Universidad Austral de Chile
Facultad de Ciencias de la Ingeniería
Escuela de Ingeniería Civil en Informática

# DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE ESTACIONAMIENTO INTELIGENTE UTILIZANDO TECNOLOGÍA IOT CONECTADOS A LA RED LoRa

Proyecto para optar al título de
**Ingeniero Civil en Informática**

PROFESOR PATROCINANTE:
CHRISTIAN LAZO RAMIREZ
INGENIERO EN GESTIÓN INFORMÁTICA
DOCTOR EN INGENIERÍA TELEMATICA

PROFESOR INFORMANTE 1:
LUIS HERNAN VIDAL VIDAL
INGENIERO CIVIL EN INFORMÁTICA
M.B.A. DIPL.

PROFESOR INFORMANTE 2:
LUIS ALBERTO ÁLVAREZ GONZÁLEZ
INGENIERO CIVIL ELECTRICISTA
MAGÍSTER EN INGENIERÍA INFORMÁTICA

**RICHARD LUCIANO VERA CISTERNAS**

VALDIVIA – CHILE
2021

# ÍNDICE

# ÍNDICE DE TABLAS

Tabla Página

# ÍNDICE DE FIGURAS

Figura Página

# RESUMEN

Con el paso del tiempo, el mercado automotriz crece en gran medida, pues con el lanzamiento de una nueva generación de vehículos a mediados de cada año, aquellos de años anteriores bajan su valor, permitiendo así al consumidor, tener múltiples opciones a la hora de adquirir un vehículo usado o nuevo. Es por esto, que cada vez se ven más automóviles en las calles, produciendo así una gran congestión vehicular en el centro de la mayoría de las ciudades. Los estacionamientos tanto municipales como privados son opciones de descongestión vehicular, sin embargo, los conductores no tienen información exacta y precisa del estado de ocupación de estos estacionamientos.

El siguiente proyecto de título tiene como fin desarrollar un prototipo de Estacionamiento inteligente o smart parking que sea capaz de identificar si un vehículo está utilizando un slot de estacionamiento, con el fin de que los conductores mediante una aplicación móvil sean capaces de visualizar el estado completo en un centro de estacionamientos.

Para la elaboración del prototipo, se investigó sobre la tecnología IoT que utiliza como interfaz de comunicación la red LoRaWAN, esto quiere decir, todos aquellos microcontroladores que usen este protocolo. Junto con esto, también la compatibilidad de aquellos sensores que serán utilizados para la detección del automóvil.

La elaboración del proyecto completo se llevó a cabo en tres etapas, la primera fue el desarrollo del prototipo IoT, la segunda etapa consistió en la elaboración de la aplicación móvil y la tercera etapa involucra las pruebas y validación de la solución.

El prototipo fue implementado mediante dos microcontroladores Heltec WiFi LoRa 32 (v2), en donde uno cumple la función de ser el emisor, aquel que determina si el slot está ocupado o no con la ayuda de sensores, envía el estado de este mediante red LoRa y el receptor, quien capta los datos enviados por el receptor, los sube mediante red WiFi a la base de datos en tiempo real de Firebase. El desarrollo de la aplicación móvil fue llevado a cabo mediante el *framework* React Native, el cual permite crear aplicaciones híbridas, es decir, que podrán funcionar tanto en Android como iOS, elaborando solo una base de código y se conectará con la base de datos para mostrar de forma visual e intuitiva el estado del estacionamiento.

Las pruebas fueron llevadas a cabo en el estacionamiento del domicilio del tesista.

# ABSTRACT

The Car markplace has been in constant growth in recent years. This is mainly due to the release of the new generation's models at the half of each year, where lasts generation model gets their prices significantly cut down and become much more affordable. This has the effect of an increase vehicle density in the streets, which produces vehicular congestion, especially in the city's center. Private and public parking lots can alleviate vehicular congestion, but drivers have information regarding the location and availability of a parking spot.

This degree project presents the development of a Smart Parking solution, aiming to solve this issue. This solution will be able to identify if a parking spot is occupied and keep track of the current state of the parking lot, to inform drivers via a smartphone app (Android and iOS) where they will be able to visualize the state of every parking spot.

IoT technology that uses the LoRaWAN communication protocol was investigated and used. This includes an array of compatible microcontrollers such as the ESP32. In addition to each one, sensor compatibility was also taken in account, for the sensors that will be used for vehicles detection.

The project consisted in three stages: first was the development of the prototype, the second of the smartphone App., and the third, the testing and validation of the complete solution.

The prototype consisted of two Heltec WiFi LoRa 32 (v2) microcontrollers with different functions. One of them is the emitter, whose using sensors, it is able to check the state of the parking spot and send it through the LoRa network to the other microcontroller, which acts as a receiver, uploading them through WiFi to a Firebase's realtime database. The App was developed using the framework React Native, which allows to create hybrids apps, that runs on both, Android and iOS operating systems, with one codebase. The App reads the database and shows visually and intuitively the state of the parking lot.

Tests were conducted in the parking spot of a urban residence.

# 1. INTRODUCCIÓN

Hoy en día las personas tienen mayor consideración en trasladarse desde las zonas rurales hasta las urbanas, esto ya que, en esta última, es más fácil el acceso a los servicios básicos, servicios de salud, transporte, acceso a educación y otros servicios o productos. Según las Naciones Unidas, en 2030 el 60% de la población mundial vivirá en zonas urbanas (United Nations, 2018). Sin embargo, así como esta entrega facilidades a la población, con el aumento de las zonas urbanas, también nacen los problemas. Entre los problemas se encuentran la contaminación ambiental del aire, producto de la emanación de gases de los vehículos; congestión vehicular, explotación de los recursos naturales para satisfacer a la población, entre otros. Sin embargo, las tecnologías de la información y comunicación (TIC) son una posible solución a las problemáticas de las ciudades urbanas. El Internet de las Cosas (IoT: Internet of things, siglas en inglés) es la conexión digital entre el Internet y dispositivos tales como, sensores, microcontroladores, prototipos, entre otros aparatos tecnológicos, capaces de enviar datos por la red (Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., & Riegg, A.,2016). Este concepto permite resolver problemáticas medioambientales, tráfico y congestión vehicular, industriales, agrícolas, entre otras, dando paso al concepto Ciudad Inteligente.

Ciudad Inteligente o Smart City, es un concepto que resuelve las problemáticas de las ciudades urbanas mencionadas anteriormente, con Tecnologías de información y comunicación, como lo son las IoT. Técnicamente, es una red basada en infraestructura que recopila información de un tipo diferente de sensores eléctricos para administrar los recursos de manera efectiva. La información recopilada se utiliza para monitorear el proveedor de agua, los hospitales, los sistemas de tráfico y transporte, y otros servicios (Bertino, E., Choo, K.-K. R., Georgakopolous, D., & Nepal, S., 2016).

La cantidad de automóviles en el mundo va en aumento, llegando a un vehículo cada seis personas en 2018, provocando grandes congestiones vehiculares tanto en los países desarrollados, como en aquellos que crecen rápidamente con el paso de los años, como lo son la India, Rusia, Brasil, entre otros. Si antes, cada 5 años, el parque automotriz aumentaba en aproximadamente 100 millones, hoy en día, aproximadamente cada año, crece en 100 millones de vehículos (Baeza M, 2018). Los estacionamientos tanto municipales como privados son opciones para descongestionar las ciudades y sitios seguros donde los conductores pueden aparcar sus vehículos. Sin embargo, los estacionamientos presentan ciertas problemáticas, como la desinformación de las plazas disponibles, el lugar de las plazas disponibles, el costo en tiempo real respecto al uso de este, aglomeraciones de vehículos en horas *peak*, entre otros. Los problemas que provocan los estacionamientos pueden afectar de manera directa a una ciudad, pero gracias al concepto Smart City, es posible encontrar soluciones tecnológicas. Entre todas las soluciones que cubre Smart City, se encuentra el Estacionamiento inteligente o Smart Parking, el cual busca automatizar los estacionamientos de la ciudad, con el fin de que los conductores sean capaces de informarse del estado de los estacionamientos, y gracias a esto serán capaces de ver tanto el número de plazas vacías, como los lugares del estacionamiento en donde se encuentran las plazas vacías (Mouchili, M. N., Aljawarneh, S., & Tchouati, W., 2018). Aportando una disminución en los tiempos de búsqueda de

plazas vacías, aumento en el flujo dentro de estacionamientos, comodidad para los conductores, baja en la congestión de vehículos y disminución en contaminación CO2 por parte de los vehículos.

Como se mencionó anteriormente, los dispositivos utilizados en soluciones Smart, deben ser capaces de enviar datos mediante tecnologías de comunicación, y es aquí, donde en los últimos años se ha innovado con el fin de que estos dispositivos sean más eficientes en cuanto a consumo de energía, considerando, además, que estos puedan estar a distancias considerables sin perder la comunicación. Una de estas tecnologías es la red LoRa, esta es una red de alto alcance y bajo consumo, que apunta directamente a solucionar los problemas de consumo de energía de las soluciones IoT (Sanchez, R., Sanchez, J., Ballesta, J., Cano, M., Skarmeta, A., 2018), con el fin de aumentar la autonomía de estos, y contribuir a mejorar el impacto medioambiental de este.

## 1.1. Contexto

En Valdivia, el parque automotriz aumenta anualmente en aproximadamente 4000 a 5000 vehículos (INE, 2019), por lo que en estacionamientos como los ubicados en el centro de la ciudad de Valdivia, el *Mall* de Valdivia o de los campus de la UACh exista una alta congestión vehicular. Esto para los conductores es perjudicial pues, afecta los niveles de estrés, provoca pérdida de tiempo buscando plazas vacías y gastos extras en combustible.

Tanto a nivel municipal como Universidad, no se han tomado las medidas correspondientes para atacar estas problemáticas que acometen a los conductores, sin embargo, gracias a los avances en los distintos ámbitos de la tecnología, es posible construir soluciones capaces de atacar las problemáticas ya mencionadas, permitiendo que la experiencia de los conductores a la hora de necesitar aparcar un vehículo sea simple, rápida y eficaz. En las figuras 1 y 2 se puede visualizar la congestión vehicular que ocurre en la costanera de la ciudad de Valdivia.

Figura 1: Costanera lado sur congestionado de vehículos

Figura 2: Costanera lado norte congestionado de vehículos

## 1.2. Objetivos

### 1.2.1. Objetivo General

Desarrollar un prototipo de Estacionamiento inteligente que utilice tecnología innovadora y sustentable, que permita a los conductores mejorar su experiencia de conducción a la hora de buscar un lugar donde aparcar.

### 1.2.2. Objetivos específicos

1. Investigar la situación actual de soluciones IoT implementadas en Smart Parking, principalmente aquellas que utilizan red LoRa o LPWAN.
2. Seleccionar la tecnología tanto IoT como software, disponible en la actualidad, para llevar a cabo un prototipo completo de Smart Parking, post evaluación de estas tecnologías.
3. Desarrollar un prototipo IoT que detecte un automóvil y que utilice la red LoRa para enviar los cambios de estado captados por los sensores.
4. Desarrollar una aplicación móvil de Smart Parking que sea intuitiva, segura y simple para el conductor.
5. Probar el sistema completo en condiciones de laboratorio.
6. Validar el sistema completo, comparándolo con soluciones existentes.

### 1.2.3. Impacto esperado del proyecto

Este proyecto pretende ser precursor en la automatización de estacionamientos dentro de la ciudad de Valdivia, ya que, con la visualización tanto del número de plazas vacías como la ubicación de estas, se logrará una mayor descongestión vehicular a nivel ciudad, los conductores no desperdiciaron recursos como tiempo, combustible y dinero, y además se logrará disminuir las emisiones de $CO_2$ de los automóviles en Valdivia.

## 2. SITUACION ACTUAL

En los países del primer mundo, tales como Estados Unidos, Canadá, Corea, China, Japón y la mayoría de los países europeos, ya se han llevado a cabo soluciones de tipo Smart Parking, con el fin de descongestionar las ciudades, y facilitar información a los conductores respecto el estado de los estacionamientos que presentan disponibilidad de plazas. Estas soluciones pueden ser distintas unas de otras, ya que pueden ser implementadas con tecnologías como: Reconocimiento de imágenes mediante algoritmos de *Machine Learning* o detección de vehículos por medio de sensores, ambas soluciones se centran en detectar si una plaza está siendo utilizada. Ya que este proyecto busca llevar a cabo una solución IoT, se llevó a cabo una revisión bibliográfica respecto a implementaciones de smart parking utilizando internet de las cosas y tanto red LoRa como otro tipo de red para enviar la información de los sensores, esto ya que al ser soluciones IoT, la arquitectura será similar, solo cambia el medio por el cual se suben los datos a la base de datos.

Entre las soluciones de Smart Parking utilizando tecnología IoT y una red distinta a LoRa se encuentra:

En 2016 en Corea, se propuso un Smart-Parking basado en IoT de bajo costo (Lee, C., Han, Y., Jeon, S., Seo, D., & Jung, I., 2016), en el cual proponen soluciones para estacionamientos cerrados y abiertos. En el primero utiliza un sensor de ultrasonido ubicado en el techo, con el fin de que este mida la distancia al suelo, si esta es menor a la configurada en el microcontrolador, entonces hay un vehículo y se marca la plaza como ocupada. En tanto en los estacionamientos abiertos, se utiliza solo un sensor magnético ubicado en el suelo, encargado de detectar los cambios en el campo magnético que percibe, si el campo magnético sufre cambios, significa que un vehículo está utilizando la plaza, por lo que se marca como ocupado. Los nodos que se encargan de enviar los cambios de estado del estacionamiento se comunican con dispositivo de enlace mediante ondas de radio frecuencias bajas, y una vez recibido estos datos, el enlace lo sube al servidor. Este sistema, además, utiliza la conexión Bluetooth de baja potencia, para que el conductor en la aplicación desarrollada para smartphones reciba un ID, entregada por el nodo ubicado en la plaza en donde se estaciono, y gracias a esta implementación el conductor puede ver en qué lugar del estacionamiento quedó aparcado su vehículo y en tiempo real el uso.

Entre las soluciones de Smart Parking utilizando tecnología IoT y red LoRaWAN se encuentra:

Libelium, una empresa de origen español se dedica a desarrollar soluciones de tipo Smart City, y entre estas se encuentra su producto Smart Parking Node (Libelium, s.f). Este dispositivo es un nodo compuesto por un microcontrolador, sensor de radar, sensor magnético y una batería con una vida útil de hasta 10 años. Esto se encuentra encapsulado en un material resistente manufacturado por Libelium. Este nodo debe ser instalado en el

centro de las plazas de un estacionamiento y anclado al piso. Para enviar los cambios de estado, el nodo utiliza LoRaWAN, y se comunica con una base LoRa, que se encarga de subir los datos a los servidores LoRa. Finalmente, para acceder a los datos obtenidos, Libelium ofrece su propio sistema Cloud, el cual se puede conectar con servicios como Microsoft Azure o Amazon Web Services. Así como también que el consumidor cree su propio sistema, con el fin de que este utilice las herramientas que estime conveniente. Este producto se encuentra certificado por la Unión Europea, por la Comisión Federal de Comunicaciones de los Estados Unidos, entre otros. El costo de un nodo cuesta alrededor de $400.000 CLP [10].

En cuanto a soluciones implementadas en Chile, en 2017 la Universidad de la Frontera presentó un prototipo de Smart Parking en la región de la Araucanía, luego de que múltiples entidades se unieran presentando un proyecto de mejora de calidad de vida de las personas en la ciudad. El proyecto de la universidad se testeó en el estacionamiento de esta, obteniendo un 90% de precisión de efectividad, pasando las pruebas técnicas y comerciales (Burgos G, 2017).

Dentro del Instituto de Informática de la Universidad Austral de Chile, en 2018 se llevó a cabo un proyecto de título relacionado con Estacionamiento inteligente, denominado “PROTOTIPO APLICACIÓN MÓVIL PARA ESTACIONAMIENTO INTELIGENTE”, el cual utilizaba una cámara para reconocer el estado de las plazas de estacionamiento del instituto y mostrarlas por medio de una aplicación móvil. El reconocimiento de las plazas se desarrolló mediante procesamiento de imágenes utilizando algoritmos de *Machine Learning*.

# 3. TECNOLOGÍAS EVALUADAS

## 3.1. Microcontroladores

Los microcontroladores son circuitos integrados que contienen todos los componentes de un computador, tales como: microprocesador, memoria principal y puertos de entrada y salida (López, G. & Margni, S., 2002). Hoy en día son utilizados para la automatización de procesos, proyectos IoT, robótica, entre otras tareas o actividades.

En el mercado, es posible encontrar un sin número de microcontroladores, cada uno con distintivas características en su hardware, sin embargo, para este proyecto, se enfocó la búsqueda en el bajo consumo de energía, diseño compacto, compatibilidad de sensores, conectividad a redes LoRa e Internet.

### 3.1.1. ESP8266 y ESP32

El ESP8266 (Espressif, 2020) es un chip de bajo costo y consumo de energía, desarrollado por Espressif Systems, que incorpora tecnología Wi-Fi compatible con el protocolo TCP/IP. Incorpora un microprocesador de doble núcleo o de solo un núcleo Tensilica Xtensa LX106, que puede trabajar hasta los 80 MHz. Entre las aplicaciones en que se utilizan este chip, están las siguientes: automatización de casas, control inalámbrico industrial, cámaras IP, redes de sensores, entre otras.

Puede ser encontrado en el mercado de múltiples formatos, ya sea incorporado en una placa de desarrollo (ver Figura 3) o en forma de chip (ver Figura 4). En formato placa de desarrollo, su utilización se vuelve más sencilla, ya que la interacción con el desarrollo de algoritmos y conexión a dispositivos tales como protoboards o sensores, es directa.

Figura 3: ESP8266 formato placa de desarrollo

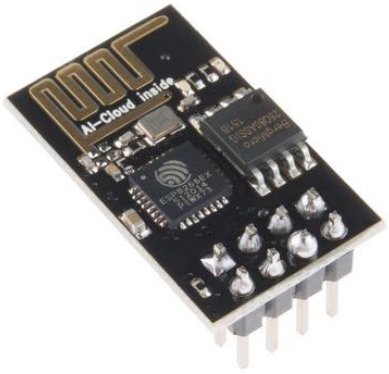



Figura 4: ESP8266 formato chip

El ESP32 (Espressif, 2021) es un chip de bajo costo y consumo de energía, desarrollado por Espressif Systems, que incorpora tecnología WiFi y Bluetooth en modo dual. Incorpora un microprocesador Tensilica Xtensa LX6 de doble núcleo o de solo un núcleo, capaz de llegar a los 160 o 240 MHz. Por sus características, este chip, es el sucesor del ESP8266 y entre las aplicaciones que puede llevar a cabo, se encuentran las siguientes: reconocimiento de imágenes, aplicaciones para el cuidado de la salud, transmisión de video, además de las ya mencionadas en el ESP8266 y otras aplicaciones.

Al igual que en su predecesor, se puede encontrar en incorporado en una placa de desarrollo (ver Figura 5) o formato chip (ver Figura 6).
Las características principales con las que cuentan ambos chips se pueden visualizar en la Tabla 1.

Figura 5: ESP32 formato placa de desarrollo


Figura 6: ESP32 formato chip

Tabla 1: Comparación ESP8266 y ESP32

| **Especificación** | **ESP8266** | **ESP32** |
|---|---|---|
| Procesador | Tensilica Xtensa LX106, un núcleo. | Tensilica Xtensa LX6, doble núcleo. |
| WI-FI | SI, HT20 | SI, HT-40 |
| Bluetooth | No | Si, Bluetooth versión 4.2. |
| Frecuencia Procesador | 80 MHz | 240 MHz |
| Pines GPIO | 17 | 36 |
| Hardware/Software PWM | No / 8 canales | 1 / 16 canales |
| SPI / I2C / I2S / UART | 2/1/2/2 | 4/2/2/2 |
| Costo | $3.500 CLP | $5.000 CLP |

Ambos microcontroladores pueden ser considerados para llevar a cabo el proyecto, sin embargo, el mercado ofrece mejores variantes del ESP32, pues estas incorporan el módulo de comunicación LoRa, de forma integrada en su placa de desarrollo.

### 3.1.2. Heltec WiFi Lora 32 (v2)

WiFi LoRa 32 v2 (Heltec, 2018) es una placa de desarrollo de IoT (ver Figura 7) diseñada y producida por Heltec Automation (TM). Incorpora el microcontrolador ESP32 de doble núcleo, por lo que es compatible con WiFi a 2.4 GHz y Bluetooth 4.2. Además, integra el chip LoRa SX1276, permitiendo así, que sea compatible con este protocolo de comunicación. Incorpora además una pantalla OLED de 0.96", que permite monitorear o visualizar información que se requiera mostrar en el panel. Es compatible con la IDE de Arduino, lo que permite que la interacción del usuario o programador con esta placa de desarrollo sea sencilla y amigable a la hora de la elaboración de algoritmos. Este dispositivo se creó como una opción para el desarrollo de soluciones Smart City, tales como: Smart Farming, Smart Home, Smart Parking, entre otras soluciones.

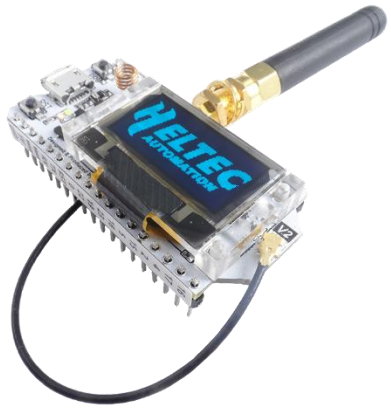



Figura 7: Heltec WiFi LoRa 32 (v2)

El detalle de las características de esta placa de desarrollo se puede visualizar en la Tabla 2.

Tabla 2: Características Heltec WiFi LoRa 32 (v2)

| **Recurso** | **Parámetro** |
|---|---|
| Chip Principal | ESP32 (Tensilica LX6 Dual Core @240 MHz) |
| Comunicación Inalámbrica | – WiFi (802.11 b/g/n de hasta 150 Mbps)<br>– Bluetooth (Version 4.2)<br>– LoRa (Comunicación nodo a nodo o LoRa WAN). |
| Chip LoRa | SX1276 |
| Memoria Flash | 8 MB (64-bit) |
| RAM | 520 KB SRAM |
| Tamaño Pantalla | 0.96" OLED |
| Recurso de Hardware | UART x 3, SPI x 2, I2C x2, I2S x 1,entrada ADC x 18 de 12 bits, salida DAC x 2 de 8 bits, GPIO x 22, GPI x 6. |

### 3.1.3. TTGO LoRa32 ESP32

TTGO LoRa32 ESP32 (LILYGO, s.f) es una placa de desarrollo IoT (ver Figura 8) diseñada y producida por LILYGO Company. Esta placa de desarrollo cuenta con el chip ESP32 de doble núcleo, lo que lo hace compatible con la comunicación WiFi y Bluetooth. Cuenta con el chip LoRa SX1276, ideal para el proyecto que se desea llevar a cabo. Incorpora una pantalla OLED de 0.96", permitiendo la visualización de información en la misma placa. Es compatible con Arduino IDE, por lo que la carga de programas en la placa de desarrollo es sencilla. Entre sus principales usos, se encuentran los proyectos que buscan resolver problemas que requieran la incorporación de las tecnologías IoT.

Figura 8: TTGO LoRa32 ESP32

El detalle de las características de esta placa de desarrollo se puede visualizar en la Tabla 3.

Tabla 3: Características TTGO LoRa32 ESP32

| Recurso | Parámetro |
|---|---|
| Chip Principal | ESP32 (Tensilica LX6 Dual Core @240 MHz) |
| Comunicación Inalámbrica | • WiFi (802.11 b/g/n de hasta 150 Mbps)<br>• Bluetooth (Version 4.2)<br>• LoRa (Comunicación nodo a nodo o LoRa WAN). |
| Chip LoRa | SX1276 |
| Memoria Flash | 4 MB (32-bit) |
| RAM | 520 KB SRAM |
| Tamaño Pantalla | 0.96" OLED |
| Recurso de Hardware | UART x 3, SPI x 2, I2C x2, I2S x 1, entrada ADC x 16 de 12 bits, salida DAC x 2 de 8 bits, GPIO x 20, GPI x 8. |

### 3.1.4. Selección de Microcontrolador

Como se observó en los puntos anteriores, tanto la versión Heltec como la TTGO, tienen características muy similares, por lo que cualquiera de estas opciones cumple con las necesidades que requiere el proyecto. Analizando el mercado de microcontroladores disponible en Chile, solo se encuentra en Stock la versión Heltec y en cuanto a la versión TTGO, esta se encuentra agotada, y la única forma de adquirir este microcontrolador, es importándolo desde sitios extranjeros. Debido a esto, en el proyecto se utilizará Heltec WiFi LoRa 32 (v2) para el desarrollo del prototipo que capturará los datos en las plazas de estacionamiento.

## 3.2. Redes y Comunicación

Como se mencionó anteriormente, los dispositivos se comunicarán entre sí a través de la red LoRaWAN, sin embargo, también será necesario utilizar otro protocolo de comunicación que permite en enlace hacia Internet, siendo así, el más apto, WiFi, pues los microcontroladores ya cuentan con la integración de este protocolo. A continuación, se detallarán los protocolos de red mencionados anteriormente.

### 3.2.1. LoRaWAN

LoRaWAN (LoRa Alliance, s.f) es un protocolo de red de área amplia y de baja potencia desarrollado por LoRa Alliance. Diseñado para dispositivos que requieran una batería como medio de alimentación, y necesiten conectarse a internet de forma inalámbrica. Su principal enfoque es el desarrollo de aplicaciones smart, para resolver problemas relacionados con: control de energía, control de la contaminación, eficiencia de infraestructura, prevención de desastres, entre muchas más. Como ya se mencionó, LoRaWAN es el protocolo de comunicación entre dispositivos, pero LoRa, es la capa física o la modulación inalámbrica utilizada para crear los enlaces de comunicación de largo alcance (ver Figura 9).

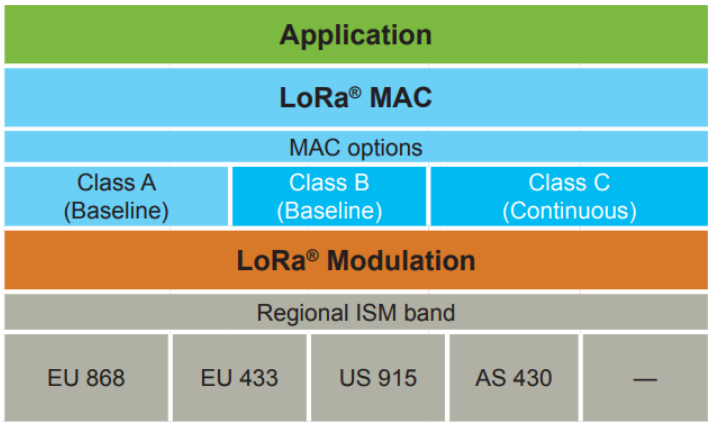



Figura 9: Capa LoRa

Entre sus principales características destacan:

- Banda de frecuencia: Varía según la región (Europa: 868 MHz, Norte América: 915 MHz, Asia: 433 MHz).
- Alcance: 10 a 15 KM.
- Data Rate: 290 bps a 50 Kbps.
- Máximo número de mensajes por día: Ilimitado.
- Estimación de consumo de una batería de 2000 mAh: 105 meses.
- Eficiencia Energética: Muy alta.
- Inmunidad a la interferencia: Muy alta.
- Seguridad: Si, Cifrado AES 128.

La compatibilidad de LoRa con los dispositivos IoT cada vez aumenta, y es posible encontrar el chip SX1276 en el mercado de forma independiente (ver Figura 10) o incorporado ya en una placa de desarrollo, como las ya mencionadas Heltec Wifi LoRa (v2) y TTGO LoRa32 ESP32. El chip mencionado tiene un valor aproximado de $10.000 CLP. Cabe mencionar que, dependiendo de las necesidades del proyecto, será necesario invertir de un Gateway LoRa (ver Figura 11), el cual se encarga de recibir los mensajes LoRa, que envían lo dispositivos, para posteriormente subirlos a Internet mediante WiFi, Ethernet o GSM (3G, 4G o 5G).

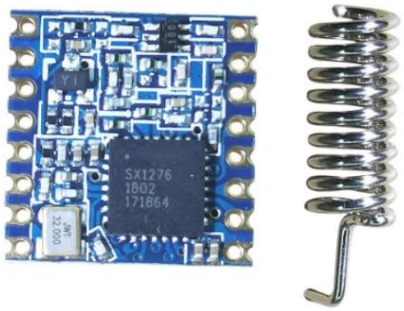



Figura 10: Chip SX1276


Figura 11: LoRa Gateway marca Dragino

### 3.2.2. WiFi

WiFi (Cisco, s.f) es una marca registrada de WiFi Alliance, y es una de las tecnologías inalámbricas más usadas en el mundo para la transmisión de datos y conexión con Internet. Lo que más destaca de esta tecnología es su ancho de banda y el alcance que tienen, por lo que hoy en día, son incontables la cantidad de dispositivos compatibles con el estándar de esta red, el 802.11. Sin embargo, tanto el alcance como el ancho de banda pueden verse afectados por algunos factores, tales como murallas, ondas de radio, condiciones climáticas, entre otros.

Entre sus principales características técnicas se encuentran:

- Alcance: Aproximadamente 100 Mts
- Ancho de Banda: Desde los 11 Mbps hasta los 9 Gbps dependiendo del estándar.
- Banda de Frecuencia: 2,4 GHz y 5 GHz.
- Protocolo: TCP/IP.

## 3.3. Sensores

Ya que en el proyecto se busca desarrollar un prototipo que sea capaz de detectar si hay un vehículo estacionado, será necesario utilizar sensores compatibles con los microcontroladores, para medir los cambios en el ambiente e interpretar si existe un cambio de estado en el slot de aparcamiento. En el mercado existen una gran variedad de sensores compatibles con los microcontroladores mencionados en la sección 3.1, pero para el proyecto, solo son necesarios aquellos que reúnan las siguientes características: bajo costo, bajo consumo energético, calidad aceptable, resistentes a condiciones ambientales de exterior entre otras (Barriga, J., Sulca, J., León, J., Ulloa, A., Portero, D., Andrade, R., Yoo, S., 2019). En esta sección se mencionan aquellos sensores que reúnan las características mencionadas anteriormente y sean potenciales candidatos por utilizar en el prototipo.

### 3.3.1. Sensores de Proximidad

Los sensores de proximidad se utilizan para medir distancias hacia obstáculos o para detectar obstáculos, existen desde aquellos que son analógicos como aquellos digitales. Aquellos que se podrían considerar en el proyecto dado su costo y stock en el mercado se mencionan a continuación:

#### 3.3.1.1. Sensor Ultrasónico HC-SR04

Los sensores ultrasónicos como el HC-SR04 (ELEC Freaks, s.f) (ver Figura 12) miden la distancia utilizando ondas ultrasónicas. Estas ondas ultrasónicas están por encima del rango de frecuencia audible del oído humano. Estos sensores emiten un pulso de energía ultrasónica, y al momento de ser reflejado por el obstáculo, el sensor recibe el eco producido mediante un receptor, finalmente con un sistema de tratamiento de señales que incorpora, calcula la distancia a la que se encuentra el objeto que detectó.


Figura 12: Sensor HC-SR04

Características:
- Voltaje de entrada: 5V
- Corriente estática: < 2mA
- Frecuencia de operación: 20 KHz - 400 KHz
- Angulo Sensor: < 15°
- Distancia de detección: 2 cm - 4.5 mts
- Precisión: 0.3 cm
- Precio: $2.000 CLP

### 3.3.1.2. Sensor ultrasónico Impermeable JSN-SR04T

El sensor JSN-SR04T (UNIT ELECTRONICS, s.f) (ver Figura 13) funciona de la misma forma en como lo hace el sensor de ultrasonido HC-SR04, sin embargo, este fue creado para ser utilizado en la intemperie, y como su nombre lo dice, es a prueba de agua. Este sensor se utiliza generalmente en autos para detectar obstáculos a la hora de estacionarse, para control de tráfico, entre otros.

Figura 13: Sensor JSN-SR04T

Características:
- Voltaje de entrada: 5V
- Corriente operación: 30 mA
- Frecuencia de operación: 40 KHz
- Angulo Sensor: < 50°
- Distancia de detección: 20 cm - 4.5 Mts
- Precisión: 0.3 cm
- A prueba de Agua
- Precio: $8.000 CLP

#### 3.3.1.3. Sensor de Proximidad Infrarrojo E18-D80NK

El sensor E18-D80NK (THINBOX, s.f) (ver Figura 14) se utiliza ampliamente para la detección de objetos. Para lograr esto, el sensor emite un haz de luz infrarroja, que impacta sobre el obstáculo, y el reflejo de este, es detectado por el sensor, el cual, como salida, entrega si se detectó o no el objeto. Este sensor, posee un potenciómetro que permite calibrar de forma manual la distancia, además al poseer una salida digital, no se requiere de la creación de algoritmos para su funcionamiento.

Figura 14: Sensor E18-D80NK

Características:
- Voltaje de entrada: 5V
- Corriente de operación: 25 mA - 100mA
- Distancia de detección: 3 cm a 80 cm
- Emisor de luz: Led infrarrojo
- Ángulo de detección: 15°
- Sensor fotoeléctrico infrarrojo
- Temperatura de trabajo: -25 a 70°C
- Precio: $9.500 CLP

### 3.3.2. Sensores de detección de metales o de campo magnético

Ya que los vehículos se pueden considerar como objetos abundantes en metal, existen sensores capaces de detectarlos si estos se encuentran próximos a ellos, esto debido a los cambios que pueden provocar los autos en el ambiente. Entre los sensores que cumplen con estas condiciones se encuentran los que se mencionan a continuación.

### 3.3.2.1. Sensor de Efecto Hall Lineal SS49E KY-024

Los sensores de efecto hall lineal como el KY-024 (Joy-IT, s.f) (ver Figura 15), se utilizan para detectar cambios en el campo magnético, por lo que, al entrar un objeto metálico o imantado en su campo de detección, afectará a la salida que entrega el sensor.

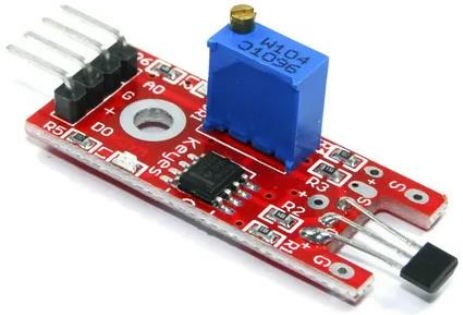


Figura 15: Sensor SS49E KY-024

Características:
- Voltaje de operación: 3.3 V a 5V
- Corriente de operación: 16 mA
- Sensor Hall: 49E
- Comparador: LM393
- Doble Salida: Digital On/Off y Análoga 0-5VDC
- Precio: $3.500 CLP

### 3.3.2.2. Sensor Magnetómetro HMC5883L

El sensor HMC5883L (Honeywell, s.f) (ver Figura 16) es un magnetómetro de 3 ejes, es decir, que mide la fuerza y dirección de un campo magnético en los ejes X, Y y Z. Generalmente se utiliza para crear brújulas o compases digitales. Este sensor, detecta el campo magnético que presenta mayor fuerza, el cual suele ser el de la tierra, pero si se acerca un objeto con mayor campo magnético, este se ve alterado, y los valores entregados por el sensor, sufren cambios.


Figura 16: Sensor HMC5883L

Características:
- Voltaje de operación: 3V a 5V
- Corriente de operación: 100uA
- Interfaz: I2C
- Precisión: 1° a 2° Sexagesimales
- Velocidad de Datos: 160 Hz
- Precio: Aproximadamente $5.000 CLP

### 3.3.3. Sensores de luz

Ya que el prototipo será utilizado en un ambiente al aire libre, la luz del sol puede ser un factor por considerar, ya que los autos al utilizar un slot de aparcamiento disminuyen la cantidad de luz que le llega a ese espacio. Entre los sensores que se pueden utilizarse encuentran los siguientes.

#### 3.3.3.1. Sensor de luz Grove 1.2

El sensor de luz Grove 1.2 (Seeed, s.f) (ver Figura 17) se utiliza para medir la luminosidad del ambiente, lo cual permite configurar que el microcontrolador al que está conectado realice ciertas acciones dependiendo del valor captado por el sensor. En cuanto al funcionamiento, la resistencia de la fotorresistencia disminuye cuando la luz aumenta y el chip que lleva montado el sensor (OpAmp dual LM358), produce un voltaje equivalente a la intensidad de la luz. La señal de salida es un valor analógico, el cual mientras más brillante sea la luz, mayor será este.


Figura 17: Sensor de luz Grove 1.2

Características:
- Voltaje de operación 3 V - 5 V
- Corriente de operación: 0.5 mA - 3 mA
- Tiempo de respuesta: 20 – 30 ms
- Longitud de onda tope: 540 nm
- Fotodiodo: GL5528
- Salida de valor analógico
- Precio: $2.500 CLP

### 3.3.3.2. Sensor de Luz de Color RGB ISL29125

El sensor de luz de color RGB ISL29125 (Rensesas, s.f) (ver Figura 18) es un sensor de luz de color rojo, verde y azul (RGB en Ingles) es de alta sensibilidad y baja potencia con una interfaz I2C (compatible con SMBus). Este sensor capta la intensidad de la luz en el espectro RGB y rechaza las fuentes de luz infrarrojas.


Figura 18: Sensor ISL29125

Características:
- Voltaje de operación 3.3 V
- Corriente de operación: 0.5 uA - 56 uA
- Salida de I2C
- Precio: $5.000 CLP

### 3.3.4. Selección de Sensores

Ya que se requiere detectar la presencia de un vehículo en un entorno exterior al aire libre, esto se puede lograr de forma precisa con solo dos tipos de sensores, uno de proximidad y uno magnético. El sensor de proximidad que se escogió es el ultrasónico impermeable, pues a diferencia del sensor de proximidad infrarrojo, no requiere calibración, y mediante programación, se escoge la distancia que se requiere para detectar un objeto, además si se compara con el de ultrasonido común y corriente, a pesar de tener un costo más alto, es resistente al agua, por lo que en días de lluvia no sufriría problemas electrónicos producto de las condiciones ambientales. En cuanto al sensor magnético, la mejor opción es el magnetómetro, pues detecta los cambios de fuerza magnética en el ambiente a una mayor distancia que los de efecto hall, en los cuales el metal debe estar tocando al sensor. En cuanto a los sensores de luz, no es una mala idea implementarlo en el prototipo, sin embargo, en horarios nocturnos, este quedaría inutilizable, pues la luminosidad de la plaza de estacionamiento no cambiará si un vehículo la utiliza.

## 3.4. Fuentes de Alimentación

Para que el prototipo pueda funcionar, es necesario que esté conectado a alguna fuente de energía, y los microcontroladores pueden suministrarse de esta por medio de diferentes fuentes de alimentación. Tanto el Heltec como el TTGO a, cuentan con las mismas entradas de energía, las cuales son por medio de micro USB (ver Figura 19) o por pines JST 2 (ver Figura 20). En esta sección se listan aquellas fuentes de energía que pueden ser una alternativa a usar en el proyecto.

Figura 19: Conector micro USB en Heltec Wifi LoRa 32 (v2)

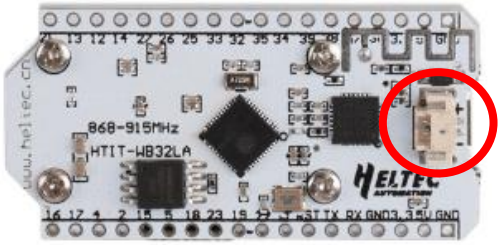



Figura 20: Conector JST en Heltec Wifi LoRa 32 (v2)

### 3.4.1. Baterías recargables

Hoy en día existen tres tipos de baterías recargables, las cuales son (Actitud Ecológica, 2020):

- Níquel Cadmio: son altamente contaminantes y poseen el efecto memoria, es decir, que reducen su capacidad debido cargas incompletas, y generalmente ocurre cuando se carga una batería que no se ha descargado completamente. Entre las ventajas con las que cuenta, es su vida útil, pues tienen alrededor de 1000 ciclos de carga.
- Níquel e hidruro metálico: Son muy similares a las mencionadas anteriormente, exceptuando la presencia de cadmio. Tienen una mayor capacidad que las de cadmio y sufren en menor medida el efecto memoria.
- Iones de litio: Este tipo de baterías, son las que más se utilizan actualmente, pues poseen una gran capacidad de carga además de no presentar el efecto memoria. Son utilizadas en laptops, smartphones, proyectos de IoT, etc. Además, no son amigables con el medio ambiente, ya que no producen contaminación.

Por lo presentado anteriormente, como alternativa a explorar entre sus variantes, serán aquellas que están compuestas por iones de litio, pues son amigables con el medio ambiente y tienen una gran capacidad de carga, lo que permitirá al prototipo tener una gran autonomía a la hora de ser desplegado en terreno.

### 3.4.1.1. Pilas de litio

Estas baterías, son las que comúnmente se conocen como pilas AA o se pueden encontrar en otros tamaños (ver Figura 21), dependiendo de la capacidad de esta. Pueden ser utilizadas en los microcontroladores mediante adaptadores, poseen una gran capacidad y tienen un precio reducido.


Figura 21: Pila de litio ICR16580

Características generales ICR 16850:
- Voltaje: 3.7 V - 4.2 V
- Capacidad: 1200 mAh - 3500 mAh
- Peso: 19 g - 42 g
- Ciclos de carga: 800 - 1200
- Tamaño: AA o mayor, dependiendo de la capacidad
- Precio: $3.000 CLP a $6.000 CLP c/u

### 3.4.1.2. Baterías de litio

Son similares a las pilas de litio, sin embargo, estas baterías ya cuentan con la salida de energía incorporada JST 2 pines para conectarse directamente a los microcontroladores (MCI electronics, s.f). Tienen forma de panel, por lo que son rectangulares, a diferencia de las pilas, que tienen forma cilíndrica (ver Figura 22).

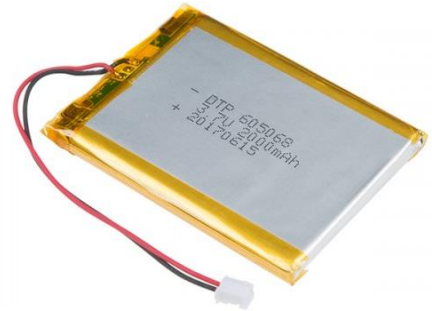


Figura 22: Batería de Litio

Características Generales de baterías de litio:
- Voltaje: 3.7 V - 4.2 V
- Capacidad: 100 mAh - 6000 mAh
- Peso: 3 g - 60 g
- Ciclos de carga: 800 - 1200
- Tamaño: 1.2×2.5×0.4cm - 6x9x0.9cm
- Precio: $3.000 CLP a $29.990 CLP c/u

#### 3.4.1.3. Baterías portátiles de litio o Powerbank

Las Baterías recargables de litio son las que se conocen como *Powerbank*. Estas se utilizan generalmente para cargar celulares inteligentes, audífonos inalámbricos, relojes inteligentes, etc. Sin embargo, también pueden ser utilizados en proyectos relacionados a IoT. Esto debido a que cuentan con una gran capacidad de carga, además de poseer salidas a 5V mediante interfaz USB, ideal para microcontroladores. Existen muchas marcas, pero se mencionará específicamente la Anker Powercore de 5000 mAh (ANKER, 2018) (ver Figura 23).


Figura 23: Powerbank Anker Powercore 5000

Características Powerbank Anker Powercore 5000:
- Voltaje: 5 V
- Amperaje: 2 A
- Capacidad: 5000 mAh
- Peso: 134 g
- Interfaz de entrada de carga: Micro USB
- Interfaz de salida de carga: USB-A
- Tamaño: 33x33x107mm
- Precio: $10.000 CLP

### 3.4.2. Paneles Solares

Los paneles solares permiten que un dispositivo sea energéticamente autónomo, pues al conectar el panel solar a la batería que está conectado el dispositivo, este se encargará de recargar la batería. Para esto se deben considerar factores energéticos, como el consumo del dispositivo y la carga que puede suministrar el panel o paneles a la batería, ya que, si el consumo es mayor que la carga, entonces, la batería se agotará con el paso del tiempo. Para interconectar la batería con el panel solar, se requiere de un controlador de carga de batería y un circuito regulador de voltaje (RANDOM NERD TUTORIALS, s.f) tal como se muestra en la Figura 24.

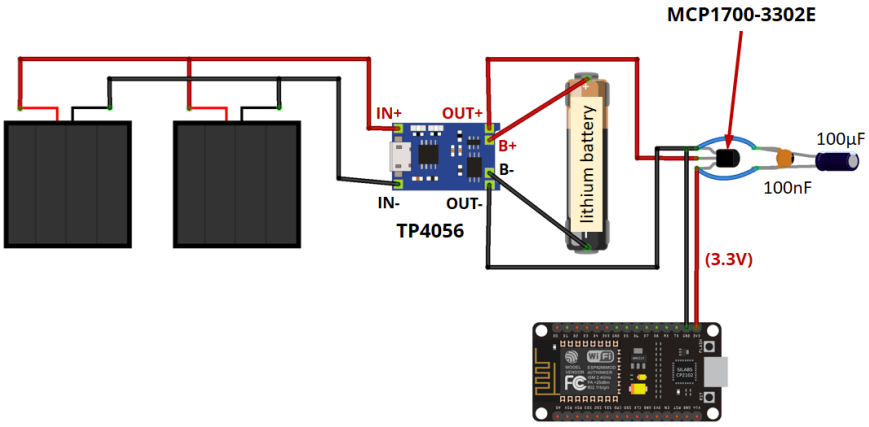



Figura 24: Diagrama de conexión panel solar a batería recargable

### 3.4.3. Selección de la fuente de energía

A partir de las opciones revisadas, para suministrar de energía al prototipo compuesto por el microcontrolador y los sensores, la mejor opción es la powerbank, debido al costo por capacidad que entrega, además de ofrecer cargas a 5V, ideal para el funcionamiento correcto del sensor de ultrasonido impermeable. En cuanto al uso del panel solar, no será considerado, debido a que el prototipo será ubicado en el suelo de la plaza del estacionamiento, requiriendo un cableado extra y una ubicación estratégica para el correcto funcionamiento de este.

## 3.5. Bases de Datos

Dentro del proyecto, es necesario almacenar los cambios de estado de las plazas de estacionamiento, ya que estos posteriormente deben ser utilizados para que los conductores puedan visualizar la información de manera gráfica en una aplicación móvil. Debido a esto, se requiere analizar las bases de datos que mejor se ajusten al proyecto, es decir, aquellas que presten mejores características al desarrollo de proyectos IoT o Smart City.

Hoy en día existen dos tipos de bases de datos, las relacionales (SQL, siglas en inglés) y las no relacionales (NOSQL, siglas en inglés), a continuación, se nombrarán sus características y aquella que mejor se ajusta al proyecto.

### 3.5.1. Bases de Datos relacionales (SQL)

Una base de datos relacional es una recopilación de elementos de datos con relaciones predefinidas entre ellos. Estos elementos se organizan como un conjunto de tablas con columnas y filas (ver Figura 25). Cada columna de una tabla guarda un determinado tipo de datos y un campo almacena el valor real de un atributo. Las filas de la tabla representan una recopilación de valores relacionados de un objeto o una entidad. Cada fila de una tabla podría marcarse con un identificador único denominado clave principal, mientras que filas de varias tablas pueden relacionarse con claves extranjeras (AWS Amazon, *¿Qué es una base de datos relacional?*, s.f ).

| ID Estacionamiento | ID Sensor | Ocupación |
|---|---|---|
| 1 | 1 | Ocupado |
| 1 | 2 | Desocupado |
| 2 | 1 | Desocupado |
| 2 | 2 | Ocupado |

Figura 25: Ejemplo tabla de Base de Datos Relacional

Entre las características que poseen las bases de datos relacionales se encuentran:

- Escalabilidad vertical, si aumenta la capacidad de los datos, será necesario aumentar la capacidad del hardware de la máquina en donde se encuentra alojada la base de datos. Por lo que este tipo de bases de datos se encuentra ligada al hardware para su procesamiento.
- El rendimiento de las consultas decae cuando estas requieren acceder a múltiples tablas relacionadas.
- Poco flexible, pues al ser estructurada, en caso de añadir un nuevo parámetro, es necesario editar la tabla en donde se agregará el nuevo parámetro, y por ende modificar la base de datos.
- Cumplimiento de las características ACID, las cuales son:
  - Atomicidad: la transacción debe ejecutarse correctamente como un todo, y si una parte de la transacción falla, queda invalidada.
  - Consistencia: los datos escritos en la base de datos como parte de la transacción cumplan todas las reglas definidas, así como las restricciones, incluidos los desencadenadores, las limitaciones y las cascadas.

- Aislamiento: cada transacción es independiente, por lo que no debe afectar a otra que se esté ejecutando.
- Durabilidad: todos los cambios realizados en la base de datos son permanentes luego de que la transacción se haya completado de forma correcta.

### 3.5.2. Bases de Datos no relacionales (NoSQL)

Las bases de datos no relacionales están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala. En una base de datos NoSQL, el registro de un libro generalmente se almacena como un documento JSON (ver Figura 26), donde los parámetros se almacenan como atributos en uno solo documento (AWS Amazon, *¿Qué es NoSQL*?, s.f).

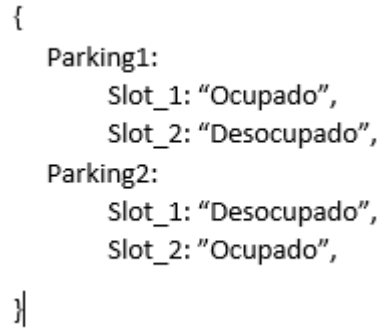



Figura 26: Ejemplo formato JSON

Entre las características que presentan este tipo de bases de datos se encuentra:
- Flexibilidad: las bases de datos NoSQL generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido e iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para datos semiestructurados y no estructurados.
- Escalabilidad Horizontal, es decir, si aumenta el volumen de datos, será necesario añadir otra máquina, y no realizar cambios caros en el hardware.
- Altamente funcional: las bases de datos NoSQL proporcionan API altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

### 3.5.3. Comparativa y selección de base de datos

A partir de lo expuesto anteriormente, se creó la Tabla 4, en donde se expondrán las principales diferencias entre los dos tipos de bases de datos, con el fin de seleccionar la mejor opción para el proyecto (Anderson B. & Nicholson B., 2020).

Tabla 4: Comparativa Base de Datos SQL vs NoSQL

| **Característica** | **SQL** | **NoSQL** |
|---|---|---|
| **ACID** | Si, en todos | Soportado en algunas |
| **Escalabilidad** | Vertical | Horizontal |
| **Flexibilidad** | No | Si |
| **Tolerancia a Fallos** | No | Si |
| **Rendimiento** | Disminuye al complejizar la base de datos | Alto |

Respecto a lo mostrado en la tabla anterior, se decidió utilizar una base de datos no relacional por lo siguiente:

- Las transacciones que se llevarán a cabo no requieren de características ACID, pues no afectarán a la base de datos, ya que cada estado de las plazas de estacionamiento es independiente entre sí.
- La escalabilidad para este proyecto debe ser horizontal, ya que en proyectos IoT, al aumentar el número de sensores, la carga se debe distribuir entre las máquinas que tienen alojada la base de datos.
- La flexibilidad debe estar presente, pues en caso de necesitar que los prototipos envíen otro tipo de dato, no será necesario migrar la base de datos a otra nueva.
- La tolerancia a fallos también debe estar presente, ya que, si un prototipo falla, la información de los demás debe poder visualizarse.
- El rendimiento no es un factor decisivo, ya que la base de datos al no presentar complejidad en caso de usar SQL, las consultas serán rápidas, al igual que en las NoSQL.

En la siguiente sección se detalla la base de datos NoSQL escogida para el desarrollo del proyecto.

## 3.6. Tecnologías para el Desarrollo Móvil

### 3.6.1. Firebase

Firebase es una plataforma digital que se utiliza para facilitar el desarrollo de aplicaciones web o móviles de una forma efectiva, rápida y sencilla. Su principal objetivo, es mejorar el rendimiento de las aplicaciones por medio de la implementación de diversas funcionalidades que van a hacer de la aplicación en cuestión, mucho más manejable, segura y de fácil acceso para los usuarios (Giraldo V., 2019). Incluso hoy en día, firebase es compatible con la tecnología IoT, ya que es posible utilizarla mediante microcontroladores como Arduino, ESP32, ESP8266, Raspberry, entre otras, mediante las librerías correspondientes a cada dispositivo.

Entre sus principales funciones se encuentran:

- Herramienta soportada en múltiples plataformas, como móvil (Android, IOS, Híbrida) o Web.
- Permite desarrollo gratuito, ya que para comenzar con el uso de firebase, no es necesario pagar inmediatamente, si no, que solicita un pago una vez que la cuota de almacenamiento y solicitudes dentro de la bases de datos supera cierto umbral.

Firebase ofrece dos tipos de bases de datos NoSQL, mencionadas a continuación:

- Realtime Database: es la base de datos NoSQL original de firebase, es eficiente y de baja latencia. Está orientada a las aplicaciones que requieran estados sincronizados entre los clientes en tiempo real. Los datos se organizan en formato JSON en una sola base de datos (Firebase, Firebase Realtime Database, 2020).
- Cloud Firestore: es una base de datos nueva de firebase orientada al desarrollo de aplicaciones móviles, reutiliza las mejores funciones de Realtime Database, mediante un modelo de datos más nuevo e intuitivo. En cuanto a escalamiento, se ajusta a un nivel más alto que Realtime Database. Los datos se almacenan mediante colecciones, donde cada una utiliza el formato JSON, para la anidación de estos (Firebase, Cloud Firestore, 2021).

Para escoger entre las dos opciones, se utilizaron las consideraciones claves otorgadas por firebase y mostradas en la Tabla 5 con el fin de compararlas (Firebase, Elige una base de datos: Cloud Firestore o Realtime Database, 2021).

Tabla 5: Consideraciones Cloud Firestore vs Realtime Database

| **Consideraciones** | **Cloud Firestore** | **Realtime Database** |
|---|---|---|
| **Función de Base de Datos** | Realizar Búsqueda, Transacciones y ordenamientos avanzados. | Sincronizar datos con consultas básicas. |
| **Operaciones con datos** | De cientos de GB o TB de datos que se leen con mucha frecuencia. | Algunos GB de datos o menos cambian con frecuencia. |
| **Modelo de datos** | Documentos organizados por colecciones. | Un árbol JSON simple. |
| **Disponibilidad** | Garantía de tiempo de actividad extremadamente alta del 99.999% | Garantía de tiempo de actividad de al menos 99.95%. |
| **Consultas sin conexión en datos locales** | Con frecuencia o rara vez. | rara vez. |
| **Cantidad de instancias de base de datos** | Más de una base de datos. | Una base de datos. |

Por lo tanto, respecto a lo mostrado en la Tabla 5 y a las necesidades del proyecto, la mejor opción es Realtime Database por las siguientes razones:

- Se realizarán consultas básicas, en donde se extraerán de esta solo el estado de las plazas de estacionamiento.
- Los datos no utilizarán más de 1 GB en la base de datos.

- El modelo de datos que se requiere es el árbol JSON, ya que los datos no requieren estar separados en colecciones.
- No se requiere una garantía de tiempo de actividad del 99.999%.
- Rara vez se harán consultas sin conexión en datos locales.
- Solo se requerirá una instancia de base de datos.

El servicio Realtime Database presenta dos planes, uno gratuito y uno de pago por uso, las diferencias serán se visualizan en la Tabla 6.

Tabla 6: Comparación plan gratuito vs de pago de Realtime Database

| | **Plan Gratuito** | **Plan de pago por uso** |
|---|---|---|
| **Conexiones simultáneas** | 100 | 200.000 por base de datos |
| **GB almacenados** | 1 GB | $3.500 CLP por GB |
| **GB descargado** | 10 GB al mes | $700 CLP por GB |
| **Varias bases de datos por proyecto** | No | Si |

### 3.6.2. Desarrollo de Aplicación Móvil

Para el desarrollo de una aplicación móvil, se pueden escoger dos caminos, mediante el desarrollo de aplicaciones nativas o desarrollo de aplicaciones híbridas (Gorka R, 2019).

Las aplicaciones nativas son aquellas que se desarrollan para ser usadas exclusivamente en Android o iOS, cada sistema operativo utiliza su propio lenguaje de programación, en Android se utiliza Kotlin o Java y en IOS se usa Objective C o Swift. Tanto Google como Apple ofrecen a los desarrolladores herramientas de desarrollo, elementos de interfaz y SDK estandarizado; Android Studio en caso de Android y XCode en caso de IOS.

Las aplicaciones híbridas se desarrollan generalmente en HTML5, CSS y Javascript, mediante algún *framework* que permite emular la vista web a una vista móvil. Este tipo de desarrollo permite mantener solo una base de código, ya que será este el que se utilice para correr la aplicación en un entorno Android o iOS.

Ambas propuestas tienen sus ventajas y desventajas, es por esto que será necesario realizar un cuadro comparativo, con el fin de tener una idea clara de cuál será la mejor opción a utilizar en el proyecto. En la Tabla 7 se describen algunos criterios a tener en cuenta al escoger entre aplicaciones nativas o híbridas.

Tabla 7: Comparación aplicación Nativa vs Hibrida

| Aplicación | Aplicación Nativa | Aplicación Híbrida |
|---|---|---|
| **Tiempo de desarrollo** | Mayor, ya que se deben crear dos bases de código, además de requerir conocimientos específicos de cada plataforma. | Menor, ya que se debe crear una sola base de código para las versiones de la aplicación. |
| **Experiencia de uso** | Similar a la nativa, debido a las adaptaciones que se han llevado a cabo en el último tiempo. Se recomiendan aplicaciones que no requieran un uso excesivo de los componentes hardware de los dispositivos. | Es fluida, debido a que son desarrolladas para utilizarse bajo los componentes del sistema operativo de cada dispositivo. Aquellas aplicaciones que requieran carga gráfica, uso intensivo de CPU, o componentes hardware, tendrán ventajas sobre aplicaciones híbridas. |
| **reutilización de Código** | Se reutiliza la misma base de código para distintas plataformas. | Se crean funciones duplicadas que realizan la misma función. |
| **Rendimiento** | Rendimiento menor a la nativa, ya que corren bajo entornos virtuales, sin aprovechar al máximo las capacidades de los dispositivos. | Rendimiento máximo, ya que utiliza al máximo las capacidades de los dispositivos bajo los cuales está corriendo la aplicación. |
| **Coste de desarrollo** | Inferior al nativo, ya que se utiliza solo una base de código para cada plataforma. | Es mayor, ya que es proporcional al número de plataformas |

Por lo visto en los criterios de la tabla anterior, la opción que más se acomoda a las necesidades del proyecto, es la aplicación híbrida, ya que, en un tiempo reducido, permitirá generar una aplicación que pueda correr tanto en Android como en IOS. Además, la aplicación no requerirá de un uso excesivo de las capacidades hardware de los dispositivos, ya que solo se requiere que esta sea: simple, eficiente y sencilla a los ojos de los conductores.

Ya que se llevará a cabo el desarrollo de una aplicación híbrida, existen múltiples *framework* como alternativa para ser utilizados. A continuación, se mencionan las opciones más famosas en el último tiempo y que tienen un mayor apoyo por parte de la comunidad de desarrolladores (Naharro A., 2019):

- **Phonegap/Apache Cordoba:** Creado en 2009, es un *framework* que permite desarrollar aplicaciones híbridas mediante tecnología web HTML5, CSS3 y JavaScript. Utiliza una API que permite acceder a elementos de hardware del sistema. Permite integrarse con otros *frameworks* como por ejemplo jQuery Mobile. Apache Cordova es la versión de código abierto de PhoneGap, la principal diferencia entre estos dos

*frameworks* es que PhoneGap tiene acceso al servicio de compilación en la nube Adobe Creative Cloud permitiendo compilar la aplicación para el sistema operativo deseado e independiente del sistema operativo donde se desarrolle. La principal diferencia de este *framework* es que se encarga de la integración dentro del sistema operativo móvil y de dar acceso a los servicios de éste, por sobre dar prioridad a la interfaz de la aplicación.

- **jQuery Mobile**: Creado en 2010, es un *framework* basado en el lenguaje de programación Javascript. Su curva de aprendizaje es muy baja, sobre todo si ya se dispone de conocimientos de jQuery y JavaScript. Se ha quedado atrás en cuanto a potencia y diseño si se compara con el resto de nuevos *frameworks* que hay a disposición, pero igualmente jQuery Mobile puede utilizarse en paralelo con otros *frameworks* de diseño como son Bootstrap, Materialize, etc.

- **Ionic:** Creado en 2013, es uno de los *frameworks* más famosos para el desarrollo de aplicaciones híbridas. En sus inicios Ionic utilizaba el *framework* AngularJS y PhoneGap para la parte de integración con plataformas móviles. Sin embargo, en sus últimas versiones ha incorporado nuevos *frameworks* Front-End con los que poder desarrollar. Actualmente permite su desarrollo con: Angular, React, Vue.JS. Ionic integra una capa de diseño con estilos css y recursos como iconos, etc. Estos estilos pueden ser adaptados a los estándares de diseño de las plataformas Android y iOS.

- **React Native:** Creado en 2015, es un *framework* para crear aplicaciones híbridas que está basado en JavaScript y en un conjunto de componentes del *framework* ReactJS. Al desarrollar una aplicación con este *framework*, lo que se obtiene es una aplicación real nativa, pues entrega un ejecutable .apk para Android y un ejecutable .ipa para IOS. Para lograr esto utiliza el mismo modelo de construcción de bloques de UI (componentes visuales con los que interacciona el usuario) que Android e IOS, pero gestionando la interacción entre las capacidades de Javascript y React. Actualmente también es uno de los más famosos, y cuenta con una gran comunidad de desarrollo. Forma parte de aplicaciones muy potentes y conocidas, como Facebook, Pinterest, Skype o Instagram.

- **Framework 7:** Creado en 2017, no es uno de los *frameworks* más conocidos, pero tiene la ventaja de que es completamente independiente y por lo tanto no tiene dependencias externas en *frameworks* como Angular o React. En cuanto a diseño, es similar a Ionic, aportando un diseño ajustado a los estándares de diseño de Android e iOS dependiendo de la plataforma en la que se ejecute. Este *framework* tiene una curva de aprendizaje más baja que React Native e Ionic gracias a que se pueden realizar aplicaciones híbridas utilizando simplemente HTML5, CSS3 y JavaScript. No obstante, permite su desarrollo también con Vue.JS o React si nos interesa. Para la emulación y ejecución requiere la combinación con Cordova o PhoneGap.

- **NativeScript:** Creado en 2015, permite el desarrollo de aplicaciones nativas mediante JavaScript y TypeScript. También permite el desarrollo mediante el *framework* Angular, y en su última versión se ha incorporado también la posibilidad de desarrollar mediante el *framework* Vue.js. NativeScript genera aplicaciones verdaderamente

nativas, utilizando las mismas APIS que se tienen disponibles desde Android Studio y XCode. Incluso permite el acceso a bibliotecas nativas de terceros, aportando así mayores capacidades respecto al rendimiento y capacidades de las aplicaciones. Su mayor inconveniente es que el desarrollo de las interfaces de usuario es más complejo ya que es XML y no HTML, así como las abstracciones necesarias para acceder al código nativo de manera independiente del sistema.

A partir de un análisis comparativo respecto a los *frameworks* mencionados anteriormente, se utilizara React Native, debido a que existe experiencia previa respecto al uso del *framework* ReactJS, por lo tanto la curva de aprendizaje será más corta que en los demás *framework* y ofrece un mayor rendimiento a las demás, debido a que genera aplicaciones nativas Además presenta una comunidad numerosa de desarrolladores, permitiendo encontrar soluciones más rápidas a posibles problemas de desarrollo que se presentarán durante la programación de la aplicación.

# 4. ARQUITECTURA Y SOLUCIÓN

En este capítulo se describe el problema que este proyecto busca resolver y se detalla la solución propuesta, abarcando el desarrollo del prototipo necesario para la toma de datos y decisión en las plazas de estacionamiento, y las herramientas de ingeniería de software para el desarrollo de la aplicación móvil.

## 4.1. Descripción del problema

A nivel de la Universidad, no se han llevado a cabo soluciones capaces de informar a los conductores que transitan por las dependencias, acerca de las plazas vacías ubicadas en los distintos lugares establecidos para aparcar vehículos. Por ende, los conductores deben recorrer la Universidad buscando una plaza vacía para aparcar su automóvil, malgastando recursos como lo son el tiempo y el combustible.

Si se escala esta problemática a nivel de la ciudad, el panorama para los conductores es el mismo, no manejan información sobre los lugares aptos para aparcar, tanto privados como municipales, llevándolos a recorrer las calles en busca de un estacionamiento. Cabe destacar que en ocasiones los conductores se estacionan en lugares no aptos para aparcar vehículos, y como consecuencia de esa acción, deben enfrentar problemas legales, en caso de que personal de carabineros o municipal visualicen la infracción que están cometiendo.

## 4.2. Solución Propuesta

Se propone implementar el desarrollo de un prototipo IoT utilizando dos módulos Heltec WiFi LoRa (v2), el primero es aquel que registra si un vehículo está utilizando una plaza de estacionamiento, es decir, un emisor, por lo que se le conectó a un sensor de ultrasonido resistente al agua para medir la proximidad hacia un vehículo y a un sensor magnetómetro, para verificar que el objeto que utiliza el espacio de estacionamiento es un automóvil. Además, el emisor debe ser energéticamente autónomo, por lo que se conectó a una powerbank recargable, que será capaz de suministrar la corriente necesaria para que el prototipo funcione durante un tiempo prolongado.

Para enviar la información al receptor, ambos montan una red LoRa local, por lo que una vez que el receptor reciba los datos mediante LoRa, este los subirá a la base de datos en tiempo real de FireBase por medio de WiFi.

Para visualizar los estados del estacionamiento en tiempo real, se creó una aplicación híbrida en React Native, que permite utilizarla tanto en Android como iOS. La Figura 27 muestra el flujo que seguirán los cambios de estado en el sistema. Cabe destacar que el prototipo y la aplicación sufrieron mejoras una vez que se realizaron las pruebas en terreno correspondientes.

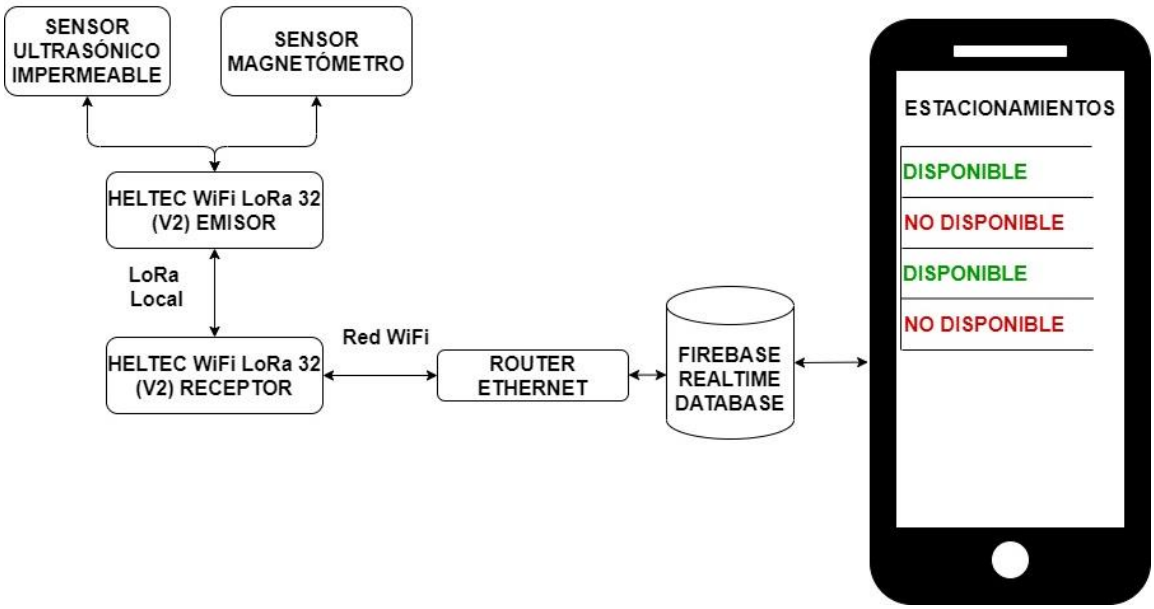



Figura 27: Flujo de datos del sistema

## 4.3. Metodología

En los últimos años, cada proyecto de investigación o proyecto de desarrollo de alguna innovación lleva a cabo la implementación de los niveles de madurez de la tecnología (TRL: Technological Readiness Level, siglas en inglés) como una guía para definir las fases que este tendrá. Los niveles de madurez tecnológica son un sistema de medición para evaluar una tecnología en particular, surge en la NASA para ser aplicado en proyectos aeronáuticos y aeroespaciales, pero con el paso de los años, se generalizó para ser aplicado en cualquier proyecto. Cada proyecto se evalúa contra los parámetros para cada nivel de tecnología y luego se le asigna una calificación TRL basada en el progreso del proyecto. Hay nueve niveles de preparación tecnológica. TRL 1 es el más bajo y TRL 9 es el más alto, la descripción de cada nivel se puede visualizar en la Tabla 8 (Ibañez, J, s.f).

Tabla 8: Niveles de maduración tecnológica de un proyecto según la NASA

| Nivel | Descripción |
|---|---|
| TRL 1 | Principios básicos observados y reportados |
| TRL 2 | Concepto y/o aplicación tecnológica formulada. |
| TRL 3 | Función crítica analítica y experimental y/o prueba de concepto característica. |
| TRL 4 | Validación de componente y/o disposición de estos en entorno de laboratorio. |
| TRL 5 | Validación de componente y/o disposición de estos en un entorno relevante. |
| TRL 6 | Modelo de sistema o subsistema o demostración de prototipo en un entorno relevante |
| TRL 7 | Demostración de sistema o prototipo en un entorno real. |
| TRL 8 | Sistema completo y certificado a través de pruebas y demostraciones. |
| TRL 9 | Sistema probado con éxito en entorno real |

Los TRL se pueden dividir en entornos de validación, en los cuales los niveles 1-4 pertenecen al entorno de laboratorio; los niveles 5-6 en entornos de simulación, con características similares al real y los niveles 7-9 pertenecen a la validación del proyecto en un entorno real. En el caso de este proyecto, se pretende completar la validación en entorno de laboratorio, por lo cual se requerirá de investigación previa y desarrollo de la solución, validando que los componentes tecnológicos como los son el hardware y software funcionen de manera correcta en las pruebas experimentales correspondientes.

En cuanto a la definición del modelo de desarrollo implementado en el proyecto, primero se realizó una búsqueda de modelos con el fin de utilizar aquella que se acomode a las necesidades requeridas. Los modelos de desarrollo se dividen en dos grandes grupos, las metodologías clásicas y las metodologías ágiles (Maida, EG, Pacienzia, J., 2015).

Las metodologías clásicas, se centran en mantener una documentación exhaustiva de todo el proyecto, la planificación y control de este, además de especificar de manera precisa los requisitos, modelado y plan de trabajo, todo esto, en la fase inicial del desarrollo. Entre las metodologías más comunes se encuentran:

- Cascada: Fue el primer modelo en originarse, y es denominado de esta forma, por la posición en las fases de desarrollo de esta, que parecieran caer en cascada hacia las siguientes fases. En esta metodología el proceso de desarrollo del proyecto se ordena de tal forma que el inicio de cada etapa debe esperar la finalización de la etapa anterior. Al final de cada etapa se debe llevar a cabo una revisión final, que se encarga de determinar si el proyecto está en condiciones de avanzar a la siguiente fase.

- Prototipo: Un prototipo es una versión preliminar de un sistema de información con fines de demostración o evaluación. El prototipo de requerimientos es la creación de una implementación parcial de un sistema, para el propósito explícito de aprender sobre los requerimientos del sistema. Un prototipo es construido de una manera rápida tal como sea posible.

- Espiral: Toma las ventajas de los modelos anteriores, añadiéndoles el concepto de análisis de riesgo. Está compuesto por 4 fases: Planificación, análisis de riesgo, ingeniería y Evaluación del Cliente. Cada fase se repite en cada iteración, por lo que, en cada una, el proyecto gana madurez hasta que en una iteración se logre el objetivo deseado.

- Incremental: En este modelo, se lleva a cabo el desarrollo en etapas incrementales y en cada una se agrega una funcionalidad. Estas etapas, consisten en requerimientos, diseño, codificación, pruebas y entrega. Por lo que permite entregar un producto más rápido en comparación al cascada. El proyecto recibe mejoras constantes en cada iteración, permitiendo así, un mayor entendimiento del problema a resolver y la solución mediante los refinamientos sucesivos.

Retomando las metodologías de desarrollo ágiles, estas se crearon tras una reunión celebrada el año 2001 en E.E.U.U, por expertos en el desarrollo de software y de

metodologías de desarrollo. Su objetivo fue ofrecer una alternativa a las metodologías clásicas, ya que estas son rígidas y regidas por la extensa documentación que requiere cada actividad. Esta alternativa se caracteriza por ser permitir un desarrollo de software más rápido y capaz de responder a los cambios que puedan surgir a lo largo del proyecto. Entre los tipos de metodologías más comunes se encuentran:

- Programación Extrema: Esta metodología se caracteriza por su énfasis en la adaptabilidad, pues considera que los cambios de requisitos son un aspecto natural e inevitable en el desarrollo de proyectos. El énfasis en la adaptabilidad proviene del objetivo que tienen, el cual es entregar un producto con los requisitos exactos que el cliente requiere. Por lo que llevan a cabo el desarrollo mediante iteraciones en las cuales el cliente pasa a ser parte del desarrollo, con el fin de entregar retroalimentación durante todo el proyecto.

- Scrum: Tiene un enfoque iterativo e incremental, que le permite optimizar la predictibilidad y el control de riesgo. Las iteración se llevan a cabo mediando bloques temporales cortos, o más bien llamados sprint. Es una metodología que se puede adaptar a cualquier tipo de proyecto, no necesariamente de desarrollo de software y es posible producir resultados en cortos periodos de tiempo.

- Proceso Unificado de Desarrollo (RUP siglas en inglés): Se basa en componentes e interfaces bien definidas, además junto con el Lenguaje Unificado de Modelado (UML siglas en inglés), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de software, áreas de aplicación, tipos de organización o diferentes tamaños de proyecto. Se lleva a cabo mediante iteraciones e incrementos, lo que permite refinar los requisitos que el proyecto debe cumplir.

En este proyecto, se implementó la metodología incremental, pues además de contar con experiencia previa en la utilización de esta, es necesario llevar a cabo una documentación constante de lo que se está realizando y gracias a los incrementos e iteraciones, será posible refinar los prototipos de hardware y software que se desarrollaran. Las etapas por las que está compuesto el proyecto son las siguientes, Adquisición de conocimiento, Búsqueda y selección de la tecnología IoT y de desarrollo, Desarrollo de prototipo hardware y software, Pruebas, Validación y Mejoras. Cabe destacar que, en el caso de las dos primeras etapas, estas solo requirieron de una sola iteración, ya que posteriormente se realiza el proyecto en base a estas.

En cuanto a las etapas mencionadas, estas consistieron en lo siguiente:

- Adquisición de conocimiento: Consta de realizar una revisión sistemática que responde a las siguientes preguntas: ¿Cuáles son los sensores IoT más usados en la detección de automóviles, en los proyectos de Smart Parking? y ¿Cuáles son las mejores fuentes de energía para sustentar proyectos IoT como el Smart Parking? Además de realizar una investigación respecto al estado del arte de los proyectos de Smart Parking.
- Búsqueda y selección de la tecnología IoT y de desarrollo: Se buscaron las tecnologías IoT disponibles en el mercado y se escogieron aquellas que cumplen en mejor medida, los objetivos de este proyecto. Además, se buscaron las tecnologías de desarrollo de software que se podían utilizar en el proyecto y su posterior selección.
- Desarrollo de prototipo hardware y software: Consiste en el desarrollo del prototipo IoT que captura los datos en una plaza de estacionamiento y el desarrollo de la aplicación móvil en la que se visualiza la información.
- Pruebas: Esta etapa se hizo en conjunto con la anterior, ya que a medida que se desarrollaba el prototipo IoT, se hacían pruebas en las que se median la respuesta de los sensores ante cambios en el estacionamiento. Respecto a las pruebas en la aplicación móvil, estas se llevaron a cabo una vez que el prototipo estuvo funcional, y era capaz de medir los cambios en el estacionamiento.
- Validación: La validación de la solución del proyecto se realizó de manera teórica, por medio de la revisión de documentos de investigación que hayan realizado un trabajo similar al desarrollado en este.
- Mejoras: Se comenta respecto a los resultados obtenidos y aquellas mejoras que se podrían implementar en este proyecto con el fin de refinarlo y optimizarlo.

## 4.4. Especificación de requisitos

Debido a que este proyecto involucra el desarrollo de un prototipo en el ámbito de hardware y software, es necesario definir los requisitos que estos deben cumplir. Por lo que a continuación se definirán los requerimientos funcionales y no funcionales para el prototipo IoT (Hardware) y para la aplicación móvil (software).

### 4.4.1. Requerimientos funcionales

#### 4.4.1.1. Requerimientos del prototipo IoT

- RF1: El prototipo IoT emisor debe reconocer si hay un vehículo estacionado.
- RF2: El prototipo IoT emisor debe conectarse a una powerbank como fuente de energía.
- RF3: El prototipo IoT emisor debe enviar los cambios de estado ocupado-desocupado del estacionamiento mediante la red LoRa al prototipo receptor.
- RF4: El prototipo IoT receptor debe recibir los datos enviados por el prototipo emisor mediante red LoRa.
- RF5: El prototipo IoT receptor debe conectarse a red WiFi y subir los datos recibidos a la base de datos.

- RF6: Ambos prototipos IoT deben reconectarse automáticamente una vez perdida la conexión mediante red LoRa.
- RF7: El prototipo IoT receptor debe reconectarse automáticamente a la red WiFi en caso de perder conexión con el punto de acceso al que está conectado.

#### 4.4.1.2. Requerimientos de la aplicación móvil

- RF8: La aplicación móvil debe mostrar el estado de las plazas de los estacionamientos en tiempo real.
- RF9: La aplicación móvil debe permitir a los usuarios registrarse.
- RF10: La aplicación móvil debe permitir iniciar sesión a los usuarios.

### 4.4.2. Requerimientos no funcionales

#### 4.4.2.1. Requerimientos del prototipo IoT

- RNF1: El prototipo emisor debe enviar datos de manera óptima, haciendo uso eficiente de la fuente de energía.

#### 4.4.2.2. Requerimientos de la aplicación móvil

- RNF2: La aplicación móvil debe ser utilizable en los sistemas Android e iOS más recientes.
- RNF3: La aplicación móvil debe ser intuitiva y eficiente ante el uso de conductores.

## 4.5. Casos de uso

A partir de los requisitos mencionados y expuestos en la sección anterior, se describen a continuación los casos de usos detectados que involucran la interacción con el usuario, la cual solo se lleva a cabo a nivel software, es decir, mediante la aplicación móvil. Los casos de uso detectados son 4: Ver las plazas disponibles de un estacionamiento, Seleccionar el estacionamiento a visualizar, Registrar Usuario, Iniciar Sesión y Cerrar Sesión.

### 4.5.1. Ver las plazas disponibles de los estacionamientos

#### Descripción

Tabla 9: Descripción caso de uso "Ver las plazas disponibles de los estacionamientos"

| **Caso de uso** | **Ver las plazas disponibles de los estacionamientos.** |
|---|---|
| Actores: | Usuario |
| Precondición | Usuario tiene la aplicación instalada en su smartphone, cuenta con conexión a internet, el prototipo envía los cambios en el estacionamiento. |
| Resumen | El usuario accede a la aplicación, selecciona el estacionamiento a visualizar y posteriormente navega sobre el estacionamiento seleccionado para ver las plazas disponibles. |
| Tipo | Primario |

#### Flujo Normal de Eventos

Tabla 10: Flujo normal de eventos caso de uso "Ver plazas disponibles de los estacionamientos"

| **Actor** | **Sistema** |
|---|---|
| Usuario accede a la aplicación | |
| | Muestra la sección estacionamientos. |
| Usuario selecciona el estacionamiento a visualizar y lo oprime. | |
| | Muestra la información de las plazas del estacionamiento |
| Usuario navega por el estacionamiento seleccionado. | |

D**iagrama de secuencia**

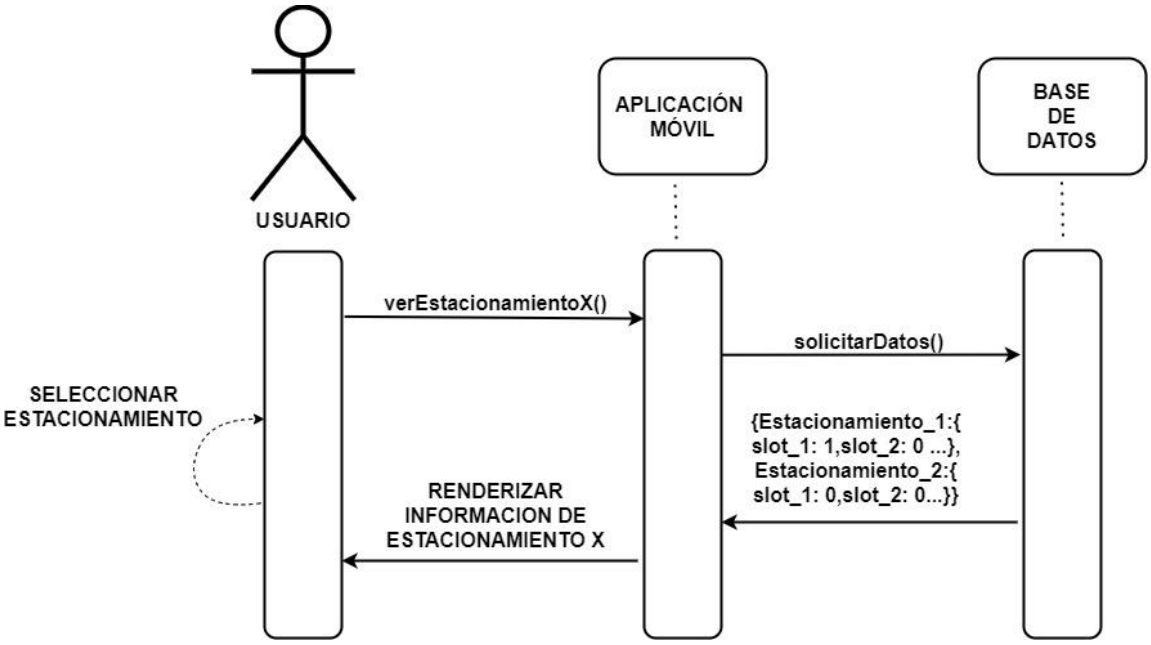



Figura 28: Diagrama de secuencia caso de uso "Ver las plazas de los estacionamientos disponibles"

### 4.5.2. Registrar usuario

**Descripción**

Tabla 11: Descripción caso de uso "Registrar usuario"

| **Caso de uso** | **Registrar Usuario** |
|---|---|
| Actores: | Usuario |
| Precondición | Usuario tiene la aplicación instalada en su smartphone, cuenta con conexión a internet y no tiene una cuenta registrada. |
| Resumen | El usuario accede a la aplicación, selecciona la ventana usuario, aprieta el botón ver cuenta, selecciona registrarse, rellena el formulario de registro, envía la información y termina el registro. |
| Tipo | Opcional |

**Flujo Normal de Eventos**

Tabla 12: Flujo normal de eventos caso de uso “Registrar usuario”

| **Actor** | **Sistema** |
|---|---|
| Usuario accede a la aplicación | |
| | Muestra las secciones Estacionamientos y Cuenta. |
| Usuario oprime la sección Cuenta. | |
| | Muestra Ventana Informativa de la aplicación y botón Ver Cuenta. |
| Usuario oprime botón ver cuenta. | |
| | Muestra el formulario de inicio de sesión y la opción de registrarse. |
| Usuario oprime la opción registrarse. | |
| | Muestra el formulario de registro. |
| Usuario rellena el formulario de registro y oprime registrarse. | |
| | Valida que los datos estén correctos, registra al nuevo usuario y lo redirige al perfil de usuario. |

**Diagrama de secuencia**

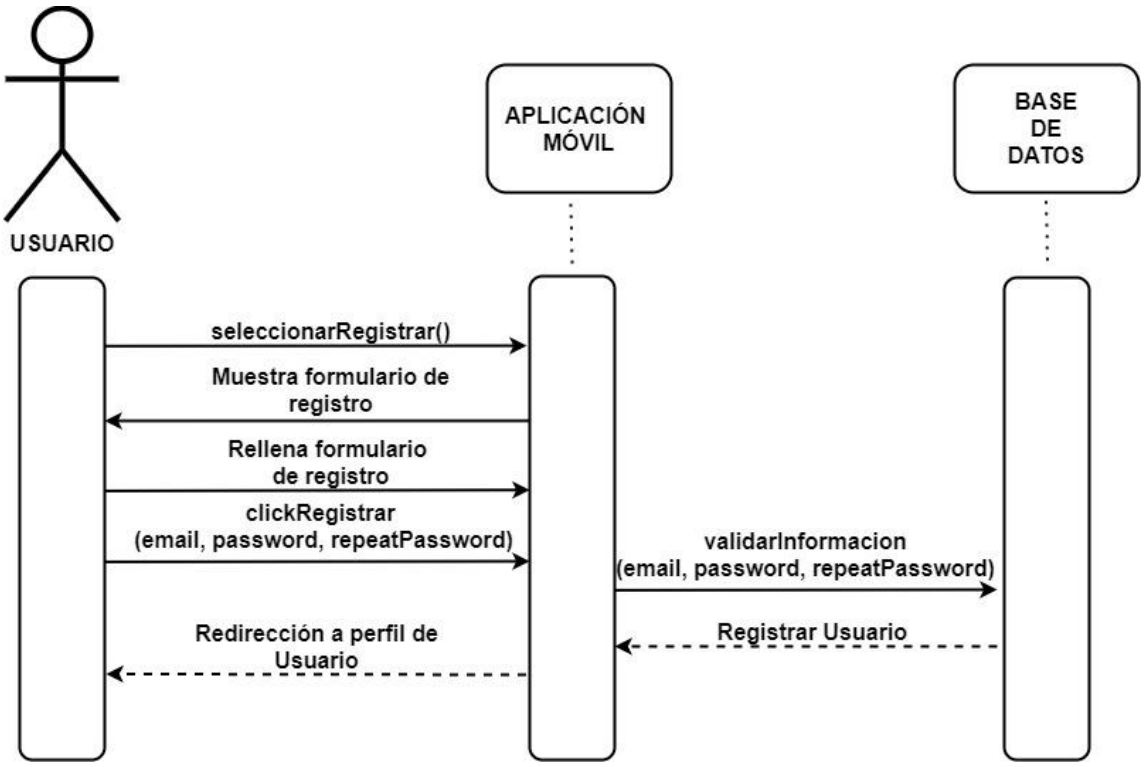



Figura 29: Diagrama de secuencia caso de uso “Registrar usuario”

### 4.5.3. Iniciar sesión

**Descripción**

Tabla 13: Descripción caso de uso “Iniciar sesión”

| **Caso de uso** | **Iniciar sesión.** |
|---|---|
| Actores | Usuario. |
| Precondición | Usuario tiene la aplicación instalada en su smartphone, cuenta con conexión a internet y con una cuenta previamente creada. |
| Resumen | El usuario accede a la aplicación, selecciona la ventana usuario, aprieta el botón ver cuenta, rellena los campos de inicio de sesión y el sistema muestra la ventana de usuario logueado. |
| Tipo | Opcional. |

**Flujo normal de eventos**

Tabla 14: Flujo normal de eventos caso de uso “Iniciar sesión”

| **Actor** | **Sistema** |
|---|---|
| Usuario accede a la aplicación. | |
| | Muestra las secciones Estacionamientos y Cuenta. |
| Usuario oprime la sección Cuenta. | |
| | Muestra Ventana Informativa de la aplicación y botón Ver Cuenta. |
| Usuario oprime botón ver cuenta. | |
| | Muestra el formulario de inicio de sesión y la opción de registrarse. |
| Usuario rellena el formulario de inicio de sesión y oprime iniciar sesión. | |
| | Valida los datos y lo redirige al perfil del usuario. |

**Diagrama de secuencia**

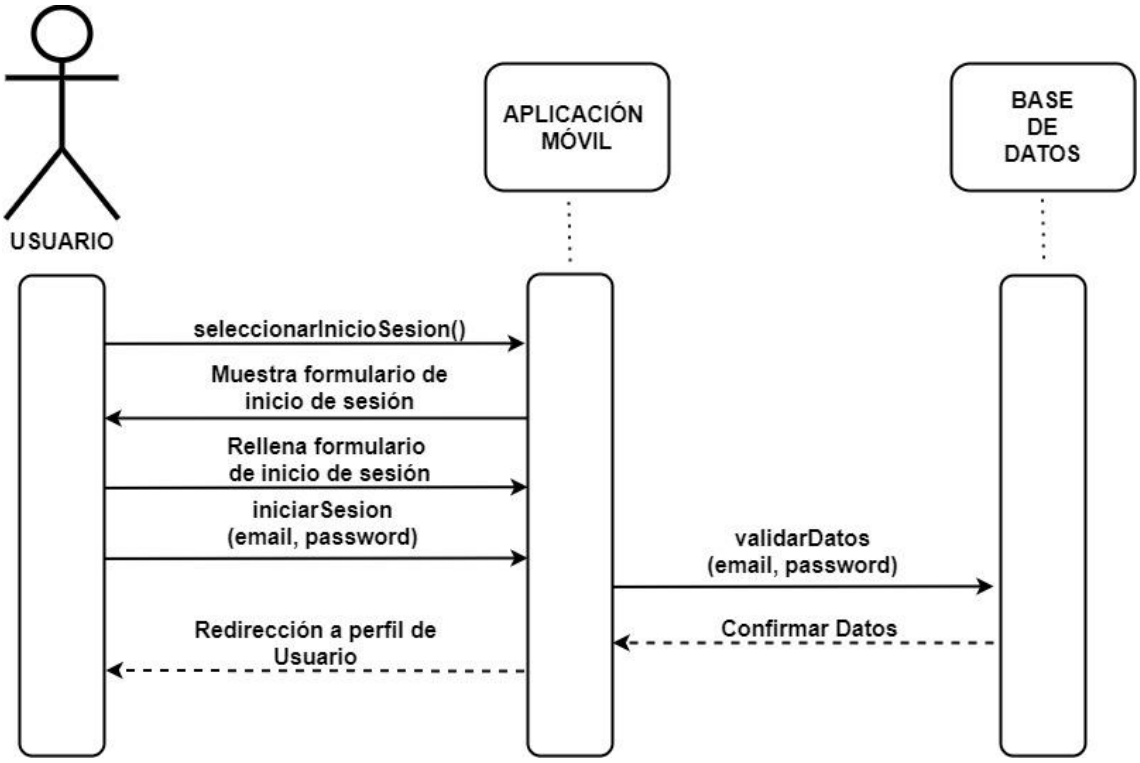



Figura 30: Diagrama de secuencia caso de uso "Iniciar sesión"

### 4.5.4. Cerrar sesión

**Descripción**

Tabla 15: Descripción caso de uso "Cerrar sesión"

| **Caso de uso** | **Cerrar sesión.** |
|---|---|
| Actores | Usuario. |
| Precondición | Usuario tiene la aplicación instalada en su smartphone, cuenta con conexión a internet y con una sesión iniciada. |
| Resumen | El usuario accede a la aplicación, selecciona la ventana usuario y aprieta la opción cerrar sesión. |
| Tipo | Opcional. |

**Flujo normal de eventos**

Tabla 16: Flujo normal de eventos caso de uso “Cerrar sesión”

| **Actor** | **Sistema** |
|---|---|
| Usuario accede a la aplicación. | |
| | Muestra las secciones Estacionamientos y Cuenta. |
| Usuario oprime la sección Cuenta. | |
| | Muestra el perfil del usuario y la opción de cerrar sesión. |
| Usuario oprime botón cerrar sesión. | |
| | Cierra la sesión de la aplicación y lo redirige a la ventana informativa de la app. |

**Diagrama de secuencia**

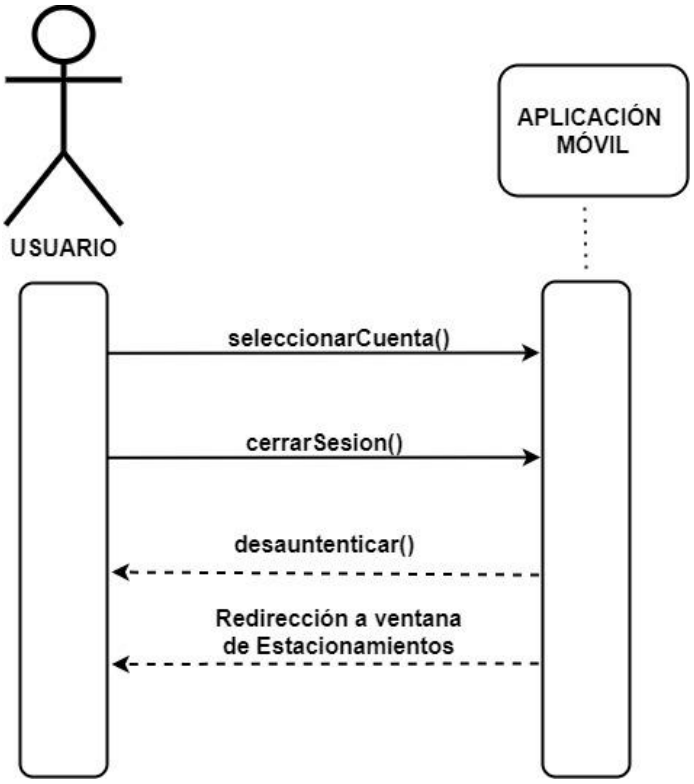



Figura 31: Diagrama de secuencia caso de uso “Cerrar sesión”

## 4.6. Modelo de proceso

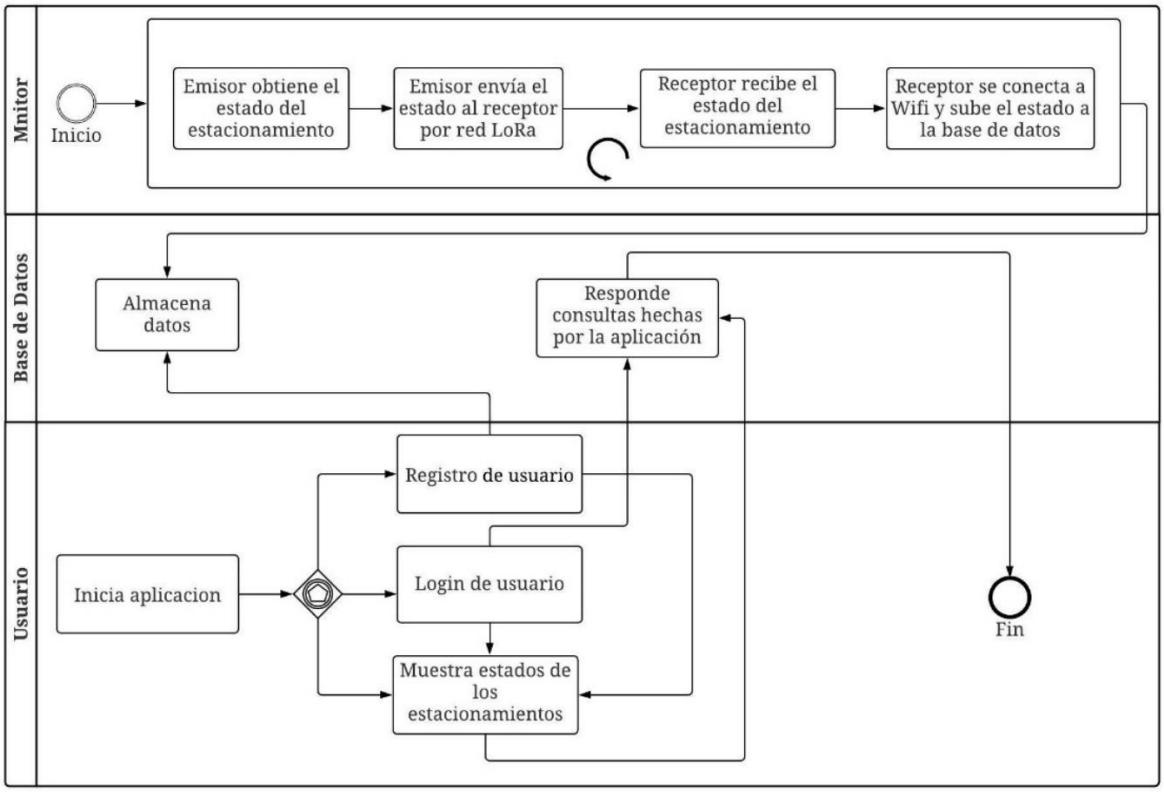



Figura 32: Modelo de proceso del sistema

## 4.7. Descripción de los actores del sistema

El modelo de procesos mostrado en la sección anterior cuenta con 3 actores principales, los cuales se describen a continuación:

- Monitor: En este actor se encuentra el prototipo IoT, tanto emisor como receptor, los cuales se encargan de verificar los cambios en el estacionamiento y enviar estos datos a la base de datos correspondiente.
- Base de Datos: Almacena los datos enviados por el prototipo IoT, además de responder frente a las peticiones que solicita la aplicación móvil. Cabe mencionar que esta base de datos es Realtime Database de firebase, la cual permite a la aplicación móvil recibir la información en tiempo real, frente a cambios en los datos almacenados.
- Usuario: Es quien desee o requiera revisar el estado en tiempo real del estacionamiento, con el fin de informarse sobre las plazas disponibles con las que cuenta.

## 4.8. Arquitectura del sistema

A continuación, se explicará en detalle la solución llevada a cabo en este proyecto, la cual consta de dos partes, la sección de hardware, que involucra el desarrollo del prototipo IoT, en donde se mostrarán las conexiones y los dispositivos utilizados, además de la sección de software, donde se detalla lo que se utilizó para programar el prototipo IoT, el uso de la base de datos y el desarrollo de la aplicación móvil.

### 4.8.1. Hardware

#### 4.8.1.1. Prototipo IoT

El prototipo IoT, como ya se mencionó en la descripción de la solución, consta de dos módulos Heltec LoRa WiFi (v2), los cuales están basados en el ESP32. El módulo emisor, mediante el sensor de proximidad y de magnetismo, se encarga de verificar si en la plaza de estacionamiento en la que se encuentra ubicado, hay un automóvil estacionado o no, y envía el estado ocupado o desocupado al módulo receptor, mediante la red LoRa. En cuanto al receptor, esta recibe la información y la sube a la base de datos por medio de una red WiFi. El prototipo emisor se conectó a una powerbank de 5.000 mAh (ver Figura 33) y el receptor se conectó directamente a la toma de corriente por medio de un adaptador de corriente de 5V y 2Ah (ver Figura 34).

El módulo emisor se instaló en una caja plástica, hecha en una impresora 3D tal como se muestra en la Figura 35, con el fin de optimizar el espacio que el prototipo utiliza, además de proporcionar comodidad a la hora de realizar las pruebas pertinentes en terreno. En cuanto al receptor, este se coloca igualmente en una caja hecha por una impresora 3D a medida como se visualiza en la Figura 36, pues solo debe conectarse a la fuente de energía y no requiere conexiones extras, exceptuando su antena correspondiente para la comunicación LoRa, en cuanto a la antena WiFi, la trae incorporada en la placa.

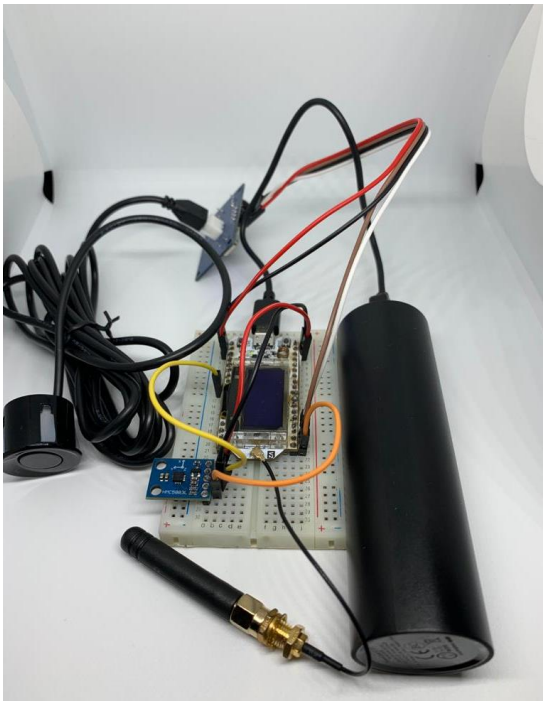


Figura 33: Prototipo emisor sin carcasa

Figura 34: Prototipo receptor sin carcasa

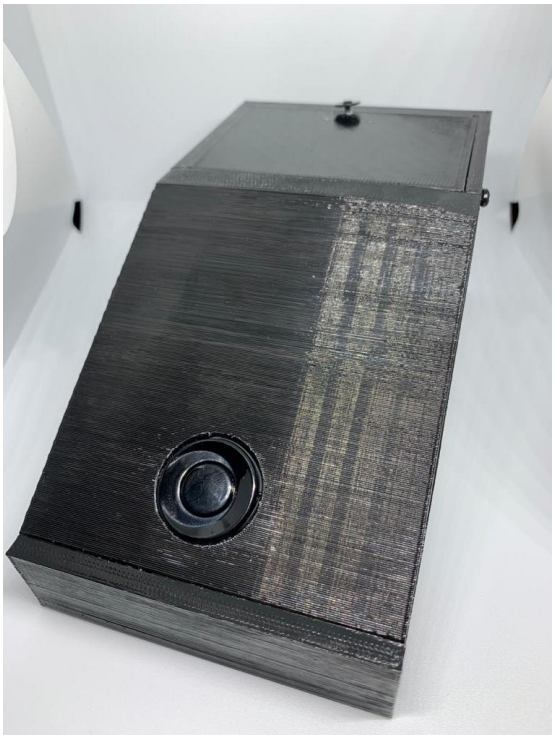


Figura 35: Prototipo emisor con carcasa

Figura 36: Prototipo receptor con carcasa

### 4.8.1.2. Diagrama de conexión del prototipo IoT

En esta sección solo se muestra el diagrama de conexión del módulo emisor, ya que este se conecta a una protoboard, además de conectarse a los sensores necesarios.

En este prototipo se tiene insertado de forma directa el módulo heltec LoRa WiFi 32 (v2) a una protoboard de 400 puntos, con el fin de interconectar de manera más sencilla los sensores JST-SR04T (ultrasónico impermeable) y HMC5883L (magnetómetro). Para interconectar el microcontrolador y sensores, se necesitó el diagrama de pines que posee el microcontrolador detallado en la Figura 37. El sensor JST-SR04T consta de 4 pines: 5V para voltaje, Trigger y Hecho para la comunicación y GND para tierra. El pin de 5V se conectó de forma directa al pin de 5V del microcontrolador y siguiendo esta metodología, el GND del sensor al GND del microcontrolador. Respecto a los pines de comunicación, Trigger se conectó al pin 12 y Echo al 13. En cuanto al HMC5883L cuenta con 4 pines, los cuales son: SDA, SCL, VCC y GND. Los pines SDA y SCL se conectaron de forma directa a los SDA y SCL del microcontrolador, 21 y 22 respectivamente. En cuanto al VCC, este se conectó de forma directa a un pin de 3.3V, cabe destacar que este sensor es capaz de funcionar a 3.3V o a 5V y en cuanto al pin GND, se conectó a un GND del microcontrolador. El diagrama de conexión completo se puede ver en la Figura 38.

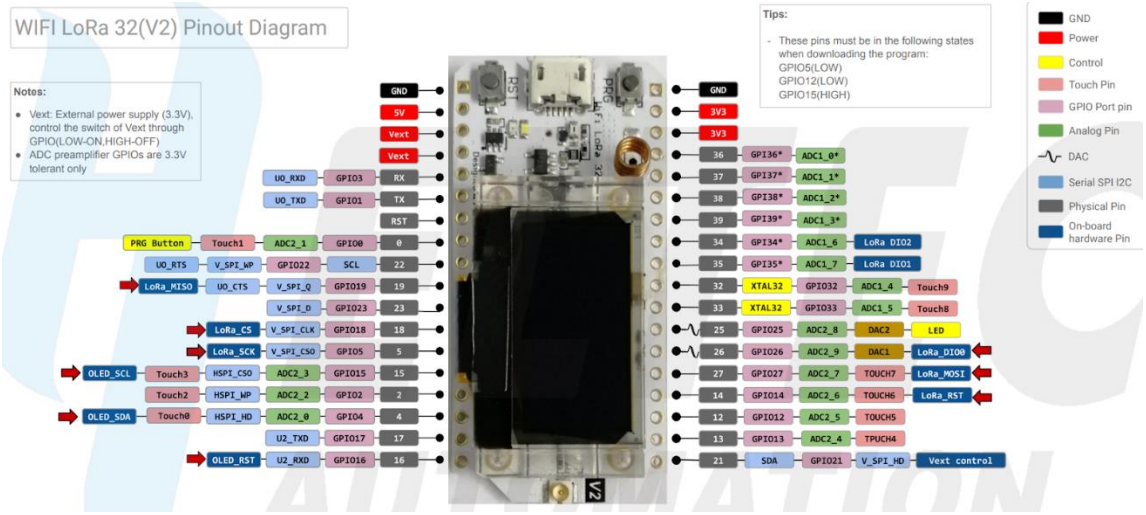



Figura 37: Diagrama de pines Heltec WiFi LoRa 32 (v2)

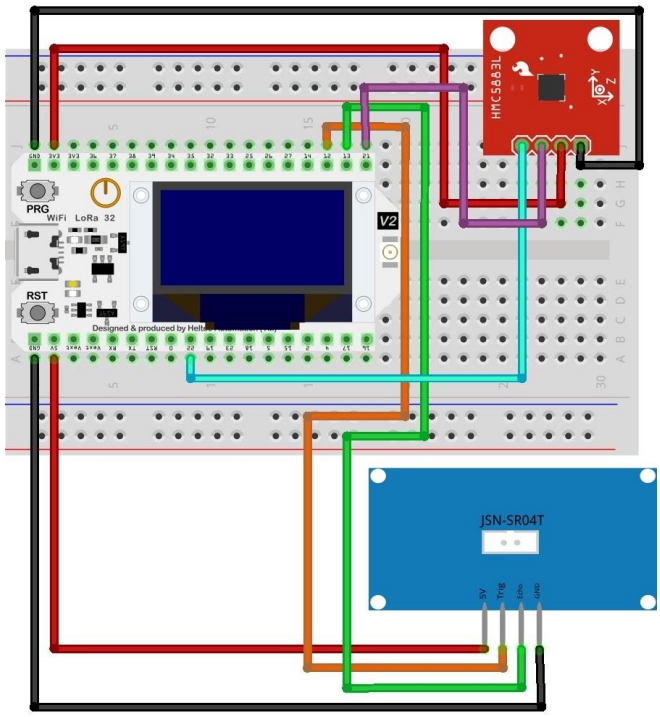



Figura 38: Diagrama de conexión prototipo emisor

### 4.8.2. Software

En cuanto al desarrollo de algoritmos para los prototipos, base de datos y aplicación móvil, se utilizaron las siguientes tecnologías de desarrollo de software:

#### 4.8.2.1. Software de Desarrollo

**Prototipos IoT**

- Arduino IDE: Es un software de código abierto que permite escribir algoritmos y subirlos a los microcontroladores compatibles con este, con el fin de que realicen alguna función específica. Es compatible con Windows, Mac OS y Linux, utiliza el lenguaje de programación Arduino, similar a C++. Cuenta con múltiples librerías, tanto propias como de la comunidad, con el fin de que la utilización de tecnología IoT sea simple. Los algoritmos que se cargan a las placas de desarrollo se llaman sketch. Este software se utilizó para crear los algoritmos de los microcontroladores y cargarlos a estos mismos.

- Librería NewPing: Es una librería de Arduino, creada por un particular, con el fin de que la utilización de sensores ultrasónicos sea más precisa, fácil y que el código escrito

en el sketch, sea legible y corto. Permite medir distancias con solo una función, a diferencia que, si se utiliza sin esta librería, es necesario realizar cálculos matemáticos con los pulsos de los sensores ultrasónicos. Se utilizó para el correcto funcionamiento del sensor JST-SR04T en el módulo emisor.

- Librería MechaQMC5883L: Es una librería de Arduino escrita por un particular, que facilita la utilización de los sensores magnetómetro QMC5883L y HMC5883L. Esta solo debe inicializarse y posteriormente asignar a variables, las fuerzas magnéticas detectadas por el sensor. Se utilizó para el correcto funcionamiento del sensor HMC5883L en el módulo emisor.

- Librería FirebaseESP32: Es una librería que permite y facilita la comunicación entre un microcontrolador basado en el ESP32 y firebase. No está creada por Google, pero permite hacer modificaciones en la base de datos y soporta múltiples tipos de variables, ya sean INT, STRING, DOUBLE, BOOLEAN o JSON. Solo soporta conexiones mediante WiFi o Ethernet. Se utilizó para la comunicación entre el módulo receptor y la base de datos.

- Librería HELTEC: Es una librería que permite controlar el funcionamiento del módulo Heltec WiFi LoRa 32 (v2). Permite controlar el uso de la pantalla OLED que trae incorporado, además del funcionamiento del módulo LoRa SX1276, tanto para recibir como enviar datos.

**Aplicación Móvil**

- React Native: Como se mencionó anteriormente, es un *framework* escrito en javascript basado en el *framework* RactJS, utilizado para el desarrollo de aplicaciones híbridas mediante componentes web. En la sección 4.8.2.3 se detallará su utilización en el proyecto.

- Expo: Expo es un *framework* y una plataforma para aplicaciones desarrolladas en React Native. Ofrece herramientas y servicios construidos alrededor de React Native y plataformas nativas que lo ayudan a desarrollar, construir, implementar e iterar rápidamente en iOS, Android y aplicaciones web desde la misma base de código JavaScript. Permite emular las aplicaciones desarrolladas en su propia aplicación Expo, para verificar el correcto funcionamiento de estas en los sistemas Android e iOS.

- React Navigation: Es una librería de React Native, que permite que la navegación por la aplicación utilice los componentes nativos de navegación para cada sistema operativo y otorga fluidez a las transiciones. Permite la navegación por pestañas, por deslizamiento lateral, por pila, o entre otras.

#### 4.8.2.2. Firebase

Como ya se mencionó anteriormente, se utilizará Realtime Database para el almacenamiento de los datos recopilados por el prototipo IoT y como API para acceder a los datos mediante la aplicación móvil. En cuanto al almacenamiento de datos, estos se almacenarán en formato JSON, y estarán separados por estacionamientos, donde en cada uno, estarán las plazas con las que cuentan y su estado ocupado o desocupado, siendo los valores 1 o 0 respectivamente, los que se almacenarán. En la Figura 39 se puede ver de forma detallada la estructura de la base de datos.

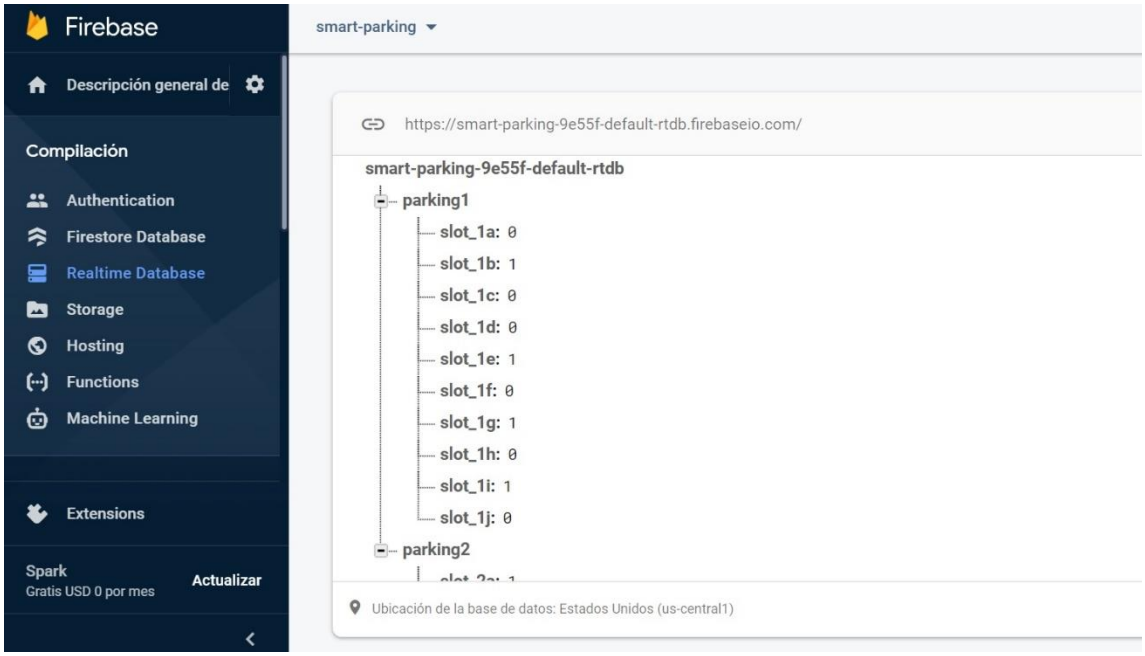



Figura 39: Realtime Database del proyecto

Además, se utilizará la funcionalidad de autenticación, que permitirá mediante a la aplicación móvil, registrar usuarios (ver Figura 40).

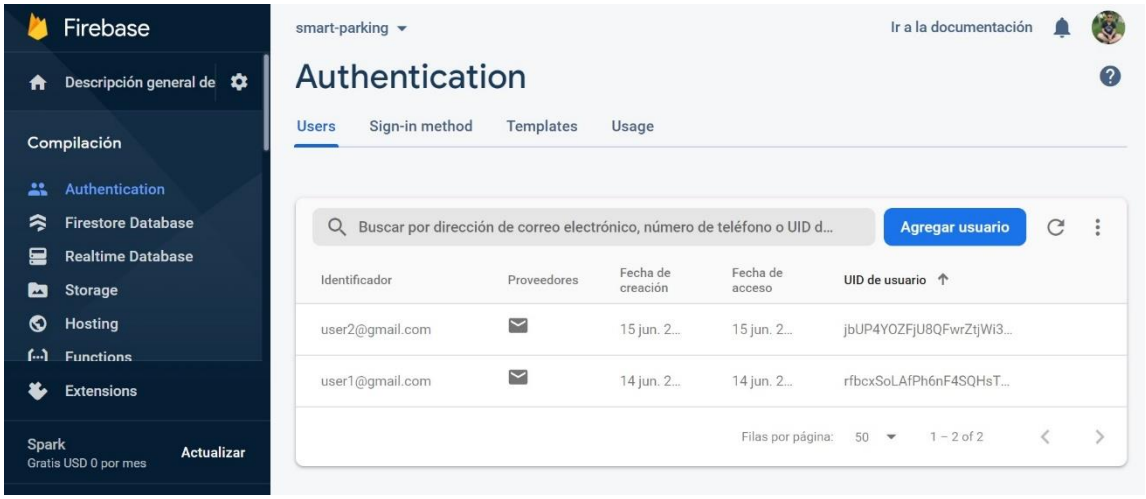



Figura 40: Vista de usuarios registrados en firebase

### 4.8.2.3. Aplicación Móvil

En cuanto al desarrollo de la aplicación móvil, la estructura de esta se puede ver en la Figura 41, siendo dos las vistas principales con las que cuenta la aplicación, las cuales son Estacionamientos y Cuenta. El funcionamiento y configuración de estas vistas, se mencionará a continuación.

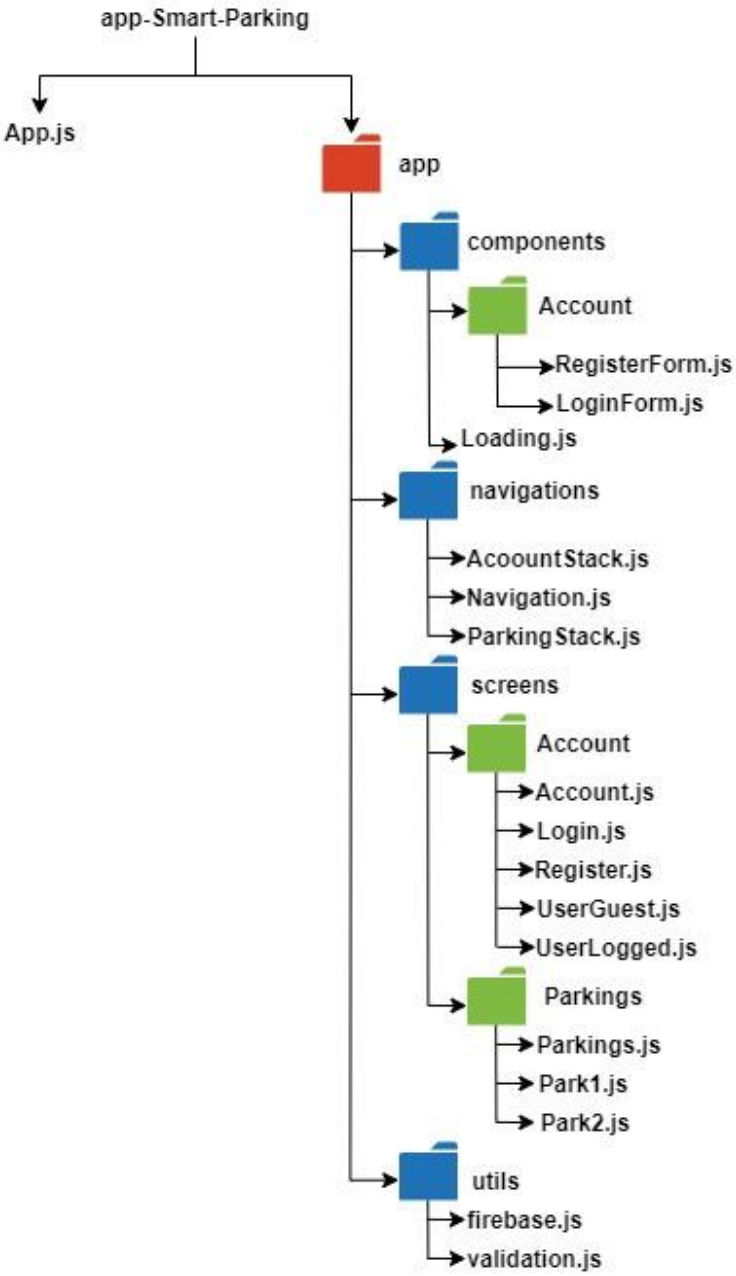



Figura 41: Estructura de la aplicación móvil

**Navegación Inferior**

Es aquella que se muestra en la Figura 42, esta permite seleccionar las vistas Estacionamientos y Cuenta. Se encuentra instanciada en el archivo app.js, que es el script principal que maneja la aplicación móvil. Cada vista en la navegación cuenta con su nombre y su respectivo icono con el fin de facilitar la visualización y selección de estas.


Figura 42: Barra navegadora inferior de la aplicación

Está controlada por el archivo Navigation.js, que se encuentra en "app/navigation", en donde se encuentran los links a los stack de las vistas. Estos stack, permiten cambiar la ventana que se muestra en la vista dependiendo de las acciones del usuario y se encuentran bajo la misma ubicación que el archivo Navigation.js

**Vista de estacionamientos**

En esta vista se visualiza el estado de las plazas de los dos estacionamientos disponibles, se puede ver un ejemplo en la figuras 43 y 44, donde se muestran plazas ocupadas y desocupadas del estacionamiento 1 en los sistemas operativos Android e iOS. Otra característica que se aprecia en la figura, es que en la parte superior, también cuenta con un propio sistema de navegación, el cual permite alternar entre los estacionamientos disponibles en la aplicación. Las plazas del estacionamiento cambian en tiempo real, es decir, que la base de datos al sufrir un cambio avisa a la aplicación que hubo una alteración en una plaza del estacionamiento, por lo que la aplicación, actualiza el estado y en caso de pasar a disponible, muestra el texto "DISPONIBLE" en verde o si está ocupado, coloca un automóvil. De la misma forma cambia el contador de plazas disponibles en la navegación superior, por lo que, dependiendo de los cambios sufridos en la base de datos, los aumenta o disminuye.

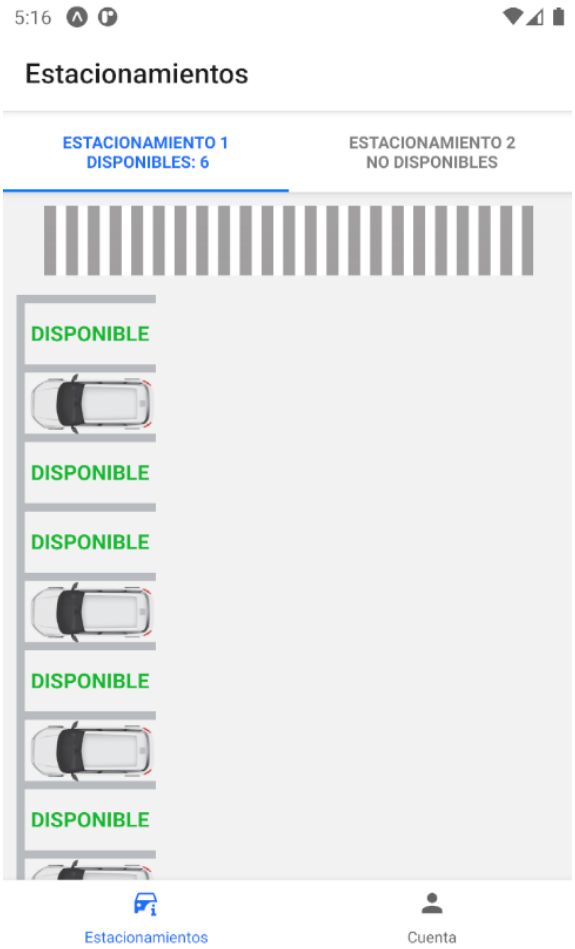



Figura 43: Vista estacionamientos en sistema Android

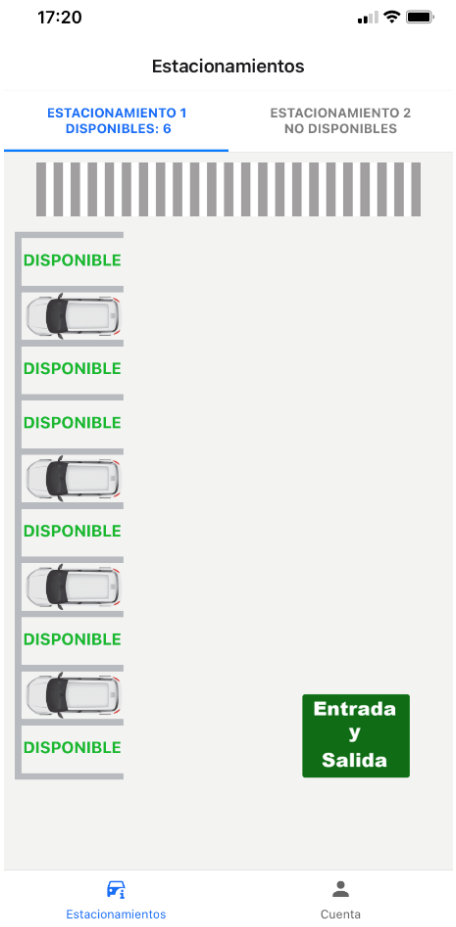



Figura 44: Vista estacionamientos en sistema iOS

El funcionamiento de esta vista está controlado por el archivo Parkings.js, ubicado en "app/screens/Parkings" dentro de la estructura de la app, se encarga de hacer las consultas pertinentes sobre el estado de los estacionamientos, y dependiendo del estacionamiento que quiere ver el usuario, se despliega sobre esta: Park1.js, que muestra la información del estacionamiento 1 o Park2.js, que muestra la información del estacionamiento 2, ubicados en la dirección ya mencionada.

**Vista Cuenta**

Esta vista dependiendo si el usuario ha iniciado o no sesión en la aplicación, desplegará la ventana correspondiente a ese estado. En caso de no tener una sesión iniciada, desplegará una ventana informativa (ver figuras 45 y 46), que mostrará mediante un texto, la funcionalidad de la aplicación. Al pulsar sobre el botón cuenta, se desplegará una ventana que solicitará las credenciales para iniciar sesión en la aplicación (ver figuras 47 y 48), en caso de ya tener una cuenta. Si no se cuenta con una, entonces tiene la opción de registrarse pulsando en el texto resaltado "Registrarse". Una vez pulsado, se desplegará la ventana de registro con el formulario de registro (ver figuras 49 y 50), el cual una vez

llenado con información consistente, al pulsar en registrar, se desplegará la ventana correspondiente al perfil del usuario (ver figuras 51 y 52). De la misma forma, en caso de iniciar sesión, se desplegará esta información.

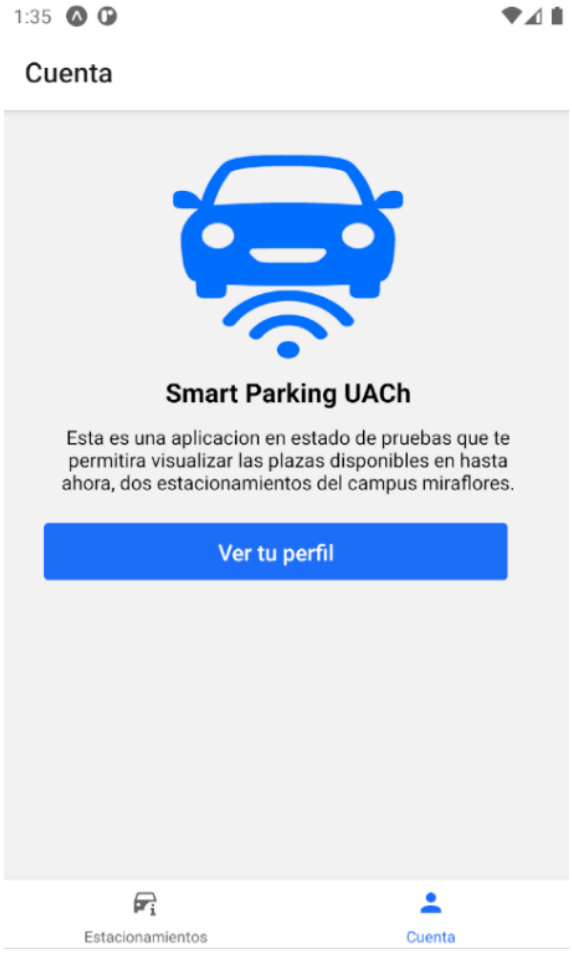



Figura 45: Vista ventana informativa Sistema Android

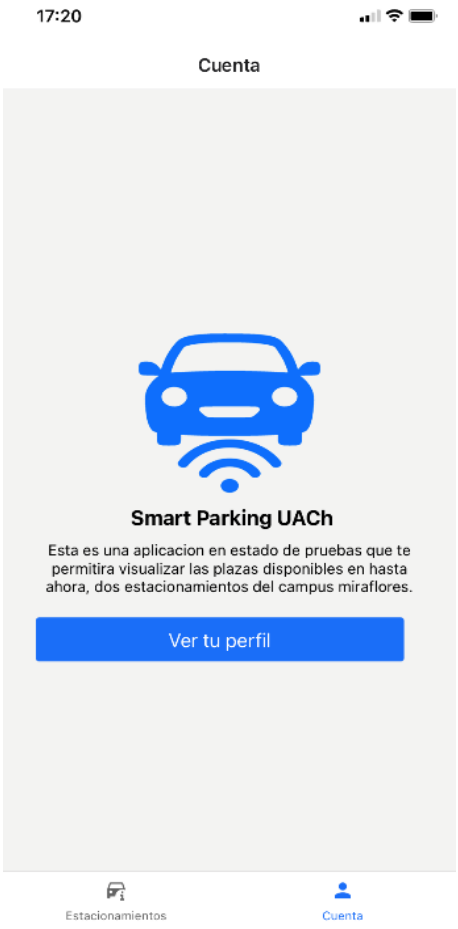



Figura 46: Vista ventana informativa Sistema iOS

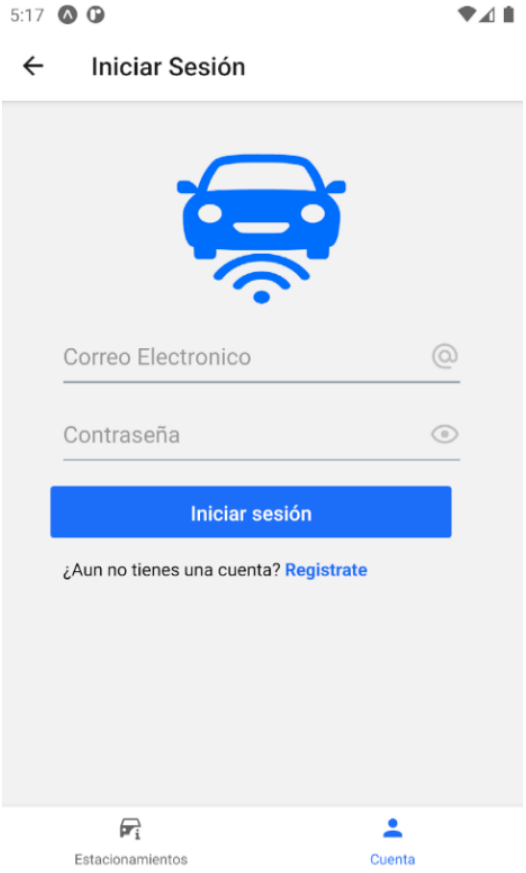



Figura 47: Vista ventana inicio de sesión sistema Android

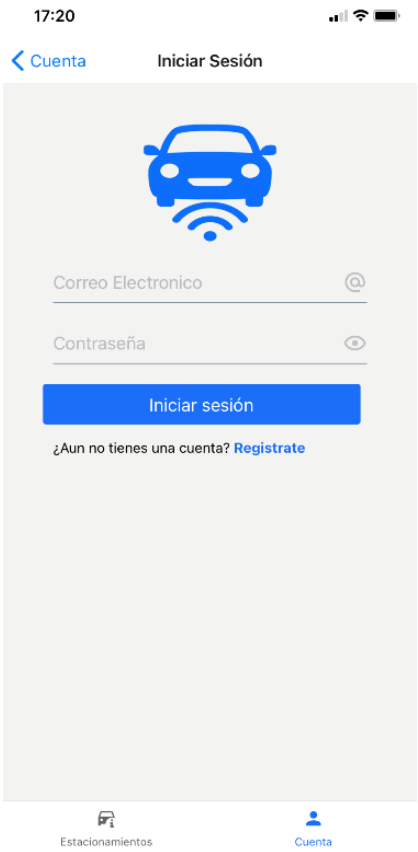



Figura 48: Vista ventana inicio de sesión sistema iOS

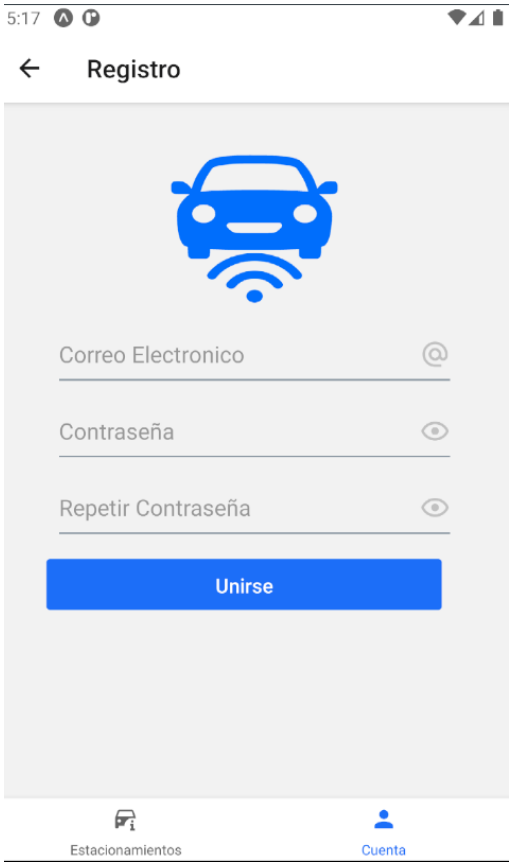



Figura 49: Vista ventana registro de usuario en sistema Android

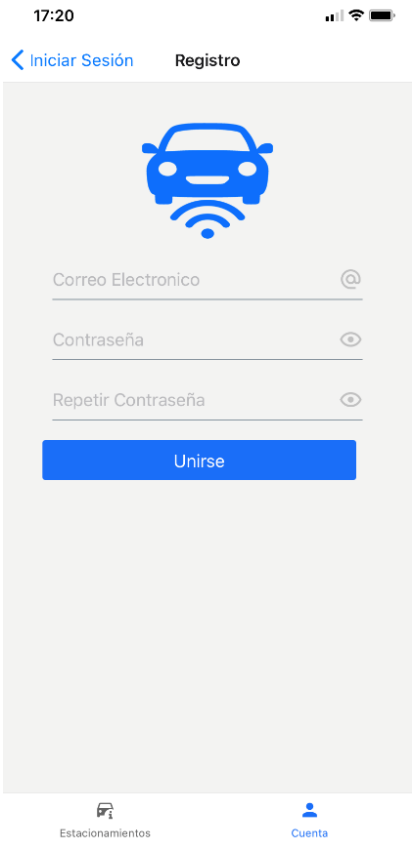



Figura 50: Vista ventana registro de usuario en sistema iOS

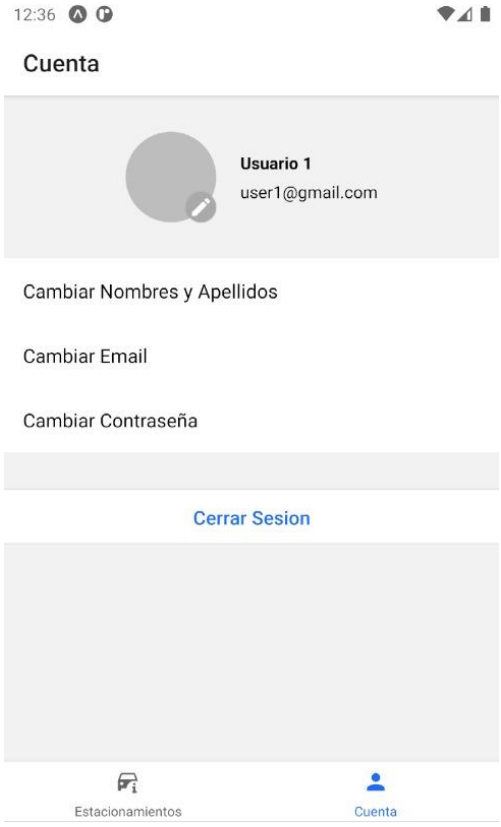



Figura 51: Vista de usuario registrado sistema android

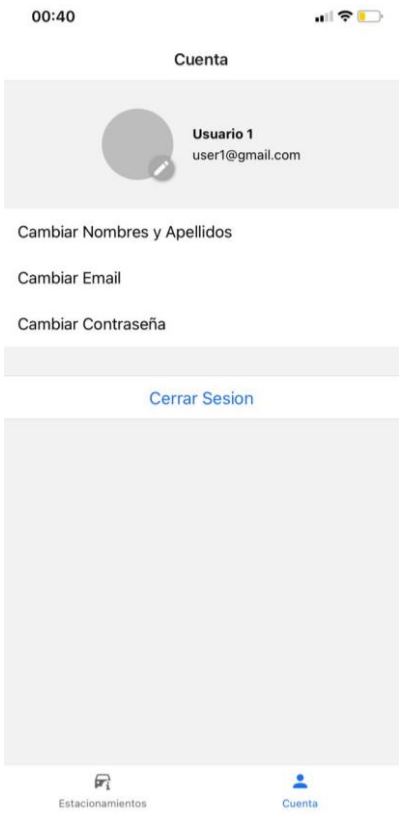



Figura 52: Vista usuario registrado en sistema iOS

El funcionamiento de esta vista está controlado por el archivo Account.js, ubicado en "app/screens/Account" dentro de la estructura de la aplicación, se encarga de cambiar el estado del usuario a logueado o no logueado, y dependiendo del estado, se despliega la ventana para "UserLogged.js", correspondiente al perfil del usuario logueado o "UserGest.js", correspondiente a la ventana informativa, que da las opciones para iniciar sesión mediante "Login.js" o registrarse mediante "Register.js".

# 5. PRUEBAS

A medida que se desarrollaron los prototipos IoT y la aplicación móvil, se llevaron a cabo distintas pruebas para cada tecnología. En los prototipos IoT, se probaron los sensores y microcontroladores, con el fin de encontrar los parámetros adecuados a la hora de crear los algoritmos que estos ejecutaran en terreno. En cuanto a la aplicación móvil, se testeo la funcionalidad de esta, ya que debe ser lo más informativa a la vista del conductor, con el fin de crear la menor distracción posible, debido a que estará conduciendo mientras busca estacionamiento en esta. Una vez completadas las pruebas, se llevaron a cabo las pruebas correspondientes al sistema completo, ya que es necesario medir la precisión de su funcionamiento y así llevar a cabo la validación correspondiente. Estas pruebas serán detalladas a continuación.

## 5.1. Prototipo IoT

### 5.1.1. Sensor Magnetómetro HCM5883L

Como se mencionó anteriormente el sensor magnetómetro mide la fuerza del campo magnético en los ejes X, Y y Z en el que se encuentra y además detectar variaciones de este, en caso de que un objeto metálico de un tamaño considerable se coloque a cierta distancia del sensor. Como un vehículo está construido con un porcentaje de metal considerable, es posible que, si se posa sobre el sensor, este provoque una variación en la lectura de la fuerza del campo magnético. Por lo que una de las pruebas que se llevaron a cabo, es estudiar la variación que un vehículo provoca en el sensor.

En esta prueba se desarrolló un algoritmo en Arduino IDE que correrá dentro del Heltec WiFi LoRa 32 (v2) conectado al HCM5883L, el cual por cada iteración hará veinte lecturas del campo magnético en X e Y, se obtendrá el promedio de estas y se enviará a una base de datos. La iteración se correrá 30 veces con el fin de encontrar un valor medio óptimo. Esta prueba se realizará con un vehículo NISSAN QASHQAI año 2013, y el proceso se llevará a cabo entre 0 [cm] y 100 [cm], con saltos de 10 [cm]. El punto 0 [cm] de las mediciones es aquel cuando el vehículo recubre completamente la caja, en donde se encuentra el motor vehículo. En la Figura 53 se visualiza el prototipo de pruebas de campo magnético dentro de la carcasa y en la Figura 54 el prototipo en el estacionamiento de pruebas.

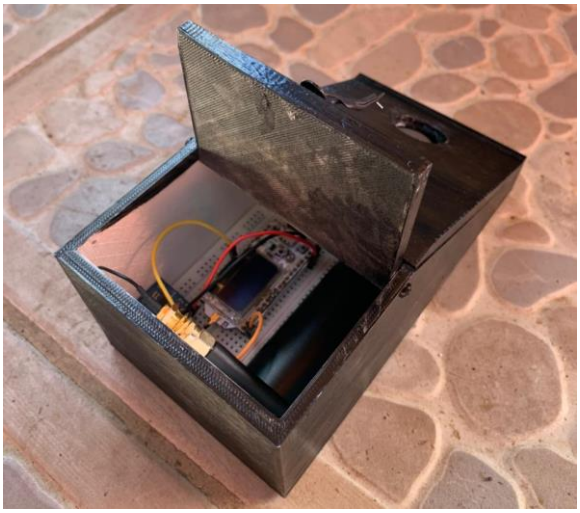


Figura 53: Prototipo de pruebas de sensor magnetómetro

Figura 54: Prototipo midiendo campo magnético

El resultado de las pruebas se puede visualizar en el gráfico de la figura 55, en el cual se visualizan claramente las variaciones en el campo magnético provocadas por el vehículo, una vez que este se encuentra encima del prototipo. Entre las distancias 20 [cm] y 100 [cm], las variaciones son muy similares, entre 30 [u] por eje. Sin embargo, entre 0 y 10 [cm], ya se observan variaciones considerables, a una distancia de 10 [cm], en X varía en aproximadamente 270 [u] y en Y varía en168 [u]. Ya cuando el automóvil esta encima del prototipo, es cuando se alcanza la máxima variación, obteniendo como resultado una variación de aproximadamente 530 [u] en X y en unas 270 [u] en Y. Otro resultado que se puede visualizar es que el campo magnético no afecta de manera proporcional a X e Y, ya que las variaciones obtenidas son distintas, para cada distancia.

El comportamiento del sensor frente a los cambios en el campo magnético producidos por el distanciamiento del vehículo, pueden caracterizarse aproximadamente mediante las expresiones matemáticas presentadas en la figura 56 para la fuerza X del campo magnético y 57 para la fuerza Y del campo magnético. En donde el eje “y” representa el valor del campo magnético y el eje “x” la distancia hacia el sensor.

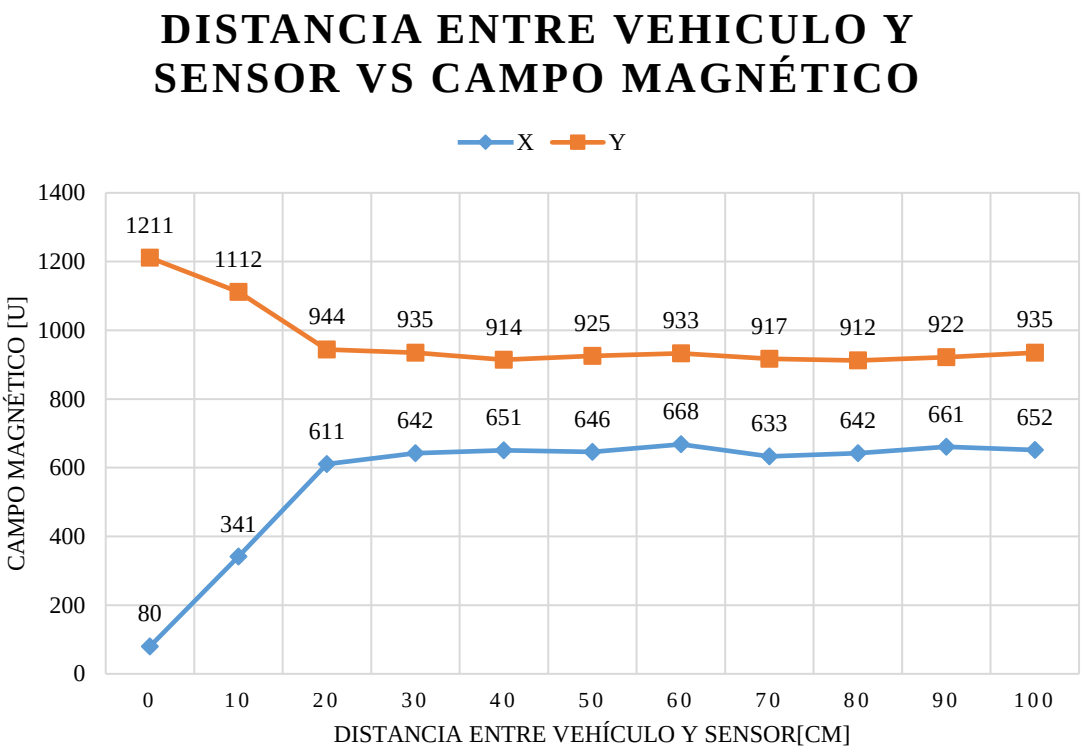



Figura 55: Grafico Distancia prototipo emisor vs campo magnético

$$y = -1{,}2214x^3 + 28{,}458x^2 - 208{,}32x + 1393{,}6$$

Figura 56: Expresión matemática del eje Y del campo magnético

$$y = 2{,}8219x^3 - 62{,}731x^2 + 436{,}25x - 283{,}26$$

Figura 57: Expresión matemática del eje X del campo magnético

### 5.1.2. Sensor JSN-SR04T

Este sensor se utiliza para medir distancias hacia obstáculos o como detector de obstáculos. En el proyecto se utilizará para medir la distancia que existe entre el vehículo y el prototipo emisor. La prueba que se llevó a cabo se realizó con el fin de estudiar la precisión de lectura del sensor en sus mediciones de distancia y así comprobar su correcto funcionamiento.

En esta prueba se desarrolló un algoritmo en Arduino IDE que se ejecutará en el Heltec WiFi LoRa 32 (v2), conectado al JSN-SR04T, el cual por cada loop imprimirá en el

monitor serial la distancia detectada por el sensor. En la prueba se ubicó el vehículo mencionado en la prueba anterior, a diferentes distancias, y hacer lecturas de esta comparándolo con una huincha de medir de construcción, chequeando la variación obtenida en cada medición. Las distancias seleccionadas son entre 20 [cm] y 160 [cm] con saltos de 20 [cm]. Cabe destacar que se comienza a los 20 [cm] ya que el sensor no logra medir distancias menores a esta. En la Figura 58, se puede visualizar el prototipo de pruebas y en la Figura 59 las pruebas en terreno junto al vehículo y la huincha de medir.

Figura 58: Prototipo de pruebas de distancia

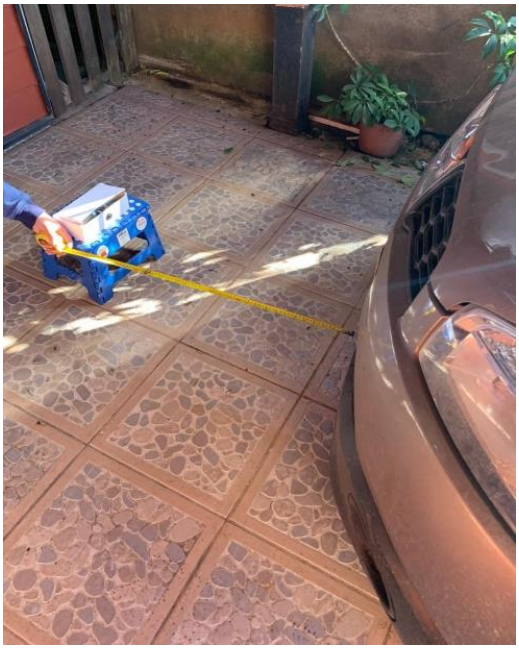


Figura 59: Prototipo midiendo distancia hacia un vehículo comparándolo con una huincha de medir

El resultado de las pruebas se visualiza en la Tabla 17, de la cual los resultados son claramente positivos. Las variaciones en las distancias son milimétricas, entre valores de 0 [mm] y 7 [mm], inferiores a 1 [cm]. Además, esta variación puede atribuirse también a error humano, ya que es posible que exista un porcentaje de error en las medidas establecidas hacia el automóvil por medio de la huincha de construcción. Debido que las distancias en el prototipo final serán en cm, estas variaciones milimétricas no afectarán en la precisión de la detección vehicular.

Tabla 17: Variaciones en valores tomados por el sensor JSN-SR04T

| Distancia | Sensor JSN-SR04T | Variación en [mm] |
|---|---|---|
| 20 | 20.4 | 4 |
| 40 | 39.3 | 7 |
| 60 | 60.0 | 0 |
| 80 | 80.1 | 1 |
| 100 | 100.0 | 0 |
| 120 | 120.4 | 5 |
| 140 | 140.5 | 5 |
| 160 | 159.6 | 4 |

Al momento de implementar el uso de este sensor dentro de la carcasa hecha a medida para el prototipo, con el fin de que este detecte la distancia hacia la parte inferior del automóvil, la medición fue de 21 [cm], muy cercano al límite que tiene el sensor.

### 5.1.3. Módulo de transmisión LoRa SX1276

El módulo de transmisión LoRa SX1276 es el que lleva insertado la placa de desarrollo Heltec Wifi LoRa 32 (v2) que se utilizara en el proyecto para la transmisión de datos entre módulo emisor y receptor. Entre sus especificaciones resalta su largo alcance, sin embargo, se especifica que este es de aproximadamente 2.6 [km] en un área abierta. Esta prueba consiste en medir la distancia e intensidad de señal con la que el receptor recibe los paquetes, una vez que el emisor los envía, con el fin de realizar estudios posteriores sobre el escalamiento de estas placas, si se utilizaran en espacios urbanos de gran tamaño.

En esta prueba se utilizaron los sketch de ejemplos de la Librería de Arduino Heltec, *OLED_LoRa_Sender* y *OLER_LoRa_Receiver*. *OLED_LoRa_Sender* es el sketch que se encarga de enviar un paquete LoRa cada 1 [s], este paquete contiene el string “hello” concatenado con el contador del loop, el cual aumenta en 1 por cada de ejecución y además muestra en su pantalla oled el número del paquete que está enviando. En cuanto a *OLED_LoRa_Receiver*, es el sketch que recibe la data enviada por el emisor, y muestra en la pantalla su oled, la intensidad con la que llega el paquete y el contenido del paquete, en este caso “hello” concatenado con “count”. Ambos sktech se cargaron en los módulos Heltec WiFi LoRa 32 (v2) emisor y receptor respectivamente mediante la IDE de Arduino. El receptor se insertó en una caja de cartón, alimentándose de energía por medio la powerbank Anker powercore 5000. En cuanto al módulo emisor, este se ubicó en las dependencias del tesista conectado directamente a la red eléctrica por medio de un adaptador de corriente de 5V y 2 Ah.

Las pruebas de medición se llevaron a cabo trazando puntos mediante Google maps, cada punto aumentara su distancia en 25 [m] con respecto a la posición del emisor. En estos puntos se medirá la intensidad de señal o RSSI con la que el paquete llega al receptor. En la Figura 60 se muestran los puntos escogidos en donde se apuntará el RSSI y se trasladará el módulo receptor.

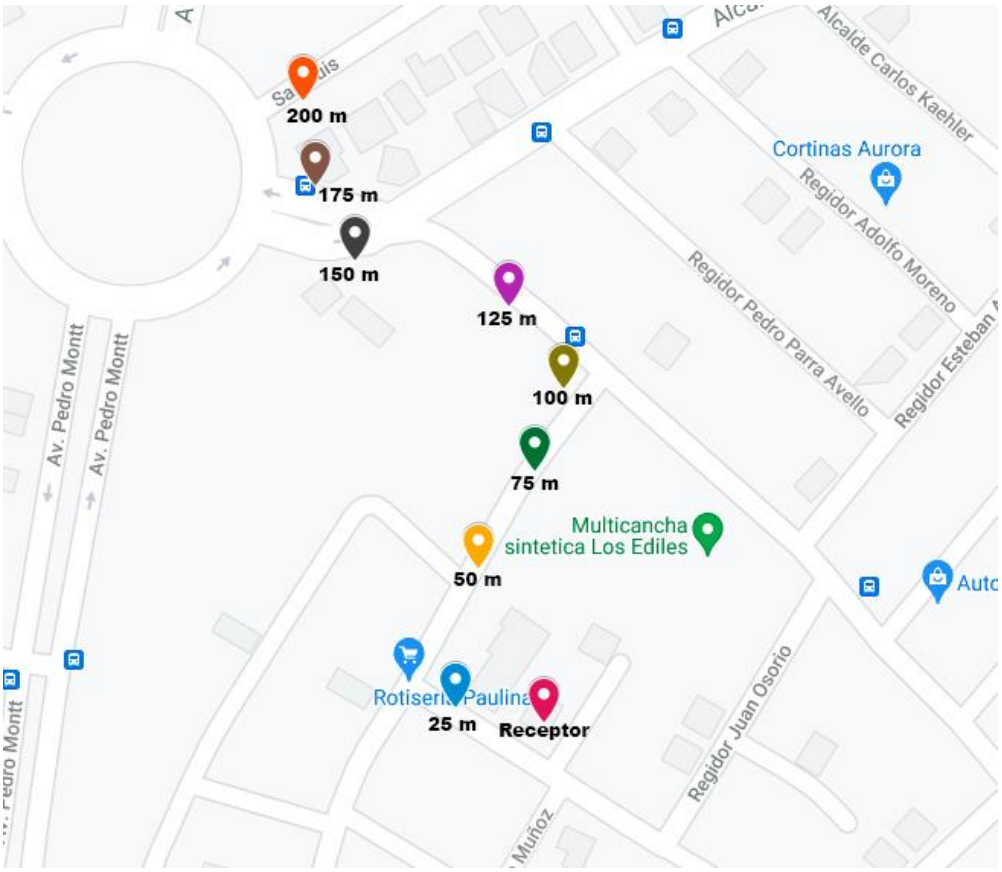



Figura 60: Puntos de toma de intensidad de señal del paquete LoRa

Los resultados obtenidos se pueden visualizar en el grafico presentado en la Figura 61, en la cual se puede ver claramente que el RSSI baja su valor, a medida que aumente la distancia entre los módulos. Otro punto para tener en cuenta durante las pruebas es que, si el receptor no recibe paquetes, significa que se alejó lo suficiente como para no captarlos, y durante la realización de las mediciones, la distancia máxima fue 200 [m], debido a que a una mayor distancia que esta, no es posible captar los paquete del emisor.

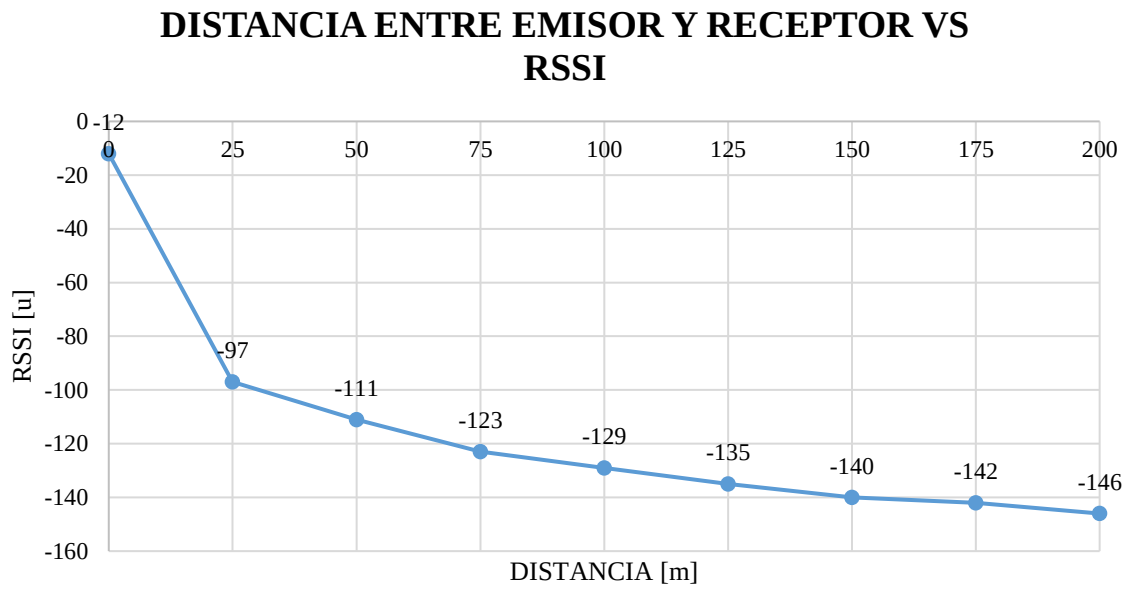



Figura 61: Grafico de Distancia entre emisor y receptor vs RSSI del paquete recibido

Los resultados obtenidos, son claramente inferiores en un entorno urbano, lo cual, igual puede atribuirse a que está conectado a una placa de desarrollo que no suministra suficiente corriente para alcanzar mayores distancias. Por otro lado, si se compara con la red WiFi, está en un radio urbano, alcanza aproximadamente los 20 [m], por lo que para proyectos acordes a tecnología IoT, tiene un alcance 10 veces mayor que la mencionada.

### 5.1.4. Funcionamiento prototipo IoT completo (Emisor y Receptor)

Tanto módulo emisor como receptor deben comunicarse para subir la información obtenida a la base de datos, es por esto que es necesario realizar las pruebas pertinentes entre la comunicación entre el módulo emisor, teniendo en este cargado el algoritmo que ejecutara para detectar un automóvil y el módulo receptor, que tendrá cargado el algoritmo que ejecutara para recibir el cambio de estado y subirlo a la base de datos.

Los algoritmos que se ejecuten en los microcontroladores emplean dos funciones principales, `setup()` y `loop()`, en `setup()` generalmente se preparan o inicializan los parámetros necesarios para que el microcontrolador funcione correctamente, esta solo se ejecuta una vez y en caso de que el microcontrolador se apague o reinicie, esta se vuelve a ejecutar. En cuanto a la función `loop()`, esta se ejecuta sin un límite de veces y se puede definir el tiempo que transcurrirá para que la función vuelva a ser ejecutada. Respecto al funcionamiento del módulo emisor, este se puede observar en el pseudocódigo mostrado en la Figura 62, en donde se utilizan las variables globales `x_global` e `y_global` que contienen los valores de campo magnético del lugar de pruebas, estas se obtuvieron mediante las pruebas de campo magnético, luego en la función `setup()` se preparan los parámetros que se utilizaran de la red LoRa mediante `LoRaInit()`. En la

función `loop()` se obtienen por medio de la función `getDistance()`, la distancia hacia el vehículo asignada a la variable `distance` y la función `getMagneticForce()`, que obtiene el campo magnético en ese instante de las fuerzas en X e Y. Ambas funciones realizan mediciones 20 veces y se promedian, con el fin de obtener valores más precisos. Luego se emplea el mecanismo de detección de vehículos, en el cual, si la distancia hacia el objeto es menor a 30 [cm] y si el campo magnético en las fuerzas "x" e "y" sufre variaciones mayores a 150 unidades, significa que el objeto detectado es un vehículo. Al verificar que es un vehículo, mediante la variable `state`, se define el estado del estacionamiento, en el cual, si es 1, significa que el objeto es un automóvil o 0 si no lo es, y por ende no está siendo utilizado el slot de estacionamiento. Finalmente, mediante la función `sendLoraPacket()`, se envía el id del slot en el que se encuentra el módulo emisor junto con el estado en el que se encuentra este por medio de un paquete LoRa al receptor, este proceso mencionado en la función `loop()`, se ejecuta cada 10 segundos.

```
int x_global;
int y_global;
Function setup():
    LoRaInit();
return
Function loop():
    int distance x, y, state;
    getDistance(distance);
    getMagneticForce(x,y);
    if distance2 < 30 then
        if (abs(x − x_global) > 150) and ((abs(y − y_global) > 150) then
            state ← 1;
        else
            state ← 0;
        end
    else
        state ← 0;
    end
    sendLoraPacket("slot_id :".concat(state));
    delay(10);
return
```

Figura 62: Pseudocódigo módulo emisor

El módulo receptor ejecutará un algoritmo más simple que el emisor, pues estará encargado de recibir el paquete LoRa, normalizar su contenido al correspondiente en la base de datos y posteriormente subirlo a la misma. Observando el pseudocódigo de la Figura 63 que explica el funcionamiento del algoritmo de este módulo, en la función `setup()`, se prepara el dispositivo como un receptor de señales LoRa mediante la función `LoRa.receive()`, se prepara la conexión WiFi mediante la función prepareWifi() y se prepara la conectividad hacia la base de datos mediante `prepareDB()`. Luego en la función `loop()`, en la variable `packet`, se escribe el contenido recibido por la red LoRa y posteriormente se normaliza en las variables `slot` y `value`, en donde slot, es el identificador del slot que tiene el módulo emisor que envió

la señal, y `value` es la ocupación de este. Finalmente, mediante la función `sendToDB()` se envía el contenido normalizado a la base a datos por medio de WiFi.

```
string packet;
Function setup():
    LoRa.receive();
    prepareWifi();
    prepapareDB();
return
Function loop():
    int cont ← 0;
    string slot, value;
    string packet ← loraPacketReceive();
    if size_packet > 1 then
        while packet[cont] != ':' do
            slot ← slot.add(packet[cont]);
            cont ← cont + 1;
        end
        value ← value.substring(cont + 1);
        sendToDatabase(slot,value);
    end
    delay(10);
return
```

Figura 63: Pseudocódigo módulo receptor

Las pruebas del prototipo IoT completo fueron llevadas a cabo en el estacionamiento de pruebas mencionado en las pruebas anteriores, en donde se verificó la correcta lectura de datos del estacionamiento por parte del emisor, el correcto envío de datos por parte de este y la correcta recepción y posterior envío de datos a la base de datos por parte del emisor. Se llevaron a cabo 20 pruebas, que correspondían a verificar si al estacionar el vehículo se enviaba un 1 a la base de datos, y si no había, que envíe un 0. Los resultados obtenidos fueron completamente satisfactorios, pues los cambios de estado del estacionamiento se reflejaban en la base de datos, obteniendo una precisión del 100%, ya que las 20 veces que se estacionó el vehículo, la base de datos cambio el estado del slot, y al momento de desocuparlo, también esta cambiaba su estado. No obstante, esta precisión es posible que varíe dependiendo del vehículo que se ubique sobre el prototipo, pues la detección está sujeta a la altura del vehículo, a la forma en que se estacione el conductor y a como el vehículo afecta el campo magnético del prototipo emisor.

### 5.1.5. Consumo de energía Módulo Emisor

Una vez testeados los sensores que se utilizaran en el módulo emisor, como ya se mencionó, este debe hacer un uso óptimo de la energía, ya que será conectado a una batería recargable, la cual tiene una cierta capacidad definida, con la cual suministrar de energía al prototipo.

Para estas pruebas se utilizó un amperímetro, como el de la Figura 64, el cual se conecta la entrada de energía a la powerbank, la salida de este a la placa de desarrollo y mediante luces led, muestra el consumo en amperios. En cuanto al algoritmo que usará el microcontrolador, será el mismo que se utilizó en el emisor en la prueba de funcionamiento entre módulo emisor y receptor. En cuanto a cómo se procederá a calcular el consumo, se dejará encendido el módulo por 30 minutos, y en lapsos de 5 min, se anotará el consumo en amperios.

Figura 64: Voltímetro y amperímetro USB utilizado en las pruebas

En los resultados a partir de las mediciones con el tester USB, se obtuvo que, en promedio, el prototipo utiliza 1mA cada 10 minutos, por lo que consumirá 6 mAh. Con ese valor ya es posible calcular el tiempo de autonomía que brindara la batería al prototipo, el cual es de aproximadamente 34.7 días, considerando que la capacidad de la batería es de 5000 mAh. Esta cifra, no es negativa, ya que se podría considerar suficiente, sin embargo, el algoritmo no tiene aplicado ninguna optimización, como métodos sleep o mejora en la toma de decisiones, en cuanto al envío de datos, por lo que el consumo de energía se podría optimizar llegando a triplicar o cuadruplicar la disminución del consumo por parte del prototipo emisor.

## 5.2. Aplicación Móvil

Como ya se mencionó anteriormente, la aplicación debe ser de fácil uso, baja interacción con el usuario e intuitiva, con el fin de crear la menor distracción posible al conductor mientras conduce. Sin embargo, estos criterios solo deben aplicar a la ventana que renderizara la información de las plazas disponibles en los estacionamientos que requiera ver el usuario. Las ventanas correspondientes a Cuenta, no será necesario optimizar, debido a que el usuario no está obligado registrarse para hacer uso de la aplicación, y se asume que en caso de que el usuario lo desee, esto lo llevara a cabo mientras no conduce.

### 5.2.1. Vista Principal de cada estacionamiento

Como se mostró anteriormente, en la sección Arquitectura de la solución, la vista principal de estacionamientos, indica si un estacionamiento está disponible mediante letras mayúsculas y color verde, en cuanto a la representación de un slot ocupado, se utilizó la imagen de un auto. La ubicación de estos se encuentra al lado izquierdo, considerando que los conductores comenzaran a buscar las plazas disponibles, una vez que entren al

estacionamiento, por lo que, al visualizar el estacionamiento desde el vehículo, los slot están ubicados a la izquierda. El usuario para visualizar los parkings deberá hacer *scroll* sobre la vista y para orientarse de una forma práctica, se utilizaron imágenes representativas de esos estacionamientos. En el caso del estacionamiento que se encuentra afuera del instituto, se utilizó la imagen que representa la entrada y salida, con el fin de indicar al conductor, que la vista del estacionamiento en la aplicación comienza desde abajo y que este termina en la figura del paso peatonal, ubicado en el comienzo de la vista. En cuanto al estacionamiento ubicado afuera el edificio 8000, se utilizó la imagen del paso peatonal para indicar el comienzo del estacionamiento el cual se encuentra en la parte inferior de la vista del estacionamiento y la imagen de una flecha indicando la entrada al estadio Miraflores, para indicar el final del estacionamiento, la cual se ubica en el comienzo de la vista del estacionamiento.

### 5.2.2. Vista de barra navegadora superior de estacionamientos

Al momento de insertar la barra navegadora superior, que permite alternar entre el despliegue de estacionamientos en la aplicación, solo se consideró, que esta cumpla esa función. Sin embargo, existe la posibilidad de que el usuario quiera estacionarse en las afueras del edificio 8000, el cual puede contar con todas las plazas ocupadas y al seleccionarlo, al navegar por él, este se informara luego de las acciones ya mencionadas, que ese estacionamiento está lleno. Debido a esta posibilidad, en la barra superior y en cada estacionamiento, se insertó el número de plazas disponibles con las cuenta. De esta manera, el conductor solo seleccionara las opciones que cuentan con disponibilidad, y así se reducirá la interacción en usuario y aplicación. En las figuras 65 y 66 se muestran el antes y después de aplicar el cambio de visualización de plazas disponibles en la barra navegadora superior.

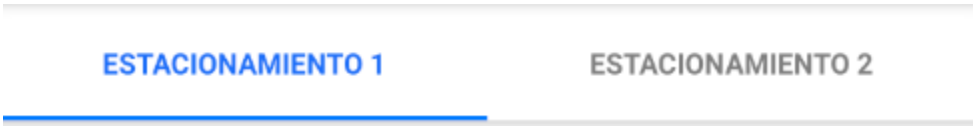



Figura 65: Barra navegadora superior anterior

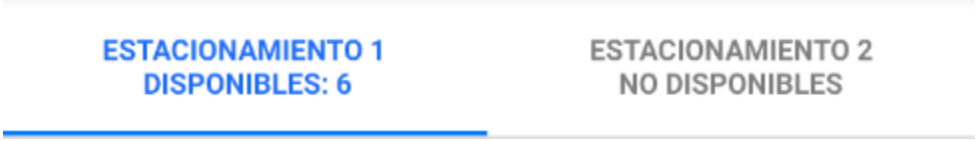



Figura 66: Barra navegadora superior actual

## 5.3. Sistema Completo

Una vez testeados tanto el prototipo IoT, como la aplicación móvil, llega la hora de evaluar el funcionamiento del sistema trabajando conjuntamente, por lo que con los microcontroladores funcionado correctamente y ubicándolos en sus lugares respectivos y la aplicación móvil siendo ejecutada en un smartphone, se procedió a probar que los cambios en el estacionamiento se veían reflejados en la aplicación móvil. La metodología que se utilizo fue estacionar el vehículo 20 veces en el estacionamiento y verificar en la aplicación que los cambios se veían reflejados. El resultado fue positivo, pues como se mencionó anteriormente, la aplicación y la base de datos cuentan con un nodo de comunicación, en el cual, ante algún cambio efectuado en la base de datos, esta notifica a la aplicación de los cambios en los datos para que así la aplicación actualice la información a renderizar. Mediante esta prueba se verifico el correcto funcionamiento en tiempo real de la aplicación frente a los cambios de estado percibidos en el estacionamiento por el prototipo IoT.

# 6. VALIDACIÓN

El sistema desarrollado necesita pasar por una prueba de validación para medir el grado de confiabilidad que esta presenta, si se decidiera escalar esta solución a un sistema más complejo. Debido a que no se cuenta con sistemas similares como el Smart Parking Node de Libelium para realizar una validación práctica, se realizó una revisión bibliográfica para validar esta solución de forma teórica, seleccionando 2 trabajos de investigación o artículos confiables, en los cuales se haya llevado a cabo una solución similar a la desarrollada en este proyecto.

En el primer trabajo de investigación seleccionado (Kodali, R. K., Borra K. Y., G. N., S. S., & Domma, H. J., 2018) se desarrolló un sistema muy similar, en el cual usan como placa de desarrollo el TTGO LoRa 32 ESP32, solo sensores de proximidad HC-SR04 y no se menciona el método de sustento energético de los módulos. Se plantea que un módulo emisor, se conecte a 3 sensores HC-SR04, y cada sensor estará ubicado en una plaza de estacionamiento. Por lo que un módulo emisor, cubre 3 slot de estacionamiento. En cuanto al almacenamiento de la información, se utilizó IBM Watson Internet of Things. El usuario para visualizar el estado del estacionamiento debe ingresar a un enlace a través de su smartphone, en el cual se desplegará la información correspondiente

Como se puede observar en el sistema desplegado por los investigadores, utilizan sensores de proximidad para la detección vehicular, también utilizan la red LoRa para la comunicación entre los módulos encargados de detectar vehículos y el receptor, que se encargara de subir la información de estos a la base de datos. Concluyen que, mediante esta metodología, los conductores solucionaran sus problemas a la hora de encontrar slot de estacionamientos vacíos.

El segundo y último trabajo de investigación escogido (Floris A., Girau R.,Porcu S., Pettorru G. & Atzori L., 2020) se desarrolla e implementa un sistema de Smart Parking basado en el magnetómetro. Este sistema es más complejo que el que se llevó a cabo en este proyecto, ya que se busca almacenar el ID del conductor que está utilizando el slot de estacionamiento, para que este pueda visualizar mediante una aplicación móvil el consumo y posteriormente pagar por este medio. Para la detección del automóvil utilizan un Arduino conectado a un Sensor Magnetómetro, la información recolectada por los módulos detectores es enviada al haber cambios de estado, a un “Telit Device”, el cual mediante bluethooth obtiene el identificador del vehículo y envía a un “concentrador” por medio de WiFi, el id de vehículo, estado del slot que cambio, el estado de batería del módulo detector de automóviles y el tiempo en que comienza el consumo por parte del conductor. En este concentrador se recolecta los datos de todos los slot de estacionamiento y se envían por medio de una red de largo alcance a una capa de virtualización, que representa un objeto social virtual por cada módulo detector de vehículos del estacionamiento. Posteriormente se habilita esta información para ser usada en el próximo nivel, denominado nivel de agregación, en el cual se cuentan con funcionalidades de procesamiento de datos que permitirán generar análisis de estadística, como por ejemplo la ocupación de una cierta área de estacionamientos. Finalmente se tiene el nivel aplicativo, que provee al usuario, métodos de pago, comunicación con la aplicación móvil

y la visualización del estado del estacionamiento. En cuanto al sistema de detección vehicular, ellos utilizan el vector magnitud del campo magnético y no los ejes X - Y - Z. Además, realizaron un estudio con 15 diferentes tipos de vehículo, con el fin de estudiar como estos afectaban las mediciones del sensor, en sus resultados se reflejó que existen vehículos que afectan de distinta forma el campo magnético, la cual puede estar asociada con el tamaño y la cantidad de metal por el que está compuesto. A pesar de haber autos que afectan en menor medida el campo magnético del sensor, concluyen que es posible detectar la presencia vehicular utilizando el magnetómetro.

Claramente es un sistema con una mayor complejidad, sin embargo, se utilizan las mismas bases en cuanto a la detección vehicular, sin contar el sensor de proximidad. Se demuestra que el uso del magnetómetro es confiable para detectar presencia vehicular, el sistema es escalable con métodos de pago y que el usuario puede visualizar el estado de las plazas del estacionamiento mediante una aplicación móvil sin problemas.

Con los estudios mencionados previamente, es posible señalar que la solución desarrollada en este proyecto es confiable, pues el módulo emisor, encargado de la detección vehicular, utiliza sensores con un grado de error muy bajo, y que son comunes en el desarrollo de Smart Parking. Por otro lado, se demuestra que este proyecto es escalable con un sistema de pago, si se implementan las tecnologías correctas y que informar a los conductores sobre el estado de las plazas de los estacionamientos, es claramente beneficioso, pues consumen menos recursos en la búsqueda de estos.

# 7. LIMITACIONES

En este capítulo se presentan las limitaciones que se detectaron en el desarrollo del proyecto.

## 7.1. Sensor de proximidad

Como se mencionó en las características del sensor de proximidad JSN-SR04T, pese a ser impermeable, lo cual lo hace ideal para sitios abiertos que están sujetos a condiciones climáticas como la lluvia, tiene un zona muerta de 20 [cm], es decir, que en un rango de 0 [cm] a 20 [cm], las mediciones serán inconsistentes. Debido a esta limitaciones, en aquellos autos en que el chasis se encuentre a un nivel bajo del suelo, menor a 20 [cm], el prototipo no podrá detectarlo, por la limitación del sensor. Sin embargo, en el mercado internacional, existe una versión mejorada de este sensor, denominado JSN-SR20-Y1, el cual incorpora dos sensores de detección impermeable y no solo uno. Las características generales son aproximadamente las mismas, con la excepción que la zona muerta es de 1 [cm], permitiendo mediciones entre los 2 [cm] y los 500 [cm]. En la Figura 67 se puede ver el sensor mencionado anteriormente.

Figura 67: Sensor JSN-SR20-Y1

## 7.2. Red (LoRa y WiFi)

Como se mencionó previamente, si se quisiera implementar un Smart Parking con múltiples dispositivos emisores encargados de detectar la presencia vehicular y enviar el estado de la plaza en el que se encuentran ubicados, es necesario la adquisición de un Gatweay LoRa. Existen múltiples tipos de Gateways y se diferencian unos de otros principalmente por la cantidad de nodos emisores soportados, los hay de 300 a 1000 nodos, e incluso un poco más. En cuanto a valor monetario, un Gateway que soporta 300 nodos ronda el precio de unos $80.000 CLP y uno que soporta 1000 unos $220.000 CLP. En cuanto al alcance que tienen los dispositivos, los resultados entregaron una distancia de 200 [mt] en una zona urbana, lo cual resulta ser positivo para un estacionamiento con una cantidad de plazas considerables. Sin embargo, en estacionamientos cerrados y con

más de 1 piso, es probable que sea vea afectada la señal, debido al material por el que están construidos, el cual suele ser el concreto. Cabe destacar, que, instalando un Gateway, el alcance de la red incrementara, pues cubre un alcance mayor al de los módulos usados en el proyecto, llegando a aproximadamente a 3 kilómetros (SEMTECH, 2018). En base a esto, dependiendo de la cantidad de plazas que se quieran automatizar, será necesario la adquisición de uno o más Gateways para la recepción de información.

Ya que el módulo receptor, está conectado a WiFi, es necesario que este se encuentre a una distancia razonable del router al que se conectara para así subir la información a la base de datos. Es por esto, que teniendo en cuenta que se instalará en una zona urbana, el módulo receptor, deberá ubicarse a unos 30 [m] de distancia del router (Decker F, s.f). Si se utilizara un Gateway, en caso de utilizar WiFi, se utiliza el mismo criterio. Cabe destacar que los Gateway pueden conectarse a Internet por WiFi, Cable LAN o por GSM mediante 3G o 4G.

En la Figura 68 se puede visualizar un diagrama de red entre nodos emisores, Gateway, base de datos o servidor en la nube y la capa de aplicación en donde se mostrará la información de esos nodos emisores.

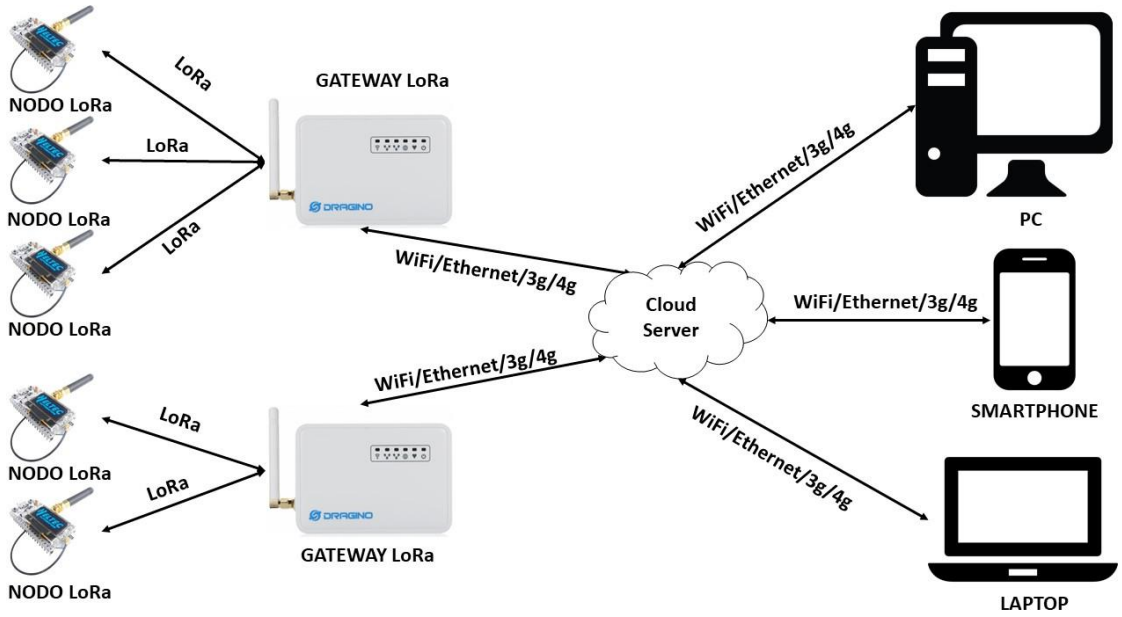



Figura 68: Diagrama de conexión entre nodos finales y capa aplicativa, utilizando Gateways

## 7.3. Base de datos

En cuanto a la base de datos en tiempo real de Firebase utilizado en el proyecto, esta ofrece 1 [GB] de almacenamiento y un total de 10 [GB] por mes de descargas de la base de datos en el plan gratuito. Los estacionamiento creados en la base de datos, cuentan cada uno con 10 slot de estacionamientos, utilizando un total de 255 [Bytes]. Si se escalara el sistema, aumentando el número de slots, sería posible ocupar el almacenamiento completo con aproximadamente 74.431.372 slots, lo cual es una cifra bastante positiva, en la Tabla 18 se puede visualizar el tamaño de la base de datos, a medida que aumenta el número de slots.

Tabla 18: Tamaño de la base de datos, a medida que se agregan slots de estacionamiento

| **Slots** | **Tamaño** |
|---:|---|
| 20 | 255 B |
| 100 | 1,275 KB |
| 1.000 | 12,75 KB |
| 10.000 | 127,5 KB |
| 100.000 | 1,275 MB |
| 1.000.000 | 12,75 MB |
| 10.000.000 | 127,5 MB |
| 74.431.372 | 1 GB |

Sin embargo, las limitaciones comienzan al momento de realizar las descargas de datos, mediante la aplicación móvil. Ya que los escenarios de uso son innumerables, se considerará lo siguiente: se cuenta con un estacionamiento de 1000 slots, por lo tanto, el tamaño de este en la base de datos es de 12,75 [KB]. Además, se cuenta con una persona que se encuentra registrada en la aplicación, y cuenta con un avatar en la aplicación, el cual pesa 1 [MB] en la base de datos. Esta persona si utiliza la aplicación una vez al día, que será solo para encontrar una plaza vacía, descargara 1,013 [MB] aproximadamente, y si la usaran 500 personas en un día, se descargarían 506.500 [MB]. Continuando con esta situación, en menos de 20 días, se utilizaría la cuota entregada por firebase y por ende sería necesario contratar un plan de pago para que el sistema pueda ser utilizado por más usuarios durante un mes.

## 7.4. Material de carcasa

El recubrimiento del prototipo emisor esta hecho de filamento para impresoras 3D, el cual fue hecho a medida con el fin de llevar a cabo las pruebas en terreno con una mayor comodidad. Sin embargo, si se utilizara en pruebas reales, exponiéndolo a un estacionamiento tanto abierto como cerrado, este fácilmente se quebraría, en caso de que algún vehículo pasara por encima de este o incluso la lluvia en un estacionamiento abierto, podría filtrarse en la carcasa, dañando el circuito electrónico por el que está compuesto. Por lo tanto, es importante tener en consideración para las pruebas en entorno real, utilizar un material resistente al agua y a golpes, como por ejemplo un tope de estacionamiento de caucho, como el que se muestra en la Figura 69. Este es posible adaptarlo para así insertar prototipo en este, y colocarlo en la posición donde este hará las lecturas pertinentes, como lo ilustrado en la Figura 70.

Figura 69: Tope de estacionamiento de 50 [cm]


Figura 70: Tope de estacionamiento en la posición donde hará mediciones

# 8. MEJORAS

Como se mencionó en el capítulo 5, correspondiente a las pruebas del sistema, el prototipo emisor no utiliza mecanismos de ahorro de energía, de envío de datos redundantes o también de obtención automática de parámetros previo funcionamiento, por lo que a continuación se implementarán estas mejoras con la finalidad que el algoritmo del módulo emisor, utilice de manera eficiente el hardware de este.

## 8.1. Calculo Automático de campo magnético

Al momento de hacer las pruebas correspondientes a la medición del campo magnético, primero se coloca el prototipo en el lugar de medición, se obtiene la media de las componentes X e Y, para posteriormente, en el algoritmo de detección vehicular, utilizar estas variables como las globales, en las cuales, si la medición actual tiene una variación de 150 unidades con la global, el objeto percibido es un vehículo. Por lo que, si se decidiera mover el prototipo a otro slot de estacionamiento, deberá ser necesario realizar nuevamente las mediciones de fuerza magnética en ese lugar, y cargarlas al algoritmo de detección. Por lo que se implementó un algoritmo que permite al emisor, detectar de forma automática el campo magnético, una vez colocado en el lugar de medición.

Se crearon dos variables globales con respecto al campo magnético, denominadas `x_global` e `y_global` y mediante la función `setup()`, se realizó la medición del campo magnético 20 veces en las componentes x e y, se normalizo el valor promediándolo y fueron asignadas a `x_global` e `y_global`. De esta manera, en la función `loop()`, cada vez que realice una nueva medición, esta será comparadas con las variables globales de campo magnético previamente calculadas.

En la Figura 71 se muestra mediante pseudocódigo, el funcionamiento de esta optimización.

```
int x_global;
int y_global;
Function setup():
    LoRaInit();
    getMagneticForce(x_global,y_global);
return
Function loop():
    //sin cambios
return
```

Figura 71: Pseudocódigo calculo automático campo magnético

## 8.2. Optimización en el envío de datos

El algoritmo cargado en el módulo emisor, realiza envíos de datos cada 10 segundos, por lo que no se discrimina si en caso de que la información se vuelva redundante, no haga envíos sobre el estado actual.

Por lo que se creó una optimización que consiste en definir una variable de estado global, llamada `state_global`, la cual comienza en 0, pues cuando el módulo se coloque en la ubicación de la medición, no habrá un vehículo estacionado. Luego en la función `loop()`, una vez que se ha realizado la detección del vehículo, se procede a comparar el estado global con el estado actual, entregado de la detección, si son distintos, entonces se realiza el envío de datos y si ambos son iguales, no se realiza el envío de datos.

En la Figura 72 se muestra mediante pseudocódigo la optimización realizada.

```
int state_global = 0;
Function setup():
    //optimizacion 8.1 agregada
return
Function loop():
    int distance x, y, state;
    getDistance(distance);
    getMagneticForce(x,y);
    if distance2 < 30 then
        if (abs(x − x_global) > 150) and ((abs(y − y_global) > 150) then
            state ← 1;
        else
            state ← 0;
        end
    else
        state ← 0;
    end
    if state_global != state then
        state_global = state;
        sendLoraPacket();
    end
return
```

Figura 72: Pseudocódigo de optimización de envío de datos

## 8.3. Optimizaciones en el ahorro de energía

Los microcontroladores como el ESP32, mediante el cual están basados los módulos Heltec WiFi LoRa 32 (v2), incorporan distintos modos de ahorro de energía, que permiten mejorar sustancialmente la autonomía de estos, si se considera usar como fuente de energía algún tipo de batería. En la Tabla 19 se muestran los modos de energía disponibles a estudiar e implementar para optimizar el uso de energía en el prototipo emisor (RandomNerdTutorials, s.f).

Tabla 19: Comparación en modos de energía del ESP32

| **Modo Consumo** | **Activo** | **Modem-Sleep** | **Light-Sleep** | **Deep-Sleep** | **Hibernación** |
|---|---|---|---|---|---|
| **CPU y Memoria** | Prendido | Prendido | Pausada | Apagado | Apagado |
| **WiFi/BT/LoRa** | Prendidos | Apagados | Apagados | Apagado | Apagado |
| **RTC (memoria y periféricos)** | Prendidos | Prendidos | Prendidos | Prendidos | Memoria rtc Apagada y Periférico rtc Prendido |
| **ULP Coprocesador** | Prendido | Prendido | Prendido | Apagado o Prendido | Apagado |

A partir de la tabla, la mejor opción a implementar es el modo Deep-Sleep, pues la mayoría de los componentes se apagan, con excepción del RTC, con esto es posible utilizar variables globales instanciadas en la memoria RTC sin que estas se eliminen al reiniciarse luego del descanso. La opción de hibernación no es posible llevarla a cabo, pues el funcionamiento del prototipo depende de variables globales que no pueden reiniciarse.

En cuanto a la optimización del algoritmo, esta se puede observar en formato pseudocódigo en la Figura 73. Será necesario adaptarlo para utilizar la opción de Deep-Sleep. Cabe destacar que cuando el dispositivo vuelve a funcionar, se ejecuta tanto la función `setup()` como `loop()`. En la función `setup()`, fue necesario adaptar la optimización de cálculo automático de campo magnético, pues después de cada Deep-Sleep `x_global` e `y_global`, se reiniciarían, provocando inconsistencias en las lecturas. A partir de aquí, estas variables globales, pasaron a instanciarse a la memoria RTC para que no pierdan su valor. Otro punto a tener en cuenta es que este cálculo se llevara a cabo cada vez que se ejecute `setup()`, y solo debe ejecutarse la primera vez que se enciende el prototipo, por lo que se declaró un booleano en la memoria RTC denominado `measure`, el cual, si es falso, realiza el cálculo y luego cambia su valor a verdadero con el fin de que en las ejecuciones posteriores no se vuelva a llevar a cabo.

Respecto a la optimización del envío de datos, también fue necesario adaptar algunos parámetros, como lo es la variable `state_global`, que se instancio en la memoria RTC, para que luego de cada ejecución post descanso, no se reinicie su valor a 0 y su valor luego de cada detección vehicular se mantenga consistente.

Ya con las optimizaciones adaptadas, es necesario definir el tiempo por el que el dispositivo descansará y para esto, se utilizará un mecanismo que comienza con 10 segundos de Deep-Sleep, y este aumenta en 10 segundos, si el estado actual es igual al global, por lo que se declaró en la memoria RTC la variable `multiplier`. El aumento máximo llega a los 60 segundos ya que es una cifra que no afectara la visualización en tiempo real del estado del slot de estacionamiento y puede estar sujeta a cambios en investigaciones futuras. Para aumentar la variable `multiplier`, se utilizó la función `constrain()` que aumenta una variable en un valor definido y hasta un valor definido, en este caso aumenta en 1 y hasta un valor de 6. En caso de que el estado actual y global sean distintos, entonces el tiempo de descanso se reiniciara a los 10 segundos y volverá a iniciarse el mecanismo.

```
RTC int state_global = 0;
RTC int x_global;
RTC int y_global;
RTC char multiplier = 1;
RTC bool measure = false;
int time = 10;
Function setup():
    Lora.Init();
    if measure = false then
        getMagneticForce(x_global,y_global);
        measure = true;
    end
return
Function loop():
    int distance x, y, state;
    getDistance(distance);
    getMagneticForce(x,y);
    if distance2 < 30 then
        if (abs(x − x_global) > 150) and ((abs(y − y_global) > 150) then
            state ← 1;
        else
            state ← 0;
        end
    else
        state ← 0;
    end
    if state_global != state then
        state_global = state;
        sendLoraPacket();
        multiplier = 1;
    else
        multiplier = constrain(multiplier + 1, 1, 6)
    end
    timeSleep(TIME*multiplier);
    startDeepSleep();
return
```

Figura 73: Pseudocódigo implementación de Deep-Sleep

Para lograr obtener un estimado de la duración de la batería con esta optimización implementada, es necesario llevar a cabo un análisis elaborado y estadístico, considerando que los eventos que afectarán el consumo energético del módulo emisor, serán la llegada

o la salida de un vehículo. Se define la variable aleatoria *k* como el numero promedio de eventos en una hora, en donde un evento es considerado la llegada o salida del slot de estacionamiento. Se asume que esta variable aleatoria pertenece a una distribución Poisson con parámetro *k*. Además, se consideran dos casos para el consumo energético: el primer caso corresponde cuando el módulo utiliza el máximo de energía posible, es decir, utiliza los sensores y envía el paquete por la red LoRa. En el segundo caso se considera que el módulo está en modo Deep-Sleep, consumiendo 0.8 mA según las especificación técnicas de la placa de desarrollo. Para simplificar el análisis se considera que cada vez que ocurre un evento el módulo estará en su modo de consumo máximo durante 2 minutos, ya que este es el tiempo aproximado que tarda en entrar a su modo de máximo ahorro de energía. Para obtener el consumo promedio en este intervalo de una hora, se puede utilizar el valor esperado de la variable aleatoria para obtener una aproximación del número de cambios de estado que sufre el slot, por lo que, mediante la siguiente expresión matemática mostrada en la Figura 74 es posible estimar el consumo de energía en mAh.

$$(x) * \left(\frac{2}{60} * E[k]\right) + (y) * (1 - (\frac{2}{60} * E[k]))$$

Figura 74: Ecuación del consumo ponderado entre los dos estados

En la expresión, "x" corresponde a los mAh en modo máximo consumo de energía, en cuanto a "y", corresponde a los mAh en modo máximo ahorro. $E[k]$ es el valor esperado de la variable aleatoria *k*. En la expresión $\frac{2}{60} * E[k]$ corresponde a la proporción del tiempo en que el módulo estará en estado máximo consumo de energía y $1 - (\frac{2}{60} * E[k])$ a la proporción del tiempo en que el módulo estará en estado ahorro máximo. Se utiliza $\frac{2}{60}$ ya que es la unidad mínima de tiempo en que se produce un cambio de estado según este modelo. Reemplazando en la formula se obtiene el siguiente resultado mostrado en la Figura 75.

$$6 * \left(\frac{2}{60} * 2\right) + 0.8 * \left(1 - \left(\frac{2}{60} * 2\right)\right) = 1.146\ mAh$$

Figura 75: Desarrollo de la ecuación matemática

El resultado obtenido es claramente menor al utilizado en el algoritmo que no presenta optimizaciones, y la mejora lograda es de hasta 500%, llegando a un aproximado de 182 días de autonomía, que representa medio año en funcionamiento sin necesidad de recargar la batería. Si se utilizara una powerbank de 10.000 mAh, se alcanzaría la autonomía de un año completo de uso sin recargas. Es importante mencionar que este cálculo es el peor caso, ya que el módulo no pasa los dos minutos completos consumiendo los 6 mAh, sino que, la disminución del consumo es gradual hasta llegar a los 0.8 mAh.

# 9. CONCLUSIONES Y TRABAJO FUTURO

## 9.1. Conclusión

Cabe destacar que el objetivo general o principal de este proyecto de título es: *"Desarrollar un prototipo de Estacionamiento inteligente que utilice tecnología innovadora y sustentable, que permita a los conductores mejorar su experiencia de conducción a la hora de buscar un lugar donde aparcar"*, el cual se cumplió de manera satisfactoria, empleando el conocimiento y las herramientas correctas, adquiridas durante la formación universitaria del tesista.

En los puntos siguientes, se indican los objetivos específicos con los que se debía cumplir, conforme se avanzó en el desarrollo del proyecto y si lograron cumplirse.

- Investigar la situación actual de soluciones IoT implementadas en Smart Parking, principalmente aquellas que utilizan red LoRa o LPWAN.

  Mediante la revisión bibliográfica, fue posible lograr este objetivo, ya que se hallaron distintas soluciones e innovaciones, cada una con su característica que lo diferenciaba de los demás artículos, lo que permitió obtener el conocimiento necesario para realizar un proyecto que se ubique a la vanguardia del Smart Parking, se implemente optimización en el uso de recursos y emplee el uso una red LPWAN como lo es LoRa.

- Seleccionar la tecnología tanto IoT como software, disponible en la actualidad, para llevar a cabo un prototipo completo de Smart Parking, post evaluación de estas tecnologías.

  Tal como se menciona en el capítulo 3, se llevó a cabo una evaluación de las tecnologías necesarias para el desarrollo de un sistema de Smart Parking disponibles actualmente, en la que se mencionaron las placas de desarrollo, sensores, baterías, redes, bases de datos y *frameworks* de programación de aplicaciones móviles. En las cuales se menciona y detalla cual es la mejor opción para utilizar en el desarrollo del proyecto, por lo que este objetivo logro ser efectuado con satisfacción.

- Desarrollar un prototipo IoT que detecte un automóvil y que utilice la red LoRa para enviar los cambios de estado captados por los sensores.

  El prototipo IoT, fue llevado a cabo con éxito, pues tal como se menciona en el capítulo 5, tanto el módulo emisor como receptor fueron desarrollados e implementados mediante las tecnologías seleccionadas, por lo que tanto el emisor, encargado detectar un automóvil y enviar el estado de la plaza en el que encuentra ubicado, se comunica sin inconvenientes con el módulo emisor mediante la red LoRa y este último al recibir los datos correspondientes, los sube a la base de datos.

- Desarrollar una aplicación móvil de Smart Parking que sea intuitiva, segura y simple para el conductor.

  Producto de la pandemia provocada por el virus COVID-19, no fue posible realizar las pruebas de la aplicación móvil con una cantidad de usuarios considerables, lo que permite obtener retroalimentación suficiente sobre el rendimiento en terreno de esta. Sin embargo, las pruebas validadas por el profesor patrocinante permiten afirmar que la aplicación móvil cumple con las condiciones necesarias para ser utilizado por un conductor.

- Probar el sistema completo en condiciones de laboratorio.

  El sistema completo fue probado en el estacionamiento del tesista, realizando las acciones pertinentes con un automóvil, entregando resultados positivos en cuanto a precisión en la detección del vehículo. Además, el funcionamiento tanto del prototipo IoT como la aplicación móvil, respondían correctamente en las pruebas realizadas.

- Validar el sistema completo, comparándolo con soluciones existentes.

  La validación fue llevada a cabo de forma parcial, pues solo fue llevada a cabo la validación basada en la comparativa con artículos que hayan implementado una solución similar a la desarrollada en este proyecto. En base a los resultados de las pruebas y las comparativas, la validación teórica se completó de manera satisfactoria. En cuanto a una validación basa en comparativa práctica, no fue posible llevarla a cabo, debido a que el costo para adquirir una solución como la comercializada por Libelium "Smart Parking Node", tiene un costo elevado. Sin embargo, no se descarta la posibilidad de que, a largo plazo, sea posible la adquisición de este con el fin de validar el sistema completo de manera práctica.

## 9.2. Trabajo Futuro

Este proyecto al desarrollar e implementar un prototipo, está sujeto a posibles mejoras que podrían aplicarse en trabajos posteriores, con el fin de escalar a un sistema más completo y que mejoraran la calidad de este. Estas mejoras se nombrarán a continuación:

- Realizar pruebas con una variedad considerable de vehículos (incluyendo motocicletas), pues de esta manera es posible medir la verdadera precisión del prototipo.
- Investigar sobre las tecnologías necesarias para escalar este prototipo con sistemas de pago, con el fin de implementar este proyecto en estacionamiento privados.
- Investigar materiales que se ajusten a las necesidades del prototipo emisor, para que así al ser instalado en el slot de estacionamiento, este no sufra daños.
- Estudiar la intensidad de señal en estacionamientos cerrados, pues en estos suele haber un bloqueo de señal significativo por el material en el que están construidos.

- Utilizar más de un prototipo emisor, empleando un Gateway como enlace, para subir los datos a Firebase.
- En la base de datos, recibir no solo el estado del módulo, sino también la lectura de los sensores, de esta manera es posible monitorear el correcto funcionamiento de estos.
- Estudiar el procesamiento de datos como herramienta para medir parámetros estadísticos con el fin obtener información sobre: sectores de estacionamiento que tienden a ser los más utilizados, horas *peak* de utilización de plazas de estacionamiento, promedio de uso de las plazas de estacionamiento, entre otras.
- Investigar a fondo el ahorro de energía en el dispositivo, ya que la solución implementada a pesar de optimizar el consumo de energía puede no ser el óptimo.
- Implementar el cambio del sensor de proximidad, pues este tiene una zona muerta de 20 [cm] y existe en el mercado internacional una versión mejorada del mismo, que su zona muerta es de 1 [cm], permitiendo mediciones entre 2 [cm] y 500 [cm].
- Implementar escalamiento de la aplicación móvil en *Android Auto* y *Apple Carplay*, pues en aquellos vehículos que cuenten con esta tecnología, la visualización de la aplicación en su vehículo será aún más cómoda y segura.

## 10. REFERENCIAS


Actitud Ecológica, (2020). *Tipos de pilas: guía completa con las pilas y baterías que existen*. Recuperado el 2 de Marzo de 2021 de https://actitudecologica.com/tipos-de-pilas/#2_Pilas_recargables

Anderson B. & Nicholson B. (2020). *SQL vs. NoSQL Databases: What's the Difference?* Recuperado el 7 de Marzo de 2021 de https://www.ibm.com/cloud/blog/sql-vs-nosql

ANKER, (2018). *PowerCore 5000*. Recuperado el 2 de Marzo de https://us.anker.com/products/a1109

AWS Amazon (s.f). *¿Qué es una base de datos relacional?*. Recuperado el 7 de Marzo de 2021 de https://aws.amazon.com/es/relational-database/

AWS Amazon (s.f). *¿Qué es NoSQL?*. Recuperado el 7 de Marzo de 2021 de https://aws.amazon.com/es/nosql/

Baeza M (2018). *Este es el último récord mundial del automóvil*. Recuperado el 5 de febrero de 2021 de https://motor.elpais.com/actualidad/el-ultimo-record-mundial-del-automovil/#:~:text=De%20los%20892%20millones%20de,autom%C3%B3vil%20por%20cada%20seis%20personas

Barriga, J., Sulca, J., León, J., Ulloa, A., Portero, D., Andrade, R., Yoo, S (2019). Smart Parking: *A Literature Review from the Technological Perspective*. Recuperado el 25 de Febrero de 2021 de https://www.mdpi.com/2076-3417/9/21/4569

Bertino, E., Choo, K.-K. R., Georgakopolous, D., & Nepal, S. (2016). *Internet of Things (IoT)*. Recuperado el 5 de Febrero de 2021 de https://dl.acm.org/doi/10.1145/3013520

Burgos G (2017). *Chile: Con esta innovación hallar estacionamiento será más fácil*. Recuperado el 11 de Febrero de 2021 de https://www.america-retail.com/chile/chile-con-esta-innovacion-hallar-estacionamiento-sera-mas-facil/

Cisco (s.f). *What is Wi-Fi*. Recuperado el 20 de Febrero de 2021 de https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html

Decker F (s.f). *¿A qué distancia todavía funciona un router inalámbrico?* Recuperado el 1 de Junio de 2021 de https://pyme.lavoztx.com/qu-distancia-todava-funciona-un-router-inalmbrico-7195.html

ELEC Freaks (s.f). *Ultrasonic Ranging Module HC - SR04*. Recuperado el 1 de Marzo de 2021 de https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Espressif (2020). *ESP8266 Datasheets*. Recuperado el 12 de Febrero de 2021. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Espressif (2021). *ESP8266 Datasheets*, Recuperado el 12 de Febrero de 2021. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

Firebase (2020). *Firebase Realtime Database*. Recuperado el 10 de Marzo de https://firebase.google.com/docs/database?hl=es-419

Firebase (2021). Cloud Firestore. Recuperado el 10 de Marzo de https://firebase.google.com/docs/firestore?hl=es-419

Firebase (2021). *Elige una base de datos: Cloud Firestore o Realtime Database*. Recuperado el 10 de Marzo de 2021 de https://firebase.google.com/docs/database/rtdb-vs-firestore?hl=es-419

Floris A., Girau R.,Porcu S., Pettorru G. & Atzori L. (2020). *Implementation of a Magnetometer based Vehicle Detection System for Smart Parking applications*. Recuperado el 17 de Mayo de 2021 de https://www.researchgate.net/publication/346569510_Implementation_of_a_Magnetometer_based_Vehicle_Detection_System_for_Smart_Parking_applications

Giraldo V. (219). *¿Ya conoces Firebase? La herramienta de desarrollo y análisis de aplicaciones mobile*. Recuperado el 10 de Marzo de https://rockcontent.com/es/blog/que-es-firebase/

Gorka, R. (2019). *¿Apps híbridas o apps nativas? Un breve análisis comparativo de tecnologías móviles*. Recuperado el 14 de Marzo de 2019 de https://blog.irontec.com/apps-hibridas-vs-apps-nativas-un-breve-analisis-comparativo-de-tecnologias-moviles/

Heltec (2018). *WiFi LoRa32 (v2)*. Recuperado el 12 de Febrero de 2021 https://heltec.org/project/wifi-lora-32/

Honeywell (s.f). *3-Axis Digital Compass IC HMC5883L*. Recuperado el 1 de Marzo de https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

Ibañes J. (s.f). *NIVELES DE MADUREZ DE LA TECNOLOGÍA TECHNOLOGY READINESS LEVELS.TRLS. UNA INTRODUCCIÓN*. Recuperado el 2 de Abril de 2021 de https://www.mincotur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/393/NOTAS.pdf

INE (2019). *Parque de vehículos en circulación Región de los ríos*. Recuperado el 10 de Febrero de 2021 de https://regiones.ine.cl/documentos/default-source/region-xiv/estadisticas-r14/boletines-informativos/parque-de-veh%C3%ADculos/parque-de-veh%C3%ADculos-en-circulaci%C3%B3n---periodo-2019.pdf

Joy-IT (s.f). *KY-024 Linear magnetic Hall Sensor*. Recuperado el 1 de marzo de https://datasheetspdf.com/pdf-file/1402035/Joy-IT/KY-024/1

Kodali, R. K., Borra K. Y., G. N., S. S., & Domma, H. J. (2018). *An IoT Based Smart Parking System Using LoRa*. Recuperado el 15 de Mayo de 2021 de https://ieeexplore.ieee.org/document/8644697

Lee, C., Han, Y., Jeon, S., Seo, D., & Jung, I. (2016). *Smart parking system for Internet of Things.* Recuperado el 11 de Febrero de 2021 de https://www.researchgate.net/publication/301610593_Smart_parking_system_for_Internet_of_Things

Libelium (s.f). *Smart Parking*. Recuperado el 11 de Febrero de 2021 de https://www.libelium.com/iot-solutions/smart-parking/

LILYGO (s.f), *TTGO LoRa32 ESP32*, recuperado el 12 de Febrero de 20 http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1134&FId=t3:50003:3

López, G. & Margni, S. (2003), *Funcionamiento de microcontroladores*. Recuperado el 12 de Febrero de 2021, de https://www.fing.edu.uy/inco/grupos/mina/pGrado/construccion2003/Documentos/IntroduccionPics.doc

LoRa Alliance (s.f). *What is LoRaWAN® Specification.* Recuperado el 20 de Febrero de 2021 de https://lora-alliance.org/about-lorawan/

Maida E.G., Pacienzia J. (2015). *Metodologías de desarrollo de software*. Recuperado el 2 de Abril de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

MCI electronics (s.f). *Baterias*. Recuperado el 2 de Marzo de https://www.mcielectronics.cl/shop/category/electronica-y-robotica-prototipo-baterias-266

Mouchili, M. N., Aljawarneh, S., & Tchouati, W. (2018). *Smart city data analysis*. Recuperado el 6 de Febrero de 2021 de https://dl.acm.org/doi/10.1145/3279996.3280029

Naharro A. (2019). *Frameworks para desarrollo de aplicaciones móviles híbridas*. Recuperado el 14 de Marzo de 2019 de https://www.campusmvp.es/recursos/post/frameworks-para-desarrollo-de-aplicaciones-moviles-hibridas.aspx

RANDOM NERD TUTORIALS (s.f). *Power ESP32/ESP8266 with Solar Panels (includes battery level monitoring)*. Recuperado el 2 de Marzo de 2021 de https://randomnerdtutorials.com/power-esp32-esp8266-solar-panels-battery-level-monitoring/

RandomNerdTutorials (s.f). *ESP32 Deep Sleep with Arduino IDE and Wake Up Sources*. Recuperado el 5 de Junio de https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/

Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., & Riegg, A. (2016). *Internet of things patterns*. Recuperado el 5 de Febrero de https://www.iaas.uni-stuttgart.de/publications/INPROC-2016-46-Internet-of-Things-Patterns.pdf

Renesas (s.f). *ISL29125*. Recuperado el 2 de Marzo de https://www.renesas.com/us/en/products/sensor-products/light-proximity-sensors/ambient-light-sensors/ambient-light-digital-sensors/isl29125-digital-red-green-and-blue-color-light-sensor-ir-blocking-filter

Sanchez, R., Sanchez, J., Ballesta, J., Cano, M., Skarmeta, A. (2018). *Performance Evaluation of LoRa Considering Scenario Conditions*. Recuperado el 6 de Febrero de 2021 de https://www.mdpi.com/1424-8220/18/3/772

SEMTECH (2018). *Wireless RF Selector Guide*. Recuperado el 1 de Junio de 2021 de https://www.semtech.com/uploads/design-support/SG-SEMTECH-WSP.pdf

Seeed (s.f). *Grove - Light Sensor*. Recuperado el 1 de Marzo de https://wiki.seeedstudio.com/Grove-Light_Sensor/

THINBOX (s.f). *Proximity Sensor/Switch E18-D80NK*. Recuperado el 1 de Marzo de https://datasheetspdf.com/pdf-file/1311840/tinkbox/E18-D80NK/1

United Nations (2018). *World Population Prospects*. Recuperado el 5 de Febrero de 2021 de https://population.un.org/wup/

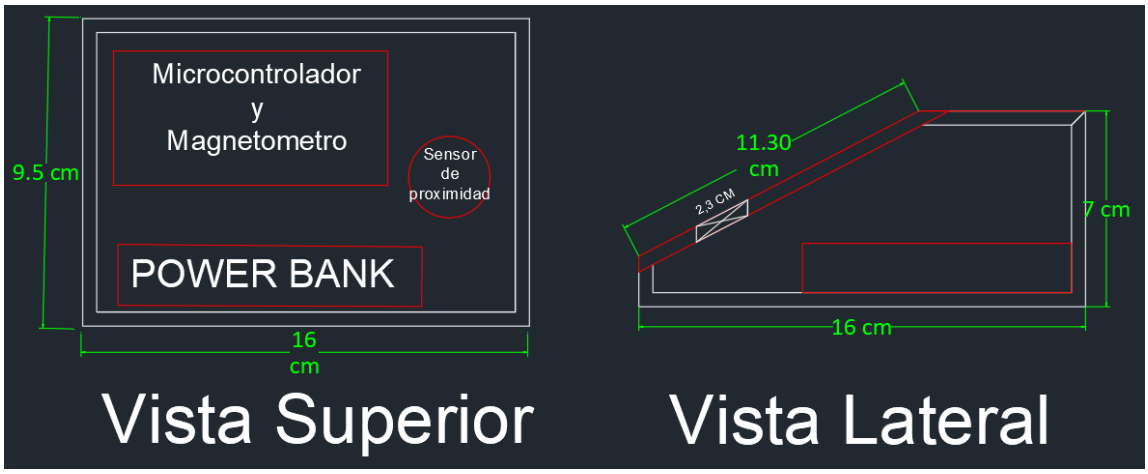
UNIT ELECTRONICS (s.f). *Sensor Ultrasónico Contra El Agua JSN-SR04T*. Recuperado el 1 de Marzo de https://uelectronics.com/producto/sensor-ultrasonico-jns-sr04t/

# ANEXOS

## Modelaje de Carcasas Prototipo IoT

El modelaje de la carcasa para el prototipo emisor y receptor fue llevado a cabo en el software Autocad. En las siguientes imágenes se detallan las medidas de ambas carcasas.

Planos de Prototipo Emisor:




Plano Prototipo Receptor: