Ciencia Latina Revista Científica Multidisciplinar, Ciudad de México, México.
ISSN 2707-2207 / ISSN 2707-2215 (en línea), noviembre-diciembre 2024,
Volumen 8, Número 6.

https://doi.org/10.37811/cl_rcm.v8i6

INTEGRACIÓN DE UN SISTEMA REMOTO DE SEGURIDAD PARA AUTOMÓVILES BASADO EN UNA TARJETA MICROCONTROLADA Y UNA APLICACIÓN ANDROID

INTEGRATION OF A REMOTE CAR SECURITY SYSTEM BASED ON A MICROCONTROLLER BOARD AND AN ANDROID APPLICATION

Pedro Guevara López
Instituto Politécnico Nacional

María Elena Mendiola Medellín
Instituto Politécnico Nacional

Heider Venancio Sánchez
Instituto Politécnico Nacional

Héctor Gabriel Ugarte Soto
Instituto Politécnico Nacional

Leslie Jacqueline Jacobo Vázquez
Instituto Politécnico Nacional

DOI: https://doi.org/10.37811/cl_rcm.v8i6.14493

Integración de un sistema remoto de seguridad para automóviles basado en una tarjeta microcontrolada y una aplicación Android

Pedro Guevara López[1]
pguevara@ipn.mx
https://orcid.org/0000-0001-5373-1403
Instituto Politécnico Nacional
México

María Elena Mendiola Medellín
mmendiola@ipn.mx
https://orcid.org/0000-0003-2121-2128
Instituto Politécnico Nacional
México

Heider Venancio Sánchez
hvenancios1400@alumno.ipn.mx
https://orcid.org/0009-0005-4626-9667
Instituto Politécnico Nacional
México

Héctor Gabriel Ugarte Soto
hugartes1500@alumno.ipn.mx
https://orcid.org/0009-0000-8064-6159
Instituto Politécnico Nacional
México

Leslie Jacqueline Jacobo Vázquez
ljacobov1300@alumno.ipn.mx
https://orcid.org/0009-0005-4626-9667
Instituto Politécnico Nacional
México

RESUMEN

Este trabajo presenta un sistema de seguridad remoto para automóviles, enfocado en reducir el riesgo de robo, especialmente en zonas metropolitanas de México donde los automóviles de modelos antiguos son vulnerables. El sistema utiliza una combinación de tecnologías accesibles como un módulo GPS U-blox NEO 6M y un módulo GSM/GPRS SIM800L, que están controlados por una tarjeta microcontrolada Raspberry Pi Pico. Estos componentes permiten el monitoreo de la ubicación del vehículo, enviando datos a un servidor web mediante protocolos GSM/GPRS. El sistema incluye una aplicación móvil Android, desde donde los propietarios pueden monitorear su vehículo y recibir alertas en caso de violaciones de seguridad. El proyecto resuelve varios problemas asociados con los sistemas de seguridad vehicular tradicionales, como la complejidad de los sistemas de rastreo GPS y la falta de notificaciones rápidas. Los resultados obtenidos del sistema en pruebas del hardware como en la comunicación con el servidor web y la aplicación móvil, fueron satisfactorios, cumpliendo con los objetivos propuestos de brindar mayor seguridad y tranquilidad a los propietarios de vehículos.

Palabras Clave: gps; gsm/gprs; rasperry pi pico; seguridad para automóvil; api rest.

[1] Autor principal
Correspondencia: pguevara@ipn.mx

Integration of a remote car security system based on a microcontroller board and an Android application

ABSTRACT

This paper presents a remote security system for automobiles, focused on reducing the risk of theft, especially in metropolitan areas of Mexico where older model cars are vulnerable. The system uses a combination of affordable technologies such as a U-blox NEO 6M GPS module and a SIM800L GSM/GPRS module, which are controlled by a Raspberry Pi Pico microcontroller board. These components enable vehicle location monitoring, sending data to a web server via GSM/GPRS protocols. The system includes an Android mobile application, from where owners can monitor their vehicle and receive alerts in case of security breaches. The project solves several problems associated with traditional vehicle security systems, such as the complexity of GPS tracking systems and the lack of fast notifications. The results obtained from the system in hardware tests, as well as in communication with the web server and mobile application, were satisfactory, meeting the proposed objectives of providing greater security and peace of mind to vehicle owners.

Keywords: gps; gsm/gprs; raspberry pi pico; car security; rest api.

Artículo recibido 09 octubre 2024

Aceptado para publicación: 13 noviembre 2024

INTRODUCCIÓN

Este proyecto propone a los propietarios de vehículos una solución confiable y eficiente para proteger sus automóviles contra robos y asegurar un monitoreo constante de su ubicación y estado de seguridad (T21, 2023), (El Financiero, 2023). El sistema de seguridad remota se basa en el uso de un módulo GPS U-blox NEO 6M y sensores conectados a una placa microcontrolada Raspberry Pi Pico, donde ésta, actúa como el centro de control del sistema, recopilando los datos provenientes de los sensores y del módulo GPS (U-blox, 2011). Estos datos son posteriormente enviados a un servidor web a través de la web utilizando la red de datos móviles utilizando un módulo GSM/GPRS SIM800L. El Servicio Web se encarga de almacenar y gestionar los datos obtenidos por los módulos y sensores, esto permite a los propietarios tener acceso a la información del vehículo en cualquier momento y desde cualquier lugar mediante una aplicación móvil Android desarrollada específicamente para este propósito. La aplicación móvil proporciona una interfaz de usuario, donde se pueden consultar los datos del vehículo mediante un mapa y recibir alertas si algún sensor es activado o si el automóvil entra en movimiento sin autorización. La integración de este sistema de seguridad remoto ofrece beneficios para los propietarios de automóviles particulares en zonas metropolitanas de México; entre ellos se incluyen una mayor tranquilidad y seguridad al saber que su vehículo está siendo constantemente monitoreado y una mayor capacidad de respuesta en caso de intento de robo o sustracción.

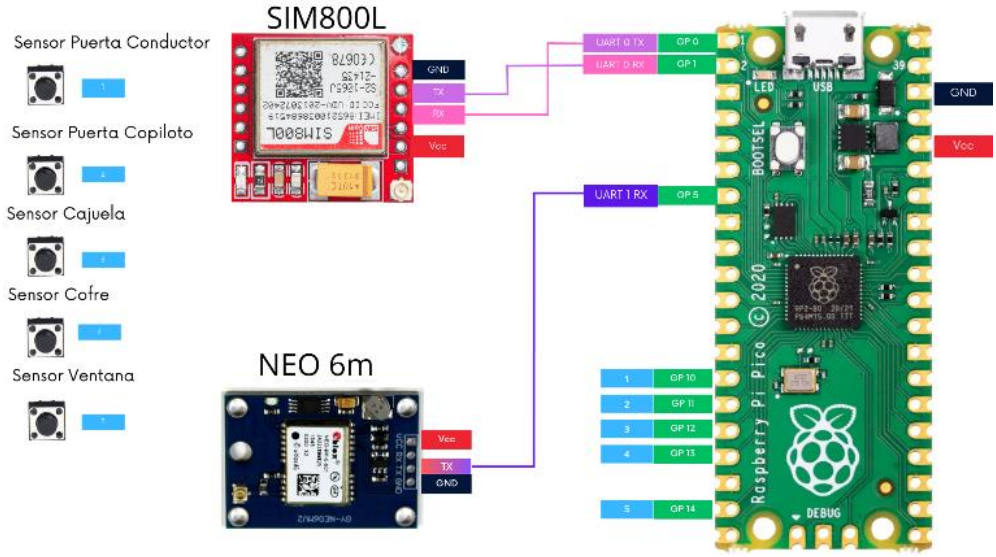
METODOLOGÍA

Este proyecto se integra por tres módulos: a) Circuito eléctrico, b) Servicio Web y c) Aplicación móvil.

Módulo A) Circuito eléctrico

Raspberry Pi, como placa microcontrolada ha sido usada en diversos proyectos, como los mostrados en Pavithra, M., & Jyothi, S. (2018) y Singh, P., & Sethi, T. (2020); esto ha motivado el uso de esta placa para el desarrollo del módulo A), que tiene la conexión física de los módulos a la placa microcontrolada Raspberry Pi Pico y programar su funcionamiento para que los módulos funcionen correctamente y se comuniquen con la placa microcontrolada. La conexión se puede ver en el diagrama de la Figura 1.

Figura 1. Diagrama de conexión del circuito utilizado en el sistema de seguridad.



Como se ve en la Figura 1, los módulos GPS NEO-6M y GSM/GPRS SIM800L están conectado de los pines que admiten el protocolo UART (Legaspi, 2023) de la placa Raspberry Pi Pico, ya que este protocolo de comunicación permite definir la forma en que los datos se transmiten y reciben a través de dos líneas de comunicación: línea de transmisión (TX) y la línea de recepción (RX). Los datos se envían en paquetes de bits, donde cada paquete contiene un bit de inicio, un número de bits de datos (entre 5 y 8 bits), un bit de paridad opcional y uno o más bits de parada. La velocidad de transmisión de datos en un enlace UART se define mediante una tasa de baudios, que representa el número de símbolos transmitidos por segundo (Campbell, 2023). Los dispositivos que se comunican deben configurarse para utilizar la misma velocidad de baudios para garantizar una transferencia de datos confiable que en este caso es de 9600 baudios para los componentes utilizados (NASA, 2023a), (NASA, 2023b).

Los módulos GPS se han utilizado para rastreo de vehículos como se muestra en Al-Rashed, M. A., Oumar, O. A., & Singh, D. (2014); en este sentido, para hacer uso de los datos obtenidos por el módulo GPS se utilizó la biblioteca micropyGPS (McCoy, 2017) , que permite la manipulación de los datos obtenidos por el módulo en el lenguaje MycroPhython. Esta biblioteca es de código abierto publicada en el repositorio GitHub por el usuario "inmcm". Dentro del algoritmo desarrollado para la obtención de las coordenadas, se obtiene el valor de latitud y longitud para después almacenar sus valores en variables. Mientras que para el caso del módulo SIM800L se programaron los comandos AT (Attention), éstos son una serie de instrucciones en formato de texto que se envían a través de un puerto

serie para controlar el comportamiento del dispositivo; para ello de utiliza el puerto serie COM3, que es el que Windows asigna por defecto a los dispositivos que requieren comunicación serie. Los comandos utilizados con sus respectivos propósitos pueden ser apreciados en la Figura 2. Una vez confirmados estos parámetros, el módulo puede acceder a la red móvil y enviar los datos al servidor a través la web y realizar peticiones HTTP al servicio web.

Figura 2. Comandos AT para establecer conexión con la red de datos móviles.

```
# Solicitar informacion de Registro de Red
uart.write('AT+CREG?\r\n')
time.sleep(1)
# Establecer Parametros de Contexto para conexion de datos
uart.write('AT+CGDCONT=1,"192.168.1.66","internet.itelcel.com"\r\n')
time.sleep(1)
# establecer el APN (Access Point Name)
uart.write('AT+CSTT="internet.itelcel.com"\r\n')
time.sleep(10)
# Conectar a la red GPRS
uart.write('AT+CIICR\r\n')
time.sleep(10)

# Configurar el acceso a Internet
uart.write('AT+CIFSR\r\n')
```

Regresando a la Figura 1, se observa que el circuito cuenta con la conexión de 5 botones que sirven para transmitir una señal que indica que ocurrió una violación de seguridad; hay que tomar en cuenta que los botones simulan las señales de los sensores que serán instalados dentro del vehículo cuando se realice la implementación del sistema. La señal de estos botones es recopilada por la placa Raspberry Pi Pico y si son presionados cambian el estado de una variable de tipo booleana en el algoritmo del circuito. Estas variables booleanas junto a las variables que contienen la información de latitud y longitud son introducidas dentro de un diccionario. Cada clave en el diccionario representa una etiqueta o nombre de una variable, y el valor asociado a esa clave es el valor de la variable correspondiente. Luego, el diccionario "data" se convierte en una cadena JSON utilizando la función "json.dumps(data)", que convierte el diccionario en una representación JSON (Google, 2021). Posteriormente, el código crea un archivo llamado "data.json" en modo de escritura ('w') y escribe la cadena JSON en el archivo utilizando el método "f.write(json_data)", (Ver Figura. 3).

Figura 3. Creación del archivo JSON con los valores de las variables.

```
data = {
    "latitud": latitud,
    "longitud": longitud,
    "cofre": cofre,
    "cajuela": cajuela,
    "puerta1": puerta1,
    "puerta2": puerta2,
    "ventana": ventana
}


json_data = json.dumps(data)

with open('data.json', 'w') as f:
    f.write(json_data)
    json_data = f.read()
```

Una vez obtenido el formato JSON de los datos, se procede a enviar una petición de tipo POST hacia el endpoint del servicio web, donde se envían los datos por medio del archivo JSON utilizando la biblioteca de MycroPython "urequest" la que permite utilizar el protocolo HTTP para distintos usos, como se muestra en la Figura 4.

Figura 4. Petición POST hacia el endpoint del servicio web

```
with open('data.json', 'r') as f:
    json_data = f.read()
    url = 'http://192.168.3.81:8080/seguridad/coordenadasGPS'
    headers = {'Content-Type': 'application/json'}
    response = urequests.post(url, data=json_data, headers=headers)
```

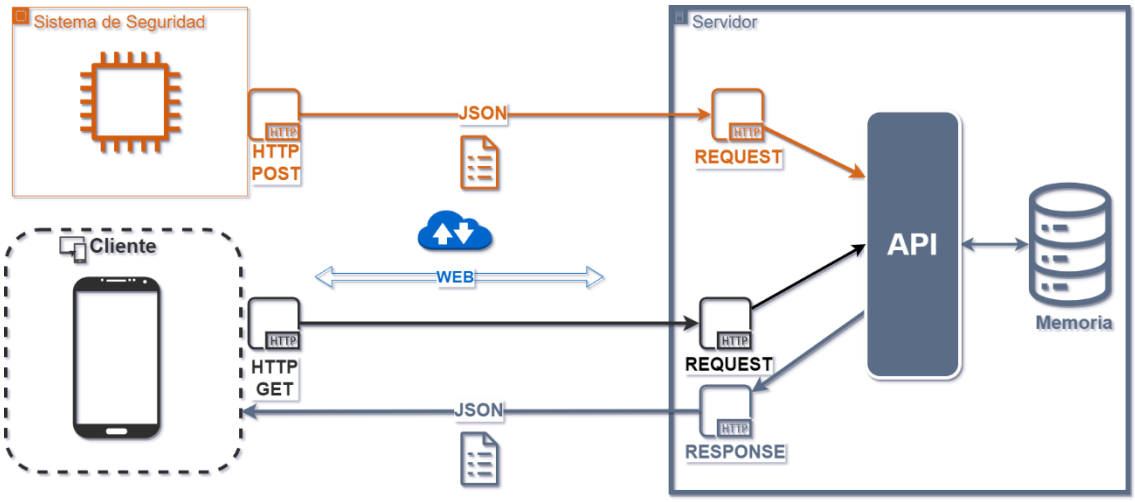
Módulo B) Servicio Web

Para el desarrollo del Web Service se utilizó una arquitectura basada en REST (Representational State Transfer). Esta arquitectura se basa en los siguientes elementos:

1. Recursos: Son los objetos a los que se puede acceder a través de la API. En este caso, el objeto es el archivo JSON que es enviado por el circuito y es alojado en la memoria del servidor.
2. URI (Identificador de Recursos Uniforme): Es la dirección que se utiliza para acceder a un recurso del proyecto.
3. Métodos HTTP: Se utilizan los verbos POST y GET para la escritura y lectura de la información alojada en el servidor. Esto se muestra en la Figura 5.
4. Respuestas: Son las que se devuelven al cliente después de realizar una solicitud. En este caso

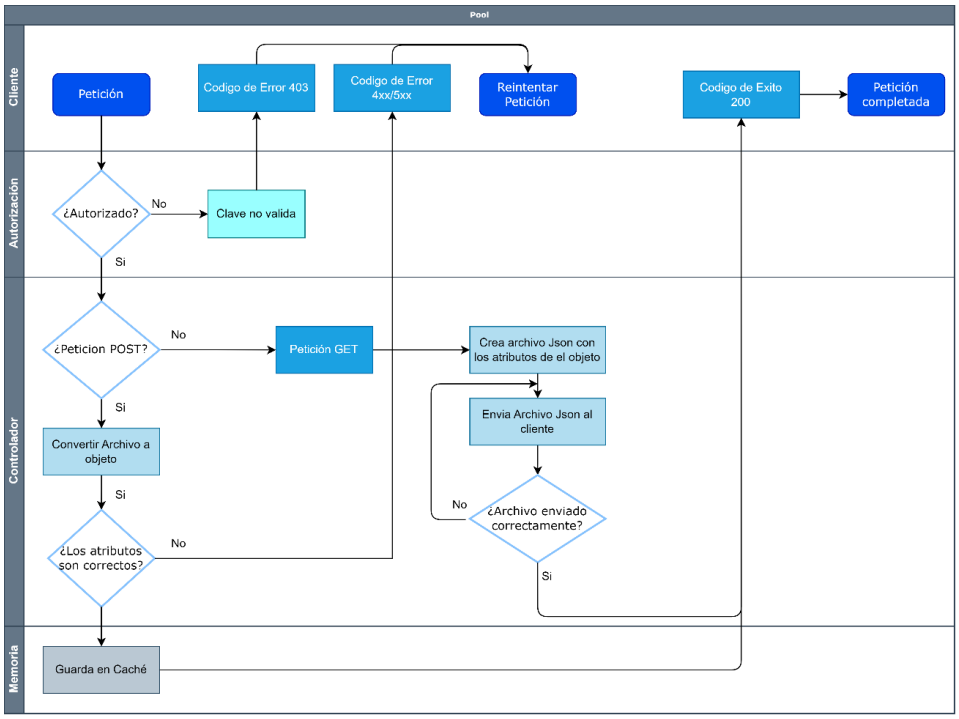
son códigos HTTP que indican el estado de la petición en conjunto.

Figura 5. Arquitectura del Servicio Web



Se puede observar que el controlador de la API (Figura 6), que es un Servlet, cuenta con dos métodos: uno para las solicitudes de tipo GET y otro para las solicitudes de tipo POST. Cuando se recibe una solicitud de tipo GET, se construye un archivo JSON con los valores del objeto que almacena la información de las coordenadas y los botones en memoria. A continuación, este archivo se envía al cliente, una aplicación móvil en este caso.

Figura 6 Diagrama de flujo con carriles de Servicio Web



En el método para las solicitudes POST se recibe un archivo JSON desde el sistema de seguridad que contiene las coordenadas y los datos de los botones. El controlador lee este archivo y descompone su información en las variables correspondientes, de manera que sus valores permanezcan almacenados hasta que sean solicitados para ser enviados al cliente o se actualice su información.

Módulo C) Aplicación móvil

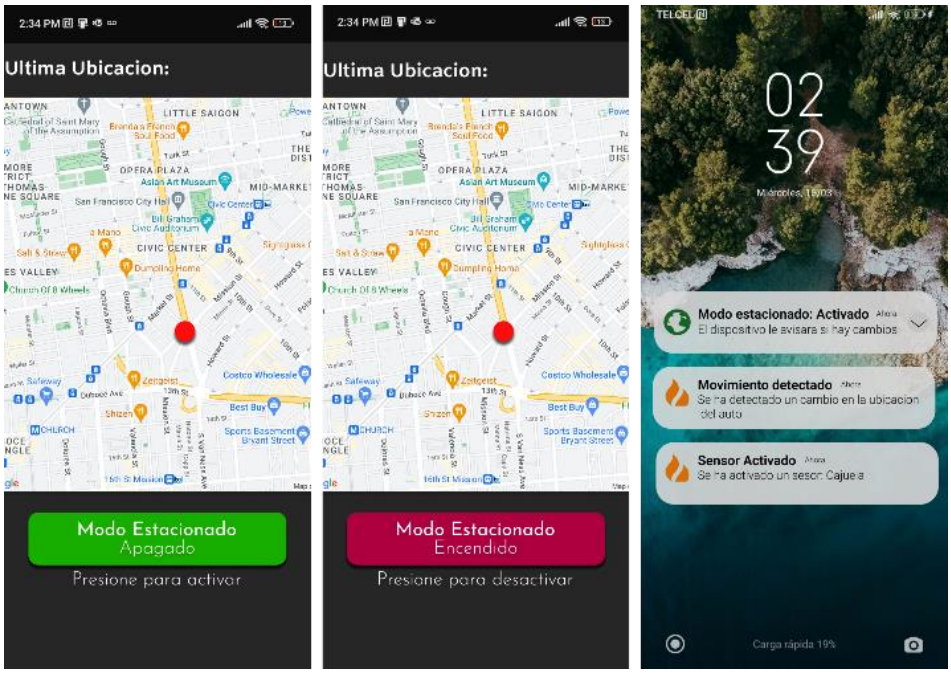
Diseño de la Interfaz de Usuario

Para el diseño de la interfaz de usuario se establecieron los siguientes requerimientos:

1. Integrar un mapa interactivo en el que se pueda apreciar claramente la ubicación del dispositivo de seguridad.
2. Implementar un botón para indicar al sistema que el inmóvil se ha dejado estacionado sin supervisión. Este botón activara el funcionamiento de toda la lógica de la aplicación e iniciara la adquisición de los datos del servidor.
3. Se deben generar notificaciones que indiquen si alguna violación de seguridad es activada.

En la Figura 7 se visualizan los mockup's de la Interfaz de Usuario que contiene los elementos para cumplir los requerimientos:

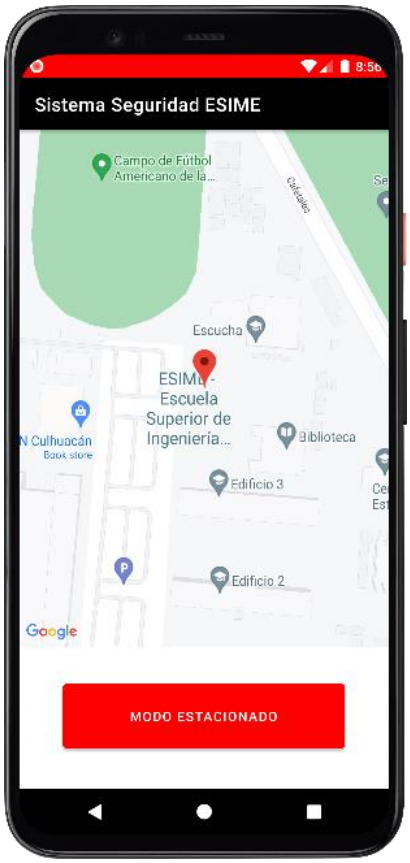
Figura 7. Mockups del diseño de la interfaz de usuario.



Desarrollo del backend

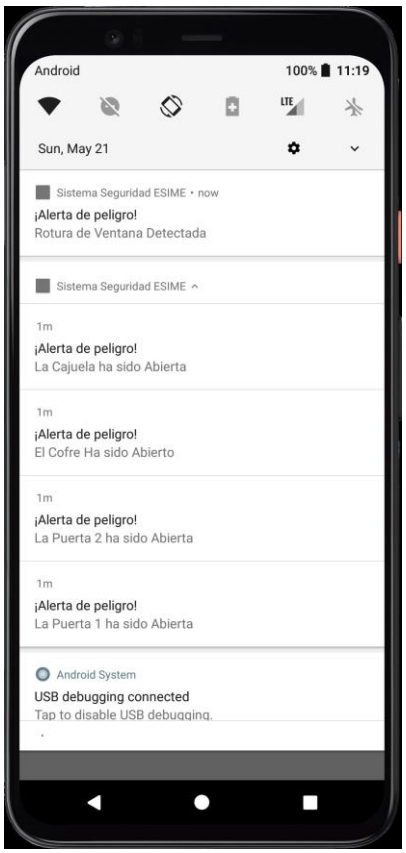
Para implementar el mapa que muestra la ubicación del dispositivo de seguridad, se utiliza la API de Google Maps. Esta API es ampliamente utilizada y ofrece un excelente funcionamiento para proyectos que requieran la inclusión de un mapa, así como una serie de herramientas que permiten una buena navegación. La API permite crear una representación gráfica del mapa donde se colocará un marcador que indica la última ubicación registrada. Para obtener los datos del servidor, la aplicación realiza una petición HTTP con el verbo GET hacia el endpoint del servicio web. Esta petición se realiza dentro de una clase que lee el archivo json devuelto por el servicio y asigna los valores obtenidos en las variables de un modelo de datos llamado “coordenadas”. Esta acción se realiza periódicamente como subproceso de la aplicación lo que permite tener una alta taza de actualización de los datos. Después se crea un marcador en el mapa con los datos obtenidos por el circuito de latitud y longitud. Este marcador indica la última localización obtenida por la aplicación como se ve en la Figura 8.

Figura 8. Marcador que indica la última ubicación obtenida por la aplicación.



Como se ve en la Figura 8, el sistema cuenta con un botón que sirve para indicar al sistema que el automóvil se ha dejado estacionado sin supervisión; esto activa el proceso de adquisición de datos y las notificaciones de la aplicación entonces, el usuario puede monitorear la ubicación del vehículo y obtener una alerta por medio de una notificación en su teléfono, si la seguridad del vehículo es violada (Figura 9). Cabe aclarar que si el botón es vuelto a presionar, la aplicación desactivara sus notificaciones y dejará de realizar peticiones al servidor y ya no se actualizarán los datos.

Figura 9. Notificaciones generadas por la aplicación cuando ocurre una violación de seguridad.



RESULTADOS Y DISCUSIÓN

Prueba de la API: Para probar el funcionamiento de la API, se utilizó la herramienta Postman, que es adecuada para realizar pruebas de peticiones HTTP a APIs. Se coloca la URL de la API junto con el verbo de la petición para confirmar que la API acepta la solicitud y devuelve una respuesta. Se realizaron pruebas con peticiones POST y con peticiones GET.

Peticiones POST: Para las pruebas POST, se ingresós el cuerpo del archivo JSON que sew deseaba enviar, después se realizó la petición a través de la URL de la API y se recibió la respuesta en Postman.

La respuesta incluye el código de estado HTTP, el tiempo de respuesta y el tamaño de los datos enviados; en este caso se obtuvo un código 200, lo que indica que los datos se recibieron correctamente y se obtuvo una respuesta en 26 ms. Se realizarons 50 peticiones y fueron exitosas (Figuras 10 y 11).

Figura 10. Petición tipo POST al servicio web mediante la herramienta Postman.

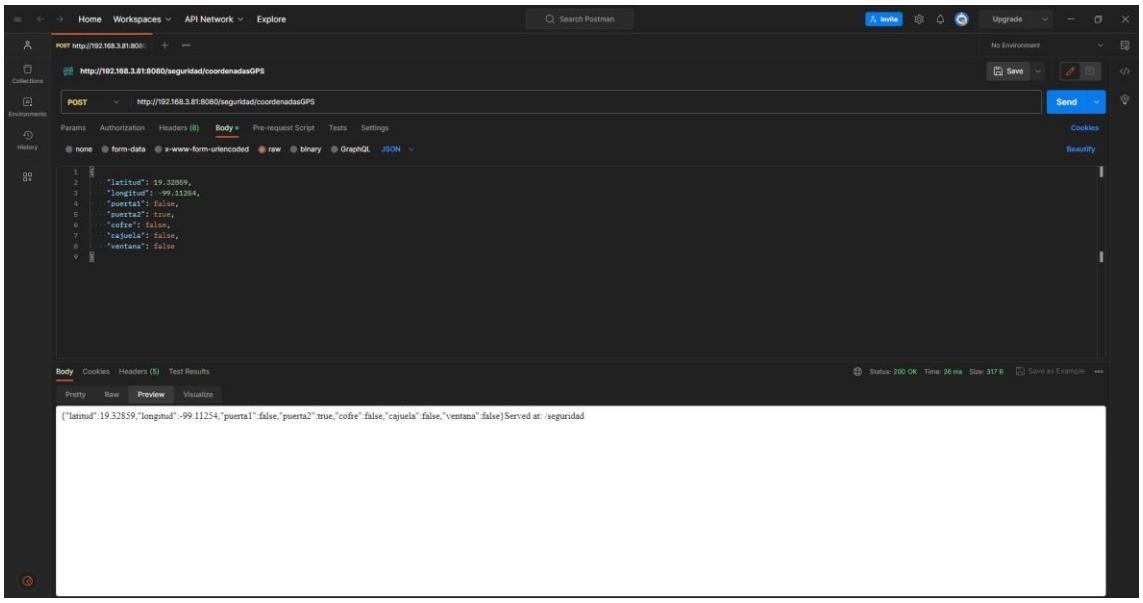


Figura 11. Peticiones POST y respuestas vistas desde la consola de Postman.

Peticiones GET: Para las pruebas GET se siguió el mismo procedimiento que con las peticiones POST; se observó que el cuerpo de la información obtenida como respuesta del servidor. En todas las peticiones, se recibieron los datos correctamente almacenados en el servidor (Figuras 12 y 13).

Figura 12. Petición tipo GET al servicio web mediante la herramienta Postman.

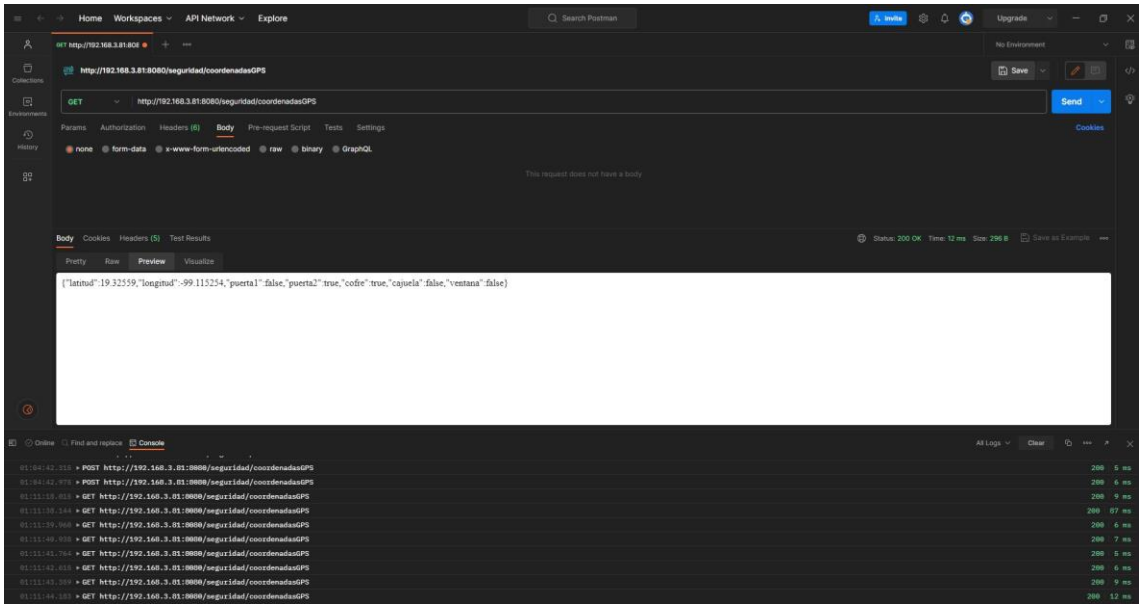


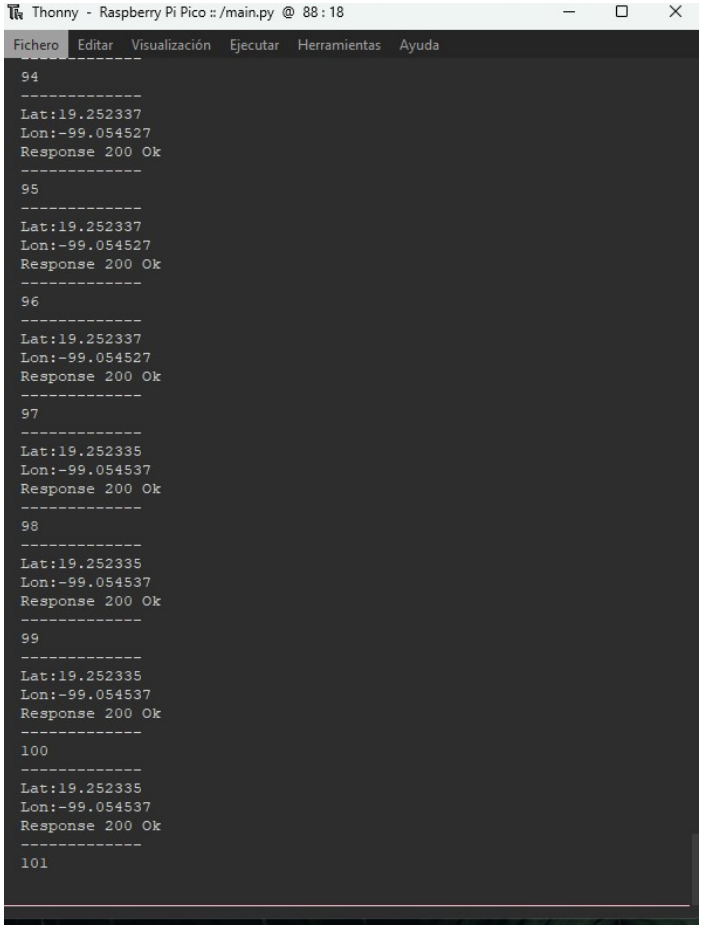
Figura 13. Peticiones GET y respuestas vistas desde la consola de Postman.

Pruebas del Sistema de Seguridad: Para probar el sistema de seguridad, se accedió a la página web de Google Maps para obtener las coordenadas del lugar donde se realizan las pruebas. Se utilizaron esas coordenadas como referencia para compararlas con las obtenidas por el dispositivo de ubicación. Se imprimió en la consola las coordenadas obtenidas a través del módulo GPS, junto con la respuesta del servidor sobre el estado de la petición POST realizada cada vez que se envían datos.

Muestras obtenidas en el Sistema de Seguridad: Se agregó un contador para registrar el número de iteraciones realizadas por el programa, llegando a 101 iteraciones en aproximadamente 50 segundos.

En todas las respuestas del servidor se obtuvo un código 200, lo que confirma que los datos fueron recibidos correctamente por la API. También se realizaron pruebas para verificar el funcionamiento de los interruptores que cambian el estado de las variables puerta1, puerta2, cofre, cajuela y ventana. Se utilizó Postman para realizar peticiones GET después de activar cada interruptor y confirmar que la información se envió correctamente al servidor. Se verificó que los archivos JSON que llegaron al servidor contenían la variable en estado "verdadero" correspondiente al interruptor presionado. Se repitió este ejercicio varias veces y en la mayoría de las ocasiones, el funcionamiento de la comunicación del sistema fue correcto al presionar los botones (Figura 14).

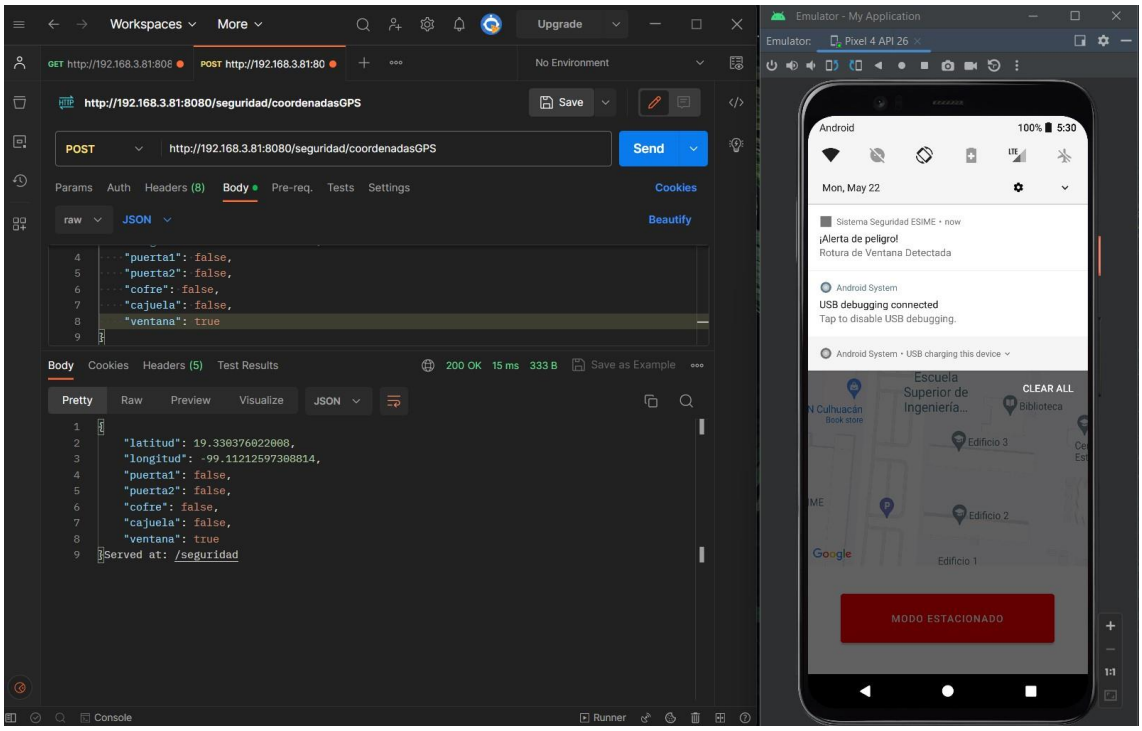
Figura 14. Impresiones de pantalla con los datos del circuito en la consola del IDE Thonny.



Pruebas del Cliente (Aplicación): Las pruebas del cliente se realizaron en el entorno de desarrollo integrado (IDE) Android Studio, utilizando el emulador de un dispositivo móvil Android modelo Pixel 4 con sistema operativo Android 8.0. Se inició la aplicación y se verificó que se iniciara correctamente; Después se activó la funcionalidad del sistema de seguridad con el botón "Modo Estacionado" para verificar la recepción de los datos almacenados en el servidor, lo cual se confirmó.

La aplicación recibió correctamente los datos de las coordenadas almacenadas y creó un nuevo marcador en el mapa para mostrarlas, obteniéndose exitosamente las coordenadas almacenadas en el servidor (Figura 15).

Figura 15. Adquisición correcta de los datos por parte de la aplicación.



CONCLUSIONES

Las conclusiones destacan tanto el éxito del sistema como las áreas potenciales de mejora, manteniendo el enfoque en la fiabilidad y efectividad del proyecto, en este sentido, los aportes son en la eficiencia del sistema de seguridad, confiabilidad de los componentes, impacto del uso de APIs y usabilidad de la aplicación móvil.

Eficiencia del sistema de seguridad: El sistema logró proporcionar una solución efectiva para el monitoreo y la protección de vehículos en línea, utilizando tecnologías accesibles y de bajo costo. La combinación de módulos GPS y GSM/GPRS permitió una comunicación estable con el servidor, lo que garantiza que los propietarios puedan monitorear la ubicación de su vehículo de manera confiable.

Confiabilidad de los componentes: A pesar de las limitaciones de precisión del módulo GPS y la sencillez del módulo GSM/GPRS, ambos demostraron ser suficientemente robustos para cumplir con

los requerimientos del sistema de seguridad. El sistema mostró una capacidad adecuada para detectar y alertar sobre posibles violaciones de seguridad.

Impacto del uso de APIs: La implementación de la API REST fue clave para la gestión de la información del sistema, demostrando que puede manejar eficientemente los datos recibidos del vehículo y enviarlos a la aplicación móvil sin redundancia ni pérdida de información.

Usabilidad de la aplicación móvil: La interfaz de usuario de la aplicación móvil facilitó el acceso a la información del vehículo y generó alertas de manera eficiente cuando se detectaba un intento de robo o movimiento no autorizado, cumpliendo con los objetivos del proyecto.

Si bien el sistema es completamente funcional, existen áreas donde podría mejorar, como la precisión del GPS y la implementación de más funciones en la aplicación móvil. Sin embargo, las funcionalidades básicas implementadas demostraron ser eficaces para su propósito principal, proporcionando tranquilidad y seguridad a los propietarios de vehículos en áreas metropolitanas.

REFERENCIAS BIBLIOGRAFICAS

Al-Rashed, M. A., Oumar, O. A., & Singh, D. (2014). A real-time GSM/GPS based tracking system. London South Bank University, London. https://www.ijariit.com

McCoy, C. (2017). A full featured GPS NMEA-0183 sentence parser for use with Micropython and the PyBoard embedded platform [Repositorio GitHub]. MIT License. https://github.com/inmcm/micropyGPS

Campbell, S. (2023). Basics of UART communication. Circuit Basics. Recuperado de [https://www.circuitbasics.com/basics-uart-communication/] (https://www.circuitbasics.com/basics-uart-communication/)

El Financiero. (2023, enero 20). Robo de autos en México: ¿Cuáles fueron los vehículos más robados en 2022? El Financiero. [https://www.elfinanciero.com.mx/nacional/2023/01/20/robo-de-autos-en-mexico-cuales-fueron-los-vehiculos-mas-robados-en-2022/] (https://www.elfinanciero.com.mx/nacional/2023/01/20/robo-de-autos-en-mexico-cuales-fueron-los-vehiculos-mas-robados-en-2022/)

Google. (2021). Java serialization/deserialization library to convert Java objects into JSON and back. GitHub. Recuperado de https://github.com/google/gson

Legaspi, M. G. (2023). UART: A hardware communication protocol. Analog Dialogue. Recuperado de [https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html] (https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html)

NASA. (2023a). How GPS works. Recuperado de https://spaceplace.nasa.gov/gps/en/

NASA. (2023b). The global positioning system. Recuperado de https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what-is-gps-k4.html

Pavithra, M., & Jyothi, S. (2018). Vehicle security using Raspberry Pi. International Journal of Advance Research, Ideas and Innovations in Technology, 4(3), 1127-1128. https://www.ijariit.com

Singh, P., & Sethi, T. (2020). Advanced vehicle security system using Raspberry Pi. National Institute of Technology Rourkela. IEEE Xplore. https://ieeexplore.ieee.org/document/9052246

T21. (2023, 6 de marzo). Primer bimestre de 2023 suma 237 pesados robados: ANERPV. ReANERPV. de [https://t21.com.mx/terrestre-2023-03-06-primer-bimestre-2023-suma-237-pesados-robados-] (https://t21.com.mx/terrestre-2023-03-06-primer-bimestre-2023-suma-237-pesados-robados-)

U-blox. (2011). Neo-6, u-blox 6, GPS modules, data sheet. Recuperado de [https://content.u-blox.com/sites/default/files/products/documents/NEO-6%20DataSheet.pdf] (https://content.u-blox.com/sites/default/files/products/documents/NEO-6%20DataSheet.pdf)