueda de plazas de estacionamiento libres. El 85% de encuestados respondieron que sí y el 15% respondieron que no.

Figura 17

Solución por medio de una aplicación móvil



Nota. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí.

Diseño del Sistema en General

Para el diseño del sistema de parqueadero inteligente se tomó como base los datos analizados previamente, los cuales nos permiten justificar la zona de parqueo en la Universidad de las Fuerzas Armadas ESPE donde implementar el sistema y el número de plazas de estacionamiento que podemos cubrir con un dispositivo Sigfox. La zona seleccionada fue los Laboratorios de Electrónica, por ser las plazas de parqueo más utilizadas por la comunidad universitaria con el 26%.

Por otro lado, el número de plazas de estacionamiento que podemos cubrir por cada dispositivo de comunicación inalámbrica Sigfox son 4. Ya que la tecnología inalámbrica LPWAN seleccionada Sigfox nos limita el número de mensajes enviados de un dispositivo por día. Recordando que el paquete para el desarrollo de soluciones proporcionado por la Universidad fue de 140 mensajes por día con un tamaño máximo de payload de 12 bytes, también se toma en cuenta el número máximo de automóviles que usan la misma plaza de estacionamiento en un día. Basado en los datos de la

sección 0, existe el 96% de probabilidad de que una plaza sea ocupada por un máximo de 15 automóviles por día, con este número aseguramos que los mensajes serán suficientes para cubrir el estado de las plazas todo el día. La fórmula utilizada para el cálculo es la siguiente:

$$N = \frac{M}{Na \times 2} \quad (3)$$

En donde:

- *N* es el número máximo de plazas que se puede cubrir.
- *M* es el número máximo de mensajes por día que podemos mandar.
- *Na* es número de automóviles que usan una misma plaza en un día.

A la variable *Na* se le multiplica por dos ya que por cada automóvil que ocupe una plaza de parqueo se enviaran dos mensajes, el primero al ocupar la plaza (cambio de estado libre a ocupado) y el segundo al desocupar la plaza (cambio de estado ocupado a libre).

Lo que nos da como resultado:

$$N = \frac{140}{15 \times 2} = 4.66 = 4 \quad (4)$$

El resultado se redondea al entero inmediato inferior para asegurar que los mensajes sean suficientes y no se acaben, ya que esto puede provocar que una plaza no pueda actualizar su estado.

Características del Sistema de Parqueadero

Para el desarrollo e implementación del sistema de parqueadero inteligente se deben recordar las características seleccionadas anteriormente y todos los componentes que forman parte del sistema completo. Entre los componentes a nivel hardware tenemos:

- Kit de desarrollo thinXtra (Arduino Uno y módulo Sigfox).

- 4 sensores ultrasónicos.
- Batería 3.7V.
- Panel Solar.
- Módulo de carga de batería.
- Convertidor DC-DC Step-up.

Entre los componentes a nivel software tenemos:

- Backend de Sigfox.
- Base de datos.
- Aplicación Móvil.

Entre las características tenemos:

- El sistema fue implementado en los Laboratorios de Electrónica de la Universidad de las Fuerzas Armadas ESPE.
- Sigfox es la tecnología inalámbrica LPWAN seleccionada.
- Un microcontrolador está encargado de controlar la red de sensores y el TX/RX Sigfox.
- El sistema esta alimentado por una batería la cual se cargará por medio de un panel solar.
- Un dispositivo de comunicación Sigfox está encargado de la comunicación de los estados de 4 plazas de estacionamiento.
- Cada sensor ultrasónico debe sensar constantemente el estado de la plaza de estacionamiento.
- Los estados de las plazas de estacionamiento son 2 (libre y ocupado).
- Un máximo de 15 automóviles puede ocupar una misma plaza en un día.
- La base de datos está en la Internet.

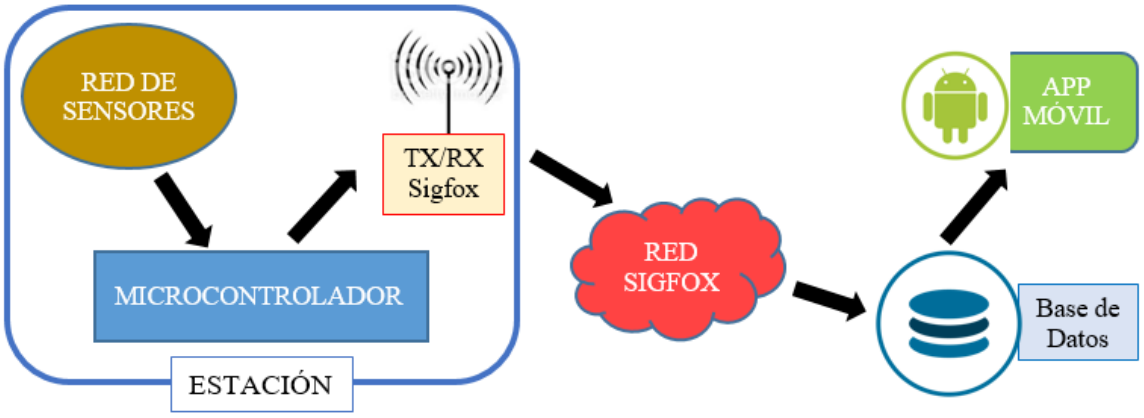
- El backend de Sigfox reenvía la información hacia nuestra base de datos cada vez que llega un nuevo mensaje.
- La aplicación móvil es desarrollada para el Sistema Operativo Android.
- La aplicación móvil trae la información desde la base de datos y actualiza el estado de las plazas de estacionamiento en tiempo real.

Diagrama de bloques del Sistema

Teniendo en cuenta los componentes y las características que debe tener el sistema, se realizó un diagrama general del sistema completo de parqueadero inteligente como se muestra en la Figura 18, la cual nos da una perspectiva general de los bloques principales y su funcionamiento en el sistema desarrollado.

Figura 18

Diagrama de bloques del Sistema de parqueadero inteligente



- Red de Sensores: Este bloque está conformado por los 4 sensores ultrasónicos, los cuales son los encargados de percibir si hay algún automóvil ocupando la plaza o no, y enviar al microcontrolador la información del estado actual de la plaza.
- Microcontrolador: Este bloque está conformado principalmente por un Arduino UNO el cual es el encargado de controlar y administrar la información recibida

por la red de sensores, esta información es transformada a estados de las plazas y enviada a la red por medio del módulo de comunicación de Sigfox.

- TX/RX Sigfox: En este bloque está el módulo que se encarga de conectar a la red LPWAN Sigfox y el microcontrolador, este chip tiene la función de mandar y recibir datos con la red.
- Red de Sigfox: Este bloque contiene toda la infraestructura de la red de Sigfox, es decir tanto la red de transporte como la red de acceso y la conexión con la Internet.
- Base de Datos: Este bloque contiene la base de datos que está en la Internet, en esta base de datos se guardan el estado actual de la plaza de estacionamiento, también guarda un historial de todos los mensajes enviados por la red de Sigfox.
- Aplicación móvil: Este bloque como su nombre lo indica contiene la aplicación desarrollada para dispositivos Android, la cual trae la información a través de *Web Services* desde la base de datos para poderla mostrar a los usuarios por medio de una interfaz gráfica y la API de *Google Maps*.

Implementación del Hardware del Sistema

El hardware del sistema de parqueadero lo constituye los tres primeros bloques descritos en la sección anterior, estos son: Red de sensores, Microcontrolar y TX/RX Sigfox, los cuales se implementan en un solo hardware llamada Estación como se observa en la Figura 18, y está físicamente ubicado junto a las plazas de estacionamiento.

Estación Central

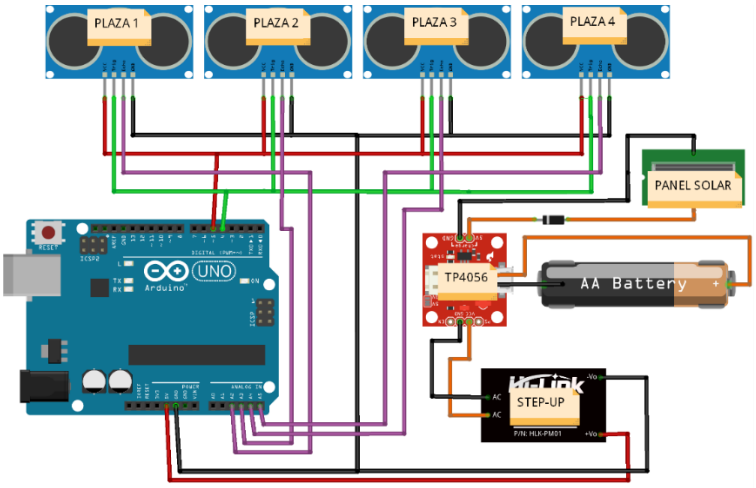
El sistema de parqueadero inteligente cuenta con una estación que contiene al microcontrolador, el cual controla la red de sensores y al transmisor/receptor de Sigfox. La estación cuenta con los siguientes componentes:

- Placa Arduino UNO como microcontrolador.
- Batería 3.7V 4400 mAh.
- Boost convertidor DC-DC Step-Up.
- Módulo TP-4056 Carga de batería.
- Panel Solar.
- Tarjeta TX/RX Sigfox Wisol RC4.
- 4 sensores Ultrasónicos.

La placa Arduino como microcontrolador es el cerebro de la estación, el cual está encargado de controlar los 4 sensores ultrasónicos para que puedan percibir el estado de cada plaza de estacionamiento, estos sensores están conectados directamente al Arduino. Una tarjeta externa se une al Arduino por encima, esta tarjeta contiene al módulo Wisol que es el Tx/Rx Sigfox encargado de establecer la comunicación entre la red Sigfox y nuestro microcontrolador. La batería de 3.7V es la encargada de alimentar energéticamente a la estación, está conectada a un módulo de carga Tp-4056 que tiene como función: en el día cargar a la batería y energizar la Estación por medio de un panel solar de 5V sin causar daños, en la noche energizar la Estación por medio de la batería. Entre la batería y el Arduino se encuentra un convertidor DC-DC Step-Up encargado de proporcionar 5V estables para que el Arduino no sufra daños por caída o picos de voltaje. En la Figura 19 se observa como está conectado la estación.

Figura 19

Conexión física y electrónica de la Estación Central



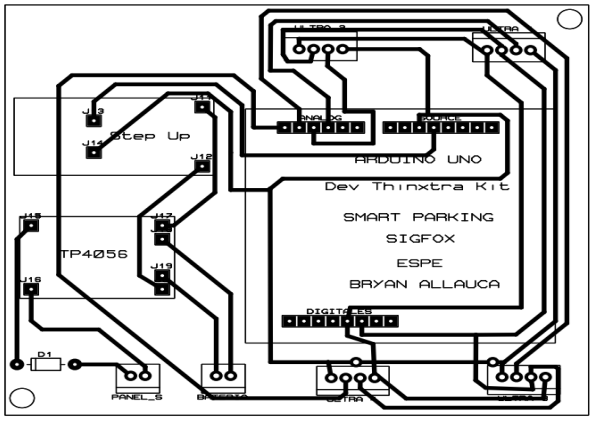
Nota. La figura representa las conexiones de los componentes con el Arduino.

Diagrama de Pistas

Una vez que conocemos como están conectados los componentes se realizó el diseño del diagrama de pistas que se muestra en la Figura 20, para hacer la placa electrónica se debe diseñar con medidas para que puedan soldarse los elementos correctamente y ver las vistas 3D para ver cómo quedaría la placa (Ver Apéndice A).

Figura 20

Diagrama de pistas de la Estación Central



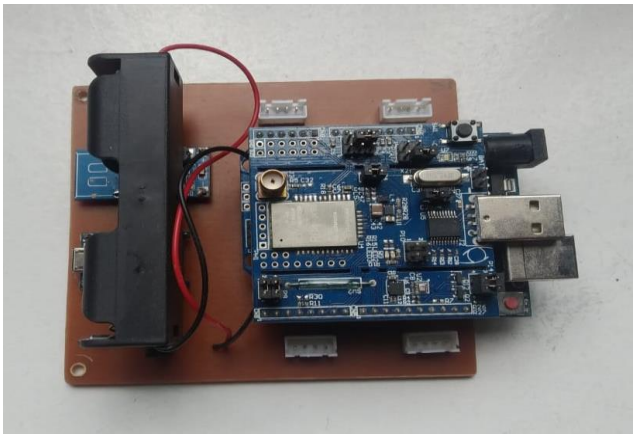
Nota. La figura muestra el diagrama de pistas diseñado para la estación central, se debe tener en cuenta que no está a escala.

Dispositivo Físico Final

En la Figura 21 se muestra la estación una vez terminada de soldar todos los componentes a la placa electrónica, se debe considerar que se puso conectores en lugar de los sensores ya que estos van conectados a cierta distancia de la estación.

Figura 21

Dispositivo Final (Estación Central)



Nota. Estación central con los componentes soldados a la placa electrónica, para el Arduino se usó borneras para poder sacar el Arduino sin desoldar nada.

Implementación del Software del Sistema

El software del sistema de parqueadero está distribuido en cuatro partes, las cuales se relacionan con cuatro de los bloques vistos en la sección 0 y hacen posible la configuración y el control para que el sistema pueda cumplir con lo requerido, los bloques detallados anteriormente son:

- Microcontrolador.
- Red Sigfox.
- Base de Datos.
- Aplicación móvil.

Microcontrolador

El microcontrolador es el cerebro de la estación, como se mencionó antes, este dispositivo es aquel que controla y ejecuta todas las acciones de la estación. Las funciones y sus características que se debe considerar para la programación son:

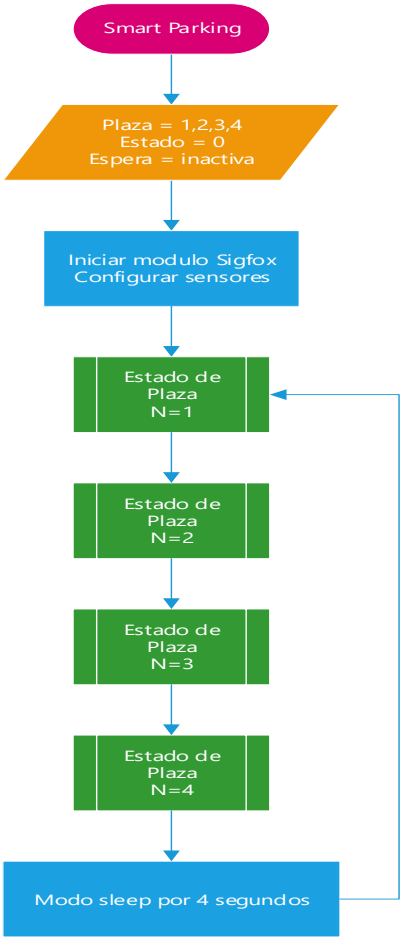
- Controla a red de sensores para que estos perciban el estado de las plazas.
- Los sensores ultrasónicos miden la distancia del objeto para determinar si la plaza está ocupada o libre.
    - Se considera ocupada si la medida está en el rango de 30cm a 100cm.
    - Se considera libre si la medida está en el rango de 0 a 30cm o mayor a 100cm.
- Se guarda la información en formato XY, donde:
    - X es el número de plaza (1, 2, 3, 4).
    - Y es el estado de plaza (1 ocupada, 0 libre).
- La estación está en modo dormido por 4 segundos, es decir que se despierta cada 4 segundos y activa los sensores para la medición, con esto se ahorra energía.
- Si la plaza cambia de estado de libre a ocupado el sensor queda en estado inactivo por 8 segundos.
- La estación envía la información solo cuando existe un cambio de estado, ya sea de libre a ocupada o viceversa, en el resto del tiempo solo registra el estado actual sin enviar la información.
- El formato con el que se envía la información es AB, donde:
    - A es un byte con el número de plaza (1, 2, 3, 4).

- B es un byte con el estado de la plaza (0,1).

La Figura 22 muestra el diagrama de flujo principal del programa con las características y especificaciones detalladas, cada subproceso cuenta con su diagrama de flujo (Ver Apéndice B).

Figura 22

Diagrama de Flujo de la Estación Central



Nota. El gráfico representa el diagrama de flujo principal de la estación central, existen subprocesos que permiten la lectura y toma de decisión por parte del microcontrolador Arduino.

La placa Arduino Uno funciona como el microcontrolador y es en donde se programa las funciones de la Estación, para esto se hace uso de Arduino IDE. Antes de empezar a programar se deben descargar todas las librerías necesarias para que el

microcontrolador pueda comunicarse con los sensores y el transmisor Sigfox, esto se puede hacer por medio del repertorio de Arduino (librerías math y LowPower) o descargando el archivo .zip desde la Internet como es el caso de las librerías WISOL y Wire que sirven para la comunicación con la tarjeta que contiene el TX/RX Sigfox y pueden ser descargadas desde el siguiente link: https://github.com/Thinxtra/Xkit-Sample, proporcionado por el fabricante (Thinxtra, 2019).

Una vez descargadas e incluidas las librerías conforme al diagrama de flujo de la Figura 22 se procede a declarar la variable que contiene el número y el estado de la plaza que será enviado a la red de Sigfox, esta variable para nuestro uso es declarada como un vector de enteros de 8 bits, así:

```
int8_t plaza[2];
```

Después en el siguiente paso se inicia el módulo de Sigfox y los pines que se van a usar para conectar con los sensores, como se muestra a continuación:

```
// Iniciar comunicacion serial entre Arduino y Modulo Sigfox
Wire.begin();
Wire.setClock(100000);
// Prueba con el modulo WISOL
Isigfox->initSigfox();
Isigfox->testComms();
// Configurar pines
pinMode(pinecho1, INPUT);
pinMode(pintrigger, OUTPUT);
pinMode(ali, OUTPUT);
```

Cuando se tiene inicializado y configurado los módulos y los pines, se procede a programar la función principal que tiene bucle infinito, primero activamos los sensores, validamos si la variable espera esta activada para dormir al sensor, caso contrario se lo inicia para ver si la plaza está libre u ocupada, considerando las especificaciones de rango de distancia detalladas anteriormente, seguido de esto verificamos si existe un

cambio de estado para proceder a enviar la información y activar la variable “espera” que deja dormido al sensor 8 segundos, como se indica a continuación:

```
// Activar los sensores
  digitalWrite(ali, HIGH);
  delay(100);
  //Plaza 1
  if(espera1==0){ // Valida variable espera para dormir al sensor
    // Validamos la distancia y actualizamos el estado de la plaza
    if(distancia1<100 & distancia1>30){ //Rango entre 30 y 100 cm
      est1=1; //Ocupada
    }else{
      est1=0; //Libre
    }
  }
// Validamos si existe un cambio en el estado de la plaza para enviar los datos
  if(est1!=est11){
    plaza[0]=1;
    plaza[1]=est1;
    Send_Pload((const char*)&plaza, sizeof(plaza)); // Envia datos
    if (est1==1){
      espera1=1; //Activa variable espera
    }
  }
```

El mismo proceso se realiza para las cuatro plazas de estacionamiento, y al finalizar se envía a dormir al Arduino por 4 segundos con el código:

```
//Dormir Arduino
LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
```

La función Send_Pload es la encargada de realizar la conexión y enviar los datos por la red de Sigfox, una vez terminado el código (Ver Apéndice C), se compila y sube el programa al Arduino.

Base de Datos

La base de datos es la encargada de guardar toda la información de las plazas de parqueo para que la aplicación móvil pueda acceder a estos datos y mostrar al usuario, para esto se necesita que la base de datos esté subida a la Internet y pueda ser

accedida desde cualquier dispositivo. Para cubrir con esta necesidad se usó un hosting gratuito (Webhost, 2020).

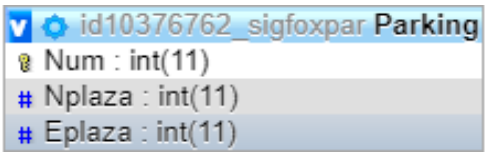
Tablas de Datos

La base de datos cuenta con dos tablas:

1. Parking: esta tabla es la encargada de guardar el estado actual de cada plaza, es decir que se va actualizando al último estado que tiene la plaza, está formado por 3 variables como lo muestra la Figura 23.

Figura 23

Tabla Parking



Nota. Representación de la tabla Parking en la base de datos

   a. Num: número de filas.
   b. Nplaza: número de la plaza.
   c. Eplaza: estado de la plaza.

2. BD_Par: esta tabla es la encargada de guardar el historial de todos los datos que manda la estación, es decir que si un dato llega se guarda en una nueva fila, adicional a esto guarda la fecha y hora que se envió el dato. Está formado por 4 variables como se muestra en la Figura 24.

Figura 24

Tabla BD_Par

Nota. Representación de la tabla DB_Par en la base de datos

a. Num: número de filas.
b. Nplaza: número de la plaza.
c. Eplaza: estado de la plaza.
d. Fecha: fecha y hora de llegada del dato.

Web Services

Una vez que se tiene la base de datos creada con las tablas y sus respectivas variables se procedió a configurar los web services en lenguaje PHP (Ver Apéndice D) que nos sirven para guardar los datos que vienen del backend de Sigfox y para enviar la información a la aplicación móvil cuando se realiza una consulta. Se configuraron dos webs services:

1. Actualizar: es la encargada de conectar el backend de Sigfox con la tabla Parking y la tabla DB_Par de la base de datos. Su función es actualizar el estado de las plazas en la base de datos y guardar el historial de toda la información que llega, para ello primero se debe crear una conexión con la base de datos con la línea de código:

```
$conn = mysqli_connect($servername, $username, $password, $database);
```

Se proceden a leer los valores de la variable que vengan con solicitud POST, aquí se debe verificar el nombre y guardar en una variable nueva así:

```
$numero=$_POST["Nplaza"];
$estado=$_POST["Eplaza"];
$tiem=$_POST["Time"];
```

Seguido de esto se sube a la tabla Parking el estado de la plaza en el número de plaza que corresponde, esto verificando el número de plaza que envió el backend de Sigfox, implementando con la siguiente línea de código:

```
$sql = "UPDATE Parking SET Eplaza='".$estado."' WHERE Nplaza='".$numero."'";
```

Al final se guarda la información en la tabla DB_Par, toda información que llega se guarda en una nueva fila, el número de plaza, el estado de la plaza y la fecha y hora del arribo de los datos, para esto primero se debe colocar en formato de fecha y hora con la respectiva zona horaria y después se guarda en la tabla así:

```
date_default_timezone_set('America/Bogota');
$tiempo=date("Y-m-d H:i:s", $tiem);
$sql1 = "INSERT INTO DB_Par (Fecha, NPlaza, EPlaza) VALUES ('$tiempo', $numero,
$estado)";
```

2. Consultar: es la encargada de conectar la tabla Parking con la aplicación móvil. Su función es permitir que la aplicación móvil acceda a la información del estado de las plazas. De la misma forma que la anterior web services primero se crea una conexión con la base de datos y después se selecciona la fila con el número de plaza que la aplicación pide en la consulta, así:

```
$sql = "SELECT * FROM Parking WHERE Nplaza = '$numero'";
Seguido de esto se crea un array para guardar toda la información:
while($fila=$res -> fetch_array()){
        $pro[]=array_map('utf8_encode',$fila);
}
```

Finalmente, se transforma el array en formato json para que pueda ser leído por la aplicación móvil, para esto se usa la función:

```
echo json_encode($pro);
```

Red de Sigfox

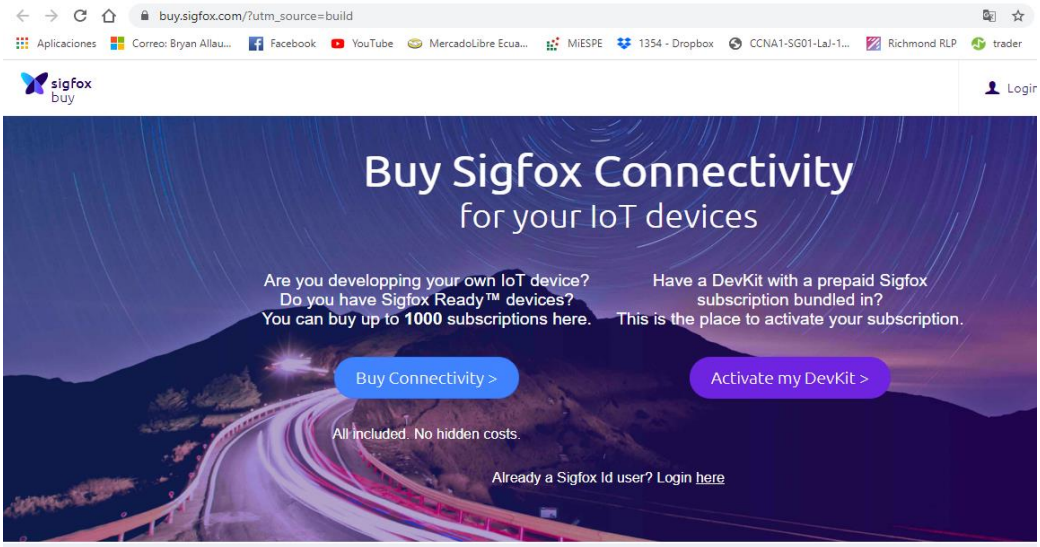
Para que la estación central pueda enviar mensajes a la red de Sigfox primero se debe registrar el comunicador Sigfox a la red, una vez que el equipo esté registrado se procede a configurar los callback hacia la base de datos creada anteriormente.

Registro del dispositivo

Para registrar el dispositivo se debe ingresar a la página oficial de Sigfox (Sigfox, Sigfox Build, 2019) en la opción de *Activate my DevKit,* como lo indica la Figura 25:

Figura 25

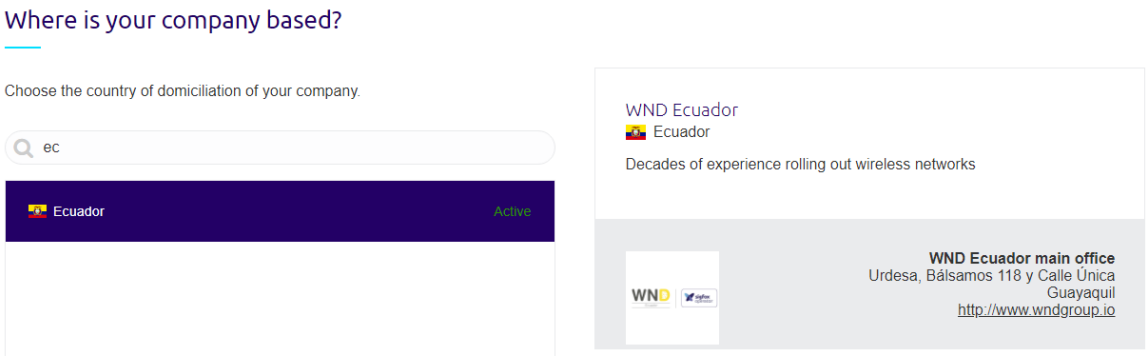
Página para el registro del dispositivo Sigfox



Desde este punto se empieza la activación de nuestro comunicador, primero seleccionamos el país como lo muestra la Figura 26, en nuestro caso Ecuador.

Figura 26

Selección del País para el registro Sigfox

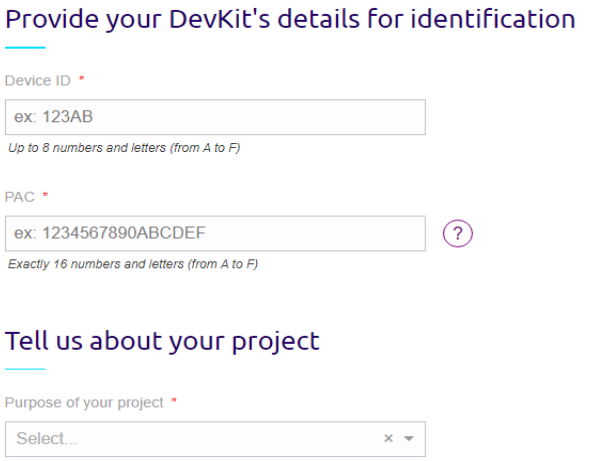


Seguido de esto tenemos que ingresar los datos de nuestro comunicador Sigfox: ID del dispositivo y PAC proporcionado para cada dispositivo y lo podemos encontrar en

la caja del kit. Adicional a esto tenemos que ingresar el propósito del proyecto y una descripción breve del mismo Figura 27.

Figura 27

Datos para el registro Sigfox: ID y PAC

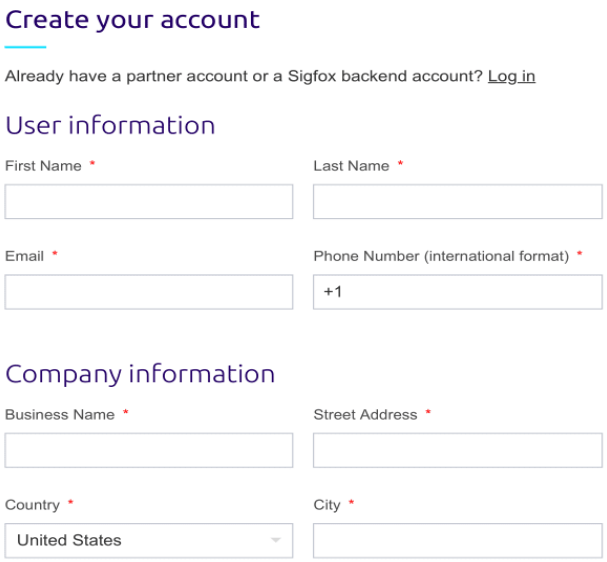


Nota. El ID y el PAC son únicos para cada dispositivo Sigfox y vienen en la caja o se puede sacar por medio de código dependiendo del fabricante.

Después se ingresa la información de la cuenta, parámetros del usuario y a que grupo va a pertenecer el dispositivo, como lo indica la Figura 28.

Figura 28

Datos del usuario que registra el dispositivo Sigfox



Nota. La información de email se usa para el ingreso al backend de Sigfox y el resto de información se usa para identificar al dispositivo.

Finalmente aparece un mensaje de confirmación y al presionar *OK* nos redirige al backend de Sigfox en donde ya se puede ver el dispositivo registrado con todas las opciones y configuraciones disponibles.

Backend Sigfox

Cuando registramos algún dispositivo, podemos ver el estado y algunas otras opciones que nos ofrece Sigfox en el backend, en la parte superior se encuentra un panel con algunas opciones:

- Dispositivo: nos aparece una lista de todos los dispositivos que hemos registrado a la misma cuenta con información de cada uno, Figura 29.
- Tipo de dispositivo: una lista de los tipos de dispositivos que hemos registrado y al grupo que pertenece.
- Usuario: Los usuarios que han registrado algún dispositivo en la cuenta.

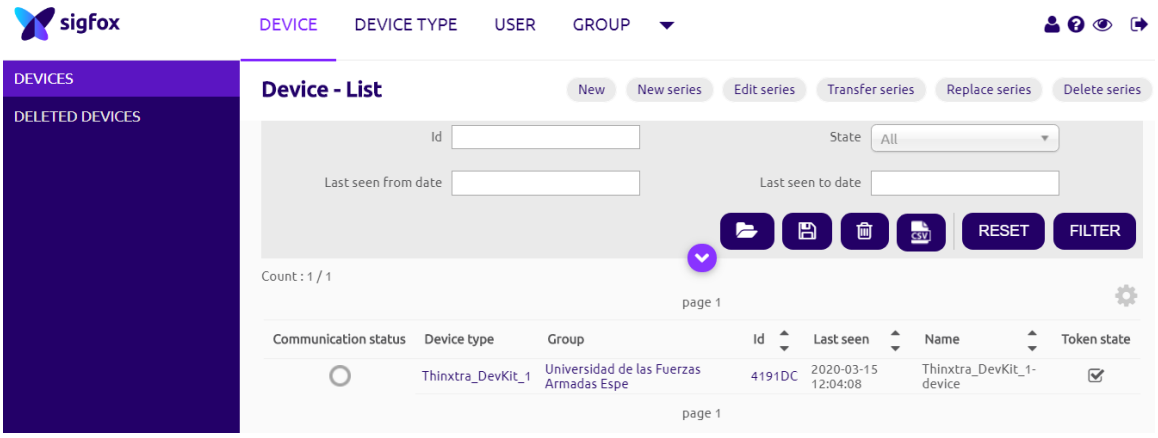
- Grupo: Los grupos que se han creado, aquí se puede tener diferentes grupos para separar proyectos.

En la opción de Dispositivo mostrado en la Figura 29 se puede encontrar información como:

- Estado de comunicación.
- Tipo de dispositivo.
- Grupo al que se registró.
- ID del dispositivo.
- Última vez que se accedió al dispositivo.
- Nombre del dispositivo.
- Estado del toque.

Figura 29

Pantalla principal Backend Sigfox



Nota. Todos los dispositivos registrados aparecen en la lista cada uno con sus características.

Si damos clic en el grupo al que se registró, se abre una nueva página relacionada al grupo Figura 30, en el lado derecho existe un panel donde se pueden escoger algunas opciones para mostrar diferentes datos, entre ellos está:

- Información: aquí se despliega información general como el nombre, una descripción, la localización, el nombre del cliente, datos de fecha y algunos códigos.
- Usuarios asociados: Aquí se despliega información de los usuarios que han registrado los dispositivos.
- Tipos de dispositivos asociados: los tipos de dispositivos que se han registrado, aquí hay varios entre los dispositivos desarrollados para prototipos de diferentes organizaciones, en nuestro caso Thinxtra.
- Contrato asociado: el contrato que se tiene para cada dispositivo, es decir el tipo de suscripción que tiene y la fecha de inicio y de fin del contrato.
- Configuración de eventos: se pueden configurar eventos que se ejecuten cuando llegan nuevos mensajes a los dispositivos, es decir se puede reenviar la información al email o a un servidor externo.
- Acceso a API: se puede asociar a una API externa de algún proveedor IoT o de alguna plataforma compatible con Sigfox.

Figura 30

Backend Sigfox, pantalla del Grupo

Si damos clic en el ID del dispositivo se abre una nueva pantalla, como lo muestra la Figura 31, con otro tipo de información y de igual manera se tiene un panel lateral derecho con opciones de pestaña:

- Información: se despliega la información general del dispositivo como el nombre, el estado de activación, el PAC, el tipo de dispositivo, la ubicación en longitud y latitud, un indicador de calidad de link y los certificados relacionados al modem y al producto de comunicación donde se puede ver la zona en donde trabaja (RCZ4), Figura 31.
- Localización: se despliega un mapa en donde se puede ver la zona en la que está el dispositivo, hay que tomar en cuenta que la precisión llega a ser de kilómetros a la redonda, Figura 32.
- Mensajes: se indica una tabla con todos los mensajes que han llegado, es decir la información que envía el TX/RX Sigfox, están ordenados cronológicamente, se puede observar los datos que llegan, la fecha y hora,

el nivel de señal con la que llega el dato y el estado del callback en caso de que esté configurado.

- Eventos: los eventos son aquellos tipos de error o avisos que envía la red, por ejemplo, datos que no llegaron completos o no se recibió la confirmación, en esta opción se puede ver detalladamente los eventos.
- Estadísticas: aquí se despliega estadísticas con relación a los mensajes que van llegando en forma de gráficas, se tiene las siguientes: número de mensajes por unidad de tiempo, cantidad de bytes por unidad de tiempo, SNR (del inglés *signal to noise ratio*) promedio (dB) por unidad de tiempo y RSSI (del inglés *Received Signal Strength Indicator*) (dBm) por unidad de tiempo, Figura 33.
- Configuración de eventos: se puede configurar devoluciones de llamadas que se ejecuten cuando se presenta un evento, es decir se puede reenviar la información del evento al email o a un servidor externo.

Figura 31

Backend Sigfox, pantalla de ID de Dispositivo

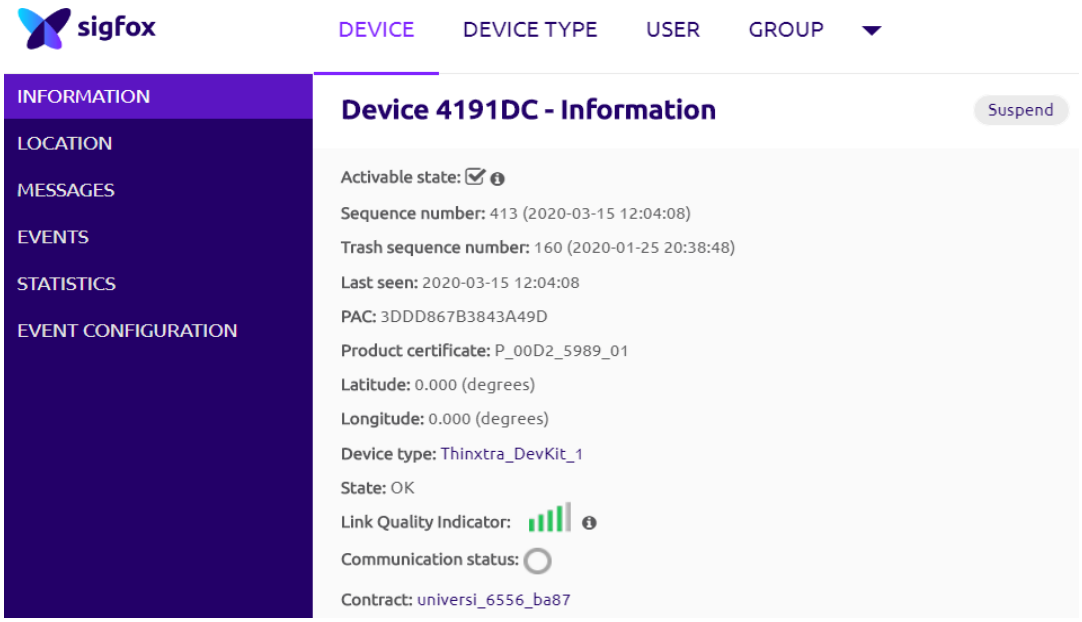


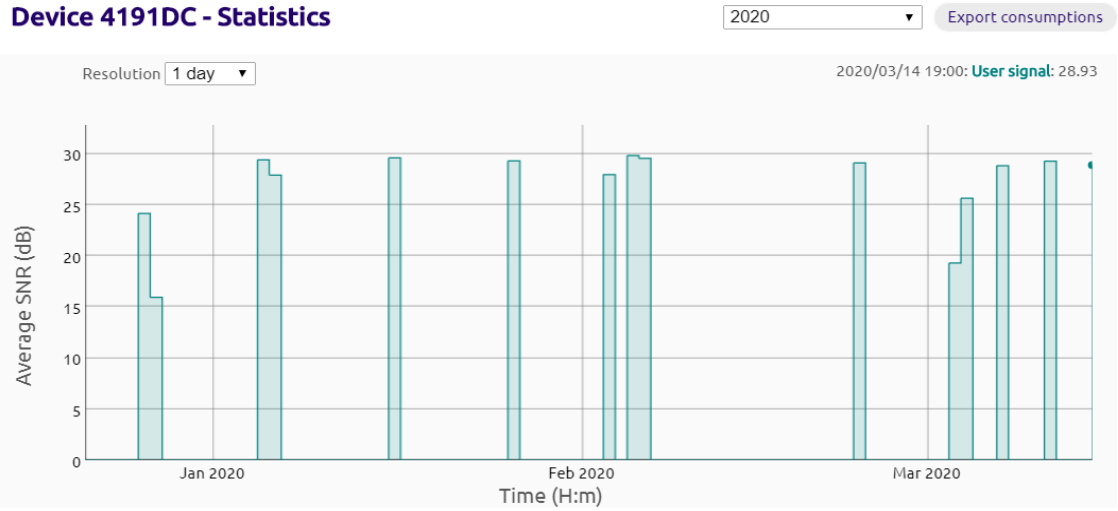
Figura 32

Backend Sigfox, zona de operación del dispositivo

Nota. La función de ubicación que nos proporciona la red nos sirve para ver la ciudad en donde está el dispositivo, aunque se está implementando una nueva herramienta más precisa.

Figura 33

Backend Sigfox, estadísticas del dispositivo



Nota. En la figura se representa la estadística de la SNR de los mensajes, aquí se puede variar la unidad en el eje del tiempo y con el mouse se puede seleccionar la zona a graficar.

Al dar clic en el Tipo de dispositivo se abre otra página que contiene información y configuración relacionada al dispositivo como lo muestra la Figura 34, de igual forma se tiene un panel lateral derecho donde se puede escoger varias opciones:

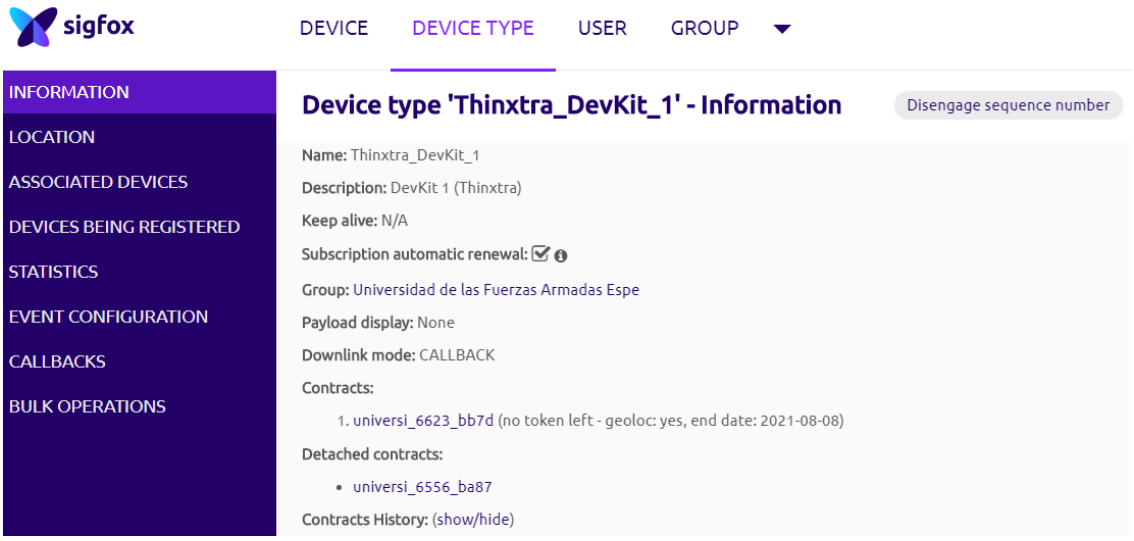
- Información: información general del kit de desarrollo, grupo al que pertenece, descripción, contrato al que está sujeto y las fechas de inicio y fin de suscripción.
- Localización: se despliega en el mapa todos los dispositivos que sean del mismo tipo con la ubicación en longitud y latitud configurada en el dispositivo.
- Dispositivos asociados: se despliega una lista con cierta información de los dispositivos que están asociados.
- Estadísticas: aquí se despliegan las estadísticas en forma de gráficas relacionadas al tipo de dispositivo, entre las cuales tenemos: números de mensajes por unidad de tiempo, número de bytes por unidad de tiempo,

promedio SNR por unidad de tiempo, callbacks con éxito por unidad de tiempo, duración del callbacks por unidad de tiempo, Figura 35.

- Configuración de eventos: aquí se observan detalladamente los eventos que ocurren con los dispositivos del mismo tipo.
- Callbacks: también llamados devoluciones de llamada, aquí se configuran acciones a ejecutar cada que llega un mensaje o dato del dispositivo, es decir cuando llega un nuevo dato este se reenvié a otro lado.

Figura 34

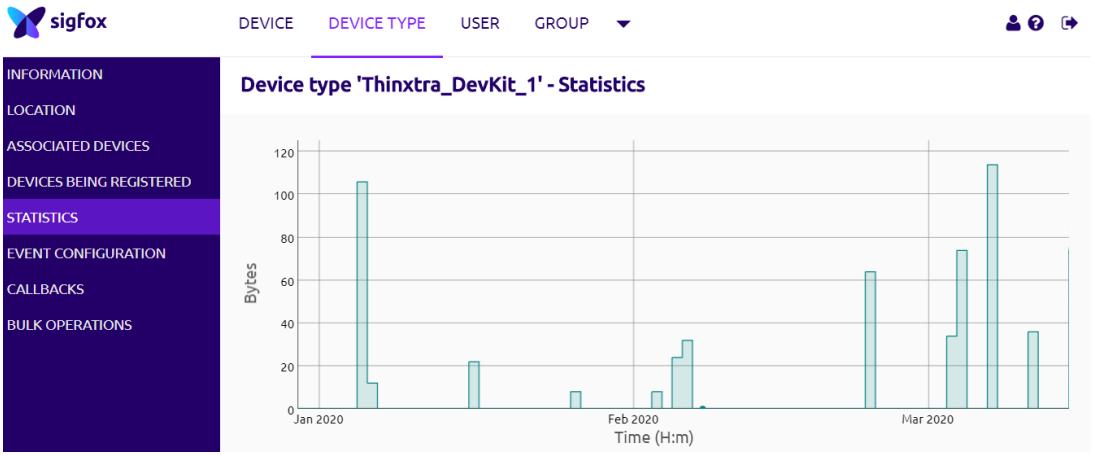
Backend Sigfox, pantalla de Tipo de Dispositivo



Nota. En la figura se representa la información de cada dispositivo donde se puede identificar el producto y su fabricante.

Figura 35

Backend Sigfox, estadísticas de los dispositivos del mismo tipo

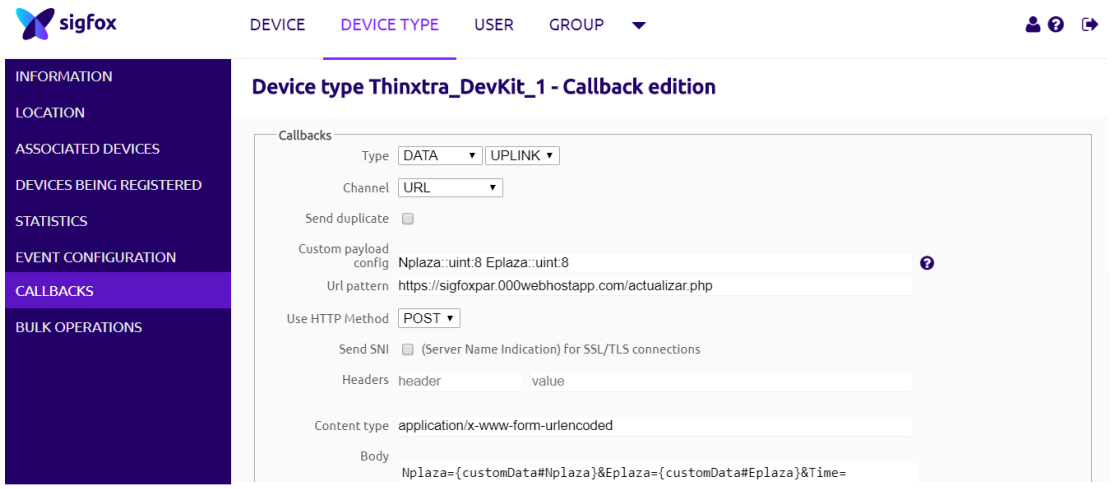


Callbacks

Como se explicó anteriormente, aquí se configuran las devoluciones de llamadas que en palabras simples es el reenvío de los datos a un sitio externos de la red de Sigfox, en este caso nosotros configuramos que los datos se reenvíen a la base de datos creada, esto por medio de web services programadas anteriormente. En la Figura 36, se puede observar la configuración del callback.

Figura 36

Configuración de Callbacks



Nota. Se debe configurar los callback dependiendo a cada proyecto.

La configuración tiene las siguientes características:

- Tipo: es de tipo “DATA” y “UPLINK” porque es el enlace de subida desde el dispositivo.
- Canal: como tenemos un servidor en la Internet por medio de la “URL” podemos ejecutar la web services programada.
- Configuración del payload: como tenemos dos variables se ingresa el nombre de las variables y los bits del mensaje que serán separados para tomar un dato así:

```
Nplaza::uint:8 Eplaza::uint:8
```

- La URL: aquí se ingresa la dirección web de donde está alojada la web services.
- Método HTTP: POST.
- Cuerpo: como cuerpo de la solicitud POST, se pone el nombre de la variable el formato del dato y como extra la variable time, que es la que guarda la fecha y hora del arribo del mensaje así:

```
Nplaza={customData#Nplaza}&Eplaza={customData#Eplaza}&Time={time}
```

Se debe mencionar que las variables disponibles que nos proporciona el backend son las siguientes: device, time, SNR, avgSNR, station, data, rssi, lat, lng y seqNum.

Aplicación Móvil

La aplicación Android fue desarrollada en el IDE de programación Android Studio con lenguaje Java (Ver Apéndice E). Esta aplicación tiene el objetivo de que sea escalable a nivel de soluciones IoT, es decir, que agrupe otras soluciones desarrolladas en un futuro relacionadas a la creación de la *Smart University: Smart ESPE.*

Antes de empezar a programar la interfaz gráfica y las actividades, primero se otorgan los permisos necesarios para que la aplicación acceda a los recursos del

Smartphone como la ubicación y la conexión a Internet, para esto se debe ir al "Manifest" y agregar las siguientes líneas de código:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Smart ESPE

Las características principales de la ventana inicial son:

- Se tiene un menú principal que engloba todas las soluciones relacionadas a Smart ESPE.
- Cuenta con botones dinámicos.
- La aplicación se llama Smart ESPE
- Tiene como fondo el patio central de la Universidad de las Fuerzas Armadas ESPE campus Sangolquí.
- Tiene un botón de información acerca de la aplicación.
- Dentro del menú tiene cuenta con un botón que permite acceder a la aplicación de parqueadero inteligente.
- Simula un botón para acceder a la aplicación de mediciones de variables ambientales.

Primero se necesita crear la interfaz gráfica con todos los botones que se va a usar, esto se puede hacer por medio del diseño gráfico o por medio de código. En este caso se realizó por medio de código xml, primero se necesita crear un constrain que contenga todos los elementos, todo constrain debe tener una altura y un ancho:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:background="#5DABFA"
    tools:context=".MainActivity">
```

Luego se agrega los botones que vamos a usar, dándoles un “id” para usar en la actividad principal, de igual forma necesita una altura, un ancho y un texto para mostrar, un ejemplo de código es el siguiente:

```
<Button
    android:id="@+id/BSPar"
    android:layout_width="80dp"
    android:layout_height="20dp"
    android:background="#00FFFFFF"
    android:text="PARQUEADERO"
    android:textColor="#FFFFFF"
    android:textSize="11sp" />
```

Para agregar la imagen de fondo primero se debe descargar la imagen (pano_espe) en la carpeta “drawable” y después se la ubica en la aplicación con el siguiente código:

```
<ImageView
    android:id="@+id/imageView2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:srcCompat="@drawable/pano_espe" />
```

Una vez que se tiene la interfaz gráfica como se lo muestra en la Figura 37, con todos los botones se procede a programar en el “MainActivity”, el cual es la actividad principal, todas las acciones que va a ejecutar la aplicación en conjunto con la interfaz gráfica.

Figura 37

Pantalla de inicio Smart ESPE

Nota. La aplicación de Variables Ambientales es una ventana adicional no abre ninguna otra ventana, pero está pensado para agregar las diversas soluciones IoT futuras.

Primero declaramos las variables de los botones a usar y después agregamos a la función “onCreate” para asociar los botones con acciones a ejecutar, esto se puede hacer con el siguiente código:

```
Button SPar;
SPar = (Button) findViewById(R.id.BSPar);
```

Una vez hecho esto se programa la acción que va a ejecutar el botón cuanto se presione, en este caso tiene que abrir una nueva ventana que muestre la aplicación parqueadero así:

```
SPar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        startActivity(intent);
    }
});
```

En el caso del botón de información se programó para que despliegue una ventana de aviso, el texto a mostrar se puede escribir directamente o puede ser un string adjuntado como una variable en la carpeta res/values/string como es en este caso:

```
GInf.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AlertDialog.Builder info=new AlertDialog.Builder(MainActivity.this);
        info.setMessage(getString(R.string.Info_App));
        AlertDialog title=info.create();
        title.setTitle("INFORMACION");
        title.show();
    }
});
```

Parqueadero

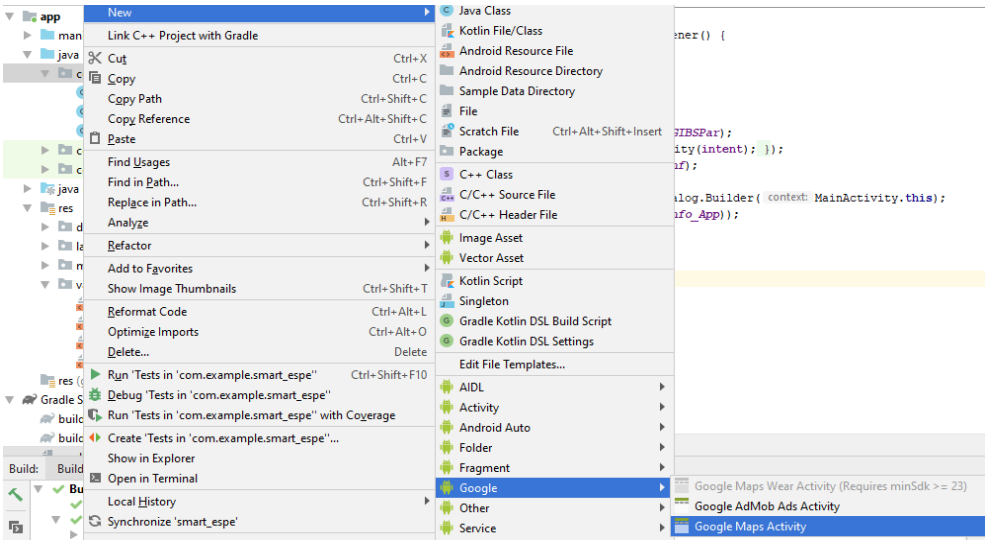
Las características principales de la ventana de la aplicación del sistema de parqueadero inteligente son:

- La ventana cuenta con el mapa de *Google.*
- Tiene la opción de escoger varios tipos de mapas: Híbrido, Terreno, Satelital, Normal.
- Tiene un botón de información general de la aplicación.
- Cuenta con un botón de la ubicación actual del usuario.
- Tienes botones de zoom.
- Dentro del mapa hay marcadores que indican la ubicación de las plazas de parqueo y estas pueden estar de color rojo (ocupado) o verde (libre).
- Tiene un botón para habilitar visualmente todas las plazas (ocupadas y libres) y para habilitar solo las que están libres.

Primero se necesita crear una nueva actividad, hay que tomar en cuenta que como se necesita usar la API de Google Maps, debemos seleccionar una Google Maps Activity como muestra la Figura 38.

Figura 38

Selección API Google Maps



Una vez creada la actividad procedemos primero a crear la interfaz gráfica con todos los elementos que vamos a usar, al igual que el caso anterior se implementó por medio de código xml. Primero creamos un constrain que contenga todos los elementos, dando un ancho y una altura que se ajuste a la pantalla así:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
```

Después de eso creamos los elementos que vamos a usar, para seleccionar el tipo de mapa se usó un spinner y para habilitar todas las plazas o solo las plazas libres se usó un switch. En el switch se creó una función “onclick” que nos servirá para

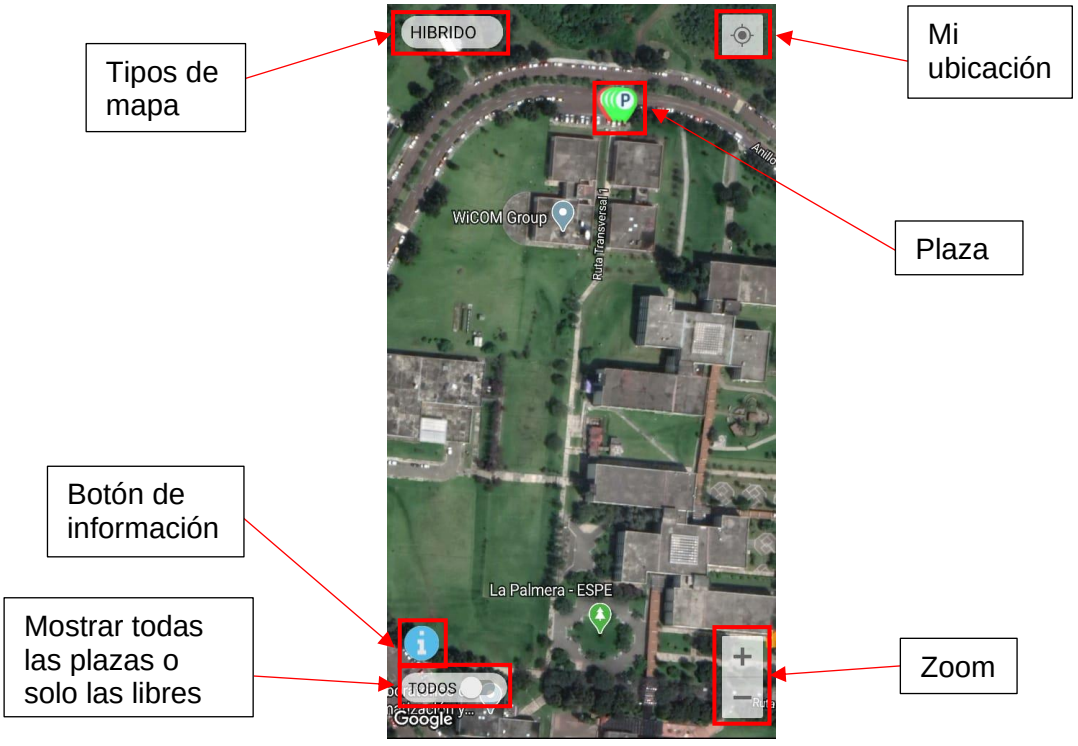
programar la acción que se ejecuta al presionar este elemento, el siguiente código muestra la implementación de estos elementos:

```
<Spinner
    android:id="@+id/Lis_M"
    android:layout_width="wrap_content"
    android:layout_height="30dp"
    android:layout_weight="1"
    android:background="@drawable/button_circle"/>

<Switch
    android:id="@+id/EPla"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:background="@drawable/button_circle"
    android:onClick="onclic"
    android:text="  TODOS"/>
```

Figura 39

Pantalla aplicación Parqueadero



Nota. La función de zoom también puede ser realizada con dos dedos en la pantalla.

Una vez que se tenga creados todos los elementos y la interfaz gráfica diseñada como se muestra en la Figura 39, se debe declarar las variables en el MapsActivity y asociar estas variables a los elementos en la interfaz gráfica de esta manera:

```
private Spinner L_M;
private Switch E_P;
L_M = (Spinner) findViewById(R.id.Lis_M);
E_P = (Switch) findViewById(R.id.EPla);
```

Para el caso del spinner se debe crear un array que contenga los nombres de los tipos de mapas, después este array adaptar al spinner y crear una función para que dependiendo de la opción seleccionada se cambie el tipo de mapa así:

```
ArrayList<String> mapas=new ArrayList<String>();
mapas.add("HIBRIDO");
mapas.add("SATELITAL");
mapas.add("TERRENO");
mapas.add("NORMAL");

L_M.setAdapter(adap); //Adaptar array al spinner

public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
    switch (i){
        case 0:
            mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
            break;
        case 1:
            mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
            break;
        case 2:
            mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
            break;
        case 3:
            mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
            break;
    }
}
```

Para la acción del switch que habilita todas las plazas o habilita solo las plazas libres, primero de debe crear la plaza como un objeto, para esto se crea una nueva

clase llamada “Plaza”. En esta clase se debe programar los atributos y métodos que contiene este objeto, los atributos son:

- Estado de la plaza.
- Número de plaza.
- Marcador: para representar gráficamente en el mapa.
- Ubicación: compuesta por latitud y longitud.

Los métodos son:

1. Crear: este método inicializa el objeto plaza, manda una primera petición a la base de datos, inicializa el marcador con un icono y coloca el título del marcador así:

```
public void crear(){
    actualizar();
    Mplaza.setIcon(BitmapDescriptorFactory.fromResource(R.drawable.pli));
    Mplaza.setTitle("Plaza "+Nplaz);
}
```

2. Buscar: este método es el encargado de conectar la aplicación móvil con la base de datos por medio de la web services programada y JSON, cuando se llama a este método se debe ingresar como atributo la URL de la web services, se puede implementar con el siguiente código:

```
private void buscar(String URL){
    JsonArrayRequest jsonArrayRequest=new JsonArrayRequest(URL, new
Response.Listener<JSONArray>() {
        public void onResponse(JSONArray response) {
            JSONObject jsonObject = null;
            for (int i = 0; i < response.length(); i++) {
                jsonObject = response.getJSONObject(i);
                Eplaz=jsonObject.getString("Eplaza");
            }
        }
    }
}
```

3. Actualizar: este método es el encargado de llamar al método “buscar” indicando la URL de la web services y dependiendo del valor del estado de la plaza, este colocará un marcador de color rojo (ocupado) o verde (libre) así:

```
public void actualizar(){
    buscar("https://sigfoxpar.000webhostapp.com/consulta.php?Nplaza="+Nplaz);
    if(Eplaz.equals("0")){
        Mplaza.setIcon(BitmapDescriptorFactory.fromResource(R.drawable.pli));
    }else{
        Mplaza.setIcon(BitmapDescriptorFactory.fromResource(R.drawable.poc));
    }
}
```

4. Mostrar: este método es usado por el switch, aquí dependiendo del atributo del método los marcadores se harán visibles o no. Por ejemplo, si se selecciona la opción mostrar todos, este método hará visible al marcador independientemente del estado de plaza que tenga, si se selecciona la opción mostrar libres, el método hará visible al marcador solo si el estado de la plaza está en libre así:

```
public void mostrarM(int swt){
    if(swt==0){
        Mplaza.setVisible(true);
    }else{
        if(Eplaz.equals("0")){
            Mplaza.setVisible(true);
        }else{
            Mplaza.setVisible(false);
        }
    }
}
```

Cuando ya se tenga configurada toda la clase con sus atributos y métodos, el siguiente paso es crear los objetos en el MapsActivity, se declara 4 objetos de la clase “Plaza” con el número de plaza y la ubicación en longitud y latitud como atributos así:

```
Plaza plaza1=new Plaza("1",this, -0.312016, -78.44572);
```

Después en el método “onMapReady” se crea el marcador y se inicializa el objeto así:

```
plaza1.Mplaza = googleMap.addMarker(new MarkerOptions().position(plaza1.Pplaza));
plaza1.crear();
```

En el mismo método se habilita el zoom y la ubicación del usuario con el siguiente código:

```
mMap.getUiSettings().setZoomControlsEnabled(true); // habilitar zoom
mMap.setMyLocationEnabled(true); // habilitar localizacion
```

Al final del método se inicia un hilo para que la actualización de los datos de las plazas sea constante, es decir en tiempo real y pueda estar ejecutándose en segundo plano. Se usó un hilo de tipo “Asynctask” para que se ejecute la actualización cada segundo y se implementó con el siguiente código:

```
public class Tiempo extends AsyncTask<Void,Integer,Boolean> {
    @Override
    protected Boolean doInBackground(Void... voids) {
        hilo();
        return true;
    }
    @Override
    protected void onPostExecute(Boolean aBoolean) {
        lectura();
        plaza1.actualizar();
    }
}
public void hilo() {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Finalmente se programa el método “onClick” que fue creado en el switch, dependiendo de la opción que se escoja se va a habilitar todas las plazas (se envía un 0) o solo las plazas libres (se envía un 1), para esto se hace uso del método “mostrar” que fue programado como parte de la clase “Plaza”.

```
public void onclic(View view) {
    if(view.getId()==R.id.EPla)
```

```
        if(E_P.isChecked()){
            E_P.setText("  LIBRES");
            plaza1.mostrarM(1);
            plaza2.mostrarM(1);
            plaza3.mostrarM(1);
            plaza4.mostrarM(1);
        }else{
            E_P.setText("  TODOS");
            plaza1.mostrarM(0);
            plaza2.mostrarM(0);
            plaza3.mostrarM(0);
            plaza4.mostrarM(0);
        }
}
```

Funcionamiento del Sistema

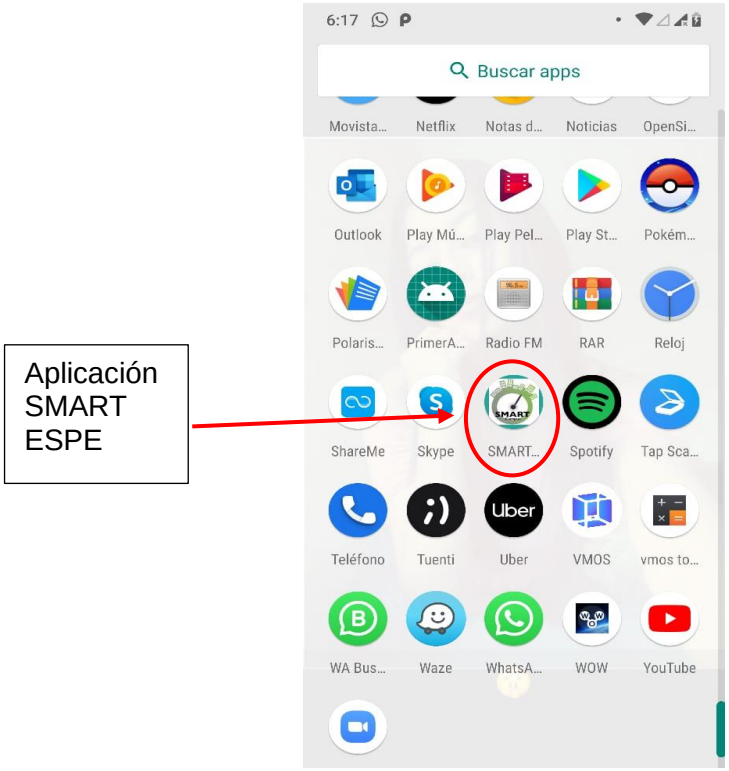
El Sistema de parqueadero inteligente fue desplegado en las plazas de parqueo de los laboratorios del Departamento de Eléctrica, Electrónica y Telecomunicaciones. Para que el usuario pueda acceder al sistema se desarrolló una aplicación móvil en sistemas operativos Android.

Instalación de la Aplicación

El primer paso es descargar la aplicación móvil que se encuentra disponible en el siguiente link: https://drive.google.com/file/d/1anWuuSmP6pUJXXcPpwNvwuoQz9wDU1vn/view?usp=sharing, seguido de esto se debe ir a la carpeta “descargas” dentro del dispositivo e instalar la aplicación con nombre “SMART ESPE”. Previamente se debe tener habilitada la opción para permitir la instalación de aplicación de terceros, es decir aplicaciones que no estén en la Play Store. Una vez instalada la aplicación debe aparecer en el menú principal como lo muestra la Figura 40.

Figura 40

Aplicación instalada en Android



Nota. Dependiendo la versión de Android se debe dar permisos para instalar la aplicación y una vez instalada se debe dar permisos para el acceso a la ubicación del dispositivo.

Prueba de Funcionamiento del Sistema

Al iniciar la aplicación se abre una ventana de menú principal, en ella se debe seleccionar el icono de la solución IoT a la que queremos acceder, en nuestro caso al sistema de parqueadero inteligente, como se muestra en la Figura 41.

Figura 41

Inicio de la aplicación Smart Espe y ventana Parqueadero

Nota. Cuando se ejecuta la aplicación y se escoge la opción de Parqueadero el mapa se centra en el campus universitario y no en la ubicación del usuario.

Cuando abrimos la aplicación se pueden ver las plazas y el estado en las que se encuentran, en la Figura 42 se muestra un ejemplo de la visualización cuando la plaza 2 está ocupada por un automóvil, en la aplicación móvil se observa la plaza 2 con color rojo y los demás con verde, en la Figura 43 se observa el mensaje de arribo al backend de Sigfox con información del mensaje.

Figura 42

Prueba 1 del funcionamiento del sistema con la plaza 2 ocupada

Nota. La grafica muestra el número de plazas configuradas en la aplicación móvil, en la cual la plaza ocupada se pone de color rojo y las otras 3 en verde representando las plazas libres.

Figura 43

Arribo de mensaje del estado de la plaza en el backend de Sigfox

Nota. La figura muestra la hora del mensaje, el delay, el número de secuencia, el mensaje, las estaciones que recibieron el mensaje con el RSSI, SNR y la frecuencia, y el estado del callback.

En la Figura 44 se muestra otro ejemplo de la visualización en la aplicación móvil, en este caso cuando las plazas 2 y 4 son ocupadas por automóviles, de igual manera en la Figura 45 se puede observar los mensajes de arribo en el backend de Sigfox.

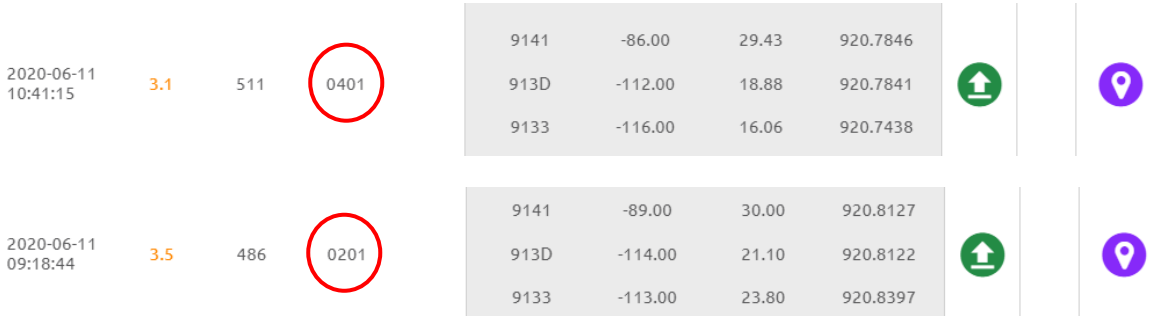
Figura 44

Prueba 2 del funcionamiento del sistema con las plazas 2 y 4 ocupadas

Nota. La grafica muestra el número de plazas configuradas en la aplicación móvil, en la cual las plazas 1 y 3 que están libres se ponen de color verde y las demás en rojo que representa las plazas ocupadas.

Figura 45

Arribo de mensaje del estado de las plazas en el backend de Sigfox



Nota. La figura muestra la hora del mensaje, el delay, el número de secuencia, el mensaje, las estaciones que recibieron el mensaje con el RSSI, SNR y la frecuencia, y el estado del callback.

Pruebas del Sistema

Tras haber desarrollado e implementado el sistema de parqueadero inteligente se realizaron pruebas para demostrar los beneficios que trae el sistema y hacer un análisis de eficiencia de la red de acuerdo con lo establecido en los objetivos. Se realizaron tres pruebas:

1. Retardos del sistema.
2. Consumo energético del sistema.
3. Tiempo en la búsqueda de una plaza de parqueo.

Las dos primeras tienen relación con la eficiencia de la red y del sistema en cuanto a los retardos y al consumo de energía que tiene el sistema, y la tercera prueba tiene relación con los beneficios que trae para el usuario el sistema de parqueadero inteligente.

Retardos del sistema

El sistema de parqueadero inteligente realiza interacciones de datos desde los sensores hasta el usuario, estas interacciones no son instantáneas ya que los datos deben pasar por sistemas de telecomunicaciones los cuales insertan retardos, esto es porque los medios de transmisión tienen recursos limitados y restricciones físicas, además se está usando tecnología inalámbrica que transmite los datos por medio de ondas que se propagan por el aire con una cierta velocidad y tiene un retardo de propagación. Existen varios retardos conocidos en las telecomunicaciones como los retardos de propagación, de transmisión, de procesamiento, etc. Se debe considerar que Sigfox al ser una red licenciada los retardos de propagación, de transmisión y de procesamiento de la red no pueden ser determinados individualmente y sólo se mide el retardo total que exista en la red, por lo que, los retardos que son insertados en el sistema y pueden ser medidos son tres:

- Retardo red Sigfox: desde la estación central a la nube de Sigfox.
- Retardo Callback: desde la nube de Sigfox a la Base de datos.
- Retardo App móvil: desde la base de datos a la aplicación móvil.

Para saber el número de mediciones que se debe tomar para que la información de los retardos propios en el sistema sea fiable se sigue el procedimiento descrito en (Ariza, 2019) como un proceso de estimación de valores según la teoría de medidas, para esto se debe calcular la dispersión de las medidas y analizar el resultado. Los pasos a seguir son:

1. Tomar tres mediciones y calcular la media.

$$X_m = \frac{(X_1 + X_2 + X_3)}{3} \quad (5)$$

2. Calcular el porcentaje de dispersión de las tres mediciones.

$$T = \frac{X_{max} - X_{min}}{X_m} \times 100\% \quad (6)$$

3. En la Tabla 5 seleccionar el mínimo número de mediciones a realizar en base al porcentaje de dispersión calculado.

Tabla 5

Consideraciones para el número de medidas en relación al porcentaje de dispersión

T para N=3	N mínimo a realizar
T<2%	3
2% < T < 8%	6
8% < T < 15%	15
T > 15%	Al menos 50

Nota. La tabla muestra el mínimo número de medidas que se debe realizar para que los resultados obtenidos sean fiables en base al porcentaje de dispersión de tres primeras muestras. Tomado de Ariza, M. J. (2019). *Teoría de medidas.* Departamento de Física Aplicada. CITE II-A

Retardo red Sigfox

Este retardo es el tiempo que toma la información en viajar desde la estación central que está situada en las plazas de estacionamiento en los Laboratorios del

campus hacia la nube de Sigfox, como se mencionó antes, Sigfox al ser red una red licenciada no se puede medir los retardos de propagación, transmisión y procesamiento por separado ya que no se tiene acceso a la infraestructura de la red, por lo que solo se puede medir el retardo total del sistema. Para tomar las medidas se realizó una instalación de prueba en cada zona de parqueo de la Universidad de las Fuerzas Armadas ESPE descrita en la sección 0, los puntos en donde se instaló se pueden observar en la Figura 46 y son:

- Punto C (Lab. Electrónica).
- Punto F (Biblioteca).
- Punto D (MED).
- Punto A (Bar).
- Punto B (Lab. Mecánica).
- Punto E (Inst. de Idiomas).
- Punto G (CICTE).

Figura 46

Puntos de prueba instalados y nodo de Sigfox

Nota. Las pruebas en el punto de prueba G (CICTE) se realizó por ser los parqueaderos más lejanos del campus con relación al nodo Sigfox.

En la Figura 46 también se puede observar la ubicación del nodo Sigfox que se encuentra en la terraza del edificio Central del campus universitario. La Tabla 6 muestra el resultado del proceso descrito anteriormente en la sección 0, que determina cuántas mediciones se deben realizar para que la información sea fiable para cada punto de prueba.

Tabla 6

Datos para determinar el número de mediciones de delay en cada punto de prueba

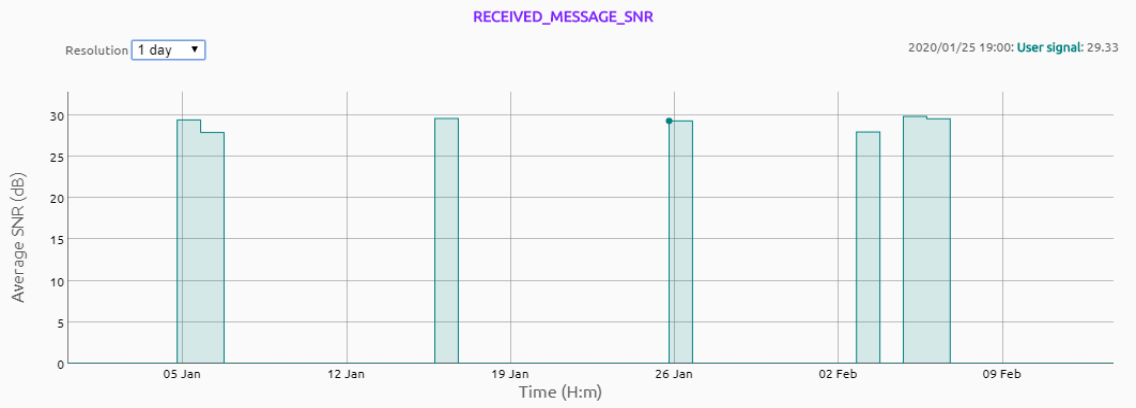
	Punto A	**Punto B**	**Punto C**	**Punto D**	**Punto E**	**Punto F**	**Punto G**
Primera medición (X1)	3172ms	2832ms	2746ms	2986ms	2832ms	2621ms	2986ms
Segunda medición (X2)	2832ms	2621ms	2621ms	2746ms	2546ms	2817ms	3167ms
Tercera medición (X3)	2986ms	2986ms	2746ms	3172ms	2746ms	2746ms	2621ms
Media de las mediciones (Xm)	2997ms	2813ms	2704ms	2868ms	2708ms	2728ms	2963ms
Porcentaje de dispersión (T)	11.3%	12.9%	4.6%	14.3%	10.5%	7.18%	14.5%
Numero de mediciones mínimas	15	15	6	15	15	6	15

Nota. Aunque el número de mediciones determinadas varía entre los puntos de prueba se realizó el mismo número para todos los puntos.

Una vez que se determinó el número de mediciones que se debe tomar para cada punto de prueba se realizaron las medidas del retardo. La información de SNR fue obtenida de la parte de “Estadísticas” del backend de Sigfox como se explica en la sección 0, en la Figura 47 se puede observar estos datos representados en una gráfica.

Figura 47

SNR del mensaje recibido.



Nota. La grafica muestra los resultados de los mensajes por día, pero se puede variar la escala de tiempo para ver el mensaje por horas o minutos.

Con todas las muestras tomadas se calculó el retardo medio y se realizó un diagrama de caja para cada caso, Figura 48. En donde se observan que las muestras para cada punto de prueba tienen un comportamiento parecido, aunque si hay casos atípicos, estos no son considerables. De la misma forma para la SNR, Figura 49, en este caso se puede observar que, a mayor distancia de separación entre los puntos de prueba y el nodo, los valores son más dispersos y se tiene una mayor variación.

Figura 48

Diagrama de caja del retardo de Sigfox para todos los puntos de prueba

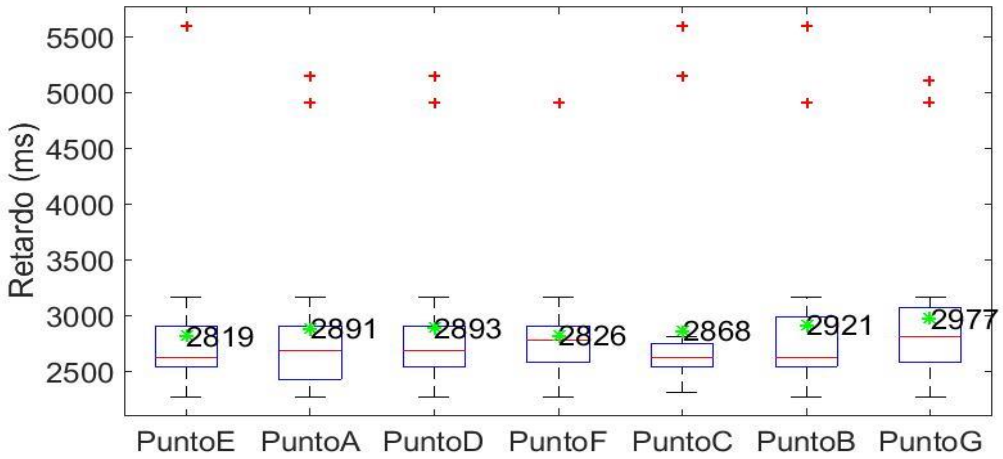
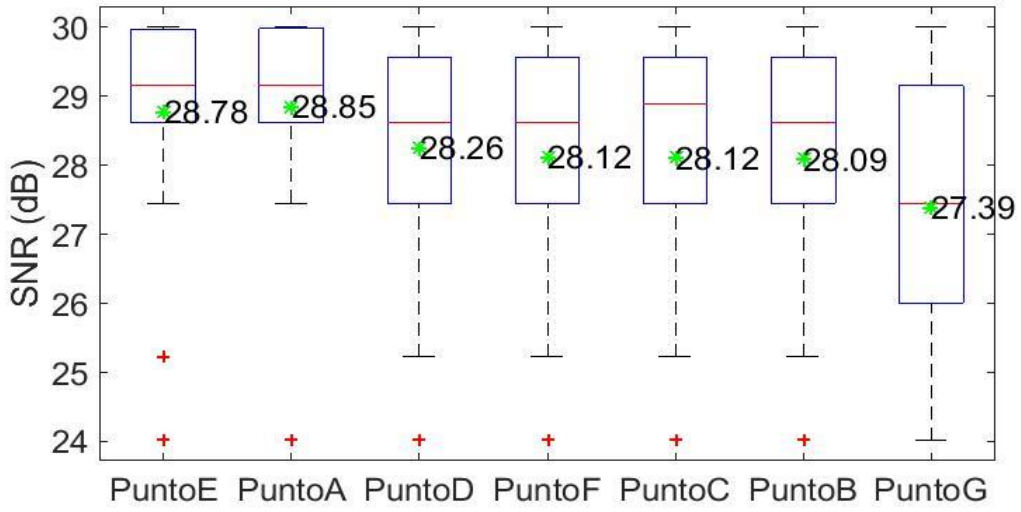


Figura 49

Diagrama de caja de la SNR para todos los puntos de prueba



También se calculó la distancia de cada punto hacia el nodo que está ubicado en el edificio central de la Universidad de las Fuerzas Armadas ESPE. La Tabla 7 muestra los resultados obtenidos de todas las mediciones realizadas, el número de mediciones que se usó para los cálculos es mayor al necesarios para que los valores sean más fiables.

Tabla 7

Retardo medio y SNR de cada punto de prueba

	Punto A	**Punto B**	**Punto C**	**Punto D**	**Punto E**	**Punto F**	**Punto G**
Retardo medio (ms)	2890	2920	2867	2893	2819	2825	2977
SNR promedio (dB)	28.85	28.09	28.11	28.26	28.77	28.12	27.38
Distancia hacia el nodo (m)	172	230	222	209	171	215	567
Numero de mediciones realizadas	20	20	20	20	20	20	20

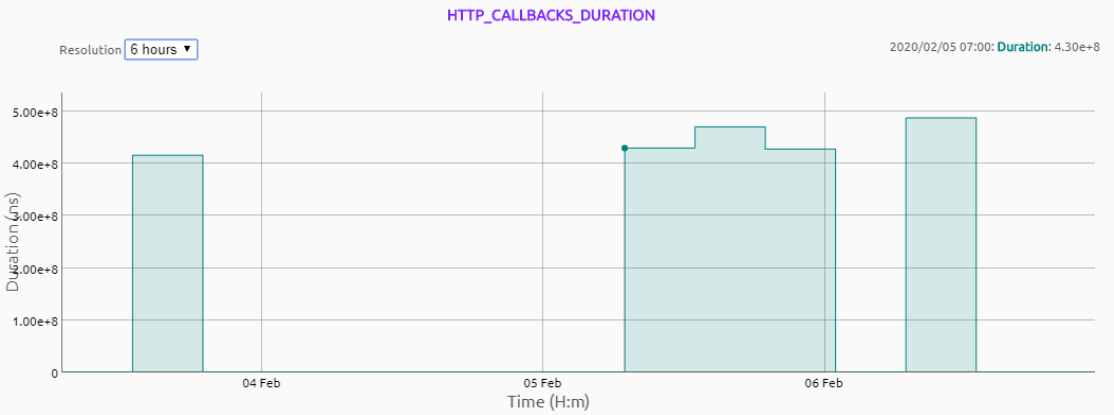
Nota. El número de mediciones realizadas fueron más de las que se calculó y la distancia de los puntos de prueba hacia el nodo se obtuvo de Google Maps.

Retardo Callback

Los retardos Callback es el tiempo que toma los datos en ir desde la nube de Sigfox hacia la base de datos, estos retardos son proporcionados por el backend de Sigfox como parte de un servicio que entrega la red en la parte de estadísticas, en la sección 0 se explica las diferentes ventanas y opciones que tiene el backend de Sigfox, dentro de la parte de estadísticas se puede obtener la información del tiempo que dura la transmisión de un callback, en la Figura 50 se puede observar estos datos representados de forma gráfica.

Figura 50

Duración de Callbacks



Para sacar el retardo medio primero determinamos el número de muestras que necesitamos tomar, siguiendo los pasos descritos anteriormente la Tabla 8 muestra los resultados del proceso.

Tabla 8

Datos para determinar el número de muestras para el retardo de callback

Primera medición (X1)	**416ms**
Segunda medición (X2)	430ms
Tercera medición (X3)	470ms
Media de las mediciones (Xm)	439ms
Porcentaje de dispersión (T)	12.3%
Numero de mediciones mínimas	15

Después de obtener el número de muestras necesarias se procedió a realizar las mediciones del retardo de callback, la Tabla 9 muestra el retardo medio de todas las mediciones tomadas, cabe mencionar que se hizo los cálculos con un número mayor de muestras para que el valor del retardo sea más fiable.

Tabla 9

Retardo medio de callbacks

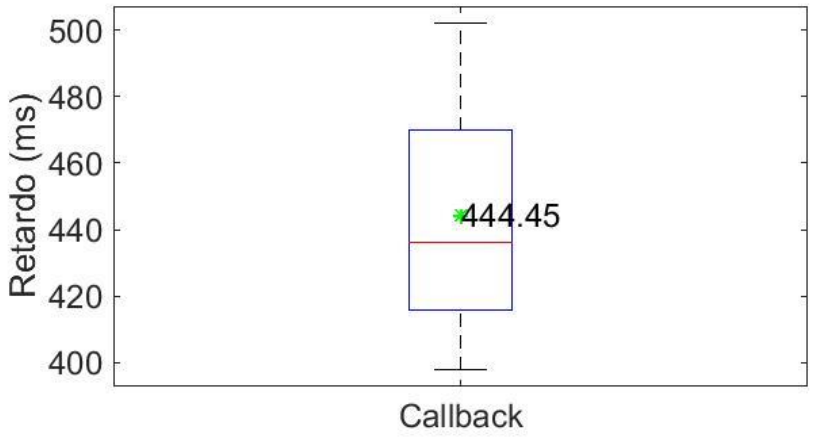
Retardo medio	**444.45ms**
Numero de mediciones realizadas	20

Nota. El retardo medio de callback es el mismo para todos los puntos de prueba porque es el delay del backend y no depende de la distancia del nodo.

En la Figura 51, se muestra el diagrama de caja de las medidas tomadas del retardo de callback, aquí se observa que los retardos no están dispersos y en la mayoría de los casos tienen el mismo valor.

Figura 51

Diagrama de caja del Retardo de Callback



Retardo App móvil

Los retardos de App móvil es el tiempo que toma la información en viajar desde la base de datos hacia la aplicación Smart Espe. En este caso se debe tener en cuenta el tipo de conexión hacia el Internet que tenemos, esto puede ser WiFi o datos móviles, ya que los retardos dependerán de la velocidad de transmisión de conexión de nuestro

smartphone. En el caso de conexión por WiFi hay varios estándares que son usados hoy en día con diferentes velocidades de transmisión, entre las más conocidas esta 802.11b y 802.11n. En el caso conexión por datos móviles hay diferentes generaciones y estándares con sus determinadas velocidades de trasmisión, entre las cuales tenemos 3G (HSPA+) y 4G (LTE).

De la misma forma que en los demás retardos primero se procede a calcular el número de mediciones que se deben realizar, para este retardos se realizó mediciones con conexión a Internet por medio de WiFi y por medio de datos móviles, la Tabla 10 muestra los datos para la determinación del número de mediciones.

Tabla 10

Datos para determinar el número de mediciones para el retardo de App móvil con conexión WiFi y datos móviles

	Conexión WiFi	**Conexión datos móviles 4G**	**Conexión datos móviles 3G**
Primera medición (X1)	18ms	27ms	38ms
Segunda medición (X2)	11ms	25ms	31ms
Tercera medición (X3)	16ms	21ms	35ms
Media de las mediciones (Xm)	15ms	24.32ms	34.67ms
Porcentaje de dispersión (T)	46.67%	24.65%	20.19%
Numero de mediciones mínimas	50	50	50

Una vez determinado el número de muestras necesarias se procede a tomar las medidas, con las mediciones tomadas se calcula el retardo medio que produce esta parte del sistema, los resultados son mostrados en la Tabla 11.

Tabla 11

Retardo medio de App móvil usando conexión WiFi y Datos móviles

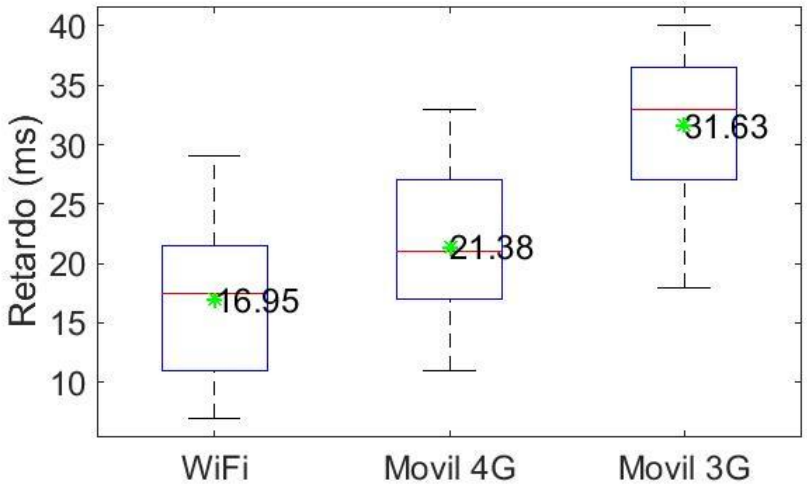
	Conexión WiFi	**Conexión datos móviles 4G**	**Conexión datos móviles 3G**
Retardo medio (ms)	16.95	21.38	31.62

Numero de mediciones realizadas	60	60	60

En la Figura 52, se muestra con diagrama de caja de las medidas tomadas del retardo de App, aquí se observa que aunque existe un incremente de retardo dependiendo de la conexión a la Internet que se tenga, el comportamiento es parecido ya que se tiene retardos que no tienen gran variación.

Figura 52

Diagrama de caja del Retardo de App



Consumo energético del sistema

El consumo de energía es parte de la eficiencia del sistema, como parte de los objetivos planteados es medir cuánta energía consume el sistema de parqueadero inteligente, para esto el hardware del sistema fue separado en partes, desde abajo hacia arriba esta:

- Sensores.
- Microcontrolador (Arduino).
- Tarjeta de comunicación Sigfox.
- Sistema completo (incluido el convertidor Step-up y el TP-4056).

Se realizó la medición de la corriente que consume cada parte por separado y después se midió la corriente en el sistema completo, se realizó en dos escenarios: cuando el sistema está dormido y cuando el sistema está activo, los resultados de las mediciones son mostrados en la Tabla 12. Adicional a esto se midió la corriente que consume la transmisión de datos a la red de Sigfox.

Tabla 12

Consumo de energía del sistema

Componente	Sistema Dormido (mA)	Sistema Activo (mA)
Sensores	0	16
Microcontrolador	20	40
Tarjeta de comunicación Sigfox	30	30
Sistema completo	76	92
Sigfox Tx	153 mA	

Nota. El sistema para ahorrar energía se pone en modo dormido y este ahorro depende del hardware, por lo que se pone mayor interés en la corriente consumida por la transmisión Sigfox.

Tiempo de búsqueda de una plaza de parqueo

El sistema de parqueadero inteligente fue desarrollado con el objetivo de reducir el tiempo de búsqueda de una plaza de estacionamiento, para poder comprobar que el sistema cumple con el objetivo se plantearon dos escenarios diferentes y se midió el tiempo que toma un vehículo llegar hacia la plaza de estacionamiento escogida. Los puntos de prueba son los mismos seleccionados en la sección 0, en donde:

- Punto C (Lab. Electrónica).
- Punto F (Biblioteca).
- Punto D (MED).
- Punto A (Bar).
- Punto B (Lab. Mecánica).
- Punto E (Inst. de Idiomas).

En total son seis las plazas de prueba, como no se tiene desarrollado tantos prototipos físicamente se simuló algunos de ellos en la aplicación móvil, en la Figura 53 se puede observar la aplicación móvil con los iconos de cada plaza en los diferentes puntos de prueba.

Figura 53

Aplicación móvil con plazas en cada punto de prueba

Como se mencionó antes, se propusieron dos escenarios de prueba, esto es porque la Universidad tiene dos entradas hacia el campus: una entrada principal por la Autopista Gral. Rumiñahui y una entrada secundaria por calle Santa Clara. Para las pruebas en ambos escenarios se consideraron algunas características y condiciones descritas a continuación:

- Vehículo con velocidad constante de 20km/h.

- Vehículo con un consumo de 32km/galón.
- El vehículo no se detuvo en ningún momento.
- En cada caso hay una plaza de estacionamiento objetivo.
- En el caso que la plaza objetivo esté ocupada se utilizó la siguiente plaza libre más cercana.
- Se realizó una prueba usando la aplicación y otra sin usar la aplicación.
- No se considera el tiempo de estacionamiento, solo el tiempo que llega a la plaza.

Escenario A

Dentro de este escenario se asume que el vehículo entra por la entrada principal del campus, en la Figura 54 se puede observar el recorrido desde el ingreso hasta la última plaza de estacionamiento más lejana.

Figura 54

Recorrido escenario A

En la Tabla 13 se observa los resultados de los tiempos que se tardó el vehículo en llegar a la plaza cuando no se usó la aplicación y en la Tabla 14 son los resultados cuando se usó la aplicación, en cada tabla sólo se presentan los casos más relevantes, ya que existen 63 posibles combinaciones para los estados de la Plaza que representan 378 casos (Ver Apéndice F). En la parte de Estado de la Plaza un 0 representa que la plaza está libre y un 1 representa que la plaza está ocupada. En la parte de Plaza Objetivo, si es el caso de que se debe seleccionar la siguiente libre más cercana se representa la plaza seleccionada a lado del tiempo.

Tabla 13

Tiempo de recorrido sin App para el escenario A

Estado de la Plaza						**Plaza objetivo (Sin App)**					
PA	PB	PC	PD	PE	PF	PA (s)	PB (s)	PC (s)	PD (s)	PE (s)	PF (s)
0	0	0	0	0	0	92	121	164	241	205	230
0	0	0	0	0	1	92	121	164	241	205	367 PE
0	0	0	0	1	0	92	121	164	241	232 PF	230
0	0	0	0	1	1	92	121	164	241	306 PD	306 PD
0	0	0	1	0	0	92	121	164	266 PC	205	230
0	0	0	1	1	1	92	121	164	266 PC	327 PC	327 PC
0	0	1	0	0	0	92	121	256 PD	241	205	230
0	0	1	1	0	0	92	121	208 PE	306 PE	205	230
0	0	1	1	1	0	92	121	324 PB	331 PF	232 PF	230
0	0	1	1	1	1	92	121	324 PB	470 PB	372 PB	372 PB
0	1	0	0	0	0	92	155 PA	164	241	205	230
0	1	1	1	1	0	92	155 PA	356 PA	331 PF	232 PF	230
0	1	1	1	1	1	92	155 PA	356 PA	504 PA	405 PA	405 PA
1	0	0	0	0	0	121 PB	121	164	241	205	230
1	1	0	0	0	0	164 PC	228 PC	164	241	205	230
1	1	1	0	0	0	241 PD	321 PD	256 PD	241	205	230
1	1	1	1	0	0	205 PE	271 PE	208 PE	306 PE	205	230
1	1	1	1	1	0	230 PF	297 PF	500 PF	331 PF	232 PF	230

Nota. En la tabla se representa los casos sin repetir de los 63 casos posibles, ya que existen casos que presentan el mismo tiempo de recorrido para las diferentes plazas objetivo.

Tabla 14

Tiempo de recorrido con App para el escenario A

Estado de la Plaza						**Plaza objetivo (Con App)**					
PA	PB	PC	PD	PE	PF	PA (s)	PB (s)	PC (s)	PD (s)	PE (s)	PF (s)
0	0	0	0	0	0	92	121	164	241	205	230
0	0	0	0	0	1	92	121	164	241	205	205 PE

Estado de la Plaza						Plaza objetivo (Con App)					
0	0	0	0	1	0	92	121	164	241	230 PF	230
0	0	0	0	1	1	92	121	164	241	241 PD	241 PD
0	0	0	1	0	0	92	121	164	164 PC	205	230
0	0	0	1	1	1	92	121	164	164 PC	164 PC	164 PC
0	0	1	0	0	0	92	121	241 PD	241	205	230
0	0	1	1	0	0	92	121	208 PE	205 PE	205	230
0	0	1	1	1	0	92	121	121 PB	230 PF	230 PF	230
0	0	1	1	1	1	92	121	121 PB	121 PB	121 PB	121 PB
0	1	0	0	0	0	92	92 PA	164	241	205	230
0	1	1	1	1	0	92	92 PA	92 PA	230 PF	230 PF	230
0	1	1	1	1	1	92	92 PA	92 PA	92 PA	92 PA	92 PA
1	0	0	0	0	0	121 PB	164	164	241	205	230
1	1	0	0	0	0	164 PC	164 PC	164	241	205	230
1	1	1	0	0	0	241 PD	241 PD	241 PD	241	205	230
1	1	1	1	0	0	205 PE	205 PE	205 PE	205 PE	205	230
1	1	1	1	1	0	230 PF	230 PF	230 PF	230 PF	230 PF	230

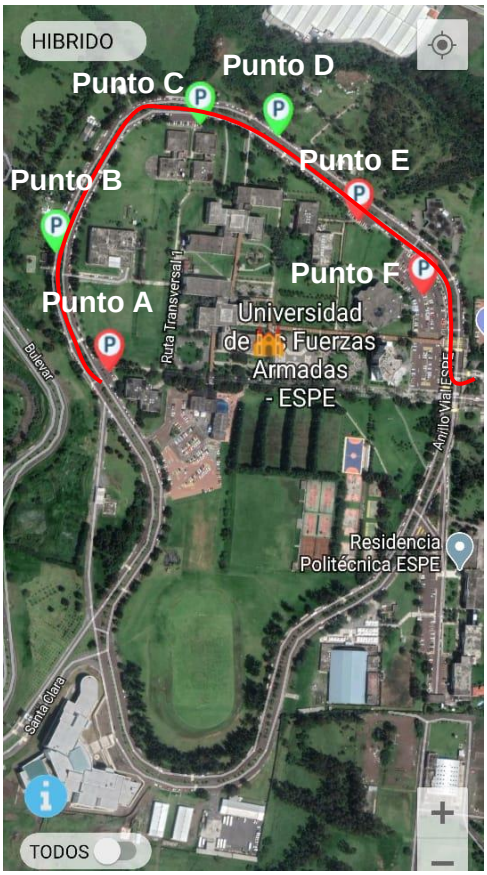
Nota. En la tabla se representa los casos sin repetir de los 63 casos posibles, ya que existen casos que presentan el mismo tiempo de recorrido para las diferentes plazas objetivo.

Escenario B

Para este escenario se asume que el vehículo ingresa por la entrada secundaria o posterior del campus, en la Figura 55 se puede observar el recorrido desde la entrada posterior hasta la última plaza de estacionamiento más lejana.

Figura 55

Recorrido escenario B



Al igual que en el escenario A se tiene dos tablas, la Tabla 15 muestras los resultados cuando no se usó la aplicación y la Tabla 16 muestra los resultados cuando se usó la aplicación. De la misma forma solo se presenta solo los casos más representativos de los 378 casos posibles (Ver Apéndice G). Se tiene las mismas consideraciones de las tablas del escenario B.

Tabla 15

Tiempo de recorrido sin App para el escenario B

Estado de la Plaza						**Plaza objetivo (Sin App)**					
PA	PB	PC	PD	PE	PF	PA	PB	PC	PD	PE	PF
0	0	0	0	0	0	173	141	94	74	135	53
0	0	0	0	0	1	173	141	94	74	135	191 PE
0	0	0	0	1	0	173	141	94	74	162 PF	53

Estado de la Plaza						Plaza objetivo (Sin App)					
0	0	0	0	1	1	173	141	94	74	235 PD	131 PD
0	0	0	1	0	0	173	141	94	97 PC	135	53
0	0	0	1	1	1	173	141	94	97 PC	257 PC	151 PC
0	0	1	0	0	0	173	141	256 PD	74	135	53
0	0	1	1	0	0	173	141	188 PE	137 PE	135	53
0	0	1	1	1	0	173	141	256 PB	163 PF	162 PF	53
0	0	1	1	1	1	173	141	PB	304 PB	315 PB	196 PB
0	1	0	0	0	0	173	175 PA	94	74	135	53
0	1	1	1	1	0	173	175 PA	286 PA	163 PF	162 PF	53
0	1	1	1	1	1	173	175 PA	286 PA	337 PA	335 PA	226 PA
1	0	0	0	0	0	207 PB	141	94	74	135	53
1	1	0	0	0	0	247 PC	247 PC	94	74	135	53
1	1	1	0	0	0	339 PD	339 PD	256 PD	74	135	53
1	1	1	1	0	0	289 PE	289 PE	428 PE	137 PE	135	53
1	1	1	1	1	0	315 PF	315 PF	428 PF	163 PF	162 PF	53

Nota. En la tabla se representa los casos sin repetir de los 63 casos posibles, ya que existen casos que presentan el mismo tiempo de recorrido para las diferentes plazas objetivo.

Tabla 16

Tiempo de recorrido con App para el escenario B

Estado de la Plaza						Plaza objetivo (Con App)					
PA	PB	PC	PD	PE	PF	PA	PB	PC	PD	PE	PF
0	0	0	0	0	0	173	141	94	74	135	53
0	0	0	0	0	1	173	141	94	74	135	135 PE
0	0	0	0	1	0	173	141	94	74	53 PF	53
0	0	0	0	1	1	173	141	94	74	74 PD	74 PD
0	0	0	1	0	0	173	141	94	94 PC	135	53
0	0	0	1	1	1	173	141	94	94 PC	94 PC	94 PC
0	0	1	0	0	0	173	141	74 PD	74	135	53
0	0	1	1	0	0	173	141	135 PE	135 PE	135	53

Estado de la Plaza						Plaza objetivo (Con App)					
0	0	1	1	1	0	173	141	141 PB	53 PF	53 PF	53
0	0	1	1	1	1	173	141	141 PB	141 PB	141 PB	141 PB
0	1	0	0	0	0	173	173 PA	94	74	135	53
0	1	1	1	1	0	173	173 PA	173 PA	53 PF	53 PF	53
0	1	1	1	1	1	173	173 PA	173 PA	173 PA	173 PA	173 PA
1	0	0	0	0	0	141 PB	141	94	74	135	53
1	1	0	0	0	0	94 PC	94 PC	94	74	135	53
1	1	1	0	0	0	74 PD	74 PD	74 PD	74	135	53
1	1	1	1	0	0	135 PE	135 PE	135 PE	135 PE	135	53
1	1	1	1	1	0	53 PF	53 PF	53 PF	53 PF	53 PF	53

Nota. En la tabla se representa los casos sin repetir de los 63 casos posibles, ya que existen casos que presentan el mismo tiempo de recorrido para las diferentes plazas objetivo.

CAPÍTULO 5

3. ANÁLISIS DE RESULTADOS

Retardos del Sistema

En la sección 0 se tabulan los retardos del sistema por partes para los seis puntos de prueba, para identificar el retardo total del sistema se tienen que sumar los retardos de cada parte. Para el retardo de App móvil se usó el retardo que se genera cuando se tiene una conexión WiFi, ya se dispone de esta red en todo el campus. En la Tabla 17 se puede observar los resultados, junto con la distancia y la SNR.

Tabla 17

Retardos del sistema, SNR y distancia de cada punto de prueba

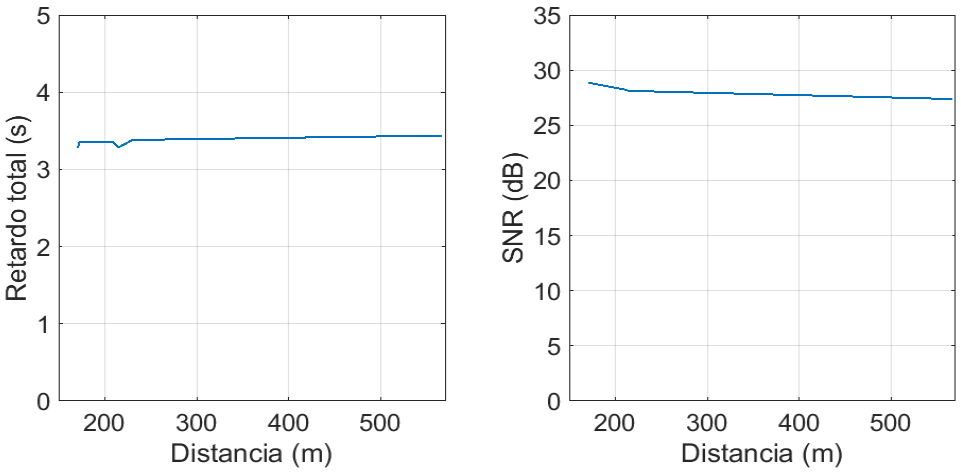
Retardo	Punto A	Punto B	Punto C	Punto D	Punto E	Punto F	Punto G
Red Sigfox (ms)	2890	2920	2867	2893	2819	2825	2977
Callback (ms)	444.45	444.45	444.45	444.45	444.45	444.45	444.45
App móvil (ms)	16.95	16.95	16.95	16.95	16.95	16.95	16.95
Total (ms)	3351.4	3381.4	3328.4	3354.4	3280.4	3286.4	3438.4
SNR (dB)	28.85	28.09	28.11	28.26	28.77	28.12	27.38
Distancia (m)	172	230	222	209	171	215	567

Nota. La tabla representa la recopilación de los datos obtenidos en las pruebas para poder obtener un retardo total con su respectiva SNR y distancia al nodo.

En la Figura 56 se muestran los resultados de la Tabla 17 en cuanto a los retardos y la SNR. Se puede observar que no hay una diferencia notable tanto en los retardos como en la SNR a medida que crece la distancia, aunque si existe un aumento del retardo y una disminución en la SNR. En (Márquez, 2015) se realizó un estudio similar pero con tecnología ZigBee, los resultados de los retardos, la distancia y la SNR que se obtuvo para cinco puntos de prueba se muestran en la Tabla 18.

Figura 56

Retardo y SNR en función de la distancia con Sigfox



Nota. La grafica representa los resultados sobre la red Sigfox, se puede observar un pico pequeño del delay en los 220m que es un poco atípico, pero no es considerable.

Tabla 18

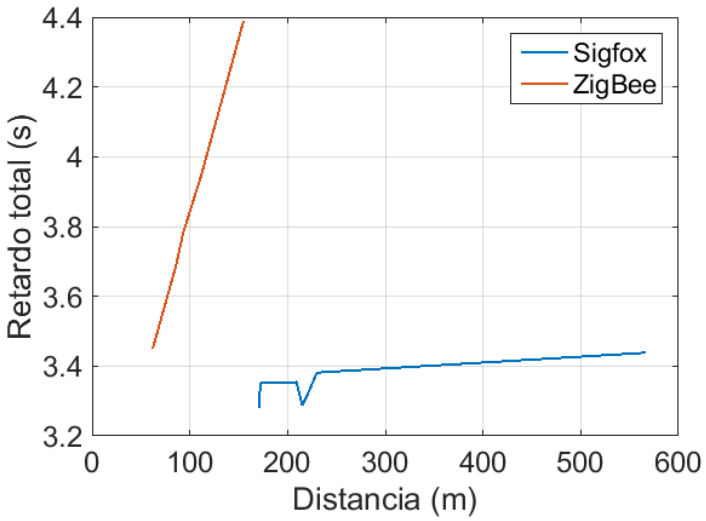
Retardo, SNR y distancia en red ZigBee

	Prueba 1	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**
Retardo (s)	3.69	3.95	4.39	3.78	3.45
SNR (dB)	38	13	4	13	28
Distancia (m)	86	112	155	93	62

Nota. La tabla muestra los resultados de pruebas hechas con tecnología ZigBee. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí.

Figura 57

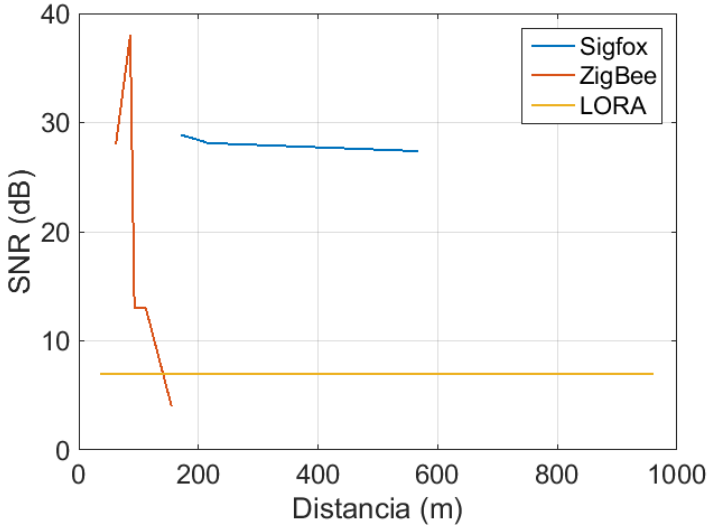
Retardos vs distancia en la red Sigfox y ZigBee



En la Figura 57 se muestran los resultados de los retardos cuando se usa la red Sigfox y ZigBee, en las grafica en forma de comparativa se puede observar que los retardos en ZigBee crecen significativamente a medida que crece la distancia hacia el nodo, mientras que en Sigfox los retardos no crecen tan significativamente a pesar de que la distancia es mucho mayor.

Figura 58

SRN vs distancia en la red Sigfox, ZigBee y LORA



En la Figura 58 se muestran los resultados en cuanto a la SNR cuando se usa la red Sigfox, ZigBee y LORA, la información de la red LORA fue sacada de un estudio relacionado en (Chiriboga, 2020). En ZigBee se puede notar una caída significativa cuando llega a los 150 metros de distancia, en LORA la SNR es más baja que ZigBee cuando está a menos de 150 de distancia, pero tiene una característica importante que se mantiene constante en 7dB, aunque la distancia incremente significativamente. Y finalmente en Sigfox a la misma distancia de LORA y ZigBee se tiene una SNR mucho mejor, y a medida que crece la distancia esta no cae bruscamente y aunque no es constante como en el caso de LORA se mantiene a más de 25 dB.

Consumo de Energía

En la sección 0 se presentó el consumo de energía del sistema separado por partes y cuando el sistema está dormido y activo. Se puede apreciar que cuando el sistema está dormido tiene un ahorro de energía de 36mA ya que los sensores están apagados y el microcontrolador está en modo *sleep*, pero la tarjeta de conexión de Sigfox al estar conectada directamente a la alimentación no puede ser puesta en modo sleep por lo que tiene un desperdicio de energía.

Recordando que el kit de desarrollo thinXtra es un dispositivo de prueba para desarrollar prototipos, este no está diseñado para ser eficiente energéticamente, ya que tiene sensores y características que no son usadas para el sistema de parqueadero inteligente. Adicional a esto se está usando un Arduino Uno como microcontrolador el cual consta de elementos que consumen mucha energía y convertidores DC-DC que desperdician energía.

Por estas razones se separó toda la parte de hardware y se procede a analizar sola el consumo que se tiene al enviar los mensajes por la red Sigfox, es decir la corriente consumida en el transmisor de Sigfox, en la Tabla 12 se ve la corriente que se

consume al enviar un mensaje y tomando en cuenta el intervalo de tiempo que toma enviar un mensaje se procede a calcular el consumo de corriente al transmitir mensajes por hora. Para calcular el consumo por hora se usa la siguiente ecuación:

$$C = \frac{N_m \times T_m}{T_h} \times C_{tm} \qquad (7)$$

En donde:

C= Corriente de consumo por hora.

Nm= Número de mensajes por hora.

Tm= Tiempo que se demora en transmitir el mensaje en segundos.

Th= Unidad de hora en segundos.

Ctm= Corriente de consumo por mensaje.

Recordando que el número máximo de mensajes que permite a red Sigfox son 140 al día, se tiene que en una hora se puede enviar 5.83 mensajes. En la sección 0 se presentó la información acerca del tiempo que demora enviar un mensaje, se tiene que la media es de 2.89s y teniendo en cuenta que una hora tiene 3600 segundos se obtuvo los resultados presentados en la Tabla 19.

Tabla 19

Consumo de corriente en el transmisor en la red Sigfox

	Consumo por mensaje	**Consumo por hora**
Transmisor Sigfox	153 mA	2.45 mAh

Nota. Se calcula el consumo de energía por hora para determinar la duración de la batería.

Estos resultados se pueden comparar con otras tecnologías inalámbricas que son usadas actualmente, entre ellas la más conocida es WiFi que se usa para diferentes propósitos y también esta ZigBee que fue comparado en la sección anterior por ser muy usada en cuando a soluciones IoT. En (Jaya, Vizcaíno, & Acosta, 2014) se hizo un estudio relacionado a la tecnología 802.11 o WiFi, aquí se presenta información acerca del consumo de corriente, al igual que en (Márquez, 2015), que presenta datos de

consumo de corriente pero para ZigBee. En la Tabla 20 se presenta información de consumo de cada tecnología.

Tabla 20

Consumo de corriente en el transmisor y receptor para ZigBee y WiFi

	ZigBee	**WiFi**
Corriente consumida en el transmisor (mA)	285	274-318
Corriente consumida en el receptor (mA)	45	180-196

Nota. Tomado de Márquez, R. M. (2015). Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos. Universidad de las Fuerzas Armadas Espe, Sangolquí y Jaya, M., Vizcaíno, I., & Acosta, F. (2014). MONITORIZACIÓN DE CONSUMO ENERGÉTICO EN REDES INALÁMBRICAS WI-FI CON DIFERENTES ESTÁNDARES DE COMUNICACIONES Y VOLÚMENES DE TRÁFICO. Sangolquí.

El sistema de parqueadero inteligente solo necesita transmitir mensajes y la corriente consumida que se determino es solo en el transmisor por lo que se comparó con las otras tecnologías solo en la transmisión de mensajes. En la Tabla 21 se presenta la diferencia de consumo que tiene WiFi y ZigBee en comparación con Sigfox.

Tabla 21

Diferencia de consumo de corriente entre Sigfox con ZigBee y WiFi

	ZigBee	**WiFi**
Diferencia de corriente consumida en el transmisor (mA)	105	121-165
Porcentaje de Ahorro	36.8%	44.1%-51.8%

Como se puede observar hay un gran ahorro de energía con Sigfox en comparación a otras tecnologías, es por esto que las soluciones con esta red tienen como característica el ahorro de energía y la capacidad de larga duración en la batería.

Para el sistema de parqueadero inteligente se está usando una batería recargable de 4400mAh, sabiendo el consumo de corriente del sistema se puede calcular cuánto dura la batería, con la siguiente ecuación.

$$D_b = \frac{C}{C_c} \quad (8)$$

En donde:

Db= Duración de la batería en horas.

C= Capacidad de la batería en mAh.

Cc= Consumo de corriente del sistema en mA.

Realizando los cálculos se llega a determinar que la batería debería durar 45 horas, pero en la práctica duró alrededor de 30 horas, esto es porque la capacidad de la batería no es la ideal y con el uso este valor llega a disminuir.

Tiempo de búsqueda de una plaza de estacionamiento

En la sección 0 se realizó las pruebas y se tabuló la información del tiempo que toma llegar desde la entrada hasta una plaza de parqueo libre usando el sistema desarrollado y cuando no se usa en los dos escenarios. Se encontró que hay varios casos en donde existe una reducción de tiempo al usar la aplicación desarrollada, en la Tabla 22 se presenta el número de casos y el porcentaje que representa para el escenario A y para el escenario B.

Tabla 22

Casos y porcentaje con y sin reducción de tiempo para el escenario A y B

	Escenario A		**Escenario B**	
	Número	Porcentaje	Número	Porcentaje
Casos con reducción de tiempo	131	34.65%	146	38.62%
Casos sin reducción de tiempo	247	65.35%	232	61.38%
Total	378	100%	378	100%

Según los resultados de la Tabla 22 se puedo observar que 7 de cada 20 casos para el escenario A y 4 de cada 10 casos en el escenario B, entran en las posibilidades de que exista un aumento en el tiempo de búsqueda de una plaza de estacionamiento

libre al no usar la aplicación desarrollada. En la Tabla 23 se presenta la reducción del tiempo separada en 5 intervalos con el número de casos tanto para el escenario A como para el escenario B y el porcentaje que representa.

Tabla 23

Reducción de tiempo al usar la aplicación desarrollada para el escenario A y B

Reducción de tiempo (s)	**Escenario A**		**Escenario B**	
	Número	Porcentaje	Número	Porcentaje
0 (sin reducción)	247	65.35%	232	61.38%
Menor a 100	63	16.67%	55	14.55%
De 100 a 200	52	13.75%	80	21.16%
De 200 a 300	11	2.91%	10	2.65%
Mayor a 300	5	1.32%	1	0.26%

Nota. Al haber gran número de casos con diferente tiempo de reducción es mejor separarlos por intervalos para poder analizar de mejor manera.

Consumo de Combustible

El consumo de combustible tiene relación con la diferencia de distancia que recorre el automóvil, al igual que en tiempo de búsqueda se separó en 5 rangos de distancia con el número de casos para cada uno y el porcentaje que representa, los resultados se representan en la Tabla 24.

Tabla 24

Diferencia de distancia recorrida al usar la aplicación desarrollada para el escenario A y B

Diferencia de distancia recorrida (m)	**Escenario A**		**Escenario B**	
	Número	Porcentaje	Número	Porcentaje
0	247	65.35%	232	61.38%
Menor de 555	63	16.67%	55	14.55%
De 555 a 1110	52	13.75%	80	21.16%
De 1110 a 1665	11	2.91%	10	2.65%
Mayor de 1665	5	1.32%	1	0.26%

Una vez que se determinó la diferencia de distancia que el automóvil debe recorrer, se puede estimar la cantidad de combustible que se ahorra en cada caso, para el cálculo se debe tomar en cuenta una de las características especificadas en la

sección 0, en donde se establece el consumo del vehículo de prueba en 32km/gal. Esta reducción de consumo de combustible se traduce en ahorro económico para el usuario, en la Tabla 25 se presenta el ahorro que se tiene al usar la aplicación desarrollada dependiendo de la distancia que se recorra en cada caso y considerando que el precio en el Ecuador de la gasolina extra o ecopaís se encuentra en 1.84 $/gal y de la gasolina súper en 2.10 $/gal (Pacheco, 2020).

Tabla 25

Ahorro diario de un automóvil para el escenario A y B

Distancia (m)	**Escenario A**	**Escenario B**	**Ahorro de combustible (gal)**	**Ahorro económico (S)**	
				Extra	Súper
Menor de 555	16.67%	14.55%	0.017	0.032	0.036
De 555 a 1110	13.75%	21.16%	0.034	0.064	0.072
De 1110 a 1665	2.91%	2.65%	0.052	0.096	0.109
Mayor de 1665	1.32%	0.26%	0.069	0.128	0.145

Nota. Los precios de la gasolina en Ecuador están sujetos a cambios constantes y dependen de diversos factores por lo que el ahorro económico puede variar.

Como se puede observar el ahorro no es tan representativo ya que solo se está considerando el de un día, ahora si el automóvil realizase el mismo recorrido por un año el consumo aumenta y el ahorro ya se vuelve más significativo. En la Tabla 26 se presentan estos valores anuales, tomando solo 5 días a la semana como días laborales en donde los usuarios de la comunidad Universitaria harían uso del sistema de parqueadero inteligente, en este caso serían 260 días.

Tabla 26

Ahorro anual de un automóvil para el escenario A y B

Distancia (m)	**Escenario A**	**Escenario B**	**Ahorro Anual de combustible (gal)**	**Ahorro económico Anual (S)**	
				Extra	**Súper**
Menor de 555	16.67%	14.55%	4.513	8.350	9.479
De 555 a 1110	13.75%	21.16%	9.027	16.701	18.958

Distancia (m)	Escenario A	Escenario B	Ahorro Anual de combustible (gal)	Ahorro económico Anual (S)	
				Extra	Súper
De 1110 a 1665	2.91%	2.65%	13.54	25.052	28.437
Mayor de 1665	1.32%	0.26%	18.05	33.402	37.916

Nota. Los precios de la gasolina en Ecuador están sujetos a cambios constantes y dependen de diversos factores por lo que el ahorro económico puede variar.

Ahora para poder observar mejor el ahorro, se consideran 2000 automóviles que son aproximadamente los que usan las plazas de estacionamiento de la Universidad diariamente (ESPE Innovativa, 2020), en la Tabla 27 se presentan estos valores anuales para el escenario A y para el escenario B.

Tabla 27

Ahorro anual de toda la comunidad universitaria para el escenario A y B

Distancia (m)	Escenario A	Escenario B	Ahorro Anual de combustible (gal)	Ahorro económico Anual (S)	
				Extra	Súper
Menor de 555	16.67%	14.55%	9 026	16 700	18 958
De 555 a 1110	13.75%	21.16%	18 054	33 402	37 916
De 1110 a 1665	2.91%	2.65%	27 080	50 104	56 874
Mayor de 1665	1.32%	0.26%	36 100	66 804	75 832

Nota. Los precios de la gasolina en Ecuador están sujetos a cambios constantes y dependen de diversos factores por lo que el ahorro económico puede variar.

Como se puede observar los valores se hacen bastante representativos al considerar el gran número de vehículos que son en la comunidad universitaria. Adicional a esto se debe tomar en cuenta que al consumir menos combustible la emisión de gases se reduce, por lo que representa no solo un ahorro económico para el usuario sino también una disminución en la contaminación del ambiente.

Costos y rendimiento del sistema

El sistema de parqueadero inteligente se desarrollado bajo condiciones de prototipo, es decir con hardware y software de prueba. Esto implica que los costos se eleven ya que los dispositivos de desarrollo cuentan con herramientas y funciones que no se usan en el sistema y es un desperdicio de recursos, tanto a nivel económico como energético. Para una implementación a gran escala se debe desarrollar un dispositivo *standalone* con un hardware y software que cumpla con la función específica que se requiere para el sistema de parqueadero inteligente, así se evita desperdicio de recursos y se ve reflejado en una disminución de costos. La Tabla 28 muestra una comparativa de los costos que se maneja para el prototipo y los que se manejarían aproximadamente para un dispositivo *standalone*.

Tabla 28

Costos del prototipo y de un dispositivo standalone

Dispositivo	**Costo ($)**
Prototipo (4 plazas)	160
Prototipo (1 plaza)	40
Standalone (1 plaza)	25

Como se observa el costo del dispositivo *standalone* es menor al del prototipo, ahora considerando las 1500 plazas de estacionamiento que dispone la Universidad de las Fuerzas Armadas ESPE (ESPE Innovativa, 2020), esta diferencia de costo se vuelve significativa como se observa en la Tabla 29.

Tabla 29

Costo del sistema para toda la Universidad

Dispositivo	Costo ($)
Prototipo	60 000
Standalone	37 500

Nota. Se debe considerar que el costo de fabricación disminuye cuando se realiza a gran escala, por lo que estos valores pueden llegar a bajar.

Como se evidencia en las pruebas realizadas en sección 0, existe cobertura en todo el campus universitario por lo que no existe inconveniente en implementar el sistema en todas las plazas de estacionamiento. Además, al desarrollar un dispositivo *standalone* su despliegue es fácil ya que no necesita estar conectado físicamente a ningún otro dispositivo y gracias a que cada Gateway Sigfox soporta aproximadamente 1000000 dispositivos conectados podemos implementar un dispositivo por cada plaza de estacionamiento sin que el rendimiento de la red se vea afectado, con esto se logra que el costo del mantenimiento sea mínimo. En la Tabla 30 se presenta la inversión estimada necesaria para desplegar el sistema en el campus universitario.

Tabla 30

Costo por desplegar el sistema en el campus universitario por primera vez

	Costo ($)
Hardware	37500
Software	600
Instalación	800
Total	38900

Nota. En la parte de software se considera el desarrollo de la base de datos y el desarrollo de la aplicación móvil, aunque esta última será gratuita para los usuarios.

Finalmente, se debe considerar los costos de renovación para la conexión con la red Sigfox y los costos de mantenimiento que se deben hacer anualmente, también se debe tomar en cuenta que con el creciente despliegue de la red en el Ecuador y la gran demanda de dispositivos que se necesita para el sistema los valores en la Tabla 31 pueden llegar a disminuir.

Tabla 31

Costos anules por el sistema de parqueadero

	Costo ($)
Renovación de conexión Sigfox	6000
Mantenimiento	1500
Total	7500

Nota. El costo del mantenimiento contempla posibles fallas a nivel de hardware y el valor por mantener activa la base de datos en la nube.

CONCLUSIONES Y RECOMENDACIONES

Los sistemas de parqueadero inteligente empleados en el Ecuador en su mayoría son sistemas inalámbricos que usan ZigBee y WiFi como protocolos de comunicación entre los sensores y el Gateway. A nivel de capa física el sensor ultrasónico es el más usado por sus características técnicas frente a sensores como el infrarrojo, y a nivel de capa de aplicación la mayoría desarrolla aplicaciones móviles como medio de interacción con el usuario. La principal deficiencia que presenta estos sistemas es la baja cobertura que da la red inalámbrica con un alto consumo de energía.

Hasta el término de esta investigación en el Ecuador las redes LPWAN no han sido usadas en sistemas de parqueaderos inteligentes, pero se implementan en otras soluciones o proyectos como la medición de variables ambientales o sistemas de tracking con LORA, mientras que no hay documentación oficial de investigación o proyectos de desarrollo con Sigfox.

El despliegue de una red de sensores para la recolección de información es bastante simple y económica energéticamente, en el sistema los sensores ultrasónicos consumen alrededor de 3mA cada uno cuando están activos, y al ser simples de administrar por medio de un microcontrolador, se puede hacer que se activen solo cuando sea necesario y el resto del tiempo pasen en estado desactivado o dormido, es decir que cada cierto intervalo de tiempo se enciende. Además, la información que recogen se puede convertir fácilmente en estado libre u ocupado por medio de código, de esta forma el ahorro de energía es mayor y se tiene un sistema eficiente.

La base de datos creada en la Internet permite cumplir con el objetivo del IoT de dar acceso global a la información, al tener la opción de crear tablas por separado se puede guardar y administrar la información dependiendo la necesidad. La tabla “Parking” guarda solo el último estado de la plaza, con esto se evita la necesidad de

buscar en todo la base de datos el último estado de la plaza para así logar simplificar el proceso de enviar el estado de la plaza actual a la aplicación móvil y evitar que se muestra información incorrecta. La tabla BD_Par guardar el historial de todos los datos que llegan, para posteriormente poder analizar información adicional del parqueadero como: la utilización de cada plaza, el tráfico, etc. Adicional a esto la estructura de la base de datos permite que la información llegue del backend de Sigfox por medio de web services a cada tabla con retardos muy cortos, en las pruebas se determinó que cada mensaje se demora 444ms en guardarse en las dos tablas.

El desarrollo de una aplicación móvil permite que el usuario pueda acceder a la información del sistema de parqueadero inteligente de manera rápida y sencilla, basta con instalar en cualquier Smartphone ya que la aplicación es compatible para todos los sistemas operativos mayores a Android 4. Además, al poder usar código abierto se puede modificar librerías y código para poder utilizar solicitudes en formato json, así se pueda conectar con la base de datos de menara simple, esto permite que la interacción de la base de datos con la aplicación móvil por medio de web services tenga retardos muy bajos. Dependiendo del tipo de conexión hacia la Internet que se tenga en el dispositivo podemos tener valores de: 16ms con WiFi, 21ms con datos móviles 4G y 32ms con datos móviles 3G. En donde se determina que la conexión más rápida la obtenemos por medio de WiFi, y la más baja por medio de datos móviles 3G.

Los retardos del sistema fueron divididos en partes para poder analizar de mejor manera, se determinó que la transmisión de la información por la red Sigfox es la que introduce el mayor retardo en el sistema, esto es porque Sigfox es una red LPWAN que se caracteriza por una velocidad de transmisión baja. Según la información proporcionada por Sigfox teóricamente un mensaje se transmitirá en 2 segundos aproximadamente en un escenario ideal, en la práctica se determinó que el retardo varía

entre los 2.8 a 2.9 segundos, con lo que se comprueba los valores teóricos tomando en cuenta que no se tiene un escenario ideal.

El sistema de parqueadero desarrollado tiene como característica presentar información en tiempo real, aunque el retardo total del sistema varía entre 3.2 y 3.4 segundos, se puede considerar que, si cumple con esta característica, ya que el retardo no es percibido por el usuario y no presenta problema al momento de ver el estado de la plaza inmediatamente. Al comparar el retardo de este sistema con el desarrollado con ZigBee se observa que a una distancia cercana los retardos son parecidos, pero a medida que aumenta la distancia los retardos con ZigBee aumentan drásticamente, mientras que con Sigfox se mantienen a pesar de que la velocidad de transmisión es más baja, 100bps para Sigfox y 250kbps para ZigBee.

Uno de los parámetros importantes que permite determinar la eficiencia del sistema dentro del área de telecomunicaciones es la SNR. Sigfox al ser una red licenciada no se puede medir directamente este parámetro, pero se la puede obtener por medio del backend en forma de estadísticas o variables, lo que es de gran ayuda para realizar análisis porque es fácil de obtener la información. Al comparar la SNR de diferentes redes inalámbricas se determina que Sigfox es mejor en este punto, ya se mantiene por encima de los 27dB con distancias de separación hacia el nodo de 500m, mientras que con ZigBee en distancias mayores a 150m cae por debajo de los 5dB, se debe considerar que ZigBee no es una red LPWAN, por lo que es necesario comparar el sistema con una red de las mismas características como LORA. Al comparar se puede observar que a pesar de que con LORA a medida que la distancia crece se mantiene su SNR constante (7dB) sin tener una caída, este valor es mucho menor que la que se obtiene con Sigfox que como se mencionó antes, aunque tiene una tendencia a caer, esta no es considerable.

La eficiencia energética es otra de las caracterices importantes dentro del IoT, este sistema se puede dividir en dos partes que producen un consumo de energía, la primera es el hardware que depende de la eficiencia de los dispositivos electrónicos, en nuestro caso los sensores ultrasónicos y el microcontrolador Arduino. Este sistema al ser un prototipo desarrollado bajo dispositivos de prueba no se tiene una buena eficiencia energética ya que Arduino y el kit de desarrollo desperdician mucha energía por lo que su consumo está entre 76mA y 92mA. La segunda parte que consume energía depende de la comunicación inalámbrica, esto quiere decir el consumo en la transmisión de datos. En los datos tomados se verifica que Sigfox tiene un gran ahorro de energía en comparación a otras redes que no son LPWAN como: con ZigBee un ahorro de 36% y con WiFi un ahorro del 44% a 51%. Con esto se comprueba que Sigfox permite el desarrollo de soluciones IoT con una larga duración de vida de la batería.

Dependiendo de los escenarios y las plazas de estacionamiento que el usuario quiera ocupar se puede presentar un ahorro del tiempo en la búsqueda de una plaza libre, para el escenario A el 34% de los usuarios se benefician del sistema desarrollado, esto quiere decir 7 de cada 20 usuarios, y para el escenario B el 38% o 4 de cada 10 usuarios. La disminución del tiempo en la búsqueda de una plaza de estacionamiento depende de la zona de parqueo seleccionada por el usuario y por donde se ingrese a la universidad, en el escenario B que el ingreso es por la entrada principal se tiene un ahorro de hasta 412 segundos, y en el escenario A que el ingreso es por la parte de atrás se tiene un ahorro de hasta 422 segundos.

Un ahorro de tiempo no es el único beneficio que trae el sistema, también existe una disminución en la contaminación y una reducción en el consumo de combustible, lo que significa un ahorro económico. Para un usuario el ahorro económico diario es de centavos y hasta anual no es tan considerable, con valores de ahorro anuales de 8 a 33

dólares en gasolina extra y de 9 a 37 dólares en gasolina súper. Pero este valor se vuelve significativo cuando se considera toda la comunidad universitaria que hace uso de los parqueaderos, los valores anuales ascienden a 16 mil y 66 mil dólares en gasolina extra y 18 mil y 75 mil dólares en gasolina súper.

El sistema de parqueadero inteligente permite tener una red de sensores conectados inalámbricamente que nos proporciona información de las plazas de estacionamiento, y gracias a la red LPWAN Sigfox se tiene conectividad en todo el campus, esto quiere decir que se puede obtener datos de todas las plazas disponibles en la Universidad. Estos datos subidos a la nube y junto con el desarrollo de la aplicación móvil SMART ESPE se obtiene un sistema que interactúa con el usuario y da información del parqueadero, con lo cual se evita que los conductores pierdan tiempo haciendo un recorrido innecesario por el campus para encontrar una plaza de estacionamiento libre, lo que significa: un ahorro de tiempo y de combustible, un ahorro económico, menos contaminación al ambiente y una mayor velocidad de desplazamiento, lo que da como resultado una mejor movilidad.

Como el parqueadero del campus esta descubierto se recomienda usar el hardware apropiado para exteriores, esto quiere decir que tengas las protecciones necesarias para poder resistir escenarios de lluvia, sol, etc.

Se recomienda utilizar paneles solares para tener un sistema autónomo y que no se tenga la necesidad de recargar la batería con un dispositivo externo, además se recomienda usar un cargador de batería como el TP4056 que evita daños al circuito y a la batería.

Dentro de las soluciones IoT para la creación de una Smart University se recomienda desarrollar un sistema de monitoreo de variables ambientales que permita corroborar la disminución de contaminación ambiental.

Nuestro grupo está interesado en continuar con esta línea de investigación por lo cual propone desarrollar la aplicación móvil para otros sistemas operativos como iOS y Windows Mobile, para poder implementar el sistema en la Universidad o en cualquier parqueadero y pueda ser compatible con todos los teléfonos inteligentes de los usuarios a nivel de capa de aplicación. Además, dentro de esta capa se debe desarrollar servicios adicionales al usuario como: información del número de plazas libres por zona de parqueo, la ruta más corta a la plaza destino, opciones de plazas libres si la plaza objetivo se encuentra ocupada, y para implementaciones en parqueaderos externos a la Universidad se debería dar información de tarifado.

Para la implementación de los dispositivos en las plazas estamos interesados en utilizar un hardware eficiente en cuanto a consumo de energía y que sea exclusivo para identificar el estado de la plaza, para así poder determinar el consumo mínimo del sistema, la duración de vida de la batería y el costo final. Adicional a esto se debería usar un transmisor Sigfox por cada plaza de estacionamiento así se evita el uso de cables y se tiene un dispositivo *standalone* autónomo, portable y con una mejor eficiencia. Con esto se puede llegar a desplegar a gran escala en los parqueaderos de la Universidad o en cualquier parqueadero de manera fácil y rápida.

Finalmente se propone realizar un estudio enfocado en el desempeño de la red Sigfox, en el backend están disponibles variables que sirven para identificar las limitaciones de la red, como son: SNR, RSSI, retardos, etc. Se debería implementar en diferentes escenarios, por ejemplo: zonas urbanas y rurales, cuando existe movimiento, en lugares cubiertos como sótanos, etc. También se necesita verificar si la información de cobertura proporcionada por Sigfox es correcta y cómo se comportan las variables mencionadas anteriormente cuando se pasa el límite de cobertura. Con esto se lograría

determinar las características y las soluciones IoT pueden ser desarrolladas usando Sigfox y en qué escenarios y lugares del país se puede desplegar.

REFERENCIAS

Ahamed, S. (2016). *Android Operating System: Architecture, Security Challenges and Solutions.* South Eastern University of Sri Lanka, Oluvil, Sri Lanka.

Akhrif, O., & Hmina, N. (2018). Smart university: SOC-based study. *Eurographics Symposium on Computer Animation (SCA), 10*(11), 1-6.

Arduino. (Noviembre de 2019). *Arduino*. Obtenido el 13 de noviembre de 2019 de: https://www.arduino.cc/

Ariza, M. J. (2019). *Teoría de medidas.* Departamento de Física Aplicada. CITE II-A.

Belloche, S. (2015). On-street parking search time modelling and validation with survey-based data. *Transportation Research Procedia, 6*(1), 313–324.

Borondo, B. S. (2015). *Implementación de una solución Internet of Things base para futuros desarrollos de aplicaciones verticales enfocadas a hacer eficientes, optimizar y gestionar ámbitos o negocios concretos.* Universidad Autónoma de Madrid, Madrid.

Cárdenes, T. D. (2016). *Diseño e implementación de una herramienta para la verificación de cobertura de la red SIGFOX. Estudio de conectividad en una zona geográfica de orografía compleja.* Máster Universitario en Ingeniería de Telecomunicaciones, Universitat Oberta de Catalunya, Sistemas de Comunicación, Catalunya.

Castillo, S. P. (2018). *Monitorización de parámetros medioambientales mediante sensores y la red Sigfox.* Escuela Politécnica Superior de Linares, Jaén.

Centenaro, M., Vangelista, L., Zanella, A., & Zorzi, M. (Octubre de 2016). Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications, 23*(3), 60-67.

Chiriboga, T. A. (2020). *DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN CONTECNOLOGÍA LORA PARA EL MONITOREO DE UBICACIÓN VEHICULAR CON UN APLICATIVO WEB.* SANGOLQUÍ.

Cobos, D. A. (2016). *Diseño e implementación de una arquitectura IoT basada en tecnologías Open Source.* Trabajo Fin de Máster, Escuela Técnica Superior de Ingeniería Universidad de Sevilla, Dep. Ingeniería Telemática, Sevilla.

Córdoba, C., & Plazas, B. (Junio de 2015). Prototipo de control y monitoreo para parqueaderos vehiculares. *Tekhnê, 12*(1), págs. 67-72.

Elmasri, R., & Navathe, S. (2011). *Fundamentals of Database System* (Vol. 6). Addison-Wesley: Pearson Education, Inc.

ESPE Innovativa. (2020). *Información sobre los estacionamientos de la Universidad.* Sangolquí.

Godoy, R. C. (2015). *Desarrollo de un prototipo de parqueadero inteligente para la automatización del sistema de aparcamiento Simert en la ciudad de Loja.* UNIVERSIDAD NACIONAL DE LOJA, Loja.

Google Developer Training Team. (2016). *Learn to develop Android Applications.* New York: Google.

Guimarães, P. Â., Benessia, A., & Curvelo, P. (2013). *Agency in the Internet of Things.* Luxembourg: Publications Office of the European Union.

ILNAS. (2018). *INTERNET OF THINGS (IoT) · TECHNOLOGY, ECONOMIC VIEW AND TECHNICAL STANDARDIZATION.* Luxembourg: ANEC.

INEC. (2017). *Tecnologías de la Información y Comunicación.* Ecuador: TIC.

ITU-T. (2012). *Series Y: Global Information Infrastructure, Internet Protocol Aspects And Next-Generation Networks.* ITU-T, Geneva.

Jaya, M., Vizcaíno, I., & Acosta, F. (2014). *MONITORIZACIÓN DE CONSUMO ENERGÉTICO EN REDES INALÁMBRICAS WI-FI CON DIFERENTES ESTÁNDARES DE COMUNICACIONES Y VOLÚMENES DE TRÁFICO.* Sangolquí.

Kirthika, B., Prabhu, S., & Visalakshi, S. (2015). Android Operating System: A Review. *International Journal of Trend in Research and Development, 2*(5), 260-264.

Kosmatos, E., Tselikas, N., & Boucouvalas, A. (2011). Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture. *Advances in Internet of Things, 1*(1), 5-12.

Liang, D. (2015). *INTRODUCTION TO JAVA PROGRAMMING* (Vol. 10). New Jersey: Pearson Education, Inc.

Márquez, R. M. (2015). *Desarrollo de un prototipo de parqueadero inteligente empleando redes de sensores inalámbricos.* Universidad de las Fuerzas Armadas Espe, Sangolquí.

Mekkia, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express, 5*(1), 1-7.

Mustafa, K. (2016). *PHP Web Services Basic Documentation.*

Oualline, S. (1995). *Practical C++ Programming.* United States of America: O'Reilly & Associates, Inc.

Pacheco, M. (02 de Marzo de 2020). *El precio de la gasolina súper baja en marzo del 2020 por quinto mes consecutivo.* Obtenido el 10 de marzo de 2020 de: https://www.elcomercio.com

Pérez, H. R. (2015). *Desarrollo de prototipo de sensor IoT usando la red SigFox.* Escuela Técnica Superior de Ingeniería Universidad de Sevilla, Sevilla.

Real Wireless . (2015). *A Comparison of UNB and Spread Spectrum Wireless Technologies as used in LPWA M2M Applications.* England : Real Wireless Limited.

Rosales, L. (2016). *Diseño e implementación de un estacionamiento inteligente utilizando arduino Yun basado en Internet de las cosas (IoT).* Trabajo de grado, Universidad Politécnica Salesiana, Guayaquil.

Sánchez, B. J. (2015). *Transformación Digital de la Universidad (XI). La experiencia en la UA.* Universidad de Alicante, Alicante.

Shoup, D. C. (2006). Cruising for parking. *Transport Policy, 13*(6), 479-486.

Sigfox. (2019). *Sigfox Build*. Obtenido el 15 de octubre de 2019 de: https://build.sigfox.com/sigfox

Sigfox. (2019). *Sigfox Pagina Oficial*. Obtenido el 15 de octubre de 2019 de: https://www.sigfox.com/en

Sigfox. (Febrero de 2020). *Backend Sigfox*. Obtenido el 4 de febrero de 2020 de: https://backend.sigfox.com/welcome/news

Smyth, N. (2015). *Android Studio Development Essentials* (Vol. 2). eBookFrenzy.

Somayya, M. (Agosto de 2015). Internet of Things: Smart Things. *International Journal of Future Computer and Communication, 4*(4), 250-253.

Sterling, B. (2005). *Shaping Things.* Cambridge: Massachusetts Institute of. Mediaworks Pamphlets.

Tahon, M., Verbrugge, S., Lannoo, B., Van Ooteghem, J., De Mil, P., Pickavet, M., & Demeester, P. (2010). Parking Sensor Network: Economic Feasibility Study of Parking Sensors in a City Environment. *Conference: Telecommunication, Media and Internet Techno-Economics, 9th Conference, Proceedings*, (págs. 1-8).

Tatroe, K., MacIntyre, P., & Lerdorf, R. (2013). *PROGRAMMING PHP* (Vol. 3). Estados Unidos: O'Reilly Media, Inc.

Technologies, C. (2013). Datasheet HCSR04 Ultrasonic Sensor.

Thinxtra. (Noviembre de 2019). *thinxtra Empowering Internet of Things*. Obtenido el 20 de noviembre de 2019 de: https://www.thinxtra.com/

Usman, O. L., Akeem, O., & Gbenga, O. (2016). INTRODUCTION TO COMPUTER PROGRAMMING (BASIC). *College of Science and Information Technology, 1*(1), 130-137.

Webhost. (2020). *000webhost*. Obtenido el 10 de enero de 2020 de: https://www.000webhost.com/

APÉNDICE