UNIVERSIDAD INTERNACIONAL SEK

FACULTAD DE INGENIERÍAS Y CIENCIAS APLICADAS

Trabajo de fin de Carrera titulado:

Diseño de un sistema prototipo de uso de parqueaderos bajo disponibilidad, mediante IoT y sensórica

Realizado por:

Carlos Andrés Hidrobo Moreno

Director del proyecto:

PhD (C). Diego F. Bustamante V. Ing.

Como requisito para la obtención del título de:

INGENIERO EN MECATRONICA

QUITO, febrero del 2024

DECLARACIÓN JURAMENTADA

Yo, Carlos Andres Hidrobo Moreno, ecuatoriano, con Cédula de ciudadanía N° 1004809800, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido presentado anteriormente para ningún grado o calificación profesional, y se basa en las referencias bibliográficas descritas en este documento.

A través de esta declaración, cedo los derechos de propiedad intelectual a la

UNIVERSIDAD INTERNACIONAL SEK, según lo establecido en la Ley de Propiedad Intelectual, reglamento y normativa institucional vigente.

Carlos Andrés Hidrobo Moreno

C.I.: 1004809800

DECLARACIÓN DEL DIRECTOR DE TESIS

Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.

---------- PhD (C). Diego Fernando Bustamante Villagómez Ing.

LOS PROFESORES INFORMANTES:

XXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Después de revisar el trabajo presentado lo han calificado como apto para su defensa oral ante el tribunal examinador.

Ing. Jaime Molina

Ing. Gabriela Mancheno

Quito, 5 de abril de 2024

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

Carlos Andrés Hidrobo Moreno

C.I.: 1004809800

Agradecimientos

PÁGINA ES OPCIONAL.

Agradecimientos a quienes los apoyaron en el proceso

Agradezco a mis padres por siempre darme el apoyo incondicional en esta etapa de mi vida, por guiarme por un buen camino lleno de conocimiento y sabiduría para afrontar las adversidades que siempre se ha presentado en el camino, por no dejarme solo y enseñarme lo que es la vida.

Resumen

El proyecto de tesis titulado "Diseño de un sistema prototipo de uso de parqueaderos bajo disponibilidad, mediante IoT y sensórica" se enfocó en abordar la problemática de congestión vehicular en la Universidad Internacional SEK, campus Miguel de Cervantes, donde la ausencia de monitoreo de espacios disponibles generaba dificultades para los usuarios al buscar estacionamiento. El objetivo fue diseñar un prototipo de parqueo inteligente utilizando tecnologías de Internet de las cosas (IoT) y sensórica para implementar un sistema eficiente de búsqueda de lugares disponibles. Se empleó una metodología de investigación experimental con enfoque mixto, utilizando entrevistas y cuestionarios para recopilar la perspectiva de los docentes usuarios del estacionamiento. Se aplicó la metodología ágil Xtreme Programming (XP) para el desarrollo del prototipo, incluyendo diseño inicial, implementación iterativa, pruebas y refinamiento continuo. Los resultados incluyeron la visualización de flujo vehicular, el uso de códigos QR para acceso web, la integración de Telegram y correo electrónico, así como la implementación de un dashboard en Node-red. Se demostró la eficacia del prototipo con pruebas en la universidad, y se concluyó que mejora en un 14% la eficiencia en la búsqueda de estacionamiento.

Palabras clave: IoT, sensórica, parqueadero inteligente, código QR

Abstract

The thesis project entitled "Design of a prototype system for parking use under availability, using IoT and sensor technology" focused on addressing the problem of vehicular congestion at the SEK International University, Miguel de Cervantes campus, where the lack of monitoring of available spaces generated difficulties for users when looking for parking. The objective was to design a smart parking prototype using the Internet of Things (IoT) and sensor technologies to implement an efficient system to search for available spaces. An experimental research methodology with a mixed approach was employed, using interviews and questionnaires to collect the perspective of the teacher's parking users. Agile Xtreme Programming (XP) methodology was applied for prototype development, including initial design, iterative implementation, testing, and continuous refinement. Results included visualization of vehicular flow, use of QR codes for web access, Telegram and email integration, as well as the implementation of a dashboard in Node-red. The effectiveness of the prototype was demonstrated with tests at the university, and it was concluded that it improves parking search efficiency by 14%.

Keywords: IoT, sensorics, intelligent parking, QR code

Tabla de Contenidos

1. Introducción 10
1.1. Antecedentes 11
1.2. Planteamiento del problema 14
1.3. Justificación 15
1.4. Hipótesis 16
1.5. Objetivos del Trabajo de Tesis 16
1.5.1. Objetivo General 16
1.5.2. Objetivos Específicos 16
2. Estado del arte 17
2.1. Referentes Teóricos 19
2.1.1. Smart Parking basado en Raspberri Pi 19
2.1.2. La comunicación en la Iot 21
2.1.3. Protocolo MQTT 22
2.1.4. Node-RED 23
2.1.5. Cámaras ANPR Y LPR 24
2.1.6. Esp32 30
2.1.7. Protocolo WebSocket 31
1.7. XAMPP 32
2.1 Protocolo MQTT (Broker) 33
3. Metodología 36
3.1. Diseño de la Investigación 36
3.2. Variables de Estudio 36
3.3. Hipótesis de Estudio 37
3.4. Población y Muestra 37
3.5. Técnicas e Instrumentos de Recolección de Datos 38
3.6. Análisis de los Datos 38
3.7. Metodología de Desarrollo 40
3.7.1. Metodología Ágil Xtreme Programming 40
3.8. Desarrollo del Prototipo 41
3.8.1. Funcionamiento General del Prototipo 41
3.8.2. Comunicación rasberry pi y esp 32 42
3.8.3. Instalación de broker en la tarjeta Raspberry PI 3 43
3.8.4. Instalación de XAMPP 46
3.8.5. Página WEB 52
3.8.6. Diagramas 61
Resultados 65
Discusión de resultados 72
Conclusiones 75
Recomendaciones 77
Bibliografía 78

Lista de tablas

Tabla 1 *Tabla de datos por día* 62

Tabla 2 *Resultados de cada espacio en el día* 63

Lista de figuras

Figura 1. *Arquitectura del estacionamiento inteligente* 20

Figura 2. *MQTT - Broker* 21

Figura 3. *Cámara ANPR IP* 23

Figura 4. *Cámara LPR* 24

Figura 5. *Sensor ultrasónico I + D3* 27

Figura 6. *Sensor MPS* 27

Figura 7. *Protocolo MQTT* 33

Figura 8 *Comprobación de la versión de Raspbian*..... 42

Figura 9 *Arranque* 43

Figura 10 *Espera de la ejecución*..... 43

Figura 11 *Creación de nuevo bot* 46

Figura 12 *Asignación de un nombre y usuario* 46

Figura 13 *Obtención del token del bot* 47

Figura 14 *Opción de configuraciones* 48

Figura 15 *Instalación de nodos* 48

Figura 16 *Arrastre del nodo para configuración* 49

Figura 17 *Especificación del tipo de mensaje a enviar* 49

Figura 18 *Instalación del nodo e-mail* 49

Figura 19 *Configuración para el envío de mensajes* 50

Figura 20 *Formulario de estacionamiento* 51

Figura 21 *Parte inicial del programa* 51

Figura 22 *Gestión de contenidos de mensajes* 52

Figura 23 *Consultas en la base de datos* 53

Figura 24 *Gestor de Base de Datos Php* 53

Figura 25 *Actualización de información* 54

Figura 26 *Función de tiempo* 54

Figura 27 *Función de tarifa* 55

Figura 28 *Bloques e interfaz del programa* 55

Figura 28 *Menú del proyecto parqueadero* 56

Figura 30 *Lectura de datos – ESP32* 57

Figura 31 *Indicadores de ocupación de espacios* ..... 57 Figura 32 *Finalización del programa* 58

Figura 33 *Llamado de variables* 58

Figura 34 *Conexión de la primera función para envío por e-mail* 58

Figura 35 *Histograma de uso del parqueo* 63

Figura 36 *Flujo de vehículos registrados* 64

Figura 37 *QR para la página web de usuarios registrados* 65

Figura 38 *Qr para la obtención de ID en Telegram* 65

Figura 39 *Dashboard del control del parqueadero* 66

Figura 40 *Envío de mensajes al correo y Telegram de los usuarios* 66

Figura 39 *Dashboard del control del parqueadero* 67

Figura 42 *Diagrama de conexión del prototipo* 67

Figura 43 *Tiempos en llegada a los espacios* 68

Lista de Diagramas

Diagrama 1 *Diagrama de funcionamiento general* 40

Diagrama 2 *Diagrama de flujo – ESP-32* 59

Diagrama 3 *Diagrama de flujo – Página WEB* 60

Diagrama 4 *Diagrama de flujo – Node Red* ..... 61 **Lista de anexos**

Anexo 1 *Formato de Entrevista* 78

Anexo 2 *Evidencia del prototipo instalado en el área de estacionamiento* 80

Anexo 3 *Código de programación para el ESP 32* 82

1. Introducción

Según Cuzco y Pinos (2022), las ciudades experimentan un continuo progreso en la automatización, especialmente en lo que respecta a los sistemas de estacionamiento inteligente. Con el crecimiento demográfico, la demanda de vehículos ha ido en aumento, lo que impulsa la necesidad de adaptar los sistemas de estacionamiento a través de la Internet de las Cosas (IoT). Esta tecnología no solo optimiza la utilización del espacio urbano, sino que también contribuye a reducir la congestión vial y las emisiones de gases contaminantes. Además, los sistemas de estacionamiento inteligente ofrecen a los usuarios una experiencia más conveniente y eficiente al facilitar la búsqueda y reserva de espacios de estacionamiento en tiempo real, mediante aplicaciones móviles y otros dispositivos conectados.

Dentro del mismo contexto, en el artículo publicado por Tenorio et al. (2022), se menciona que el estacionamiento público, tanto dentro como fuera de las áreas urbanas, ha experimentado un aumento del 11% en su nivel de inteligencia, reflejando una creciente adopción de sistemas inteligentes. Se espera que esta tendencia continúe en ascenso, alcanzando un 16% para el año 2023. Esta evolución hacia la inteligencia aplicada al estacionamiento no solo mejora la experiencia del usuario al facilitar la búsqueda de espacios disponibles, sino que también contribuye significativamente a reducir la congestión vial y las emisiones de carbono al minimizar el tiempo de circulación en busca de estacionamiento, generando así beneficios tanto para la movilidad urbana como para el medio ambiente.

El estacionamiento avanzado incorpora diversas tecnologías, como sensores de identificación, parquímetros automatizados, tarifas de estacionamiento inteligentes, cámaras vehiculares, reconocimiento de placas, aplicaciones de navegación para asistencia en estacionamiento y señalización digital. Un sistema de parquímetro completo no únicamente proporciona una solución de extremo a extremo para los usuarios, sino que también permite el monitoreo de la ocupación de forma remota y automática en tiempo real por medio de un

operador. Más importante aún, el valor real de integrar elementos tecnológicos en los sistemas de estacionamiento radica en los datos de estacionamiento generados. Y cuando estos datos se combinan con datos de distribuidores, agentes y proveedores, los procesos y sistemas pueden conducir a una verdadera innovación en las ciudades inteligentes (Espinoza, 2022).

Esta investigación se centra en la Universidad Internacional SEK en el campus Miguel de Cervantes; el cual consta con tres áreas de parqueo la primera parte es la facultad de medicina, el segundo apartado es en la facultad de ingeniería y ciencias aplicadas y como tercer punto es el parqueadero del área administrativa ya que esta es el área al que más vehículos ingresan ya que profesores, personal administrativo y estudiantes ingresan constantemente.

1.1. Antecedentes

Conforme a lo expuesto por Cárdenas et al. (2023), ante las crecientes demandas urbanas, la gestión de estacionamientos mediante IoT surge como una solución innovadora. El proceso de monitoreo dentro de estos sistemas adquiere una importancia destacada al proporcionar datos en tiempo real sobre la ocupación de espacios, patrones de tráfico y tendencias de uso. Esta recopilación de información permite una gestión eficiente y fluida del estacionamiento, optimizando la asignación de recursos y reduciendo los tiempos de búsqueda para los usuarios. En consecuencia, el enfoque proactivo facilitado por el monitoreo en los sistemas de estacionamiento IoT contribuye a la creación de entornos urbanos más inteligentes y sostenibles.

Ante la necesidad de definir y comprender las aplicaciones en tiempo real, se identifica este tipo de software como sistemas informáticos que interactúan con el entorno físico y responden a estímulos ambientales de manera inmediata, garantizando la coherencia temporal en el flujo de información. Estas aplicaciones no solo se limitan a la esfera tecnológica, sino que también desempeñan un papel crucial en campos como el transporte, la medicina y la

industria. Además de mantener la sincronización temporal de los datos, las aplicaciones en tiempo real posibilitan una interacción más dinámica con los usuarios, ofreciendo experiencias personalizadas y eficientes (Estrella, 2023).

Con el crecimiento de la demanda de estacionamientos en entornos urbanos, la Internet de las cosas (IoT) se concibe como una herramienta esencial para la gestión eficiente de estos espacios. En este contexto, la capacidad de los dispositivos IoT para recopilar y analizar datos en tiempo real ofrece una serie de ventajas significativas. Entre estas, destaca la capacidad de los sensores IoT para monitorizar la ocupación de los espacios de estacionamiento, lo que permite una asignación más efectiva de los recursos disponibles y una mejor gestión del flujo vehicular (Calderon y Escalante, 2023).

Además, la automatización de procesos como el pago de tarifas y la reserva de espacios a través de dispositivos IoT agiliza las operaciones y mejora la experiencia del usuario. Por lo tanto, se puede decir que la IoT está transformando la forma en que se gestionan los estacionamientos, promoviendo una mayor eficiencia y comodidad para los usuarios en entornos urbanos cada vez más congestionados (Rodríguez et al., 2019).

Las técnicas convencionales de monitoreo de estacionamiento utilizan sensores de alto costo para identificar la disponibilidad de espacios de estacionamiento, lo cual dificulta su aplicación a gran escala. Sin embargo, con el advenimiento de la Internet de las Cosas (IoT), ahora es posible utilizar cámaras integradas para monitorear el uso de espacios de estacionamiento a un costo significativamente inferior. Para enfrentar el problema vinculado a la obligación de establecer las ubicaciones de los lugares de estacionamiento antes de que los conductores hagan uso de dichos dispositivos, se ha sugerido por parte de los investigadores un sistema de estacionamiento avanzado asistido por la Internet de las Cosas (IoT-AIPS), respaldado por una plataforma en la nube (Verdejo, 2023).

Los parqueaderos inteligentes son un componente importante de las ciudades inteligentes o *smart cities*. Estos parqueaderos utilizan tecnología avanzada que tienen como fin el mejorar la seguridad y eficiencia del estacionamiento de vehículos. A continuación, se presentan algunas investigaciones relevantes sobre parqueaderos inteligentes en smart cities:

"Smart Parking System: A review" de Velasco y Pinzon (2018) señalan que los parqueaderos inteligentes pueden reducir el tiempo que los conductores tardan en encontrar un lugar de estacionamiento y disminuir la congestión del tráfico. Los autores destacan que la tecnología utilizada en estos parqueaderos incluye sensores, redes de comunicación, aplicaciones móviles y sistemas de información geográfica.

Sai y Suresh (2017) describen un sistema de parquímetro que utiliza tecnología IoT para monitorear y gestionar el uso de los parqueaderos en tiempo real. Los autores destacan que este sistema puede proporcionar información útil a los conductores, como la disponibilidad de espacios de estacionamiento cercanos y las tarifas de estacionamiento.

Se presenta un sistema inteligente que utiliza tecnología de redes de sensores inalámbricos y IoT. Los autores destacan que este sistema puede mejorar la eficiencia del estacionamiento al permitir que los conductores reserven un espacio de estacionamiento con anticipación y también “proporcionar información en tiempo real sobre la disponibilidad de espacios de estacionamiento” (Al-Waisy y Zaidan, 2019, p. 23).

Un estudio exhaustivo sobre los sistemas revisados en este trabajo también fue aplicado en smart cities, llegando a revisar las tecnologías y los enfoques utilizados en estos sistemas, así como los beneficios y desafíos asociados con su implementación. También discuten temas importantes como la seguridad y la privacidad de los datos de los usuarios (Al-Waisy y Zaidan, 2019).

1.2. Planteamiento del problema

Recientemente, la congestión vehicular se ha convertido en un grave problema en las ciudades ya que la gestión del aparcamiento es inadecuada y los usuarios buscan plazas de aparcamiento cerca de sus destinos, provocando accidentes debido al tráfico ocasionado, esto provoca que los usuarios se movilicen más para la búsqueda de un parqueo disponible, aumentando la circulación y perturbando el tráfico, Smart Parking es una solución digital que optimiza el uso eficiente de los espacios de estacionamiento mediante el monitoreo y diagnóstico de la disponibilidad, la demanda y los patrones de uso del estacionamiento.

Las soluciones de estacionamiento inteligentes brindan soporte a las empresas a mejorar sus resultados al tener una mayor cantidad de espacios disponibles para el parqueadero, optimizar el uso del inventario de estacionamiento existente y reducir los espacios vacíos debido a que los clientes se quedan más tiempo.

Estas soluciones también ayudan a las empresas a superar los desafíos y obtener una ventaja competitiva al brindarles a los clientes y empleados una mejor experiencia de estacionamiento y entrega (Cubillos y Rodríguez, 2018).

Para este trabajo de investigación, se ha seleccionado la Universidad Internacional SEK, en el campus Miguel de Cervantes, ya que contenía las carreras más demandadas como ingenierías, carreras de salud y el área administrativa. A este lugar era al que más vehículos ingresaban diariamente y no se tenía un monitoreo de los espacios disponibles. Debido a esto, se planteó esta investigación para solventar los congestionamientos que solían ocasionarse dentro del campus, facilitando la búsqueda de un parqueadero mediante la IoT.

En esta investigación se ha tomado en consideración la cantidad de autos que circulan diariamente y los espacios que existen para aparcarse todo tipo de vehículos que ingresen a la universidad, con ello analizar los datos para lograr una reducción considerable en los tiempos

de espera para la localización de un parqueo disponible, también es importante considerar la contaminación que se produce, debido a que los usuarios al no encontrar un parqueo se movilizan más de lo necesario, al no saber dónde se encuentran los espacios disponibles.

1.3. Justificación

Las soluciones de estacionamientos que combinan IoT e IA pueden ayudar a resolver estos problemas guiando a los conductores a los espacios de estacionamiento disponibles, monitoreando el uso y garantizando la seguridad de los clientes. Uno de los aspectos más importantes de las soluciones de aparcamiento inteligentes es el uso de la IA. La IA se puede utilizar para predecir el número óptimo de plazas de aparcamiento prediciendo la demanda en función de datos históricos, las condiciones meteorológicas actuales e incluso la hora del día.

La IA también puede utilizar datos históricos para determinar los niveles de ocupación, señalando a los clientes que han excedido su tiempo de espera. Las soluciones de estacionamiento inteligente utilizan inteligencia artificial para ayudar a los conductores a encontrar espacios de estacionamiento disponibles, estimar los tiempos de llegada y crear rutas a sus destinos. La IA también se puede utilizar para identificar vehículos que impiden que otros vehículos entren en un estacionamiento.

La propuesta de la Universidad Politécnica de Madrid indica que un sistema de estacionamiento inteligente se define como un mecanismo de control y gestión que supervisa la disponibilidad de lugares para estacionar, ya sea en la vía pública o en estacionamientos privados, de manera remota. La instalación de sensores en áreas públicas permite determinar si un espacio específico está ocupado o disponible. Esta información puede ser revisada por medio de cualquier dispositivo que tenga acceso a la nube, y la solución puede integrarse con sistemas de pago (Herrador, 2023).

1.4. Hipótesis

La implementación de un prototipo de parqueo inteligente utilizando tecnologías de Internet de las cosas (IoT) ayudará a mejorar la eficiencia en la búsqueda de lugares disponibles en estacionamientos, optimizando la gestión del espacio y reduciendo los tiempos de espera para los usuarios.

1.5. Objetivos del Trabajo de Tesis

1.5.1. Objetivo General

Diseñar un prototipo de parqueo inteligente utilizando tecnologías de Internet de las cosas (IoT) y Sensórica para implementar un eficiente sistema de búsqueda de lugares disponibles.

1.5.2. Objetivos Específicos

1. Diseñar la sensorica y control mediante el uso del microcontrolador rasberry pi y Node-red para la comunicación del sistema de control mediante Python y Arduino.
2. Integrar la sensorica y comunicación IoT para el envío de datos al microcontrolador mediante las IP dadas por el esp32 y el protocolo mqtt.
3. Diseñar página web para la visualización de los espacios disponibles mediante la programación en visual estudio.
4. Comunicar el prototipo mediante wifi para la verificación de espacios en el lugar.
5. Evaluar la reducción de tiempos mediante diseño experimental en tiempo real para la validación del funcionamiento del sistema

2. Estado del arte

El Smart Parking con Raspberry Pi proporciona una solución modular y escalable que optimiza la administración del estacionamiento, llegando a beneficiar tanto a los operadores del lugar como a los conductores. La flexibilidad de la Raspberry Pi le permite adaptarse a diversas necesidades e integrarse con las últimas tecnologías en estacionamientos inteligentes. Actualmente existen varias investigaciones sobre lo que es parking inteligente, al igual que varios proyectos que ya se encuentran en funcionamiento, entre los cuales se exponen los siguientes:

Como un primer estudio se destaca el trabajo de investigación de Barake et al. (2023), el cual se basó en el desarrollo de un sistema digital que proporciona información en tiempo real sobre la disponibilidad de estacionamiento en entornos urbanos. El objetivo del estudio fue crear un sistema completo que permitiera a los usuarios verificar la disponibilidad de espacios en un parqueadero recurrente. La metodología incluyó un análisis detallado de las necesidades del estacionamiento y el diseño de un sistema que considera la disposición del edificio, condiciones climáticas extremas y necesidades de usuarios con movilidad reducida. Los resultados indicaron que la implementación exitosa del sistema puede mejorar la experiencia de los usuarios, reducir el estrés asociado a la búsqueda de espacios y transformar la interacción con los estacionamientos.

Seguido de ello, como un segundo proyecto se resalta el estudio de Campoz y Guillin (2022), cuyo tema se centró en la diseño e implementación de un parqueadero automatizado mediante el uso de una red IoT con sensores. El objetivo principal fue desarrollar un sistema que permitiera monitorear la disponibilidad de espacios en tiempo real mediante una red de sensores inalámbricos. La metodología incluyó la instalación de sensores de proximidad HCSR04 en cada plaza de parqueo, la integración con una computadora Raspberry Pi 3 utilizando el protocolo MQTT y la visualización de los datos en una pantalla LED. Los

resultados obtenidos mostraron una mejora significativa en la gestión del estacionamiento, permitiendo a los usuarios verificar la disponibilidad de espacios de manera eficiente y reduciendo los inconvenientes asociados con la búsqueda de estacionamiento en el área de FIEC.

Por otro lado, también se expone el trabajo abordado por Saltos (2018), el cual se centró en el diseño de un prototipo de sistema de parqueo inteligente mediante el uso de IoT. El objetivo del trabajo de titulación fue desarrollar un sistema de estacionamiento inteligente empleando tecnologías basadas en IoT. La metodología implicó la creación de una red inalámbrica con nodos sensor y recolector de datos para monitorear en tiempo real las plazas de estacionamiento. Se utilizaron sensores ultrasónicos para detectar vehículos y transmitir datos al nodo recolector a través de Bluetooth, que luego se enviaron al servidor K2S01(FLUS) mediante Wi-Fi. Los resultados de las pruebas demostraron una precisión de +/- 1 cm en las mediciones de distancia y una repetitividad de datos inferior al 1%, sin pérdida de paquetes al conectarse con la Plataforma IoT.

Así mismo, en un cuarto estudio se resalta el trabajo desarrollado por Vijay (2021), el cual consistió en el diseño e implementación de un prototipo para un sistema de parqueo utilizando una app móvil. El proyecto de investigación se llevó a cabo en varias etapas, en primer lugar, se llevó a cabo una investigación bibliográfica para recopilar información relevante sobre los sistemas y equipos necesarios para diseñar un prototipo de monitoreo de estacionamiento móvil a través de plataformas digitales, buscando soluciones óptimas y de fácil acceso considerando los costos económicos. Posteriormente, se llevaron a cabo pruebas de campo en un parqueadero público para observar el funcionamiento y los resultados del proyecto. Finalmente, se analizaron los resultados en tiempo real proporcionados por la aplicación móvil, lo que permitió establecer la validez del proyecto.

Y, finalmente como quinto y último estudio, se tiene al trabajo desarrollado por Méndez (2021), el cual se basó en el estudio y diseño de un parqueo inteligente haciendo uso de Arduino a través de IoT. Se explicó la base teórica de la tecnología IoT, los sensores y las placas de Arduino, así como la comunicación entre ellas. Como metodología se estableció un medio físico y un protocolo de comunicación utilizando RS-485 con comunicación HalfDuplex para los Arduinos, con un enfoque de maestro y esclavos. Finalmente, como resultados, se proyectó que la implementación de este sistema reduciría la huella de carbono de los vehículos al optimizar el tiempo de estacionamiento de los estudiantes. También se elaboró un presupuesto para los elementos necesarios en el diseño del sistema.

2.1. Referentes Teóricos

2.1.1. Smart Parking basado en Raspberri Pi

De acuerdo con Terán (2020), el Smart Parking basado en Raspberry Pi constituye un sistema innovador que aprovecha la versatilidad y capacidad de procesamiento de la Raspberry Pi para optimizar la gestión y eficiencia de los estacionamientos. Estos sistemas suelen incorporar diversas tecnologías, como sensores, cámaras y conexiones inalámbricas, para ofrecer funcionalidades variadas. A continuación, se presenta una visión general del Smart Parking con Raspberry Pi basado en el estudio de Terán (2020):

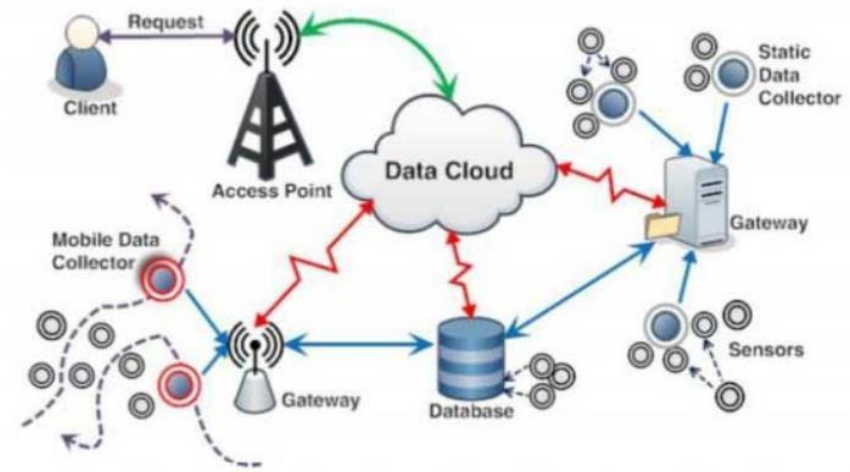
- Sensores de Estacionamiento: Cada espacio de estacionamiento cuenta con sensores, como sensores ultrasónicos o cámaras, para detectar la presencia de vehículos. La Raspberry Pi procesa esta información para determinar la ocupación de cada espacio.
- Sistema de Monitoreo en Tiempo Real: Un monitor conectado a la Raspberry Pi se utiliza para supervisar la disponibilidad de espacios, reconocer matrículas y brindar seguridad adicional.

- Conexión Inalámbrica: Mediante conexiones inalámbricas como Wi-Fi, la Raspberry Pi puede transferir datos de manera instantánea a un servidor central o dispositivo móvil, permitiendo al usuario revisar la disposición de lugares libres en el estacionamiento antes de llegar al lugar.
- Paneles de Información: Los estacionamientos inteligentes suelen incluir paneles informativos que muestran en tiempo real los espacios disponibles, facilitando a los conductores la búsqueda eficiente de estacionamiento.
- Pago en Línea: Integrándose con sistemas de pago electrónico, como lectores de tarjetas o aplicaciones móviles, la Raspberry Pi gestiona eficientemente las transacciones de estacionamiento sin el uso de efectivo.
- Gestión Centralizada: La información recopilada por la Raspberry Pi se envía a un sistema centralizado que permite a los operadores monitorear y gestionar la ocupación, analizar patrones de uso y realizar un mantenimiento proactivo.
- Integración con IoT: Mediante la utilización de Internet de las Cosas (IoT), la Raspberry Pi puede conectarse a otros dispositivos y servicios para mejorar las capacidades de estacionamiento inteligente.
- Personalización: Los desarrolladores pueden aprovechar la Raspberry Pi para crear aplicaciones personalizadas que ofrezcan servicios adicionales, como alertas de disponibilidad de estacionamiento, estadísticas de uso y mensajería instantánea.

2.1.2. La comunicación en la Iot

El artículo de Farnell plantea un modelo arquitectónico para la gestión de estacionamiento inteligente mediante la Iot, este sistema plantea una integración de sensores, pasarela y nube de datos.

Figura 1. *Arquitectura del estacionamiento inteligente*



Fuente: Farnell (2023).

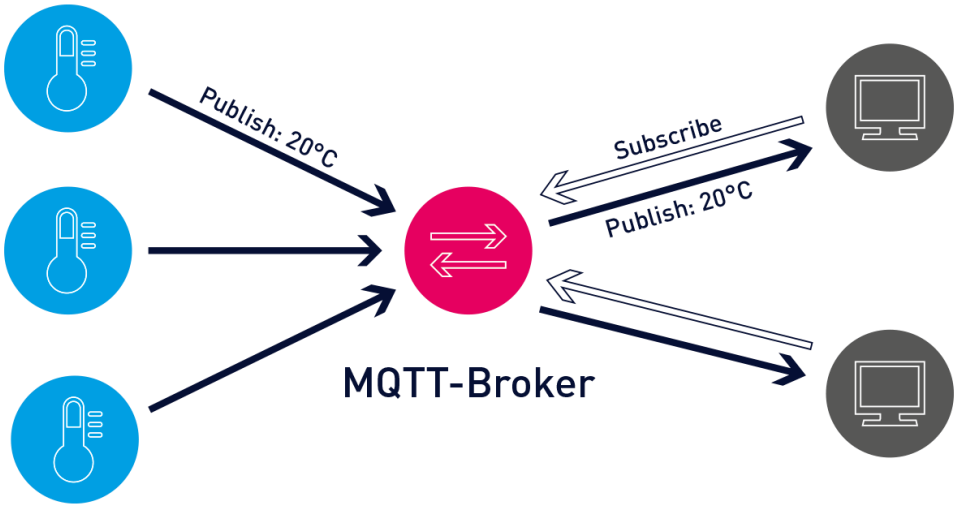
En la figura 5 podemos observar la arquitectura planteada en Farnell “para la comunicación con la base de datos y la transmisión de estos para la comunicación de la sensorica y la IoT” (Farnell, 2023, p. 2).

En la IoT es importante conocer los diferentes dispositivos que ya existen en la actualidad como es el nodo smart parking, el cual es un sensor radar inteligente el cual nos permite identificar las plazas disponibles en zonas urbanas tanto indoor como outdoor. Con este tipo de sistemas se está buscando una respuesta eficaz al congestionamiento, contaminación y una mejor calidad de vida para la población en general, así evitar estrés en largas filas por congestionamiento además de una reducción considerable en la contaminación.

2.1.3. Protocolo MQTT

Es un protocolo de comunicación (M2M), donde dos máquinas se conectan entre si mediante la Iot, diseñado como un transporte de mensajes de publicación ligero. MQTT opera sobre TCP/IP y utiliza una topología de inserción/suscripción. En la arquitectura MQTT, se distinguen dos elementos principales: los clientes y los corredores. Los clientes son sistemas que se conectan al corredor, un servidor que termina siendo un intermediario entre los usuarios. El corredor recibe notificaciones de ciertos clientes y los retransmite a otros, estableciendo así la comunicación entre ellos. Los clientes, ya sea como editores, suscriptores o ambos, no se comunican directamente entre sí, sino que lo hacen a través del corredor (Paessler, 2023).

Figura 2. *MQTT - Broker*



Fuente: Paessler (2023)

MQTT se caracteriza como un protocolo basado en eventos, eliminando la necesidad de transmisiones de datos periódicas o continuas. Este enfoque minimiza la cantidad de transferencia de datos. Los clientes publican mensajes solo cuando es necesario enviar información, y el corredor distribuye estos mensajes a los suscriptores únicamente cuando recibe nuevos datos.

2.1.4. Node-RED

Node-RED, se trata de un entorno de desarrollo visual de código abierto, simplifica la creación de aplicaciones para el Internet de las Cosas (IoT) y flujos de trabajo de desarrollo. Este marco, basado en JavaScript, presenta una interfaz gráfica intuitiva que posibilita a los usuarios conectar nodos predefinidos para construir lógica y automatización de manera sencilla (Torres et al., 2023).

Desarrollado por IBM, Node-RED ha ganado popularidad como una herramienta versátil para diseñar flujos de trabajo y automatizar tareas en diversos entornos, desde hogares inteligentes hasta soluciones empresariales complejas. Su interfaz nodal hace más fácil el integrar dispositivos y servicios, permitiendo a los usuarios desarrollar aplicaciones y automatizaciones sin requerir un profundo conocimiento de programación (González et al., 2023).

En Node-RED, los nodos representan diversas funciones y servicios, y los usuarios pueden conectarlos y configurarlos de manera intuitiva. Esto posibilita la creación rápida de sistemas de trabajo más elaborados sin que sea necesario escribir extensos fragmentos de código. Además, Node-RED es altamente escalable, permitiendo a los usuarios incorporar fácilmente nuevos nodos y funciones para satisfacer sus necesidades específicas (Choosumrong et al., 2023).

Esta herramienta resulta especialmente valiosa en entornos de IoT, donde la interconexión y la automatización de dispositivos son esenciales. Node-RED simplifica el proceso de desarrollo al ofrecer una plataforma potente e intuitiva para la creación de flujos de trabajo, permitiendo a los usuarios convertir eficientemente sus conceptos en aplicaciones del mundo real (Omidi et al., 2023).

2.1.5. Cámaras ANPR Y LPR

2.1.5.1. ANPR

Para este proyecto de investigación se propuso el uso de una cámara ANPR (*Automatic Number Plate Recognition*) o LPR (*Licences Plate Recognition*) la cual según el trabajo de Mesias (2020), se trata de un dispositivo inteligente para la detección de placas vehiculares, además de constar con una IP la cual permite la conexión a una red WIFI para la comunicación que se requiera. Estas cámaras cuentan con la tecnología de reconocimiento óptico, mediante visión artificial

Las cámaras inteligentes ANPR o LPR se denominan estos términos debido al software y hardware que integran para el reconocimiento óptico de las placas vehiculares.

Figura 3. *Cámara ANPR IP*

Fuente: Imasdetres (2023).

Las LPR, o Reconocimiento de Matrículas, es un término ampliamente empleado para referirse a sistemas especializados de cámaras de CCTV (cámaras de reconocimiento de

matrículas) capaces de capturar imágenes de las matrículas en diversas condiciones climáticas, como niebla, lluvia o durante la noche, así como a diferentes velocidades de los vehículos.

Figura 4. *Cámara LPR*

Fuente: Traxpark (2023)

2.1.5.2. LPR

> Significa reconocimiento automático de matrículas y es un término que se utiliza generalmente para describir un programa que utiliza imágenes de una cámara LPR (cámaras lpr) y reconoce caracteres alfanuméricos en las placas de matrícula. Luego puede almacenarlos en una base de datos o relacionarlos con registros en una base de datos. El software puede instalarse en una computadora o integrarse en una cámara, esto se llama cámara ANPR (CamarasVigilancias, 2023, p. 12).

Para esta tesis se utilizó una cámara LPR ya que este dispositivo será colocador al exterior para el sistema de parqueo y cumple con la función de resistencia a agua, polvo y de más factores que influyen en los exteriores de un parqueadero, además estas cámaras logran escanear las placas incluso si el vehículo se encuentra a 160 km/h aproximadamente. Este dispositivo almacena las imágenes de los vehículos que ingresan y sales del parqueadero para así tener un control de la gestión del parqueadero.

Traxpark (2023), menciona las siguientes características de la cámara:

- Versiones de 30 a 360 km/h.
- Detección del tipo de vehículo: moto, auto, camioneta, camión.
- Detección del color del vehículo.
- Soporte para más de 50 países.
- Detecta dos carriles.
- Hasta 8 vehículos a la misma vez.

Se argumenta que un estacionamiento inteligente se define como un sistema para supervisar y administrar áreas de estacionamiento de manera remota, evaluando la disponibilidad de espacios tanto en la vía pública como en estacionamientos privados. La colocación de sensores en áreas públicas permite determinar si un espacio está ocupado o libre para estacionarse. La información resultante puede ser revisas por cualquier dispositivo que posea una conexión web, y esta solución puede ser incorporada a un sistema de pago (Cubillos y Rodríguez, 2018).

El Departamento de Asuntos Económicos y Sociales de las Naciones Unidas pronosticó que todo el crecimiento demográfico global se centrará en zonas urbanas. Se prevé que la tasa de crecimiento alcance el 68% para el año 2050. El incremento demográfico no solo representa un desafío urgente para las autoridades gubernamentales, sino que también se ha transformado en una experiencia diaria para la mayoría de los habitantes. La investigación de USA Today destaca que la búsqueda de lugares de estacionamiento conlleva una notable pérdida tanto de tiempo como de dinero, estimándose en un 35% del tiempo y 345 dólares. Es fundamental abordar estos problemas de estacionamiento, subrayando la necesidad imperante de un sistema de parqueadore inteligente que haga uso del IoT (Mercado et al., 2022).

En Ecuador, se observó un notable aumento en la venta de automóviles entre 2016, con 6,560 unidades vendidas, y 2018, con casi 12,088 unidades comercializadas. Esto señala un

incremento de casi el doble en el número de vehículos en circulación, según datos proporcionados por la AEADE (Asociación de Empresas Automotrices del Ecuador, 2022).

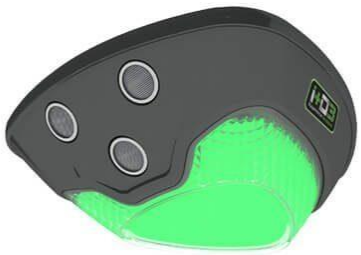
Asimismo, se destaca que, “en 2018 la mayor adquisición de vehículos livianos y comerciales se concentró principalmente en las provincias de Pichincha, con aproximadamente un 36.5%, y Guayas, con un 28.5%” (El Universo, 2023, p. 3). Es evidente que esta dinámica ha llevado a una escasez de espacios de estacionamiento, por lo que tanto los sectores públicos como privado deben tomar medidas para optimizar los estacionamientos de la ciudad. Una de las acciones adoptadas ha sido la implementación de tarifas por el uso de espacios de estacionamiento. Un ejemplo de esto es el caso del Centro Comercial Mall del Sol en la ciudad de Guayaquil, donde se ha observado una falta de disponibilidad de espacios debido a que los vehículos son estacionados por horas, incluso si los propietarios no son clientes del centro comercial (El Universo, 2023).

“Los sistemas tradicionales de monitoreo de estacionamiento utilizan sensores costosos para determinar la disponibilidad de espacios de estacionamiento, lo que dificulta su implementación a escala” (Cuzco y Pinos, 2022, p. 4). Pero la llegada del IoT ha hecho posible rastrear el uso del estacionamiento a un costo menor utilizando cámaras integradas. Para resolver el problema de proporcionar a los conductores la ubicación de las plazas de aparcamiento antes de utilizar dichos dispositivos, los investigadores propusieron un sistema de aparcamiento inteligente de IoT-AIPS con un respaldo en la red (Verdejo, 2023).

La empresa de sensórica Imasdetres presenta un sensor ultrasónico para la detección de los espacios disponibles en una plaza de estacionamiento, este sensor tiene por objetivo el facilitar y mejorar el estacionamiento de los usuarios dentro de un parqueadero inteligente, ya que la tecnología que integra este sensor es la unión de dos dispositivos en uno, debido a que este consta con la tecnología ultrasónica para detección de objetos para este caso de vehículos

y además tiene integrado un indicador LED, con esto al tener integradas las dos necesidades resulta una medida más económica y eficiente (Imasdetres, 2023).

Figura 5. *Sensor ultrasónico I + D3*



Fuente: Imasdetres (2023)

En el artículo web de Dimonoff nos encontramos con los componentes que se emplean en una gestión de parking inteligente, en el artículo nos habla de la integración de sensores objetos conectados, la inclusión de un hardware y software para garantizar el servicio óptico de parqueo inteligente.

En mencionado artículo se nos presenta el sensor MPS de Dimonoff, el cual es un escáner del entorno LiDAR lo que hacen es detectar al mismo tiempo 2 plazas de aparcamiento ya sea si están disponibles u ocupadas (Dimonoff, 2023).

Figura 6. *Sensor MPS*

Fuente: Dimonoff (2023).

2.1.5.3. Sensor fotoeléctrico (NPN)

Acorde a Vargas (2022)En este proyecto se tuvo en cuenta los sensores fotoeléctricos ya que estos nos ayudaran a la detección de vehículos al ingreso y en cada espacio del parqueadero, en este caso se utilizó un sensor fotoeléctrico NPN el cual consta con un alcance de dos metros es un tipo de sensor que utiliza un transistor NPN para su salida. Este tipo de salida es común en la industria y la automatización, y se refiere a la configuración de los elementos de salida del sensor.

A continuación, se presentan algunas características típicas y conceptos clave relacionados con los fotosensores NPN, según Vargas (2022):

- Configuración de salida: Una salida NPN es una configuración de transistor bipolar en la que el transistor NPN se utiliza para controlar una carga (como un relé o un dispositivo de entrada digital). Cuando el sensor detecta un objeto, el circuito se cierra y la corriente fluye desde la carga a la salida del sensor.
- Actividades básicas: Los fotosensores emiten luz y luego detectan la luz reflejada por el objeto. Cuando se detecta luz, se activa la salida del sensor.
- Solicitud: Estos sensores se utilizan en diversas aplicaciones como detección de objetos en líneas de montaje, control de acceso, sistemas de conteo, etc.
- Área de prueba: El rango de detección de los sensores fotoeléctricos NPN puede variar según el modelo y el fabricante. Al buscar un sensor para su aplicación, es importante comprobar que el rango de detección cumpla con sus requisitos.
- Configuración de conexión: Los fotosensores NPN suelen tener tres cables: uno para alimentación (V), uno para salida (NPN) y otro para común o tierra (GND). La correcta conexión de estos cables es fundamental para garantizar un correcto funcionamiento.

- Instalación y configuración: Algunos fotosensores NPN permiten realizar ajustes para adaptarse a condiciones de aplicación específicas, como ajuste de sensibilidad, tiempo de retardo, etc.

2.1.6. Esp32

Según Ochoa et al. (2023), el ESP32, desarrollado por Espressif Systems, es un microcontrolador asequible y de bajo consumo, reconocido por su versatilidad y diversas funciones, especialmente en el ámbito de IoT. Aquí se presentan aspectos clave que es importante conocer sobre el ESP32 de acuerdo con el trabajo de Ochoa et al. (2023):

- Función Principal y Arquitectura: El ESP32 opera con una arquitectura de procesador de doble núcleo Tensilica Xtensa LX6, permitiendo la ejecución paralela de tareas para mejorar el rendimiento.
- Conexión Inalámbrica: Ofrece compatibilidad con Wi-Fi y Bluetooth de bajo consumo, convirtiéndolo en una elección ideal para aplicaciones IoT que requieren conectividad inalámbrica.
- Bajo Consumo de Energía: Gracias a sus modos de bajo consumo, el ESP32 es apto para dispositivos y aplicaciones alimentados por baterías, destacándose por su eficiencia energética.
- Puertos y Periféricos: Dispone de múltiples pines GPIO, interfaz I2C, SPI, UART y puertos analógicos para facilitar la conexión con diversos dispositivos y periféricos.
- Memoria: Incorpora memoria flash interna para almacenar programas y datos, y algunos modelos cuentan con RAM adicional.
- Entorno de Desarrollo: Es programable mediante entornos como Arduino, PlatformIO, o el marco de desarrollo ESP-IDF de Espressif.

- Integración de Sensores y Periféricos: Algunas variantes del ESP32 incluyen sensores integrados como acelerómetros y giroscopios, ampliando sus capacidades para aplicaciones específicas.
- Accesibilidad y Comunidades Activas: Conocido por su accesibilidad, es una elección popular tanto en proyectos de código abierto como en aplicaciones comerciales, respaldado por una comunidad activa de desarrolladores que comparten recursos y soluciones.
- Aplicaciones Comunes:
- Internet de las Cosas (IoT): Ampliamente utilizado para adquisición y transmisión de datos en proyectos de IoT.
- Automatización del Hogar: Aplicado en la construcción de dispositivos como interruptores inteligentes, termostatos y sistemas de monitoreo.
- Electrónica de Bricolaje: Popular en proyectos de hardware de código abierto, como robots, estaciones meteorológicas y dispositivos personalizados.
- Desarrollo de Prototipos: Elegido comúnmente para la rápida creación de prototipos debido a su versatilidad y facilidad de programación.
- Educación: Utilizado en instituciones educativas para enseñar conceptos de programación y electrónica.
- Industria Automatizada: Aplicable en actividades industriales para el registro y control remoto de equipos y procedimientos.

2.1.7. Protocolo WebSocket

WebSockets se conciben como protocolos de comunicación bidireccionales que proporcionan conexiones persistentes y en tiempo real entre clientes y servidores. En el

contexto de Node-RED, WebSockets permite la comunicación en tiempo real entre páginas web y flujos de Node-RED (Changoluisa y Imbaquingo, 2022)

2.1.7.1.Características Clave de WebSockets:

Así mismo, dentro del estudio de Changoluisa y Imbaquingo (2022), a continuación, se mencionan algunas características clave de WebSockets:

1. Comunicación Bidireccional: WebSockets permite la comunicación bidireccional, lo que significa que tanto el cliente como el servidor pueden enviar mensajes de manera independiente en cualquier momento.
2. Persistencia de Conexión: A diferencia de las solicitudes HTTP tradicionales, las conexiones WebSocket permanecen abiertas después de la conexión inicial, lo que permite la transmisión de datos en tiempo real.
3. Bajo Latencia: Al proporcionar una conexión persistente y eliminar la necesidad de abrir y cerrar conexiones para cada solicitud, WebSockets pueden ofrecer una baja latencia en comparación con otros métodos de comunicación.
4. Soporte para Diversos Lenguajes de Programación: WebSockets no está vinculado a un lenguaje de programación específico, lo que significa que puede ser implementado en diversas plataformas utilizando diferentes lenguajes.

1.7. XAMPP

XAMPP es una suite de software libre y de código abierto que facilita la creación y gestión de entornos de desarrollo local para sitios web. Su nombre proviene de las iniciales de los componentes principales que incluye: X (para cualquier sistema operativo), Apache (servidor web), MySQL (sistema de gestión de bases de datos), PHP (lenguaje de programación) y Perl (lenguaje de programación) (Batista et al., 2023).

2.1 Protocolo MQTT (Broker)

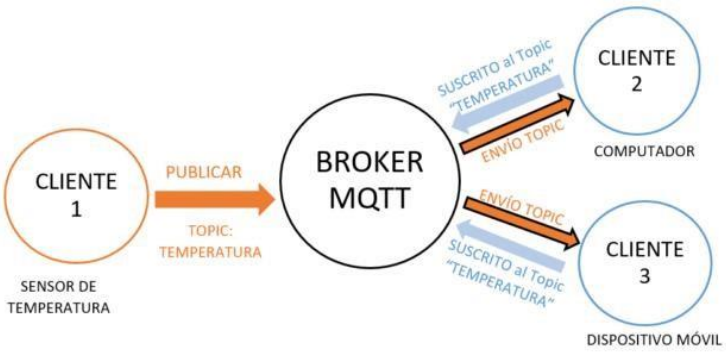
Este es un protocolo diseñado para facilitar el envío de mensajes, tanto para la publicación como la suscripción, siendo particularmente idóneo para su implementación en el IoT. El protocolo opera en un contexto Máquina a Máquina (M2M), permitiendo que dispositivos y máquinas se comuniquen entre sí de manera eficiente. La versatilidad de MQTT se refleja en su capacidad para ser utilizado con una variedad de dispositivos, incluyendo Arduino, Raspberry Pi, entre otros. En el contexto de la recopilación de datos, MQTT desempeña un papel crucial al recibir información proveniente de diversos sensores y transmitirla de manera inalámbrica a través de la red (Mahedero, 2020).

Una de las características más destacadas de MQTT es su simplicidad y ligereza. Este protocolo demanda un bajo ancho de banda, presenta una alta velocidad de respuesta (baja latencia) y requiere un consumo energético reducido en los dispositivos. De acuerdo con Mahedero (2020), en cuanto a su arquitectura, las redes MQTT adoptan una topología en estrella compuesta por elementos fundamentales:

- Broker: Actúa como el nodo central y, por ende, asume la responsabilidad de receptar y repartir datos hacia los usuarios.
- Cliente: Se refiere a cualquier dispositivo que se conecta al broker para enviar y recibir información.
- Tema: Define la manera en que los mensajes son conocidos o identificados. Los clientes pueden publicar temas, suscribirse a ellos o realizar ambas acciones.
- Publicación: Implica enviar información a los brokers para su distribución a los clientes interesados.
- Suscribirse: Con la subscripción de un cliente a un tema, todos los mensajes publicados en el broker son entregados a ese cliente.

Este enfoque en topología estrella ofrece claridad al momento de comprender como están conformadas las redes MQTT. Un ejemplo adicional se proporciona para ilustrar de manera más detallada esta topología y su funcionamiento.

Figura 7. *Protocolo MQTT*



Fuente: Mahedero (2020).

Este dispone de una configuración que involucra tres clientes: un sensor, una computadora y un dispositivo móvil. El CLIENTE 1, que funge como sensor, opera en calidad de editor. Su función principal es realizar mediciones de la temperatura en la habitación y transmitir los datos obtenidos al broker mediante el uso de un tema específico, en este caso, "temperatura". El broker, por su parte, tiene la responsabilidad de dirigir los datos recibidos del CLIENTE 1 hacia todos aquellos usuarios que poseen una subscripción al tema "TEMPERATURA" (Mahedero, 2020).

En cuanto al protocolo MQTT, incorpora la noción de Calidad de Servicio (QoS), lo que permite especificar la forma en que los mensajes son entregados a diferentes clientes. Acorde a lo expuesto por Mahedero (2020), se distinguen tres niveles de QoS:

- QoS 0: Se entrega una vez como máximo. Es decir, el mensaje puede no ser entregado, y si se recupera, los datos pueden ser utilizados de manera intermitente, sin que la pérdida de algunos datos sea crítica.

- QoS 1: La entrega del mensaje es garantiza en al menos una ocasión, aunque pueden producirse duplicaciones.
- QoS 2: Se garantiza que los mensajes llegarán solo una vez.

Entre las ventajas destacadas de MQTT se encuentran su estabilidad, asincronía y aislamiento entre clientes. Su diseño simple y ligero lo hace especialmente adecuado para aplicaciones de IoT, donde los dispositivos llegan a tener recursos limitados. La eficiencia en el uso de recursos conlleva a un menor consumo de energía, mientras que los requisitos mínimos de ancho de banda resultan esenciales en entornos con redes inalámbricas propensas a problemas de calidad. Además, MQTT proporciona seguridad, calidad de servicio, robustez y confiabilidad en el sistema desarrollado (Mahedero, 2020).

3. Metodología

Dentro de este capítulo se exponen cada uno de los aspectos metodológicos empleados dentro de la presente investigación, los cuales fueron de suma importancia para el desarrollo del proyecto. A continuación, se exponen cada uno de los aspectos:

3.1. Diseño de la Investigación

En el presente estudio, se optó por emplear un diseño de investigación experimental con un enfoque mixto, donde se combinan aspectos cuantitativos y cualitativos. Esta decisión se basó en la necesidad de analizar datos numéricos y evaluar variables específicas relacionadas con el uso de parqueaderos bajo disponibilidad mediante tecnologías IoT y sensórica. Al elegir un diseño experimental, se buscó controlar y manipular activamente las variables del entorno de estudio para comprender mejor el impacto del sistema prototipo en la eficiencia del estacionamiento, la disponibilidad de espacios y la satisfacción del usuario. Esto, además, permitió obtener resultados descriptivos, cuantificables y precisos que brindaron una comprensión detallada del funcionamiento y la efectividad del sistema.

3.2. Variables de Estudio

Conforme al diseño y enfoque del presente trabajo de investigación, las variables de estudio que intervinieron fueron las siguientes:

Variables Independientes:

- Tecnologías IoT utilizadas en el sistema prototipo (sensores, dispositivos de comunicación, plataformas de software, etc.).

- Configuración del sistema (ubicación y número de sensores, tipo de comunicación, protocolos utilizados, etc.).
- Factores ambientales (ubicación geográfica, clima, condiciones físicas del entorno, etc.).

Variables Dependientes:

- Disponibilidad de espacios de estacionamiento.
- Eficiencia del sistema de monitoreo y comunicación.
- Nivel de satisfacción del usuario con el sistema.
- Tiempo promedio de búsqueda de estacionamiento.
- Número de reservas de espacios de estacionamiento realizadas a través del sistema.

Variables de Control:

- Horarios de mayor demanda de estacionamiento.
- Tamaño y capacidad del parqueadero.
- Políticas y regulaciones de estacionamiento existentes.

Gracias a estas variables, fue posible analizar y evaluar diferentes aspectos del sistema prototipo de uso de parqueaderos bajo disponibilidad, facilitando la comprensión de su funcionamiento, impacto y efectividad en la gestión del estacionamiento.

3.3. Hipótesis de Estudio

La implementación del sistema prototipo de parqueaderos mediante IoT y sensórica ayudará a mejorar la eficiencia y la satisfacción del usuario en comparación con métodos tradicionales de gestión de estacionamiento.

3.4. Población y Muestra

La población de este estudio estuvo compuesta por un grupo de docentes que utilizan el estacionamiento en la institución educativa. Se buscaba indagar acerca de sus perspectivas y

puntos de vista con respecto al prototipo de parqueadero inteligente mediante IoT. Por otro lado, se optó por un tipo de muestreo no probabilístico por conveniencia, lo que significa que el investigador tuvo la libertad de seleccionar la cantidad de sujetos de estudio con los cuales trabajar, sin necesidad de aplicar una fórmula específica para el cálculo de la muestra.

3.5. Técnicas e Instrumentos de Recolección de Datos

Como técnica de investigación, se empleó una entrevista dirigida a los docentes que hacían uso del estacionamiento, con el objetivo de comprender su perspectiva y recabar su feedback sobre el prototipo diseñado. En cuanto al instrumento utilizado, se aplicó un cuestionario de preguntas abiertas para permitir una mayor profundidad en las respuestas y una comprensión más completa de las opiniones y experiencias de los participantes.

3.6. Análisis de los Datos

Después de entrevistar al grupo de docentes sobre su experiencia con el estacionamiento en el campus, se observó un consenso generalizado sobre los desafíos y las necesidades insatisfechas en relación con la disponibilidad de espacios de estacionamiento. Los docentes expresaron una frustración común con la dificultad para encontrar estacionamiento durante las horas pico, lo que a menudo resulta en retrasos en su llegada a clases, reuniones y otros compromisos universitarios. Esta situación genera estrés adicional y puede afectar la puntualidad y la productividad en el entorno académico.

Uno de los aspectos más destacados de las respuestas fue la importancia que los docentes atribuyen a la disponibilidad de información en tiempo real sobre los espacios de estacionamiento. Muchos expresaron su descontento con la falta de sistemas que proporcionen actualizaciones en tiempo real sobre la disponibilidad de espacios, lo que conduce a una búsqueda prolongada y a menudo infructuosa de lugares de estacionamiento adecuados. Esta falta de transparencia y visibilidad sobre la disponibilidad de espacios de estacionamiento contribuye significativamente a su experiencia negativa en el campus.

En cuanto a las características que consideran importantes en un sistema de parqueaderos, los docentes enfatizaron la necesidad de que el sistema sea fácil de usar, conveniente en términos de ubicación y acceso, y capaz de proporcionar información precisa y actualizada sobre la disponibilidad de espacios. La conveniencia y la accesibilidad emergieron como aspectos críticos, ya que los docentes buscan soluciones que simplifiquen su experiencia de estacionamiento y minimicen el tiempo dedicado a buscar un lugar adecuado.

Al discutir el potencial de la tecnología IoT y los sensores para mejorar la gestión de los parqueaderos universitarios, los docentes mostraron interés en las soluciones innovadoras que podrían optimizar la asignación de espacios y proporcionar una experiencia más fluida para los usuarios. Sin embargo, también expresaron preocupaciones legítimas sobre la privacidad y la seguridad de los datos recopilados por estos sistemas, subrayando la importancia de implementar medidas robustas de protección de datos y garantizar la transparencia en el manejo de la información del usuario.

Finalmente, en términos de recomendaciones y sugerencias, los docentes ofrecieron una serie de ideas para mejorar la eficiencia y la efectividad del sistema de parqueaderos. Estas incluyen integrar aplicaciones móviles para facilitar la reserva de espacios, mejorar la señalización en el campus para guiar a los conductores hacia espacios disponibles y proporcionar una comunicación clara y transparente sobre el funcionamiento del sistema. Además, destacaron la importancia de involucrar a la comunidad universitaria en el diseño y la implementación del sistema, asegurando que las soluciones propuestas aborden adecuadamente las necesidades y preocupaciones de los usuarios finales.

3.7. Metodología de Desarrollo

3.7.1. Metodología Ágil Xtreme Programming

Para el proyecto de diseño de un sistema prototipo de parqueaderos bajo disponibilidad mediante IoT y sensórica, se utilizó la metodología XP (Extreme Programming), la cual se divide en las siguientes fases:

1) Diseño Inicial:

- Diseño de la arquitectura del sistema, incluyendo la disposición de los sensores, la infraestructura de comunicación IoT y la interfaz de usuario.
- Identificación de los componentes principales del sistema y sus interacciones.
- Creación de prototipos de bajo nivel para validar conceptos y tecnologías. **2) Implementación Iterativa**:
- Desarrollo iterativo de funcionalidades basadas en las historias de usuario identificadas.
- Aplicación de prácticas de programación como TDD (Desarrollo Dirigido por Pruebas) para garantizar la calidad del código desde el principio.
- Integración continua de nuevas funcionalidades para garantizar la estabilidad del sistema en todo momento.

3) Pruebas y Validación:

- Realización de pruebas unitarias y de integración para cada componente del sistema.
- Pruebas de extremo a extremo para validar el funcionamiento general del sistema.
- Recopilación de retroalimentación y corrección de errores identificados durante las pruebas.

4) **Refinamiento y Mejora Continua**:

- Evaluación de la retroalimentación de los usuarios y los stakeholders para identificar áreas de mejora.
- Refactorización del código y optimización de la arquitectura para mejorar la escalabilidad y la mantenibilidad.
- Iteración en el diseño y la implementación basada en la retroalimentación recibida.

5) **Despliegue del Prototipo**:

- Preparación del entorno de producción para el despliegue del prototipo.
- Implementación de medidas de seguridad y control de acceso según sea necesario.
- Despliegue del sistema prototipo en un entorno controlado para su evaluación final

3.8. Desarrollo del Prototipo

3.8.1. Funcionamiento General del Prototipo

En el Diagrama 1 se presenta un análisis del funcionamiento general del prototipo, el cual incluye el mecanismo que regirá el uso del parqueadero y el envío de datos según la programación establecida en el sistema.

Diagrama 1 *Diagrama de funcionamiento general*

Fuente: Elaboración propia

3.8.2. Comunicación rasberry pi y esp 32

"La comunicación entre el ESP32 y la Raspberry Pi puede establecerse de diversas maneras, siendo una de las más frecuentes la comunicación inalámbrica a través del protocolo MQTT (Message Queue Telemetry Transport)" (Calot et al., 2017, p. 85). Este protocolo se destaca especialmente en entornos IoT, facilitando la transferencia eficiente de datos entre dispositivos conectados. Al implementar el protocolo MQTT, se alcanza una fluidez de conexión entre el ESP32 y la Raspberry Pi, proporcionando un canal eficiente para el intercambio de datos en entornos IoT.

El ESP32, un microcontrolador versátil con capacidades inalámbricas puede desempeñar el rol de cliente MQTT. Se utiliza una biblioteca MQTT específica en el entorno de desarrollo del ESP32 para establecer conexión con un servidor MQTT, que puede residir tanto en la Raspberry Pi como en otro lugar de la red. La Raspberry Pi, por su parte, puede actuar como servidor MQTT para recibir los datos transmitidos por el ESP32. Configurando un bróker MQTT en la Raspberry Pi, se gestiona eficientemente la comunicación entre estos dispositivos. La transmisión de datos sigue una estructura de publicación-suscripción, donde el ESP32, al convertirse en un cliente MQTT, publica datos en "temas" o canales específicos. La Raspberry Pi, configurada como cliente o suscriptor MQTT, se conecta a estos temas específicos para recibir y procesar los datos enviados por el ESP32.

Este enfoque permite una comunicación bidireccional, por ejemplo, el ESP32 puede enviar datos de sensores a un tema específico, y la Raspberry Pi, suscrita a dicho tema, recibirá y procesará estos datos para análisis, visualización u otras manipulaciones deseadas. La versatilidad de MQTT y la capacidad de la Raspberry Pi para ejecutar diversas aplicaciones y servicios hacen que esta configuración sea idónea para proyectos de IoT. Además, la conexión inalámbrica elimina la necesidad de cables, facilitando su implementación en entornos y proyectos del mundo real que requieran movilidad.

En conclusión, la comunicación eficiente entre el ESP32 y la Raspberry Pi mediante MQTT proporciona una solución potente y efectiva para proyectos de IoT, garantizando una transferencia de datos confiable y bidireccional entre ambos dispositivos.

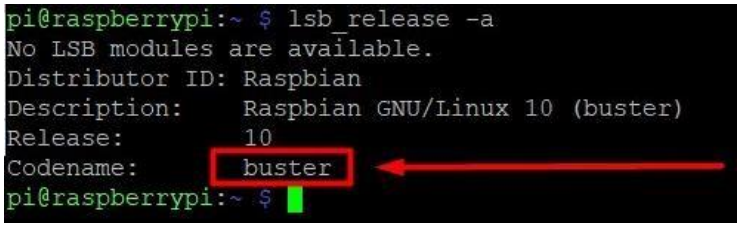
3.8.3. Instalación de broker en la tarjeta Raspberry PI 3

Para la instalación del broker MQTT se distinguen los siguientes pasos:

1. Comprobación de la versión de Raspbian: En el terminal de comandos se escribe el comando **lsb_release –a**, donde se despliega la información sobre la versión que

tiene actualmente el dispositivo, como se muestra en la siguiente figura, esta versión debe ser búster ya que es compatible para el broker a instalar.

Figura 8 *Comprobación de la versión de Raspbian*

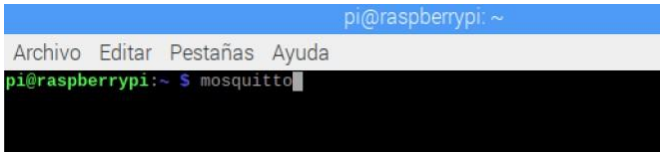


Fuente: Elaboración propia

2. Descargar del paquete mosquitto en el repositorio de Raspbian mediante el comando:
    - wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key - Para almacenar el archivo en un directorio actual se usa el comando:
    - cd /etc/apt/sources.list.d/.
3. Instalar el paquete de datos según la versión de Raspbian en la dirección escogida anteriormente con el comando:
    - sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list
4. Actualizar el sistema con el comando: **sudo apt-get update**
5. Cargar MQTT en la tarjeta Raspberry, el cual puede reconocer a la Raspberry como cliente o como servidor, el broker mosquitto hace las veces de servidor a la vez que necesita reconocer los clientes con los que interactúa. El comando **sudo apt install mosquitto mosquitto-clients** sirve para instalar los paquetes de clientes.

6. Para arrancar el broker mosquitto ingresar al terminal de la Raspberry Pi, e ingresar el comando **mosquitto**

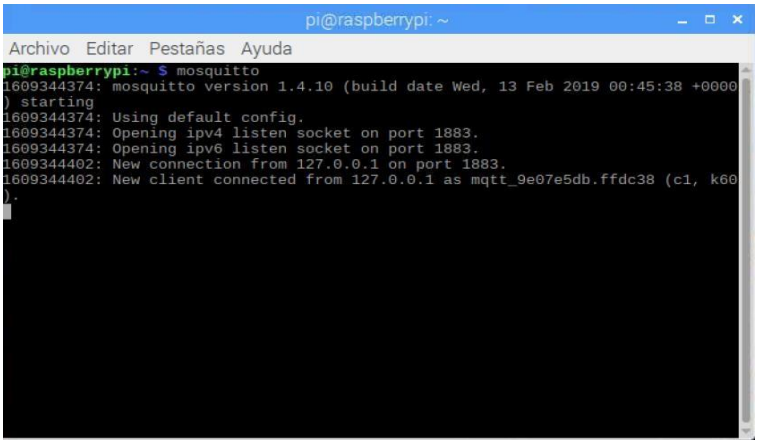
Figura 9 *Arranque*



Fuente: Elaboración propia

7. Esperar a que el bróker se ejecute

Figura 10 *Espera de la ejecución*



Fuente: Elaboración propia

Node-RED es una herramienta de programación para conectar dispositivos de hardware, API y servicios en línea de formas nuevas e interesantes.

Proporciona un editor basado en navegador que facilita la conexión de flujos entre sí utilizando la amplia gama de nodos de la paleta que se pueden implementar en su tiempo de ejecución con un solo clic. (Más en información en: nodered.org)

3.8.4. Instalación de XAMPP

Se descarga e instala XAMPP desde el sitio oficial (https://www.apachefriends.org/). Durante la instalación, elige los componentes que necesitas, como Apache y PHP.

- **Configuración de Apache:**
  - Después de la instalación, abre el panel de control de XAMPP.
  - Inicia los servicios de Apache haciendo clic en el botón correspondiente.
- **Verificación del Funcionamiento Local:**
  - Abre tu navegador web y visita http://localhost o http://127.0.0.1. Deberías ver la página de inicio de XAMPP, lo que indica que Apache está funcionando correctamente en tu máquina local.
- **Configuración del Archivo de Hosts:**
  - Localiza el archivo de hosts en tu sistema operativo (por ejemplo, C:\Windows\System32\drivers\etc\hosts en Windows o /etc/hosts en Linux/macOS).
  - Edita el archivo añadiendo una entrada como la siguiente:

plaintextCopy code 192.168.1.2nombre-elegido.local

  - Sustituye "192.168.1.2" con la dirección IP de tu máquina y "nombreelegido.local" con el nombre que desees darle a tu sitio.
- **Configuración del Archivo de Configuración de Apache:**
  - Localiza el archivo de configuración de Apache (httpd.conf o apache2.conf dependiendo del sistema) en la carpeta de instalación de XAMPP. o Busca la

sección que empieza con <Directory "C:/xampp/htdocs"> (o una ruta similar) y cambia AllowOverride None a AllowOverride All.

- **Reinicio de Apache:**
  - Reinicia los servicios de Apache desde el panel de control de XAMPP para aplicar los cambios.
- **Prueba desde Otros Dispositivos:**
  - Asegúrate de que ambos dispositivos (la máquina que ejecuta XAMPP y el dispositivo que desea acceder al sitio) estén en la misma red.
  - Abre un navegador en el otro dispositivo e ingresa la dirección IP de la máquina que ejecuta XAMPP o el nombre que le asignaste seguido de ". local" (por ejemplo, http://192.168.1.2/nombre-elegido.local).´ ○ Ahora, la página web alojada en XAMPP debería ser accesible desde otros dispositivos en la misma red local. Ten en cuenta que, para hacer esto en un entorno de producción en Internet, necesitarías configurar la red y el servidor de una manera más segura y considerar temas de seguridad.

- **Pasos para creación del Bot**

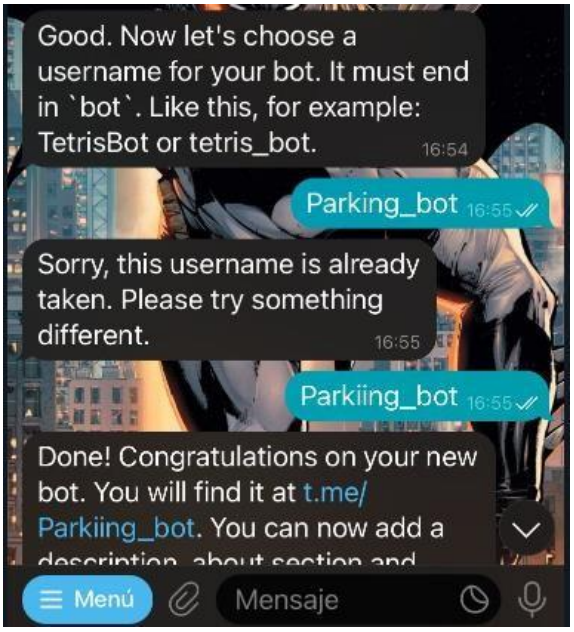
1) Abrir Telegram: Abrir la aplicación Telegram en un dispositivo e Iniciar una conversación con el BotFather:
2) Crear un Nuevo Bot: Enviar el comando /newbot al BotFather para crear un nuevo bot y seguir las instrucciones proporcionadas.

Figura 11 *Creación de nuevo bot*

Fuente: Elaboración propia

3) Asignar un Nombre y un Usuario: Elegir un nombre para el bot y asignar un nombre de usuario que termine en "bot" (verificar que el nombre esté disponible, para este ejemplo, "Parkiing_bot").

Figura 12 *Asignación de un nombre y usuario*

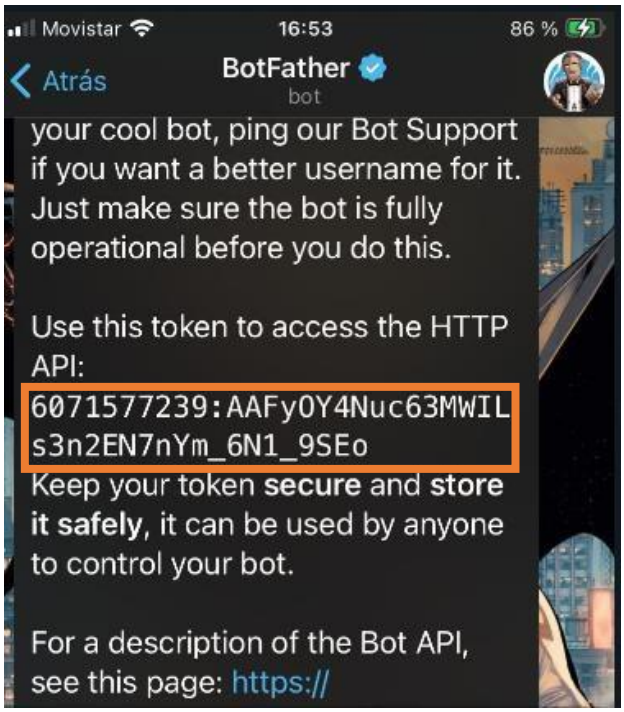


Fuente: Elaboración propia

4) Obtener el Token del Bot: Anotar el token proporcionado por el BotFather, ya que será necesario para autenticar el bot y comunicarse con la API de

Telegram.

Figura 13 *Obtención del token del bot*



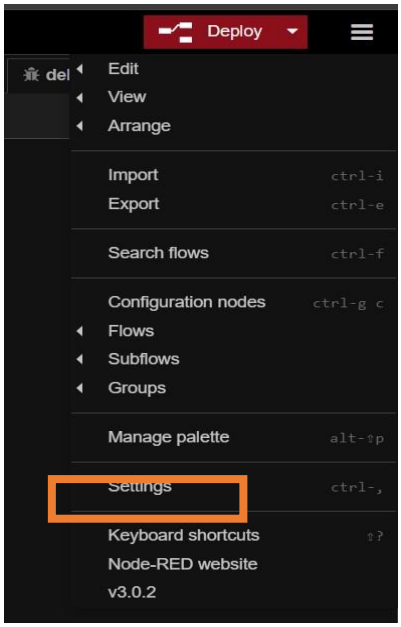
Fuente: Elaboración propia

8. COMUNICACIÓN BOT -NODE RED

9. Para poder enviar datos desde Node Red a Telegram se debe seguir los siguientes pasos:

- Instalar los Nodos de Telegram desde la opción Manage Palette

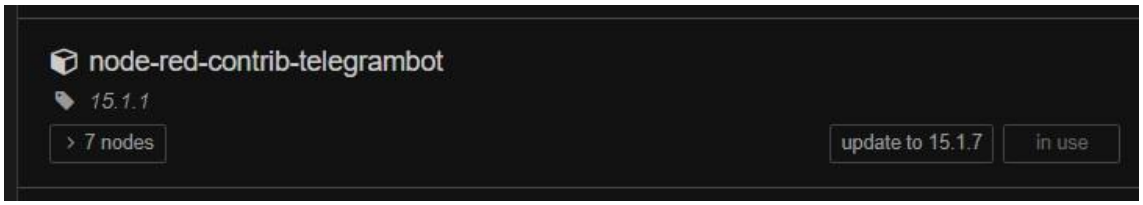
Figura 14 *Opción de configuraciones*



Fuente: Elaboración propia

E instalar los nodos

Figura 15 *Instalación de nodos*



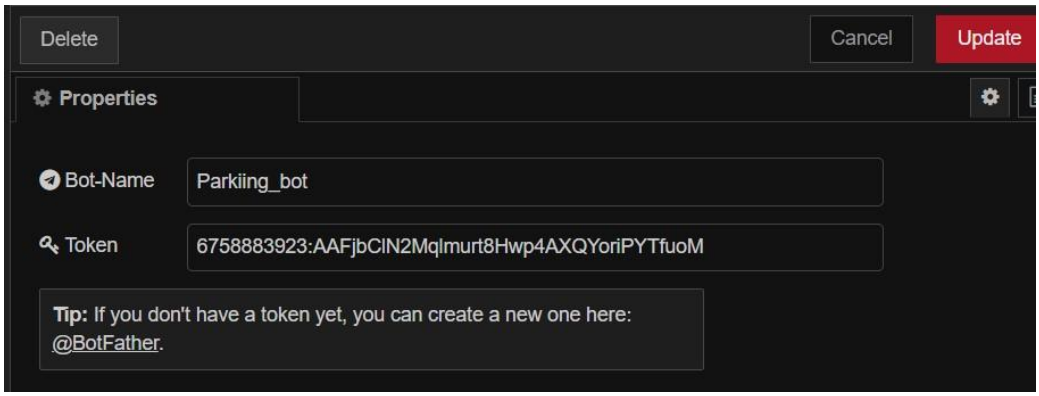
Fuente: Elaboración propia

- Una vez instalado, lo primero a realizar es, arrastrar el nodo al espacio de trabajo y configurarlo

Figura 16 *Arrastre del nodo para configuración*

Fuente: Elaboración propia

Figura 17 *Especificación del tipo de mensaje a enviar*



Fuente: Elaboración propia

Una vez haya sido configurado, ya se pueden enviar mensajes anteponiendo un nodo de función donde se especifique el tipo de mensaje a enviar, junto con el contenido

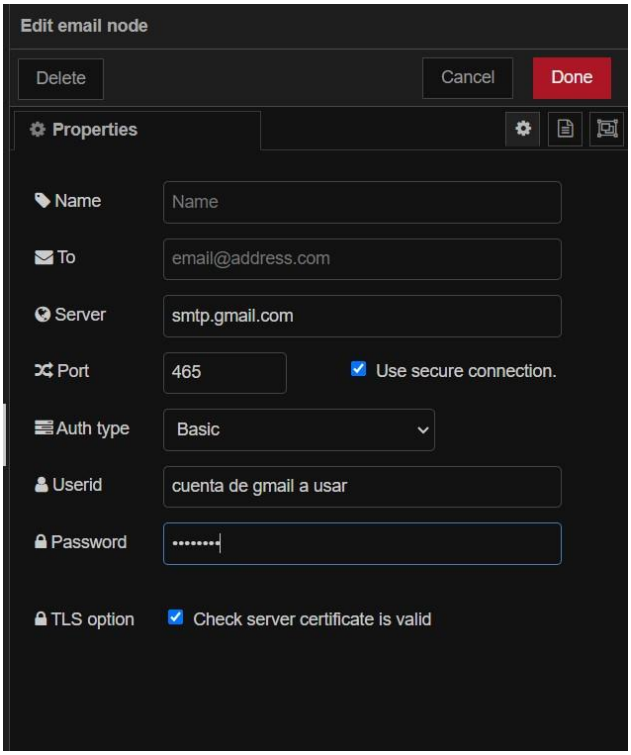
NODO – MAIL

Al igual que con el nodo de telegram, se debe instalar el nodo de e-mail desde el manage palette, la opción a instalar se muestra en la siguiente figura:

Figura 18 *Instalación del nodo e-mail*

Fuente: Elaboración propia
Y dándole la configuración:

Figura 19 *Configuración para el envío de mensajes*



Fuente: Elaboración propia

Una vez haya sido configurado, ya se pueden enviar mensajes anteponiendo un nodo de función donde se especifique el tipo de mensaje a enviar, junto con el contenido.

3.8.5. Página WEB

Para este punto del proyecto se usa un tipo de programación hecho por nodos en donde el programa fue dividido tanto en las funciones que son necesarias para la actualización de la página, para el registro de datos, la delimitación de valores desde la interfaz y el envío de mensaje por mail o mensaje al usuario.

Así pues, a continuación, se detalla el funcionamiento de este programa con sus debidas funciones.

3.8.5.1. Recepción de datos por formulario

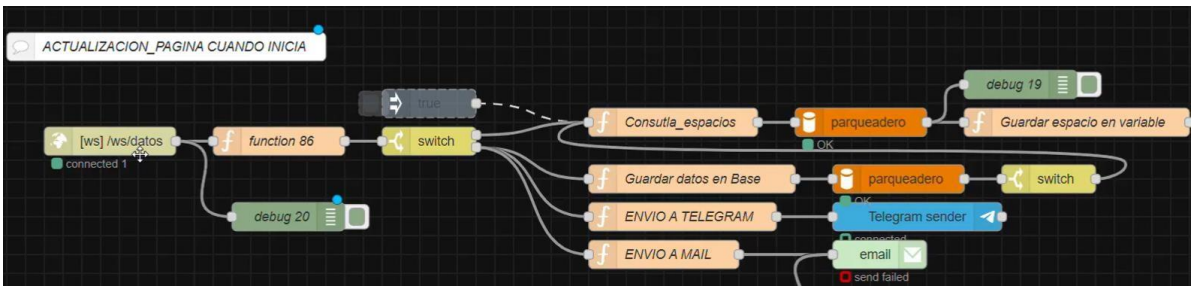
Como base de esta parte del programa existe el diagrama de flujo que se encuentra en la Figura 12 del presente trabajo. Principalmente, por medio de esta conexión de nodos es posible registrar la información de; nombre, correo e ID de telegram, siendo este último el medio por el que se ha considerado relevante enviar la notificación de espacios disponibles y registro de ingreso (En el caso de que se desee, esto puede cambiarse).

Figura 20 *Formulario de estacionamiento*

Fuente: Elaboración propia

En la siguiente figura se observa esta primera parte del programa:

Figura 21 *Parte inicial del programa*

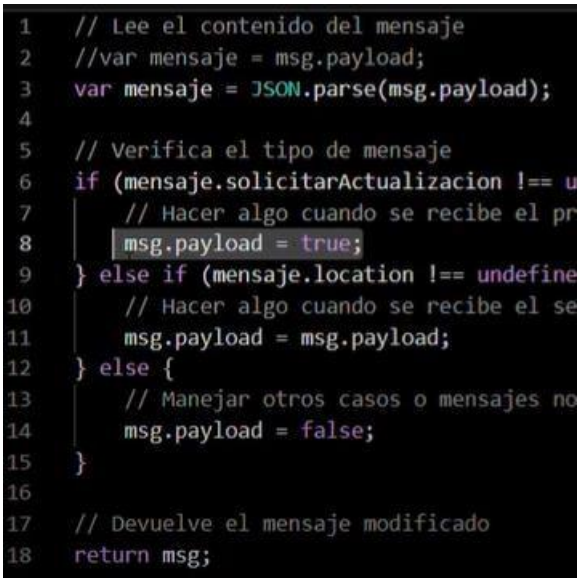


Fuente: Elaboración propia

Entre las acciones que tienen lugar en esta programación, tenemos:

- **Función 86:** Si este es verdadero se envía la condición para ingresar la información, caso contrario este no funciona y se tiene que volver a llenar el formulario.

Figura 22 *Gestión de contenidos de mensajes*



Fuente: Elaboración propia

- **Switch:** Condición para continuar el programa bien sea para la consulta de espacios disponible, o llevar la información a la base de datos, realizar el envío por telegram y email.
- En el caso de que la condición sea verdadera, el programa llena la base de datos “parqueadero” con sus respectivas variables de disponibilidad y espacio.

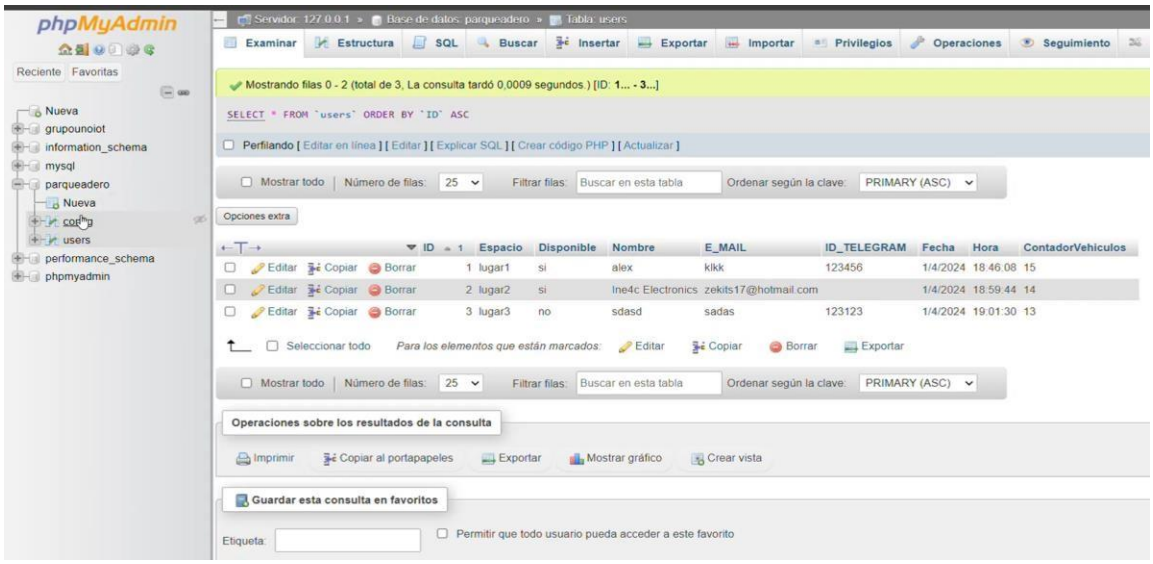
Figura 23 *Consultas en la base de datos*

```
// Respuesta de la base de datos (puede variar dinámicamente)
var respuestaBaseDatos = msg.payload;  // Asumiendo que la respuesta de tu nodo anterior está en msg.payload

// Verificar si hay lugares disponibles
if (respuestaBaseDatos && respuestaBaseDatos.length > 0) {
    // Formatear la respuesta para que coincida con la estructura esperada
    var lugaresDisponibles = respuestaBaseDatos.map(function (item) {
        return {
            lugar: item.Espacio,
            disponible: item.Disponible
        };
    });

    // Enviar la respuesta formateada a la página web
    msg.payload = JSON.stringify(lugaresDisponibles);
} else {
    // Si no hay lugares disponibles, podrías enviar un mensaje indicando eso
    msg.payload = "No hay lugares disponibles en este momento.";
}

return msg;
```

Fuente: Elaboración propia

- **Parqueadero:** Siendo esta la base de datos que se guardan en la variable "Guardar espacios", llegando a actualizar la información disponible y presentándola por medio de una página web que permitirá la gestión de este por un controlador:

Figura 24 *Gestor de Base de Datos Php*

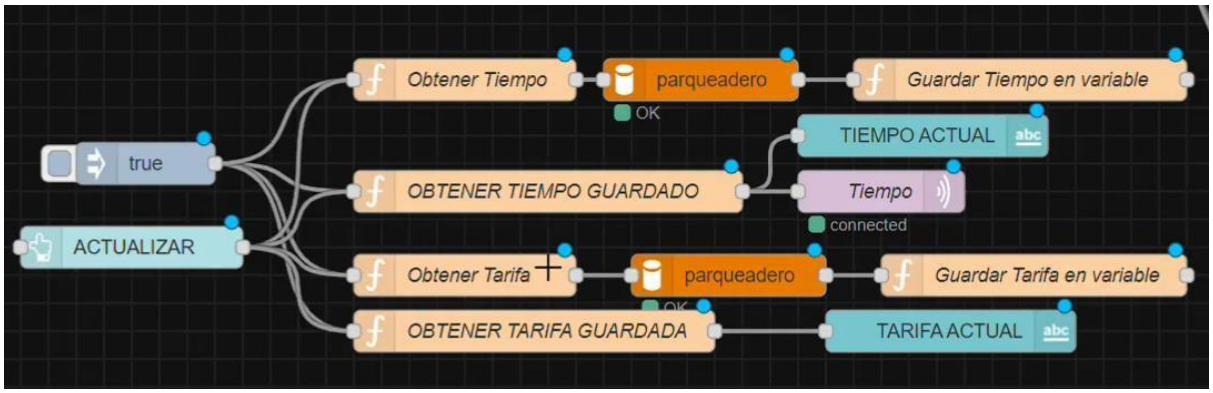


Fuente: Elaboración propia

3.8.5.2. Actualizar la información

Esta parte del programa también sirve a manera de respaldo que se actualiza cada que existe un nuevo ingreso de datos, ya sea por errores técnicos, por cortes de luz o cualquier eventualidad técnica. A continuación, se observan los bloques con los que cuenta esta parte de la programación:

Figura 25 *Actualización de información*



Fuente: Elaboración propia

Además de lo mencionado, se cuenta con las variables de tiempo y tarifa, datos que registrados en un arreglo en base a los datos mencionados con anterioridad. Por medio de estos nodos es posible completar debidamente la información de los usuarios que desean desocupar ya un espacio en el parquímetro. A continuación, se observa las líneas de código de estas dos funciones:

Figura 26 *Función de tiempo*

```
// ID del registro del cual deseas obtener el tiempo
var idRegistro = 1;  // Reemplaza con el ID del registro que te interesa

// Crear y ejecutar la consulta SQL para seleccionar el tiempo desde la tabla
var consultaSQL = `SELECT Tiempo FROM config WHERE ID = ${idRegistro}`;
msg.topic = consultaSQL;

return msg;

```

Fuente: Elaboración propia

Figura 27 *Función de tarifa*

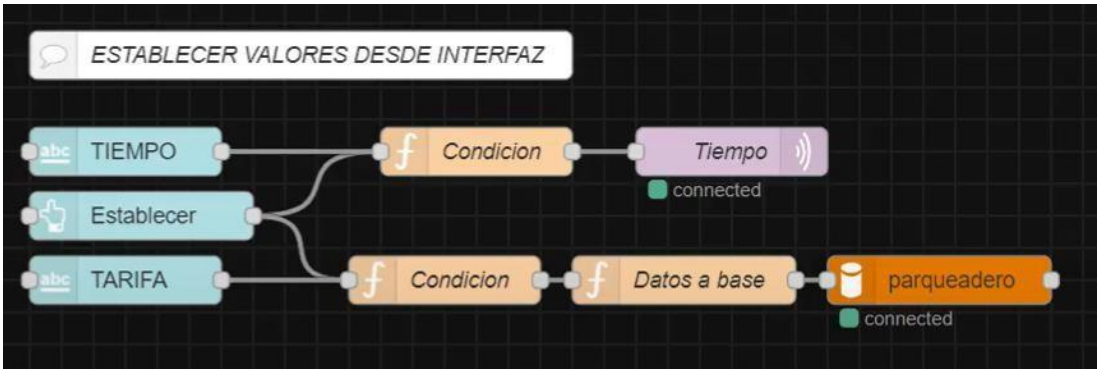
```
// ID del registro del cual deseas obtener el tiempo
var idRegistro = 1;  // Reemplaza con el ID del registro que te interesa

// Crear y ejecutar la consulta SQL para seleccionar el tiempo desde la tabla
var consultaSQL = `SELECT Tarifa FROM config WHERE ID = ${idRegistro}`;
msg.topic = consultaSQL;

return msg;

```

Fuente: Elaboración propia

3.8.5.3. Establecer valores desde la interfaz

Esta parte del programa está formada tanto por sus debidos bloques y por una interfaz gráfica que se observan a continuación:

Figura 28 *Bloques e interfaz del programa*



Fuente: Elaboración propia

Figura 29

Menú del proyecto parqueadero

Fuente: Elaboración propia

Para esta parte tomar en consideración lo siguiente:

- La interfaz se ha divido en secciones, estando al lado izquierdo la sección de indicadores que muestras que puestos están ocupados y cuales están libres. Al lado derecho se encuentra la sección de configuración.
- El botón de “establecer” ha sido colocado como condicionante para la actualización de la base de datos y no provocar recursividad innecesaria mientras se está escribiendo los datos de tiempo y tarifa.
- La información se guarda en la base de datos “parqueadero”, página menciona con anterioridad y donde se actualiza tanto lo del formulario, como lo de esta sección que es la interfaz.

3.8.5.4. Lectura de datos – ESP32

Esta sección consiste únicamente en los datos que son enviados desde el Arduino y son procesados para el funcionamiento de la interfaz. Como se están usando led´s, estos solo

detectan señales de verdadero y falso, razón por la cual estos están conectados como se ve en la siguiente figura de nodos:

Figura 30 *Lectura de datos – ESP32*

Fuente: Elaboración propia

Con el led prendido se observa que se ha registrado un ingreso y mientras este se quede encendido en la interfaz significará que el lugar está siendo ocupado.

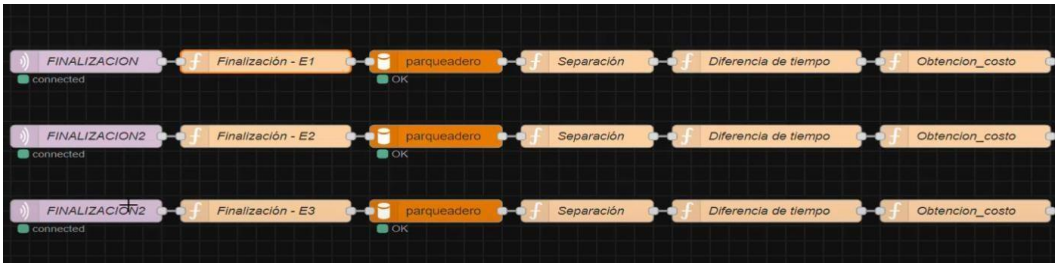
Figura 31 *Indicadores de ocupación de espacios*

Fuente: Elaboración propia

3.8.5.5. Finalización

Desde el ESP-32 llega el mensaje de registro a esta parte del programa conocida como "Finalización", en donde se coteja todas las variables ingresadas y analizadas para poder fijar la tarifa y mantener actualizada la base de datos.

Figura 32 *Finalización del programa*



Fuente: Elaboración propia

Variables como; tiempo, ingreso, nombre, mail y telegramID han sido definidas como globales para poder llamarlas en esta y cualquier otra función:

Figura 33 *Llamado de variables*

```
var monto = msg.payload;
var final = global.get("hf1");
var ingreso = global.get("hi1");
var nombre = global.get("n1");
var mail = global.get("m1");
var telegram = global.get("t1");

// Separar valores en variables individuales
var espacio = "Lugar 1";
```

Fuente: Elaboración propia

Todo esto finalmente es enviado y conectado con la primera función para poder enviar la notificación tanto por email como por telegram, llegando finalmente a hacer una revisión de espacios disponibles, limpiar el formulario de registro y actualizar la interfaz.

Figura 34 *Conexión de la primera función para envío por e-mail*

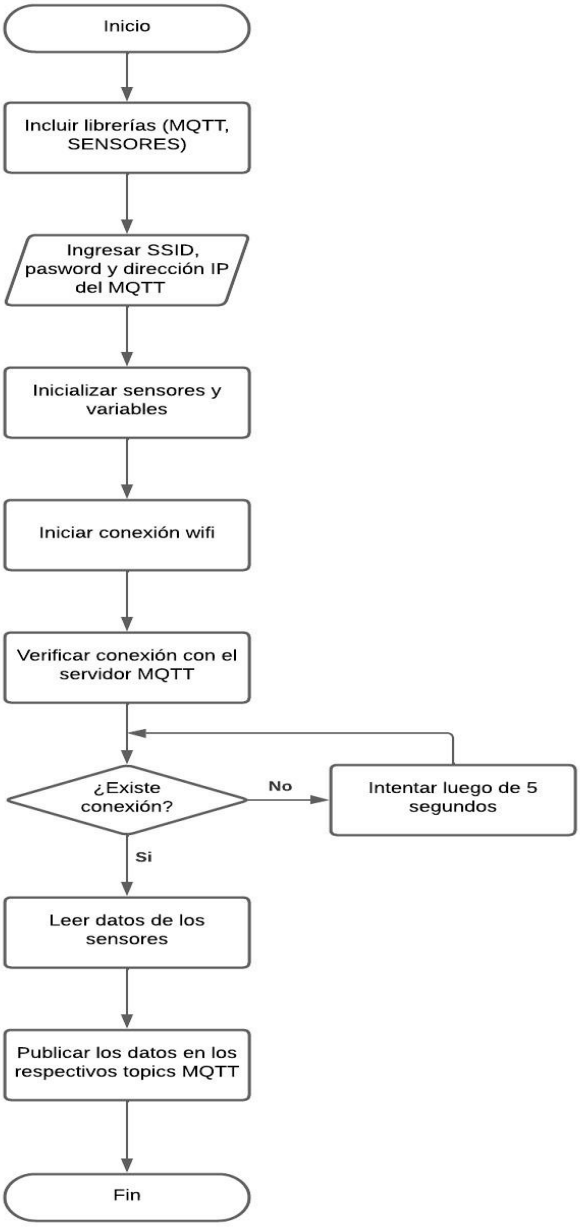
Fuente: Elaboración propia

3.8.6. Diagramas

3.8.6.1. ESP-32

En este diagrama se explica la funcionalidad del programa implementado para los sensores y el esp 32 para la obtención de la comunicación de los sensores del prototipo y como la comunicación con el broker mqtt.

Diagrama 2 *Diagrama de flujo – ESP-32*



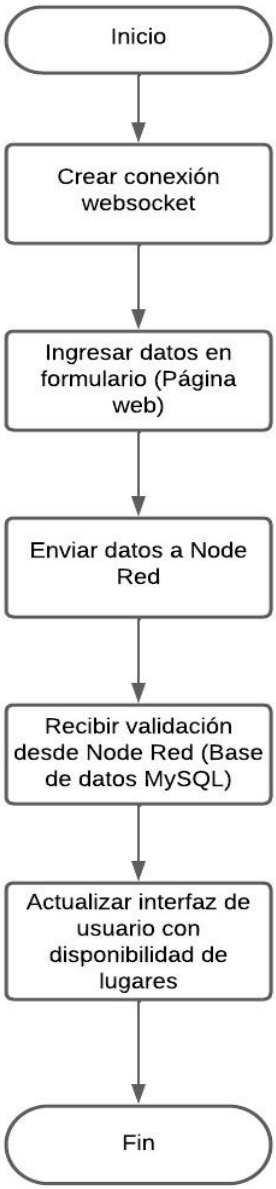
Fuente: Elaboración propia

3.8.6.2. Página WEB

En el diagrama 3 del presente trabajo tenemos el flujograma de la creación de la página web en visual estudio la cual nos permite la visualización de los espacios disponible, así como la selección de los mismos.

Integrando también la conexión del websocket con node-red.

Diagrama 3 *Diagrama de flujo – Página WEB*



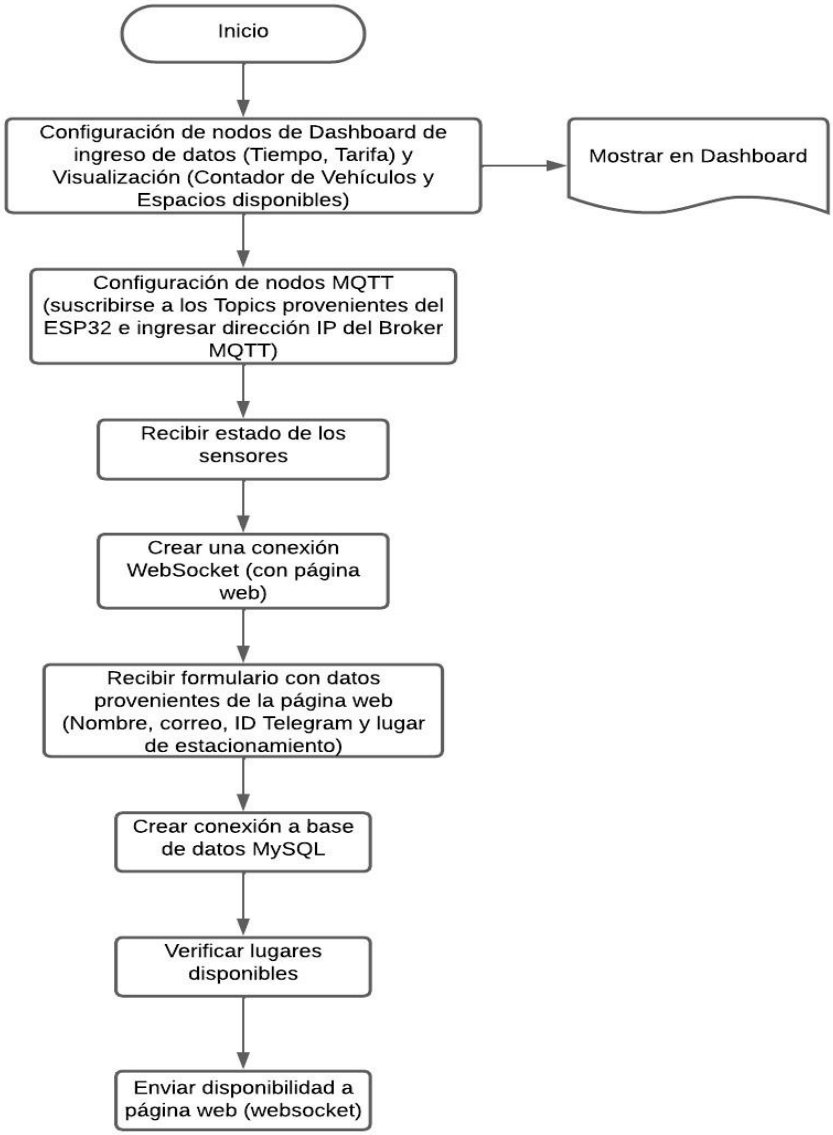
Fuente: Elaboración propia

3.8.6.3. Node red

En este diagrama nos da a conocer el proceso y programación en Node-red que se realiza para la comunicación del broker con la rasberry, el esp32 y la página web creada y así obtener la toma del tiempo de uso del estacionamiento, además del ingreso de la tarifa. También se integra un Dashboard para la visualización de la comunicación de los sensores con el Node-red.

Diagrama 4 *Diagrama de flujo – Node Red*

Fuente: Elaboración propia



Fuente: Elaboración propia

Resultados

1.- Tabla de datos por día

Tabla 1 *Tabla de datos por día*

HORAS DIA	ESPACIO 1	ESPACIO 2	ESPACIO 3	INGRESO E1	SALIDA E1	INGRESO E2	SALIDA E2	INGRESO E3	SALIDA E3
0:00:00	0	0	0						
1:00:00	0	0	0						
2:00:00	0	0	0						
3:00:00	0	0	0						
4:00:00	0	0	0						
5:00:00	0	0	0						
6:00:00	0	0	0						
7:00:00	0	0	0						
8:00:00	1	0	0	8:25:31	8:55:50				
9:00:00	0	1	1			9:15:11		9:05:24	
10:00:00	1	1	1	10:05:01	10:45:54				
11:00:00	0	1	1						11:07:28
12:00:00	0	1	0				12:18:57		
13:00:00	0	0	1					13:19:21	13:51:49
14:00:00	0	0	0						
15:00:00	1	1	0	15:02:51	15:58:49	15:12:28	15:31:16		
16:00:00	0	0	0						
17:00:00	0	0	0						
18:00:00	0	0	0						
19:00:00	0	0	0						
20:00:00	0	0	0						
21:00:00	0	0	0						
22:00:00	0	0	0						
23:00:00	0	0	0						

Fuente: Elaboración propia

Los datos de la tabla proporcionan un análisis detallado del uso del parqueo a lo largo del día, desglosando el flujo de vehículos en tres espacios distintos. Se observa que el espacio 1 y el espacio 2 tienen un uso más frecuente en comparación con el espacio 3, siendo utilizados en diversas horas del día. En este caso, se observa que el espacio 1 registra un ingreso a las

8:25:31 y una salida a las 8:55:50, mientras que el espacio 2 tiene un ingreso a las 9:00:00 y una salida a las 9:05:24, seguido de un nuevo ingreso a las 9:15:11 y una salida a las 9:05:24, lo que sugiere una alta rotación de vehículos en ese espacio. Por otro lado, el espacio 3 muestra una menor actividad, con solo tres eventos registrados durante el día. Estos datos son fundamentales para comprender los patrones de uso del parqueo, identificar picos de demanda y posibles áreas de mejora en la gestión del espacio disponible.

2.- Resultados de cada espacio en el día

Tabla 2 *Resultados de cada espacio en el día*

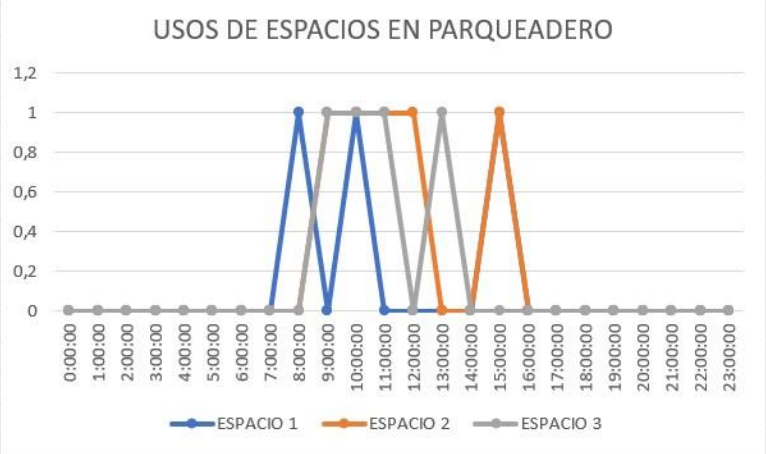
Total	ESPACIO 1	ESPACIO 2	ESPACIO 3
	3	2	2

Fuente: Elaboración propia

La Tabla 2 muestra los resultados de cada espacio de estacionamiento a lo largo del día, registrando un total de 3 eventos. Se observa una distribución equitativa de la actividad, con 2 eventos en cada uno de los Espacios 1 y 2, y también 2 eventos en el Espacio 3. Aunque la actividad total es moderada, esta distribución sugiere una demanda equilibrada entre los espacios, con el Espacio 3 contribuyendo de manera significativa a pesar de su menor actividad.

3.- Histograma del uso del parqueo

Figura 35 *Histograma de uso del parqueo*

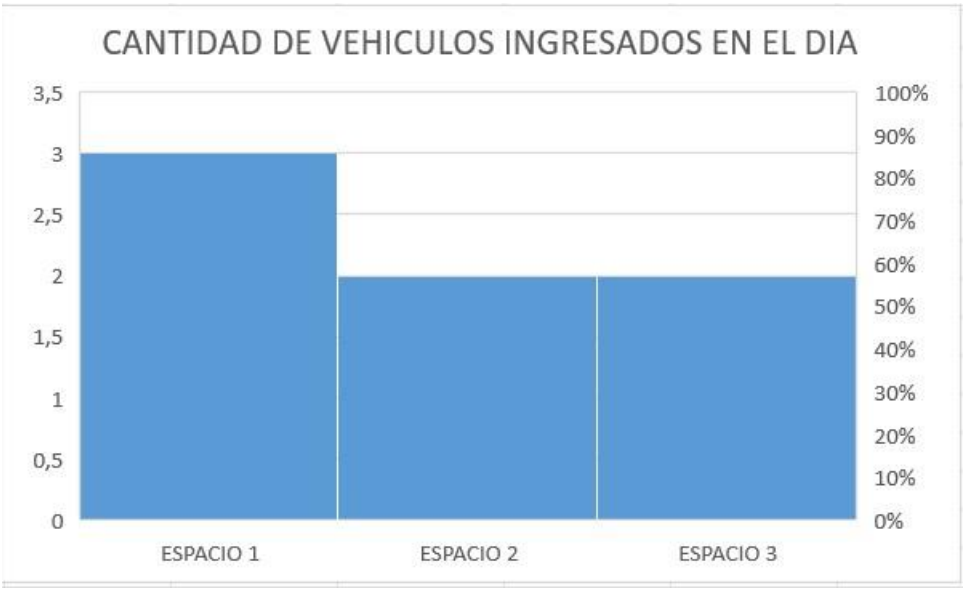


Fuente: Elaboración propia

El histograma representa de manera visual el uso de los espacios en el parqueadero a lo largo del día, donde cada barra corresponde a la frecuencia de eventos de estacionamiento en cada espacio. Se observa que tanto el Espacio 1 como el Espacio 2 presentan una distribución similar, con una frecuencia de 2 eventos cada uno, mientras que el Espacio 3 muestra una frecuencia también de 2 eventos. Esta representación destaca la igualdad en la utilización de los espacios durante el período de observación, sugiriendo una distribución equitativa de la demanda entre ellos.

4.- Flujo de vehículos registrados

Figura 36 *Flujo de vehículos registrados*

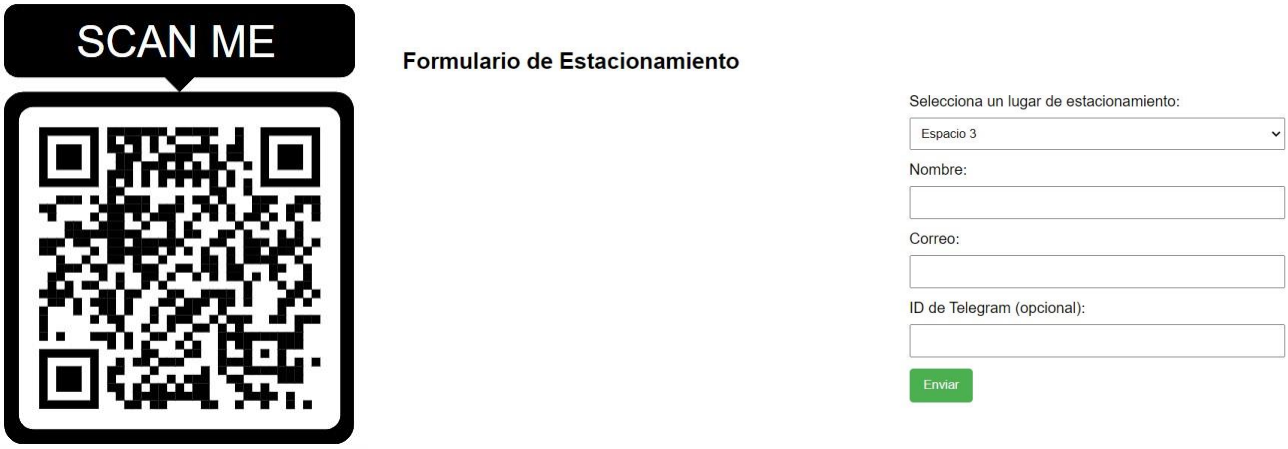


Fuente: Elaboración propia

En este apartado se obtuvo un diagrama de bloques de la cantidad de vehículos ingresados en cada espacio dentro de 24 horas, y tal como se puede apreciar, en el espacio 1 ha habido una mayor afluencia de ingreso de vehículos. Seguidos del espacio 2 y 3 quienes presentan una cantidad equilibrada en cuanto a la cantidad ingresada.

5.- QR para la página web dirigida a los usuarios

Figura 37 *QR para la página web de usuarios registrados*

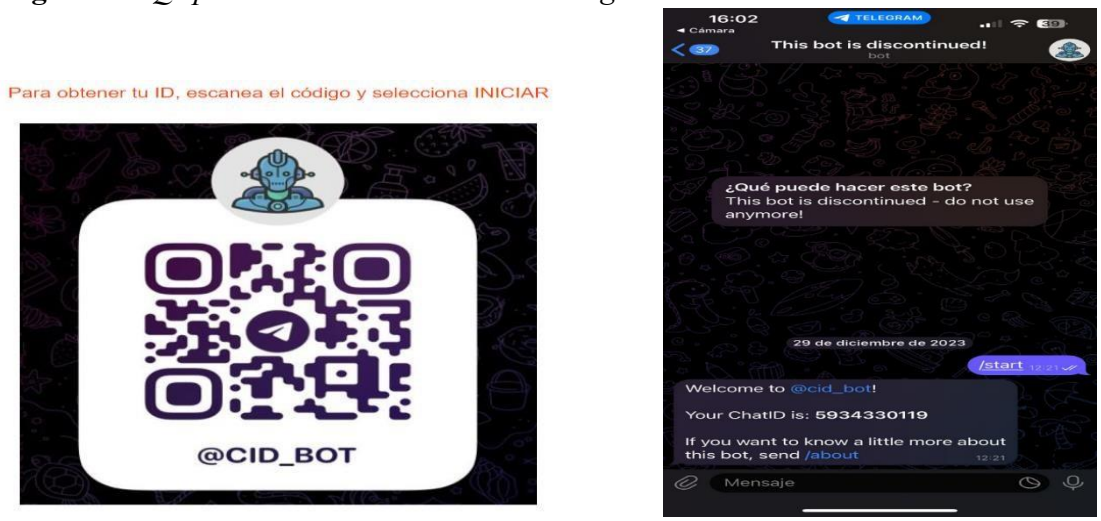


Fuente: Elaboración propia

La figura muestra el código QR para visualización de espacios en el parqueadero y registro del usuario. Al escanear, al usuario se le mostrará la cantidad de espacios disponibles, a fin de que pueda seleccionar el que más le convenga acorde a su necesidad.

6.- QR para obtención de ID de Telegram

Figura 38 *Qr para la obtención de ID en Telegram*



Fuente: Elaboración propia

Con este código el usuario podrá obtener su ID de telegram para el registro en la selección del espacio para envió de mensajes de facturación a telegram.

7.- Dashboard del control del parqueadero.

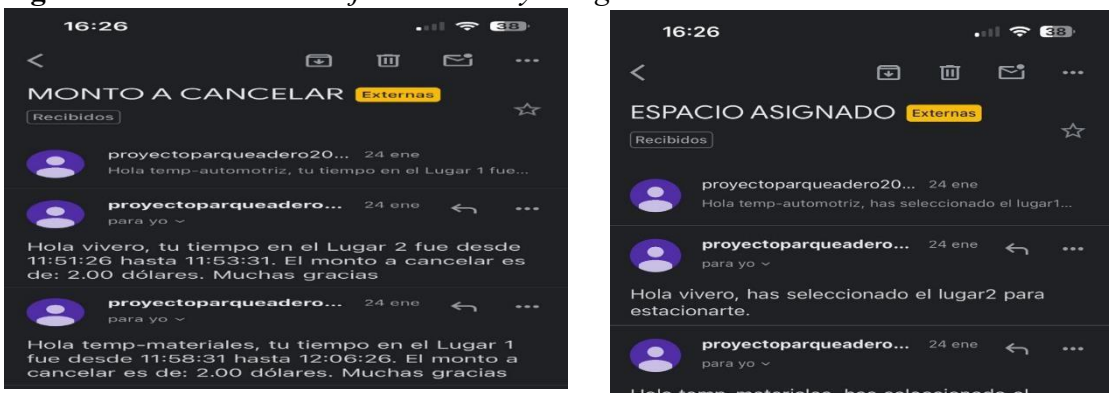
Figura 39 *Dashboard del control del parqueadero*

Fuente: Elaboración propia

Esta figura muestra cómo se puede verificar la llegada del vehículo al sensor, tal como se aprecia, se ingresa la tarifa y tiempo, y una vez seleccionado, se marca el espacio en el apartado de indicadores, el cual cambia su estado.

8.- Envío de mensajes al correo y telegram de los usuarios

Figura 40 *Envío de mensajes al correo y Telegram de los usuarios*



Fuente: Elaboración propia

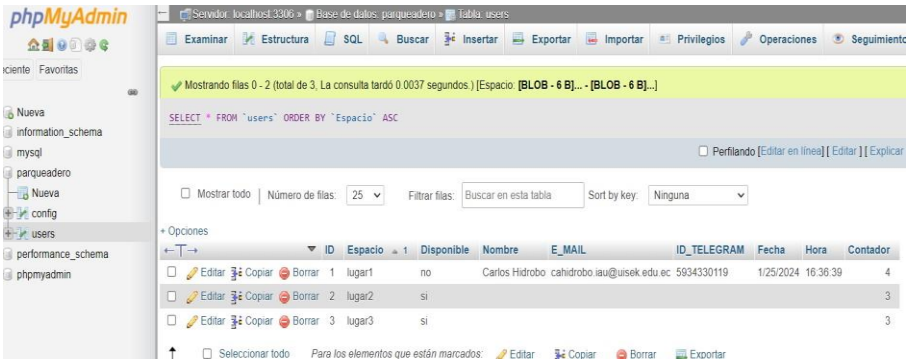
La figura muestra como son enviados los mensajes a través de correo electrónico y el Telegram de los usuarios. En este caso se muestran dos escenarios, el primero donde se hace referencia

al lugar para estacionarse, y el segundo, donde se avisa al usuario acerca de su tiempo transcurrido y la totalidad de su monto a cancelar.

9.- Base de datos

Figura 41

Dashboard del control del parqueadero



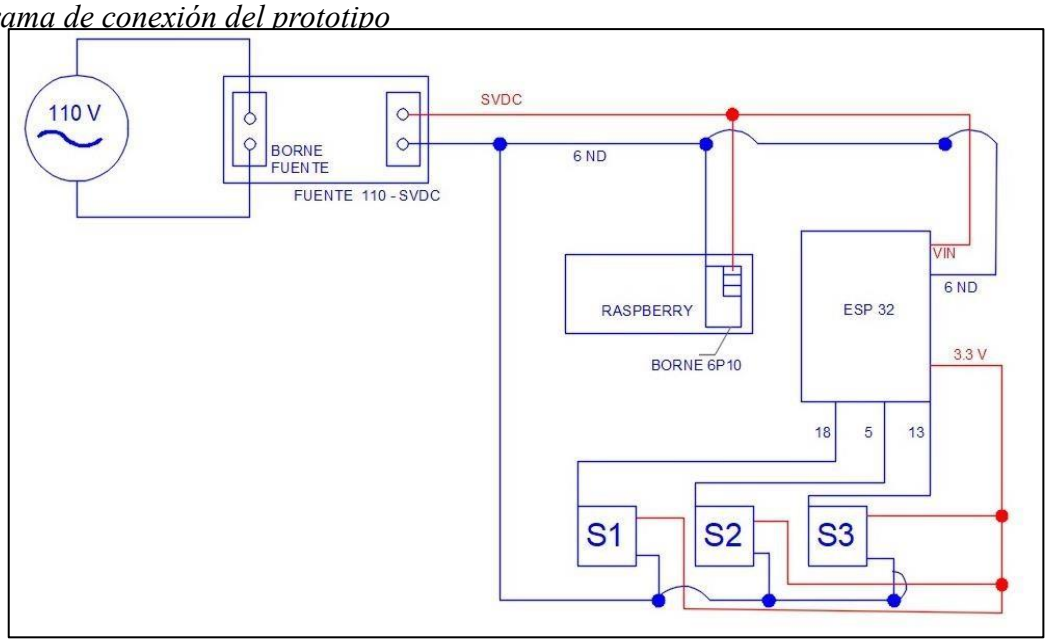
Fuente: Elaboración propia

En este resultado se obtiene la página creada en phpmyadmin para el registro de datos del usuario, donde se muestran los mensajes de selección y tiempo de uso del parqueadero.

10.- Diagrama de conexión del prototipo.

Figura 42

Diagrama de conexión del prototipo

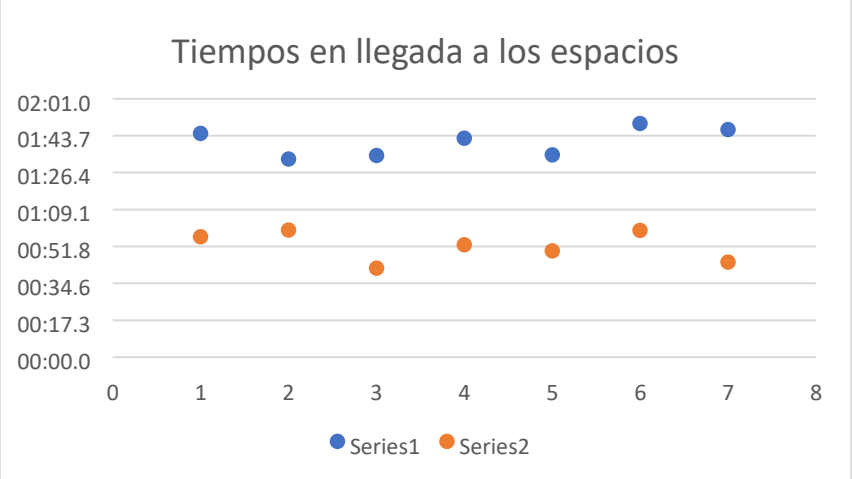


Fuente: Elaboración propia

El diagrama muestra la conexión de una fuente de alimentación con una Raspberry Pi y un ESP32, junto con sus sensores correspondientes. Esta disposición garantiza el suministro de energía constante a los dispositivos y facilita la recolección de datos ambientales. La integración de los sensores en el sistema permite monitorear variables como temperatura, humedad o movimiento. La estructura del diagrama refleja la base técnica del proyecto y facilita la comprensión de su funcionamiento, además de señalar la necesidad de medidas de seguridad y regulación de voltaje para un desempeño óptimo.

11.- Histograma de las pruebas realizadas evaluando la eficiencia del prototipo.

Figura 43 *Tiempos en llegada a los espacios*



Fuente: Elaboración propia

La Serie Uno comprende la recolección de tiempos cuando el usuario no cuenta con la capacidad de verificar la disponibilidad de espacios de estacionamiento, lo que implica una búsqueda activa de lugares libres dentro del área designada. Por otro lado, la Serie Dos corresponde a la toma de tiempos utilizando el prototipo IoT, donde los usuarios tienen la capacidad de seleccionar un espacio específico y dirigirse directamente hacia él, lo que simplifica y agiliza el proceso de estacionamiento al eliminar la necesidad de buscar un lugar disponible de manera manual.

Discusión de resultados

La investigación realizada por Barake et al. (2023) se centra en proporcionar información en tiempo real sobre la disponibilidad de estacionamiento en entornos urbanos. Aunque su enfoque difiere ligeramente del presente estudio, ambos comparten el objetivo de mejorar la experiencia de los usuarios en la búsqueda de espacios de estacionamiento. Mientras que Barake et al. se centran en la disponibilidad en entornos urbanos, el presente proyecto se enfoca en un contexto universitario.

Sin embargo, ambos proyectos resaltan la importancia de la implementación exitosa de sistemas digitales para mejorar la experiencia del usuario, reducir el estrés asociado a la búsqueda de espacios y transformar la interacción con los estacionamientos. En el caso de este proyecto, a través del uso de IoT y sensores, ofrece una solución específica para la gestión eficiente del estacionamiento en un entorno universitario, lo que representa un avance significativo en términos de adaptabilidad y personalización para las necesidades específicas de la comunidad universitaria.

Por otro lado, el trabajo de Campoz y Guillin (2022) también presenta similitudes con el presente proyecto en términos de utilizar una red IoT con sensores para monitorear la disponibilidad de espacios en tiempo real. Ambos proyectos buscan mejorar la gestión del estacionamiento y reducir los inconvenientes asociados con la búsqueda de espacios vacíos. No obstante, mientras que su estudio se enfoca en un parqueadero específico en el área de FIEC, este proyecto se desarrolla en un contexto universitario más amplio. Además, el presente trabajo incorpora elementos adicionales, como la creación de códigos QR para el acceso a la información de disponibilidad y la integración de un Dashboard de Node-red para la visualización de datos, lo que amplía la funcionalidad y la utilidad del sistema.

Por consiguiente, el trabajo de Saltos (2018) presenta una perspectiva similar en cuanto al uso de tecnología IoT para el diseño de un sistema de parqueo inteligente. Ambos proyectos se enfocan en el monitoreo en tiempo real de las plazas de estacionamiento y utilizan sensores para detectar la presencia de vehículos. Sin embargo, este proyecto se diferencia al integrar un mayor nivel de conectividad y funcionalidad, como el envío de mensajes al correo electrónico del usuario y la creación de una página para el registro de datos en caso de fallos de energía. Estas adiciones mejoran la experiencia del usuario y la confiabilidad del sistema, lo que representa una mejora significativa en comparación con enfoques anteriores.

Dentro de mismo contexto, el estudio de Vijay (2021) se centra en el diseño e implementación de un prototipo para un sistema de parqueo utilizando una aplicación móvil. Aunque su enfoque difiere en términos de la plataforma utilizada, ambos proyectos comparten el objetivo de mejorar la eficiencia y la experiencia del usuario en la gestión del estacionamiento. Mientras que el proyecto de Vijay se centra en una solución móvil, el presente trabajo utiliza una combinación de tecnologías IoT y sensores para proporcionar información en tiempo real sobre la disponibilidad de espacios de estacionamiento. Además, este proyecto ofrece una mayor flexibilidad al proporcionar múltiples opciones de acceso a la información, como códigos QR y correo electrónico, lo que aumenta la accesibilidad para los usuarios.

Por último, el trabajo de Méndez (2021) se enfoca en el diseño de un parqueo inteligente utilizando Arduino a través de IoT. Aunque ambos proyectos comparten el objetivo de implementar tecnología IoT para mejorar la gestión del estacionamiento, difieren en cuanto a la plataforma utilizada y los enfoques de comunicación. Mientras que el estudio de Méndez se centra en la comunicación utilizando RS-485, el presente proyecto utiliza una combinación de WiFi y Bluetooth para la transmisión de datos.

Además, este proyecto se destaca por su enfoque en la personalización y la adaptabilidad, como se evidencia en la creación de códigos QR para el acceso a la información de disponibilidad y la integración de un Dashboard de Node-red para la visualización de datos en tiempo real. Estas características adicionales mejoran la experiencia del usuario y la eficiencia del sistema en comparación con enfoques anteriores.

Conclusiones

El diseño e implementación del prototipo de parqueo inteligente logró cumplir con el objetivo general del proyecto, demostrando la eficacia de combinar tecnologías de Internet de las Cosas (IoT) y sensórica avanzada para facilitar un sistema altamente eficiente en la identificación y sugerencia de espacios de estacionamiento disponibles. Este enfoque innovador no solo optimizó la gestión de los espacios de estacionamiento, sino que también mejoró la experiencia del usuario al reducir el tiempo y el estrés asociados con la búsqueda de un lugar libre. La aplicación de estas tecnologías permitió el desarrollo de un sistema interconectado que opera en tiempo real, proporcionando actualizaciones instantáneas sobre la disponibilidad de estacionamiento.

En relación con el primer objetivo específico, la implementación efectiva de la sensórica y el control a través del microcontrolador Raspberry Pi y Node-Red, utilizando Python y Arduino para la comunicación del sistema de control, representó un hito clave en el proyecto. Esta configuración proporcionó una base sólida para el procesamiento y manejo de datos, asegurando una comunicación fluida entre los componentes del sistema. La integración de estas tecnologías permitió el monitoreo preciso de los espacios de estacionamiento, la activación de los sensores al detectar vehículos y la actualización en tiempo real del estado de los espacios en la interfaz de usuario.

El segundo objetivo se centró en la integración de la sensórica y la comunicación IoT para la transmisión de datos hacia el microcontrolador, empleando las direcciones IP proporcionadas por el ESP32 y el protocolo MQTT. Esta estrategia de comunicación aseguró un flujo constante de información entre los sensores de estacionamiento y el servidor, permitiendo la actualización dinámica de la disponibilidad de los espacios en la página web. La implementación exitosa de esta infraestructura de comunicación resalta la importancia de una red confiable y segura para el manejo de datos en sistemas de IoT, garantizando que los

usuarios reciban información precisa y oportuna para tomar decisiones informadas sobre dónde estacionar.

En lo que respecta al diseño de la página web para visualizar los espacios disponibles, este tercer objetivo específico fue alcanzado mediante el desarrollo de una interfaz de usuario intuitiva y accesible, programada en Visual Studio. Esta plataforma digital no solo facilita la interacción con el sistema por parte de los usuarios, sino que también destaca por su diseño adaptativo, el cual es compatible con una variedad de dispositivos móviles y computadoras. La página web se convierte así en un componente crucial del sistema, ofreciendo una solución práctica para la consulta de espacios de estacionamiento en tiempo real, lo que mejora significativamente la accesibilidad y la experiencia del usuario.

Finalmente, el quinto objetivo implicó la evaluación de la reducción de tiempos a través de un diseño experimental en tiempo real, confirmado la eficacia del sistema en la mejora de la gestión del estacionamiento. Los resultados de esta evaluación demostraron una disminución significativa en el tiempo que los usuarios gastan buscando estacionamiento, validando así el impacto positivo del prototipo en la optimización del uso de espacios y en la mejora de la experiencia general de estacionamiento. Este éxito subraya el potencial de aplicar tecnologías avanzadas de IoT y sensórica en la resolución de problemas cotidianos, marcando un avance significativo hacia la implementación de soluciones de estacionamiento inteligente más sostenibles y eficientes.

Recomendaciones

1.- Utilizar una rasberry con un buen procesador, para el presente trabajo se utilizó una rasberry pi 3 model B+ con un procesador ARM Cortex A53, con una frecuencia de 1.4 GHz.

2.- Verificar la versión de Raspbian para la instalación de mosquitto, si está utilizando Raspbian, debe tener Raspbian Jessie como versión mínima. Raspbian Búster es la versión compatible actualmente.

3.- Instalar el paquete de datos según la versión de Raspbian.

4.- Verificar que la memoria integrada a la rasberry sea capaz de almacenar y procesar todos los datos y programas que se vayan a ejecutar.

5.- Instalar realVNC para monitorear la rasberry desde un computador portátil.

6.- Verificar que la conexión red sea estable y con alta señal para que no existan interrupciones en el funcionamiento del sistema.

7.- El cableado tiene que mantenerse fijo para que no existan fallos en la conexión del prototipo.

8.- Tener los paquetes requeridos instalados en Arduino ide.

9.- Comprobar la conexión del websocket en Node-red al momento de ejecutar en la rasberry.

10.- Si el prototipo se conecta a otra red wifi verificar la nueva IP asignada y cambiar en los programas para volver a cargarlos.

Bibliografía

Al-Waisy, H., & Zaidan, A. (2019). Smart parking system: a survey. *Journal of Ambient Intelligence and Humanized Computing, 10*(5), 1785-1803. https://doi.org/https://doi.org/10.1007/s12652-018-0861-3

Asociación de Empresas Automotrices del Ecuador. (11 de enero de 2022). *Estadísticas del sector automotor muestran una recuperación productiva*. AEADE: https://www.aeade.net/

Barake, I., Guete, M., & Rueda, D. (2023). *Diseño y simulación de sistema de control de disponibilidad para parqueaderos de usuarios recurrentes [Tesis de grado, Universidad del Norte]*. https://manglar.uninorte.edu.co/bitstream/handle/10584/11823/Informe_Final_Barake_Guete_DR.pdf?sequence=1&isAllowed=y

Calderon, J., y Escalante, E. (2023). Movilidad y transporte inteligente: Una revisión de aplicaciones y tecnologías emergentes en el contexto de una ciudad inteligente. *Revista Ingeniería, Matemáticas y Ciencias de la Información, 10*(20), 79-87.

Calot, E., Maluf, M., & Neffa, M. (2017). *Estacionamiento Inteligente con IoT.* Universidad Nacional de La Plata, Facultad de Informática.

CamarasVigilancias. (30 de abril de 2023). *¿Cuál es la diferencia entre LPR y ANPR?* CamarasVigilancias: https://www.camarasvigilancias.com/cual-es-la-diferencia-entrelpr-y-anpr/

Campoz, C., y Guillin, J. (2022). *Diseño e implementación de un parqueadero automatizado en el edificio principal de la FIEC utilizando una red de sensores basada en IoT [Proyecto Integrado, Escuela Superior Politécnica del Litoral]*. https://www.dspace.espol.edu.ec/bitstream/123456789/57154/1/T-113083%20%20Campoz%20-%20Guillin.pdf

Cárdenas, K., Lugo, J., & Trujillo, L. (2023). *U'Parking: Desarrollo de un aplicativo PWA para el control y gestión de estacionamiento en tiempo real [Tesis de grado, Corporación Universitaria Minuto de Dios]*. http://uniminuto-dspace.scimago.es:8080/bitstream/10656/17610/2/T.IS_C%c3%a1rdenasKevinLugoJhon-TrujilloLuis_2023.pdf

Cubillos, N., & Rodríguez, J. (2018). *Arquitectura IoT para parqueaderos inteligentes en la ciudad de Bogotá.* Universidad Católica de Colombia.

Cuzco, D., & Pinos, S. (2022). *Diseño de un sistema de parqueo inteligente para el centro histórico de la ciudad de Cuenca, aplicando una red LPWAN [Tesis de grado, Universidad Politécnica Salesiana]*. https://dspace.ups.edu.ec/bitstream/123456789/22405/1/UPS-CT009717.pdf

Dimonoff. (18 de agosto de 2023). *Componentes del sistema de gestión inteligente de aparcamientos*. Dimonoff Movilidad: /soluciones/movilidad-aparcamientosinteligente/productos/

El Universo. (10 de noviembre de 2023). *Inseguridad y economía impactan a la venta de vehículos nuevos que en octubre cayó 6,3 %.* ElUniverso: https://www.eluniverso.com/noticias/economia/inseguridad-y-economia-impactan-ala-venta-de-vehiculos-nuevos-que-en-octubre-cayo-63-nota/

Espinoza, J. (2022). *Sistema de estacionamiento inteligente aplicando Internet de las Cosas (IoT), para gestionar el parqueo vehicular del garaje Ebenezer, Bagua Grande 2023 [Tesis de posgrado, Universidad César Vallejo]*. https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/108766/Espinoza_CJBSD.pdf?sequence=1&isAllowed=y

Estrella, G. (17 de febrero de 2023). *Nuevos parqueaderos inteligentes en Quicentro shopping*. Quicentro: https://www.quicentro.com/noticias/nuevos-parqueaderosinteligentes-en-quicentro-shopping/

Farnell. (24 de junio de 2023). *Ciudades inteligentes: soluciones de detección de estacionamiento inteligente con el IoT*. Farnell An Avnet Company.

Herrador, D. (2023). *Sistema de aparcamiento inteligente aplicado a las Smart Cities [Tesis de grado, Universidad Politécnica de Madrid]*. https://oa.upm.es/21414/1/PFC_DAVID_HERRADOR_MU%C3%91OZ.pdf

Imasdetres. (24 de septiembre de 2023). *Sensores de guiado de parking para detección de plazas ocupadas*. i+D3: https://imasdetres.com/mx/camaras-anpr-ip-lectoras-deplacas/

Méndez, E. (15 de septiembre de 2021). *Estudio y diseño de un parqueo inteligente utilizando Arduino a través del Internet de las cosas (IoT) [Tesis de grado, Universidad Católica de Santiago de Guayaquil]*. http://repositorio.ucsg.edu.ec/bitstream/3317/17170/1/T-UCSG-PRE-TEC-ITEL414.pdf

Mercado, G., Álvarez, L., Bocaccini, L., Ledda, M., Membrives, J., Muros, M., . . . Dumé, S. (2022). *Diagnóstico y Metodología para la Implementación de "Internet of Things" en el Planeamiento y Desarrollo de Ciudades Inteligentes.* Instituto Regional de Estudios de Energía .

Node-RED. (6 de septiembre de 2022). *Low-code programming for event-driven applications*. Node-Red: https://nodered.org/

Paessler. (2023). *¿Qué es MQTT?* Paessler The monitoring Experts: https://www.paessler.com/es/it-explained/mqtt

Rodríguez, G., Santos, R., Ordaz, C., & López, J. (2019). Estacionamiento Inteligente. *Revista de Ingeniería Innovativa, 3*(9), 34-39.

Sai, K., & Suresh, K. (2017). Intelligent parking system for smart cities. *International Journal of Pure and Applied Mathematics, 114*(6), 149-158. https://doi.org/10.12732/ijpam.v114i6.7

Saltos, E. (2018). *Diseño de un prototipo de sistema de parqueo inteligente para el edificio de la FIE utilizando tecnologías basado en el internet de las cosas [Tesis de grado, Escuela Superior Politécnica de Chimborazo]*. http://dspace.espoch.edu.ec/bitstream/123456789/10930/1/98T00224.pdf

Superinventos. (25 de mayo de 2022). *Nuevas cámaras IP ANPR para el reconocimiento de matrículas*. Superinventos: https://superinventos.blog/2022/05/25/nuevas-camaras-ippara-el-reconocimiento-de-matriculas/

Tenorio, F., Bolaños, C., & Arce, E. (2022). Prototipo basado en IoT y Arduino® para validar la disponibilidad de estacionamientos en la sede norte de la UNIAJC. *Revista Sapientía, 14*(28), 33-47.

Traxpark. (5 de mayo de 2023). *Cámara LPR*. Traxpark: https://traxpark.com/camaras-lpr/

Velasco, J., & Pinzon, M. (2018). *Prototipo basado en internet de las cosas para el monitoreo ambiental como contribución a la orientación de Bucaramanga (Colombia) hacia Smart City.* Universidad Autónoma de Bucaramanga .

Verdejo, N. (9 de marzo de 2023). *Investigadores proponen sistema inteligente de estacionamiento asistido por IoT*. WhatsNew: https://wwwhatsnew.com/2023/03/09/investigadores-proponen-sistema-inteligentede-estacionamiento-asistido-por-iot/

Vijay, F. (2021). *Diseño e implementación de un prototipo para un sistema de parqueo con aplicación móvil.* https://dspace.ucacue.edu.ec/server/api/core/bitstreams/5956c2629ad8-40ce-bc04-adb8e9e6f4da/content

ANEXOS

Anexo 1 *Formato de Entrevista*

ENTREVISTA DIRIGIDA A LOS DOCENTES QUE HACEN USO DE PARQUEADERO EN LA UNIVERSIDAD INTERNACIONAL SEK

1. **¿Cómo describirían su experiencia actual al buscar espacio de estacionamiento en el campus universitario?**

_____

_____

_____

2. **¿Cuáles son los principales desafíos que enfrentan al intentar estacionar su vehículo durante las horas pico?**

_____

_____

_____

3. **¿Creen que la disponibilidad de espacio de estacionamiento impacta su productividad o puntualidad en el campus?**

_____

_____

_____

4. **¿Qué características consideran más importantes en un sistema de parqueaderos que facilite su experiencia de estacionamiento?**

5. ¿Han experimentado problemas relacionados con la falta de información en tiempo real sobre la disponibilidad de espacios de estacionamiento?

6. ¿Cómo creen que la implementación de tecnologías IoT y sensores podría mejorar la gestión de los parqueaderos universitarios?

7. ¿Qué preocupaciones o consideraciones tendrían respecto a la privacidad y seguridad de los datos si se implementara un sistema de parqueaderos basado en IoT?

8. ¿Qué recomendaciones o sugerencias tendrían para el diseño y la implementación efectiva de un sistema prototipo de parqueaderos basado en disponibilidad y tecnología IoT en nuestro campus universitario?

Anexo 2 *Evidencia del prototipo instalado en el área de estacionamiento*

Anexo 3 *Código de programación para el ESP 32*

```
#include <WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.
const char *ssid = "Ine4c_Electronics"; const char
*password = "Ine4c4000";
const char *mqtt_server = "192.168.100.229";

WiFiClient espClient;
PubSubClient client(espClient);

const int sensor1 = 14; const
int sensor2 = 18; const int
sensor3 = 5;
const int sensor4 = 13;

int PS1, PS2, PS3, PS4;


int cont, cont1, cont2, cont3, total; int
ct1, ct2, ct3;

String t1, cot1, cot2, cot3;
String est1, est2, est3;

int t00, t11, t22, t33;

int cest1, cest2, cest3;
int resp1, resp2, resp3;

char p1[10], p2[10], p3[10], p4[10]; char
p11[10], p22[10], p33[10], to1[10];
char resp11[10], resp22[10], resp33[10];

void setup() {
  pinMode(sensor1, INPUT);
pinMode(sensor2, INPUT);   pinMode(sensor3,
INPUT);   pinMode(sensor4, INPUT);

  Serial.begin(9600);   setup_wifi();
client.setServer(mqtt_server, 1883);
client.setCallback(callback);
```

```
}

void setup_wifi()
{
  delay(10);

  Serial.println();
  Serial.print("Conectando a:");
  Serial.println(ssid);
  WiFi.begin(ssid, password);   while
(WiFi.status() != WL_CONNECTED)
  {
delay(500);
    Serial.print(".");
}
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Direccion IP: ");
  Serial.println(WiFi.localIP());
}


void callback(char* topic, byte* payload, unsigned int length) {
 String mensaje = topic;
  Serial.print("Mensaje Recibido del topic: ");
 Serial.println(topic);

 if (mensaje  == "tiempo") {
t1 ="" ;
  for (int i = 0; i < length; i++) {
t1+=((char)payload[i]);
  }
 }

  /// CONDICIONES _ CONTADOR DE VEHICULOS
  if (mensaje  == "ct1") {
cot1 ="" ;
  for (int i = 0; i < length; i++) {
      cot1+=((char)payload[i]);
  }  }   if (mensaje  ==
"ct2") {    cot2 ="" ;
  for (int i = 0; i < length; i++) {
      cot2+=((char)payload[i]);
```

```
  } }   if (mensaje  ==
"ct3") {    cot3 ="" ;
  for (int i = 0; i < length; i++) {
cot3+=((char)payload[i]);     }
 }

 /// CONDICIONES _ ESTADO -SENSOR

  if (mensaje  == "estado1") {
est1 ="" ;     for (int i = 0; i <
length; i++) {
       est1+=((char)payload[i]);
    }
  }

  if (mensaje  == "estado2") {
est2 ="" ;     for (int i = 0; i <
length; i++) {
       est2+=((char)payload[i]);
    }
  }   if (mensaje  == "estado3") {
est3 ="" ;     for (int i = 0; i <
length; i++) {
       est3+=((char)payload[i]);
    }
  }

  else { Serial.println("error de mensaje"); }
    t00 = t1.toInt();
ct1 = cot1.toInt();
ct2 = cot2.toInt();
ct3 = cot3.toInt();

    Serial.print("TIEMPO: "); Serial.println(t00);

    Serial.print("estado1: "); Serial.println(est1);
    Serial.print("estado2: "); Serial.println(est2);
    Serial.print("estado3: "); Serial.println(est3);

    t11 =  t00;
t22 =  t00;     t33
=  t00;

}
```

```
void loop() {

  if (!client.connected()) { reconnect(); }
  client.loop();
  delay(100);

 // LECTURA DE PUERTOS

    PS1 = digitalRead(sensor1);
    PS2 = digitalRead(sensor2);
    PS3 = digitalRead(sensor3);
    PS4 = digitalRead(sensor4);


 // PUBLICACION PUERTOS DIGITALES
    if (est1== "no") { cest1 =1 ; }       else if
(est1== "si") {cest1 = 0; resp1 = 0;}

    if (est2== "no") { cest2 =1 ; }       else if
(est2== "si") {cest2 = 0; resp2 = 0;}

    if (est3== "no") { cest3 =1 ; }       else if
(est3== "si") {cest3 = 0; resp3 = 0;}


   if (PS2 == LOW && cest1 == 1) {
cont1=1; delay(200);          resp1++;
delay(200);          dtostrf(resp1, 0, 0, resp11);
if(resp1 == 1 ){
client.publish("VALIDACION1", resp11);
        }
    }

   else if (PS2 == HIGH && resp1 >= 1) {
       t11--;           delay(1000);
if(t11<=0){           t11=t00;
cont1=0;
client.publish("Fin1","E1");
         ct1++;           delay(200);           dtostrf(ct1,0, 0,
p11); client.publish("AUTOS1",p11);
         // resp1=0;
```

```
        }
}

    if  (PS3 == LOW && cest2 == 1 ) {
cont2=1; delay(200);          resp2++;
delay(200);          dtostrf(resp2, 0, 0,
resp22);
        if(resp2 ==1 ){
         client.publish("VALIDACION2", resp22);
        }
}

       else if (PS3 == HIGH && resp2 >= 1) {
          t22--;           delay(1000);
if(t22<=0){            t22=t00;
cont2=0;
client.publish("Fin2","E2");
          ct2++;            delay(200);           dtostrf(ct2,0, 0,
p22); client.publish("AUTOS2",p22);
        }
}

  if (PS4 == LOW && cest3 == 1) {
cont3=1; delay(200);          resp3++;
delay(200);          dtostrf(resp3, 0, 0,
resp33);
        if(resp3 ==1 ){
client.publish("VALIDACION3", resp33);
        }
}

 else if (PS4 == HIGH && resp3 >= 1) {
        t33--;           delay(1000);
if(t33<=0){            t33=t00;
cont3=0;
client.publish("Fin3","E3");
          ct3++;            delay(200);           dtostrf(ct3,0, 0,
p33); client.publish("AUTOS3",p33);
        }
}

     Serial.print("validacion1: "); Serial.println(resp1);
     Serial.print("validacion2: "); Serial.println(resp2);
     Serial.print("validacion3: "); Serial.println(resp3);
```

```
    Serial.print("t1: ");Serial.println(t11);
    Serial.print("t2: ");Serial.println(t22);
    Serial.print("t3: "); Serial.println(t33);
    Serial.print("estado1: "); Serial.println(est1);
    Serial.print("estado2: "); Serial.println(est2);       Serial.print("estado3: ");
Serial.println(est3);

    delay(500);

    total = ct1+ct2+ct3;

    dtostrf(cont,0, 0, p1); client.publish("SENSOR1",p1);
dtostrf(cont1,0, 0, p2); client.publish("SENSOR2",p2);
dtostrf(cont2,0, 0, p3); client.publish("SENSOR3",p3);
dtostrf(cont3,0, 0, p4); client.publish("SENSOR4",p4);

    dtostrf(total,0, 0, to1); client.publish("AUTOST",to1);

}

void reconnect() {


  while (!client.connected()) {
   Serial.print("Attempting MQTT connection...");
   // Attempt to connect
   if (client.connect("ESP8266Client")) {
Serial.println("connected");
client.publish("/conexion", "Conectado");
client.subscribe("tiempo");
client.subscribe("ct1");      client.subscribe("ct2");
client.subscribe("ct3");

    client.subscribe("estado1");
client.subscribe("estado2");      client.subscribe("estado3");

   } else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
   }
```

```
  }
}
```