

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"SISTEMA DE ALERTA Y REPORTE ANTE ACCIDENTES DE CICLISTAS EN ZONAS DE ALTO TRÁFICO DE LA PROVINCIA DEL CARCHI."

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

Línea de investigación: Innovación Tecnológica y de productos

AUTOR:

Jordan Raúl Manguay Chacua

DIRECTOR:

Luis Edilberto Suárez Zambrano

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	Manguay Chacua Jordan Raúl

DATOS DE LA OBRA	
TÍTULO:	SISTEMA DE ALERTA Y REPORTE ANTE ACCIDENTES DE CICLISTAS EN ZONAS DE ALTO TRÁFICO DE LA PROVINCIA DEL CARCHI
AUTOR (ES):	MANGUAY CHACUA JORDAN RAUL
FECHA: DD/MM/AAAA	17/06/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN TELECOMUNICACIONES
DIRECTOR /ASESOR:	MSC. LUIS EDILBERTO SUÁREZ ZAMBRANO / MSC. JORGE RICARDO BENALCÁZAR GÓMEZ

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 17 días del mes de junio de 2026

EL AUTOR:



Manguay Chacua Jordan Raúl

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 17 de junio de 2026

MSC. SUÁREZ ZAMBRANO LUIS EDILBERTO

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.



(f) ……

Msc. Suárez Zambrano Luis Edilberto

C.C.: 1002304291

DEDICATORIA

Dedico este trabajo principalmente a Dios por darme la vida, salud y la sabiduría necesaria para culminar esta etapa de mi vida, a él por guiar siempre mis pasos y darme la fortaleza para no rendirme ante las adversidades de este proceso.

Dedico este triunfo a mi madre, cuyo amor incondicional y palabras de aliento que nunca me faltaron fueron mi refugio en momentos difíciles, a ella el ejemplo más grande de perseverancia y bondad que conozco, este título es tan tuyo como mío.

A mi padre, por ser mi mayor ejemplo de puntualidad, lucha, trabajo y rectitud, quien me ayudo a forjar mi carácter y enseñarme el verdadero valor de la disciplina, el que nunca se rinde a pesar de las adversidades gracias por ser mi guía, un gracias se queda corto, pero gracias por todo lo que has luchado y hecho por mí, este éxito es tu éxito.

A mis hermanas, por su compañía, sus risas, paciencia, palabras de ánimo, que fueron un impulso fundamental a lo largo de este camino. A mi gordo que a pesar de tenerme lejos nunca dejo de darme fuerza para seguir luchando, gracias por todo.

Finalmente dedico este triunfo a mi familia, quienes de una u otra forma, supieron brindar su apoyo moral y confiaron en mi desde el primer momento, no me queda más que decirle gracias familia por todos sus consejos y palabras que supieron darme la fuerza para llegar a cumplir este triunfo.

Jordan Manguay Chacua

AGRADECIMIENTO

En primer lugar, agradezco a mi familia, cuyo esfuerzo silencioso y constante fueron el verdadero motor a lo largo de la carrera, gracias por no dejar de creer en mis capacidades y darme el aliento, incluso en los momentos más difíciles, les dedico este triunfo porque cada paso que di en la universidad no sería fue protegido por su amor.

Asimismo, quiero agradecer de manera especial al MSc. Luis Suárez, tutor del presente Trabajo de Integración Curricular, por compartir su experiencia profesional, por su guía, su paciencia y valiosos aportes durante el desarrollo de este proyecto; así como al MSc. Jorge Benalcázar, asesor del trabajo, por sus observaciones, apoyo académico y orientación constante que permitió llegar a culminar una etapa más.

A mis amigos y compañeros, que durante esta travesía pude compartir momentos inolvidables que también hicieron que este camino sea más llevadero y enriquecedor.

Finalmente, expreso mi sincero agradecimiento a la a Universidad Técnica del Norte y a la Facultad de Ingeniería en Ciencias Aplicadas, por brindarme conocimientos y la formación académica en mi etapa profesional, mis docentes que gracias a ellos estoy formando para ser un buen profesional y a todas las personas que directa o indirectamente estuvieron presentes durante mi formación universitaria.

Jordan Manguay Chacua

RESUMEN

El presente trabajo de titulación tiene como objetivo el desarrollo e implementación de un sistema de alerta y reporte para ciclistas en rutas de alto tráfico, específicamente en la provincia del Carchi. Frente a la creciente vulnerabilidad física que enfrentan los ciclistas ante siniestros viales, los sistemas de seguridad tradicionales resultan insuficientes. Para ello, se implementó una arquitectura de hardware y software que sea capaz de detectar caídas e invasiones de carril a través de una plataforma que permita orquestar la adquisición, procesamiento y transmisión de los datos bajo exigencias estrictas de latencia mínima y el bajo consumo energético.

Mediante el uso del modelo Edge Computing, en donde se utiliza un microordenador Raspberry Pi Zero 2W como Gateway, el cual maneja una conexión inalámbrica WLAN para realizar un enlace con el microcontrolador ESP32-WROOM-32 encargado de recepción de datos inerciales y ultrasónicos localmente. Asimismo, la comunicación local se estableció de manera asíncrona mediante el protocolo MQTT gestionada a través del bróker HiveMQ. Para la comunicación con la nube, la conexión WAN fue desplegada a través del módulo LTE Quectel EG25-G.

Los resultados han demostrado que tiene una alta estabilidad de operabilidad en movimiento continuo. Al integrar el algoritmo Watchdog, se lograron erradicar las limitaciones de la red celular. Adicionalmente, el uso del formato JSON ayudo a reducir el consumo del ancho de banda, logrando despachar las alertas SMS hacia los familiares con una latencia menor a 10 segundos, incluso en zonas de baja cobertura. Por consiguiente, esto consolida una solución capaz de reducir los tiempos de rescate médico y salvaguardar la integridad física del ciclista.

Palabras clave: IoT; Edge Computing; Sistemas Embebidos; Protocolo MQTT; Conectividad LTE; Seguridad Vial**.**

ABSTRACT

This thesis aims to develop and implement an alert and reporting system for cyclists on high-traffic roads, specifically in the province of Carchi. Given the increasing physical vulnerability of cyclists in road accidents, traditional safety systems are insufficient. To address this, a hardware and software architecture was implemented capable of detecting falls and lane incursions through a platform that orchestrates the acquisition, processing, and transmission of data under strict requirements for minimal latency and low energy consumption.

Using Edge Computing, a Raspberry Pi Zero 2W microcomputer is used as a gateway, managing a WLAN wireless connection to link with the ESP32-WROOM-32 microcontroller, which is responsible for receiving inertial and ultrasonic data locally. Local communication is established asynchronously using the MQTT protocol, managed through the HiveMQ broker. For cloud communication, the WAN connection was deployed using the Quectel EG25-G LTE module.

The results have demonstrated high operational stability under continuous movement. By integrating the Watchdog algorithm, the limitations of the cellular network were eliminated. Additionally, the use of the JSON format helped reduce bandwidth consumption, enabling SMS alerts to be sent to family members with a latency of less than 10 seconds, even in areas with low coverage. Consequently, this consolidates a solution capable of reducing medical rescue times and safeguarding the cyclist's physical well-being.

Keywords: IoT; Edge Computing; Embedded Systems; MQTT Protocol; LTE Connectivity; Road Safety.

LISTA DE CONTENIDOS

1. Capítulo I: Antecedentes 22

1.1 Tema 22

1.2 Problema 22

1.3 Objetivos 26

1.3.1. Objetivo General 26

1.3.2. Objetivos Específicos 26

1.4 Alcance 26

1.5 Justificación 29

2. Capítulo II: Marco Teórico 32

2.1 Estado del arte 32

2.1.1. Sensor de Proximidad 34

2.1.2. Giroscopios 36

2.1.3. GPS 38

2.1.4. Comunicación 4G/LTE 43

2.1.5. Prevención de Accidentes 47

2.1.6. Sistemas de Monitoreo en tiempo real 50

3. Capítulo III: Diseño del Sistema 52

3.1 Enfoque metodológico 53

3.2 Situación actual 54

3.3 Requerimientos del sistema 55

3.3.1. Nomenclatura de requerimientos 55

3.3.2. Especificaciones del sistema 56

3.3.3. Estructura del sistema 57

3.4 Diseño del sistema de monitoreo 59

3.4.1. Selección del hardware 59

3.5 Selección del software 69

3.6 ETAPA DE DISEÑO 73

3.6.1. Arquitectura del sistema 73

3.6.2. Comunicación y protocolos 75

3.6.3. Diagrama de bloques y flujos de datos 77

3.7 ETAPA DE IMPLEMENTACIÓN 79

3.7.1. Desarrollo del dispositivo 79

3.7.2. Sistema de alimentación y autonomía 80

3.7.3. Integración de sensores 81

3.7.4. Módulos de comunicación inalámbrica 83

3.7.5. Implementación física del Sistema Principal 85

3.7.6. Distribución e instalación de sensores en la bicicleta 86

3.7.7. Ensamblaje y diseño del case 90

3.8 Procesamiento y almacenamiento de datos 93

3.8.1. Programación del microcontrolador 94

3.8.2. Conexión con la nube..... 97

3.8.3. Validación de los datos capturados 101

3.9 Desarrollo de la aplicación móvil 103

3.9.1. Estructura de la aplicación móvil..... 104

3.9.2. Funcionalidades 108

3.9.3. Estructura de Navegación por Roles..... 124

3.9.4. Comunicación con la nube..... 132

3.10 Servicios y gestión en la nube..... 140

3.10.1. Diseño de servicios para integración 140

3.10.2. APIs de comunicación y gestión de datos..... 143

4. Capítulo IV Resultados 145

4.1 Resultados del funcionamiento del nodo (ESP32)..... 146

4.1.1. Pruebas de adquisición de datos 147

4.1.2. Calibración de Sensores..... 150

4.1.3. Rendimiento del microcontrolador (ESP32)..... 155

4.1.4. Pruebas de transmisión inalámbrica..... 157

4.2 Resultados del funcionamiento del microordenador (Raspberry Pi Zero 2W)159

4.2.1. Despliegue de servicios locales 160

4.2.2. Rendimiento del microordenador 162

4.2.3. Pruebas de enlace LTE (Módulo EG25-G) 165

4.3 Resultados del sistema de alimentación y autonomía 167

4.3.1. Medición de consumo energético 168

4.3.2. Prueba de autonomía 170

4.4 Resultados de la comunicación con la nube 172

4.4.1. Envío y recepción de datos 173

4.4.2. Pruebas de APIs 175

4.5 Resultados de la aplicación móvil 177

4.5.1. Visualización de Telemetría y Geoposicionamiento en Tiempo Real 179

4.5.2. Funcionamiento del sistema de alertas 181

4.6 Validación del sistema 183

4.6.1. Pruebas funcionales 184

4.6.2. Desvanecimiento por Multitrayecto 186

4.6.3. Latencia por Traspaso de Celdas (Handover) en Ruta 188

4.6.4. Degradación de la Relación Señal a Interferencia más Ruido (SINR) 190

4.6.5. Pruebas de estabilidad 192

4.6.6. Análisis comparativo con los requerimientos 195

4.7 Análisis de resultados en escenarios reales 199

4.8 Discusión de resultados 204

4.9 Costo del sistema 205

Conclusiones 208

Recomendaciones 210

5. Bibliografía 211

6. Anexos 218

LISTA DE TABLAS

Tabla 1 *Nomenclatura de Requerimientos* 55

Tabla 2 *Requerimientos del sistema.* 56

Tabla 3 *Estructura del sistema.* 57

Tabla 4 *Selección del microordenador* 60

Tabla 5 *Selección placa de conectividad inalámbrica* 62

Tabla 6 *Selección de sensor de proximidad.* 64

Tabla 7 *Consumo energético del sistema* 67

Tabla 8 *Librerías utilizadas* 94

Tabla 9 *Diccionario de Datos del Backend (Supabase)* 97

Tabla 10 *Comparativa técnica* 101

Tabla 11 *Resumen de Parámetros de Validación Crítica.* 102

Tabla 12 *Mapeo de Funcionalidades y Componentes de Interfaz.* 107

Tabla 13 *Componentes Funcionales* 112

Tabla 14 *Comparativa de ruteo espacial* 119

Tabla 15 *Diccionario de Colores y Significados de Alerta.* 122

Tabla 16 *Matriz de Control de Acceso.* 126

Tabla 17 *Servicios de Nube Integrados en la Arquitectura SARC.* 142

Tabla 18 *Análisis de Precisión del Sensor de Proximidad* 147

Tabla 19 *Parámetros temporales de adquisición y transmisión en el nodo ESP32* 149

Tabla 20 *Comportamiento Térmico y Rendimiento del Microordenador* 163

Tabla 21 *Registro de temperaturas del microordenador durante operación continua* 164

Tabla 22 *Latencia promedio del sistema en diferentes zonas* 166

Tabla 23 *Análisis de Consumo Energético en Estado Activo (5V)* 170

Tabla 24 *Proyección de Autonomía del Sistema SARC (batería 10000 mAh)* 171

Tabla 25 *Matriz general de cumplimiento de requerimientos funcionales (EF)* 196

Tabla 26 *Matriz general de cumplimiento de requerimientos de arquitectura (EA)* 197

Tabla 27 *Costo General del dispositivo* 205

Tabla 28 *Resumen del costo general del sistema* 207

LISTA DE FIGURAS

Figura 1 *Árbol de problemas* 24

Figura 2 *Arquitectura genérica del dispositivo electrónico* 28

Figura 3 *Tipos de Sensores de Proximidad* 36

Figura 4 *Giroscopio* 38

Figura 5 *Trilateración GPS* 40

Figura 6 *Arquitectura LTE* 45

Figura 7 *IoT para prevención de accidentes.* 48

Figura 8 *Arquitectura sistema de monitoreo* 51

Figura 9 *Fases de la metodología modelo Waterfall* 53

Figura 10 *Raspberry Pi Zero 2W* 61

Figura 11 *Esp32 Wroom-32* 62

Figura 12 *Módulo LTE EG25-G* 63

Figura 13 *HC-SR04* 65

Figura 14 *GY-521 (MPU6050)* 65

Figura 15 *GPS NEO-6M* 66

Figura 16 *Raspberry Pi OS basado en Debian Linux* 70

Figura 17 *Arduino IDE* 70

Figura 18 *Base de datos Supabase* 71

Figura 19 *Visualización de datos Node-Red* 72

Figura 20 *Arquitectura del sistema* 74

Figura 21 *Topología lógica de red* 76

Figura 22 *Diagrama de bloques del sistema* 78

Figura 23 *Diagrama de flujo del sistema* 79

Figura 24 *Diagrama de alimentación del sistema* 80

Figura 25 *Diagrama de conexión de los sensores* 82

Figura 26 *Topología de red híbrida. Integración red local (WLAN) y celular (WAN).* 84

Figura 27 *Disposición e interconexión interna del sistema* 85

Figura 28 *Montaje final del Edge Gateway ensamblado* 86

Figura 29 *Detalle de la instalación del sensor ultrasónico* 86

Figura 30 *Montaje del módulo GPS, posicionado para garantizar una línea de vista (LOS)* 87

Figura 31 *Montaje del sensor (MPU6050) sobre el eje del manubrio* 88

Figura 32 *Vista panorámica de la bicicleta* 89

Figura 33 *Diseño y dimensiones de la carcasa principal para el sistema.* 91

Figura 34 *Diseño de la carcasa protectora para el módulo GPS NEO-6M.* 91

Figura 35 *Diseño y protección para el sensor ultrasónico HC-SR04* 92

Figura 36 *Diseño compacto para el sensor de inclinación y aceleración MPU6050.* 92

Figura 37 *Flujo Lógico en Node-RED* 94

Figura 38 *Algoritmo de Detección de Caídas* 95

Figura 39 *Captura de la terminal Linux (consola)* 97

Figura 40 *Diagrama de Node-RED.* 99

Figura 41 *DProtección Row Level Security (RLS) en Supabase* 99

Figura 42 *Diagrama lógico de validación* 102

Figura 43 *Estructura del Proyecto SARC* 104

Figura 44 *Implementación de Concurrencia MQTT* 106

Figura 45 *Lógica de Enrutamiento por Roles* 107

Figura 46 *Vista Principal (Dashboard) del SARC* 109

Figura 47 *Vista Principal, comportamiento a desconexión de red* 110

Figura 48 *Lógica de actualización asíncrona del sistema de telemetría.* 113

Figura 49 *Interfaz de Monitoreo Cartográfico* 114

Figura 50 *Implementación del Filtro Haversine en Python*.....116

Figura 51 *Trazado Dual de Trayectorias (Polylines)* 117

Figura 52 *Widget de Estado* 120

Figura 53 *Confirmación No Bloqueante (SnackBar)*..... 121

Figura 54 *Ejecución Asíncrona de Emergencia (Latencia Cero)* 122

Figura 55 *Topología de Red Desacoplada (Acceso Múltiple)* 123

Figura 56 *Comprobación de Operatividad en Red WAN*..... 124

Figura 57 *Modelo de Control de Acceso Basado en Roles (RBAC)* 125

Figura 58 *Interfaz principal del usuario Nivel 0 (Admin)* 127

Figura 59 *Interfaz principal del usuario Nivel 1 (Ciclista)* 128

Figura 60 *Interfaz principal del usuario Nivel 2 (Familiar)*..... 130

Figura 61 *Validación Segura de Credenciales en Capa de Datos* 131

Figura 62 *Diagrama de Jerarquía de Tópicos*..... 132

Figura 63 *Filtro de Deriva y Persistencia en Capa de Lógica* 133

Figura 64 *Estandarización Temporal (UTC-5) en el Backend* 134

Figura 65 *Decodificación y Des-serialización de Carga Útil*..... 134

Figura 66 *Delegación de Integridad Cronológica en PostgreSQL*..... 135

Figura 67 *Manejo de Excepciones y Prevención de Fugas de Memoria* 136

Figura 68 *Analítica de Riesgos y Puntos Calientes.* 137

Figura 69 *Interoperabilidad de Rescate (Deep Link)* 139

Figura 70 *Modelo de Orquestación y Respuesta ante Eventos.* 140

Figura 71 *Generación Dinámica de Voz (TwiML)* 141

Figura 72 *Aislamiento de Credenciales y Entorno Seguro* 143

Figura 73 *Inyección del Token* 143

Figura 74 *Conexión TCP/IP Directa a Base de Datos* 144

Figura 75 *Captura del monitor serial del ESP32-WROOM-32 durante su funcionamiento* 146

Figura 76 *Implementación del filtro de promedio para la señal ultrasónica.* 148

Figura 77 *Gestión de estados durante el arranque en frío del receptor NEO-6M.* 149

Figura 78 *Autocalibración por software para el cálculo de error del MPU6050* 151

Figura 79 *Validación física de inclinación del MPU6050 en consola local* 151

Figura 80 *Algoritmo de filtrado y conversión de señal del sensor HC-SR04* 152

Figura 81 *Gestión de alertas ante detección de objetos en zona de riesgo* 153

Figura 82 *Comparación de la ubicación GPS del módulo NEO-6M con Google Maps.* 154

Figura 83 *Arquitectura de doble núcleo del microcontrolador ESP32-WROOM-32* 156

Figura 84 *Recepción de mensajes MQTT en la Raspberry Pi con marcas de tiempo (ms).* 156

Figura 85 *Monitoreo del tráfico MQTT en tiempo real desde la Raspberry Pi* 158

Figura 86 *Reconexión automática del ESP32 al broker MQTT tras pérdida de red.* 159

Figura 87 *Comprobación de estado del servicio Eclipse Mosquitto* 160

Figura 89 *Diagrama de flujo en Node-Red, para peticiones HTTP* 161

Figura 90 *Configuración del servicio systemd para la gestión de reinicio automático* . 162

Figura 90 *Medición de temperatura del microprocesador bajo operación continua* 163

Figura 91 *Latencia de red LTE hacia servidores en la nube mediante protocolo ICMP* 166

Figura 92 *Ejecución de monitoreo persistente y recuperación autónoma del enlace LTE.* 167

Figura 93 *Medición de voltaje estabilizado mediante multímetro digital.* 168

Figura 94 *Arquitectura de la estrategia de respaldo y mitigación.* 172

Figura 95 *Implementación del flujo de orquestación de datos en Node-RED.* 173

Figura 96 *Validación del tiempo de respuesta del sistema de alertas vía SMS.* 174

Figura 97 *Confirmación de peticiones HTTP/REST exitosas.* 175

Figura 98 *Lógica de discriminación de alertas para optimización de recursos* 176

Figura 99 *Auditoría de persistencia de datos en el gestor de base de datos Supabase* .. 177

Figura 100 *Arquitectura lógica y flujo de navegación de la aplicación móvil SARC.* ... 178

Figura 101 *Arquitectura lógica y flujo de navegación de la aplicación móvil SARC.* ... 179

Figura 102 *Monitor de estado de conexión del cliente Paho-MQTT.* 181

Figura 103 *Reactividad visual de la interfaz ante cambios de estado de emergencia* 182

Figura 104 *Arquitectura de procesamiento asíncrono para despachar de alertas críticas* 182

Figura 105 *Activación de la alerta y registro de ejecución asíncrona* 184

Figura 106 *Lógica de filtrado de proximidad y control de umbrales de seguridad* 185

Figura 107 *Panel de configuración y gestión remota de actuadores* 186

Figura 108 *Esquema de interferencia por multitrayecto en señales LTE y GPS.* 187

Figura 109 *Implementación de hilo asíncrono en Python* 188

Figura 110 *Diagrama de secuencia de mitigación SOS durante el Handover celular.* .. 189

Figura 111 *Diagrama de flujo lógico Watchdog para la nueva asociación celular.*..... 191

Figura 112 *Curva de temperatura del microprocesador durante operación continua*... 192

Figura 113 *Registro de la consola durante una prueba de estrés de concurrencia*..... 193

Figura 114 *Restricción temporal de llamadas de voz* 194

Figura 115 *Análisis comparativo de alertas emitidas en función de los kilómetros recorridos.* 200

Figura 116 *Clasificación de alertas de proximidad vehicular según su nivel de severidad y peligro.* 201

Figura 117 *Trazabilidad histórica del recorrido generada por el módulo GPS durante la fase de campo.* 202

Figura 118 *Consolidado de alerta y detecciones emitidas por el sistema en entorno real.* 203

Capítulo I: Antecedentes

1.1 Tema

Sistema de alerta y reporte ante accidentes de ciclistas en zonas de alto tráfico de la provincia del Carchi.

1.2 **Problema**

En la provincia del Carchi, muchas personas usan la bicicleta no solo para transporte sino también para entrenamiento deportivo. Sin embargo, esto se ha vuelto cada vez más riesgoso, especialmente en zonas de mucho tráfico, rutas interprovinciales o carreteras no suficientemente seguras como la ruta E -35 en el tramo comprendido desde el Puente de Rumichaca al sector Obelisco dentro de la ciudad de Tulcán. Según datos del Instituto Nacional de Estadística y Censos (2024), en el Ecuador ocurrieron aproximadamente 10,500 accidentes de tránsito durante el año 2023, de los cuales un 12% de las víctimas fueron ciclistas. Según información de La Hora (2024), personal de la Policía Nacional reportó que en la provincia del Carchi se suscitaron alrededor de 30 accidentes con ciclistas involucrados durante el año 2023, con un saldo fatal de 5 fallecidos y 18 heridos graves, donde el factor común en este tipo de acontecimientos se atribuye principalmente a la falta de infraestructura, añadiendo la deficiente señalización vial que se mezcla con la alta velocidad de los vehículos motorizados y la convivencia directa entre ciclistas y automotores en vías sin carriles exclusivos.

Dentro de estos hechos, se puede el mencionar aquel suscitado al joven ciclista Kevin Pozo, quién perdió la vida tras ser atropellado mientras entrenaba en la Panamericana Norte, sector de Piquiucho (La Hora, 2024). Así mismo otro hecho ocurrido fue el de Aníbal Benavides, ciclista máster, atropellado por un camión en el sector de El Capote (Carchi) ocurrido el 28 de octubre de 2022; casos sumados a otros de igual envergadura, han despertado profunda preocupación en la

población, debido que demuestran la vulnerabilidad de los ciclistas en carreteras donde conductores circulan a altas velocidades sin contar con infraestructura adecuada ni sistemas de respuesta inmediata.

Informes de la Agencia Nacional de Tránsito indican que el 25% de los accidentes viales en provincias, como Carchi, Imbabura y Pichincha, están relacionados con ciclistas y que la mayoría de estos accidentes ocurren en vías sin carriles exclusivos ni indicaciones adecuadas (Chamba & Pesántez, 2023); dada esta situación cada vez es mayor el reto para autoridades a la hora de buscar soluciones integrales para reducir la elevada incidencia y mejorar el tráfico rodado.

El GAD Provincial del Carchi y la ANT, independientemente, han impulsado campañas de educación y seguridad vial, tales como la iniciativa Juntos Salvando Vidas, sin embargo, los resultados, según reporta La Hora (2024), aún son limitados; reconociendo que dichas campañas ayudan a concienciar, pero no solucionan uno de los problemas más críticos: la falta de un sistema que detecte y avise automáticamente cuando se produce el accidente, permitiendo reaccionar rápidamente desde los servicios de asistencia. En muchos casos, cuando un ciclista sufre un accidente en el campo, los testigos cercanos podrían informar de inmediato lo sucedido, en cmabio, si la persona afectada está inconsciente o alejada de zonas pobladas, los retrasos en la atención pueden ser mortales; por ende, la ausencia de un sistema tecnológico que automatice seriamente las alertas y los informes limita la capacidad de respuesta de las autoridades y pone en peligro la vida de los usuarios.

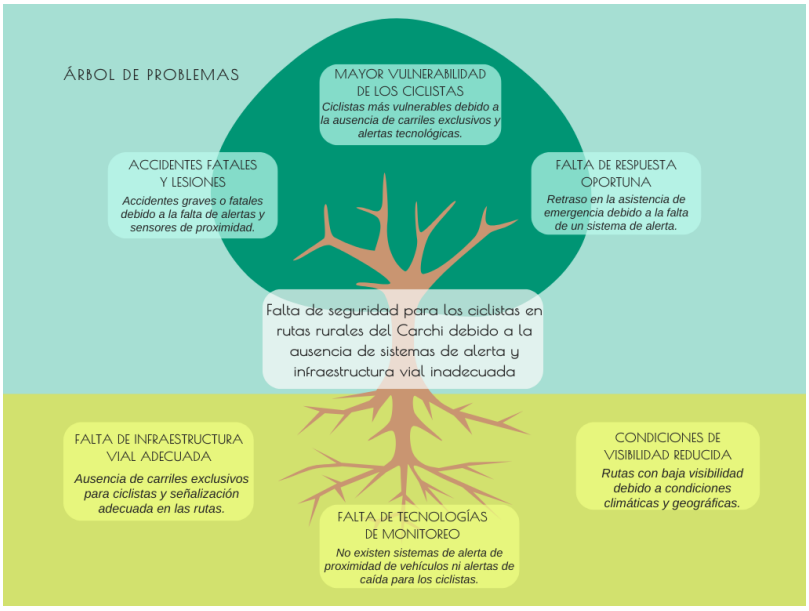
Trabajos de investigación previos han mostrado que la implementación de dispositivos con sensores y sistemas de comunicación puede reducir estos tiempos de respuestas; por ejemplo, (Moscoso & Romero, 2020) desarrollaron un prototipo de radiobaliza para ciclistas que detecta impactos y envía alertas automáticas mediante tecnología LoRaWAN, lo que demuestra que es

viable aplicar soluciones similares en contextos locales; sin embargo, hasta el momento no existe en la provincia del Carchi un sistema de alerta y reporte adaptado a su realidad geográfica y tecnológica, donde gran parte del territorio carece de cobertura constante de red celular o internet. Lo expresado se evidencia más aún al considerar que, según datos del Instituto Nacional de Estadística y Censos, la tasa de siniestralidad en Ecuador alcanzó 12,81 fallecimientos por cada 100.000 habitantes, y los ciclistas están entre los grupos más vulnerables (INEC, 2024). Estas cifras reflejan que los esfuerzos actuales en materia de seguridad vial no son suficientes para garantizar una atención oportuna y eficiente ante los accidentes.

Es así como, los factores mencionados (infraestructura, condiciones geográficas, falta sensores y sistemas de alertas temprana) provocan efectos negativos de gran relevancia, como el incremento en la cantidad y gravedad de accidentes, el aumento de lesiones y muertes, la baja capacidad de respuesta ante emergencias, la sensación generalizada de inseguridad, y el impacto negativo en la salud pública y calidad de vida de la población ciclista. La Figura 1 ilustra cómo la problemática central, definida como la alta inseguridad vial para ciclistas, se origina en diversas causas agrupadas en tres categorías principales: infraestructura vial inadecuada, falta de sistemas tecnológicos de monitoreo y alerta, y factores humanos y sociales.

Figura 1

Árbol de problemas



Nota. Se presentan las causas y consecuencias de la inseguridad vial para ciclistas.

Para abordar las causas identificadas en el árbol de problemas y mitigar los efectos negativos sobre la seguridad vial de los ciclistas en la ruta E -35 en el tramo comprendido desde el Puente de Rumichaca al sector Obelisco, en la ciudad de Tulcán se propone el desarrollo e implementación de un sistema de alerta y reporte, el mismo integrará sensores que permitan detectar la cercanía de vehículos motorizados y las caídas de ciclistas, generando alertas. Parar la comunicación se empleará tecnología 4G/LTE, garantizando de esta manera la transmisión en tiempo real de las alertas a los ciclistas y a sus contactos de emergencia con el fin de facilitar una respuesta rápida y oportuna (Caín, 2021).

Es por toda la problemática actual presentada, que el sistema propuesto más allá de un avance tecnológico representa una solución innovadora y viable que busca aportar a la seguridad, la prevención y la tranquilidad de la comunidad ciclista de la provincia del Carchi. De esta forma, se trata de unificar la innovación con la empatía, desarrollando una herramienta que salve vidas y promueva una movilidad más humana y responsable.

1.3 Objetivos

1.3.1. *Objetivo General*

Desarrollar un sistema de alerta y reporte de accidentes de ciclistas en rutas de alto tráfico de la provincia del Carchi, utilizando tecnología de comunicación 4G/LTE, mejorando la respuesta ante emergencias y fortalecer la seguridad vial.

1.3.2. *Objetivos Específicos*

- Establecer el estado del arte sobre tecnologías de monitoreo y comunicación inalámbrica aplicadas a sistemas de alerta para ciclistas, revisando sensores de proximidad, GPS, giroscopios y comunicación 4G/LTE en proyectos similares tanto a nivel nacional como internacional.
- Determinar los requerimientos técnicos y recursos necesarios para implementar un sistema de alerta para ciclistas en las rutas de alto tráfico de la provincia del Carchi, para garantizar la eficacia del sistema
- Diseñar el sistema de alerta de colisión que integre sensores para detectar la proximidad de vehículos y las caídas del ciclista, y utilizar comunicación 4G/LTE para transmitir alertas en tiempo real a los familiares.
- Realizar pruebas de funcionamiento del sistema, midiendo la precisión de las alertas y evaluando el impacto en la seguridad vial para los ciclistas en condiciones reales de tráfico.

1.4 Alcance

El trabajo a realizar se enfoca en el diseño, desarrollo, implementación y evaluación de un sistema autónomo de alerta para ciclistas que tiene como fin mejorar la seguridad vial y prevenir accidentes en zonas de alto tráfico vehicular; es así como, el sistema que integrará sensores de proximidad, GPS, giroscopios enlazados mediante tecnología de comunicación 4G/LTE que

permita monitorear en tiempo real la actividad del ciclista y al mismo tiempo se envíe alertas tanto al usuario como a sus contactos de emergencia, sobre posibles obstáculos y/o accidentes suscitados (Jabbar et al., 2020).

El tramo comprendido desde el Puente de Rumichaca al sector Obelisco con una longitud aproximada 13 km sobre la ruta E35 de la ciudad de Tulcán, capital provincial del Carchi, se convertirá en el escenario idóneo debido a su alta circulación vehicular, uso frecuente por ciclistas amateurs y que cuenta con condiciones propias de la región que permitirá validar la efectividad del sistema en un entorno real.

Para alcanzar los objetivos planteados, primero se realizará una revisión exhaustiva del estado del arte sobre tecnologías de monitoreo y comunicación inalámbrica aplicadas a sistemas de alerta para ciclistas, considerando proyectos similares a nivel nacional e internacional. Luego con los conocimientos adquiridos, se determinarán los requerimientos técnicos, así como recursos necesarios para la implementación del sistema en la ruta seleccionada.

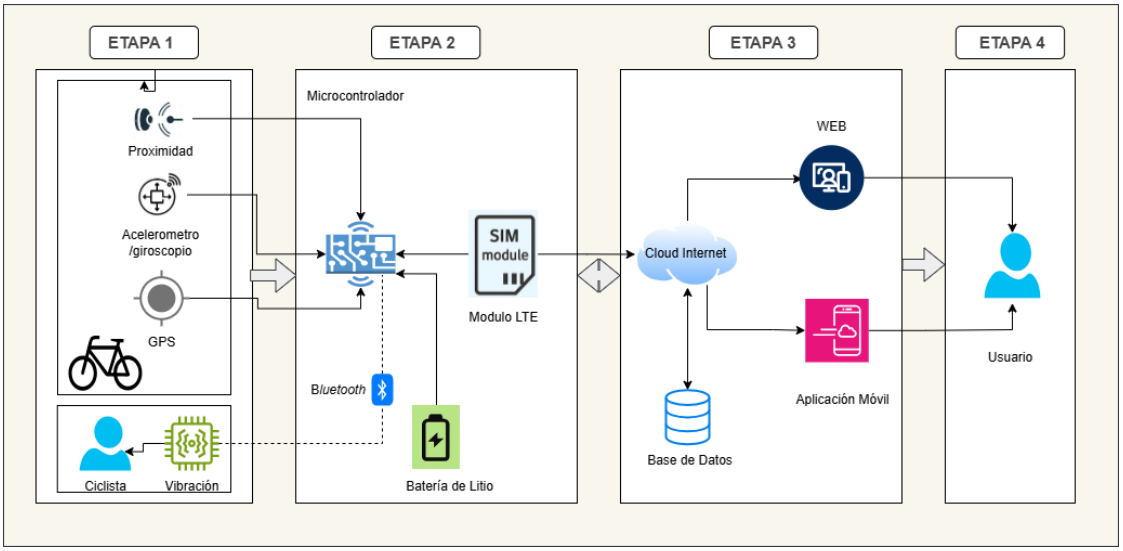
La siguiente etapa, constará del diseño del sistema, el mismo que presentará la integración de sensores de proximidad y comunicación 4G/LTE. A priori, se realizarán pruebas de campo en condiciones reales de tráfico para medir la precisión de las alertas y evaluar el impacto en la seguridad vial, asegurando así la efectividad del sistema.

Para la presente investigación se empleará una metodología en cascada ya que su característica principal viene definida con un enfoque secuencial en el desarrollo del sistema; es así que en la fase de análisis de requisitos, se definirá el alcance del sistema autónomo de alerta para ciclistas y se establecerá la ruta E -35 (tramo desde el Puente de Rumichaca al sector Obelisco) como área de estudio; luego en la fase de diseño, se especificarán los componentes del sistema, como los sensores, el microcontrolador y el módulo de comunicación, además del diseño de la

arquitectura de comunicación con la plataforma en la nube; durante la fase de implementación, se procederá al ensamblaje del prototipo, integrando los sensores y configurando el sistema de transmisión de datos en tiempo real, por último, como ay se había mencionado durante las pruebas verificará el correcto funcionamiento del sistema en condiciones reales; finalmente, en la fase de mantenimiento, se generará la documentación técnica y se propondrán recomendaciones para futuras mejoras del sistema. Esta metodología permite un desarrollo estructurado y ordenado, asegurando que cada etapa se complete de forma clara antes de avanzar a la siguiente (Vaca, 2020).

Figura 2

Arquitectura genérica del dispositivo electrónico



Nota: Se muestra un sistema que integra sensores con un microcontrolador y comunicación 4G/LTE

La arquitectura del sistema presentado en la Figura 2 se encuentra compuesto de cuatro etapas principales; en la etapa 1, se lleva a cabo la recolección de datos mediante sensores de proximidad, acelerómetro/giroscopio y GPS, además de un sensor de vibración incorporado al cuerpo del ciclista. Estos dispositivos permiten detectar condiciones críticas como caídas o la cercanía de vehículos en movimiento. Luego, los datos obtenidos, son procesados en la etapa 2 mediante un microcontrolador el mismo que determina si se debe emitir una alerta o no. En caso

de ser un evento positivo, los datos se transmiten a través de un módulo (chip) LTE/4G, que es el encargado de realizar el envío inmediato de dicha información a la nube mediante la red celular. Es así como en la etapa 3, el sistema gestiona los datos recibidos en la nube, almacenándolos en una base de datos y canalizándolos hacia plataformas de visualización como una aplicación móvil y una interfaz web. Por último, durante la etapa 4 el usuario interactúa con la interfaz, pudiendo acceder a la información en tiempo real, visualizar alertas y tomar decisiones a fin de contribuir a la mejora de la seguridad vial del ciclista.

1.5 Justificación

La seguridad de los ciclistas se ha convertido en una preocupación creciente en la provincia del Carchi, región reconocida por su cultura deportiva y su aporte al ciclismo ecuatoriano; Sin embargo, detrás de esta pasión también se esconde una realidad alarmante: los accidentes de tránsito que involucran a ciclistas siguen aumentando, dejando víctimas que en muchos casos no reciben atención inmediata por la falta de mecanismos de alerta eficientes, esto pone en evidencia la urgente necesidad de implementar soluciones tecnológicas que mejoren la seguridad vial. En zonas de alto tráfico como Tulcán, Huaca o el eje vial Panamericano, los ciclistas comparten la vía con transporte pesado y automóviles que en su mayoría no respetan las leyes de tránsito y por ende las distancias seguras, es por ello que, según datos de la ANT, en el Ecuador se registran más de 20.000 siniestros de tránsito al año, ya sea debido a la falta de infraestructura, ausencia de sistemas de alerta temprana y/o condiciones geográficas complejas, hacen que los ciclistas sean especialmente vulnerables (Sánchez, 2020).

Desde un punto de vista tecnológico, este proyecto contribuye directamente al cumplimiento de los Objetivos de Desarrollo Sostenible (ODS) #9, #3 y #6: Industria, Innovación e Infraestructura, que busca fomentar la innovación tecnológica y promover infraestructuras

resilientes, sostenibles y accesibles. Mediante el desarrollo de un sistema de alerta y reporte basado en sensores y comunicación 4G/LTE, se impulsa la aplicación de soluciones tecnológicas en entornos urbanos; Además de que se propone una solución viable, de bajo costo y adaptable al contexto local. Por ello, esta iniciativa representa un avance en la digitalización de la seguridad vial, y al mismo tiempo fortalece la conectividad y la inclusión tecnológica en zonas con alta vulnerabilidad, debido que, al integrar dispositivos embebidos, transmisión de datos en tiempo real y plataformas en la nube, se promueve una cultura de innovación orientada al bienestar social, alineada con los principios del desarrollo sostenible (ONU, 2015).

Según menciona UNIR (2025), LTE es un estándar de comunicación inalámbrica que permite la transmisión confiable y rápida de datos (hasta 100 Mbps de bajada y 30 Mbps de subida), lo cual la convierte en la tecnología idónea para el envío de alertas en tiempo real y enlazar el sistema con plataformas en la nube para monitoreo. La selección de la ruta E-35 (Puente de Rumichaca-Sector Obelisco) con una distancia aproximada de 13 km, para la implementación del sistema, no fue esporádica, dado que responde a la disponibilidad y estabilidad de la cobertura 4G/LTE, esto debido a que contar con constante cobertura asegura la operatividad del sistema, facilitando la comunicación instantánea entre el ciclista y sus contactos de emergencia; y garantiza así la efectividad y factibilidad técnica del sistema.

En el ámbito académico, el desarrollar sistemas de esta índole, representa una oportunidad para integrar conocimientos de electrónica, telecomunicaciones e informática aplicada a la prevención de accidentes; además de que contribuye al avance de la ingeniería incentivando la innovación tecnológica con propósito social.

Por todo lo mencionado, es que precisamente el desarrollo de un sistema de monitoreo y alerta que utilice sensores de proximidad, giroscopios y comunicación 4G/LTE se presenta como

una alternativa innovadora para reducir accidentes, ofreciendo alertas en tiempo real tanto a los ciclistas como a sus familiares; y al mismo tiempo busca ser un aporte tangible a la provincia del Carchi, uniendo la tecnología con la empatía humana, haciendo que la implementación del sistema a escala permita que cada pedalazo sea más seguro y que los ciclistas, símbolos del esfuerzo y orgullo carchense, puedan desplazarse con mayor confianza, sabiendo que, ante cualquier eventualidad, habrá un sistema que los cuide y alerte a tiempo.

Capítulo II: Marco Teórico

2.1 Estado del arte

Para comenzar con este estudio es importante destacar diferentes trabajos que aportan al proyecto propuesto, con enfoque en la aplicación de tecnologías para comunicación en tiempo real.

En el ámbito nacional, el trabajo realizado por Moscoso & Romero (2020), es uno de los pioneros donde se desarrolló un prototipo de radiobaliza para ciclistas capaz de detectar impactos y enviar señales de emergencia mediante tecnologías de bajo consumo como LoRa; A pesar de que su enfoque no utilizó redes 4G, representa un aporte importante en la integración de sensores de impacto y módulos de comunicación inalámbrica en el contexto ecuatoriano, además de evidenciar la viabilidad de sistemas portátiles aplicados a la seguridad vial.

Otro trabajo se denomina: Diseño e implementación de un sistema inteligente basado en GPS para la mejora del servicio de transporte público urbano en la ciudad de Ibarra (Reyes & Quishpe, 2017); En el trabajo se propone integrar una Raspberry Pi como CPU, un módulo GPS para la geolocalización, una pantalla LED para visualización, y un SO con programación en Python, obteniendo una arquitectura tecnológica permite el anuncio automático de paradas y la localización en tiempo real de los autobuses, optimizando la experiencia del usuario y facilitando la movilidad urbana, así mismo su enfoque demuestra cómo el uso de dispositivos embebidos, sensores y software libre puede ofrecer soluciones viables y de bajo costo para modernizar los sistemas de transporte urbano en ciudades intermedias.

El sistema de alerta para ciclistas propuesto para la presente investigación, comparte similitudes con el desarrollado por (Reyes & Quishpe, 2017), especialmente en el uso de GPS y plataformas de procesamiento como microcontroladores o miniPCs; sin embargo, el enfoque del trabajo en curso se adapta a un contexto diferente que es la seguridad vial en rutas de alto tráfico

del Carchi, con el cual se quiere demostrar que la adaptación de tecnologías GPS y comunicación móvil a un entorno rural permite aprovechar la experiencia previa en transporte urbano, pero orientada a los ciclistas, promoviendo su seguridad mediante soluciones tecnológicas innovadoras.

La propuesta de red LTE desarrollada por (Álvarez, 2014) sirve como base técnica para comprender la importancia de la cobertura y estabilidad de la tecnología 4G/LTE, la cual es fundamental para el funcionamiento del sistema de alerta para ciclistas, dentro de este proyecto, la conectividad LTE garantiza que las alertas generadas por sensores de proximidad o giroscopios sean transmitidas en tiempo real; en contraste del caso urbano en Quito, aquí se aprovecha la cobertura ya disponible en el Carchi para conectar un sistema embebido con plataformas en la nube, asegurando que las notificaciones lleguen rápidamente a familiares o servicios de emergencia; de esto modo, se evidencia cómo los principios técnicos del diseño de redes LTE pueden ser adaptados a soluciones referentes a la seguridad vial y en diversos contextos geográficos.

Con estos antecedentes, ya varios repositorios universitarios del Ecuador han documentado proyectos de detección de caídas y envío de alertas vía GSM o GPS, donde se demuestra la preocupación por brindar protección a ciclistas y motociclistas mediante soluciones accesibles y replicables en contextos tanto rurales como urbanos.

A nivel internacional, se pueden citar a Schnee et al. (2021), quienes propusieron un método probabilístico para la clasificación en línea de choques de bicicletas utilizando datos de acelerómetros y giroscopios montados en el cuadro, lo que permitió detectar caídas con un alto nivel de fiabilidad; en cambio, el realizado por De Smith et al. (2024), probablemente asentó la base para sistemas preventivos de alerta, al integrar redes neuronales CNN-LSTM utilizadas en la predicción de maniobras de giro a partir de señales captadas por IMU, con lo cual logran anticipar

movimientos bruscos así como potenciales pérdidas de control; Por su parte el estudio de Hadi et al. (2021) presenta un sistema de detección de accidentes para ciclistas mediante fusión GPS-IMU y filtros de Kalman, reduciendo los falsos positivos en entornos urbanos complejos; es por estos antecedentes que en la plataforma ResearchGate, se publicaron prototipos de bajo costo que emplean módulos GSM y GPS para enviar mensajes automáticos a contactos de emergencia o centros de monitoreo, lo cual se asemeja al objetivo de fortalecer la respuesta ante siniestros en zonas de tráfico elevado.

Otro concepto de relevancia en este estudio es la comunicación inalámbrica, por ende, es menester hablar sobre los proyectos Vehicle-to-Bicycle (V2B) hechos en Australia por Imove (2021) demostraron la utilidad de la tecnología 4G/LTE para conectar ciclistas con vehículos cercanos, reduciendo el riesgo de colisiones en cruces o avenidas de alta densidad vehicular, es así, como estos ensayos validan la eficiencia de las redes móviles en tiempo real y la posibilidad de integrarlas con sistemas embebidos o smartphones.

También cabe mencionar, el proyecto ***Integrated Smart Bike Safety System*** propuesto por Azzez et al. (2024) donde incorporan sensores de proximidad, módulos GPS y comunicación LTE para generar alertas automáticas ante caídas o proximidad con vehículos; este tipo de propuestas demuestran la convergencia de varias tecnologías que tu proyecto busca adaptar al contexto ecuatoriano, donde las condiciones geográficas, la cobertura móvil y la infraestructura vial representan desafíos particulares.

2.1.1. *Sensor de Proximidad*

Según Breda et al. (2025) un sensor de proximidad: es un dispositivo que detecta la presencia de objetos o personas cercanas sin necesidad de contacto físico; normalmente, estos sensores emiten un campo (eléctrico, magnético, acústico u óptico) o una señal (ultrasonido,

infrarrojo, láser, radio) y miden alteraciones en ese campo o el retorno de la señal para estimar la distancia al objetivo; dentro del presente estudio, estos sensores van a permitir detección vehículos, obstáculos o situaciones de riesgo que requieren alerta inmediata al ciclista o al sistema de gestión. Por lo expresado, es que existen diferentes tipologías de sensores de proximidad que podrían ser empleados para ciclistas como:

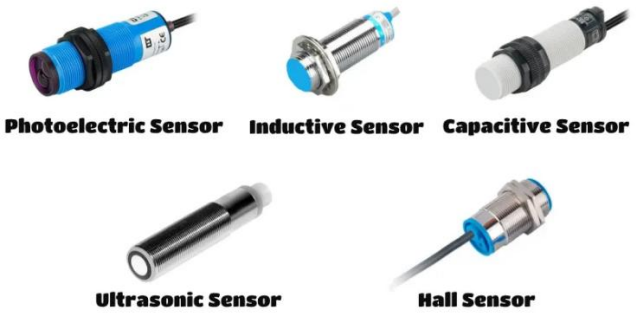
- **Ultrasonido:** emite ondas acústicas de alta frecuencia, mide el eco para estimar distancia. Resulta económicamente barato, por ende, puede verse afectado por condiciones de lluvia, viento o superficies irregulares (Reddit, 2020).
- **Infrarrojo (ToF):** emite luz infrarroja (láser), mediante la cual se mide el tiempo que tarda el reflejo en volver; Tiene buen rendimiento en dispositivos pequeños y puede ofrecer buena resolución para detectar objetos próximos. El trabajo ProxiCycle emplea sensores ToF para medir adelantamientos cercanos en ciclistas (Breda et al., 2025).
- **Inductivos:** Se utilizan para detectar objetos metálicos a través de un campo electromagnético; son ideales para entornos húmedos o sucios, puesto que tanto los líquidos como la suciedad no interfieren con su desempeño (Eaton, 2025).
- **Capacitivos:** Pueden detectar tanto objetos metálicos como no metálicos, incluyendo líquidos y materiales en polvo; utiliza principios dieléctricos de capacitancia para establecer un campo de detección cerca de la cara del sensor que crea una zona de detección, causando que el sensor se encienda cuando un objetivo cerca del sensor altera ese campo y alcanza el punto de operación; por contrario, el sensor se apaga cuando el objetivo se aleja del sensor y alcanza el punto de liberación (Weg, 2024)
- **Magnéticos:** Detectan un campo magnético, como el de un imán en un actuador neumático, para confirmar la posición del émbolo (Festo, 2024).

- **Radar/microondas:** emite ondas de radio/microondas que penetran mejor en condiciones adversas (lluvia, polvo) y pueden medir velocidad relativa además de distancia; no se emplean frecuentemente en prototipos de bajo coste para bicicletas todavía.
- **Bluetooth/BLE/etiquetas/beacons:** menos medición de distancia exacta y más presencia; Puede usarse para detección de cercanía entre vehículos y ciclistas (vehículo detecta etiqueta del ciclista) o zonas peligrosas (CycleAlert, 2023).

En cuanto a las ventajas de los sensores de proximidad es que se destacan por su alta fiabilidad, bajo consumo energético, tamaño compacto y facilidad de integración con microcontroladores y sistemas embebidos; Sin embargo, presentan desafíos asociados a la interferencia ambiental, el alcance limitado en algunos modelos y la necesidad de calibración precisa para evitar falsas detecciones (Ahmed et al., 2019).

Figura 3

Tipos de Sensores de Proximidad



Nota. Autoría propia.

2.1.2. *Giroscopios*

Según la definición de Woodman (2007), un giroscopio es un sensor que permite medir y mantener la orientación angular de un objeto en el espacio, se basa en el principio de conservación del momento angular, en otras palabras, este dispositivo detecta cambios en la velocidad de rotación alrededor de uno o más ejes, convirtiéndolo en un componente básico de observación o

componente de navegación; los giroscopios de manera tradicional, se basaban en la conservación del momento angular mediante discos o rotores giratorios; sin embargo, los giroscopios utilizados en los deportes como automovilismo son electrónicos y se basan principalmente en el efecto Coriolis, dicho efecto, se presenta cuando un cuerpo en movimiento giratorio experimenta un cambio en la dirección de rotación crea una fuerza aparente perpendicular al movimiento; por su parte, los giroscopios electrónicos detectan cambios y los convierten en señales eléctricas proporcionales a la velocidad angular (IAD, 2024). La característica sobresaliente de un giroscopio es que puede medir la rotación en los tres ejes tridimensionales:

- Eje X (horizontal): Movimiento de cabeceo o pitch que detecta las inclinaciones longitudinales.
- Eje Y (Lateral): Movimiento de balanceo o roll, detecta inclinación lateral del vehículo.
- Eje Z (Vertical): Movimiento de guiñada o yaw, observa los giros hacia la izquierda o derecha.

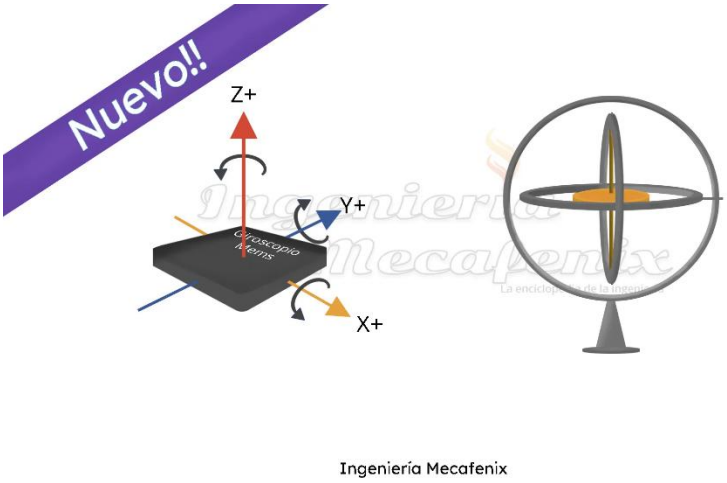
Hoy en día, los giroscopios se integran principalmente en plataformas electrónicas miniaturizadas conocidas como sistemas microelectromecánicos (MEMS), que les permiten integrarse en teléfonos inteligentes, drones, vehículos y dispositivos de seguridad para ciclistas.

Cuando se trata de sistemas de señalización para ciclistas, los giroscopios juegan un papel vital a la hora de detectar eventos críticos como caídas, giros bruscos o desequilibrios; Su integración en dispositivos portátiles o montados en bicicletas permite reconocer patrones de movimiento anormales que pueden estar relacionados con accidentes, enviando alertas automáticas a una aplicación o centro de emergencia utilizando tecnologías de comunicación como 4G/LTE o Bluetooth (Wang et al., 2022). De esta manera, los giroscopios no sólo contribuyen a la seguridad

de los usuarios, sino que también facilitan la recopilación de datos de comportamiento y dinámica de conducción útiles para la investigación de la movilidad y la prevención de accidentes.

Figura 4

Giroscopio



Nota. Autoría propia.

Por su parte, los giroscopios MEMS tienen la característica principal de ser de bajo consumo de energía, tamaño pequeño y alta sensibilidad, lo que los vuelve perfectos y necesarios para sistemas portátiles y de tiempo real; sin embargo, presentan desafíos técnicos como el ruido electrónico, la deriva angular y la necesidad de una calibración constante para mantener la precisión de las mediciones (Titterton et al., 2005). Para superar estas limitaciones, se suelen utilizar técnicas de fusión de sensores (uso combinado de GPS, acelerómetro y giroscopio) que ofrecen mejora tanto en la confiabilidad como estabilidad de los datos.

2.1.3. *GPS*

El Sistema de Posicionamiento Global (GPS), originalmente conocido como Navstar GPS, hace referencia al sistema que permite a un dispositivo receptor determinar su ubicación en la Tierra con precisión cm o m. El sistema fue desarrollado, implementado y operado por el Departamento de Defensa de Estados Unidos, y actualmente es propiedad de la Fuerza Espacial

de EE. UU. El usuario utiliza cuatro o más satélites y trilateración para determinar su ubicación. Si se desea determinar una posición tridimensional, el receptor utilizado para ello busca automáticamente al menos cuatro satélites en la red, de los que recibe señales que indican la identificación y hora de cada uno de ellos, así como información sobre la constelación de satélites incluidas en el sistema. A partir de estas señales, el dispositivo sincroniza su reloj con la hora del sistema GPS y calcula el tiempo que tardan las señales en llegar al equipo, midiendo así la distancia al satélite. Utilizando el método de trilatación inversa, calcula su posición. También se calcula con alta precisión en el tiempo basándose en relojes atómicos de cada satélite y GPS de segmento terrestre; su integración con tecnologías móviles e Internet de las Cosas (IoT) ha abierto un abanico de posibilidades para mejorar la movilidad sostenible y la seguridad de los ciclistas, uno de los grupos más vulnerables en el tránsito urbano (Kaplan & Hegarty, 2017). El sistema de satélites se conforma de una constelación de 24 satélites centrados en la Tierra; aunque, sin embargo, solo se requieren de tres satélites para determinar una ubicación en la superficie terrestre, a menudo se utiliza un cuarto satélite para validar la información de los otros tres, donde el cuarto satélite permite calcular la altitud de un dispositivo.

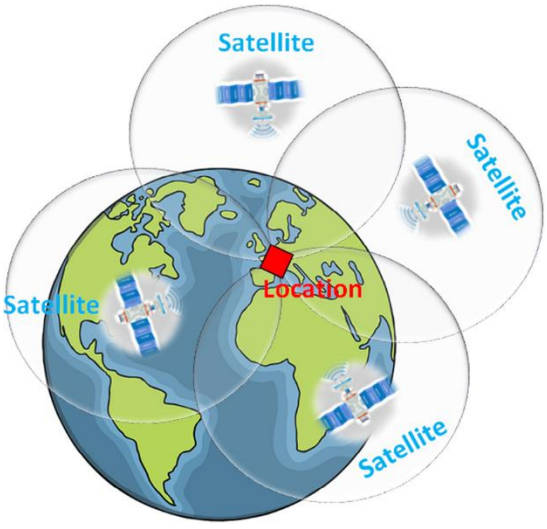
Como ya se mencionó el GPS funciona mediante la técnica llamada trilateración, que sirve para calcular la ubicación, la velocidad y la elevación, a continuación, se resume un poco de su funcionamiento:

Los satélites que orbitan la Tierra envían señales para que sean leídas e interpretadas por un dispositivo GPS cercano o dentro de la Tierra; luego, cada uno de los 4 satélites necesarios para calcular la ubicación realiza dos órbitas a la Tierra en 24 horas y envía una señal (parámetros orbitales e información horaria únicos); es por esto, por lo que el uso de un solo satélite proporciona solamente información acerca de la distancia desde un satélite, por ende, no puede proporcionar

información mejor detallada sobre la ubicación ya que, cuando un satélite envía una señal se crea un círculo con un radio medido desde el dispositivo GPS al satélite; sin embargo, al agregar un segundo satélite también se crea un segundo círculo y la ubicación se reduce a uno de los dos puntos donde los círculos intersecan; ya con la intervención del tercer satélite, la ubicación del dispositivo se puede determinar definitivamente, debido a que el dispositivo (a encontrar) está en la intersección de los tres círculos, por lo que si un dispositivo se mueve, el radio cambiará; así, cuando sucede esto, se generan nuevas esferas, lo que arrojará una nueva posición, pudiendo utilizar esos datos en conjunto con la hora del satélite, para determinar la velocidad y calcular la distancia a nuestro destino como también la hora de llegada estimada (Geotab, 2025).

Figura 5

Trilateración GPS



Nota. Autoría propia.

El GPS es una herramienta poderosa y confiable para negocios y empresas de muchas industrias diferentes. Topógrafos, científicos, pilotos, capitanes de barcos, socorristas y trabajadores de la minería y la agricultura son algunas de las personas que utilizan el GPS en su trabajo diario. Utilizan información de GPS para preparar planos y mapas precisos, realizar

mediciones de tiempo precisas, rastrear la posición o ubicación y para la navegación. El GPS funciona de forma continua y en casi todas las condiciones climáticas. Hay cinco usos principales del GPS:

- Ubicación: Posicionamiento.
- Ubicación: determinar una posición.
- Navegación: moverse de una ubicación a otra.
- Seguimiento: supervisión del movimiento de objetos o personas.
- Cartografía: creación de mapas del mundo.
- Temporización: permite realizar mediciones precisas del tiempo.

A continuación, se detallan los casos donde más se emplean los GPS:

- **Respuesta de emergencia:** si en algún momento se presenta algún tipo de emergencia o un desastre natural, los servicios de emergencia utilizan GPS para mapear, rastrear y pronosticar el clima y la ubicación de los servicios de emergencia. En la UE y Rusia, la regulación de llamadas de emergencia eCall se basa en la tecnología GLONASS (alternativa al GPS) y la telemática para enviar datos a los servicios de emergencia en caso de accidente de vehículo, reduciendo el tiempo de respuesta.
- **Entretenimiento:** El GPS se puede integrar en juegos y actividades como Pokémon Go y Geocaching (Posicionweb, 22d. C.).
- **Salud y estado físico:** Los relojes inteligentes y la tecnología portátil se pueden utilizar para realizar un seguimiento de la actividad física (como carreras de distancia) y compararla con un grupo demográfico similar.
- **Construcción, minería y transporte por carretera:** localiza equipos, mide y mejora la asignación de activos, permitiendo a las empresas aumentar el rendimiento de sus activos.

- **Transporte:** con el fin de mejorar la productividad y garantizar la seguridad de los conductores, diferentes compañías de logística están implementando sistemas telemáticos; así la geolocalización de camiones se está utilizando para mejorar la optimización de rutas, la eficiencia del combustible, la seguridad del conductor y el cumplimiento normativo.

La precisión de un dispositivo GPS depende de muchas variables, incluidas, entre otras, la cantidad de satélites disponibles, la ionosfera y el entorno urbano.

Algunos de los factores que pueden afectar la precisión del GPS incluyen:

- **Obstáculos físicos:** las mediciones del tiempo pueden alterarse por grandes obstrucciones como montañas, edificios, árboles entre otras.
- **Efectos atmosféricos:** los dispositivos GPS pueden verse afectados por las tormentas eléctricas severas y las tormentas solares.
- **Efemérides:** a pesar de que esto es cada vez menos común, el modelo de órbita interna de un satélite puede ser incorrecto o estar desactualizado.
- **Cálculos numéricos incorrectos:** este aspecto puede verse afectado si el hardware del dispositivo no está diseñado según las especificaciones.
- **Interferencia artificial:** esto incluye dispositivos de interferencia o suplantación de GPS.

La precisión para un GPS realmente es mayor en áreas abiertas sin edificios altos cercanos que bloqueen las señales, este efecto se conoce como cañón urbano. Por ejemplo, en el centro de Manhattan o Toronto, una señal de satélite primero se bloquea, luego rebota en un edificio y finalmente es captada por un dispositivo; esto puede ser la causal para obtener cálculos erróneos de la distancia al satélite. Para fortuna nuestra, los diferentes problemas críticos identificados que afectan a la tecnología GPS ya están a punto de resolverse, teniendo como evidencia que los

receptores de alta calidad proporcionan una precisión superior a 2,2(m) el 95 % del tiempo y superior a 3(m) el 99 % del tiempo.

A pesar de que el GPS proporciona una ubicación geográfica precisa, no siempre puede determinar el contexto del movimiento. Por ello, los sistemas modernos de alerta de accidentes integran el GPS con sensores acelerómetros, giroscopios y sensores de proximidad; el acelerómetro permite detectar cambios bruscos en la aceleración lineal, característicos de una caída; por su parte, el giroscopio mide la rotación o inclinación del dispositivo, útil para determinar la pérdida de equilibrio; y de igual manera, el sensor de proximidad cumple con su principal función de reconocer la cercanía de objetos o vehículos, anticipando riesgos potenciales. La fusión de estos datos, conocida como **sensor fusion**, ofrece un modelo más robusto para determinar cuándo ocurre un accidente real y evitar falsas alarmas (Teunissen & Montenbruck, 2017).

2.1.4. *Comunicación 4G/LTE*

A lo largo del tiempo, el sector de las telecomunicaciones ha experimentado una continua evolución fruto de la necesidad de la gente de mantenerse conectada a cualquier hora, lugar y momento; Dada dicha necesidad, es que desde la primera generación 1G (basada en señales analógicas), hasta las redes digitales 2G y 3G, el desarrollo tecnológico ha buscado mejorar la calidad, velocidad y cobertura de los servicios móviles; En este contexto, 4G o Cuarta Generación, y en específico la tecnología LTE (Long Term Evolution), representa un salto significativo hacia las comunicaciones móviles de alta velocidad y baja latencia, orientadas a satisfacer las crecientes demandas de transmisión de datos, aplicaciones multimedia e Internet de las Cosas (IoT) (Sauter, 2014).

La adopción generalizada de 3G/acceso a paquetes de alta velocidad (HSPA) ha provocado un enorme aumento en el consumo de datos móviles. Estos tres factores clave (capacidad,

experiencia del usuario y menor costo por bit) han llevado a la especificación de Desarrollo a Largo Plazo (LTE) del Acceso Universal Terrestre por Radio (UTRAN). La principal especificación de un sistema móvil se define como System Architecture Evolution (SAE), también llamado Enhanced Packet Core (EPC). Junto con EPC, LTE forma el Sistema de Paquetes Evolucionado (EPS). EPS consta de UTRAN evolucionada (E-UTRAN), Núcleo de paquetes evolucionado (EPC) y conectividad con sistemas de acceso heredados del Proyecto de asociación de tercera generación (3GPP) y sistemas de acceso que no son 3GPP. Este capítulo presenta el Sistema de Acceso 3GPP EPS, que fue creado como una evolución de la arquitectura 2G/3G existente y la arquitectura de Calidad de Servicio (QOS) para el servicio portador LTE. También muestra las categorías de equipos de usuario (UE) de LTE y sus respectivas velocidades de datos máximas teóricas basadas en las opciones de modulación (Kukushkin, 2018).

4G o Cuarta Generación hace referencia al conjunto de estándares que definen las características de las redes móviles de su categorización; mientras que LTE es la tecnología de acceso radioeléctrico más ampliamente adoptada dentro de esta categoría; El LTE fue desarrollado por el 3rd Generation Partnership Project (3GPP) con el objetivo de ofrecer un servicio de banda ancha móvil con velocidades superiores a las de las redes 3G (UMTS/HSPA), alcanzando tasas de hasta 100 Mbps en uplink y 50 Mbps en downlonk dependiendo de la configuración y el ancho de banda disponible (Dahlman et al., 2018).

A diferencia de las generaciones anteriores, LTE se basa exclusivamente en conmutación de paquetes y utiliza la tecnología IP (Internet Protocol) para la transmisión de voz, datos y video; Esto elimina la dependencia de la conmutación de circuitos y permite integrar múltiples servicios en una red unificada, reduciendo costos y mejorando la eficiencia (Holma & Toskala, 2011).

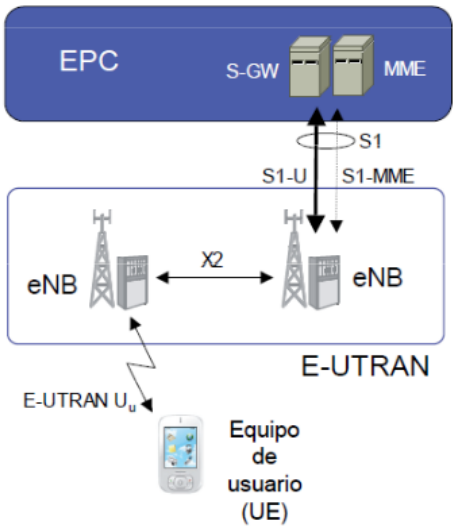
Arquitectura del Sistema LTE

Este sistema se compone de tres elementos principales que interactúan para proporcionar conectividad continua y eficiente:

- Evolved Universal Terrestrial Radio Access Network (E-UTRAN): aquí se encuentran las estaciones base o eNodeB, las mismas que realizan la transmisión y recepción de señales de radio, la gestión de la movilidad del usuario y la asignación dinámica de recursos del espectro.
- Evolved Packet Core (EPC): su papel como núcleo de la red es manejar la autenticación, el enrutamiento del tráfico IP y la conexión con redes externas (Internet o redes privadas); Está formado por entidades como el Mobility Management Entity (MME) y el Serving/Packet Gateway (S-GW/P-GW).
- User Equipment (UE): corresponde al dispositivo final o de usuario, como un smartphone, tablet, módulo IoT o sensor conectado.

Figura 6

Arquitectura LTE



Nota. Autoría propia.

LTE utiliza modulación OFDM, Multiplexación por División de Frecuencia Ortogonal, para dividir el canal de transmisión en múltiples subportadoras ortogonales, para de esta manera optimizar la eficiencia espectral y la resistencia a interferencias; Además, el empleo de MIMO (Multiple Input Multiple Output) mejora la capacidad y la velocidad mediante el uso simultáneo de múltiples antenas en transmisión y recepción (Rappaport et al., 2013).

Características técnicas del 4G/LTE

Dada a su continua evolución y gran proliferación, LTE posee un conjunto de características que lo convierten en la base de las comunicaciones móviles modernas, entre ellas:

- Alta velocidad de transmisión (tx) de datos: 100 Mbps en downstream y 50 Mbps en upstream.
- Baja latencia: ideales para aplicaciones interactivas o en tiempo real gracias a sus tiempos de respuesta promedio de 10 ms.
- Alta eficiencia espectral: gracias al uso de OFDMA en enlace descendente y SC-FDMA en ascendente.
- Calidad de servicio o QoS: el sistema LTE da prioridad al tráfico acore a las necesidades del usuario (voz, video, datos críticos).
- Escalabilidad y compatibilidad: soporta diferentes anchos de banda (1.4 MHz a 20 MHz) y puede coexistir con tecnologías antiguas como 3G y/o WiFi.

Por todas estas características expresadas, LTE se ha convertido en una tecnología central para aplicaciones que requieren una transmisión de datos continua, entre los que se incluyen sistemas de alerta en tiempo real, vehículos conectados o redes de sensores IoT.

Usos y relevancia en el IoT

Con el despliegue masivo de redes 4G/LTE han surgido múltiples soluciones tecnológicas, como por ejemplo en ámbito de la seguridad vial, LTE facilita la comunicación en tiempo real entre dispositivos móviles y servidores en la nube, permitiendo enviar alertas instantáneas en caso de accidentes o emergencias; Así por ejemplo, sistemas basados en LTE pueden transmitir la ubicación obtenida por GPS junto con datos de sensores (como acelerómetros o giroscopios) hacia plataformas que gestionan alertas automáticas o reportes de accidentes (Brambilla et al., 2024).

De igual manera, LTE es una base sólida del Internet de las Cosas, al ofrecer conectividad confiable y cobertura amplia para dispositivos distribuidos en entornos urbanos; en particular, su variante LTE-M (Machine Type Communication) y NB-IoT (Narrowband IoT) están optimizadas para bajo consumo energético y transmisión de pequeños volúmenes de datos, ideales para sensores de monitoreo ambiental, rastreo vehicular o sistemas inteligentes de transporte (ITS) (Sauter, 2014).

Limitaciones en la transición a 5G

Más allá de sus ventajas, LTE presenta limitaciones frente a las crecientes demandas de conectividad, siendo principalmente su capacidad de conexión simultánea por celda y la latencia mínima alcanzable insuficientes para soportar algunas aplicaciones emergentes de ultra baja latencia o gran densidad de dispositivos, como los vehículos autónomos o la realidad aumentada; Por ello, la evolución natural del 4G hacia LTE-Advanced y posteriormente hacia 5G NR (New Radio) busca superar estas limitaciones con mayores anchos de banda, latencias menores a 1 ms y mayor eficiencia energética (Dahlman et al., 2018).

2.1.5. *Prevención de Accidentes*

En la sociedad moderna, el prevenir accidentes viales y laborales asociados al uso de vehículos, maquinaria y entornos urbanos congestionados ha tomado especial relevancia. La

seguridad casi todo el tiempo, tiene su foco en medidas reactivas mediante la intervención una vez ocurrido el accidente; Sin embargo, el avance de la tecnología ha permitido desarrollar sistemas de prevención proactiva, capaces de anticipar, detectar y responder a situaciones de riesgo en tiempo real (Shriharshan S et al., 2025).

Figura 7

IoT para prevención de accidentes.



Nota. Tomado de LinkedIn.

El empleo de IoT brinda una salida a este inconveniente, gracias a que su principio fundamental de la prevención radica en el monitoreo continuo; es así que, los sensores integrados en los dispositivos recogen información sobre variables relevantes (velocidad, aceleración, inclinación, temperatura, proximidad y frecuencia cardíaca, entre otras); estos datos son transmitidos a una unidad de procesamiento o a un servidor remoto, donde algoritmos de análisis determinan si existe un patrón que indique una posible situación de riesgo; en caso de detectarse una anomalía, el sistema genera una alerta inmediata, ya sea visual, sonora o digital, notificando al usuario o a los servicios de emergencia (Diego & Morocho, 2025).

Otra funcionalidad de los sistemas basados en IoT es que también permiten incorporar análisis predictivos mediante la recopilación de datos históricos; de este modo, mediante la

comparación de patrones, estos sistemas pueden anticipar zonas o momentos de alto riesgo, contribuyendo a la planificación urbana, la gestión del tránsito y la reducción de accidentes recurrentes; por ejemplo, plataformas de movilidad inteligente utilizan información obtenida por sensores y dispositivos IoT para identificar puntos críticos en las vías y emitir alertas preventiva a los usuarios o autoridades competentes (Jabbar et al., 2020)

La ventaja de Iot sobre la interoperabilidad con tecnologías complementarias como son el Big Data, la IA y los GPS, fortalece su capacidad de análisis y respuesta; dicha ventaja permite que la prevención de accidentes evolucione desde simples alertas automáticas hasta sistemas de asistencia integral, capaces de coordinar la respuesta entre múltiples actores, como servicios médicos, policías de tránsito o familiares del afectado (Hadi et al., 2021)

Por otro lado, la prevención basada en IoT no se limita al ámbito vial, sino que puede abordar sectores como la industria minera, la construcción, el transporte público o los sistemas inteligentes de monitoreo, en donde su empleo ha reducido significativamente los incidentes laborales y mejorado la seguridad del personal; esto se puede corroborar a través de la información compartida por OIT (2023), donde expresa que el uso de dispositivos portátiles y sensores IoT en entornos laborales ha permitido disminuir los accidentes en un 25%, gracias a la detección temprana de condiciones inseguras y la automatización de alertas.

La prevención de accidentes basada en IoT es un avance trascendental en la gestión de la seguridad, de modo que al conectar el entorno físico con plataformas inteligentes, pasamos de la seguridad reactiva a la seguridad preventiva y predictiva, donde la tecnología no sólo detecta los peligros, sino que aprende de ellos y actúa con antelación; en el ámbito de la movilidad, estas innovaciones ofrecen soluciones efectivas para proteger a los usuarios más vulnerables y fortalecer los esfuerzos para crear ciudades más seguras, sostenibles e inteligentes..

2.1.6. *Sistemas de Monitoreo en tiempo real*

Los sistemas de monitorización en tiempo real son una herramienta esencial para la gestión, control y prevención de incidentes críticos en diversos ámbitos, entre ellos la industria, la salud, el transporte y la seguridad vial. Su función principal es la adquisición, procesamiento y presentación inmediata de información para que los responsables de las operaciones o el usuario final puedan tomar decisiones inmediatas basadas en datos precisos y actualizados. La creciente necesidad de dar una respuesta rápida y eficiente a situaciones de riesgo ha impulsado el desarrollo de sistemas que combinan sensores, comunicaciones inalámbricas, procesamiento de datos y análisis predictivo para formar plataformas capaces de operar de forma autónoma o con asistencia; En este sentido, los sistemas de monitoreo en tiempo real han demostrado ser esenciales para mejorar la eficiencia operativa, prevenir accidentes y optimizar los recursos (Shi et al., 2016).

Los sistemas de monitoreo en tiempo real presentan ciertas principales características, incluyendo entre ellas:

- Actualización continua: esto garantiza información vigente ya que los datos se registran y transmiten al instante.
- Procesamiento inmediato: permite reacciones rápidas ya que la información es analizada por los algoritmos de manera simultánea a la adquisición.
- Alertas y notificaciones automáticas: l sistema comunica de forma automática a los usuarios la detección de eventos críticos.
- Integración: permiten recopilar información detallada del entorno mediante el empleo conjunto de sensores de proximidad, acelerómetros, giroscopios, cámaras o módulos GPS.
- Escalabilidad: la arquitectura de estos sistemas permite expandir la red de sensores o incorporar nuevas funcionalidades sin comprometer la velocidad de respuesta.

Estas características hacen que los sistemas de monitoreo en tiempo real sean particularmente útiles en entornos de alto riesgo donde los relays en la información pueden provocar accidentes o pérdidas importantes.

Arquitectura de un sistema de monitoreo

De manera general, estos sistemas están compuestos por tres componentes principales:

La capa de adquisición de datos que se encarga de la recopilación de información física o digital, mediante los sensores y dispositivos IoT; por ejemplo, en sistemas de seguridad vial, los sensores pueden medir velocidad, posición, aceleración y proximidad de vehículos o ciclistas;

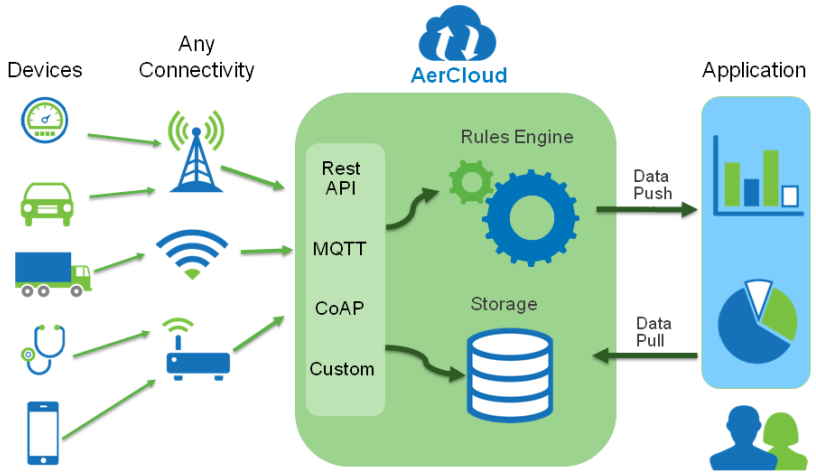
La capa de procesamiento y análisis que se encarga de interpretar los datos para determinar la presencia de eventos críticos por medio de los servidores, plataformas en la nube o unidades locales;

Y la capa de presentación y acción cuya función dentro de la arquitectura es permitir tomar acciones inmediatas por parte del usuario o del sistema automatizado al ver lo que está ocurriendo en las distintas interfaces que muestran los datos en tiempo real mediante dashboards, aplicaciones móviles o sistemas de notificación;

La eficiencia de un sistema en tiempo real depende de la velocidad de transmisión, la capacidad de procesamiento y la integridad de los datos, aspectos que deben ser cuidadosamente optimizados durante el diseño (Stankovic, 2014).

Figura 8

Arquitectura sistema de monitoreo



Nota. Tomado de https://aprendiendoarduino.wordpress.com/tag/arquitecturas-iot/.

Entre las desventajas presentes dentro de los sistemas de monitoreo en tiempo real se encuentran: dependencia de la conectividad ya que la transmisión de datos requiere redes confiables donde interrupciones pueden afectar la eficiencia del sistema; Otra desventaja es el tema de la seguridad de la información, puesto que la exposición de datos críticos requiere medidas de protección frente a accesos no autorizados o ciberataques; además el consumo energético de dispositivos portátiles deben equilibrar autonomía y capacidad de monitoreo constante; finalmente otro de los problemas graves se da en el procesamiento de grandes volúmenes de datos, donde la velocidad de adquisición y análisis puede generar sobrecarga si no se dimensiona correctamente la infraestructura.

Capítulo III: Diseño del Sistema

Este capítulo presenta en detalle el proceso del desarrollo del sistema, empieza con el desglose de la metodología utilizada y considerando el funcionamiento, eficiencia, seguridad y la facilidad de uso, asimismo, se presentan los requerimientos del sistema definidos a partir de la infraestructura vial para vehículos no motorizados. Este capítulo, conecta la información teórica

con la práctica, complementando mediante criterios establecidos para el diseño e implementación del dispositivo electrónico.

3.1 Enfoque metodológico

La metodología elegida para sustentar el desarrollo del proyecto es modelo Waterfall, adaptado a cuatro fases adecuarse al desarrollo de la tesis, en la cual ayuda a facilitar la integración de todas las etapas del sistema con el fin de conformar una estructura compleja y completa. En la Figura 8 están ilustradas las fases que conforman la metodología, las cuales se detallan a continuación.

Figura 9

Fases de la metodología modelo Waterfall



Nota. Adaptado de:(Adenowo & Adenowo, 2013)

En la fase de análisis conceptual, se llevará a cabo mediante la recopilación de información bibliográfica mediante un estudio de investigación, conformando información sustentable para la elaboración del dispositivo electrónico. En esta fase se evidencia en los capítulos I y II.

Durante la fase de diseño, se definen los requerimientos, la arquitectura del sistema, los componentes del hardware y software, diagramas de flujo que interactúan entre módulos, los requerimientos mencionados permiten conformar la estructura compacta de un sistema funcional. Esta fase se ve reflejada en el capítulo III.

Posterior a la etapa del diseño, se procede con la fase de implementación, se lleva a cabo la construcción del sistema, programación, montaje y la configuración de los dispositivos electrónicos. Esta etapa se consolida en el capítulo III.

Finalmente, en la fase de evaluación y resultados, se analiza los datos obtenidos del sistema implementado para asegurar el funcionamiento estructural y lógico, para llevar el cumplimiento estructurado de los objetivos. Esta etapa se desarrolla en el capítulo IV.

3.2 Situación actual

Actualmente la infraestructura vial del país es insuficiente en el ámbito de vehículos no motorizados, debido a la maña señalización vial y alta velocidad de los vehículos motorizados. El ciclismo ha experimentado un crecimiento exponencialmente como una actividad deportiva y en ciertas ocasiones como transporte, esto ha incrementado los riesgos viales de los ciclistas debido a la falta de infraestructura, tanto en zonas urbanas como en semiurbanas.

En tema de infraestructura el déficit de no tener carriles exclusivos en la mayoría de zonas urbanas como en semiurbanas y conjuntamente con las condiciones geográficas complejas, lo que aumenta la tasa de accidentes y dificulta la circulación segura de los ciclistas en la vía. A su vez,

la falta de sistemas de alertas termina de limitar la capacidad de respuesta ante situaciones de riesgo.

En consecuencia, todos estos factores provocan un impacto negativo en personas que utilizan este medio como transporte o como actividad y como resultado se obtiene el aumento de accidentes, lesiones y por consiguiente muertes.

3.3 Requerimientos del sistema

Posteriormente, tomando en cuenta la etapa de diseño de la metodología descrita anteriormente, se mencionan algunos de los requerimientos necesarios para el ensamblaje del dispositivo electrónico. En este apartado se abarca desde la arquitectura general del sistema, desglosando el flujo de bloques, con el fin de llegar a facilitar el diseño y posteriormente su implementación.

Los requerimientos planteados en esta sección, se dividieron en dos grupos diferentes como son: Funcionales y de Arquitectura, asignado así prioridades desde la mas baja hasta la mas alta para los diferentes requerimientos, en donde, las altas son requisitos obligatorios para que el sistema funcione correctamente y las bajas son mejoras adicionales. Al tener estos requerimientos claros permitirá elegir correctamente cada elemento a utilizar y software que sea necesario en la implantación del sistema.

3.3.1. *Nomenclatura de requerimientos*

Para mejorar la facilidad de comprender los términos utilizados en el desarrollo del dispositivo electrónico, en la tabla 3 se representan las abreviaturas técnicas utilizadas junto con el significado de cada criterio.

Tabla 1

Nomenclatura de Requerimientos

Abreviatura	Significado
EF	Requerimientos funcionales
EA	Requerimientos de arquitectura

La tabla anterior muestra los requerimientos del sistema indicando los funcionales y de arquitectura con el fin de facilitar la trazabilidad durante las etapas de diseño, implementación y evaluación de resultados, los requerimientos funcionales designan las acciones que el dispositivo electrónico debe realizar mientras que los de arquitectura establecen como deben ir conectados los diferentes componentes.

3.3.2. *Especificaciones del sistema*

Los requerimientos del sistema definen las acciones esenciales que el sistema debe ejecutar, las cuales se detallan en la Tabla 4.

Tabla 2

Requerimientos del sistema.

Requerimientos del sistema				
No.	Requerimientos	Prioridad		
		Alta	Media	Baja
Requerimientos Funcionales				
EF-01	El sistema debe identificar eventos de impacto o caídas.	x		
EF-02	El sistema tiene que ser capaz de registrar la localización en tiempo real.	x		
EF-03	El sistema transmitirá una alerta después de cada evento o accidente.	x		

EF-04	El sistema almacenará los datos registrados en la memoria de la Raspberry PI y luego enviado a la nube.		x	
Requerimientos de Interfaz de la aplicación				
EF-05	La interfaz incluye visualmente las alertas en tiempo real.		x	
EF-06	La aplicación permite acceder a los eventos registrados.		x	
EF-07	Aplicación móvil descargable.	x		
EF-08	El sistema contara con un rol especifico de usuario, para funciones básicas, para administrar y para configuraciones.		x	
Requerimientos de uso				
EF-09	El dispositivo prioriza el uso del usuario final.	x		
EF-10	Ofrecer manual del usuario.			x
EF-11	El sistema garantiza operabilidad continua durante el recorrido.	x		

3.3.3. *Estructura del sistema*

Las especificaciones de arquitectura (EA) determinan los componentes del sistema, lo que permite facilitar la elección del hardware, software y los elementos de diseño necesarios para tener un funcionamiento eficiente, estos aspectos se muestran en la Tabla 5.

Tabla 3

Estructura del sistema.

Requerimientos de arquitectura EA				
No.	Requerimientos	Prioridad		
		Alta	Media	Baja
Requerimientos de Diseño				
EA1	El sistema debe contar con una estructura ligera y compacta.		x	
EA2	El sistema requiere que la arquitectura permita distribuir los elementos de manera eficiente, que garanticen comodidad.	x		
EA3	Se espera que el dispositivo sea capaz de soportar condiciones exteriores.		x	
Requerimientos de Hardware				
EA4	Es importante que el procesador permita ejecutar múltiples tareas simultáneamente.	x		
EA5	El microcontrolador necesita admitir varias interfaces de conexión, analógicas y digitales.		x	
EA6	El dispositivo debe integrar comunicación LTE/4G para transmisiones continuas.	x		
EA7	Es importante verificar que los sensores trabajen con un voltaje compatible.		x	
EA8	El dispositivo deberá operar con una fuente de energía autónoma.		x	
Requerimientos de Software				
EA9	Se plantea utilizar software de desarrollo libres, para mayor flexibilidad.		x	

EA10	El software debe comprender la compatibilidad de librerías.		x
EA11	El sistema debe permitir la comunicación y el envío de notificación en tiempo real cuando detecte un evento.	x	

3.4 Diseño del sistema de monitoreo

El diseño del sistema se basa en el análisis desarrollado en los capítulos 2 y 3, en los cuales se evidenciaron las especificaciones necesarias, las cuales se estructuran en módulos que ayudan a la recolección, transmisión y la recepción de los datos obtenidos. El dispositivo comprende la conexión de sensores que tratan de medir las variables del entorno, en donde, un microordenador realiza el proceso de datos obtenidos y mediante el soporte del módulo LTE/4G permite la transmisión de datos en la nube.

Además, el sistema permite tener una autonomía energética eficiente y la confiabilidad del envió y reenvió de datos en tiempo real, aspectos que permites que el dispositivo funcione correctamente.

3.4.1. *Selección del hardware*

Para el desarrollo del sistema, se tomaron en cuenta diferentes criterios técnicos y funcionalidad de acuerdo al estudio de requerimientos de arquitectura, entre ellos se destaca el criterio de compatibilidad con los componentes sean analógicos o digitales (EA-5), la capacidad que le permite realizar comunicación inalámbrica (EA-6), la habilidad de realizar tareas

simultaneas sin entorpecer el funcionamiento (EA-4). Además, se valoró el factor de carga energética el cual debe funcionar por largos periodos (EA-8).

3.4.1.1 Microordenador

Bajo la consideración de estos requerimientos se realizó una tabla técnica que permite evaluar diversas alternativas mediante los requisitos establecidos, donde permite visualizar el cumplimiento de cada alternativa.

La Tabla 4 presenta un enfoque detallado para la elección del microordenador de manera más específica y adecuada.

Tabla 4

Selección del microordenador

Hardware	**Requerimientos**											**Valoración**
	EF-01	EF-02	EF-03	EF-04	EA4	EA5	EA6	EA7	EA8	EA9	EA10	Total
ESP32	1	1	1	1	1	1	0	1	1	1	1	10
NodeMCU	1	1	1	0	0	1	0	1	1	1	1	8
STM32	1	1	1	1	1	1	0	1	1	1	1	10
Raspberry Pi Zero 2 W	1	1	1	1	1	1	1	1	1	1	1	11

Considerando los resultados, y con base en las capacidades de procesamiento, funcionalidad, compatibilidad y conectividad del microordenador se optará por elegir la placa Raspberry Pi Zero 2W para el desarrollo del sistema, la tabla 4 muestra el análisis y evaluaciones de cuatro tecnologías frente a 11 requerimientos críticos del sistema, aunque el ESP32 y el SMT32 alcanzan una puntuación binaria alta, pero su velocidad de transmisión es limitada, la Raspberry

Pi Zero 2 W por otro lado tiene un mejor rendimiento en tareas simultaneas y una conexión inalámbrica adaptable que no depende solo de wifi. Como resultado, esta opción proporciona una solución robusta en el sistema, por su capacidad de memoria y su adaptabilidad módulos sin necesidad de integrarlos.

Figura 10

Raspberry Pi Zero 2W



Nota. (Raspberry Pi Foundation, 2023).

3.4.1.2 Microcontrolador (ESP32 WROOM-32)

El microcontrolador basado en ESP32 WROOM-32 constituye uno de los componentes principales dentro del sistema propuesto, ya que es el encargado de la adquisición, procesamiento inicial y transmisión de los datos obtenidos a partir de los sensores. Este dispositivo actúa como un enlace directo entre el entorno físico y la infraestructura digital del sistema, permitiendo capturar información relevante de manera continua y confiable.

El ESP32 WROOM-32 fue seleccionado debido a sus características técnicas, entre las que destacan su capacidad de procesamiento, bajo consumo energético y la integración nativa de módulos de comunicación inalámbrica como Wi-Fi y Bluetooth. Estas prestaciones lo convierten en una solución adecuada para aplicaciones de tipo IoT, donde se requiere eficiencia, conectividad y flexibilidad en un solo dispositivo.

Además, el microcontrolador no se limita únicamente a la recolección de datos, sino que también realiza tareas básicas de procesamiento, como la validación y organización de la información antes de su envío al sistema central. De esta manera, se reduce la carga de procesamiento en etapas posteriores y se mejora la eficiencia general del sistema, garantizando una transmisión de datos más ordenada y confiable.

Figura 11

Esp32 Wroom-32



Nota. (Espressif Systems, 2026).

3.4.1.3 Comunicación Inalámbrica

Con el propósito de establecer una tecnología inalámbrica estable entre varias tecnologías, basándose en los parámetros obtenidos de los requerimientos puntuales antes mencionados se realizó una valoración general de cada tecnología compatible mencionadas en la tabla 5.

Tabla 5

Selección placa de conectividad inalámbrica

Hardware	**Requerimientos**					**Valoración**
	EA6	EA8	EA11	EF-02	EF-03	Total
LoRa	0	1	0	1	0	2

Esp32	1	0	1	1	1	4
Bluetooth Low Energy (BLE)	0	1	0	1	0	2
Módulo LTE EG25-G	1	1	1	1	1	5

De acuerdo al análisis realizado en la tabla 5 se puede concluir mediante una puntuación binaria (1=cumple, 0=no cumple), que la mejor placa para realizar la conectividad inalámbrica fue el Módulo LTE EG25-G ,este componente es ideal para llevar a cabo en la implementación, debido a la garantía que ofrece al momento de tener una salida con conexión a internet estable, aunque las tecnologías Esp32 y Lora ofrecen una mejora en el tema de consumo energética no tienen la capacidad para ofrecer conectividad por su rango. En contraste con otras tecnologías, el módulo LTE ofrece una garantía de conexión más amplia y estable, esto gracias a que el módulo permite la conexión mediante datos móviles. Al considerar todos los factores, esta tecnología hace frente a los requerimientos críticos del sistema, por consiguiente, se optará por elegir esta tecnología para el desarrollo final del sistema.

Figura 12

Módulo LTE EG25-G



Nota. (Quectel Wireless Solutions, 2026).

3.4.1.4 Sensores

El sistema incorporará sensores de respuesta ante eventos críticos, movimientos anormales y geolocalización en tiempo real, de acuerdo a los requerimientos establecidos y a la compatibilidad con el microcontrolador se analizó y eligió los diferentes sensores, para el sensor de proximidad se realizó una evaluación de acuerdo a requerimientos específicos para su elección, los sensores a comparar son el HC-SR04 basado en ondas ultrasónicas a distancia y el VL53L0X basado en láser de proximidad o Time of Flight. En la tabla 6 se realiza la comparación binaria para su elección.

Tabla 6

Selección de sensor de proximidad.

Hardware	**Requerimientos**						**Valoración**
	EF-01	EF-02	EF-03	EA5	EA7	EA8	
HC-SR04	1	0	1	1	0	1	6
VL53L0X	1	1	1	1	1	1	4

Con el objetivo de seleccionar el sensor de proximidad que cumpla los requerimientos propuestos, se concluye que el sensor HC-SR04 será el sensor ideal para el dispositivo. La tabla 6 expone una valoración superior frente al sensor VL53L0X, el cual presenta limitaciones con precisión y poca compatibilidad de voltaje lo que implicaría adaptar reguladores de voltaje, por otro lado, el sensor HC-SR04 permite medir la proximidad mediante ultra sonido siendo más preciso, además que utiliza buses de comunicación IC2. Además, utiliza tecnología láser para medir las distancias con mayor precisión en cortas distancias. Por lo tanto, se considera más conveniente utilizar dicho sensor para implementar en el sistema.

Figura 13

HC-SR04

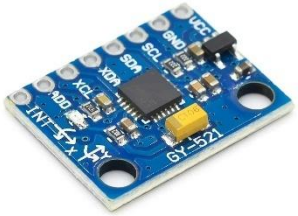


Nota. (Elecfreaks, 2026).

Para garantizar la precisión de cada evento relevante durante el recorrido el sensor GY-521 (MPU6050) permite detectar movimientos bruscos o impactos, mide tanto la aceleración lineal como la rotación angular en los tres ejes, lo que permite verificar si el usuario realiza un giro peligroso o un movimiento anormal lo que lleva a tener algún tipo de caída o accidente,

Figura 14

GY-521 (MPU6050)



Nota. (InvenSense, 2026).

Con el fin de garantizar alertas oportunas se incorporó luces led cuenta con una flexibilidad de uso, consumo energético bajo y sensibilidad manejable que permite detectar eventos

ocasionados como sacudidas o caídas, con el fin de identificar impactos en el trayecto, tiene el funcionamiento de un interruptor interno que se acciona al detectar oscilaciones físicas.

Además, el sensor GPS NEO-6M cumple uno de los roles más importantes dentro del sistema, el cual permite obtener la ubicación geográfica especifica del usuario y en tiempo real, este sensor es clave para enviar alertas incluyendo las coordenadas y permitiendo geolocalizar al usuario en caso de eventos como accidentes o alertas.

Figura 15

GPS NEO-6M



Nota. (u-blox, 2011).

3.4.1.5 Selección fuente de alimentación

Después de realizar la elección de los sensores y dado que se integrarán a un único dispositivo el cual será portátil y deberá estar en constante movimiento, es imprescindible elegir una fuente de alimentación que cumpla con todas las características de voltaje de placas y sensores. Para determinar la fuente más adecuada al sistema, en la tabla 7 se presentan los amperajes de los componentes electos cuando se encuentran en funcionamiento. Con el fin de garantizar un rendimiento optimo y autonomía adecuada del sistema, en la tabla se realiza una evaluación de las demandas energéticas.

Tabla 7

Consumo energético del sistema

Componente	En Funcionamiento	Modo Sleep
Raspberry Pi Zero 2 W	260 mA	120 mA
Módulo LTE EG25-G	180 mA	10 mA
GPS NEO-6M	45 mA	-
SW-420	3 mA	-
GY-521 (MPU6050)	3.9 mA	0.5
VL53L0X	20 mA	1
Total	511.9 mA	131.5 mA

De acuerdo a la tabla 7, la evaluación de amperaje que consume cada componente en funcionamiento es de 511.9 mA, para determinar el consumo promedio del sistema y valorar la duración de la fuente de alimentación, la ecuación (1) permite realizar el cálculo de consumo de batería.

$$Consumo = \frac{(T_{cn} * I_{cn}) + (T_{cd} + I_{cd})}{T_{cn} + T_{cd}} \quad (1)$$

Donde:

T_{cn}: Tiempo en funcionamiento activo (segundos)

I_{cn}: Corriente total en funcionamiento activo (mA)

T_{cd}: Tiempo en modo de bajo consumo o espera (segundos)

I_{cd}: Corriente total en modo sleep (mA)

Remplazamos los datos en los parámetros:

T_{cn}: 5 horas = 18000 segundos

I_{cn}: 511.9 mA

T_{cd}: 0

I_{cd}: 131.5 mA

Reemplazando en la ecuación (1) se obtiene:

$$Consumo = \frac{(18000 * 511.9)(mA) + (18000 + 131.5)(mA)}{18000(s)} \quad (1)$$

$$Consumo = 511.9\ (mA) \quad (1)$$

Para obtener el tiempo estimado de duración de la fuente de alimentación se utiliza la ecuación (2).

$$Duración = \frac{(C_b)}{C_p} \quad (2)$$

Donde:

C_b: Capacidad de la batería (mAh)

C_p: Consumo promedio (mA)

$$Duración = \frac{5000\ (mAh)}{511.9\ (mA)} = \quad (2)$$

$$Duración = 9.76\ (horas) \qquad (2)$$

Para el sistema será necesario una batería de litio de 5000 (mA) a 5 (V), en donde el sistema pueda auto sustentarse de forma continua durante un periodo aproximado de 10 horas, llegando a superar los requerimientos. Esto garantiza un funcionamiento continuo y una adecuada autonomía para trayectos largos sin la necesidad de realizar una recarga continua del sistema.

3.5 Selección del software

De acuerdo a la selección del hardware, y en función de los requerimientos se ha priorizado mantener la compatibilidad del microordenador con los componentes, la gestión eficiente de los sensores y transmisión en tiempo real. Por esta razón la elección del software se enfocó en aspectos como la capacidad de la memoria del microordenador, librerías que permiten cubrir actividades como el censado y control. De esta manera, se logra cumplir la operabilidad del sistema mediante la elección de requerimientos y asegurar que se está cumpliendo los objetivos propuestos.

3.4.2.1 Plataforma de desarrollo del microordenador

Dado que el microordenador será la Raspberry Pi Zero 2 W utiliza el sistema operativo por defecto Raspberry Pi OS basado en Debian Linux, este sistema permite efectuar múltiples tareas, para la programación de la plataforma y comunicación entre los sensores mediante la comunicación del microcontrolador se utilizara el lenguaje de desarrollo Python, esto gracias a su compatibilidad con comunicación digital, analógica y por su variedad en librerías, además, se integrara el software Node-Red para efectuar la parte visual o dashboard de la comunicación inalámbrica y envió de datos por el protocolo MQTT (Mosquitto). En conjunto, estas herramientas de desarrollo permiten garantizar la operabilidad, compatibilidad y rendimiento en el consumo energético para el sistema propuesto.

Figura 16

Raspberry Pi OS basado en Debian Linux



Nota. (Raspberry Pi Foundation, 2023).

3.4.2.2Plataforma de desarrollo del microcontrolador (ESP32-WROOM-32)

Para el desarrollo de la programación en este nodo, se eligió el software Arduino IDE por su compatibilidad con la mayoría de sensores, en donde permite programarlos utilizando sus librerías nativas del mismo, además, tiene un buen manejo con el funcionamiento de comunicación serial y además con la comunicación mediante un puente denominado Broker que este maneja a su vez el protocolo MQTT, con la finalidad de que este software permita la comunicación con los sensores y el microcontrolador principal este cuenta con las librerías nativas del protocolo IEEE 802.11 b/g/n para realizar una comunicación inalámbrica en una red LAN o WAN. De acuerdo a estas medidas, el software elegido es el indicado para realizar la programación de la primera etapa de la arquitectura con el fin de que permita enviar los datos sin problema a cualquier base de datos de forma inalámbrica.

Figura 17

Arduino IDE



Nota. (Arduino, 2025).

3.4.2.2 Bases de datos

Con base a los requerimientos se ve en la necesidad de tener un mecanismo que permita almacenar y gestionar los datos generados de los sensores o cualquier tipo de eventos, en este sentido, la base de datos es una herramienta que cumple con la función de reunir los datos del nodo incorporado estratégicamente en el sistema, los datos enviados se contemplan eventos como alertas, registros históricos que permiten verificar los eventos y realizar un análisis, además de verificar el estado del sistema como la monitorización de la batería o la conectividad. De este modo, la información almacenada puede ser consultada posteriormente por la aplicación web o por servicios de monitoreo. El software a utilizar es Supabase y es una distribución de SQL, fue elegido por su compatibilidad y amplio manejo de datos almacenados, sean estos visibles en la nube sin la necesidad de estar conectado a la misma red del microordenador. Finalmente, la elección de una base de datos radica en la necesidad de garantizar funcionamiento estable del sistema y además de la optimización de recursos en el microordenador principal.

Figura 18

Base de datos Supabase



Nota. (Kanwal et al., 2024)

3.4.2.3 Herramientas de visualización y monitoreo

Para la visualización y monitoreo de los datos obtenidos mediante los sensores, se propone utilizar una plataforma que permita observar el comportamiento de todo el sistema en tiempo real, además de registrar cada evento en distintos tipos como crítico, moderado y leve. De este modo, Node-Red se presenta como una plataforma adecuada debido a que se complementa con aplicaciones IoT, porque permite realizar el diagrama de flujo y mediante la utilización del dashboard incorporado, es posible representar el comportamiento de los sensores de una manera grafica. De igual manera, el sistema permite identificar los eventos críticos mediante las alertas visuales que se implementan en la aplicación, tanto para el usuario como para el servidor. Por este motivo, esta herramienta es fundamental porque ayuda a supervisar el sistema y que el usuario tenga una mejor experiencia al momento de verificar los datos del sistema.

Figura 19

Visualización de datos Node-Red



Nota. (OpenJS Foundation, 2026).

3.6 ETAPA DE DISEÑO

3.6.1. *Arquitectura del sistema*

La arquitectura del sistema adopta una topología de cinco capas, en donde está diseñada para garantizar la baja latencia y la alta disponibilidad del mismo, este flujo de información transita desde la adquisición de datos del ciclista hasta llegar a visualizar en la nube, para ello, se ha implementado un modelo denominada Edge Computing, el cual sirve para dispersar el procesamiento y así poder tener una optimización en el uso del ancho de banda celular como se observa en la figura 20.

En la primera capa física, se encuentra distribuido todos los sensores utilizados en el sistema que actúan como un punto de contacto directo hacia el ciclista, en donde realiza capturas de telemetría en tiempo real. El acoplamiento de los sensores se realiza mediante buses de comunicación como I2C y UART, con lo que se garantiza la confiablidad de las señales físicas. Además, el nodo ESP32 tiene la función de centralizar toda la información recolectada en variables antes de pasar a su transmisión inalámbrica.

Para el proceso de conectividad inalámbrica en redes WLAN para una transmisión local, se estableció un enlace inalámbrico basado en el estándar IEEE 802.11 b/g/n, que permite enlazar tecnologías mediante una frecuencia de 2.4 GHz, esto permite crear un sistema de red privada y segura. Para él envió de datos, esta capa se apoya específicamente en el conjunto de protocolos TCP/IP en donde el protocolo TCP (Transmission Control Protocol) ayuda a establecer una conexión confiable entre el origen y el destino, esto lo hace dividiendo los paquetes y el protocolo IP (Internet Protocol) se encarga de redireccionar y enrutar los paquetes a través de la red mediante el uso de IPs únicas.

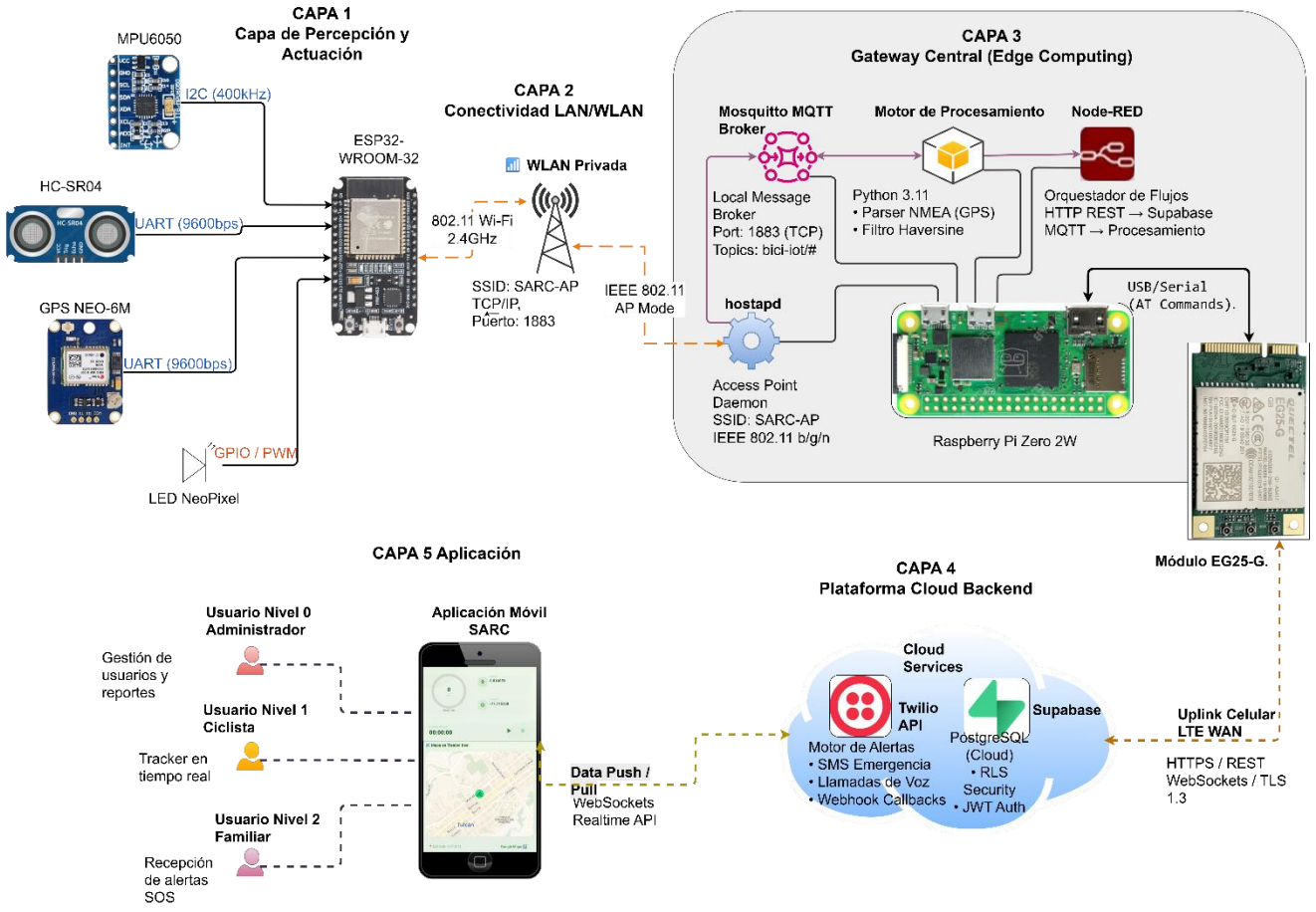
En la capa tres se encuentra el núcleo de procesamiento local, el cual ejecuta algoritmos matemáticos como el filtro de Haversine, sin depender de la nube. Al tener le procesamiento en Python se permite minimizar el tiempo de respuesta ante posibles accidentes críticos. Además, la salida hacia la red WAN esta gestiona mediante la salida a través del módulo LTE, el cual establece la puerta de enlace entre la red local y los servidores externos.

Pasando a la capa cuatro en la cual el backend el cual gestiona el procesamiento de datos mediante una lógica interna para llevar la comunicación con la base de datos y servicios externos, lo que asegura la integridad de los datos del ciclista, gracias al motor de alertas que se encarga de interactuar mediante APIs para realizar una notificación inmediata a los contactos de emergencia.

Finalmente, en la capa cinco se encuentra la aplicación móvil nativa que será la encargada de entregar interfaces personalizadas según el rol en el que cuente el usuario, con esto se garantiza, una experiencia fluida y en tiempo real.

Figura 20

Arquitectura del sistema



Nota. Autoría propia.

3.6.2. *Comunicación y protocolos*

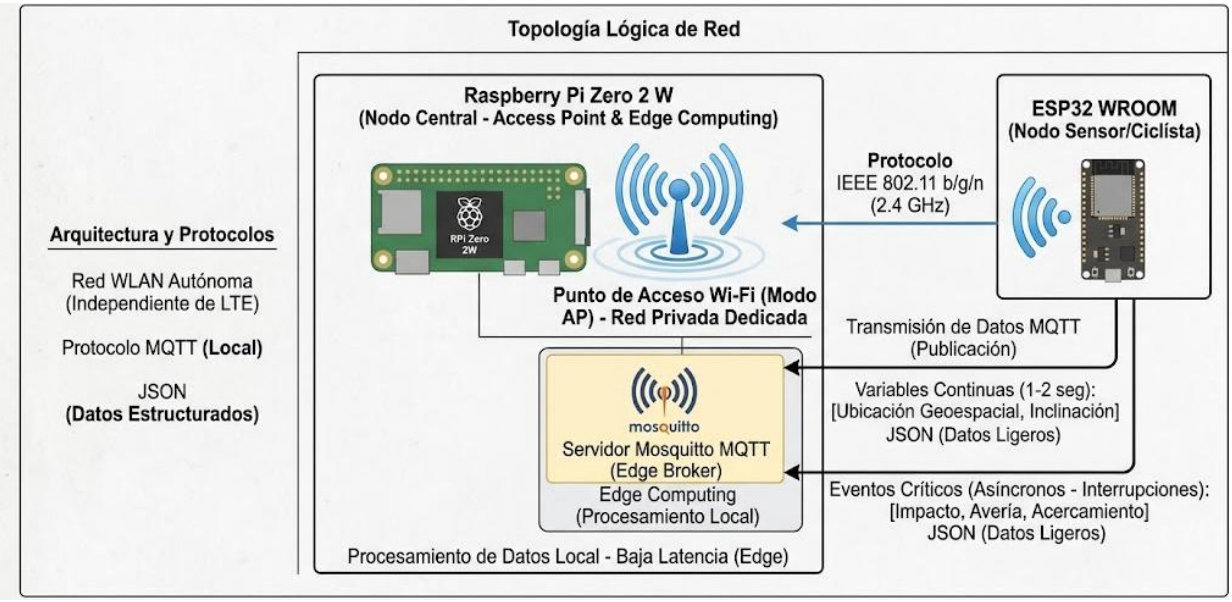
Para garantizar el enlace de los datos, la arquitectura del sistema se basa en un modelo de red WLAN (área local inalámbrica) que es totalmente autónomo, en este sentido, el microordenador central o Raspberry Pi Zero 2W es configurada con el fin de que pueda operar como Access Point (punto de acceso), lo cual permite crear una red privada dedicada exclusivamente al sistema de seguridad. El microcontrolador basado en la placa ESP32 WROOM se comunica con el microordenador central a través del protocolo IEEE 802.11 b/g/n. Esta es una tecnología que permite la transmisión de datos mediante ondas de radio en la frecuencia de 2.4 GHz. De esta manera, fue necesario implementar el servidor de mensajería denominado Mosquitto

dentro del microordenador central que obedece al modelo Edge Computing, este modelo realiza el procesamiento de datos lo más cerca posible a la fuente de donde se generan, en lugar de enviarlos a un servidor local o la nube, permitiendo así reducir el tiempo de respuesta y latencia obteniendo una eficiencia alta. Bajo este esquema, el protocolo MQTT trabaja de forma local lo que al mantener al bróker de forma local garantiza que el sistema siga funcional y sea capaz de procesar los datos incluso si el sistema no cuenta con una cobertura de celular LTE por la geográfica del entorno, paralelamente, el empaquetado de los datos se realiza mediante el estándar JSON, el cual es un formato de texto estándar que permite almacenar y transferir datos estructurados basados en sintaxis de objetos de JavaScript, estos objetos en texto plano resultan ligeros para el protocolo MQTT.

Por consiguiente, la transmisión de datos hacia el bróker implementa un esquema hibrido el cual está diseñado para optimizar el tiempo e respuestas del sistema ante emergencias, las variables de estado continuo, como la ubicación geoespacial y la inclinación del chasis de la bicicleta se envían periódicamente en intervalos de uno a dos segundos, por el contrario, los eventos críticos suscitados en la ruta y que comprometan la seguridad del ciclista, como detección de impacto, avería mecánica o el acercamiento peligrosos de un vehículo, esto desencadena una transmisión asíncrona controlada mediante interrupciones de hardware con el fin de asegurar una latencia mínima en la alerta.

Figura 21

Topología lógica de red



Nota. Autoría propia.

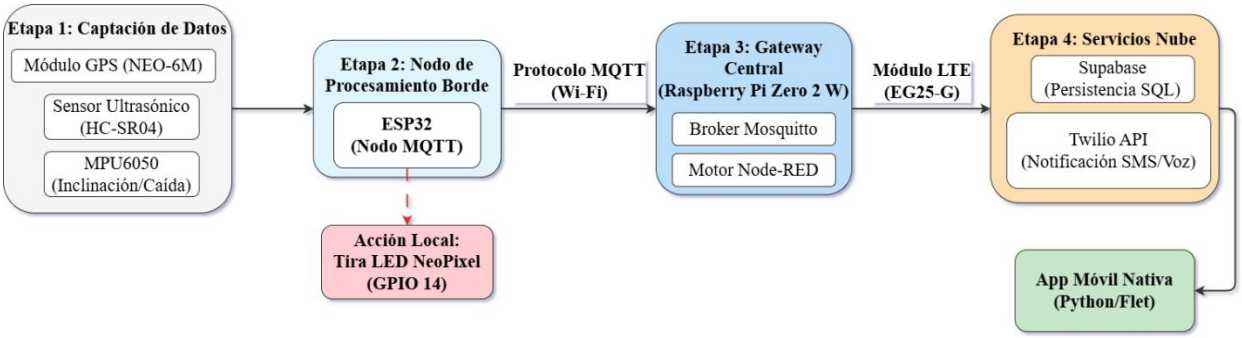
3.6.3. *Diagrama de bloques y flujos de datos*

Para separar el manejo de datos, el modelo estructural se divide jerárquicamente en cinco capas, está estructurada bajo el funcionamiento de Fog Computing (computación en la niebla), en donde es una arquitectura informática descentralizada que permite extender las capacidades de procesamiento y almacenamiento desde los centros de datos hacia los nodos más cercanos donde generan los datos, como routers, Gateway o servidores locales. En la capa inicial denominada Captación de datos, la adquisición de los parámetros físicos del sistema, los cuales recaen específicamente en los sensores de proximidad y de inclinación encargados de monitorear el perímetro y la estabilidad del chasis de la bicicleta. Siguiendo este flujo de datos, la capa Nodo, ESP32 Wroom-32 asume el rol como nodo primario del sistema, el cual está configurado como cliente mediante el protocolo MQTT con el fin de empaquetar y digitalizar las lecturas de los datos en estructuras JSON antes de ser transmitidos por Wi-Fi local. En este nivel, capa Gateway, el microordenador principal o Raspberry Pi Zero 2W actúa como el Gateway del sistema, en donde está albergando al bróker Mosquitto y el motor de eventos Node-Red que permite filtrar y enrutar

las tramas de telemetría que no depende principalmente de internet, de esta manera el aviso físico el microordenador principal procesa las métricas y gestiona él envió hacia la nube mediante el enlace del módulo LTE. Como mecanismo de respuesta, en la capa Nube, permite que los datos históricos y geoespaciales se consoliden en la base de datos Supabase, esto permite actuar de forma inmediata en escenarios de caída o impacto, el motor de flujos dispara el consumo de la API Twilio la cual permite enviar notificaciones de SMS a contactos de emergencia. A través del enlace inalámbrico, permite que la capa cinco actúe como la interfaz final del usuario, permitiendo materializar toda la infraestructura mediante una aplicación móvil nativa, proporcionando acceso a las métricas del recorrido.

Figura 22

Diagrama de bloques del sistema

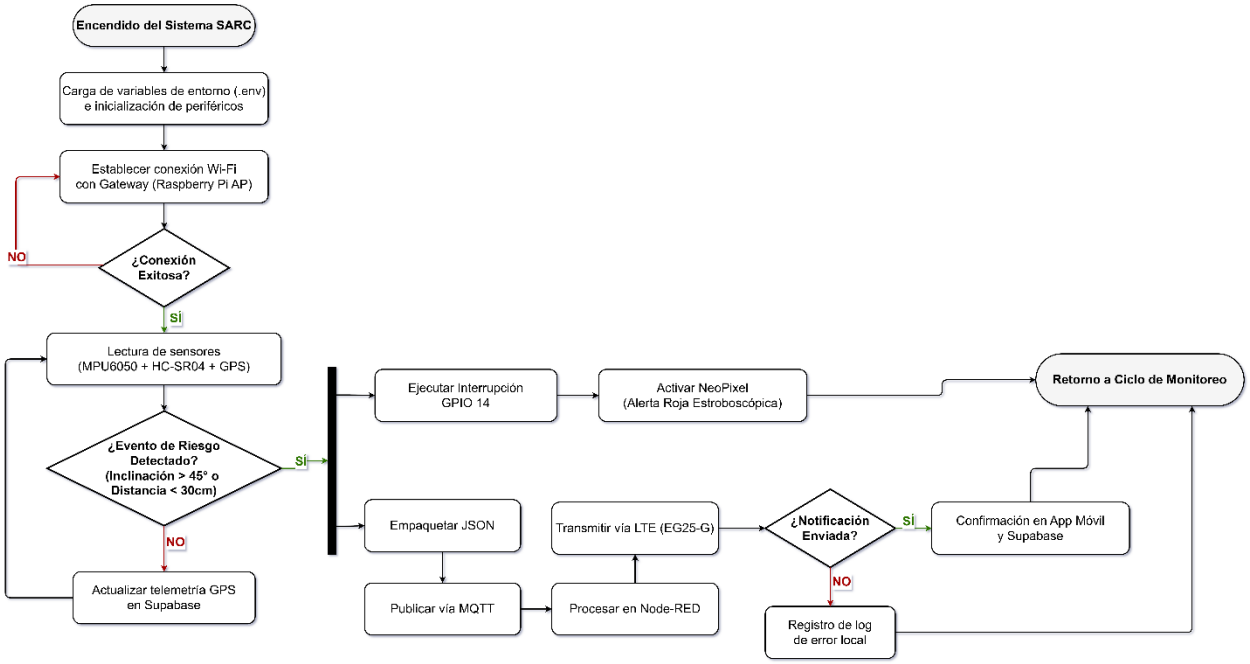


Nota. Autoría propia.

Finalmente para detallar la lógica del sistema se realiza un diagrama de flujo en el cual se verifica la lógica que utiliza el sistema para realizar la consolidación de cada bloque mencionado anteriormente como se observa en la figura 23, la integración permite que el sistema actúe con eficacia y reduciendo la latencia ante eventos críticos, además, que el usuario siempre está en constante localización para permitir que cualquier incidente ocurrido en el recorrido el familiar acuda de manera inmediata reduciendo así el tiempo de respuesta antes accidentes.

Figura 23

Diagrama de flujo del sistema



Nota. Autoría propia.

3.7 ETAPA DE IMPLEMENTACIÓN

3.7.1. *Desarrollo del dispositivo*

Posteriormente, y de acuerdo con la metodología propuesta durante el enlazamiento del diagrama de bloques de la arquitectura, se lleva a cabo el desarrollo y la integración individual de cada componente. El dispositivo se combina principalmente de dos componentes: adquisición de datos y procesamiento de datos, el cual es el encargado de capturar datos enviados mediante el nodo (ESP32-Wroom-32) y enviarlos a través de conexión a internet a la base de datos en la nube Supabase.

En esta sección, se explica detalladamente la infraestructura física involucrada en la construcción del dispositivo, abordando la conexión de componentes y su integración con el fin de garantizar el correcto funcionamiento del sistema.

3.7.2. *Sistema de alimentación y autonomía*

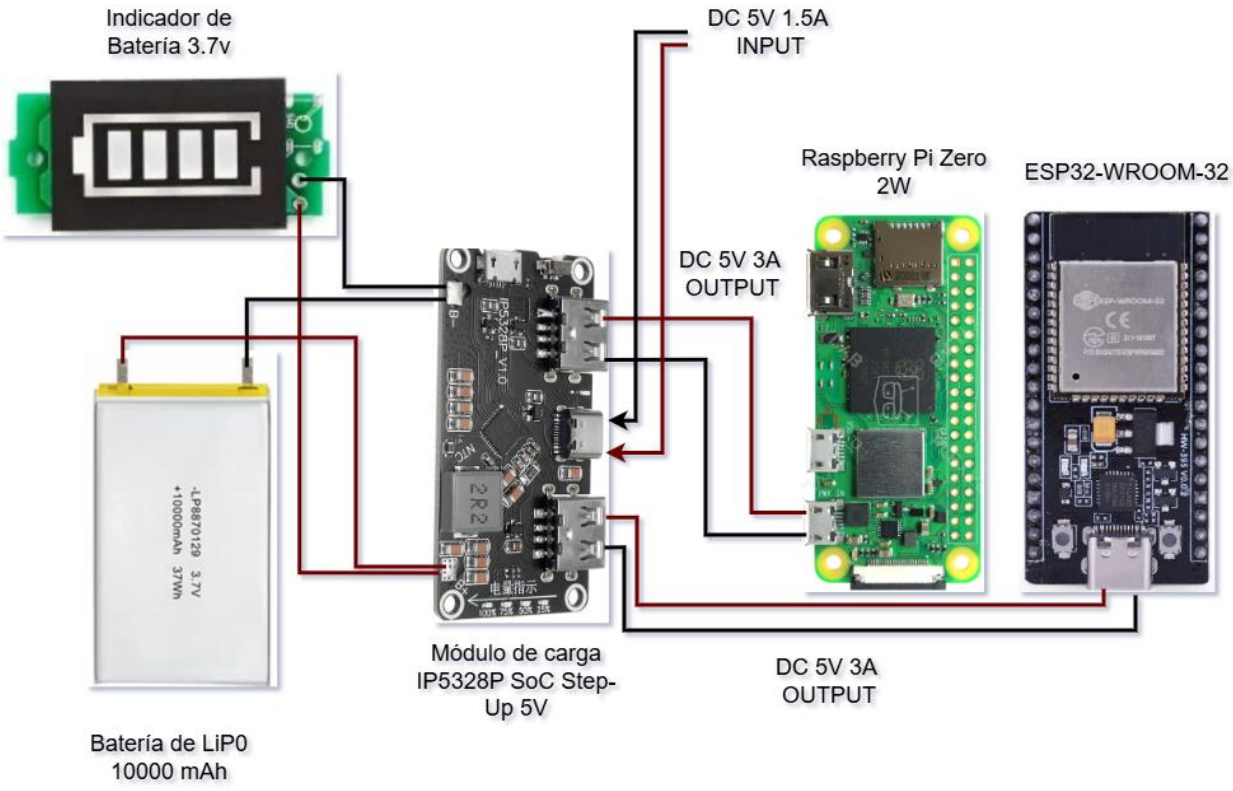
Para la alimentación de todo el sistema, utilizando un bloque de polímero de litio de 3.7V con una capacidad de 10000mAh, esto con el fin de no alterar el peso y el balance mecánico de la bicicleta. Para satisfacer las altas exigencias del hardware de la arquitectura, el diagrama de conexiones de la figura 24 detalla cómo se implementa el módulo de alimentación bidireccional IP5328P como el principal motor de potencia, la elección de este componente está relacionado a su capacidad de manejar y entregar capacidad robusta de corriente y a su alta eficiencia al momento de actuar como Step-Up, permitiendo elevar la tensión nominal de una celda a 5V constantes.

Bajo esta configuración, en la figura 24 se muestra la implementación de un módulo indicador de voltaje en tecnología LED, el cual otorga al usuario la visibilidad inmediata de carga de la batería antes de salir a ruta y luego de llegar de ruta.

Esto garantiza que el sistema actúe de manera continua durante un periodo de 4 a 5horas, cubriendo con un margen de seguridad durante todo el recorrido en el tiempo de un entrenamiento estándar.

Figura 24

Diagrama de alimentación del sistema



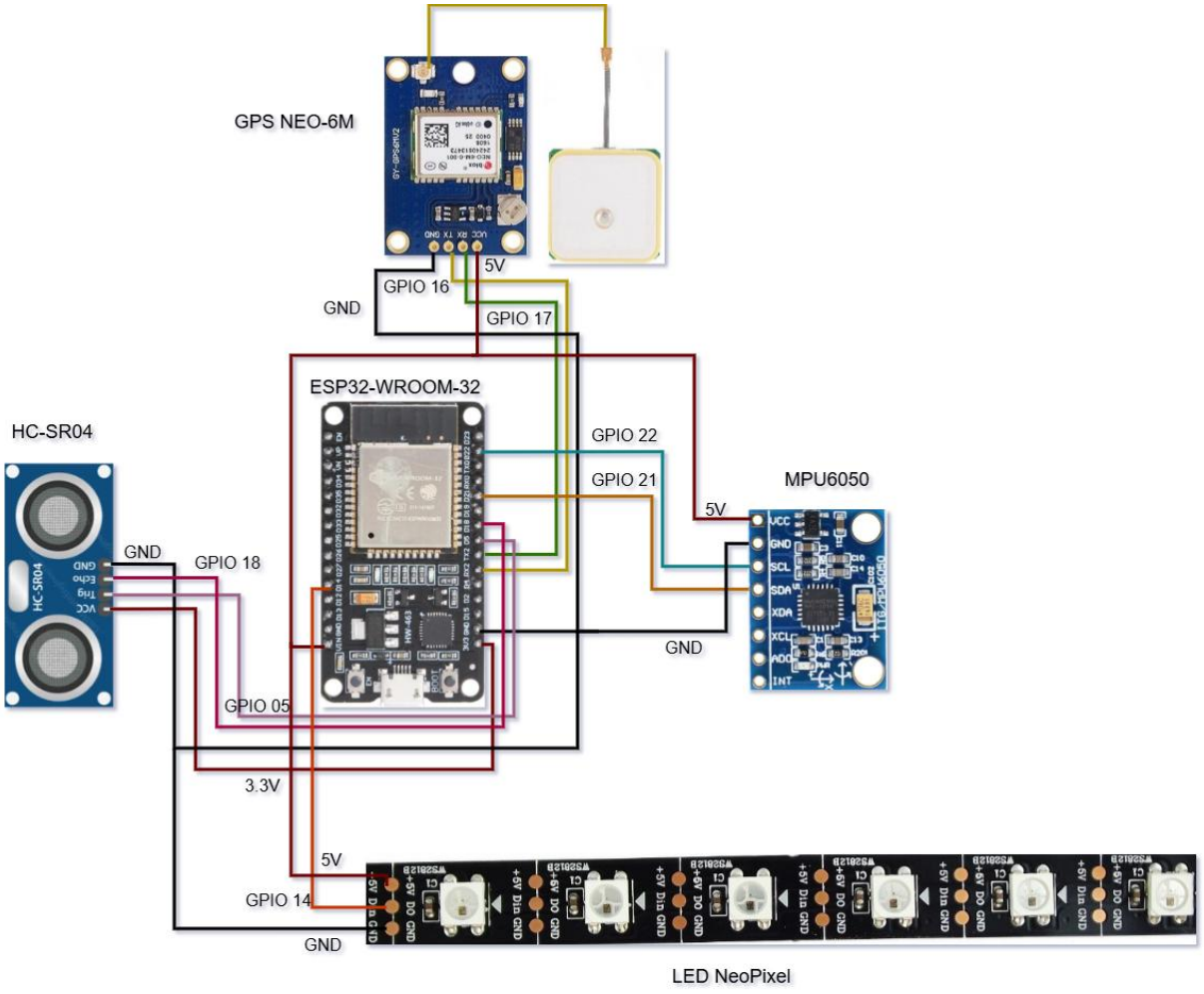
Nota. Autoría propia.

3.7.3. *Integración de sensores*

En la figura 25 se detalla la ubicación estratégica de cada elemento, con el fin de garantizar el funcionamiento correcto esto involucra que el punto de medición corresponda a la variable física que se va a evaluar. Para el sistema de geolocalización en tiempo real, se instaló el módulo GPS NEO-6M en el cuadro de la bicicleta, para mitigar esta atenuación, el sensor se instaló lejos de la electrónica de potencia y sin incomodar o quitar la conformidad del ciclista, esta posición del sensor permite asegurar una línea de vista (Line-of-Sight, LoS) con el fin de minimizar la interferencia de las ondas de radiofrecuencia (RF). La conexión de este sensor está ligado a la comunicación asincrónica (UART) a través de los pines GPIO16 y GPIO17 que son dedicados especialmente al flujo de geolocalización (véase figura 25).

Figura 25

Diagrama de conexión de los sensores



Nota. Autoría propia.

A nivel de firmware y filtrado, el sensor de inclinación o MPU6050 fue fijado rígidamente en la parte del manillar del ciclista que coincide con el centro de gravedad aproximado del sistema, este sensor resulta critico porque permite capturar las fuerzas G y calcular por firmware la magnitud del vector de aceleración que resulta mediante la ecuación:

$$||a|| = \sqrt{a_x{}^2 + a_y{}^2 + a_z{}^2} \qquad (3)$$

En cuanto a la detección de caídas y la inclinación del sensor está conectado mediante I2C detallada en la figura 25 que permite que el sensor se comunique con el nodo mediante los puertos GPIO21 (SDA) y GPIO22 (SCL).

Posteriormente, el sensor ultrasónico HC-SR04 fue orientado hacia atrás y con una visión panorámica con un leve giro a su izquierda, esto por motivo que en Ecuador el carril izquierdo está destinado para que puedan rebasar los vehículos con motor. Para este sensor, se implementó una lógica de acondicionamiento en el ESP32 para descartar anomalías y disminuir el ruido acústico de la vía. Como se observa en la figura 25, el esquema de conexión de este módulo asigna de forma directa los pines GPIO5 y GPIO18 que permite gestionar los pulsos Trigger y Echo.

Sumado a esta depuración por software, se implementó este procesamiento al nodo principal permitiendo mitigar falsas alarmas para que las tramas no saturen el ancho de banda del canal inalámbrico.

3.7.4. *Módulos de comunicación inalámbrica*

En lo que respecta a la capa transporte, el sistema está estructurado mediante una topología hibrida, combinando una red de área local inalámbrica (WLAN) para los datos recibidos del nodo principal y un enlace celular (LTE) para el acceso de red de área amplia (WAN) que permite la salida a internet. Como se observa en la figura 26, el microordenador central actúa como un punto de accesos (AP), lo que permite propagar una red privada bajo el estándar IEEE 802.11 b/g/n.

Debido a las características de propagación, la elección de operar exclusivamente en una radio frecuencia de 2.4GHZ se debe primordialmente a las directrices técnicas del nodo principal, que es la única frecuencia en la que trabaja nativamente este nodo ESP32-Wroom-32, por otra parte, debido a su longitud de onda, la banda de 2.4GHz orece mayor alcance de conectividad en

referencia a la frecuencia de 5GHz, lo que permite que la onda pueda atravesar obstáculos físicos sin sufrir perdidas de propagación de la misma.

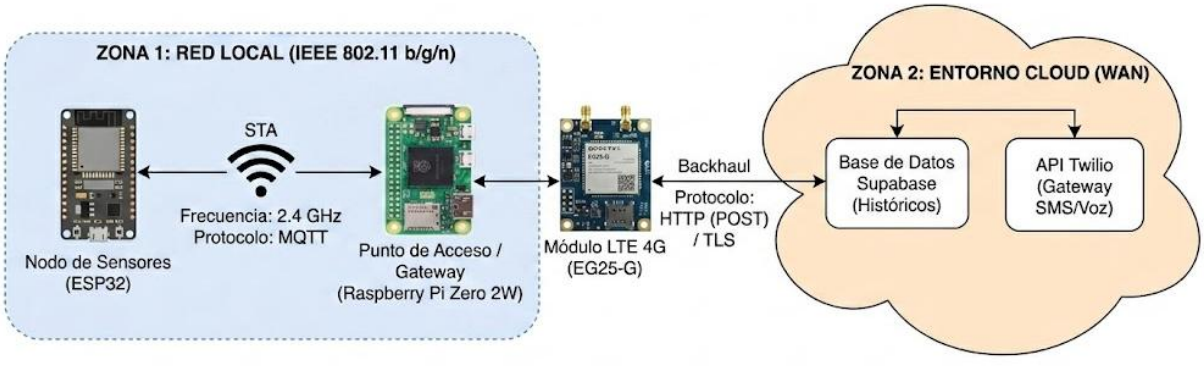
Bajo esta topología de red, al actuar como punto de acceso (AP) el microordenador central permite que el nodo ESP32 se asocie como una estación (STA) estableciendo un canal exclusivo para la transmisión por MQTT. Esta independencia de la red local es de gran ayuda porque permite que el sistema funcione incluso en los entornos adversos como transitar bajo un puente o por rutas de montaña que carecen de señal celular.

Por consiguiente, la transición de datos al entorno WAN actúa a través del módulo LTE, (véase en la figura 26) permitiendo observar cómo los datos escalan a la nube, además, este módulo celular 4G (EG25-G) actúa como el backhaul, que es la parte de la red que permite conectar todos los nodos o puntos intermedios con el núcleo central de la red, que permite realizar un túnel de los registros históricos hacia la base de datos Supabase en la nube.

En definitiva, esta arquitectura hibrida permite a través del enlace LTE despachar las solicitudes HTTP (POST) hacia la API Twilio en caso de que el sistema detecte un accidente o cualquier evento crítico, asegurando el envío de SMS de emergencia.

Figura 26

Topología de red híbrida. Integración red local (WLAN) y celular (WAN).



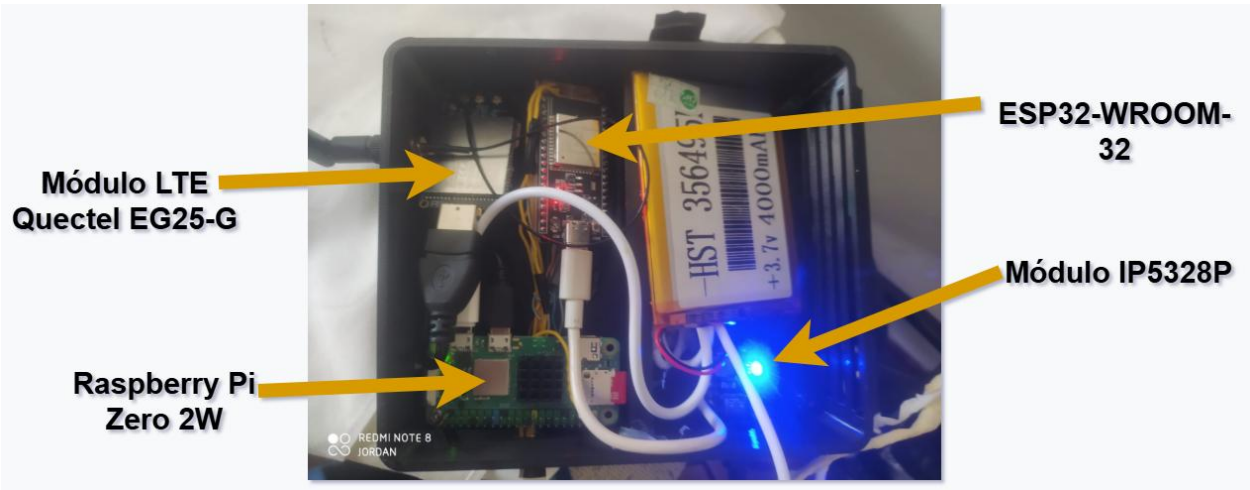
Nota. Autoría propia.

3.7.5. *Implementación física del Sistema Principal*

Para la integración del sistema, fue necesario realizar una estratégica ubicación de las placas base dentro de la carcasa principal. Se ubicó el microordenador (Raspberry Pi Zero 2W) y el microcontrolador (ESP32-WROOM-32) de forma paralela con el fin de reducir la distancia de conexión inalámbrica entre las dos placas, en su lado opuesto al microordenador se ubica el módulo LTE Quectel EG25-G, con el fin de reducir la longitud de cableado de datos con el microordenador, optimizando así el espacio físico y facilitando con ello el acceso a los conectores de energía como se detalla en la figura 27.

Figura 27

Disposición e interconexión interna del sistema



Nota. Autoría propia.

El montaje físico del sistema principal se ubicó debajo de la estructura del asiento de la bicicleta para las respectivas pruebas operativas, de modo que no influyan en el manejo de la misma. Para esto, fue necesario utilizar los soportes mecánicos del chasis y del diseño de la carcasa para obtener una rigidez con la carcasa de manera firme y estable en las barras de la bicicleta, garantizando así el funcionamiento correcto como se observa en la figura 28.

Figura 28

Montaje final del Edge Gateway ensamblado



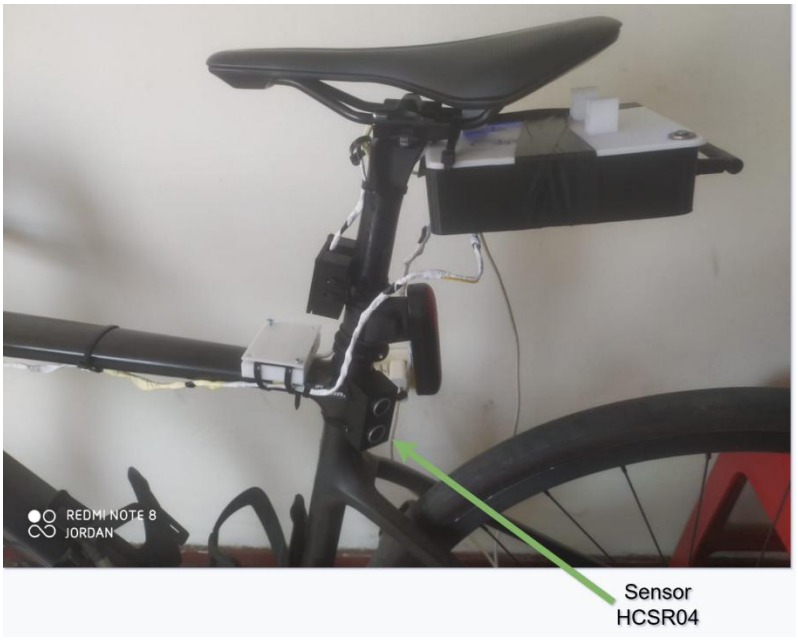
Nota. Autoría propia.

3.7.6. *Distribución e instalación de sensores en la bicicleta*

Para la implementación y ubicación de los sensores de telemetría, fue determinante la posición, ya que en todo momento deben estar estáticos y capturando datos, para ello el sensor de proximidad exigió aislar cualquier barrido de acústico que ocurra con el chasis de la bicicleta. Por ello, fue necesario anclarlo en la sección posterior de la bicicleta, específicamente en la barra que actúa como soporte del asiento y con una vista hacia el lado izquierdo de la misma apuntando hacia la retaguardia, con el fin de garantizar un cono de detección sin interferencias y capaz de medir la aproximación de vehículos en un rango de 1.5 metros como se detalla en la figura 29.

Figura 29

Detalle de la instalación del sensor ultrasónico

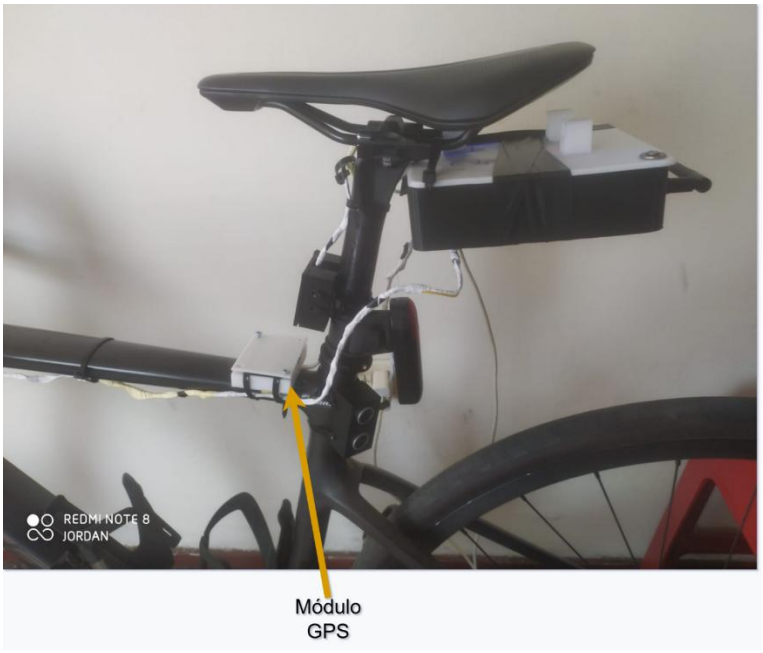


Nota. Autoría propia.

Con respecto a la geolocalización del sistema, el sensor GPS requería una disponibilidad alta de cielo despejado para realizar la triangulación de las coordenadas, para ello, fue necesario ubicarlo en la barra central de conexión entre el asiento y el manubrio, de acuerdo a esta decisión el GPS podrá mantener la línea de vista (LOS) hacia los satélites, reduciendo así la atenuación de la señal de radiofrecuencia (RF) que naturalmente genera una masa, como se observa en la figura 30.

Figura 30

Montaje del módulo GPS, posicionado para garantizar una línea de vista (LOS)

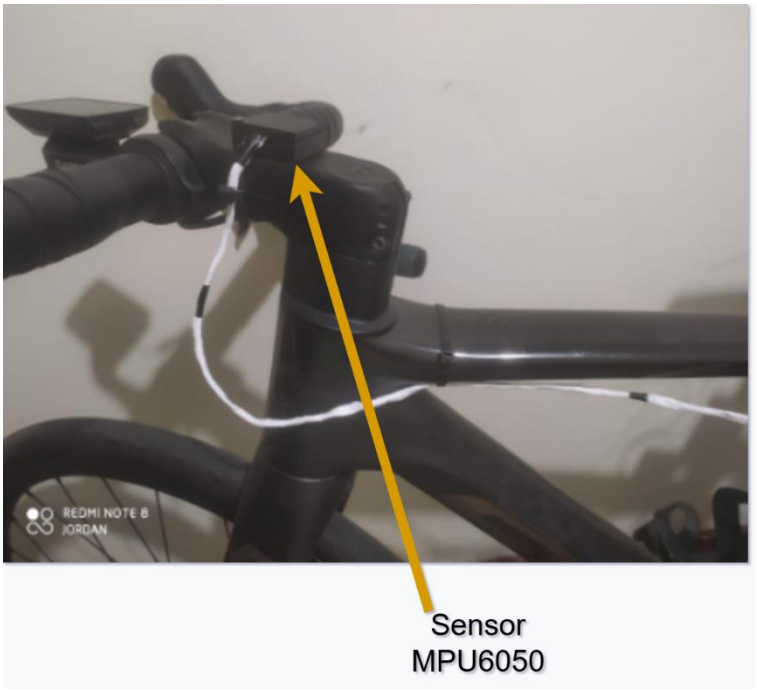


Nota. Autoría propia.

Con respecto a la detección de impactos o volcaduras, el sensor de medición inercial acelerómetro fue fijado sólidamente en el eje horizontal del manubrio. Esta ubicación es óptima, puesto que, el manubrio actúa como la palanca principal de giro de la bicicleta, de acuerdo a esto, cuando exista cualquiera perdida de equilibrio, impacto frontal o inclinación superior al umbral se refleja de manera inmediata en este punto, permitiendo así al sistema reaccionar y procesar la alerta sin latencias mecánicas como se observa en la figura 31.

Figura 31

Montaje del sensor (MPU6050) sobre el eje del manubrio



Nota. Autoría propia.

Finalmente, a nivel de ensamblaje el cableado fue guiado a través de los perfiles del marco de la bicicleta impidiendo cualquier invasión sobre los componentes críticos de la misma. De esta manera la ubicación del hardware a lo largo del chasis preserva un balance natural del vehículo salvaguardando la integridad física y mecánica del ciclista durante sus trayectos (ver figura 32).

Figura 32

Vista panorámica de la bicicleta



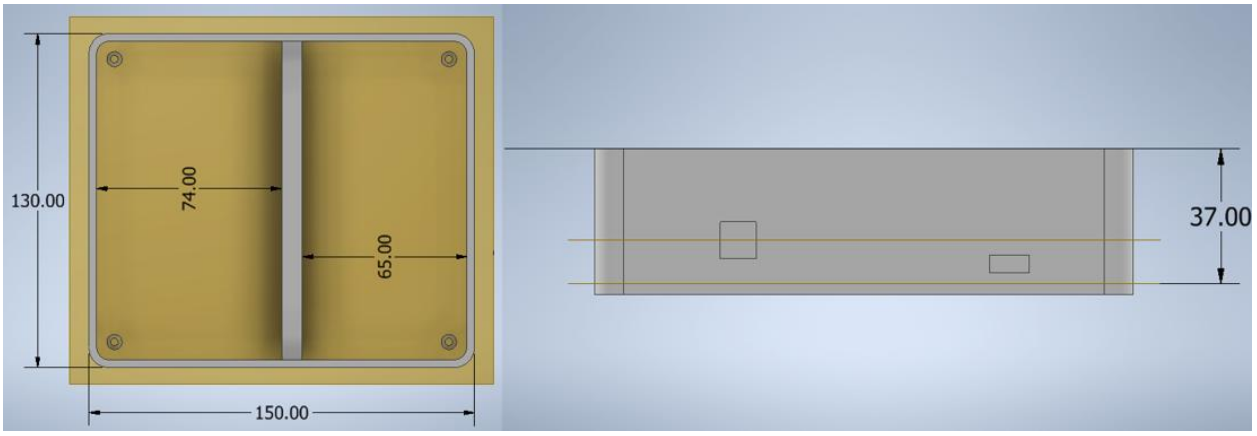
Nota. Autoría propia.

3.7.7. *Ensamblaje y diseño del case*

En cuanto al diseño de la carcasa, las piezas primero fueron moldeadas en bocetos 2D, para luego realizar una extrusión volumétrica 3D solidos utilizando un material fundido (impresión 3D) lo que lo hace resistente y ligero, garantizando que el hardware del sistema tenga una distribución diseñada a medida. Bajo esta estructura, en la figura 33 se puede observar el boceto final de la carcasa principal, que está compuesta por una base y tapa superior moldeadas a su medida, cuyas dimensiones son 150 mm de largo por 130 mm de ancho y una altura de 39 mm, en donde albergan los componentes como el microordenador central, nodo principal, el módulo de comunicación LTE y el circuito de alimentación del sistema.

Figura 33

Diseño y dimensiones de la carcasa principal para el sistema.

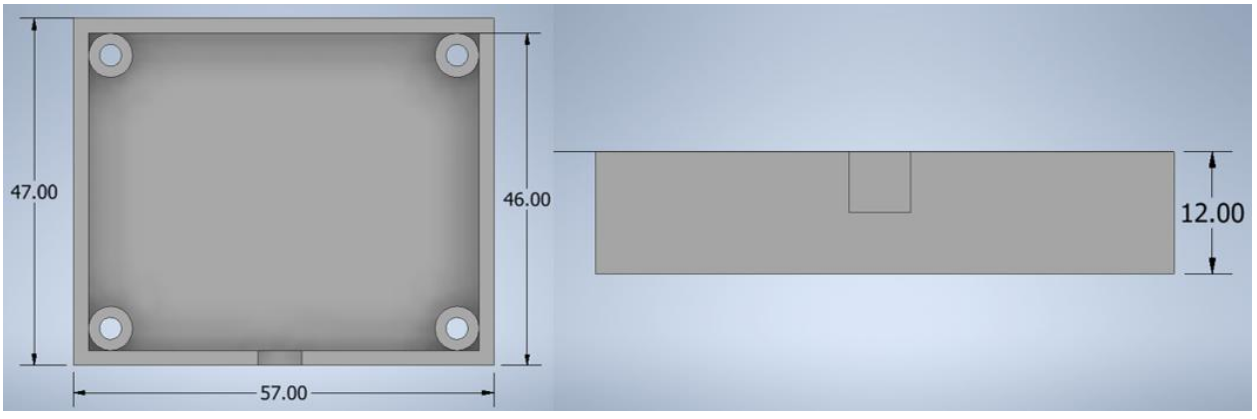


Nota. Autoría propia.

Siguiendo este esquema de distribución, fueron diseñadas las carcasas de los diferentes sensores ubicados en el chasis de la bicicleta con el fin de obtener una durabilidad y trabajo continuo y estable del sistema. De este modo, la carcasa del módulo independiente GPS NEO-6M, fue moldeado en bocetos 2D y posteriormente 3D con unas dimensiones exactas de 57mm de largo por 47mm de ancho y una altura de 12mm, permitiendo así anclar al cuadro del chasis de la bicicleta y sin interferir en el usuario.

Figura 34

Diseño de la carcasa protectora para el módulo GPS NEO-6M.

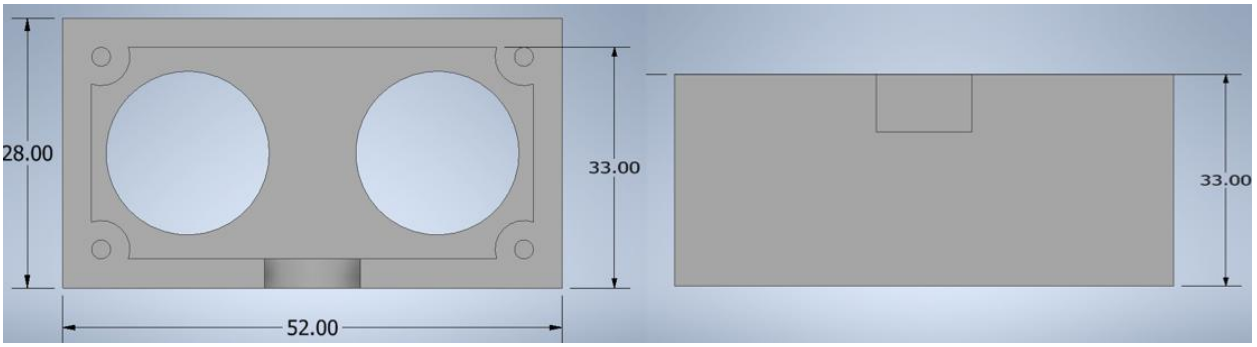


Nota. Autoría propia.

De manera similar, la carcasa del sensor ultrasónico HC-SR04 tiene las siguientes medidas: 52mm de largo por 28mm de ancho y una altura de 33mm, el cual está fijado bajo el asiento de la bicicleta, el cual otorga un ángulo de visión despejado hacia los vehículos que se acercan por la parte trasera.

Figura 35

Diseño y protección para el sensor ultrasónico HC-SR04

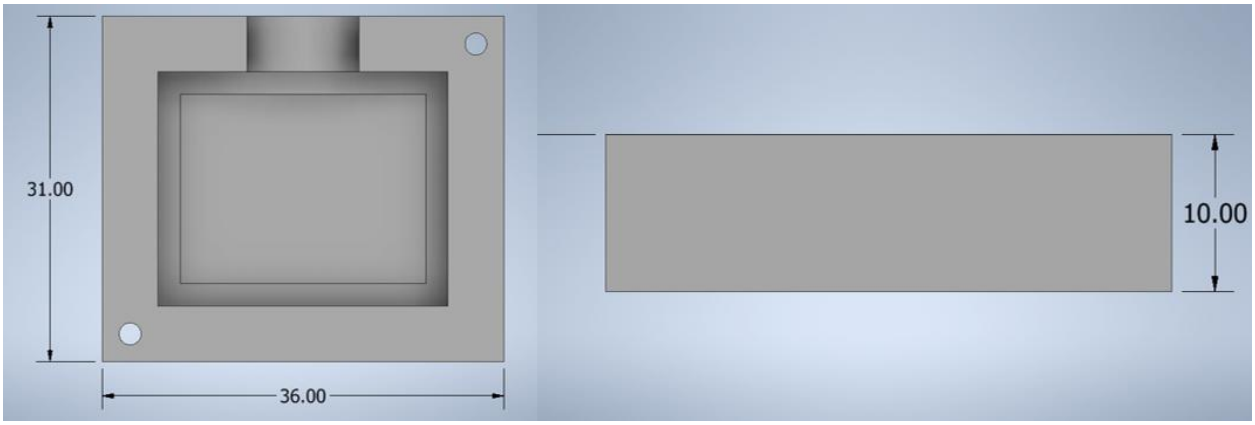


Nota. Autoría propia.

Complementando la red de sensores, el giroscopio MPU6050 se protege por una carcasa compacta de 36mm de largo por 31mm de ancho y una altura de 10mm, el cual está anclado rígidamente al manubrio de la bicicleta para registrar la inclinación.

Figura 36

Diseño compacto para el sensor de inclinación y aceleración MPU6050.



Nota. Autoría propia.

En consecuencia, un factor importante el diseño de la carcasa del sistema es la hermeticidad de las piezas impresas, puesto que, al tratarse de un vehículo de uso exterior, los acoples entre las tapas y bases fue mediante el uso de tornillos y las tapas fueron impresas en un material más denso para soportar el peso del circuito frente a vibraciones y lluvia logrando un acople compacto y seguro del sistema, que permite prolongar la vida útil de los dispositivos en la intemperie.

3.8 Procesamiento y almacenamiento de datos

Para lograr el funcionamiento autónomo del sistema se necesita procesar la información en dos capas: la primera está en los sensores del nodo que esta localmente y la segunda que se encuentra en el Gateway de forma centralizada con lo cual se realiza la conexión a la nube. Para ello se ha implementado una lógica que permite limpiar los datos crudos que llegan al ESP32, con esta lógica se garantiza que solo información valiosa navegue a través de la red local hacia el cerebro del sistema. También, el microordenador fue configurado con un buffer temporal o de almacenamiento temporal que permite con esto seguir registrando datos históricos, aunque la bicicleta este transitando por un entorno en el cual no tenga acceso a la señal celular 4G. con el fin de garantizar la seguridad del usuario, se automatizo la toma de decisiones críticas en el sistema, en donde el Gateway es el encargado de enviar una alerta de tipo SMS de emergencia con el fin de

salvaguardarla integridad del usuario en información confidencial y seguridad física que se lleva a cabo mediante datos históricos guardados que al activarse hacen uso del API de Twilio,

3.8.1. *Programación del microcontrolador*

Para la etapa de adquisición de datos, el nodo principal EPS32-Wroom-32 ejecuta rutinas desarrolladas en el lenguaje C++ mediante el software Arduino Ide, esto con el fin de asegurar un procesamiento de bajo nivel, pero con un tiempo estricto. De este modo, como se observa en la tabla 8, las librerías seleccionadas permiten orquestar las operaciones a nivel de firmware sin comprometer la memoria del dispositivo.

Tabla 8

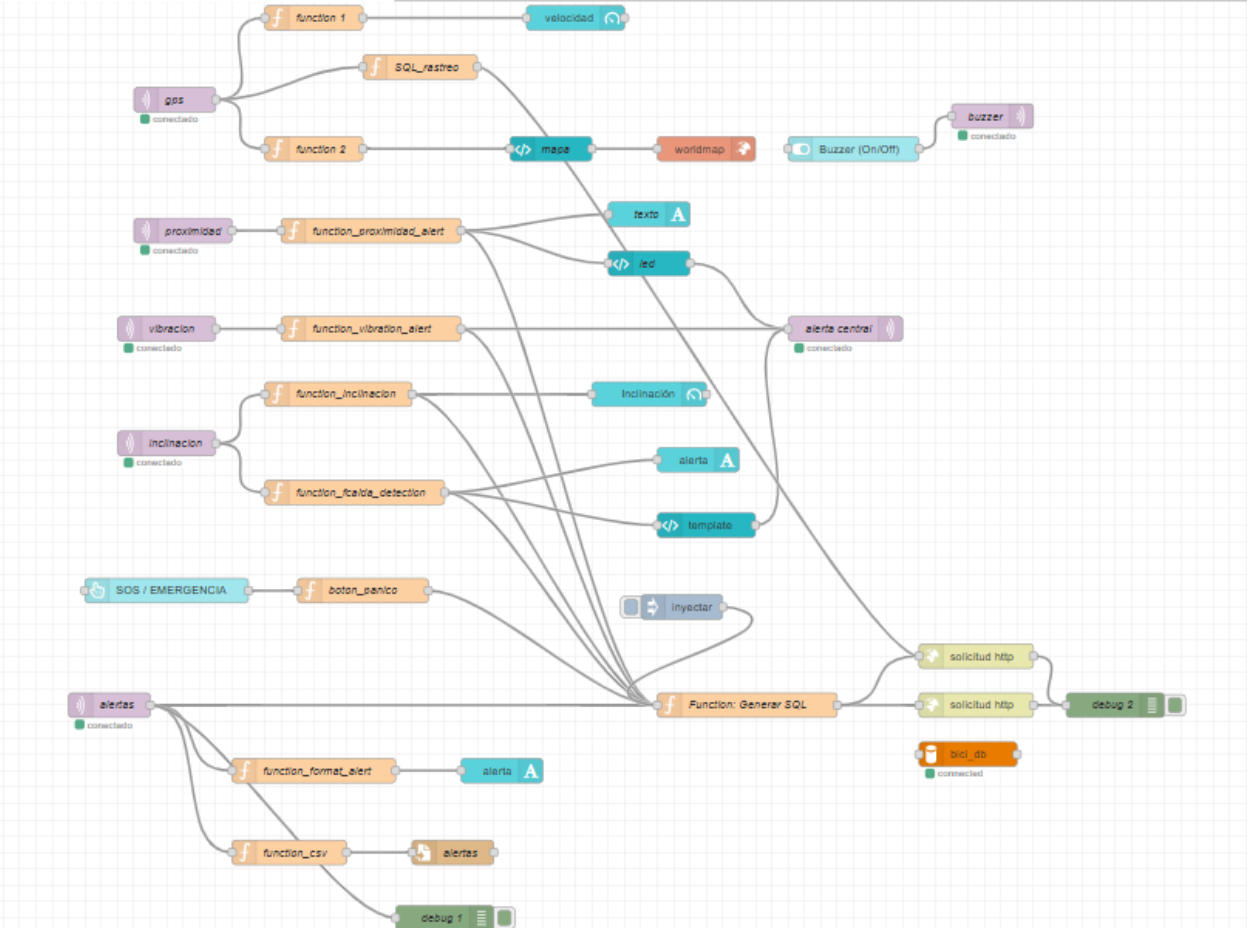
Librerías utilizadas

Nombre de la Librería	Función
PubSubClient.h	Implementación cliente MQTT para publicación.
TinyGPS++.h	Parseo eficiente de las complejas tramas estándar.
Adafruit_MPU6050.h	I2C para lectura y gestión del acelerómetro y giroscopio.
ArduinoJson.h	Serialización dinámica de variables de estado y conformación de estructuras JSON ligeras.

Posteriormente, las tramas son enviadas hacia el Gateway central, en donde el motor de flujos Node-Red realiza tareas de procesamiento, como la depuración de datos nulos. Como se observa, en la figura 37 esta capa intermedia interviene en el enrutamiento de los paquetes, separando del tráfico de datos periódicos de aquellas alertas asíncronas que necesitan atención inmediata.

Figura 37

Flujo Lógico en Node-RED

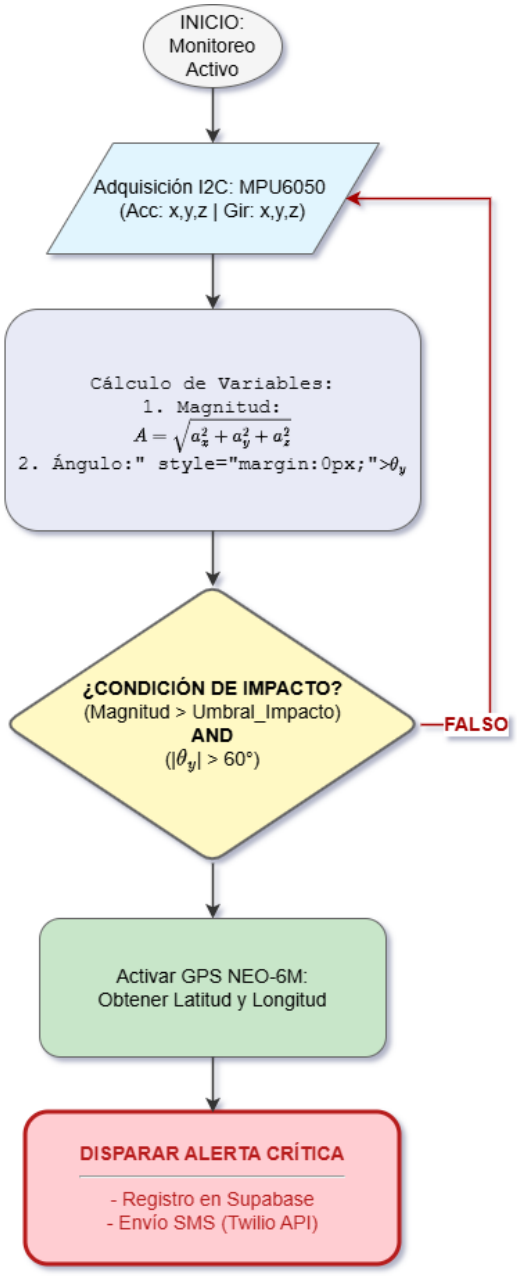


Nota. Autoría propia.

En la lógica de la seguridad activa, la validación de accidentes no depende solo de un simple interruptor, el sistema verifica las métricas enviadas calculando mediante la ecuación (3) antes mencionada, la implementación de esta fórmula matemática en el código, permite al algoritmo diferenciar analíticamente entre un frenado brusco o natural de un impacto contra el asfalto. En la figura 38 se observa cómo funciona la lógica en el código mediante un flujo para diferenciar los eventos mencionados demostrando como el pico vectorial evalúa de manera concurrente una desviación de inclinación superior a los 60 grados.

Figura 38

Algoritmo de Detección de Caídas



Nota. Autoría propia.

Después de realizar el procesamiento, el sistema realiza una depuración y clasificación de información con el objetivo de diferenciar los eventos de registros periódicos como de alertas asíncronas, además los registros históricos son almacenados en la base de datos Supabase alojada en la nube, de acuerdo a esto, los datos son estructurados como se ve en la tabla 9 que permite organizar de forma eficiente el rastreo espacial, las variables crudas de alertas y la bitácora de

emergencias, asegurando así las constantes consultas que realiza la aplicación móvil, lo que ayuda a garantizar la persistencia integra de los registros y cumplimiento de los estándares.

Tabla 9

Diccionario de Datos del Backend (Supabase)

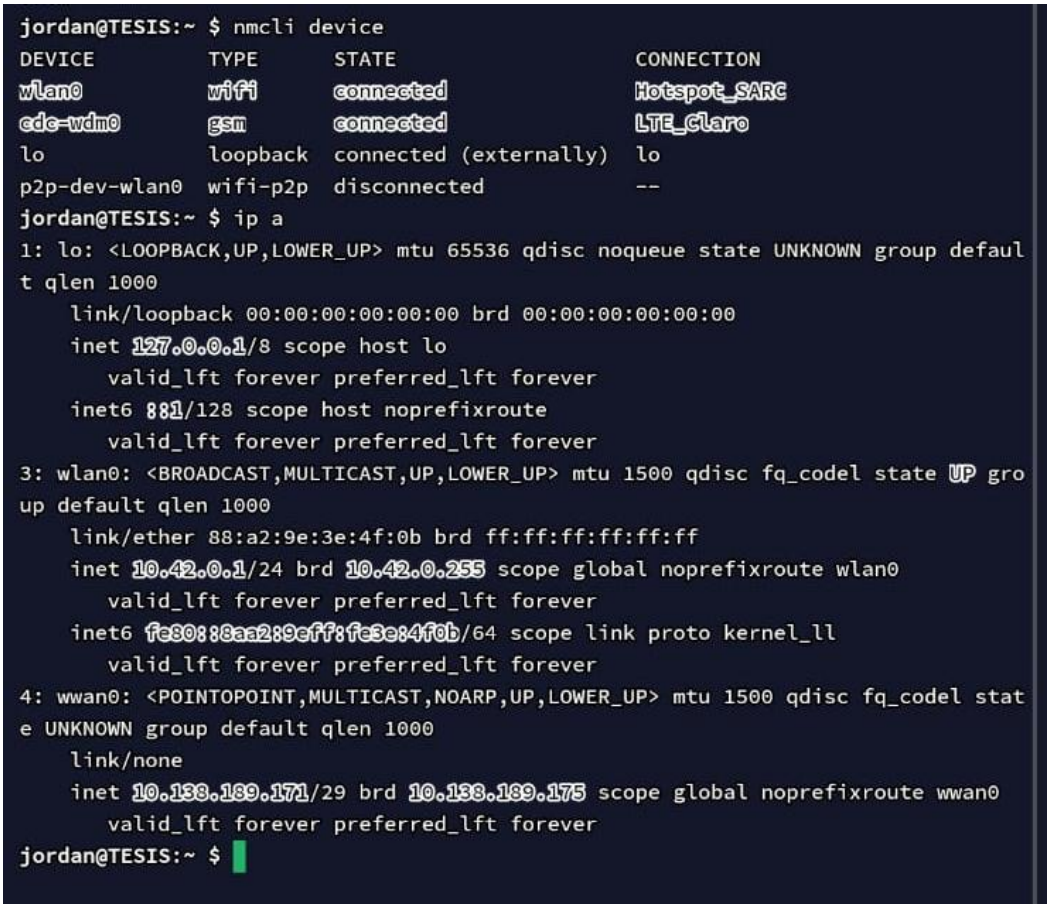
Nombre Tabla	Campos principales (atributos	Tipo dato (SQL)
telemetria_cruda	id_registro, timestamp, distancia_cm, magnitud_g	uuid, timestamptz, float4, float4
rastreo_gps	id_ruta, timestamp, latitud, longitud, velocidad_kmh	uuid, timestamptz, float8, float8, float4
eventos_alerta	id_alerta, timestamp, tipo (Caída/Proximidad), sms_enviado	uuid, timestamptz, text, boolean

3.8.2. *Conexión con la nube*

Para consolidad la salida a internet, el sistema trabaja mediante conexiones Wi-Fi externas para permitir al microordenador central navegar de manera absoluta, esto mediante, el establecimiento del enlace celular a través del módulo LTE que está gestionado por NetworkManager. Como se observa en la figura 39, el establecimiento del protocolo punto a punto (PPP) ayuda a levantar una interfaz virtual de la red denominada wlan0, la cual está actuando como el backhaul de internet del Gateway.

Figura 39

Captura de la terminal Linux (consola)



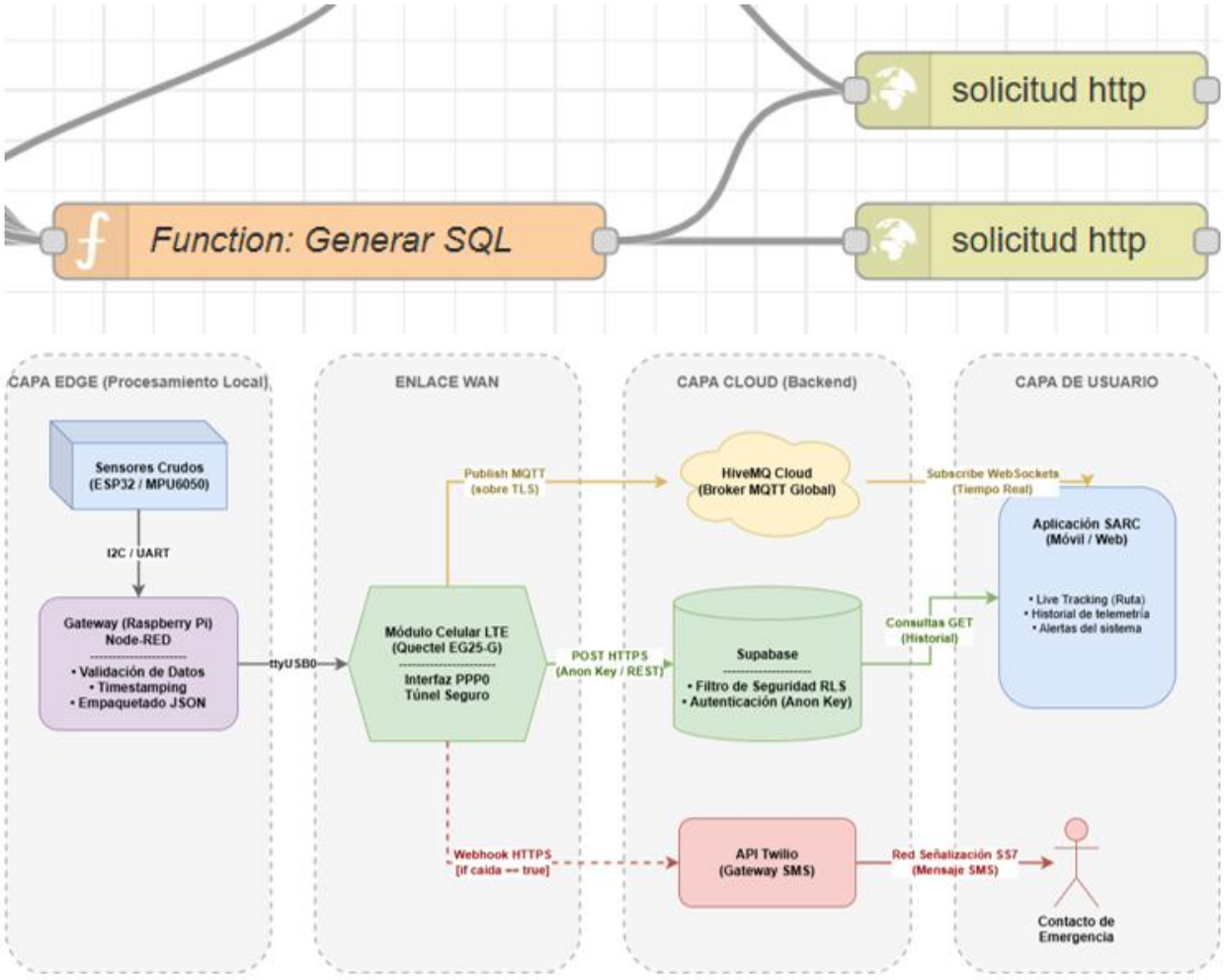
Nota. Autoría propia.

Para garantizar la conexión constante en trayectos largos del microordenador, en esta interfaz se encuentra un mantenimiento respaldado por rutinas de systemd, en donde permite forzar el reinicio y la reconexión automática del módulo LTE ante cualquier perdida temporal de cobertura celular por la geografía de la ruta. Después de asegurar el enlace de capa física y transporte, la comunicación del backend hacia Supabase se ejecuta exclusivamente mediante la solicitud de peticiones RESTful que actúan sobre el protocolo HTTPS.

El enrutamiento de esta comunicación esta evidenciada en la figura 40, en la cual hace uso específico del nodo node-red-contrib-supabase el cual ayuda a procesar de forma eficiente las transacciones de forma asíncrona.

Figura 40

Diagrama de Node-RED y Flujo Cloud (nube)

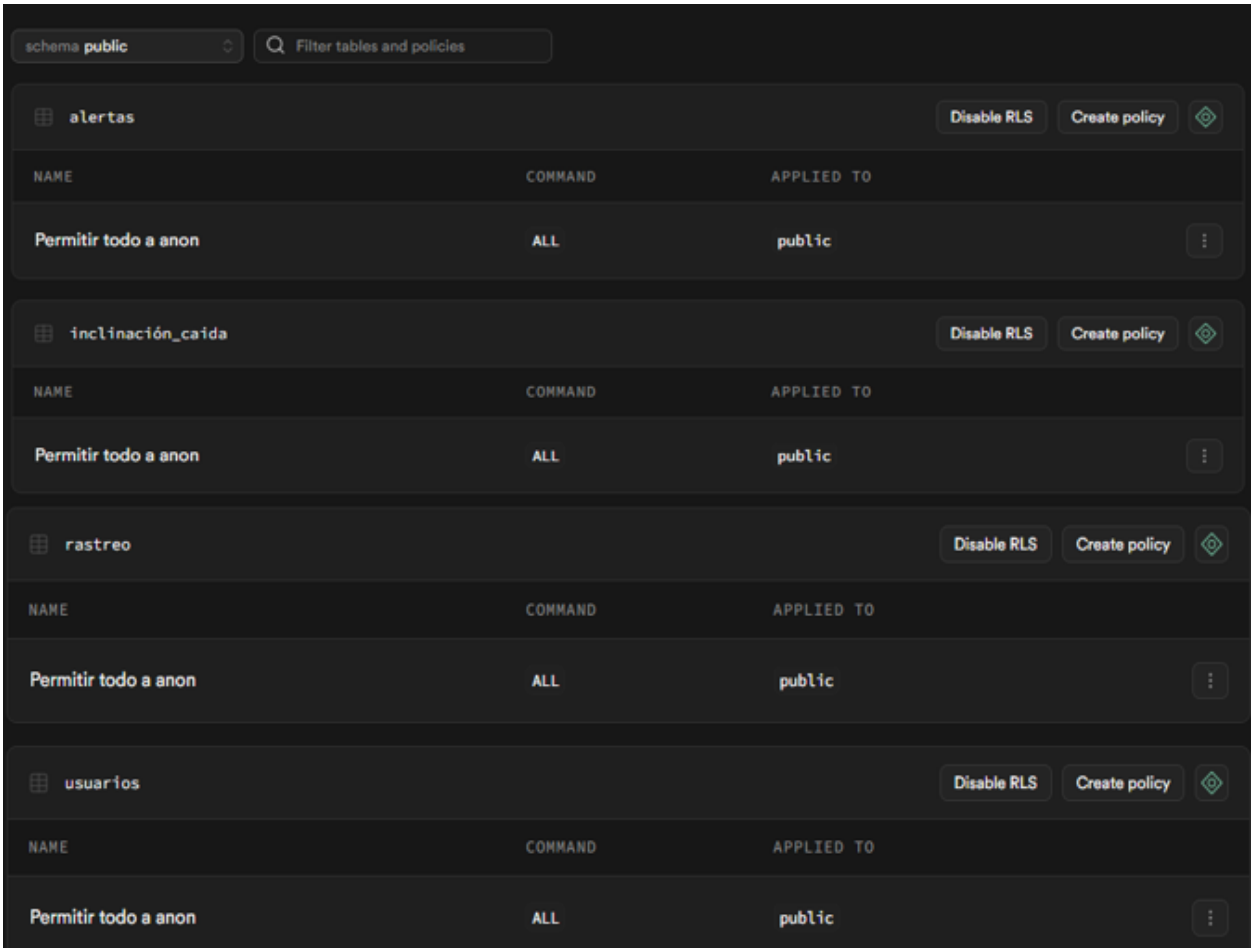


Nota. Autoría propia.

En torno a la ciberseguridad y la protección de los registros, la base de datos rechaza por defecto cualquier transacción publica, esto se realiza mediante políticas de seguridad en el gesto de Supabase denominadas *Row Level Security* (RLS), ilustradas en la figura 41.

Figura 41

DProtección Row Level Security (RLS) en Supabase



Nota. Autoría propia

En virtud a la seguridad implementada, el Gateway inyecta Anon Keys que son una clave publica de la API del gestor de datos, estas claves van autenticadas en las cabeceras HTTP de cada paquete que este saliendo, esto ayuda a certificar que cada paquete viene estrictamente de la bicicleta del usuario.

Finalmente, para las alertas criticas ante acidas, se optó por utilizar una API de Twilio, que envían mensajería de texto corto (SMS) a los contactos de emergencia, en la tabla 10 se resumen los parámetros técnicos que justifican la utilización de protocolos de SMS ante el servicio de mensajería IP (WhatsApp) en escenarios de emergencia.

Tabla 10

Comparativa técnica

Características	Protocolo SMS	Mensajería IP
Capa Red	Canal de Señalización (SS7)	Canal de Datos (TCP/IP).
Disponibilidad zonas rurales	Muy Alta (solo requiere señal 2G/voz)	Baja (requiere 3G/4G estable).
Dependencia del Receptor	Nativo en cualquier celular	Requiere instalación de terceros y plan de datos activo.

Esta elección se basa primordialmente en el canal de señalización que utiliza la mensajería SMS, lo que asegura la entrega de del mensaje de auxilio incluso en zonas rurales donde la conexión de datos es demasiado débil.

3.8.3. *Validación de los datos capturados*

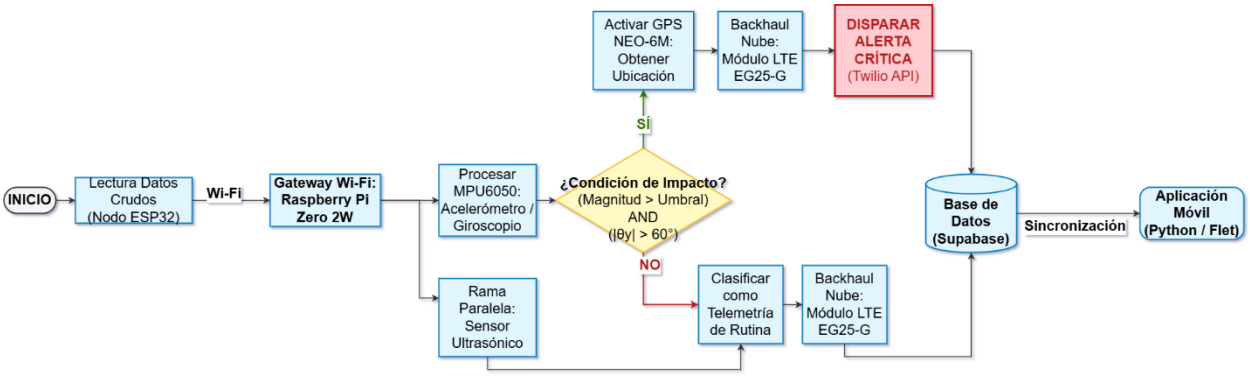
Con el fin de eliminar valores de datos erróneos, el sistema incorpora rutinas de curación de datos con el fin de descartar el ruido electromagnético, en el caso del sensor de ultrasonido, las medidas crudas, las cuales están sometidas a un filtro digital de promedio móvil y acotadas a una ventana espacial estrictamente acotada considerando únicamente los ecos de retorno como puntos validos que representan distancias entre 0.1m y los 1.5m.

En la figura 42, se puede observar el comportamiento sobre la magnitud vectorial que permite identificar los picos de fuerza G, pero siempre que sean suficientes como para dictaminar una caída, por esta razón, el algoritmo confirmará que hubo una inclinación pronunciada solo cuando el giroscopio reporta una inclinación (tilt9) sostenida superior a los 60 grados con respecto

a su eje vertical, distinguido así un frenado de un impacto. De esta manera la validación de datos permite que el sistema garantice la precisión de monitoreo y reduciendo errores que sucedan dentro del sistema, asegurando que solo los datos sean almacenados y visualizados fortaleciendo así los resultados que se reflejan en entornos reales del entorno en que se monitorea.

Figura 42

Diagrama lógico de validación



Nota. Autoría propia

En cuanto a la consistencia del posicionamiento global, antes de realizar el envío de alerta de auxilio el firmware del microcontrolador verifica que las coordenadas enviadas no sean nulas (0,0), adicionalmente el sistema evalúa que el índice de precisión horizontal (HDOP) del módulo GP se encuentre dentro de márgenes de tolerancia aceptable, con esto se garantiza que el punto de rescate sea exacto.

En consecuencia, en la tabla 11 se representa los umbrales críticos establecidos para la activación de alertas y las reglas de descarte determinadas en el algoritmo que se encuentra en el código fuente, lo que se traduce en una reducción de falsas alarmas.

Tabla 11

Resumen de Parámetros de Validación Crítica.

Variables	Rango operativo	Condición alerta
Radar Ultrasónico (HC-SR04)	0.1 m a 1.5 m	Distancia < 1.0m (Proximidad Peligrosa)
Acelerómetro (MPU6050)	> 0 G	Magnitud Resultante > Umbral de Impacto
Giroscopio (MPU6050)	-90° a +90°	Inclinación Absoluta > 60°
Receptor GNSS (NEO-6M)	HDOP < 2.0	N/A (Registro continuo)

3.9 Desarrollo de la aplicación móvil

En esta sección se presenta el diseño, lógica y desarrollo de la ampliación nativa móvil, detallando herramientas utilizadas y su funcionalidad de cada una. También, se describe la arquitectura del sistema, la lógica del algoritmo y las principales acciones que realiza el sistema, con el fin de garantizar una aplicación funcional, estable y escalable para futuras correcciones.

La implementación se llevó a cabo en base al framework Flet basado en la lógica de Python que permite la creación de aplicaciones móviles con un motor de renderización de gráficos y con el bajo consumo de la memoria en móviles. Esta tecnología se escogió debido a la compatibilidad de librerías, su flexibilidad y su gran eficiencia de rendimiento en entornos móviles en cualquier sistema operativo de estas tecnologías.

A continuación, se detallan las fases claves implementadas en el desarrollo de la aplicación incluyendo el rol jerárquico de los usuarios, interfaz interactiva de los usuarios y la implementación de la lógica conjuntamente con la integración de los servicios externos en la nube sea estos para almacenar o realizar acciones ante emergencias.

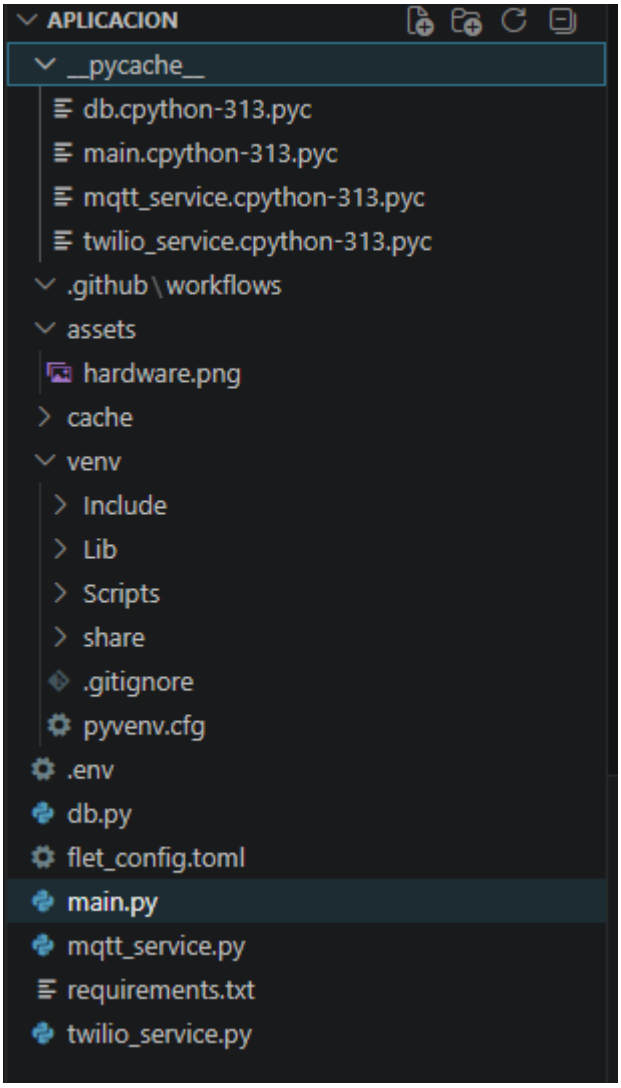
3.9.1. *Estructura de la aplicación móvil*

En cuanto a la arquitectura lógica de la aplicación móvil del sistema, fue construida mediante el framework Flet, puesto que, este framework ayuda a aprovechar la máxima potencia de renderizado de Fluter bajo la lógica y configuración de Python con esto se logra llegar a un rendimiento alto tanto en gráficos como un eficiente empaquetado nativo en formato apk. Para garantizar la escalabilidad del código a correcciones futuras, el sistema se le adopto a una arquitectura de tres capas denominada 3-Tier, en la cual, ayuda a dividir todas las responsabilidades operativas sin cargar en un solo archivo toda la lógica de la aplicación.

Bajo este enfoque, la estructura y organización de los archivos creados para la aplicación nativa de un móvil se detalla en la figura 43 en la que demuestra los directorios organizados, esto se realizó para tener separación física de la lógica entre la interfaz del usuario y los servicios backend.

Figura 43

Estructura del Proyecto SARC

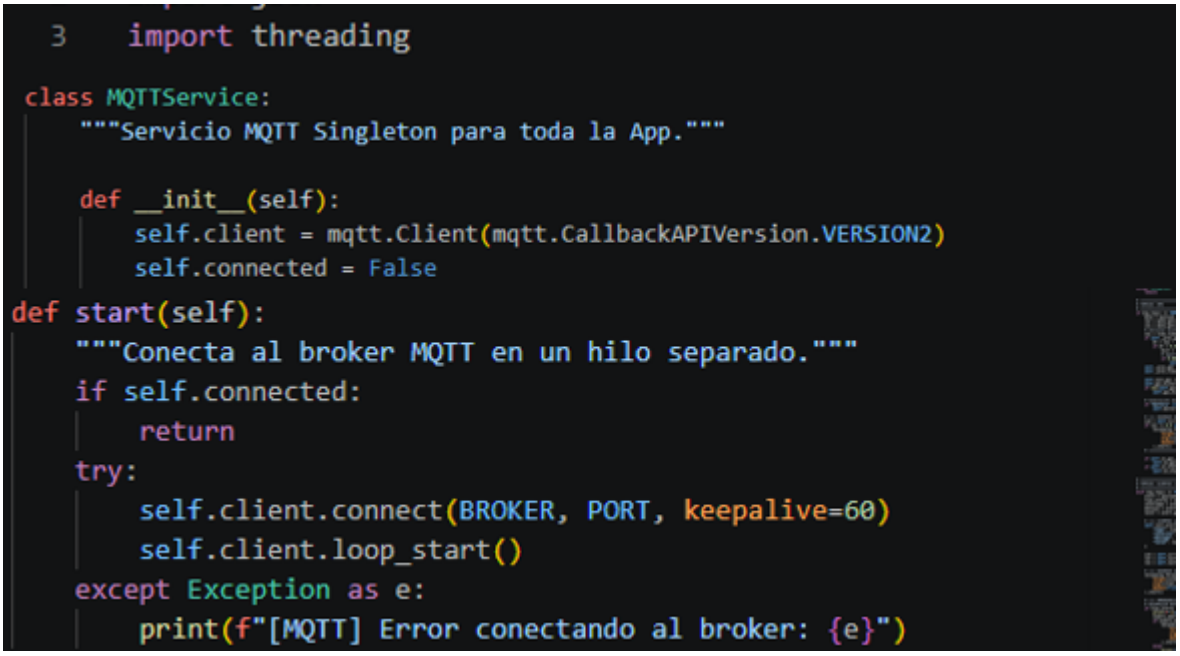


Nota. Autoría propia

En la capa de datos, gira en torno al script db.py, en donde se gestionan las transacciones hacia PostgreSQL mediante la utilización de un controlador puro pg8000, el cual ayuda a, absorber la complejidad de la comunicación asíncrona con el servidor en la nube Supabase. Como se detalla el fragmento de código en la figura 34, la capa de control que está regida bajo el script mqtt_service.py que tiene como fin receptar continuamente la telemetría hacia el módulo, en donde se implementa concurrencia mediante la ayuda de hilos de ejecución (Threads).

Figura 44

Implementación de Concurrencia MQTT



```
3   import threading

class MQTTService:
    """Servicio MQTT Singleton para toda la App."""

    def __init__(self):
        self.client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
        self.connected = False
def start(self):
    """Conecta al broker MQTT en un hilo separado."""
    if self.connected:
        return
    try:
        self.client.connect(BROKER, PORT, keepalive=60)
        self.client.loop_start()
    except Exception as e:
        print(f"[MQTT] Error conectando al broker: {e}")
```

Nota. Autoría propia

El manejo de esta concurrencia es de vital ayuda puesto que, permite al sistema ráfagas de intensas de datos del GPS en segundo plano sin que la aplicación se congele, adicionalmente, en esta capa alberga al script twilio_service.py, en donde aísla las peticiones HTPP para emergencias de las tareas de la aplicación.

En lo que respecta a la capa presentación, el script main.py es el que actúa como el centro maestro, implementando enrutamiento dinámico que es capaz de reaccionar según la jerarquía del usuario que inicia sesión. Tal como se detalla en la figura 45, la lógica de este script permite tener presentaciones diferentes para cada rol, desplegando una interfaz activa para gestión para el usuario “Administrador”, un entorno de gestión activa para el usuario “Ciclista” y un panel de seguimiento solo de lectura para el usuario “Familiar”.

Figura 45

Lógica de Enrutamiento por Roles

```
def on_ok(user):
    """Manejador de enrutamiento exitoso tras login."""
    page.clean()
    # Diccionario de enrutamiento por roles
    v = {"ciclista": build_ciclista_view, "familiar": build_familiar_view, "admin": build_admin_view}.get(user.get("rol"))
     # Renderiza la vista correspondiente al rol o muestra error
    page.add(v(page, user, on_logout) if v else ft.Text(f"Rol desconocido: {user.get('rol')}", color=RED))
    page.update()
```

Nota. Autoría propia

Finalmente, las dependencias que sostienen esta arquitectura, que se ven resumidas en la tabla 12, fueron seleccionadas bajo criterios estrictos de seguridad, y un factor importante como el consumo bajo de memoria para dispositivos móviles. Asegurando una experiencia para el usuario de manera interrumpida.

Tabla 12

Mapeo de Funcionalidades y Componentes de Interfaz.

Librería	Función Principal	Desempeño
flet	Motor de Interfaz (Capa de Presentación)	Permite una visualización fluida de alertas sin saturar la RAM del teléfono.
paho-mqtt	Conectividad de Borde (Capa de Control)	Garantiza la recepción de la telemetría con baja latencia (QoS 0/1).
flet-map / folium	Renderizado Cartográfico	Proporciona geolocalización visual exacta en caso de operaciones de rescate.

pg8000	Controlador PostgreSQL (Capa de Datos)	Asegura transacciones encriptadas y directas hacia Supabase sin fugas de datos.
twilio	Integración de API SMS	Facilita la comunicación redundante hacia familiares mediante la red celular SS7.

3.9.2. *Funcionalidades*

En el ámbito funcional, la aplicación sale de lo convencional de recolectar simplemente los datos para llegar a convertirse en un centro de comando interactivo y fácil de manejar para los usuarios, priorizando la visualización para no llegar a distraer al usuario en el recorrido.

De esta manera, la aplicación maneja diferentes paneles de control de acuerdo al rol de jerarquía, en donde pueden realizar las acciones estipuladas ante o en su defecto solo llegar a consultar acciones determinadas por el rol, esto se llevó a cabo, para no tener conflicto con la parte del familiar al realizar funciones que puedan llegar a distraer o en otro caso perder la conexión del sistema y vulnerar la seguridad del ciclista.

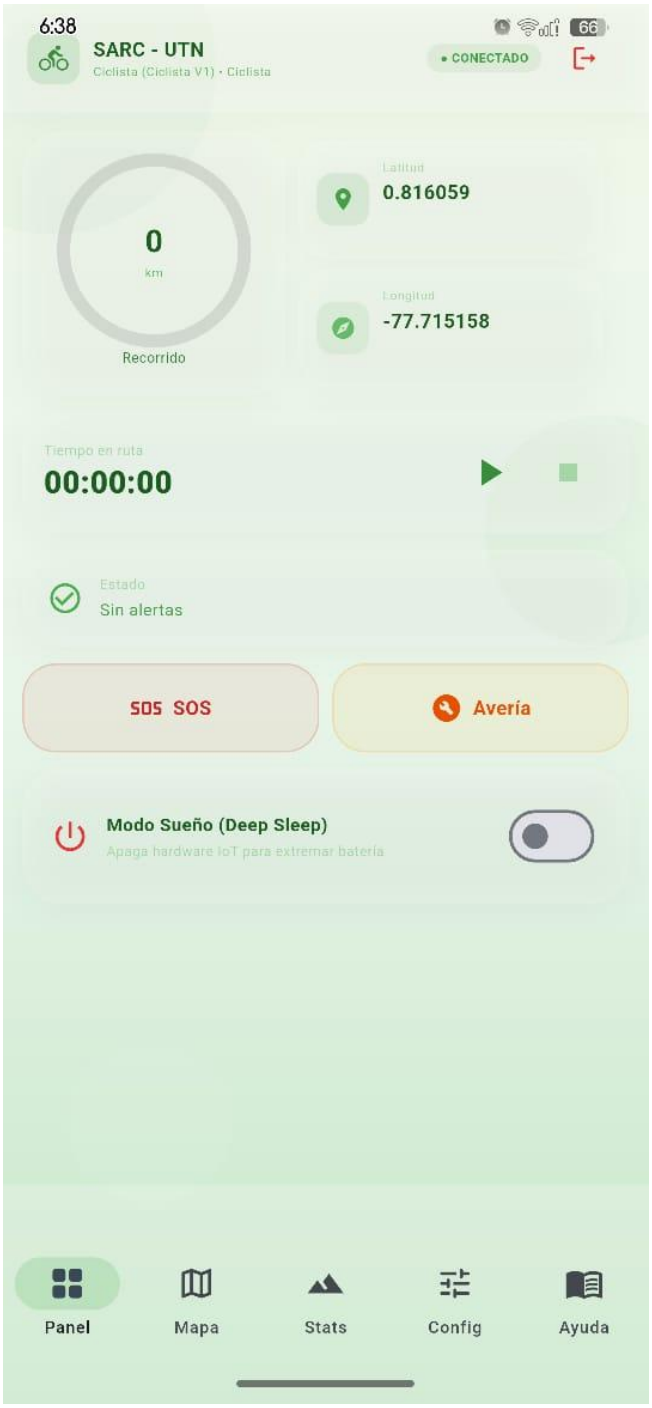
3.9.2.1 Panel principal

En el diseño de la interfaz central, la cabecera de la aplicación asume un rol conocido como Watchdog visual, en el cual, actúa como un monitor de estado M2M (Machine-to-Machine) el cual permite auditar constantemente la salud del encalce que no sea cortado o desconectado.

Como se puede evidenciar en la figura 46, en la interfaz el sistema despliega un indicador de color verde en el cual indica el estado del sistema con una leyenda “CONECTADO”, esto se lleva a cabo cuando la conexión a la suscripción MQTT mantiene un flujo de datos estable.

Figura 46

Vista Principal (Dashboard) del SARC

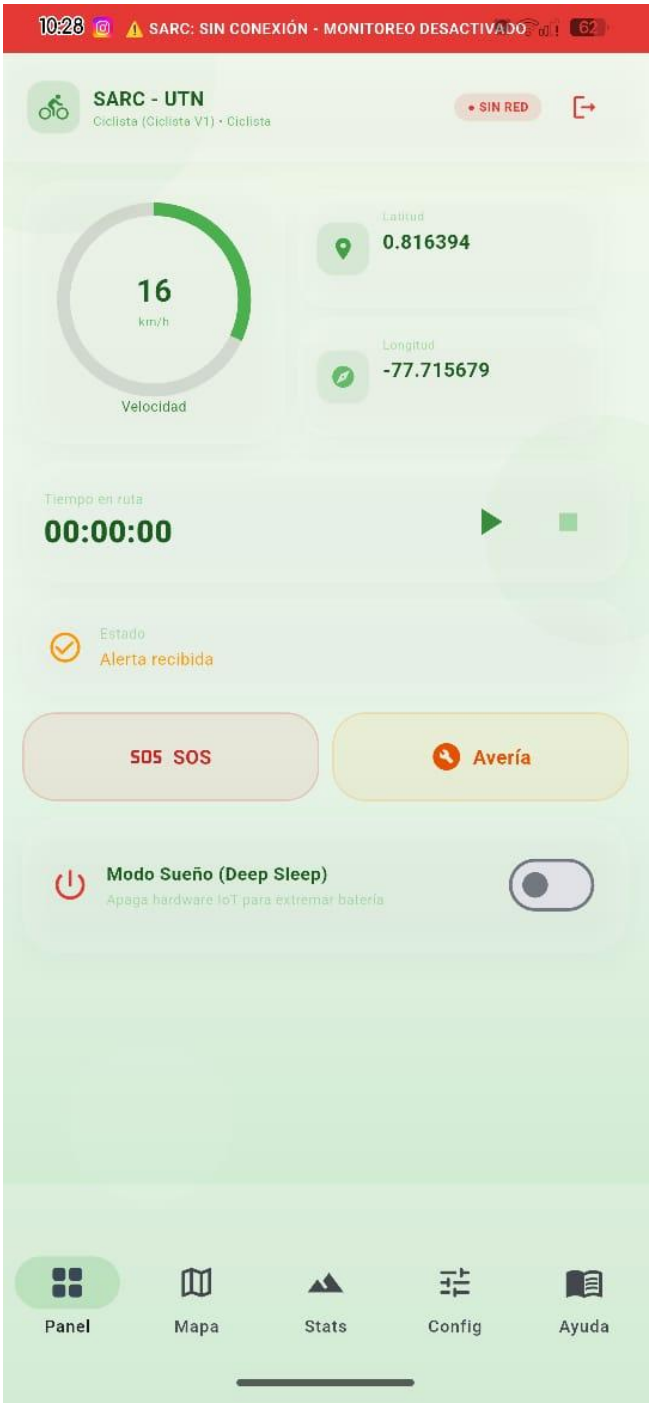


Nota. Autoría propia

En el caso adverso de que el corazón del sistema tenga una interrupción con la suscripción al MQTT, la interfaz del sistema reacciona de manera asíncrona inyectando una advertencia de color rojo con una leyenda “SIN RED”, este comportamiento de dicha alerta fue capturada en la figura 41. Además, para mitigar la carga cognitiva, se hizo uso de componentes tipo Gauge como se observa en la figura 47, permite reducir esta carga hacia el ciclista, permitiendo así verificar estos datos simplemente de un vistazo.

Figura 47

Vista Principal, comportamiento a desconexión de red



Nota. Autoría propia

Para le manejo de alertas o crisis, en la interfaz se implementa una lógica denominada Standard One-Tap mediante la utilización de botones denominados SOS y Avería Mecánica, los cuales están diseñados para actuar bajo condiciones de estrés o guantes de ciclismo. Al accionar

cualquier de los dos botones el sistema captura los datos de las coordenadas satelitales en la memoria cache y dispara inmediatamente el servicio de Twilio, con el fin de acortar la ventana de tiempo para realizar la emisión de la señal socorro.

Además, la aplicación esta diseña para poder realizar una comunicación bidireccional Full-Duplex, esto lo realiza el usuario mediante la inyección de un payload a MQTT hacia el ESP32 lo que permite entrar en modo Deep Sleep a este nodo. Algunos componentes funcionales están detallados en la tabla 13.

Tabla 13

Componentes Funcionales

Elemento	Función Principal	Beneficio
Header (Watchdog)	Monitoreo del *Heartbeat* MQTT.	Provee certeza absoluta sobre la operatividad del sistema de alerta.
Progress Ring (Odómetro)	Renderizado parcial de métricas numéricas.	Minimiza la carga cognitiva y el tiempo de distracción visual en ruta.
Botón (One-Tap)	Disparador de peticiones HTTP (Twilio API).	Facilita la petición de auxilio bajo escenarios de movilidad limitada o estrés.
Switch Deep Sleep	Transmisión *Full-Duplex* de control.	Permite gestionar remotamente el ahorro energético del hardware.

Finalmente, la lógica de actualización asíncrona, se puede observar en la figura 48 cuyo fragmento de código permite entender que al asegurar los callbacks del protocolo puedan modificar solamente los fragmentos afectados, esto se lleva a cabo utilizando la función page.update(), aumentando así el rendimiento que supondría refrescar la pantalla en su totalidad. Maximizando la capacidad de respuesta ante siniestros reales.

Figura 48

Lógica de actualización asíncrona del sistema de telemetría.

```
def on_gps(d):
    #Callback asíncrono: Actualiza solo widgets afectados por telemetría GPS
    cur[0], cur[1] = d['lat'], d['lon']
    # Modificación exclusiva de fragmentos UI específicos
    lat_w.value = f"{d['lat']:.6f}"; lon_w.value = f"{d['lon']:.6f}"
    map_info.value = f"📍 {d['lat']:.6f}, {d['lon']:.6f}"
    marker.coordinates = fm.MapLatitudeLongitude(d['lat'], d['lon'])
    dist = d.get("dist_km", 0.0)
    speed_w.value = f"{dist:.1f}"
    speed_ring.value = min(dist / 20.0, 1.0) # Llenar el anillo basandose en 20km
    # Refresco parcial del lienzo para optimizar rendimiento
    try: page.update()
    except: pass

def on_al(a):
    #Callback asíncrono: Actualiza widgets ante siniestros/alertas
    t = a.get("type", "")
    # Mutación local de iconos y textos
    alert_ic.name = ALERT_ICONS.get(t, ft.Icons.INFO_OUTLINE)
    alert_ic.color = ALERT_COLORS.get(t, MINT_500)
    alert_tx.value = a.get("message", "")
    alert_tx.color = ALERT_COLORS.get(t, MINT_600)
    # Refresco parcial
    try: page.update()
    except: pass
# Asignación de callbacks al protocolo de comunicación
mqtt_service.on_gps_update = on_gps
mqtt_service.on_alert_update = on_al
mqtt_service.start()
```

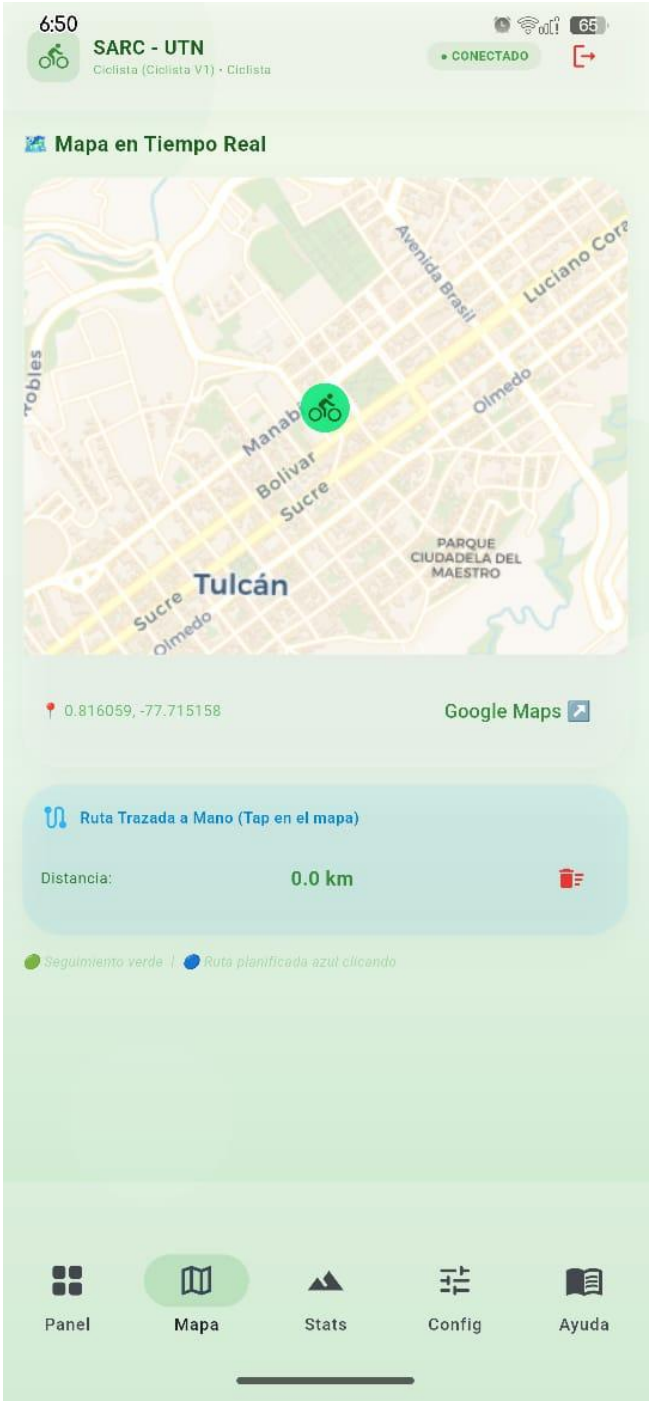
Nota. Autoría propia

3.9.2.1 Visualización de mapa y ubicación GPS

Ara el trazado de las rutas, la aplicación integra en su lógica el motor de trazado flet_map, este componente se utilizó debido a su minimalismo visual y por ende su bajo consumo de datos móviles por el descarte de capas satelitales densas. Como se visualiza en la figura 49, el marcador del ciclista se proyecta sobre un mapa de contraste alto que facilita la lectura bajo la exposición de la luz.

Figura 49

Interfaz de Monitoreo Cartográfico



Nota. Autoría propia

Para garantizar la confiabilidad de los datos espaciales enviados desde el sensor GPS y reducir el efecto de deriva (GPS drift), se hace uso de una fórmula matemática que se utiliza como un filtro de estabilidad, el cual es de vital importancia sobre todo en los entornos urbanos, puesto

que, al ser una zona concurrida y con obstrucción de señal o generar rebotes falsos que pueden dar el posicionamiento erróneo del ciclista como estar dentro de un edificio o fuera de la calzada.

Para mitigar está perdida de datos y haciendo uso del algoritmo de Haversine (véase en la figura 50), descarta los saltos físicamente imposibles y además omite actualizaciones a menos de 5 metros, con el fin de evaluar la distancia real mediante el uso de la ecuación:

$$d = 2r \arcsin\left(\sqrt{sin^2\left(\frac{\Delta\phi}{2}\right) + cos\phi_1 cos\phi_2 sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \quad (4)$$

Donde:

$d = distancia\ geodesica$

$r = radio\ medio\ de\ la\ tierra$

$\Delta\phi = \ delta\ de\ la\ latitud$

$\Delta\lambda = delta\ de\ longitud$

$\phi_{1,}\phi_2 = latitudes$

Figura 50

Implementación del Filtro Haversine en Python

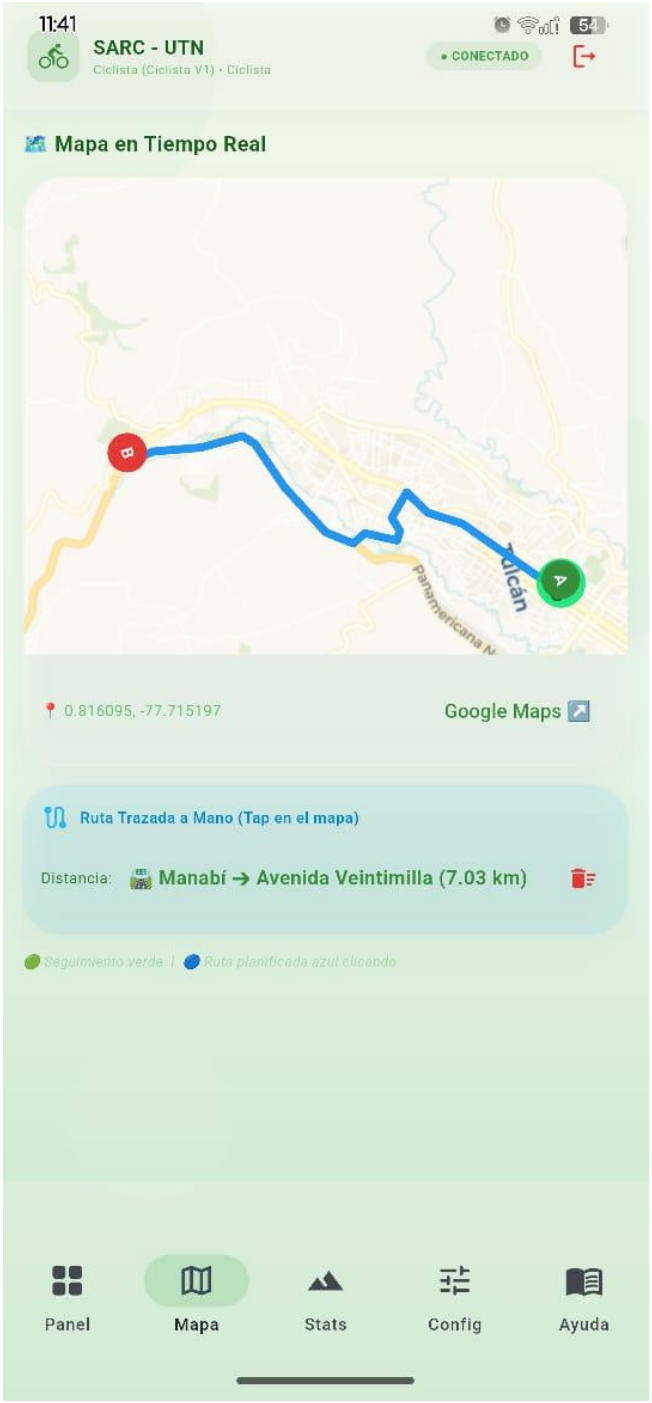
```
def _haversine_km(self, lat1, lon1, lat2, lon2):
    """Calcula distancia en km entre dos coordenadas  considerando la curvatura de la Tierra."""
    R = 6371 # Radio de la Tierra en kilómetros
    dlat = math.radians(lat2 - lat1)
    dlon = math.radians(lon2 - lon1)
    a = math.sin(dlat/2)**2 + math.cos(math.radians(lat1)) * math.cos(math.radians(lat2)) * math.sin(dlon/2)**2
    return R * 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
# Calcular distancia acumulada en lugar de velocidad (con filtro de ruido/drift)
now = time.time()
if self._prev_gps and self._prev_time:
    dt = now - self._prev_time
    if dt > 0.5:
        dist = self._haversine_km(self._prev_gps[0], self._prev_gps[1], lat, lon)
        # Filtro Haversine:
        # Ignorar micro-saltos estacionarios menores a 5m (0.005 km)
        # Ignorar drift irreal mayor a 500m (0.5 km) en menos de 1 segundo
        if 0.005 < dist < 0.5:
            self.distancia_recorrida_km += dist
self._prev_gps = (lat, lon)
self._prev_time = now
```

Nota. Autoría propia

En la figura 51 se expone la lógica de cómo funciona el trazado dual (Polylines) implementada en el sistema contrastando visualmente la ruta planifica y el recorrido del ciclista. Por un lado, el sistema indica mediante un marcador de color verde que indica la posición actual del ciclista, paralelamente, la aplicación proyecta una ruta inteligente de color azul la cual consume la API de OpenRouteService (ORS) bajo el perfil de cycling-regular, lo que ayuda a priorizar mediante este perfil es trayectos de ciclovías y calles aptas para bicicletas.

Figura 51

Trazado Dual de Trayectorias (Polylines)



Nota. Autoría propia

Para mejorar la experiencia del usuario, la interfaz realiza una lógica de geocodificación inversa lo que permite extraer los metadatos de los segmentos de ORS y traducir las coordenadas crudas a nombres legibles de las calles. Además, la interfaz cuanta con un botón de salto (Deep

Link) el cual permite exportar las coordenadas de destino hacia la aplicación nativa del móvil, lo que permite delegar la carga de navegación asistida por voz (véase en la figura 51).

Finalmente, en la tabla 14 se contrastan las métricas y del beneficio al obtener los datos crudos desde el sensor y posteriormente gestionados por ORS llegando a la optimización de servicios por terceros. Garantizando así una visualización precisa y libre de ruido.

Tabla 14

Comparativa de ruteo espacial

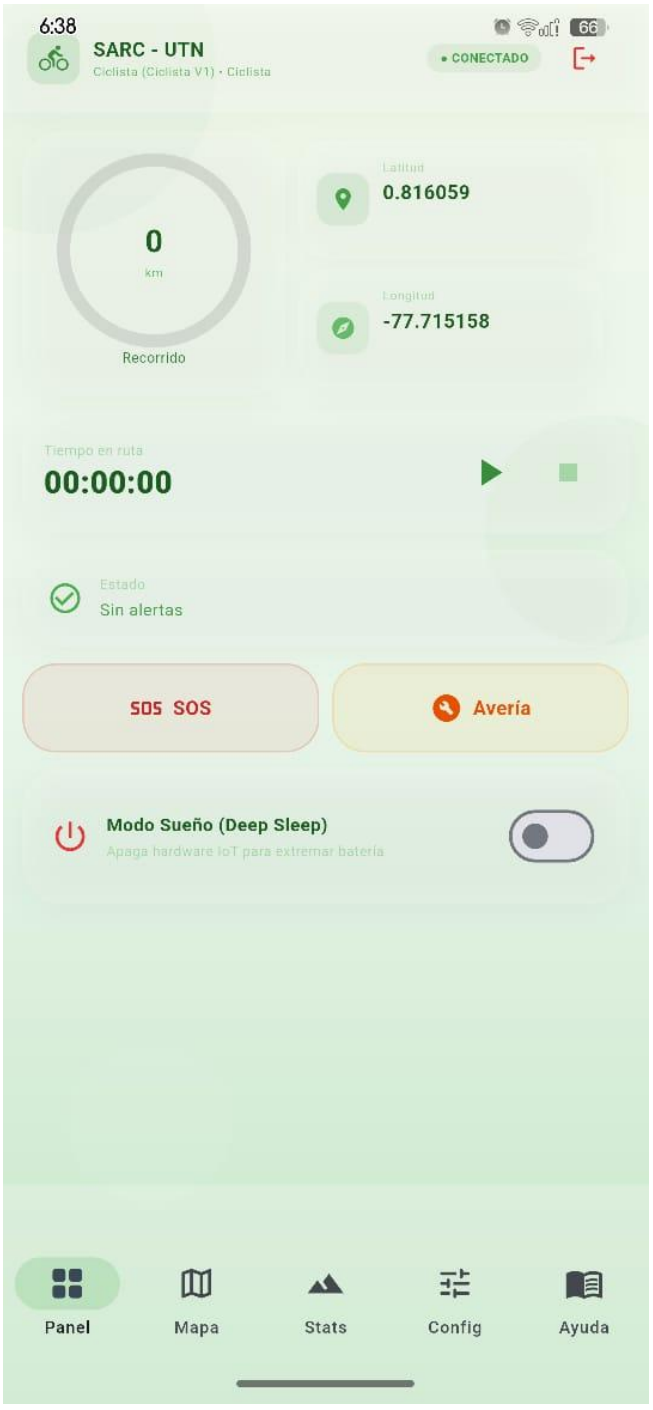
Característica	Ruteo Crudo (GPS Sensor)	Ruteo Inteligente (ORS)
Origen de la Información	Coordenadas emitidas por el hardware NEO-6M.	Nodos del proyecto OpenStreetMap (OSM).
Naturaleza del Trazado	Línea recta entre puntos registrados (Histórico).	Ajuste exacto a la trama vial y ciclovías (Proyectado).
Consumo de Procesamiento	Mínimo (Manejo local de arrays).	Medio (Requiere peticiones asíncronas RESTful).
Precisión de Ubicación	Sujeta al margen de error del hardware GNSS.	Topológicamente perfecta (Alineada a la calle).

3.9.2.2 Gestión de alertas y notificaciones

Para manejo de crisis en tiempo real, el sistema fue diseñado para operar bajo el comportamiento asíncrono, lo que permite que el widget de informe de estado actúe en tiempo real sin congelar ni bloquear la interfaz principal. Como se observa en la figura 52, el sistema aplica la perspectiva del color para clasificar el riesgo, la interfaz tiene la habilidad de cambiar a color naranja para advertencias de proximidad y en un rojo intenso frente a una situación crítica.

Figura 52

Widget de Estado



Nota. Autoría propia

Para el accionar de los botones de pánico manuales (SOS o avería mecánica), el sistema provee una visual inmediata, pero sin ser intrusiva para el usuario, además como se puede observar en la figura 53, se utilizó una de las mejores prácticas utilizando SnackBars, esto ayuda a proporcionar un feedback instantáneo al usuario, pero sin interrumpir su acción en la ruta ni de cerrar ventanas emergentes.

Figura 53

Confirmación No Bloqueante (SnackBar)



Nota. Autoría propia

La arquitectura de la aplicación y del protocolo de latencia cero, se evidencia en la figura 54, en donde permite que el sistema pueda priorizar el “Tiempo de Rescate” esto en base a los fallos positivos, priorizando el rescate efectivo, el sistema tiene la capacidad de invocar directamente la ejecución para emergencias, con el fin de asegurar el despacho instantáneo del payload de auxilio.

Figura 54

Ejecución Asíncrona de Emergencia (Latencia Cero)

```
def on_sos(e):
    #Manejador del botón físico/virtual de emergencia (SOS)
    # cur[0] es Latitud, cur[1] es Longitud en tiempo real
    # Envía la alerta MQTT con la latitud y longitud en tiempo real
    mqtt_service.send_sos(uid, cur[0], cur[1])
    # Confirmación No Bloqueante (SnackBar)
    sb = ft.SnackBar(ft.Text(f"🚨 ¡SOS ENVIADO! Coordenadas: {cur[0]:.4f}, {cur[1]:.4f}", color=WHITE), bgcolor="#C62828")
    page.overlay.append(sb)
    sb.open = True
    page.update()
```

Nota. Autoría propia

Finalmente, en la tabla 15 se consolidaron los tipos de colores y significado frente a un accidente o notificación, mapeando con precisión los eventos suscitados en hardware y enlazando con su correspondiente código de color y la accione que ejecuta.

Tabla 15

Diccionario de Colores y Significados de Alerta.

Evento	Semántica Color (UI)	Iconografía	Acción ejecutada
Aproximación Vehicular	Naranja (*Warning*)	Advertencia	Mutación de Widget UI (Temporal).
Caída / Impacto	Rojo Intenso (*Critical*)	Señal de Alerta	Mutación UI permanente + Envío de API Twilio (SMS)

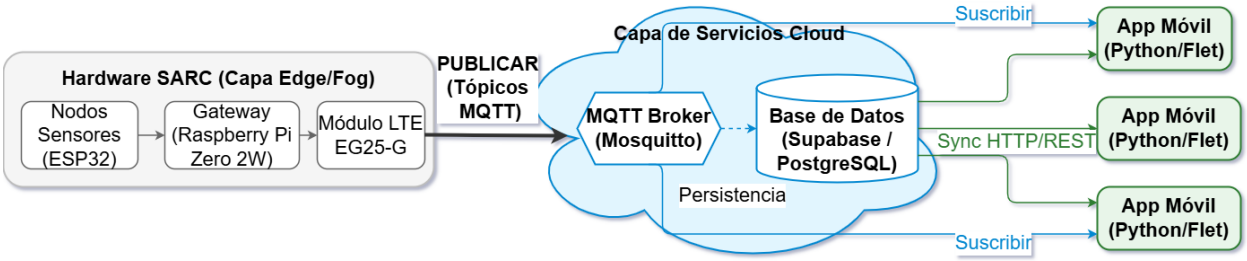
SOS / Avería Mecánica	Carmesí / Mostaza	Llave Inglesa / SOS	Confirmación visual por *SnackBar* + Envío de Coordenadas.

3.9.2.3 Acceso remoto universal

Bajo la persistencia remota de datos, el sistema permite transcender las limitaciones físicas para ofrecer un monitoreo sin restricciones del entorno. Como se ilustra en la figura 55, la topología implementada fue planteada con la visión de eliminar la conexión punto a punto (P2P) entre el hardware de la bicicleta y la aplicación móvil del usuario.

Figura 55

Topología de Red Desacoplada (Acceso Múltiple)



Nota. Autoría propia

Gracias al enlace de datos LTE que provee al microordenador central, el nodo central le permite conectarse con la aplicación mediante una sincronización asíncrona entre el MQTT y el backend de Supabase. De este modo, la aplicación permite que múltiples usuarios sean capaces de conectarse simultáneamente, sin sobrecargar los límites de los recursos de procesamiento del hardware instalado en la bicicleta.

La capacidad de acceso remoto, evidenciada en la figura 56, demuestra que el usuario que está supervisando puede observar el estado del ciclista de cualquier red celular (4G/5G) o teniendo

acceso a internet a varios kilómetros de distancia. Para mitigar el impacto de la latencia de red global, es de gran importancia destacar que, a pesar de operar bajo una red WAN, el uso del protocolo MQTT permitió reducir el impacto de la latencia de dicha red superando así los clásicos retardos de peticiones que se obtiene en el protocolo HTTP.

Figura 56

Comprobación de Operatividad en Red WAN



Nota. Autoría propia

Aprobando así que, la persistencia en la nube ayuda a que cualquier usuario auditor pueda acceder al último estado conocido y al historial de la ruta, incluso si la aplicación se abre luego de que el ciclista haya finalizado el recorrido o apagado el sistema.

3.9.3. *Estructura de Navegación por Roles*

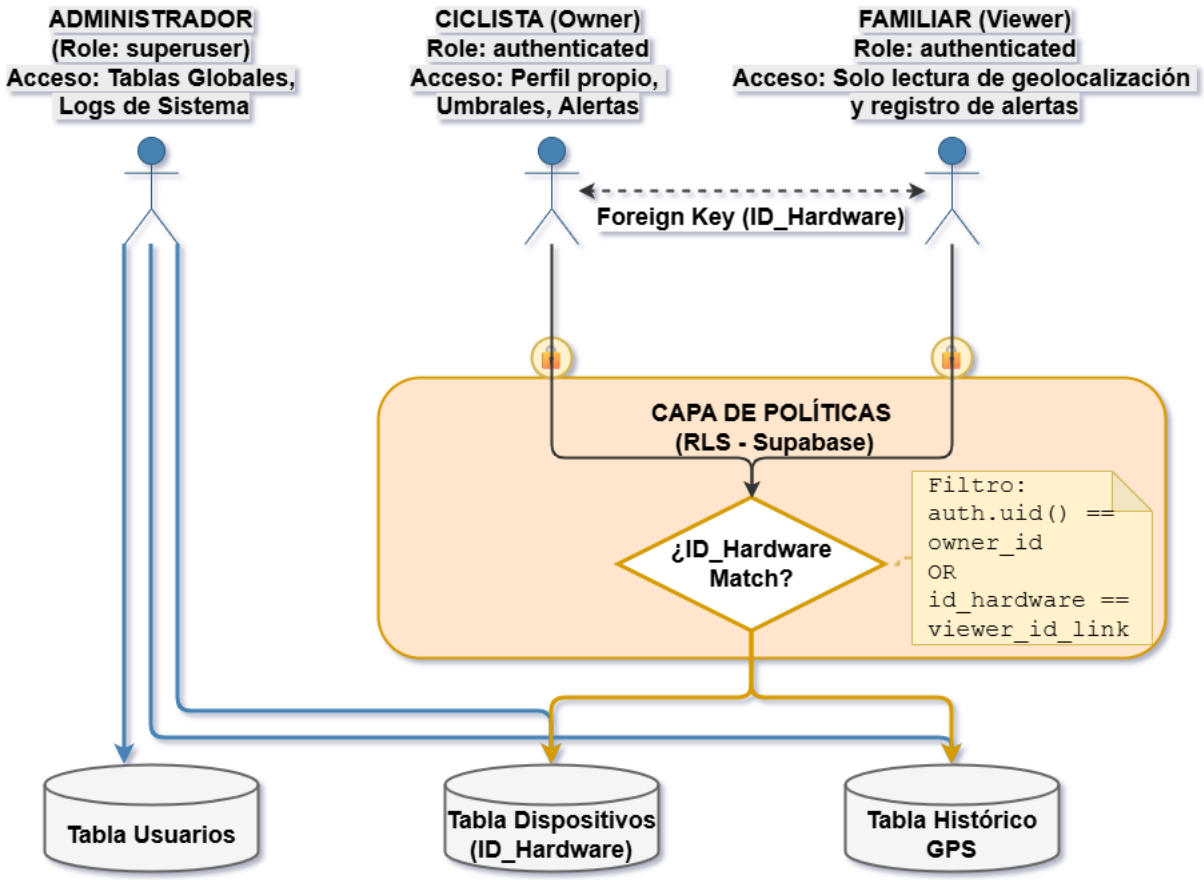
Para la autenticación de usuarios, el sistema implementa un modelo de Control de acceso Basado en roles (RBAC), separando de manera estricta las capacidades de la aplicación. En el proceso de autenticación, el sistema descarta cualquier tipo de terceros lo que permite ejecutar una

validación personalizada (Custom SQL), para ello se empleó el controlador pg8000 antes mencionado.

Con el fin de restringir el nivel de acceso, la figura 57 ilustra el diagrama de jerarquía, en la cual la plataforma se define tres perfiles de acceso, el Administrador (Nivel 0), el Ciclista quien será el propietario del hardware (Nivel 1) y el Familiar o monitor (Nivel 2).

Figura 57

Modelo de Control de Acceso Basado en Roles (RBAC)



Nota. Autoría propia

Con el fin de restringir el acceso funcional, se realizó una matriz de permisos que se puede observar en la tabla 16, en donde la asignación de privilegios se encuentra mapeada de acuerdo al rol de usuario, bloqueando cualquier escala de privilegios en la aplicación.

Tabla 16

Matriz de Control de Acceso.

Evento	Nivel 0	Nivel 1	Nivel 2
Gestionar y Eliminar Usuarios	Permitido	Denegado	Denegado
Enviar Comando MQTT (Deep Sleep)	Permitido	Permitido	Denegado
Actualizar (Contactos)	Permitido	Permitido	Denegado
Monitorear Mapa y Telemetría	Permitido	Permitido	Permitido
Recepción de Alertas Automáticas	Permitido	Permitido	Permitido

3.9.3.2 Interfaz Usuario Nivel 0 (Administrador)

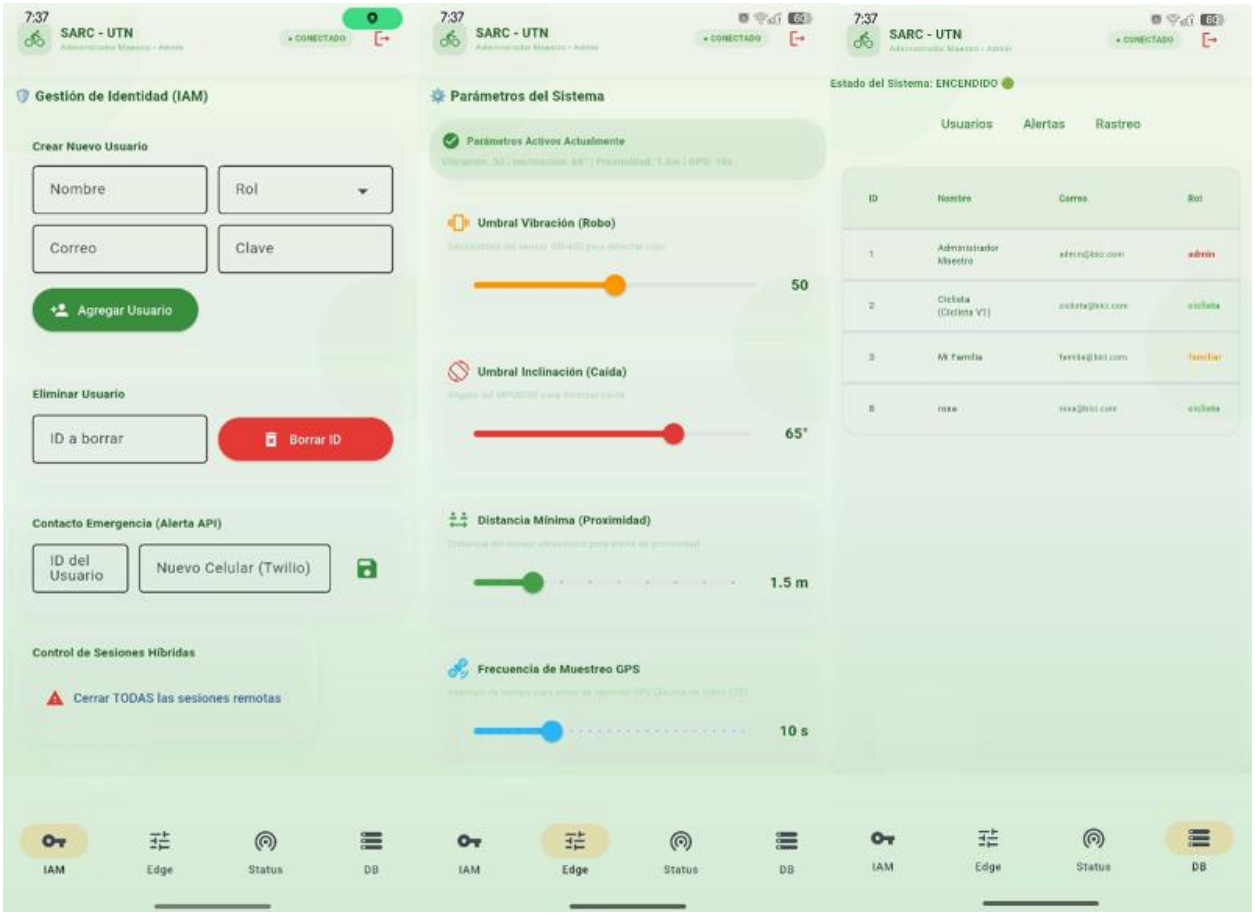
En coherencia con este nivel de privilegios, el rol de Administrador es quien centraliza el control lógico del sistema denominad SARC en la aplicación, puesto que, es aquel usuario que tiene todos los privilegios en donde puede realizar acciones operacionales CRUD (Creas, Leer, Actualizar, Borrar) sobre la base de datos de usuarios y modificación de parámetros del sistema.

Como se puede evidenciar en la figura 58, dicho usuario posee todo privilegio con el fin de realizar cualquier cambio sin la necesidad de ingresar directamente al gestor de datos. Además, puede visualizar los datos históricos registrados durante todo el tiempo en el que el sistema estuvo en uso esto con el fin de verificar los resultados obtenidos y verificar que tipo de alerta y en qué

fecha y hora sucedió tal incidente, junto a esto tiene una leyenda “Estado del sistema” en donde verifica este usuario si el sistema está en funcionamiento o caso contrario apagado.

Figura 58

Interfaz principal del usuario Nivel 0 (Admin)



Nota. Autoría propia

3.9.3.3 Interfaz Usuario Nivel 1 (Usuario-Ciclista)

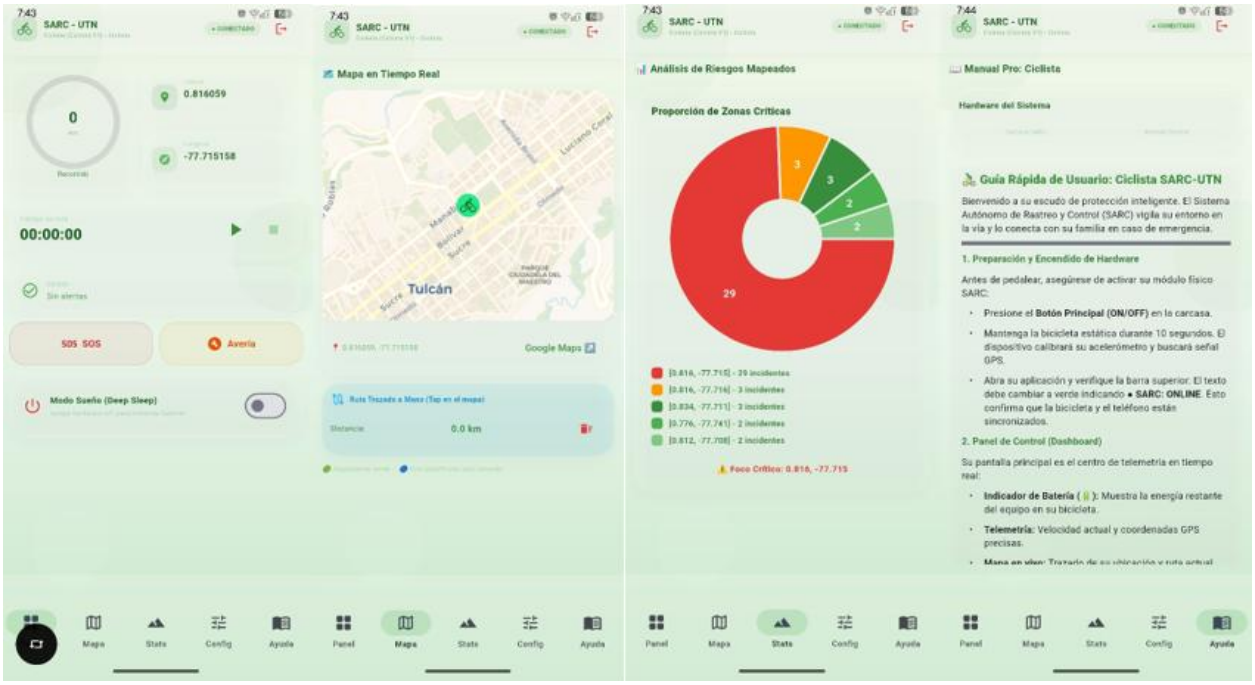
En el Nivel 1, está el perfil del Ciclista el cual está actuando como Owner del sistema, con una comunicación bidireccional que le permite realizar configuraciones MQTT hacia el hardware en el sistema como activar el modo Deep Sleep. Además, que cuenta con un privilegio de Nivel 0 para realizar cualquier cambio en cuanto se refiere a un contacto de emergencia, esto se puede evidenciar en la tabla 16 en donde se realizó un análisis de los privilegios de cada usuario según

su jerarquía. En la figura 59 se puede observar como la interfaz del usuario Nivel 2 además tiene dos botones en su panel principal, estos están ubicados con el fin de permitir que el usuario pueda enviar alertas críticas a cualquier contacto de emergencia fue asignado.

De la misma manera, este usuario posee 5 paneles en donde están segregados por diferente tipo de contenido, en el primer panel se encuentra la interfaz principal en la que detalla Gauges de kilometraje, latitud y longitud, cronometro, y los botones que se ha mencionado para los eventos críticos, en el panel 2 se encuentra el mapa en el cual el usuario podrá trazar la ruta y esta a su vez tendrá un icono de color verde el cual indica en que posición se encuentra el ciclista, en el panel 3, encontramos una interfaz en la que indica un análisis mapeado de riesgos mediante una gráfica proporcionando información de las zonas críticas que se encontraron y en que ubicaciones fueron. En la pestaña de config, el usuario puede modificar el contacto de emergencia esto con la finalidad de que el sistema pueda enviar la alerta a el contacto correcto y finalmente en la pestaña 5 de la interfaz el usuario cuenta con un manual de cómo funciona el sistema.

Figura 59

Interfaz principal del usuario Nivel 1 (Ciclista)



Nota. Autoría propia

3.9.3.4 Interfaz usuario Nivel 2 (Usuario-Familiar)

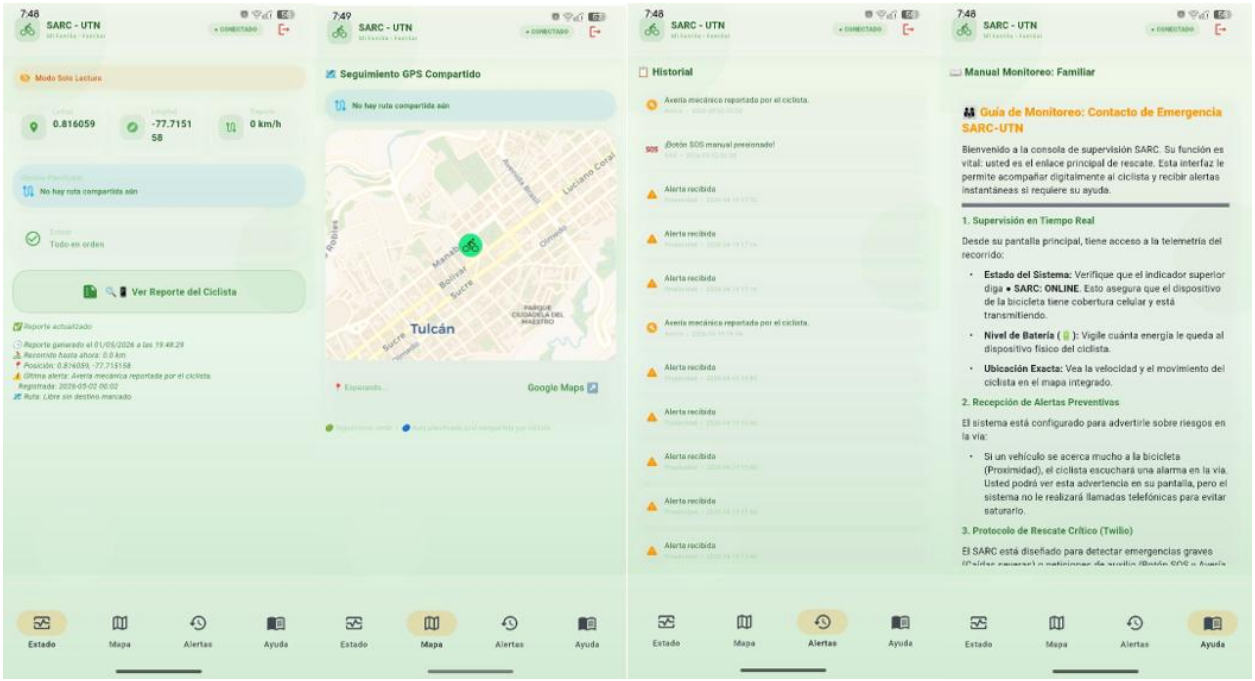
Para el perfil de Nivel 2, en el cual está el Familiar donde tiene una vista dinámica que genera una interfaz estrictamente solo de lectura por la seguridad del ciclista, que está orientado exclusivamente a la recepción satelital y la recepción asíncrona de alertas. Para mitigar la intervención externa, en la figura 60 se evidencia esta separación: por el lado del usuario Nivel 2 el cual carece botones de control en la pantalla, en donde, no es por falta de desarrollo, sino que es por una medida al Principio Mínimo Privilegio (PolP).

Además, este usuario cuenta con 4 paneles en su interfaz, en el primer panel se encuentra detallado la ubicación del usuario Nivel 1, tanto las coordenadas, kilometraje, la ruta compartida por el ciclista y un botón que ayuda a verificar el reporte del ciclista en tiempo real, en su panel 2 está ubicado el mapa de seguimiento compartid, en donde, el usuario podrá ver la ruta establecida por el ciclista y además monitorearlo en tiempo real durante su trayecto, también cuenta con un botón de salto Deep Link que permite abrir el mapa nativo de cada celular para poder tener un

servicio de terceros como es una navegación por voz asistida. En su panel 4 se encuentra una historial de todas las alertas que el hardware de la bicicleta vaya registrando y por último en la pestaña 5 se encuentra de igual manera un manual de usuario en el que detalla como intervenir en caso de un evento crítico y como será enviado esta notificación hacia el contacto de emergencia.

Figura 60

Interfaz principal del usuario Nivel 2 (Familiar)



Nota. Autoría propia

3.9.3.5 Gestión de credenciales y perfiles (Supabase)

En la figura 61, el fragmento de código evidencia la lógica de la función verify_login, en la cual realiza una consulta segura y directa a la tabla de control de usuarios alojada en el gestor de datos en la nube Supabase. Mediante este argumento de seguridad, se garantiza por diseño que un usuario secundario pueda alterar accidentalmente o maliciosamente el estado de todo el hardware de la bicicleta y por arquitectura se establece mediante un único id y clave foránea (Foring Key) blindando la privacidad de coordenadas contra accesos no autorizados.

Figura 61

Validación Segura de Credenciales en Capa de Datos

```python
def verify_login(correo, password):
    """Verifica credenciales contra la tabla usuarios."""
    conn = get_db_connection()
    if conn is None:
        return None, "Verifique conexión a internet"
    try:
        cursor = conn.cursor()
        # Consulta segura a Supabase mediante paso de parámetros (%s)
        # Esto previene inyecciones SQL y protege el acceso a la telemetría.
        query = "SELECT id, nombre, correo, rol, contacto_emergencia FROM usuarios WHERE correo = %s AND password_text = %s"
        cursor.execute(query, (correo, password))
        user = dictfetchone(cursor)
        if user:
            return user, "Login exitoso"
        else:
            return None, "Correo o contraseña incorrectos"
    except Exception as e:
        print(f"Error en consulta DB: {e}")
        return None, "Error de sistema"
    finally:
        # Cierre seguro de conexiones a la nube
        if conn:
            try: cursor.close()
            except: pass
            try: conn.close()
            except: pass
```

Nota. Autoría propia

3.9.3.6 Interfaz de Usuario y Experiencia (UX/UI)

Desde la perspectiva del usuario, el diseño de la aplicación móvil descarta los enfoques genéricos para adoptar una interfaz ergonómica, como se observa en la figura 46, esta elección cromática basada en una estética *Soft-Glass* con tonos verdes color menta no es simplemente con la intención de estética, sino que, responde a la necesidad de reducir el reflejo o efecto espejo ocasionado con la luz solar, superando de manera amplia la legibilidad.

A nivel de reacción interactiva, el sistema cuenta con un esquema que permite retroalimentación multimodal, que fue diseñado para no interrumpir, como se observa en la figura 46 este comportamiento permite advertir visualmente la alerta por perdida de conectividad MQTT solo captando la visión periférica del ciclista. Simultáneamente, la confirmación de botones

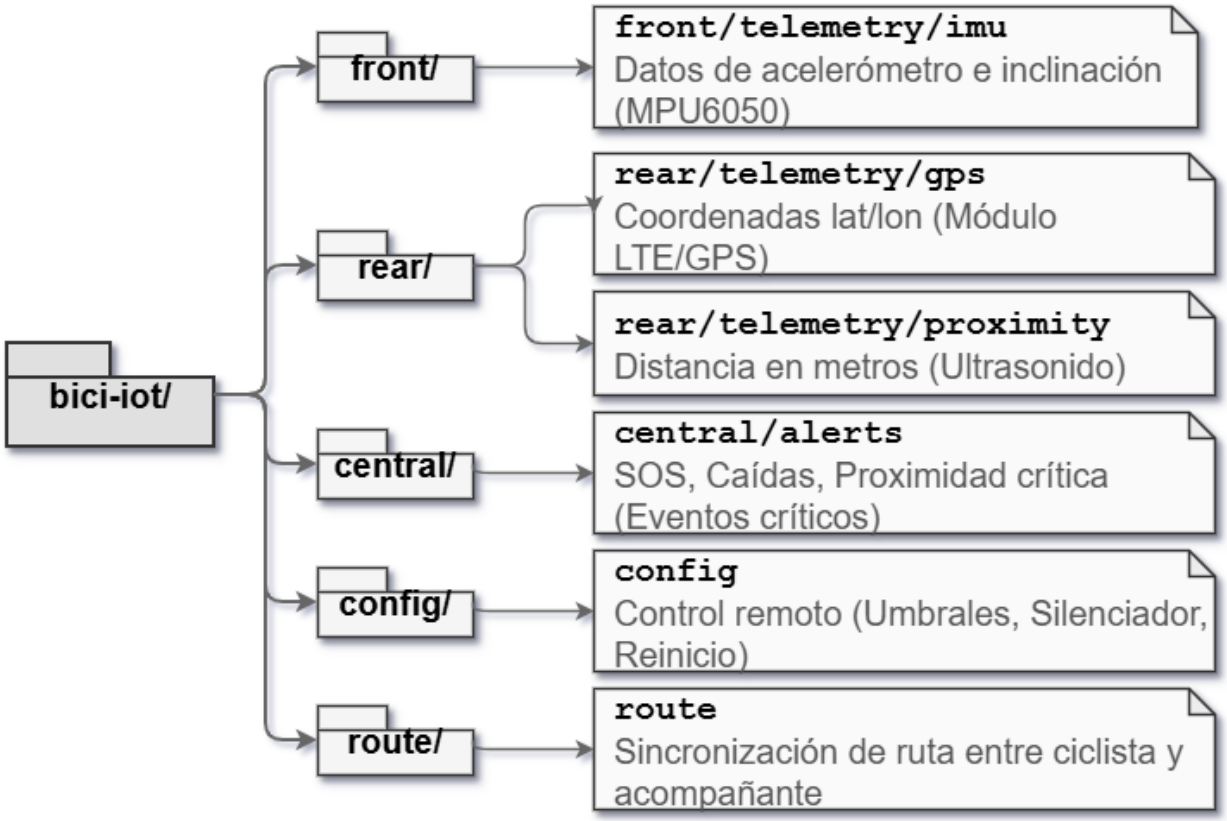
manuales se los realiza mediante Snackbars que aparecen y desaparecen en la parte inferior sin tener que cierre ventanas emergentes y sin perder visualización completa de la interfaz.

3.9.4. *Comunicación con la nube*

En lo que respecta a la capa transporte, el sistema está estructurado mediante jerarquía de tópicos MQTT, como se detalla en la figura 62, la separación de los tópicos permite aislar cualquier tipo de tráfico proveniente de la red, garantizando que la escalabilidad del hardware no tenga colisiones de datos.

Figura 62

Diagrama de Jerarquía de Tópicos



Nota. Autoría propia

De acuerdo a la lógica de persistencia filtrada, se detalla en la figura 63, en donde garantiza la integridad de los datos almacenados en Supabase al rechazar de forma directa las coordenadas correspondientes al vector nulo (0,0).

Figura 63

Filtro de Deriva y Persistencia en Capa de Lógica

```
if msg.topic == TOPIC_GPS:
    data = payload.get("data", {})
    lat = data.get("lat", 0.0)
    lon = data.get("lon", 0.0)
    # Calcular distancia acumulada en lugar de velocidad (con filtro de ruido/drift)
    now = time.time()
    if self._prev_gps and self._prev_time:
        dt = now - self._prev_time
        if dt > 0.5:
            dist = self._haversine_km(self._prev_gps[0], self._prev_gps[1], lat, lon)
            # Filtro Haversine:
            # Ignorar micro-saltos estacionarios menores a 5m (0.005 km)
            # Ignorar drift irreal mayor a 500m (0.5 km) en menos de 1 segundo
            if 0.005 < dist < 0.5:
                self.distancia_recorrida_km += dist
    self._prev_gps = (lat, lon)
    self._prev_time = now

# === GUARDAR GPS EN BASE DE DATOS ===
# 2. Persistencia en Capa de Datos (Supabase)
uid = self.current_user_id if self.current_user_id else 2
# Validación condicionada para omitir coordenadas vacías
if lat != 0.0 and lon != 0.0:
    threading.Thread(
        target=self._save_gps_to_db,
        args=(uid, lat, lon),
        daemon=True
    ).start()
```

Nota. Autoría propia

Finalmente, el sistema realiza una marca de tiempo de todos los datos bajo la marca de tiempo en la zona horaria UTC-5 tal como se evidencia en la figura 64, esta sincronización de horario es una métrica fundamental con respecto a la seguridad del ciclista, ya que asegura las

auditorias de la ruta y el registro histórico de datos correspondan a la hora local de Ecuador sin importar con la latencia en la transmisión.

Figura 64

Estandarización Temporal (UTC-5) en el Backend

```
    if isinstance(v, datetime):
        if v.tzinfo:
            d[k] = v.astimezone(timezone(timedelta(hours=-5))).replace(tzinfo=None)
        else:
            d[k] = v - timedelta(hours=5)
```

Nota. Autoría propia

3.9.4.1 Recepción de datos

Bajo este esquema de procesamiento concurrente, garantiza que el socket TCP/IP y el sondeo de mensajes MQTT puedan ocurrir en segundo plano, sin interrumpir la navegación del usuario, la lógica de la deserialización JSON esta detallado en el fragmento de código de la figura 65, donde se asegura que las tramas binarias en texto plano se decodifiquen y transformen en diccionarios nativos de Python. Para evitar la colisión de variables el sistema procesa la cabecera de cada paquete antes de pasarlo.

Figura 65

Decodificación y Des-serialización de Carga Útil

```
def _on_message(self, client, userdata, msg):
    # Filtro de seguridad: ignorar telemetría si el sistema está en Deep Sleep
    if self.is_sleeping and msg.topic != TOPIC_ROUTE:
        return  # Ignora telemetría si está simulando Deep Sleep
    try:
        # 1. msg.payload.decode() -> Decodificación de Bytes a String (UTF-8)
        # 2. json.loads()         -> Des-serialización de String a Diccionario Python
        payload = json.loads(msg.payload.decode())
    except Exception:
        # Si el mensaje no es un JSON válido (basura o interferencia), se ignora
        return
```

Nota. Autoría propia

Minimizando así la latencia, el sistema opera bajo un modelo Event-Driven, en donde la aplicación descarte el buffer de espera reduciendo así el consumo de la memoria volátil. En su lugar, el callback _on_message está actuando como un actuador instantáneo que inyecta información directamente a los componentes visuales, obteniendo como resultado una latencia mínima.

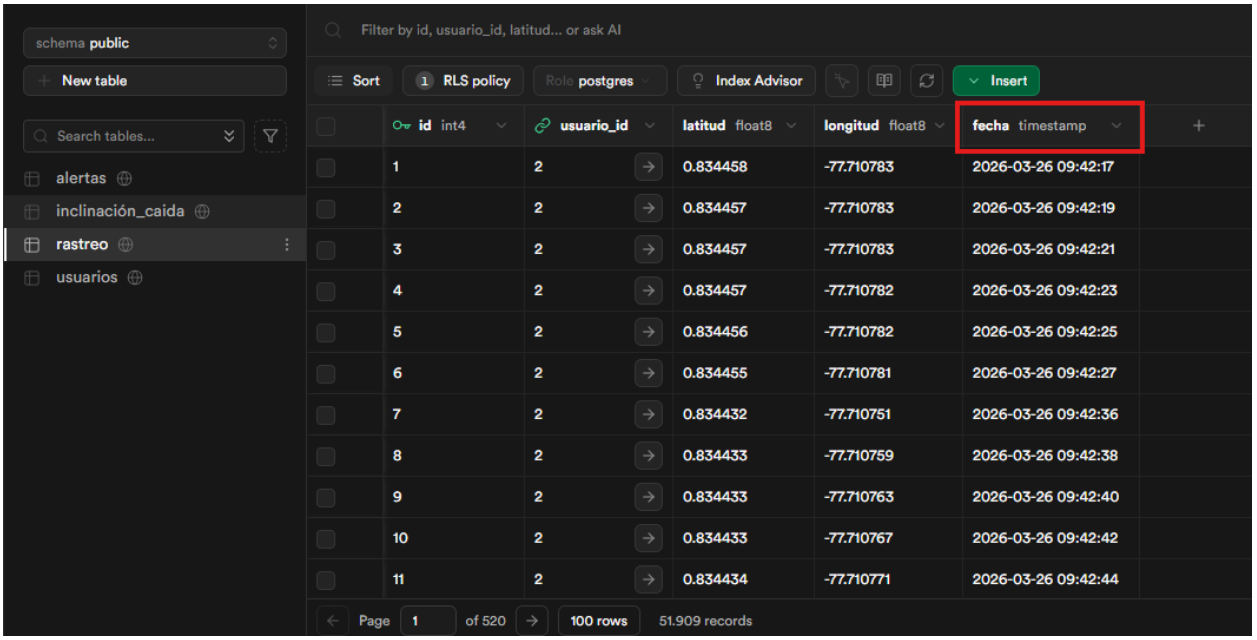
3.9.4.2 Almacenamiento y actualización

Con respecto al almacenamiento, sistema está implementando una capa de preprocesamiento selectivo en recepción, esto para que, restringa las escrituras en la nube a datos válidos. Con esto se optimiza drásticamente el volumen de almacenamiento requerido en los servidores de Supabase.

Como se puede observar en la figura 66, la intensidad cronológica no depende del reloj interno del dispositivo del cliente, más bien, el sistema delega esta facultad mediante una marca de tiempo en el motor PostgreSQL, es fundamental garantizar la obtención de datos inalterables para tener una auditoría de rutas y poder reconstruir dicho accidente.

Figura 66

Delegación de Integridad Cronológica en PostgreSQL



Nota. Autoría propia

Para los procesos de actualización de información sensible, como contactos y perfiles de usuarios, la arquitectura emplea sentencias SQL parametrizada, bloqueando al acceso no autorizado de terceros. En entornos de movilidad, la perdida de paquetes es segura, para ello la arquitectura permite manejar excepciones como se detallan en la figura 67, que actúa ante fallos de red, esta estructura garantiza el cierre sistemático del cursor y la conexión a la base de datos bajo cualquier escenario de fallos eliminando el riesgo de fuga de memoria que como consecuencia podrían colapsar el sistema operativo del celular.

Figura 67

Manejo de Excepciones y Prevención de Fugas de Memoria

```python
def actualizar_contacto(user_id, nuevo_contacto):
    """Actualiza el número de emergencia en la BD (Twilio/Edge)."""
    conn = get_db_connection()
    if conn is None:
        return False
    try:
        # Bloque de ejecución principal
        cursor = conn.cursor()
        query = "UPDATE usuarios SET contacto_emergencia = %s WHERE id = %s"
        cursor.execute(query, (nuevo_contacto, user_id))
        conn.commit() # Confirma los cambios si todo sale bien
        return True
    except Exception as e:
        # Manejo de Excepciones: Captura cualquier error SQL o de Red
        print(f"Error actualizando contacto: {e}")
        if conn: conn.rollback()  # Revierte cambios parciales corruptos
        return False
    finally:
        # Prevención de Fugas de Memoria:
        # Este bloque se ejecuta SIEMPRE, haya error o no.
        if conn:
            try: cursor.close()
            except: pass
            try: conn.close()
            except: pass
```

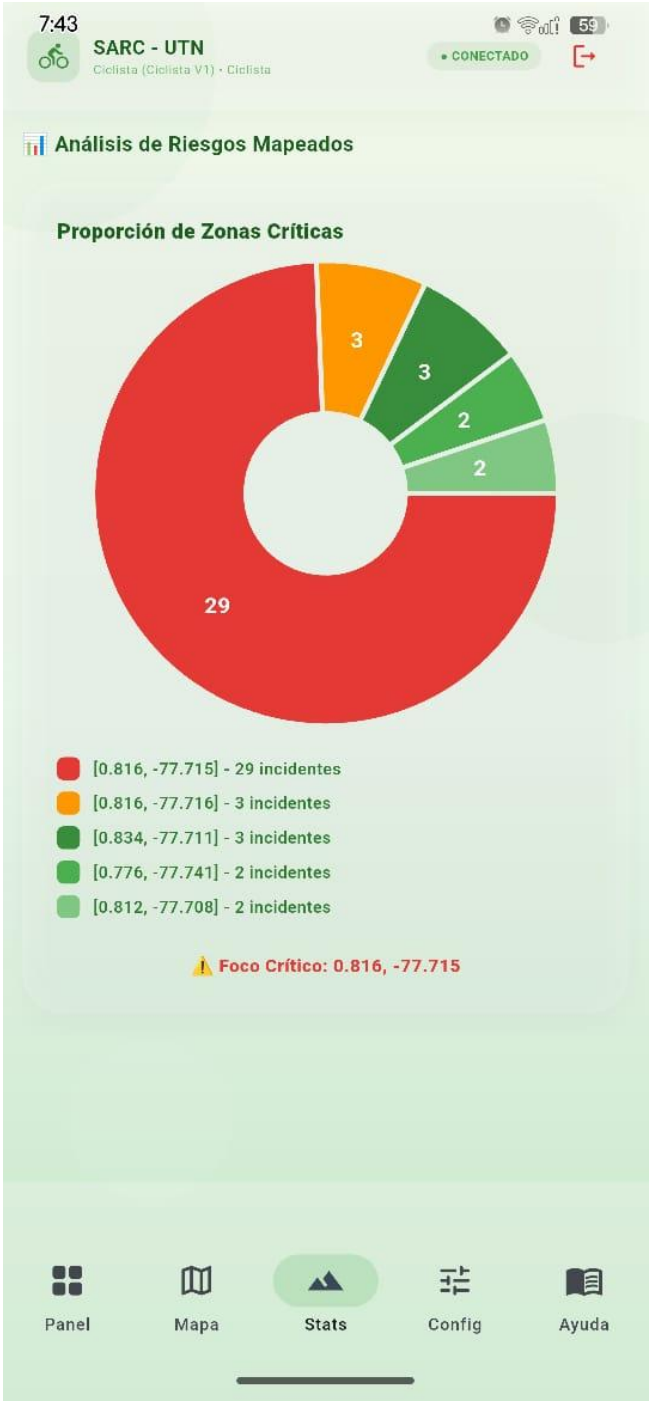
Nota. Autoría propia

3.9.4.3 Gestión de alertas y registro histórico

Con respecto a la monitorización de eventos, la aplicación móvil actúa de forma que transforma los registros crudos en la nube en información legible, que esta diseña prácticamente para tomar decisiones bajo presión. Además, el sistema implementa un análisis básico espacial que se ve ilustrado en la figura 68, que agrupa las incidencias de forma inteligente y realiza un análisis de riesgos.

Figura 68

Analítica de Riesgos y Puntos Calientes.



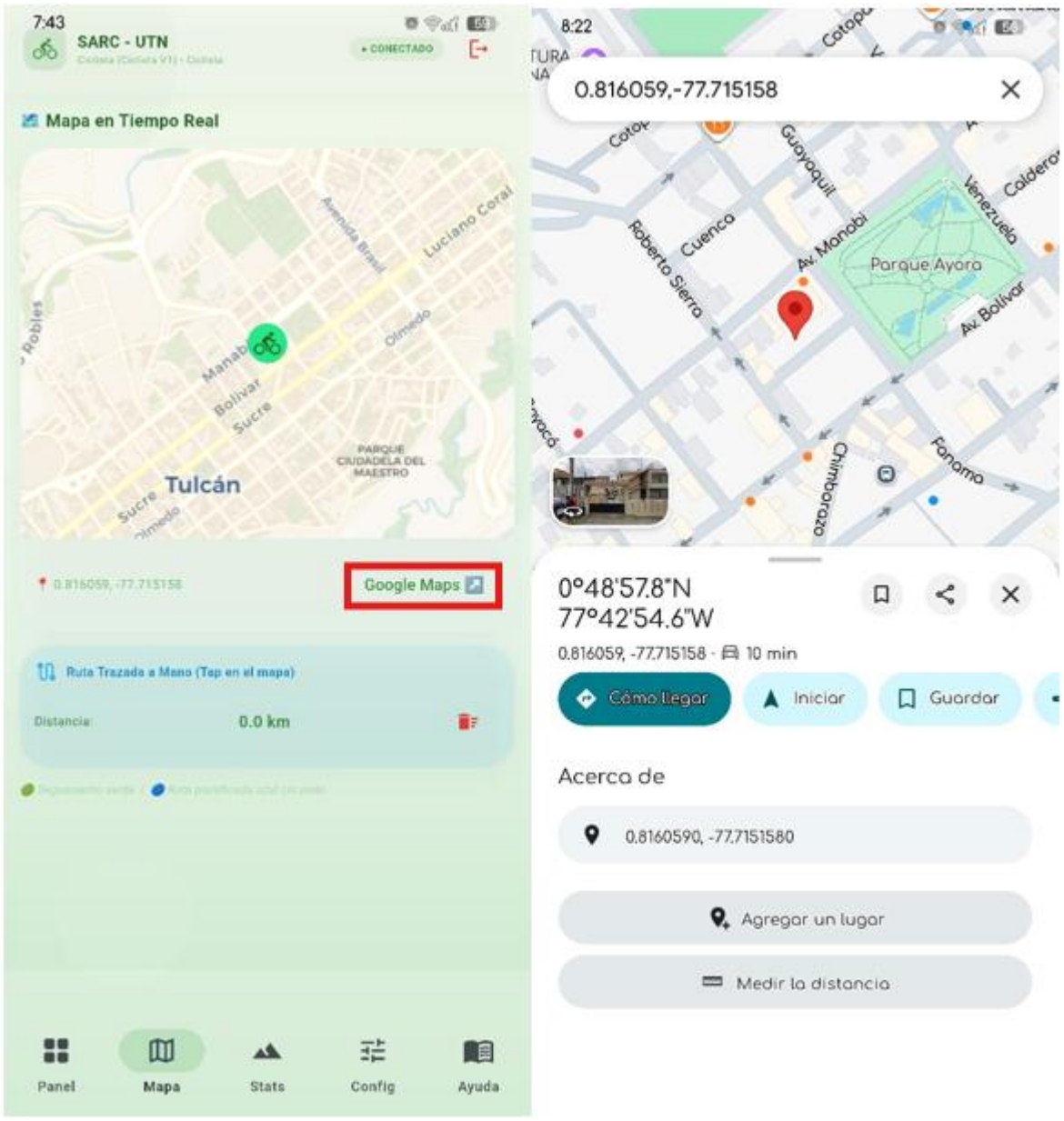
Nota. Autoría propia

Para los escenarios críticos los cuales demanda de acción física inmediata, la arquitectura del software está diseñada para garantizar la operabilidad en cuanto al sistema de rescate (véase en la figura 69). La integración de la localización externa permite que al momento de acceder al

incidente que se encuentra en el SMS, la aplicación exporta las coordenadas de la ubicación mediante la longitud y latitud exactas hacia el API de Google Maps.

Figura 69

Interoperabilidad de Rescate (Deep Link)



Nota. Autoría propia

Con esta operabilidad de navegación nativa del móvil se pretende optimizar significativamente los tiempos de respuesta, mediante la trazabilidad de rutas exactas de intercepción que permita enlazar al familiar o unidades de emergencia.

3.10 Servicios y gestión en la nube

En esta sección, el sistema integra plataformas de terceros, pero de grado empresarial, lo que permite delegar cargas transaccionales y de comunicaciones a infraestructuras de mayor disponibilidad. Bajo la adopción de modelos como Software as a Service (SaaS) y el Backend as a Service (BaaS) ayudan a garantizar que los protocolos de comunicación sobre emergencia operen bajo estándares con una alta disponibilidad.

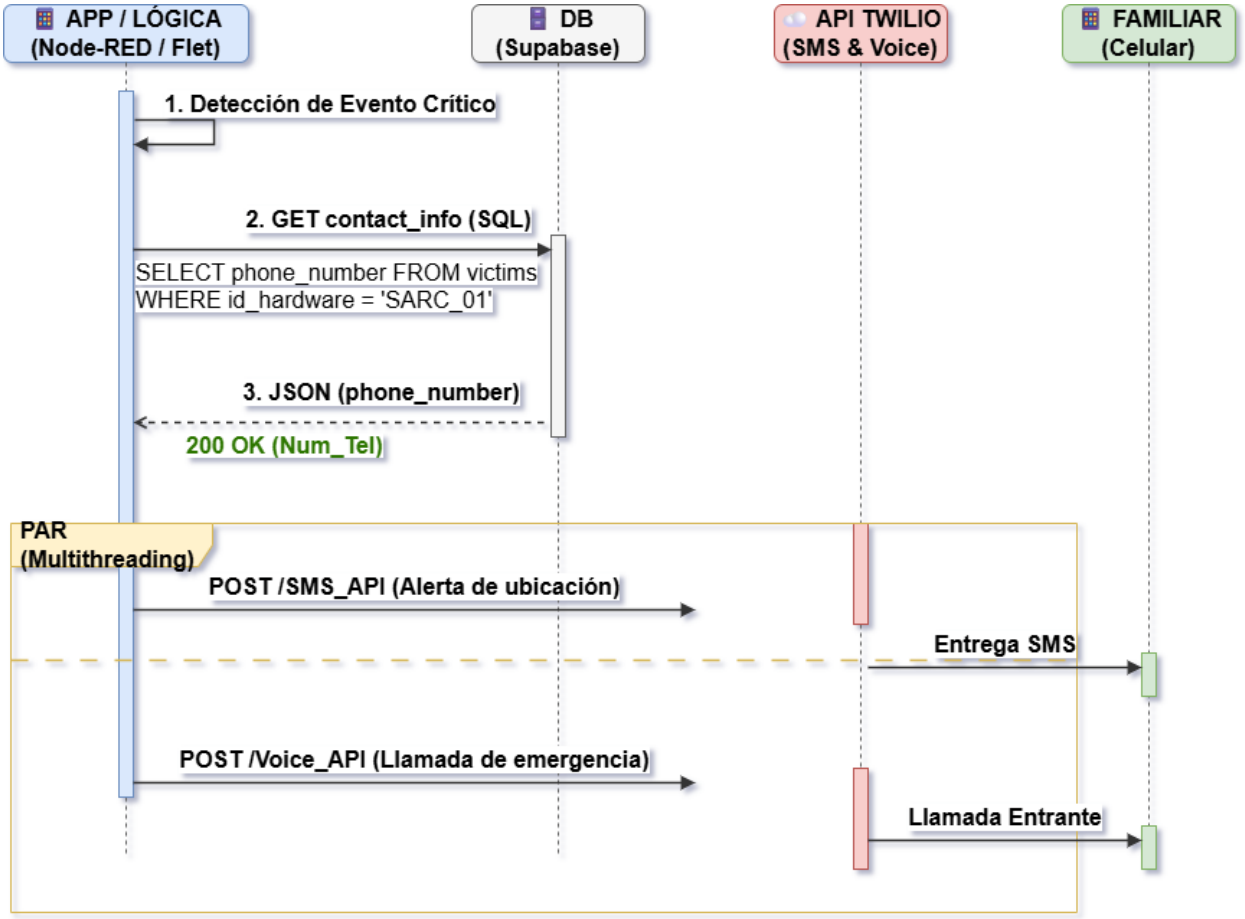
Con el fin de garantizar una operabilidad funcional y sin contratiempos, este sistema traslada el procesamiento del microordenador pesado, el enrutamiento de llamadas y almacenamiento de grandes datos hacia clústeres externos, con lo cual, la nube asume un rol de cerebro principal independizando las lógicas de rescate de las limitaciones del hardware de borde asegurando la persistencia del dato y la ejecución autónoma de las cadenas de alertas.

3.10.1. *Diseño de servicios para integración*

De acuerdo a lo mencionado anteriormente en la figura 70, el modelo Event-Response opera mediante un ciclo de rescate de tres fases, la detección asíncrona del nodo borde (Edge), la recuperación de los contactos de emergencia mediante la utilización de SQL y la ejecución simultánea de estos hilos de alerta remota.

Figura 70

Modelo de Orquestación y Respuesta ante Eventos.



Nota. Autoría propia

En lo que respecta a la telefonía celular, la plataforma hace uso de las APIs de *Programmable Messaging* (SMS) y *Programmable Voice* (Llamadas) que están inmersas en la plataforma Twilio, donde, estos servicios actúan de manera simultánea para asegurar el aviso de socorro, en la figura 71 se evidencia el uso del lenguaje avanzado de marcado TwiML, en el cual una etiqueta denominada <Say> es la encarga de invocar al motor de síntesis de voz (Alice) con el fin de verbalizar la estructura del mensaje y las coordenades del incidente.

Figura 71

Generación Dinámica de Voz (TwiML)

```
    # TwiML directo: sin necesitar dominio ni servidor web
    # Voz "alice" en español: máxima compatibilidad con todos los planes y regiones
    twiml = f"""<?xml version="1.0" encoding="UTF-8"?>
<Response>
    <Say language="es-MX" voice="alice">{texto_voz}</Say>
    <Pause length="1"/>
    <Say language="es-MX" voice="alice">Fin del reporte SARC.</Say>
</Response>"""
```

Nota. Autoría propia

Finalmente, en la tabla 17 se realiza una consolidación de acuerdo al panorama de los servicios integrados en la nube.

Tabla 17

Servicios de Nube Integrados en la Arquitectura SARC.

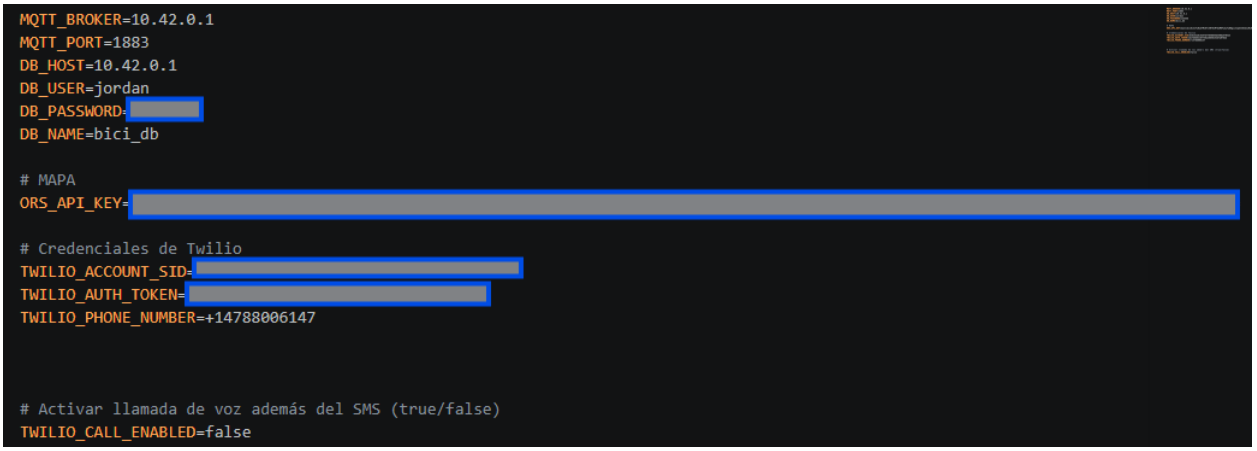
Servicio	Proveedor	Función	Protocolo/librería
Backend as a Service (BaaS)	Supabase (AWS)	Gestión de identidades, almacenamiento histórico y auditoría.	TCP/IP (Librería pg8000).
Programmable Messaging	Twilio API	Despacho de alertas escritas (SMS) en zonas de baja cobertura.	HTTPS RESTful / twilio.rest.
Programmable Voice (TwiML)	Twilio API	Síntesis de voz dinámica (TTS) para llamadas de emergencia.	HTTPS RESTful + marcado XML.
Localización	OpenRouteService	Optimización de trazado vial y geocodificación inversa.	JSON API sobre HTTPS.

3.10.2. *APIs de comunicación y gestión de datos*

Simultáneamente, para garantizar la seguridad de las llaves de acceso, el esquema a deportación está basado en variables de entorno, que están detalladas en la figura 72, al hacer uso de la librería python-dotenv, los tokens de acceso y las contraseñas de la base de datos se activan exclusivamente en tiempo de ejecución, asegurando que ninguna llave maestra resida en texto plano dentro del código fuente.

Figura 72

Aislamiento de Credenciales y Entorno Seguro



Nota. Autoría propia

De acuerdo a esto, todas estas comunicaciones salientes como se evidencia en la figura 73, al utilizar inyecciones de tokens se realiza una protección del sistema contra ataques de intercepción o sniffing, en el código de la figura 72 se detalla como inyecta estos tokens de manera silenciosa sin hacer uso específicamente de las claves de los tokens.

Figura 73

Inyección del Token

```
def _get_client():
    if not TWILIO_SID or not TWILIO_TOKEN:
        print("[SARC] ❌ Faltan credenciales de Twilio en el archivo .env")
        return None
        # Inyección de Tokens para comunicación cifrada y segura (Autenticación)
    return Client(TWILIO_SID, TWILIO_TOKEN)
```

Nota. Autoría propia

Finalmente, la conexión de arquitectura directa esta detallada en la configuración de la figura 74, en donde permite una reducción drástica de la latencia, ocurre cuando evade la construcción de cabecera HTTP en operaciones de alto volumen de escritura. Además de la velocidad para trabajar asegura un control absoluto sobre el estado de la sesión, en donde, el sistema se mantiene estable sin tener consumo de memoria RAM.

Figura 74

Conexión TCP/IP Directa a Base de Datos

```python
def get_db_connection():
    #Establece conexión TCP/IP directa con el motor PostgreSQL de Supabase.
    try:
        # 1. Parseo de la cadena de conexión (Connection String)
        url = urllib.parse.urlparse(DATABASE_URL)
        username = url.username
        password = url.password
        database = url.path[1:]
        hostname = url.hostname
        port = url.port

        # 2. Conexión nativa (Socket TCP) usando el driver pg8000
        connection = pg8000.dbapi.connect(
            user=username,
            password=password,
            host=hostname,
            port=port,
            database=database
        )
        return connection
    except Exception as e:
        print(f"Error conectando a la BD (Supabase - pg8000): {e}")
        return None
```

Nota. Autoría propia

A partir de esta base técnica en el siguiente capitulo se procederá a exponer los resultados obtenidos en todas las pruebas de campo que se realizaron en entornos reales.

Capítulo IV Resultados

En este capítulo se presenta la cuarta fase establecida en la metodología Waterfall, la cual está enfocada en la verificación integral obtenidos del sistema desarrollado. En esta fase se lleva a cabo la evaluación de los parámetros a nivel de capa de recepción de datos complementados con representaciones graficas que demuestran su rendimiento. Del mismo modo, se detalla el proceso

de la integración total de los componentes de hardware y software, conjuntamente con la ejecución de pruebas funcionales especialmente diseñadas para evaluar la operabilidad en entornos reales. Estas pruebas se enfocaron principalmente en validar el apartado de prevención de

accidentes para ciclistas, con el fin de buscar una reducción de tiempos de respuestas antes incidentes. Estas evaluaciones fueron realizadas con la colaboración de ciclistas experimentados, quienes aportaron datos en recorridos tanto en lo urbano como en zona rurales y en escenarios competitivos.

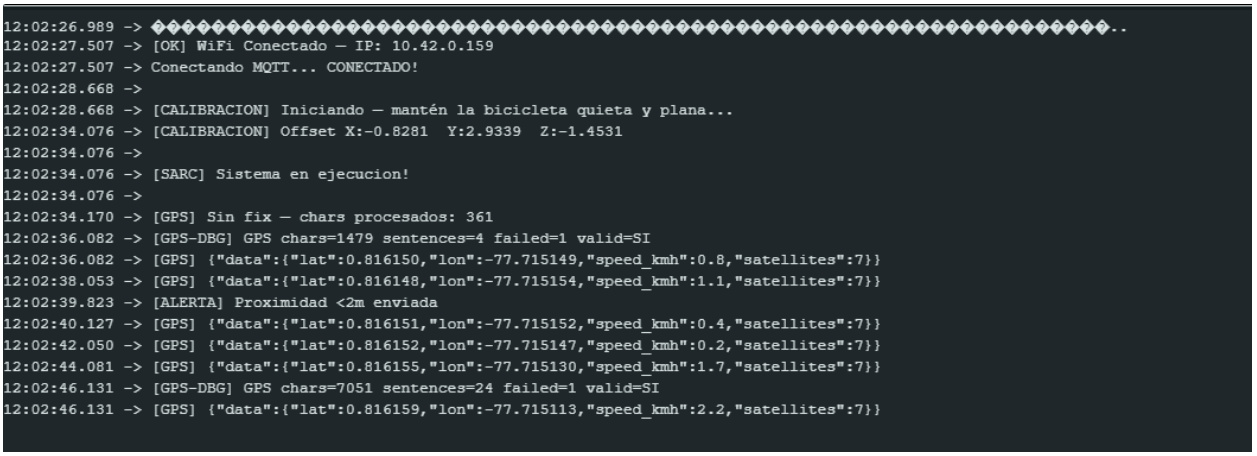
4.1 Resultados del funcionamiento del nodo (ESP32)

Como punto de partida, el ESP32-WROOM-32 actúa como el nodo de percepción de datos física en la arquitectura propuesta, instalado directamente en el chasis de la bicicleta en donde se verifico la estabilidad del microcontrolador (ESP32). En este sentido, se evaluó la confiabilidad técnica del nodo bajo ciclos constantes de trabajo continuo, en donde fue capaz de mantener la capacidad de orquestar continuamente la lectura de los sensores del hardware y empaquetando la información garantizando así la transmisión de la telemetría en tiempo real y la estabilidad operativa del nodo y la capacidad real que tiene para mantener un enlace de transmisión de datos hacia el puente de borde.

El sistema permite ejecutar las rutinas sin ningún tipo de bloqueo lógico, además, la placa central es capaz de administrar los requerimientos a nivel de hardware del vehículo en todo el recorrido. En consecuencia, el comportamiento del nodo durante estas pruebas sirve como una base para validar la arquitectura como se observa en la figura 75, lo que demuestra la viabilidad técnica de la placa y dejando el sistema apto para realizar un análisis a los componentes individualmente.

Figura 75

Captura del monitor serial del ESP32-WROOM-32 durante su funcionamiento



Nota. Autoría propia

4.1.1. *Pruebas de adquisición de datos*

Respecto a la adquisición de las lecturas de los datos, la arquitectura del software opero correctamente mediante la utilización de rutinas de códigos no bloqueante sin la utilización de retardos o delays, con el fin de permitir que el ESP32 adquiera los datos simultáneamente y pueda mantener la transmisión MQTT sin colapsarse.

En la adquisición de datos del radar ultrasónico HC-SR04 se estableció un intervalo de 300 ms, pero de acuerdo al fragmento del código de la figura 76 se realizó una implementación de un filtro realizado por software que se basa en promedios para poder eliminar las lecturas erráticas y los ecos que se producen por el ruido del ambiente o rebote de la señal, así mismo, se implementó un promedio de cinco muestras continuas que ayudan a erradicar estos ecos. En la siguiente tabla 18 se puede observar como este filtro ayuda a reducir drásticamente los falsos positivos dentro del campo.

Tabla 18

Análisis de Precisión del Sensor de Proximidad

Configuración del sensor	Total, Detecciones	Falsos Positivos (%)	Detecciones Efectivas (%)

Sin Filtro (Radio 1.5m)	300,000+	> 60.0 %	< 40.0 %
Con Filtro Aplicado (Radio 1.0m)	65,527	12.3 %	87.7 %

Figura 76

Implementación del filtro de promedio para la señal ultrasónica.

```
// --- FILTRO DEL SENSOR ULTRASÓNICO ---
// Propósito: Mitigar lecturas erráticas y ecos fantasma del entorno.
// Método: Se toman 5 muestras consecutivas, se descartan valores nulos (ruido)
// y se calcula el promedio de las muestras válidas.
float getFilteredDistance() {
  float totalDist = 0;
  int validSamples = 0;
  // Bucle de muestreo rápido (5 iteraciones)
  for (int i = 0; i < 5; i++) {
    // Generación del pulso de disparo (Trigger)
    digitalWrite(TRIG_PIN, LOW); delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    // Lectura del eco con timeout de 30ms para evitar bloqueos del sistema
    long pulse = pulseIn(ECHO_PIN, HIGH, 30000);
    // Validación de la muestra (descarta pulsos en 0 por fallos de lectura)
    if (pulse > 0) {
      // Conversión de tiempo a distancia en metros (Vel. del sonido = 0.0343 cm/us)
      totalDist += (pulse * 0.0343 / 2) / 100.0;
      validSamples++;
    }
    // Breve pausa para disipar el eco físico antes del siguiente disparo
    delay(10);
  }
  // Retorna el promedio si hubo muestras válidas; caso contrario, retorna 99.0 (fuera de rango)
  return (validSamples > 0) ? totalDist / validSamples : 99.0;
}
```

Nota. Autoría propia

Durante las pruebas de encendido en frio del sistema (*Cold Start),* el módulo GPS NEO-6M tiene un retardo de entre uno y cinco minutos para realizar la triangulación para la

geolocalización, el sistema actúa ante este incidente mediante él envió de una alerta de espera de por la red, proceso que se evidencia en la figura 77.

Figura 77

Gestión de estados durante el arranque en frío del receptor NEO-6M.



Nota. Autoría propia

A pesar de estas pausas de procesamiento temporal de entre 50 y 100 ms generadas por las lecturas del sensor HC-SR04, estos tiempos no afectaron en la recolección de coordenadas del sensor NEO-6M, esto se debe a la implementación del buffer serial administrado por el hardware nativo del ESP32, por otro lado, para evitar la saturación del bus I2C, la adquisición inercial del sensor MPU6050 se fijó en ciclos de tiempo de 500 ms en donde permite capturar variaciones físicas bruscas.

Finalmente, con el objetivo de garantizar la sincronización de la topología de red local, el microordenador central necesita un margen de tiempo operativo para inicializar su red, para ello, el nodo central ejecuta un tiempo espera inicial de 40 ms otorgando un margen tiempo suficiente para que el microordenador central inicie su interfaz Wi-Fi y por ende despliegue su bróker MQTT. Para esto, se realizó una comparativa de tiempos de adquisición y transmisión en el nodo principal como se observa en la tabla 19.

Tabla 19

Parámetros temporales de adquisición y transmisión en el nodo ESP32

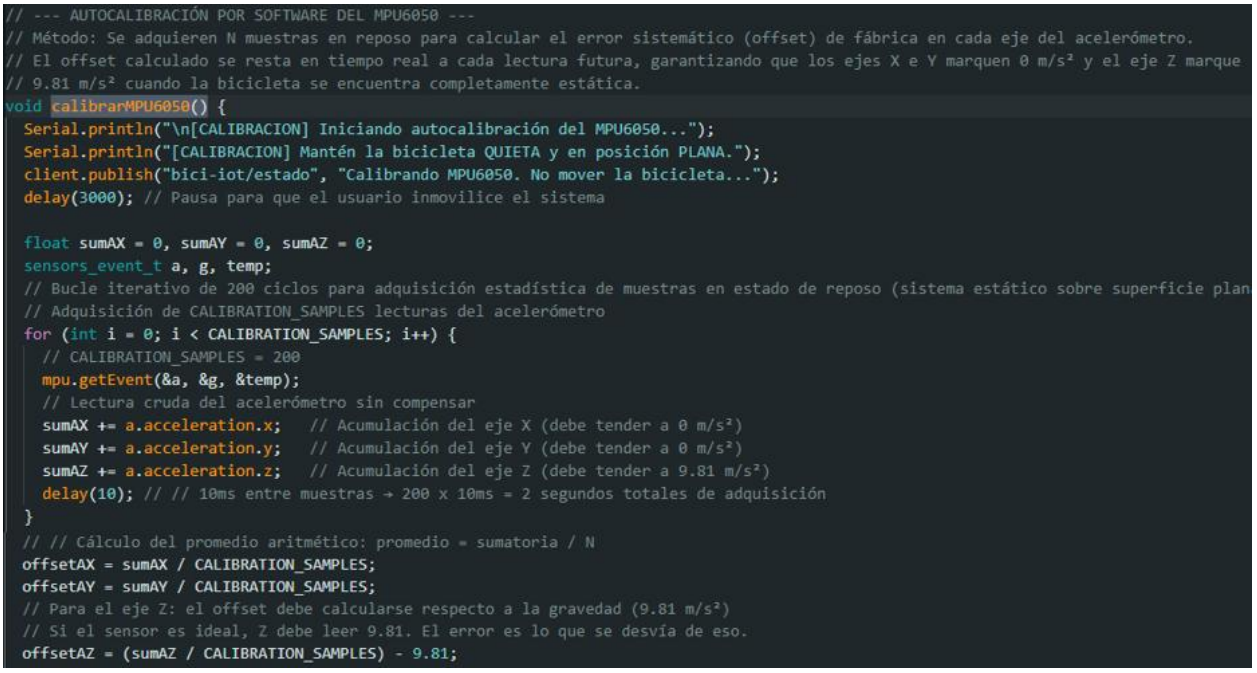
Hardware	Parámetro	Valor Asignado	Justificación
HC-SR04	Frecuencia de muestreo	300 ms	Equilibrio entre detección temprana y prevención de sobrecarga de ecos.
MPU6050	Frecuencia de muestreo	500 ms	Prevención de saturación en las líneas de comunicación del bus I2C.
NEO-6M	Frecuencia de muestreo	2000 ms	Tiempo suficiente para validar el *checksum* de tramas NMEA extensas.
Gateway Raspberry Pi	Margen de inicialización	40000 ms	Tiempo de espera para el levantamiento total de los servicios de red locales.
MQTT (ESP32)	*Cooldown* de alerta	15000 ms	Prevención de inundación de paquetes (*flooding*) ante múltiples detecciones seguidas.

4.1.2. *Calibración de Sensores*

Para aislar el ruido de los sensores, el microcontrolador ejecuta una rutina de auto calibración por software para el sensor MPU6050 (acelerómetro y giroscopio), en la cual se captura doscientas muestras en reposo durante un tiempo de dos segundos con el fin de calcular y restar el error sistemático u offset en tiempo real, como se observa en la figura 78.

Figura 78

Autocalibración por software para el cálculo de error del MPU6050



```
// --- AUTOCALIBRACIÓN POR SOFTWARE DEL MPU6050 ---
// Método: Se adquieren N muestras en reposo para calcular el error sistemático (offset) de fábrica en cada eje del acelerómetro.
// El offset calculado se resta en tiempo real a cada lectura futura, garantizando que los ejes X e Y marquen 0 m/s² y el eje Z marque
// 9.81 m/s² cuando la bicicleta se encuentra completamente estática.
void calibrarMPU6050() {
  Serial.println("\n[CALIBRACION] Iniciando autocalibración del MPU6050...");
  Serial.println("[CALIBRACION] Mantén la bicicleta QUIETA y en posición PLANA.");
  client.publish("bici-iot/estado", "Calibrando MPU6050. No mover la bicicleta...");
  delay(3000); // Pausa para que el usuario inmovilice el sistema

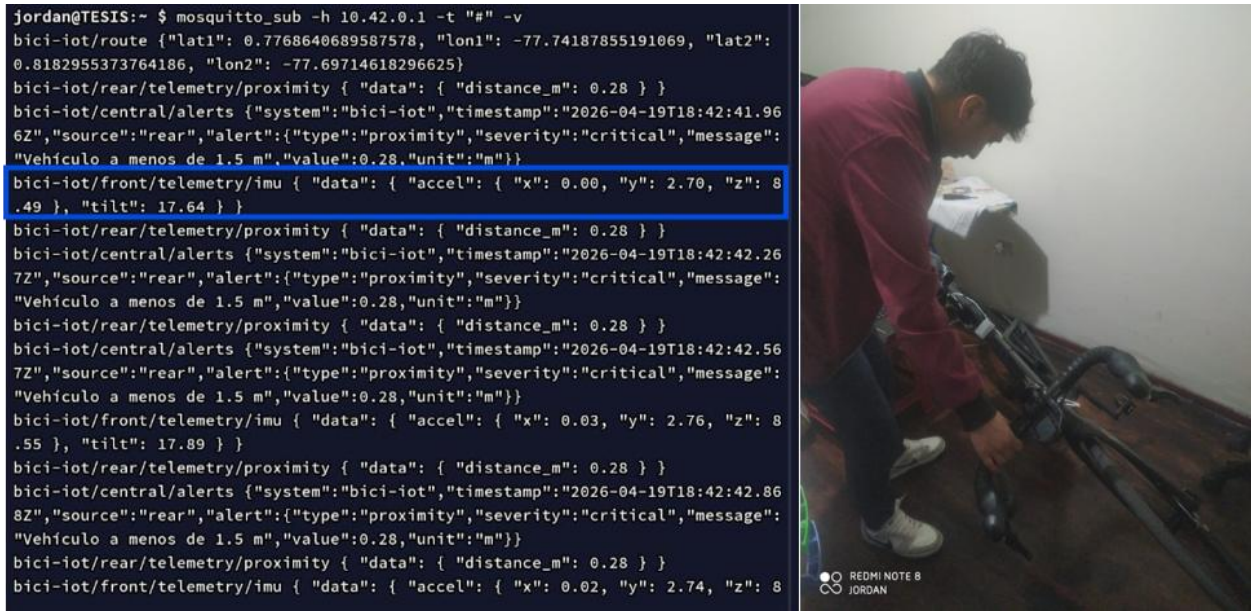
  float sumAX = 0, sumAY = 0, sumAZ = 0;
  sensors_event_t a, g, temp;
  // Bucle iterativo de 200 ciclos para adquisición estadística de muestras en estado de reposo (sistema estático sobre superficie plan
  // Adquisición de CALIBRATION_SAMPLES lecturas del acelerómetro
  for (int i = 0; i < CALIBRATION_SAMPLES; i++) {
    // CALIBRATION_SAMPLES = 200
    mpu.getEvent(&a, &g, &temp);
    // Lectura cruda del acelerómetro sin compensar
    sumAX += a.acceleration.x;   // Acumulación del eje X (debe tender a 0 m/s²)
    sumAY += a.acceleration.y;   // Acumulación del eje Y (debe tender a 0 m/s²)
    sumAZ += a.acceleration.z;   // Acumulación del eje Z (debe tender a 9.81 m/s²)
    delay(10); // // 10ms entre muestras → 200 x 10ms = 2 segundos totales de adquisición
  }
  // // Cálculo del promedio aritmético: promedio = sumatoria / N
  offsetAX = sumAX / CALIBRATION_SAMPLES;
  offsetAY = sumAY / CALIBRATION_SAMPLES;
  // Para el eje Z: el offset debe calcularse respecto a la gravedad (9.81 m/s²)
  // Si el sensor es ideal, Z debe leer 9.81. El error es lo que se desvía de eso.
  offsetAZ = (sumAZ / CALIBRATION_SAMPLES) - 9.81;
```

Nota. Autoría propia

Ante la imposibilidad técnica de simular caídas reales, la validación física del sensor MPU6050 se realizó mediante la implementación del módulo en el cuadro de la bicicleta y llegando a realizar inclinaciones manuales para poder comprobar las los ángulos que arroja en la consola del microcontrolador central, como se ilustra en la figura 79.

Figura 79

Validación física de inclinación del MPU6050 en consola local



Nota. Autoría propia

Las pruebas del sensor HC-SR04 se rigieron a una validación de tipo cualitativa en campo (véase en la figura 80), donde los datos que son enviados por MQTT permiten reflejar datos coherentes al aplicar el factor de conversión de velocidad del sonido $(0.0343\ cm/\mu s)$ frente a una temperatura ambiente estima en $20^{\circ}C$, lo que a nivel de funcionalidad, las pruebas del sensor HC-SR04 demostraron tener una precisión suficiente para activar las alertas visuales y las alertas lógicas que se activan exactamente al cruzar el umbral de seguridad de un metro como se observa en el fragmento de código de la figura 81.

Figura 80

Algoritmo de filtrado y conversión de señal del sensor HC-SR04

```
// Propósito: Mitigar lecturas erráticas y ecos fantasma del entorno.
// Método: Se toman 5 muestras consecutivas, se descartan valores nulos (ruido)
// y se calcula el promedio de las muestras válidas.
float getFilteredDistance() {
  float totalDist = 0;
  int validSamples = 0;
  for (int i = 0; i < 5; i++) {
    // Generación del pulso de disparo (Trigger)
    digitalWrite(TRIG_PIN, LOW); delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    // Lectura del eco con timeout de 30ms para evitar bloqueos del sistema
    long pulse = pulseIn(ECHO_PIN, HIGH, 30000);
    // Validación de la muestra (descarta pulsos en 0 por fallos de lectura)
    if (pulse > 0) {
      // Conversión de tiempo a distancia en metros (Vel. del sonido = 0.0343 cm/us)
      // Fórmula: d = (pulso_us x 0.0343 cm/us) / 2 / 100
      // 0.0343 = velocidad del sonido a 20°C en cm/us
      // Se divide entre 2 porque el pulso es el viaje de ida y vuelta
      totalDist += (pulse * 0.0343 / 2) / 100.0;
      validSamples++;
    }
    // Breve pausa para disipar el eco físico antes del siguiente disparo
    delay(10);
  }
  // Retorna el promedio si hubo muestras válidas; caso contrario retorna 99.0 (fuera de rango)
  return (validSamples > 0) ? totalDist / validSamples : 99.0;
}
```

Nota. Autoría propia

Figura 81

Gestión de alertas ante detección de objetos en zona de riesgo

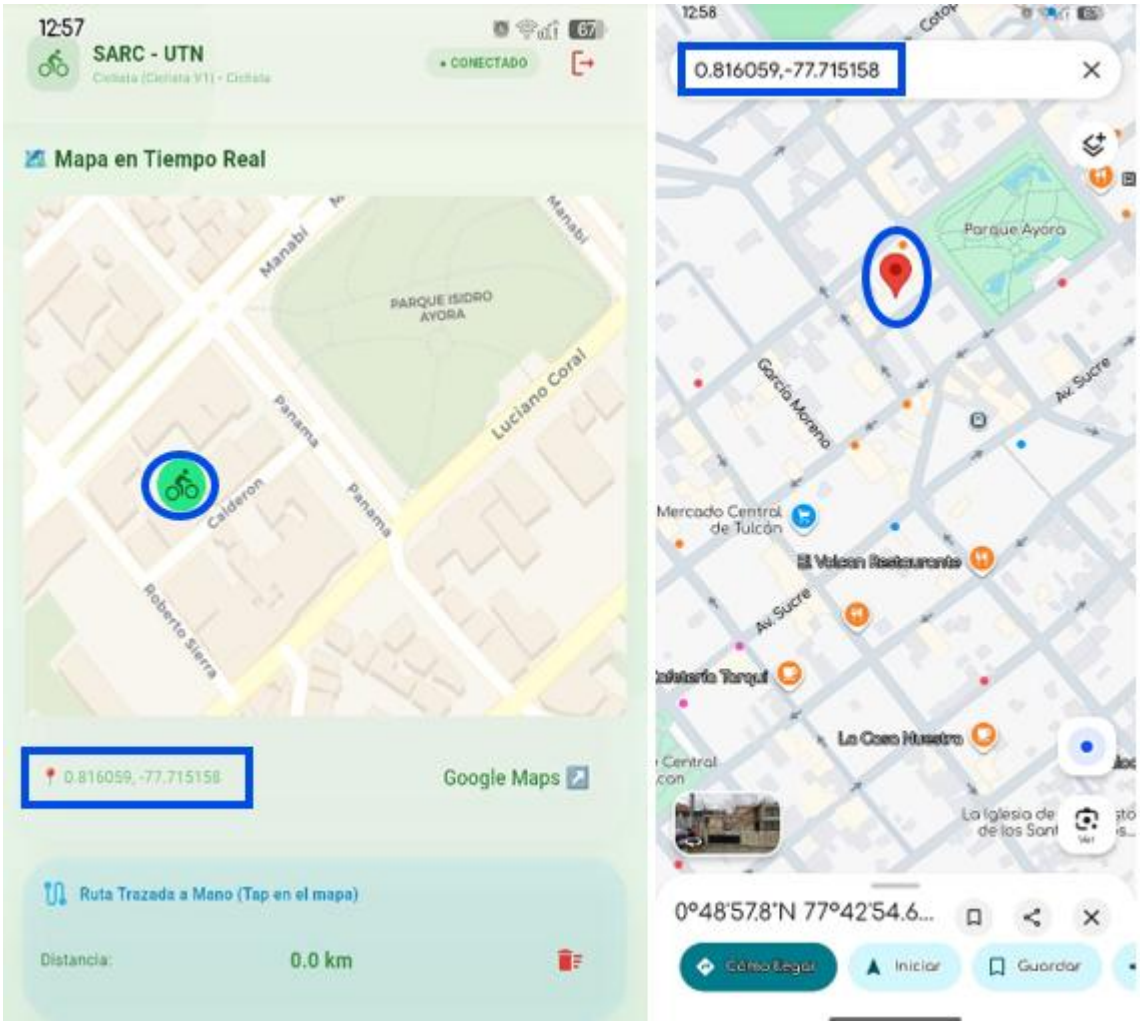
```
// --- 1. ADQUISICIÓN DE DATOS: RADAR DE PROXIMIDAD ---
// Temporizador no bloqueante: Ejecuta la alerta cada 300 ms
if (currentMillis - prevProxMillis >= proxInterval) {
  prevProxMillis = currentMillis;
  // Llamada a la función con el filtro de ruido por sotfware
  float dist = getFilteredDistance();
  // Evalucación de la zona de peligro (Rango: 0 a 2.0 metros )
  bool objetoCerca = (dist > 0 && dist <= 2.0);
  String proxData = "{ \"data\": { \"distance_m\": " + String(dist) + " } }";
  client.publish("bici-iot/rear/telemetry/proximity", proxData.c_str());
  // MANEJO DE LA TIRA LED NEOPIXEL
  if (objetoCerca && alertasVisualesActivas) {
    // Pinta la tira de color ROJO intenso
    for(int i=0; i<NUMPIXELS; i++) {
      tiraLed.setPixelColor(i, tiraLed.Color(255, 0, 0));
    }
    tiraLed.show();
  } else {
    // Apaga la tira si no hay peligro
    tiraLed.clear();
    tiraLed.show();
  }
  // Activación de alerta lógica (MQTT) con cooldown de 15 segundos
  if (objetoCerca && (currentMillis - lastProxAlert >= proxCooldown)) {
    String payload = "{ \"alert\": { \"type\": \"proximity\", \"val\": " + String(dist) + " } }";
    client.publish("bici-iot/rear/alerts", payload.c_str());
    Serial.println("-> ALERTA: Proximidad detectada (<2m), enviando al Broker...");
    lastProxAlert = currentMillis;
```

Nota. Autoría propia

Finalmente, debido a la interferencia electromagnética (EMI) detectada en las pruebas del sensor NEO-6M generadas de las placas principales lo que impidió realizar la triangulación satelital, fue necesario reubicar el sensor y sacarlo al chasis de la bicicleta, el margen de error detectado en el posicionamiento estático fue de 2.5 metros a 3 metros con referencia a la plataforma Google Maps como se observa en la figura 82.

Figura 82

Comparación de la ubicación GPS del módulo NEO-6M con Google Maps.



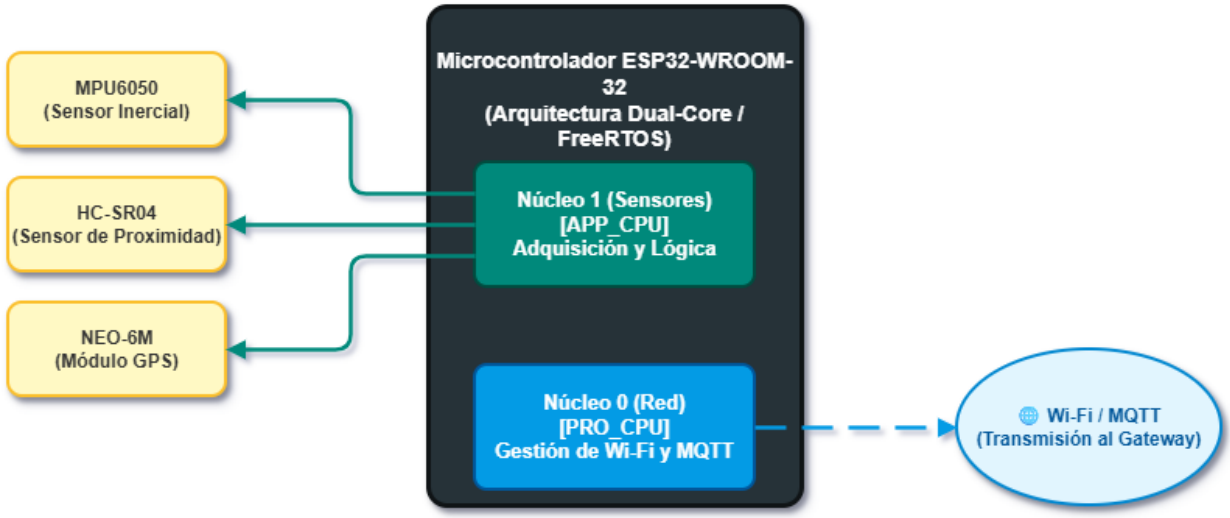
Nota. Autoría propia

4.1.3. *Rendimiento del microcontrolador (ESP32)*

Al evaluar la carga térmica y el consumo del microcontrolador, no presento incrementos térmicos que sean perceptibles al tacto durante el funcionamiento en las pruebas prolongadas, esto debido a que se implementaron temporizadores asíncronos llegando así a mantener estable la memoria RAM sin provocar reinicios por *Watchdog,* aprovechando la capacidad multitarea del hardware, permitió tener una gestión concurrente sin lentitud en la trasmisión, llegando así a delegar la conexión inalámbrica Wi-Fi y MQTT a un núcleo y la parte de lectura secuencial de los sensores al otro como se detalla en la figura 83.

Figura 83

Arquitectura de doble núcleo del microcontrolador ESP32-WROOM-32

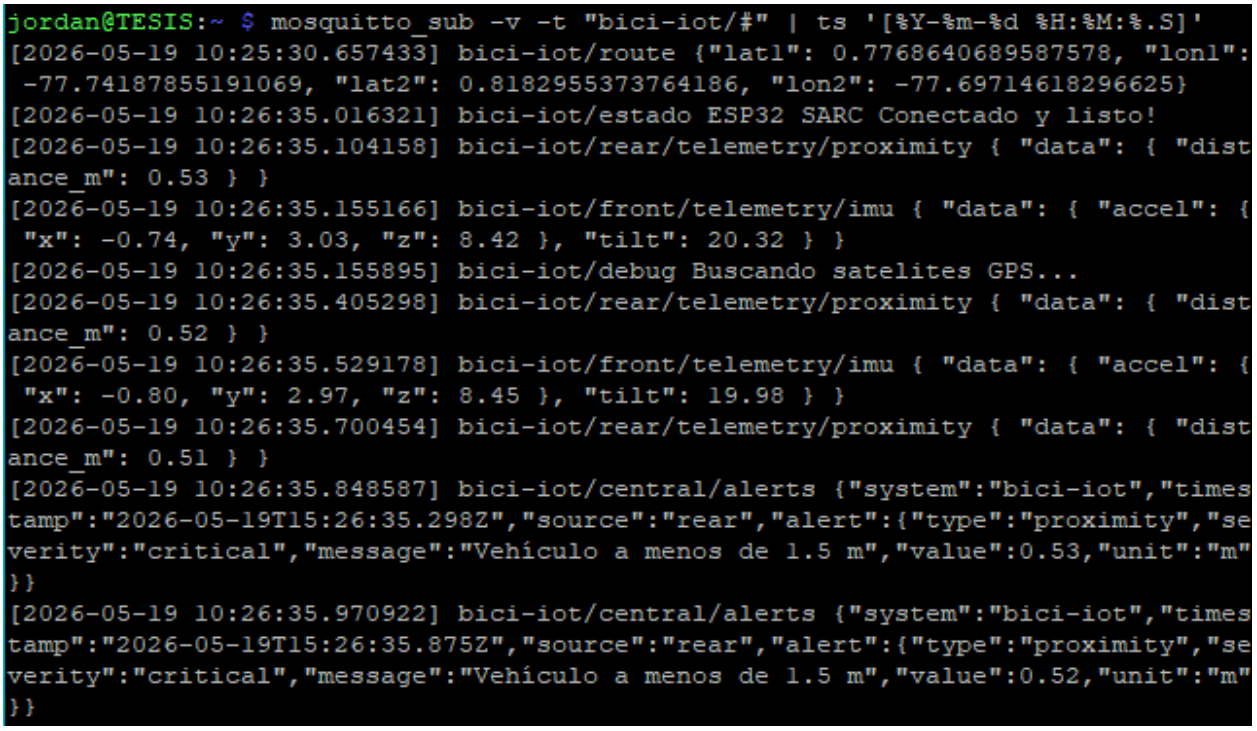


Nota. Autoría propia

Esta distribución de tareas internas permitió disminuir la carga del microcontrolador, donde la telemetría proveniente de los sensores MPU6050 muestreada cada 500 ms y las coordenadas del NEO-6M que son extraídas cada 2 segundos lograron entrar al puente del microordenador de forma paralela y con una fluidez total como se observa en la figura 84.

Figura 84

Recepción de mensajes MQTT en la Raspberry Pi con marcas de tiempo (ms).



```
jordan@TESIS:~ $ mosquitto_sub -v -t "bici-iot/#" | ts '[%Y-%m-%d %H:%M:%.S]'
[2026-05-19 10:25:30.657433] bici-iot/route {"lat1": 0.7768640689587578, "lon1":
 -77.74187855191069, "lat2": 0.8182955373764186, "lon2": -77.69714618296625}
[2026-05-19 10:26:35.016321] bici-iot/estado ESP32 SARC Conectado y listo!
[2026-05-19 10:26:35.104158] bici-iot/rear/telemetry/proximity { "data": { "dist
ance_m": 0.53 } }
[2026-05-19 10:26:35.155166] bici-iot/front/telemetry/imu { "data": { "accel": {
 "x": -0.74, "y": 3.03, "z": 8.42 }, "tilt": 20.32 } }
[2026-05-19 10:26:35.155895] bici-iot/debug Buscando satelites GPS...
[2026-05-19 10:26:35.405298] bici-iot/rear/telemetry/proximity { "data": { "dist
ance_m": 0.52 } }
[2026-05-19 10:26:35.529178] bici-iot/front/telemetry/imu { "data": { "accel": {
 "x": -0.80, "y": 2.97, "z": 8.45 }, "tilt": 19.98 } }
[2026-05-19 10:26:35.700454] bici-iot/rear/telemetry/proximity { "data": { "dist
ance_m": 0.51 } }
[2026-05-19 10:26:35.848587] bici-iot/central/alerts {"system":"bici-iot","times
tamp":"2026-05-19T15:26:35.298Z","source":"rear","alert":{"type":"proximity","se
verity":"critical","message":"Vehículo a menos de 1.5 m","value":0.53,"unit":"m"
}}
[2026-05-19 10:26:35.970922] bici-iot/central/alerts {"system":"bici-iot","times
tamp":"2026-05-19T15:26:35.875Z","source":"rear","alert":{"type":"proximity","se
verity":"critical","message":"Vehículo a menos de 1.5 m","value":0.52,"unit":"m"
}}
```

Nota. Autoría propia

Finalmente, el bajo impacto de retardo en referencia a lo acústico del HC-SR04 sobre el rendimiento del microcontrolador y la recepción ininterrumpida de los datos del NEO-6M en la consola del microordenador, permite la transmisión fluida de las alertas de inclinación descartando así problemas como el desbordamiento de la memoria del microcontrolador.

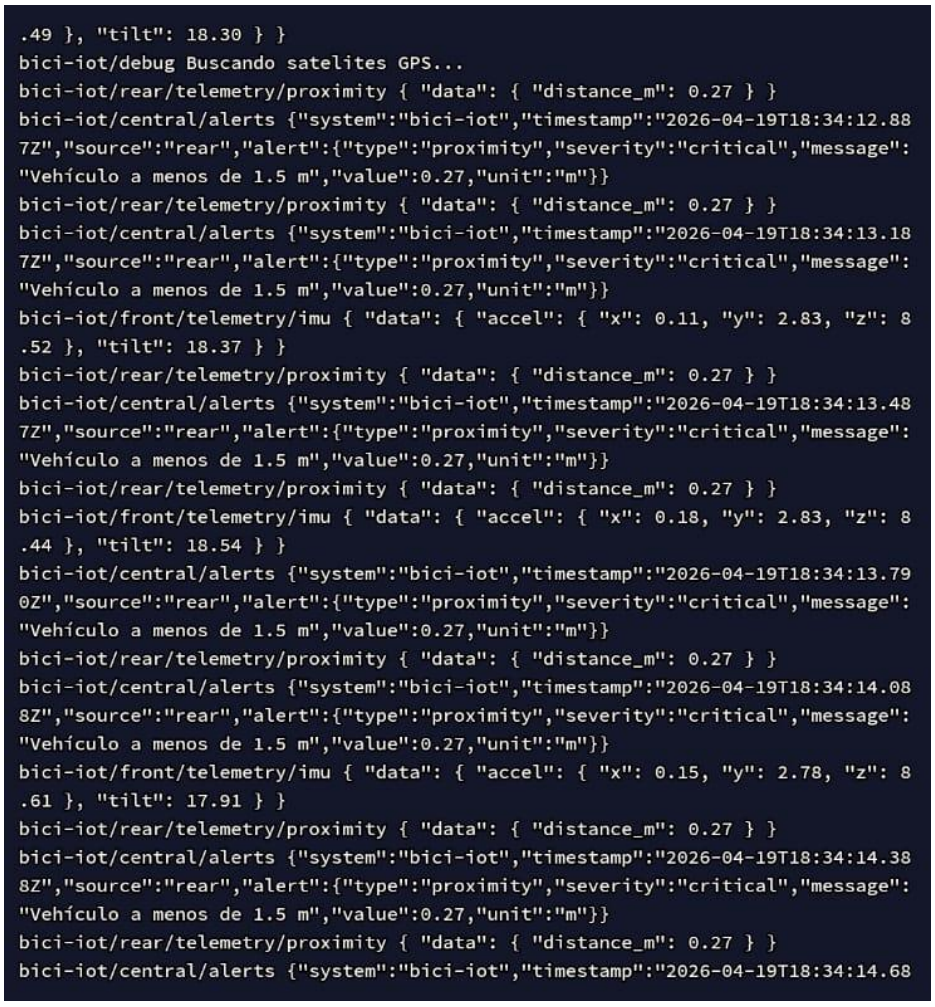
4.1.4. *Pruebas de transmisión inalámbrica*

Durante la prueba de estrés de conectividad inalámbrica, el microcontrolador garantizo niveles óptimos de señal anulando completamente cualquier atenuación generada por el cuerpo del ciclista o chasis, esto se debe a que el microcontrolador y el microordenador están ubicados en la misma carcasa a una distancia de entre 5 y 10 centímetros. Al analizar el entorno de propagación inalámbrica, la creación de un SSID cerrado “SARC-AP” en el microordenador permitió aislar el tráfico de red local, con esto se evita, tener colisiones de canales de dispositivos que provengan

desde el exterior y asegurando así que los paquetes MQTT de los sensores lleguen con una tasa de pérdida del 0% como se observa en la figura 85.

Figura 85

Monitoreo del tráfico MQTT en tiempo real desde la Raspberry Pi



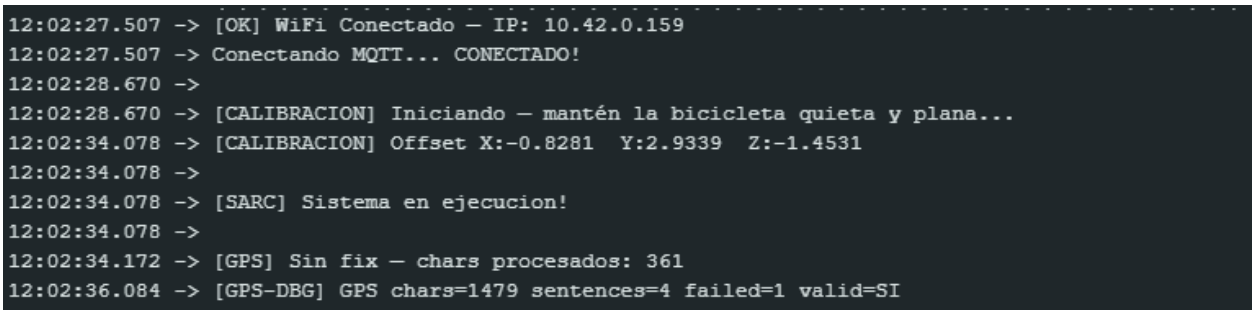
Nota. Autoría propia

Para validar la eficiencia y estabilidad del enlace inalámbrico, se apagó deliberadamente el enlace entre el microordenador y el microcontrolador simulando así un fallo de energía total, lo que provoco así que el microcontrolador realice una reconexión estructurada en 80 intentos durante un periodo de tiempo de 40 segundos esto con el fin de sobrecalentar su radio Wi-Fi. Una vez

restaurada la energía, el nodo microcontrolador es capaz de detectar la red local y lograr conectarse y enlazarse nuevamente al broker MQTT, esto lo realiza en menos de 5 segundos de manera autónoma, esto se observa en la figura 86.

Figura 86

Reconexión automática del ESP32 al broker MQTT tras pérdida de red.



Nota. Autoría propia

Finalmente, esta capacidad de reconexión automática entre el microcontrolador y microordenador es de crucial importancia para que el sistema pueda retomar las alertas de los sensores sin que requiera una intervención manual por parte del ciclista en medio de la ruta.

4.2 Resultados del funcionamiento del microordenador (Raspberry Pi Zero 2W)

Como núcleo de procesamiento central, el microordenador asume la función operativa de puente de borde (Edge Gateway) y a la vez de servidor local dentro de la arquitectura física del sistema SARC. La principal tarea de este microordenador es de recibir ininterrumpidamente la telemetría adquirida por el microcontrolador con el fin de procesarla a nivel local mediante el despliegue del broker MQTT.

A partir del procesamiento interno que realiza, el microordenador es el encargado de organizar la comunicación y el enrutamiento de los paquetes de datos entre la red local y la red global que se encuentra alojada la base de datos en la nube. Con el fin de validar la capacidad

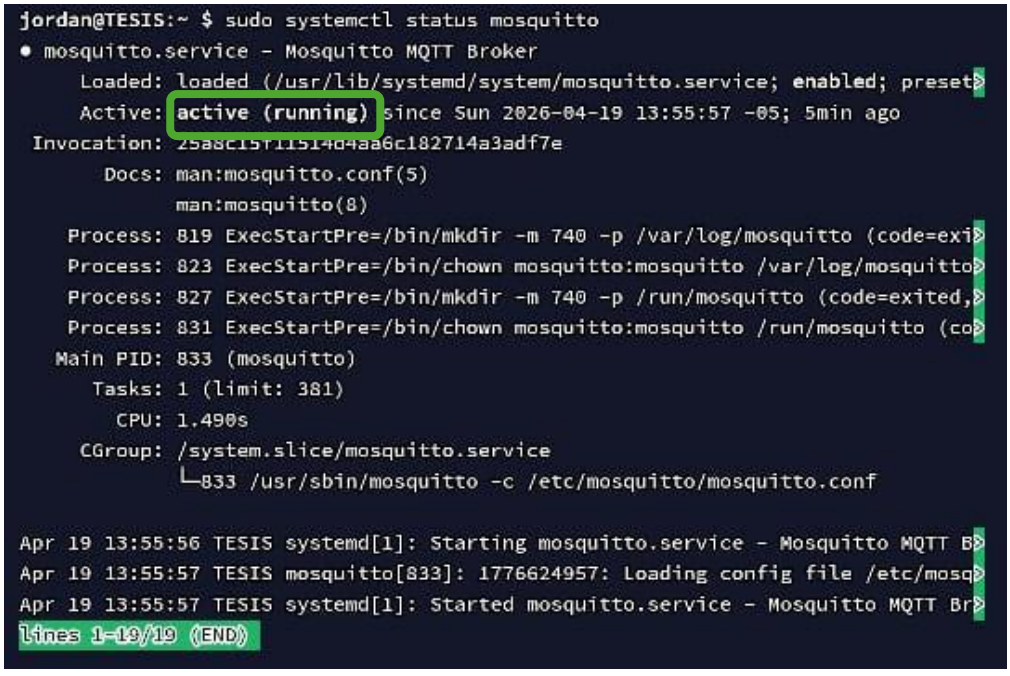
operativa de este hardware como un puente de red, los siguientes subapartados tiene la finalidad de evaluar las métricas específicas como su carga de procesamiento, su eficiencia en la gestión de red y su estabilidad en ciclos de trabajos continuos.

4.2.1. *Despliegue de servicios locales*

Para la organización de datos, el microordenador fue centralizo para permitir el despliegue del servidor de mensajería Eclipse Mosquitto, en donde se realizó habilitaciones en el sistema de cortafuegos o firewall necesarios con el objetivo de permitir la publicación continua de los datos recibidos desde el nodo de percepción (ESP32). Este estado de ejecución se puede observar en la figura 87.

Figura 87

Comprobación de estado del servicio Eclipse Mosquitto



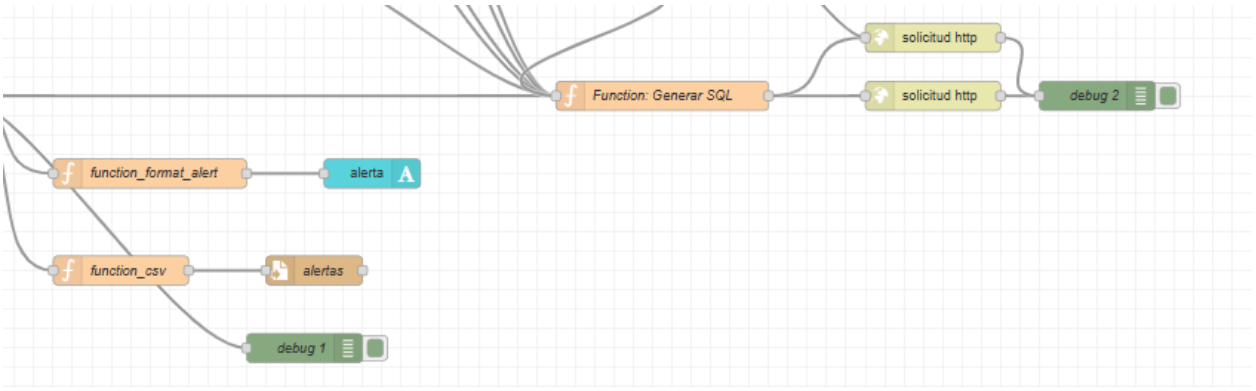
Nota. Autoría propia

Como motor de orquestación lógico para la mensajería, Node-Red asumió ese rol esto para que realice el consumo de las tramas de JSON provenientes desde el broker MQTT con el fin de

permitir la ejecución de peticiones de formato HTTP/REST las cuales están orientadas a los registros que llegan al gestor de datos en la nube Supabase y mediante esto permita disparar las alertas a través de la API Twilio, de este modo, la lógica de enrutamiento y el escalamiento a la nube permitió operar de forma paralela junto a una interfaz desarrollada en el software de Python, en donde cuya integración se consolida en la figura 88.

Figura 88

Diagrama de flujo en Node-Red, para peticiones HTTP



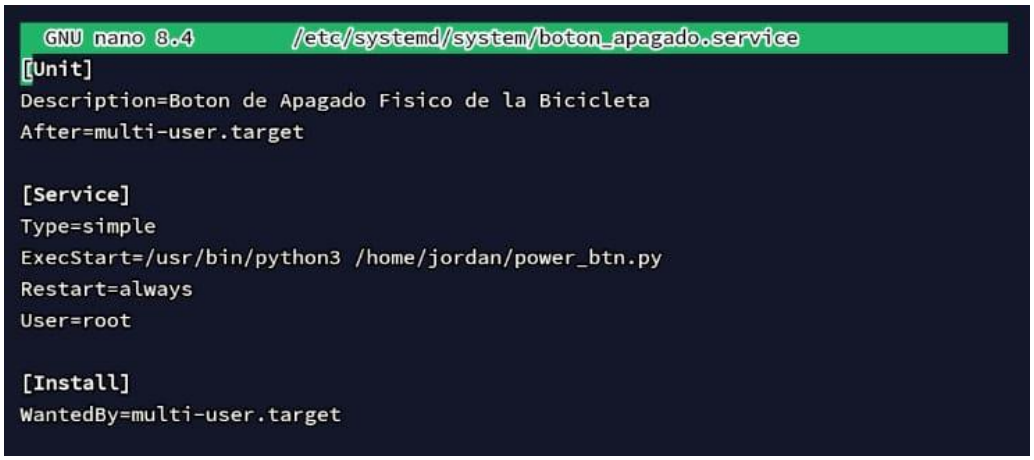
Nota. Autoría propia

Finalmente para garantizar un el funcionamiento sin interrupciones del sistema, se realizó un descarte de la intervención de ejecuciones manuales provenientes desde la terminal del microordenador, para esto se realizó una encapsulación del entorno (Eclipse Mosquitto, Node-RED y los módulos en Python) bajo un archivo de configuración de servicio administradas por el gestor systemd de Linux, la política de reinicio continuo que está gestionada por este mismo, cuya estructura de la configuración esta visualmente destalla en la figura 89, permite asegurar que los procesos vuelvan a funcionar como demonios en segundo plano sin realizar esfuerzos máximos en el microordenador y de forma autónoma que actúan ante cortes energéticos o vibraciones extremas

de la bicicleta, con esto garantizamos el funcionamiento autónomo del sistema sin la intervención humana.

Figura 89

Configuración del servicio systemd para la gestión de reinicio automático



Nota. Autoría propia

4.2.2. *Rendimiento del microordenador*

En cuanto al consumo de memoria física, el microordenador demostró ser completamente estable al momento de ejecutar procesos simultáneos y en segundo plano servicios como Mosquitto, Node-Red y el script de control de Python, siendo así una garantía al momento de procesar los datos provenientes de la telemetría sin generar cuellos de botellas. A nivel del sistema operativo, la gestión que realiza internamente para la memoria virtual se realizó de forma automática mediante los parámetros por defecto, con esto se logró que el espacio de intercambio (Swap) resista a la carga de estos procesos sin realizar una intervención de ampliaciones manuales o de una caída del sistema.

A pesar de tener un encapsulamiento físico impreso en 3D, el microordenador no mostro evidencias de síntomas térmicos, con esto permite mantener un rendimiento fluido de la unidad central de procesamiento bajo trabajos continuos. Para consolidar este análisis térmico, se realizó

una evaluación matemática partiendo de una muestra n mediciones extradías mediante la utilización de herramientas nativas del sistema (véase en la figura 90), además, se definió la temperatura en ese instante de mayor estrés térmico operativo en una ecuación (5):

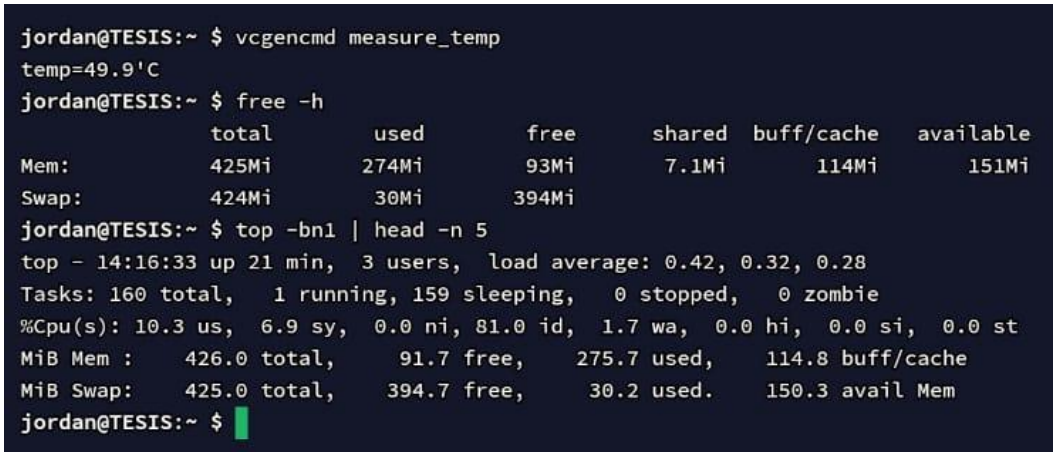
$$T_{max} = \max(T_1, T_2, T_3 \ldots, T_n) \quad (5)$$

Y la temperatura en estado de reposo o estrés térmico mínimo con la ecuación (6):

$$T_{min} = \min(T_1, T_2, T_3 \ldots, T_n) \quad (6)$$

Figura 90

Medición de temperatura del microprocesador bajo operación continua



Nota. Autoría propia

Para garantizar la estabilidad del sistema se extrajeron los datos térmicos del sistema bajo estrés continuo, obteniendo los siguientes resultados como se observa en la tabla 20.

Tabla 20

Comportamiento Térmico y Rendimiento del Microordenador

Estado del Hardware	Temperatura CPU	Consumo RAM	Estado de la placa

En Reposo (Standby)	~ 35.0 °C	~ 114 MiB	Estable
En Carga Máxima (Estrés)	49.9 °C	274 MiB (Max 425 MiB)	Operativo

Finalmente, para completar este análisis, el comportamiento térmico general del microordenador se consolida bajo el cálculo de la temperatura promedio expresado en la ecuación:

$$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i \qquad (7)$$

En donde T_i representa la temperatura térmica individual registrada en la base temporal, lo que confirma el buen manejo térmico del diseño de la carcasa, cuyos valores se pueden observar en la tabla 21.

Tabla 21

Registro de temperaturas del microordenador durante operación continua

Medición (i)	**Temperatura $T_i(°C)$**	**Condición operativa**
1	41.2	Inicio del sistema (reposo)
2	46.8	Mosquitto activo
3	49.5	Node-RED procesando
4	52.4	Script Python + MQTT activo
5	56.8	Telemetría GPS en transmisión
6	53.0	Alerta crítica procesada (pico)

7	53.5	Carga moderada
8	45.2	Estabilización
9	40.1	Reducción de carga
10	56.8	Reposo (post-operación)
$\boldsymbol{T_{max}}$	38.5	Estrés máximo registrado
$\boldsymbol{T_{min}}$	47.7	Estrés mínimo registrado
$\boldsymbol{\overline{T}}$	41.2	Temperatura promedio

4.2.3. *Pruebas de enlace LTE (Módulo EG25-G)*

Respecto a la conectividad móvil externa, el microordenador realizo una delegación de transmisión al modem EG25-G en donde se utilizó la infraestructura del operador Claro, la cual fue seleccionada estratégicamente por sus amplias coberturas en zonas periféricas, consiguientemente, como el peso de las tramas de telemetría (JSON) enviadas por el microcontrolador resulto ser tan liviano que el enlace 4G no tiene saturaciones y permite tiempos de envió muy rápidos, de esta manera, al realizar estas pruebas se pudo comprobar que la red 4G era lo suficientemente rápido para una emergencia.

Con el fin de obtener datos reales y no teóricos solamente, se realizaron pruebas de conectividad apoyados en el protocolo ICMP/PING, para obtener la latencia promedio del sistema de alertas se aplicó la siguiente ecuación:

$$\bar{L} = \frac{1}{m}\sum_{j=1}^{m} L_j \qquad (8)$$

En donde L_j es el retardo medido en milisegundos de cada paquete enviado y m representa la cantidad de paquetes enviados y evaluados en la ruta, para ello, se enviaron 50 paquetes hacia

la nube con esto se pudo medir el tiempo de viaje a cada uno los resultados de esta prueba se puede observar en la figura 91, aplicando esta métrica se obtuvo un retardo promedio con el que se puede concluir que el sistema SARC tiene una respuesta que no está lejana de una red domestica afirmando así que las alertas de críticas llegaran a tiempo. En resumen, estas configuraciones, latencia observada y las justificaciones técnicas se detallan a continuación en la tabla 22.

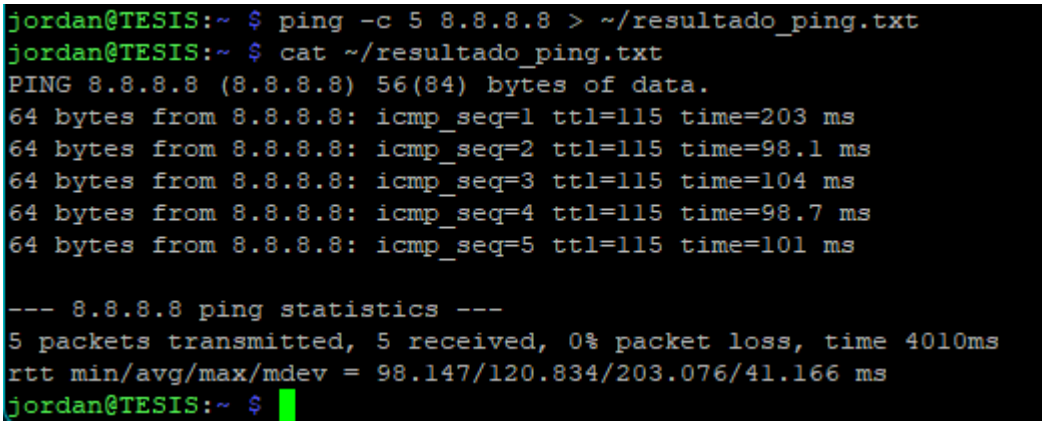
Tabla 22

Latencia promedio del sistema en diferentes zonas

Zona de Prueba	Cobertura	Latencia Promedio	Envió a la nube
Zonas Urbanas	Alta (4G)	2.5 a 4.1 s	Inserción exitosa
Ruta Periférica (E-35)	Media/Inestable	6.4 a 9.5 s	Inserción exitosa
Zonas Rurales	Nula	> 15.0 s	Guardado en caché local

Figura 91

Latencia de red LTE hacia servidores en la nube mediante protocolo ICMP

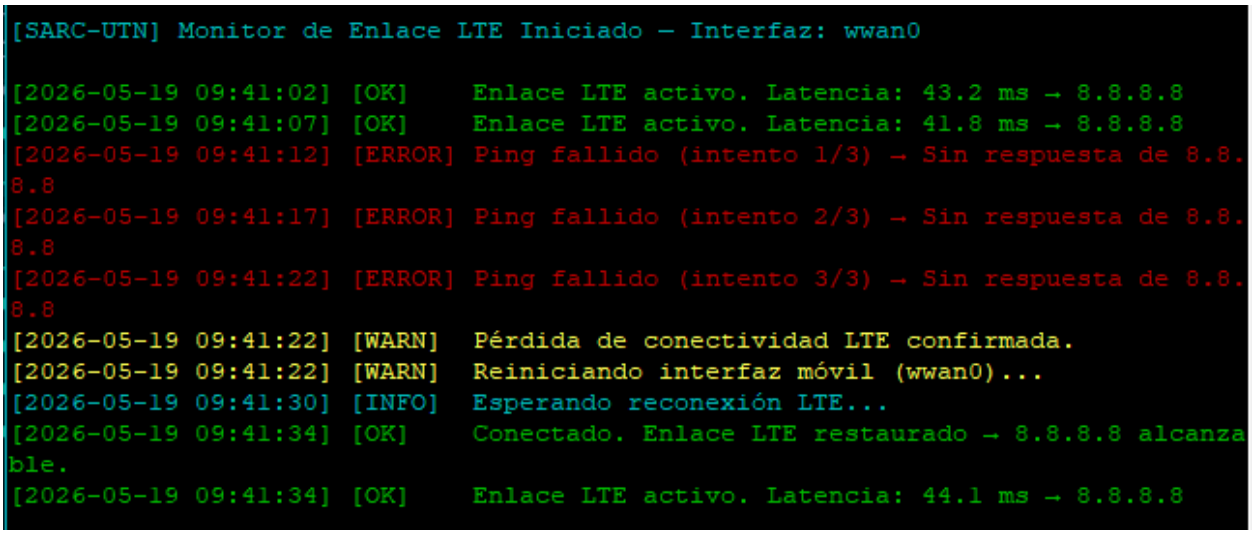


Nota. Autoría propia

Finalmente para la inicialización del enlace, fue necesario diseñarlo para operar de forma desatendida, esto se llevó a cabo mediante la implementación de rutinas de automatización que configuran el punto de acceso (APN) de la red móvil que actúa inmediatamente después de haber encendido el microordenador, así mismo, durante las pruebas de conexión en movilidad, se detectaron breves caídas de conexión proveniente al cambio físico entre torres celulares (handoff), para solucionar esta interrupción se realizó una mitigación mediante el apoyo de un script de monitoreo persistente que evalúa la salud del enlace (véase en la figura 92).

Figura 92

Ejecución de monitoreo persistente y recuperación autónoma del enlace LTE.



Nota. Autoría propia

4.3 Resultados del sistema de alimentación y autonomía

Para comprobar la autonomía del sistema, recae estructuralmente sobre la capacidad energética y su independencia, en donde, se exige un diseño de hardware capaz de sostener la telemetría y el enrutamiento sin requerir conexiones a fuentes de alimentación estáticas. El análisis

de la demanda de corriente hacia el sistema constituye como eje central de esta validación, en el cual se abarca a la alimentación de toda la capa percepción en la que abarca el nodo conectado con los sensores de telemetría y además del puente de enlace de borde conformado por el microordenador y el módulo LTE.

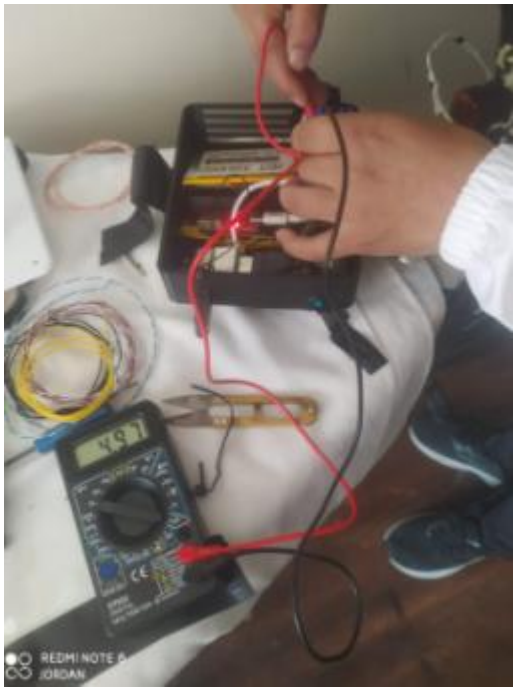
Para garantizar la protección activa en escenarios de ruta prolongados, sus apartados posteriores permiten evaluar con referencia a las baterías de litio frente a la carga del procesamiento local.

4.3.1. *Medición de consumo energético*

Con lo que respecta a la validación de energía del sistema, esta se fundamenta principalmente en una fuente de batería de una celda de litio de 10,000 mAh a 3.7 V nominales, cuya descarga es administrada por el módulo elevador de voltaje IP5328P con una función de Step-UP con el objetivo de garantizar un bus de alimentación estable de 5V y con una capacidad de entrega de hasta 3 A (véase figura 93).

Figura 93

Medición de voltaje estabilizado mediante multímetro digital.



Nota. Autoría propia

Mediante el uso del multímetro digital, permitió realizar la validación de magnitudes eléctricas en los puntos críticos del sistema, comprobando así la estabilidad del voltaje de salida del IP5328P se mantiene constante a la demanda de corriente incluso al momento de buscar o enlazarse a la estación base que realiza el módulo LTE. El análisis del consumo energético total del sistema se basa en la suma de consumo de cada componente en su estado de operación activa, definida por la ecuación matemática de la corriente total del sistema:

$$I_{Total} = I_{RPi} + I_{ESP} + I_{LTE} + I_{Sensores} \qquad (9)$$

donde se contabilizan los consumos individuales de los componentes como el microordenador, microcontrolador y modulo LTE.

Consecuentemente, para realizar la cuantificación de tasa de la trasferencia de corriente y con el fin de asegurar que el módulo IP5328P funcione correctamente dentro de sus límites térmicos y eléctricos, se aplicó la ecuación de potencia eléctrica:

$$P = V * I \qquad (10)$$

confirmando así que el flujo de corriente de 3 A (ver tabla 23) es suficiente para sostener el funcionamiento concurrente de todos los componentes que conforman el sistema.

Tabla 23

Análisis de Consumo Energético en Estado Activo (5V)

Hardware	Símbolo	Corriente Medida (mA)	Potencia Calculada (W)
Microordenador (Raspberry Pi)	I_{RPi}	600 mAh	3.00 W
Microcontrolador (ESP32)	I_{ESP}	600 mAh	3.00 W
Módem (Quectel EG25-G LTE)	I_{LTE}	100 mAh	0.50 W
Sensores (LED)	$I_{Sensores}$	250 mAh	1.25 W
Consumo Máximo del Sistema	I_{Total}	1550 mAh	7.75 W

4.3.2. *Prueba de autonomía*

A partir de las pruebas de consumo energético en campo, se consideraron en recorridos de entre 3 a 4 horas de operación continua, en donde se observó el funcionamiento energético de los elementos microordenador, microcontrolador, modulo LTE y sensores en donde el sistema fue capaz de mantener una estabilidad absoluta sin interrupciones en el suministro de energía. Inicialmente se optó por usar el módulo TP4056 para el sistema, pero dado a que el sistema sufría

de apagones esto debido a su salida de corriente que tenía como límite de 1A, lo que resultaba insuficiente para cubrir los picos de demanda del sistema, se corrigió con la utilización del módulo IP5328P, el cual soporta 3A, lo que garantiza la operabilidad sin ningún inconveniente.

Finalmente, para la proyección del tiempo de funcionamiento constante se basa en la relación entre la carga almacenada y el consumo promedio del sistema, que se expresa mediante la ecuación:

$$T_{(h)} = \frac{C_{(mAh)}}{I_{total(mA)}} * n \qquad (11)$$

En donde $C_{(mAh)}$ esta representado por la capacidad nominal de la batería (10000 mAh) y n es el factor de eficiencia que está regido por el módulo elevador IP5328P, este análisis ajustado a la eficiencia térmica y de conservación que fue probada en $n = 0.9$, permite calcular el equipo mediante condiciones continuas de uso. Los resultados se pueden observar en la tabla 24.

Tabla 24

Proyección de Autonomía del Sistema SARC (batería 10000 mAh)

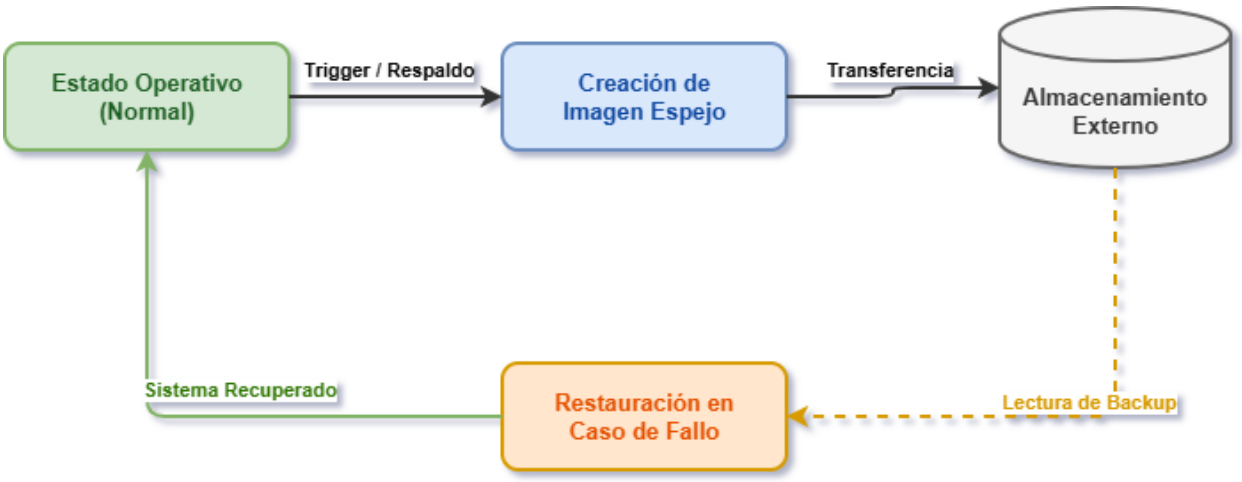
Escenario	**Corriente Estimada** $I_{total(mA)}$	**Capacidad útil** $C_{(mAh)}$	**Autonomía Proyectada** $T_{(h)}$
Demanda Máxima	$1550\ mAh$	9000 mAh	~ 5.8 horas
Operación Promedio	$700\ mAh$	9000 mAh	~ 10.0 horas
Modo Reposo / Sleep	$270\ mAh$	9000 mAh	~ 30.0 horas

4.4 Resultados de la comunicación con la nube

Al evaluar la eficiencia del enlace de red con servicios cloud, constituye el punto final de la arquitectura SARC, en donde la telemetría procesada por el microordenador es almacenada de forma constante en el gestor de datos Supabase y posteriormente notificada en tiempo real por Twilio. Durante la etapa en la que se integró a la red, un incidente energético imprevisto que provocó la corrupción del sistema de los archivos de la memoria micro SD, dejando el software inoperativo y consecuentemente perdiendo datos que no fueron sincronizados. Para realizar esta mitigación de vulnerabilidad del sistema del microordenador por la energía, se implementó una estrategia en la que se basó la creación de imágenes espejos de todo el sistema completo, con el objetivo de asegurar todas las dependencias de los servidores alojados en el mircoordenador, como se visualiza en la figura 94.

Figura 94

Arquitectura de la estrategia de respaldo y mitigación.



Nota. Autoría propia

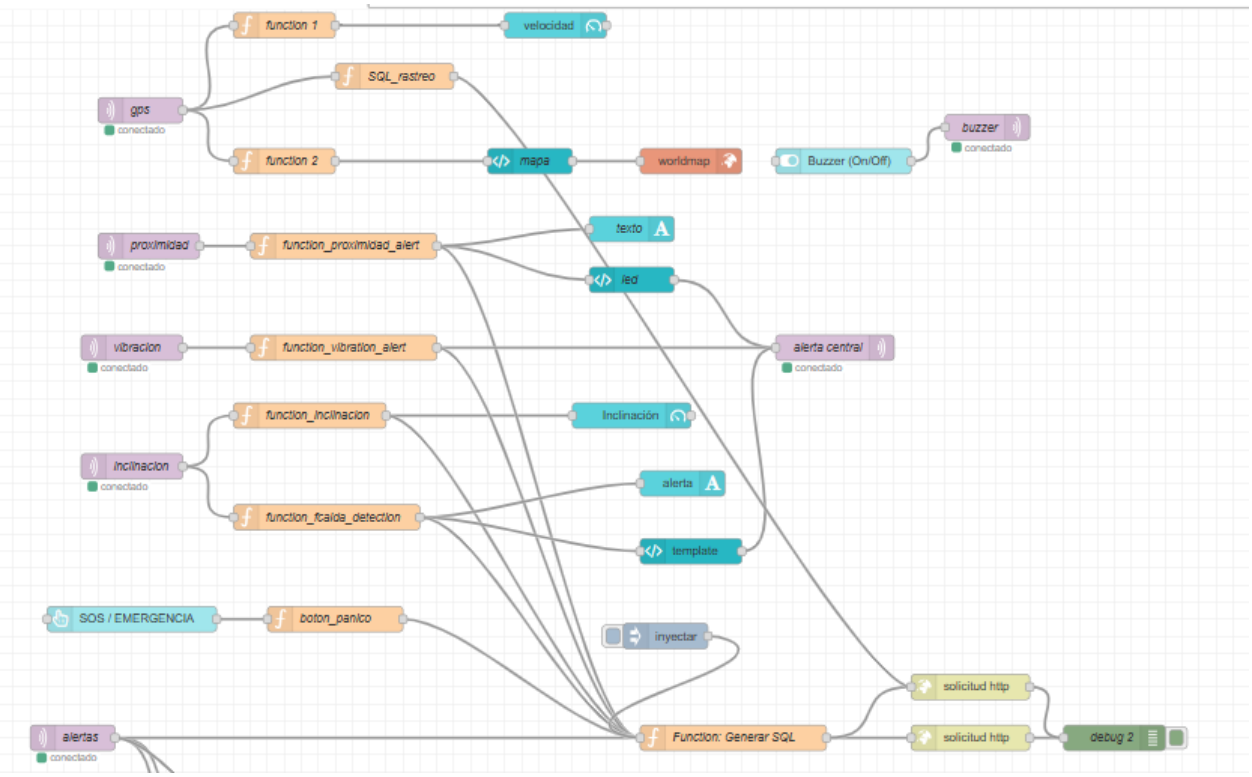
Esta política establecida ante incidentes energéticos garantiza la funcionalidad del sistema incluso si existen fallos de energía que colapsen todo el sistema permitiendo una restauración inmediate del software sin la necesidad de ejecutar una instalación desde cero.

4.4.1. *Envío y recepción de datos*

En cuanto al procesamiento de las tramas de datos, el microcontrolador centraliza toda la telemetría proveniente de todos los sensores mediante trama de paquetes en formato JSON, las cuales son transmitidas mediante el protocolo MQTT hacia el mircoordenador, una vez recibida la información, el motor orquestación Node-Red realiza un proceso lógico en el que divide la lógica útil de carga , en donde una copia es direccionada precisamente al almacenamiento de datos en Supabase y, ante las alertas mediante la detección de eventos críticos, se dispara una notificación redundante hacia la API Twilio, proceso detallado en la figura 95.

Figura 95

Implementación del flujo de orquestación de datos en Node-RED.

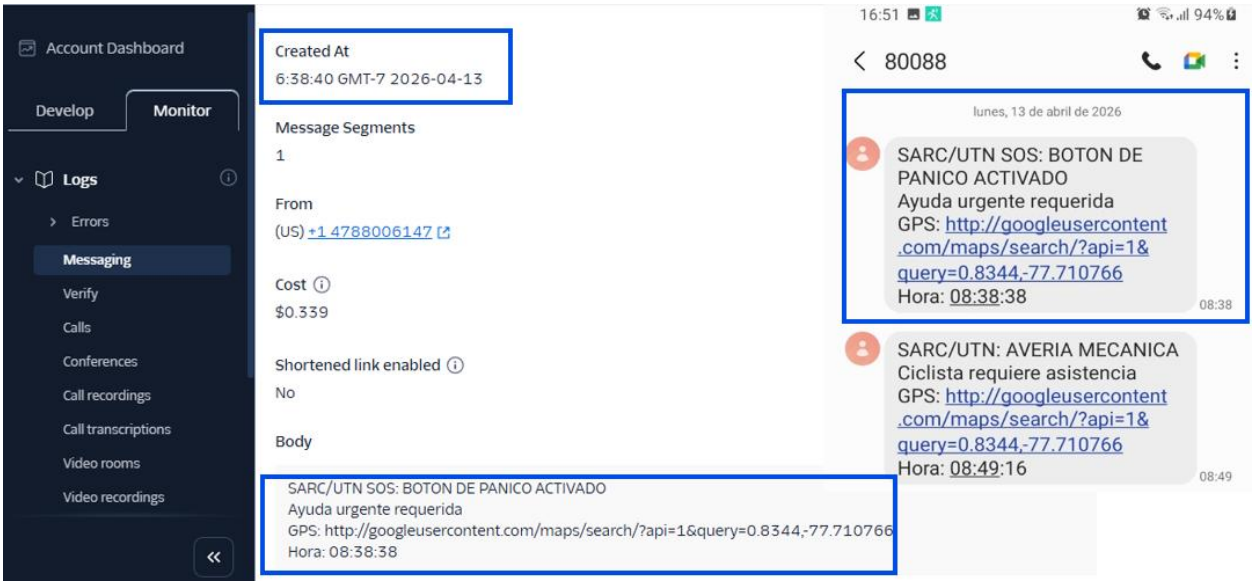


Nota. Autoría propia

Finalmente, la latencia de la red para actualizar el registro histórico de datos en Supabase fue capaz de mantenerse por debajo de 2 segundos desde la detección del evento en el nodo, además el tiempo de respuesta para recepción de mensajes SMS gestionado por el API Twilio se mantuvo entre un rango de 5 y 10 segundos, asegurando así un canal de comunicación efectivo para el rescate, como se observa en la figura 96.

Figura 96

Validación del tiempo de respuesta del sistema de alertas vía SMS.



Nota. Autoría propia

Como se observa en la figura 96 existe una discrepancia nominal de exactamente dos horas entre la creación del mensaje y la recepción del mismo, este desfase no representa el retardo de llegada del mensaje, sino que representa el desfase natural de las zonas horarias entre las plataformas. El microordenador opera en la zona horaria de Ecuador (UT-5), como se observa en el mensaje y la zona horario de Twilio registra los eventos bajo el uso horario del Pacifico (PDT, UTC-7) correspondiente a los servidores que alberga en California.

4.4.2. *Pruebas de APIs*

En lo que respecta a la respuesta de la API, se ejecutó mediante le protocolo de peticiones HTTP/REST direccionados hacia la base de datos Supabase, con esto se logra validar que Node-Red es capaz de procesar de forma asíncrona sin comprometer información delicada del flujo principal de datos, según se documenta en los registros visualizados en la figura 97.

Figura 97

Confirmación de peticiones HTTP/REST exitosas.



Nota. Autoría propia

En el servicio de alertas mediante el uso de Twilio se sometió a pruebas de latencia extremo a extremo, en donde se registró tiempos de respuesta constantes entre 5 a 10 segundos desde que ocurre el disparo de la petición en el mircoordenador. Con el fin de mitigar el uso incensario de créditos en el servidor de mensajería, se implementó a nivel de software en el backend de Python, utilizando un condicional para verificar que tipo de alerta está llegando por MQTT, además, el sistema separa las alertas preventivas de las críticas, así que las alertas de proximidad solo se ven en la app como se observa en la figura 98.

Figura 98

Lógica de discriminación de alertas para optimización de recursos

```
# === EMERGENCIA CRÍTICA: Disparar Twilio ===
# EXCEPCIÓN ESTRICTA: Las alertas de proximidad NUNCA disparan mensaje, aunque vengan como CRITICAL.
if tipo_alerta != "proximity" and (tipo_alerta in ["fall", "manual_sos"] or severidad == "CRITICAL"):
    print(f"\n[Sistema] ⚠ EMERGENCIA DETECTADA: tipo={tipo_alerta} | usuario_id={uid}")
    numero_emergencia = obtener_contacto_emergencia(id_usuario=uid)
```

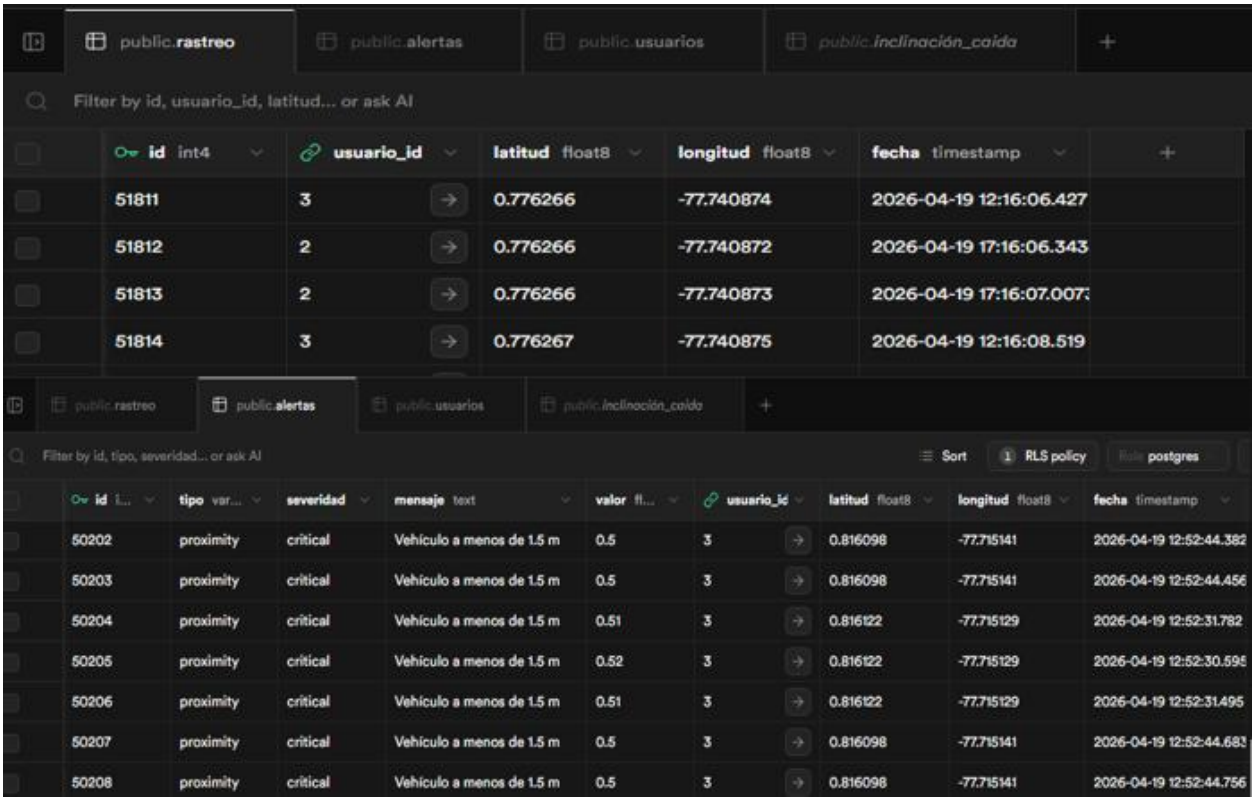
Nota. Autoría propia

Finalmente la integridad de los datos almacenados en Supabase, se realizó mediante auditorias directas en el panel de control tras cara recorrido de prueba, con el objetivo de verificar que no hubiera perdida de datos a pesar de que el internet celular 4G en ocasiones puede ser

inestable por el movimiento, analizando si las coordenadas mostradas en la base de datos se guardan correctamente, igualmente verificando que la marca de tiempo de los eventos coincida con la vida real, como se visualiza en la figura 99. De esta manera, esta auditoria confirma que Supabase es una herramienta fiable para el manejo de los datos, con el fin de que los registros almacenados puedan ser utilizados posteriormente.

Figura 99

Auditoría de persistencia de datos en el gestor de base de datos Supabase



Nota. Autoría propia

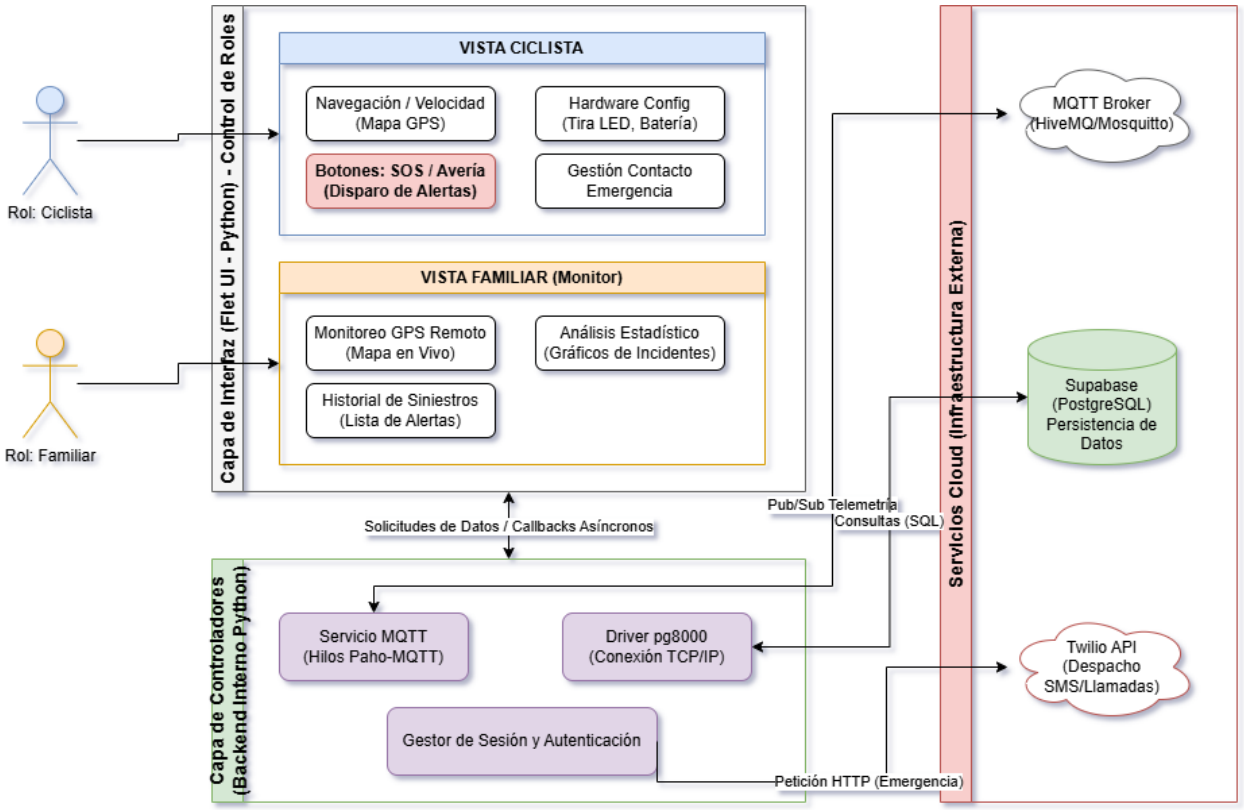
4.5 Resultados de la aplicación móvil

En lo que corresponde al desarrollo de la aplicación desarrollada en el software Python, se enfoca en la interfaz entre hombre-maquina (HMI) el cual es la encarga de cerrar el ciclo de

información del sistema SARC, cuya arquitectura lógica de los componentes esta detallada en la figura 100.

Figura 100

Arquitectura lógica y flujo de navegación de la aplicación móvil SARC.



Nota. Autoría propia

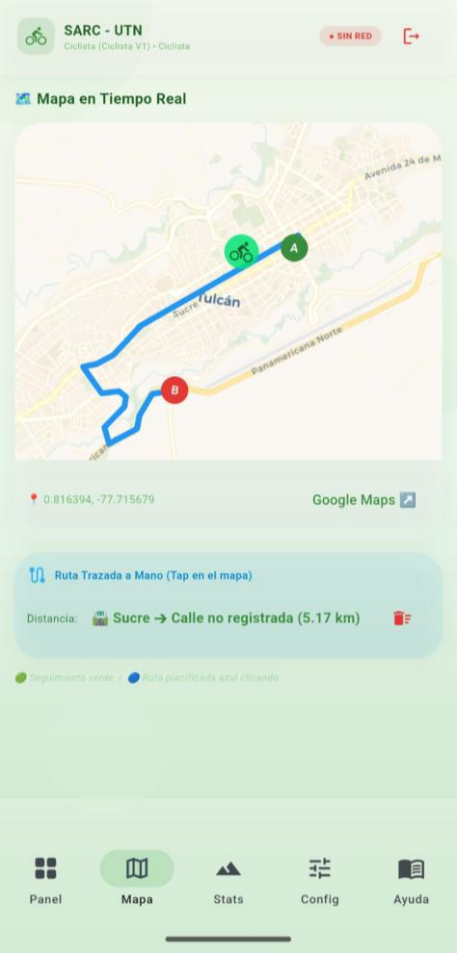
El objetivo principal de esta plataforma, es permitir la centralización de la visualización de la telemetría procesada y almacenada en la nube mediante Supabase, permitiendo así al usuario un flujo de navegación técnico que está diseñada especialmente para la gestión activa y la oportuna intervención ante alertas de emergencia. En este apartado se establece las bases para evaluar la usabilidad de la interfaz, el rendimiento del mapa de seguimiento y la reducción de la latencia de las respuestas inmediatas mediante la ejecución en segundo plano.

4.5.1. *Visualización de Telemetría y Geoposicionamiento en Tiempo Real*

La aplicación móvil está diseñada para implementar un esquema de refresco asíncrono orientado a los eventos, donde, esta implementación se ejecuta mediante un renderizado dinámico de la ubicación del ciclista esto gracias a las coordenadas procesadas y proporcionadas por el sensor NEO-6M mediante el microcontrolador en intervalos de tiempo de 2 a 5 segundos y las cuales están recuperadas desde la base de datos Supabase. Para la reconstrucción del recorrido histórico y la guía de navegación, se realizó mediante la integración de una capa de polilíneas mediante la API de OSRM, en la cual permite trazar de manera precisa el recorrido del ciclista sobre el mapa de la aplicación. cómo se observa en la figura 101.

Figura 101

Arquitectura lógica y flujo de navegación de la aplicación móvil SARC.

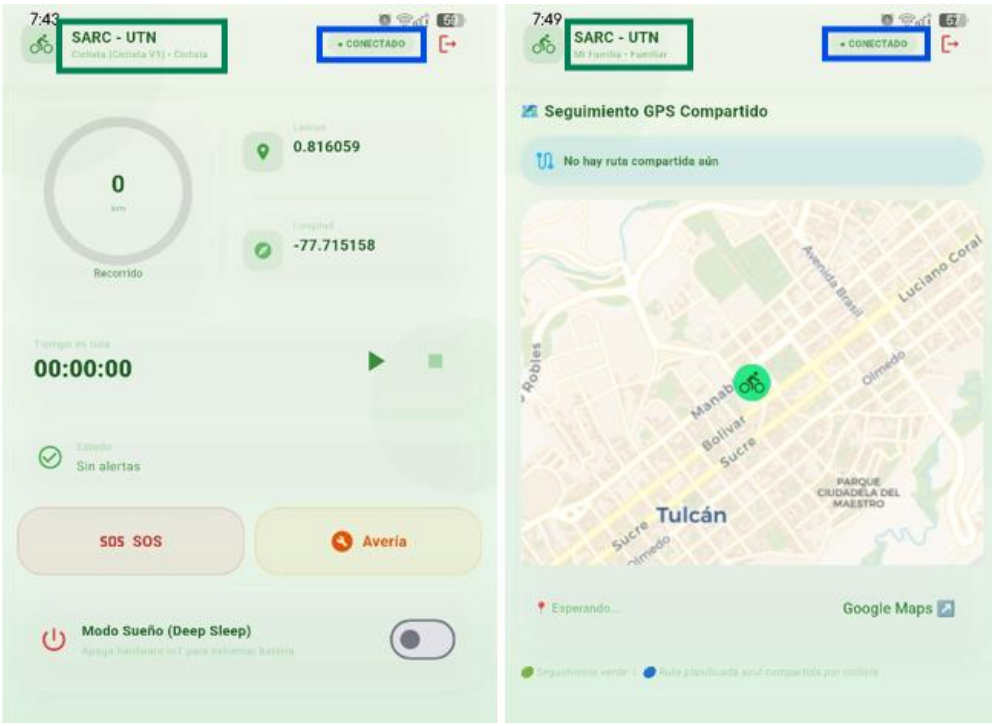


Nota. Autoría propia

Mediante la infraestructura de la base de datos, el sistema permite aplicar políticas de RBAC con el fin de diferenciar el rol de cada usuario, mientras que el familiar acede a una vista de monitoreo en donde se destacan las alertas de siniestros y el recorrido, el acceso a los actuadores del hardware y los indicadores visuales permanece restringido para el rol del usuario ciclista. Así mismo, la persistencia del establecimiento del enlace y conexión se gestiona a través de un monitor de estado en la cabecera de la interfaz, permitiendo al usuario identificar si la negociación del protocolo TCP/IP fue exitosa mediante el cliente Paho-MQTT y el estado de conexión con el bróker como se observa en la figura 102.

Figura 102

Monitor de estado de conexión del cliente Paho-MQTT.



Nota. Autoría propia

Finalmente, debido a la baja latencia que proporciona el enlace LTE, el flujo de telemetría permite que el marcador de posición en el mapa se desplace sin ningún tipo de inconvenientes y además sin presentar saltos bruscos, logrando así una visual coherente con el movimiento real de la bicicleta.

4.5.2. *Funcionamiento del sistema de alertas*

El sistema SARC implementa una jerarquía visual de las alertas en su panel en el cual diferencia niveles de riesgo mediante códigos de color, amarillo/naranja fue utilizado para comunicar alertas de proximidad y para las alertas criticas fue utilizado un fondo rojo de alta visibilidad como se ilustra en la figura 103.

Figura 103

Reactividad visual de la interfaz ante cambios de estado de emergencia

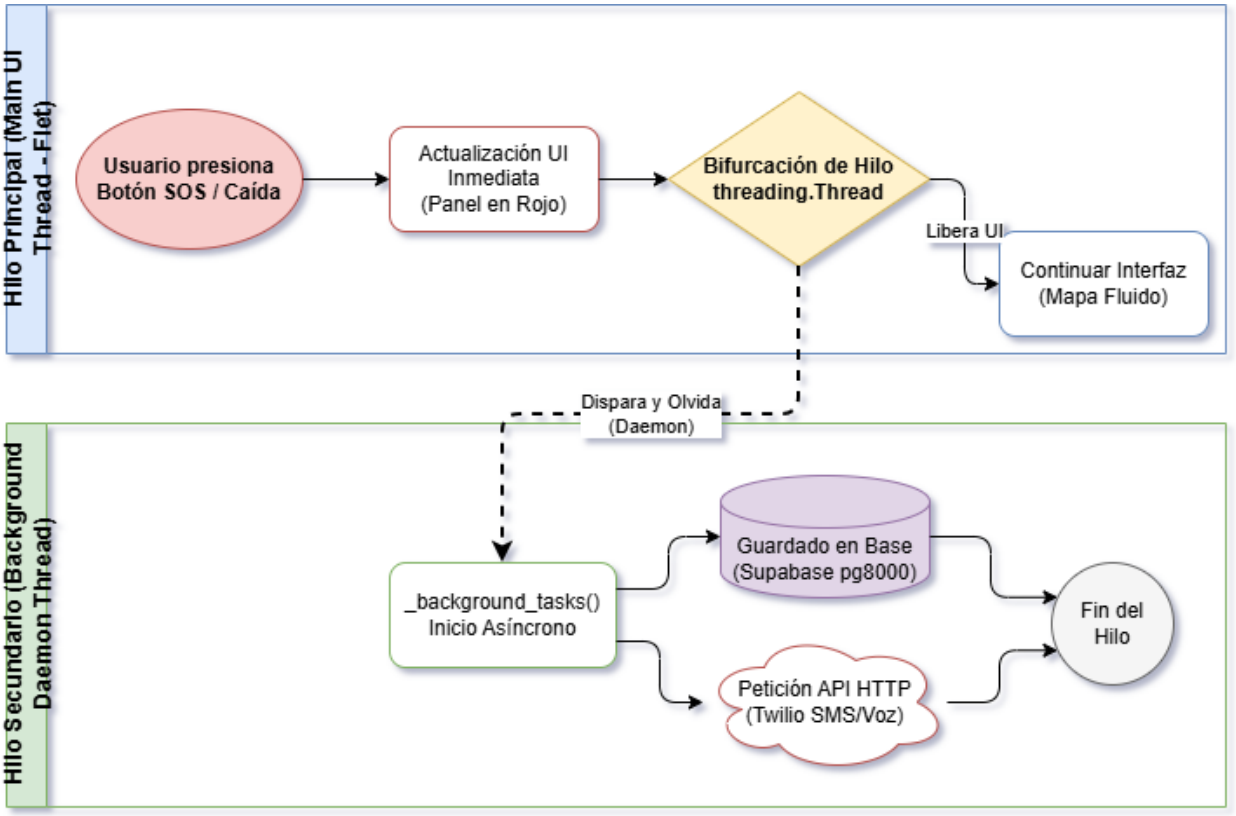


Nota. Autoría propia

Mediante el uso de Threading en Python, el manejo del procesamiento del botón SOS se maneja con la ejecución de forma asíncrona, permitiendo así que el mapa y la telemetría siga operando sin interrupciones mientras se gestiona la alerta. Como medida de seguridad estricta y orientada a salvaguardar la integridad del usuario, prioriza el despacho inmediato hacia la base de datos y a la API Twilio, asegurando que la ayuda médica o policial no tenga retrasos ante incidentes críticos. Para un mejor entendimiento de esta arquitectura, el flujo lógico se detalla en la figura 104.

Figura 104

Arquitectura de procesamiento asíncrono para despachar de alertas críticas



Nota. Autoría propia

4.6 Validación del sistema

Una vez concluida la verificación de los módulos individuales, resulta determinante ejecutar una validación integral que ayude a certificar el comportamiento del sistema SARC, este proceso de evaluación se centra en garantizar la funcionalidad entre el nodo de percepción y la comunicación con los servicios cloud, verificando si cumplen con rigurosidad los estándares de seguridad vial y de telecomunicaciones establecidos en la fase de diseño.

La estrategia de validación se basa en pruebas de funcionalidad lógica, estabilidad en prolongados usos y el cumplimiento estricto de los requerimientos iniciales del sistema. Bajo este enfoque ordenado, lo que se busca es asegurar la fiabilidad de la arquitectura implementada en el

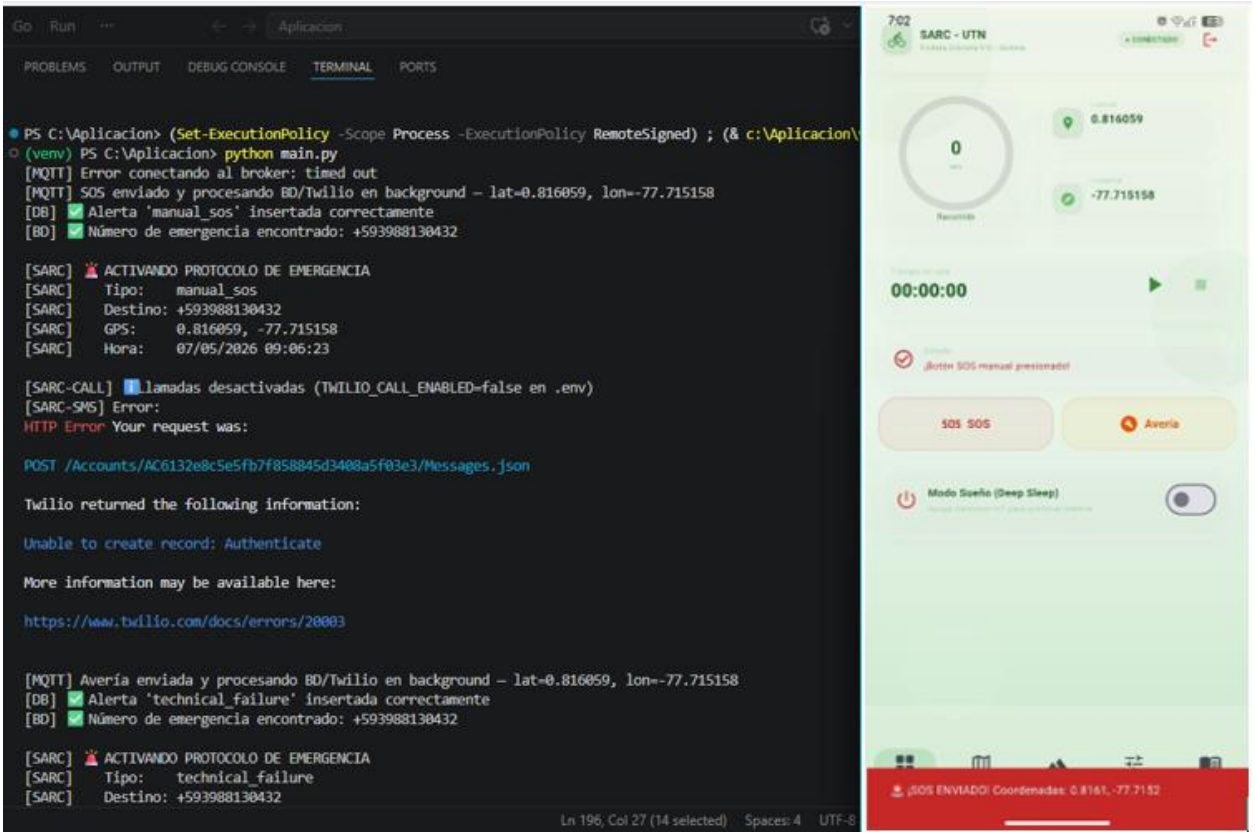
sistema, comprobando así que las respuestas del sistema SARC son consistentes y además capaces de sostener las demandas de protección propuestas.

4.6.1. *Pruebas funcionales*

Para esta etapa de pruebas, se evaluó la fluidez y la respuesta instantánea de los actuadores digitales para SOS y Avería Mecánica, los cuales están integrados en la aplicación móvil mediante el software Python y visibles para la interfaz del ciclista. Confirmando que la activación de estos botones realiza un disparo lógico de emergencia hacia el backend sin contraer retardos que sean perceptibles, como se observa en la figura 105.

Figura 105

Activación de la alerta y registro de ejecución asíncrona



Nota. Autoría propia

Para el proceso de descarte del ruido visual fue comprobado exitosamente, en la lógica el código se implementó para ignorar automáticamente cualquier presencia que sea detectada a una distancia ≥ 2m, esto se llevó a cabo con el objetivo de que el ciclista no pierda la concentración durante el recorrido cuando la distancia de proximidad no sobrepase esa distancia y logrando así que la alerta visual accione en el momento en que el sensor detecte una distancia menor a esa, como se detalla en la figura 106.

Figura 106

Lógica de filtrado de proximidad y control de umbrales de seguridad

```
// Evalucación de la zona de peligro (Rango: 0 a 2.0 metros )
bool objetoCerca = (dist > 0 && dist <= 2.0);
String proxData = "{ \"data\": { \"distance_m\": " + String(dist) + " } }";
client.publish("bici-iot/rear/telemetry/proximity", proxData.c_str());
// MANEJO DE LA TIRA LED NEOPIXEL
if (objetoCerca && alertasVisualesActivas) {
  // Pinta la tira de color ROJO intenso
  for(int i=0; i<NUMPIXELS; i++) {
    tiraLed.setPixelColor(i, tiraLed.Color(255, 0, 0));
  }
  tiraLed.show();
} else {
  // Apaga la tira si no hay peligro
  tiraLed.clear();
  tiraLed.show();
}
```

Nota. Autoría propia

Con el fin de garantizar la operabilidad del sistema durante todo el recorrido sin interrupciones, se realizó el testeo de la arquitectura mencionado en el apartado anterior para la recuperación de la red, en el cual se hacía énfasis al uso de un script de monitoreo el cual ayuda reiniciar de forma automática el servicio del módulo LTE, esto ante fallos hacia servidores externos, esto esta detallado en la figura 94.

Finalmente, para validar la funcionalidad de la alerta visual, se ejecutaron protocolos de prueba de integración de software. En esta fase se evaluó el tiempo de respuesta del sistema al enviar los comandos necesarios para activar o desactivar la tira de Led NeoPixel directamente de desde la interfaz del usuario ciclista en la aplicación móvil, demostrando así un correcto funcionamiento mediante la comunicación bidireccional hacia MQTT mediante los tópicos permitiendo al usuario manejar los actuadores lumínicos a su voluntad y respondiendo instantáneamente los requerimientos a la interfaz, como se observa en la figura 101.

Figura 107

Panel de configuración y gestión remota de actuadores



Nota. Autoría propia

4.6.2. *Desvanecimiento por Multitrayecto*

En el entorno urbano, la propagación de la señal de radiofrecuencia desde la radio base (eNodeB) que tiende a viajar en línea recta únicamente (LOS) hacia la antena del sistema del módulo LTE, esta señal tiende a experimentar varios fenómenos conocidos como reflexión al

chocar con edificios o montañas debilitando la potencia de la señal, difracción que ocurre cuando tiende la señal a rodear bordes esquineros de obstáculos y dispersión que esta presenta cuando la señal choca con superficies rugosas o el follaje del entorno como se observa en la figura 108. Como resultado múltiples reflejos de ondas llegan a la antena receptora provocando fenómenos de interferencia, matemáticamente este desfase se expresa mediante la ecuación:

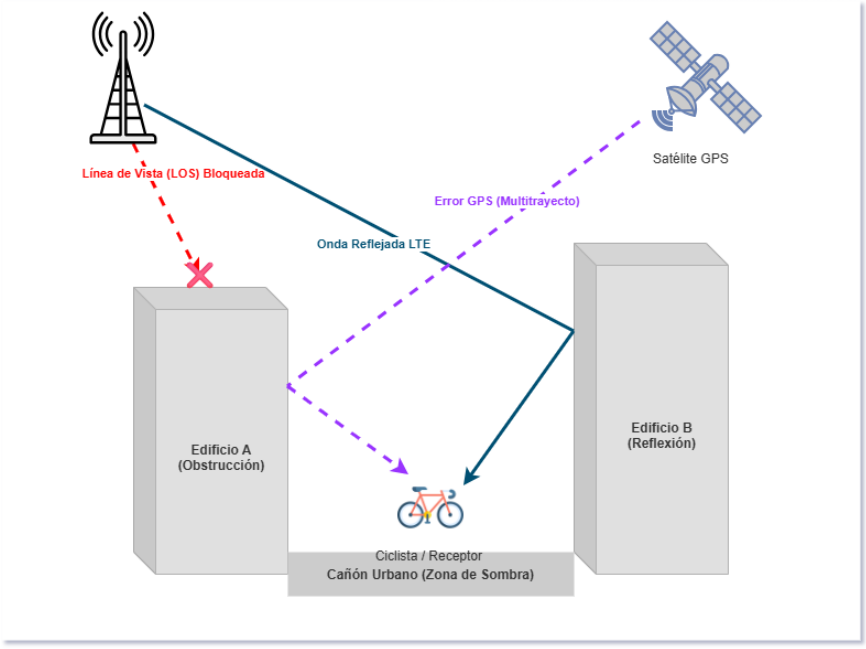
$$\Delta\varphi = \frac{2\pi\Delta d}{\lambda} \qquad (12)$$

En donde una diferencia de fase de 180° induce a un desvanecimiento profundo (Deep Fade) que es capaz de corromper las tramas TCP/IP que están en tránsito.

Con el fin de conservar un enlace compacto, se optó la decisión de operar el modem Quectel EG25-G en una topología SISO (*Single-Input Single-Output*) con una única antena externa que sale sobre la carcasa que protege a todo el sistema sin tener atenuación de RF.

Figura 108

Esquema de interferencia por multitrayecto en señales LTE y GPS.



Nota. Autoría propia

4.6.3. *Latencia por Traspaso de Celdas (Handover) en Ruta*

Para escenarios de movilidad extendida, la cobertura continua del sistema no se puede garantizar con en el enlace de un único eNodeB. Con el fin de mantener el enlace, el sistema tiene la capacidad de evaluar constantemente la Potencia de la Señal de Referencia Recibida (RSPR), en el instante que el sistema detecte que una celda vecina ofrece una mayor potencia, la red celular permite orquestar automáticamente un protocolo de Handover (traspaso de celda).

Durante esta sincronización física con la nueva antena y con la reasignación de nueva IP, el flujo de datos experimenta una pausa temporal estimada entre 100ms y 2s. Esta desconexión temporal no presenta un fallo estructural, pero se manifiesta en el Dashboard como pequeños retrasos en la actualización de la telemetría GPS, este comportamiento es esperado al operar bajo redes celulares públicas.

Finalmente, para salvaguardar los datos críticos ante estas desconexiones temporales, la arquitectura de la aplicación móvil permite aislar el manejo de fallos, cuando ocurre una alerta mediante el proceso del Handover, el sistema omite él envió de los datos hacia el buffer físico del hardware y lo que realiza es un encolamiento en RAM directamente en el smartphone del usuario, con esto se asegura la retención temporal de la petición. De acuerdo a esto, si el usuario presiona el botón de emergencia virtual durante este evento, la librería Paho-MQTT tiene la tarea de retener el paquete de auxilio como se observa en la figura 109, el cual está apoyado en una arquitectura de Threading.

Figura 109

Implementación de hilo asíncrono en Python

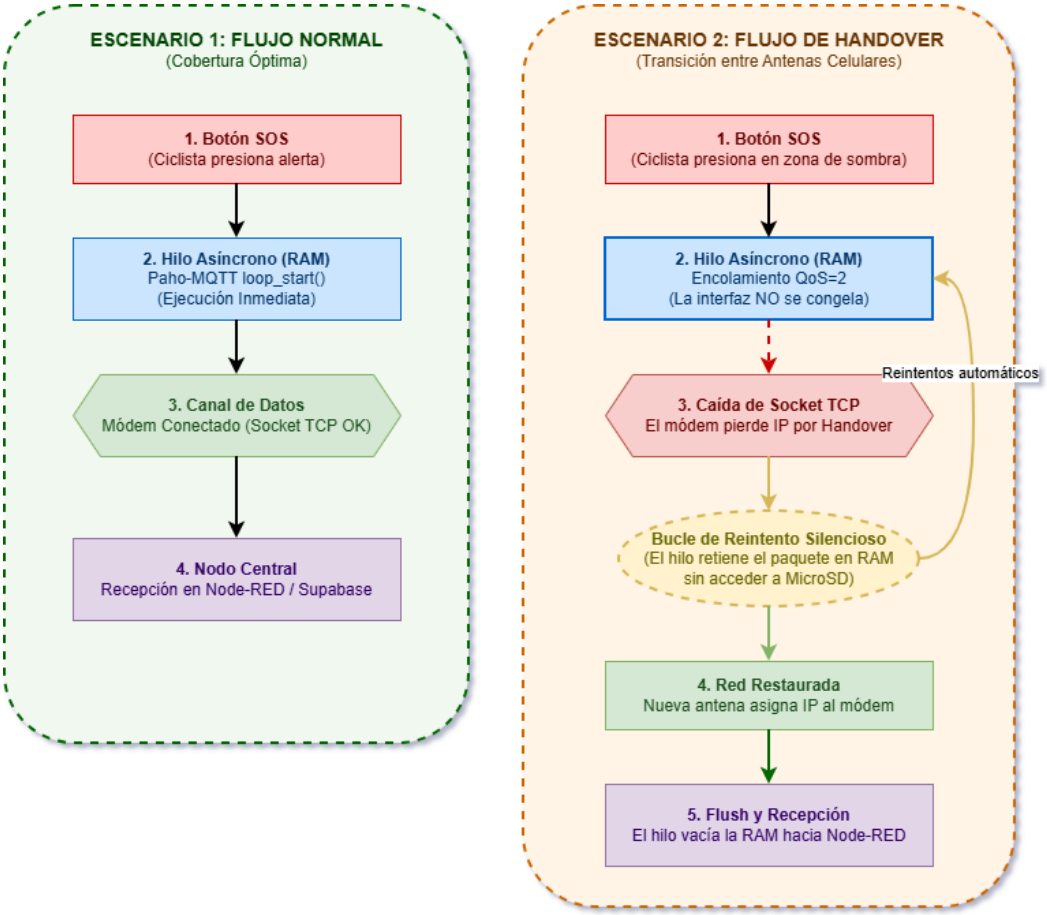
```
75      def start(self):
76          """Conecta al broker MQTT en un hilo separado."""
77          if self.connected:
78              return
79          try:
80              self.client.connect(BROKER, PORT, keepalive=60)
81              self.client.loop_start()
82          except Exception as e:
83              print(f"[MQTT] Error conectando al broker: {e}")
```

Nota. Autoría propia

En donde el sistema activa un bucle de reintento silencioso en segundo plano, cuando el socket se restablece luego del handover, el mensaje que fue retenido se dispara automáticamente (ver figura 110).

Figura 110

Diagrama de secuencia de mitigación SOS durante el Handover celular.



Nota. Autoría propia

4.6.4. *Degradación de la Relación Señal a Interferencia más Ruido (SINR)*

Al realizar una evaluación de manera aislada la potencia de la señal RSPR resulta engañosa en entornos saturados electromagnéticamente, esto se debe a que el sistema puede experimentar un estado falso de conexión a internet provocado por el ruido térmico y la interferencia cocanal de las caldas vecinas, este nivel de ruido se determina mediante la Relación Señal a Interferencia más ruido:

$$SINR = \frac{S}{I + N} \qquad (13)$$

Este impacto de degradación física está regido bajo el teorema de Shannon-Hartley, el cual permite definir el límite superior absoluto de la tasa de trasferencia libre de errores.

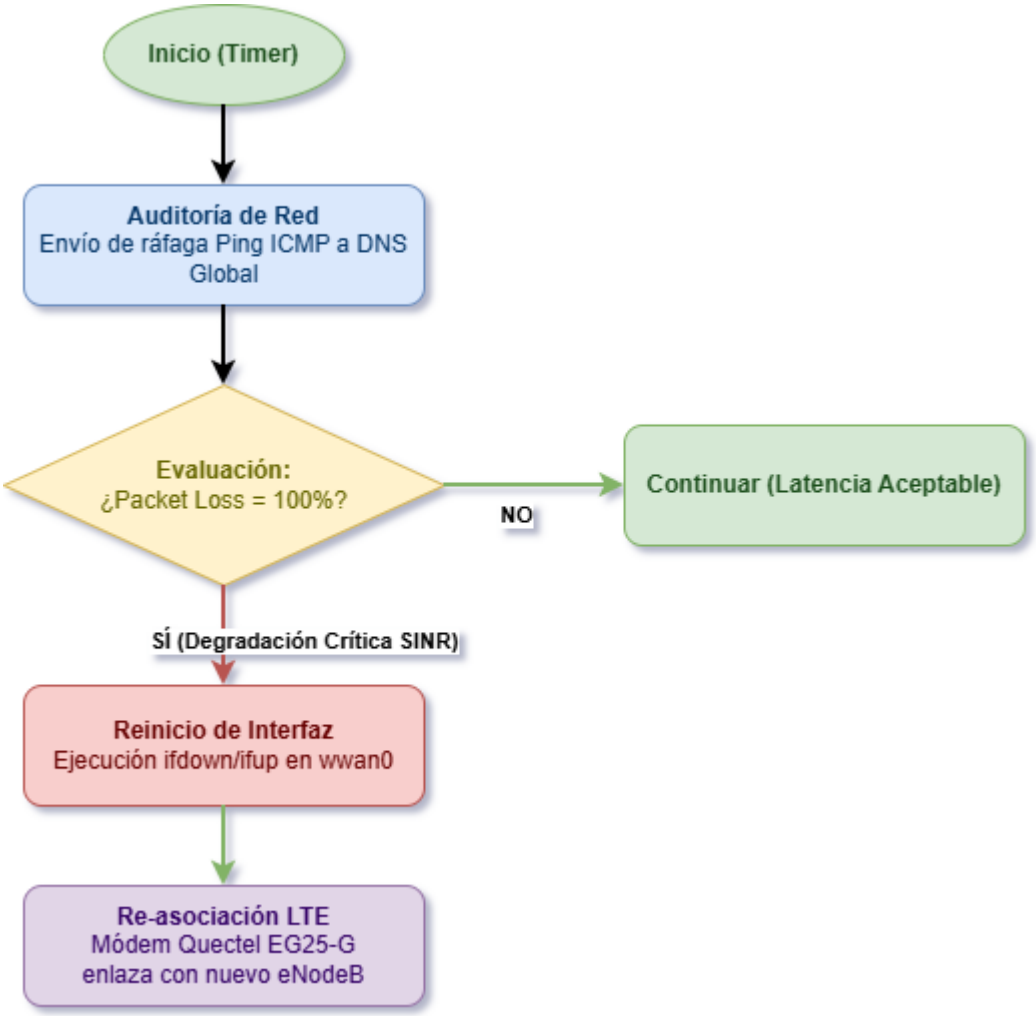
$$C = Blog_2(1 + SINR) \qquad (14)$$

Bajo este teorema, una caída drástica del SINR en el enlace LTE fuerza a la red a degradar el esquema de modulación de 64-QAM a QSPK con el objetivo de salvaguardar la carga útil.

Finalmente, para invalidar esta latencia, se implementó un proceso antes mencionado Watchdog, el cual rechaza el estado de conexión reportado por el modem, en su lugar audita las salidas a internet mediante una ráfaga de ICMP periódicas a servidores DNS globales, de modo que, al registrar fallos, obliga al microordenador a reiniciar la interfaz wann0, con lo que obliga al módulo LTE a soltar la celda y realizar un enlace nuevo con el eNodeB para renegociar el enlace (ver Figura 111).

Figura 111

Diagrama de flujo lógico Watchdog para la nueva asociación celular.



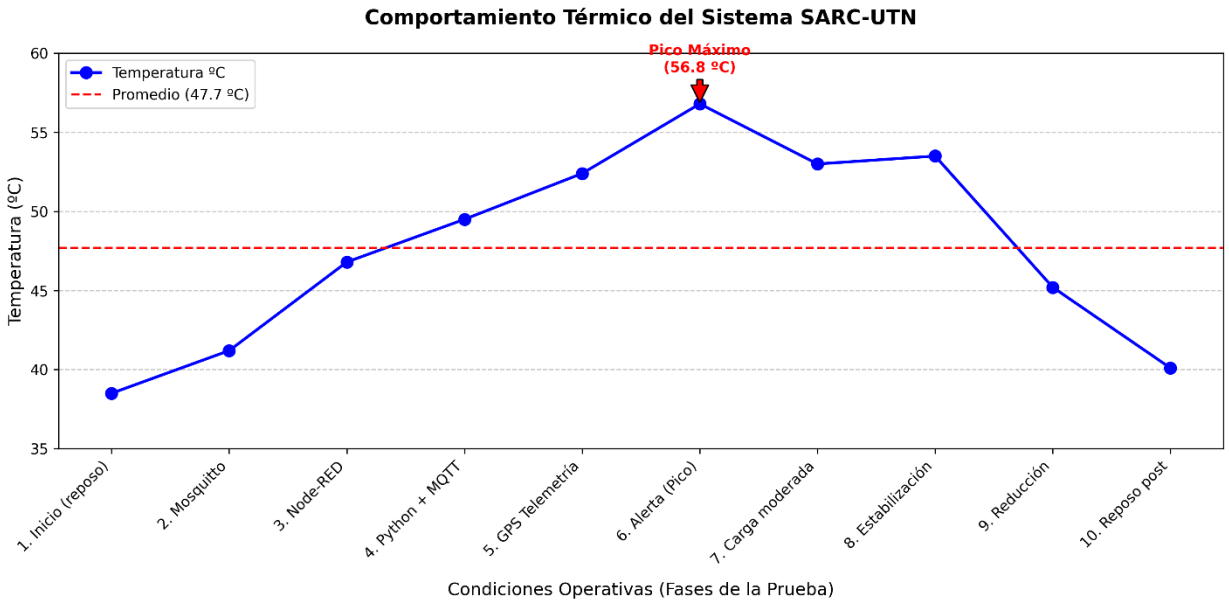
Nota. Autoría propia

4.6.5. *Pruebas de estabilidad*

Al someter la arquitectura a pruebas de carga de energía continua, el microordenador demostró la capacidad térmica plenamente consistente y de acuerdo a las ecuaciones de estrés máximo (5), el estrés mínimo (6) y de la temperatura promedio que fueron analizadas en el apartado [4.2.2], se verifica que el comportamiento térmico del microordenador es factible para este sistema como se observa en la figura 112.

Figura 112

Curva de temperatura del microprocesador durante operación continua



Nota. Autoría propia

La estabilidad de la memoria RAM y la optimización de la mejora de intercambio de espacio (SWAP) permitieron que los servicios alojados en el microordenador y scripts de automatización se ejecutaran sin tener problemas durante la jordana, validando así los resultados obtenidos en la tabla 19 y la figura 84. para evaluar la concurrencia del sistema de las alertas, se realizó una prueba de estrés mediante el uso del botón SOS de forma consecutiva en cinco ocasiones, gracias a la arquitectura de Python Threading (véase figura 104), la interfaz gráfica de la aplicación móvil no presento bloqueos, esta prueba de accionamiento del botón SOS se ve detallada en la figura 113.

Figura 113

Registro de la consola durante una prueba de estrés de concurrencia

```
(venv) PS C:\Aplicacion> python main.py
[MQTT] Error conectando al broker: timed out
[MQTT] SOS enviado y procesando BD/Twilio en background — lat=0.816059, lon=-77.715158
[MQTT] SOS enviado y procesando BD/Twilio en background — lat=0.816059, lon=-77.715158
[MQTT] SOS enviado y procesando BD/Twilio en background — lat=0.816059, lon=-77.715158
[MQTT] SOS enviado y procesando BD/Twilio en background — lat=0.816059, lon=-77.715158
[DB] ✅ Alerta 'manual_sos' insertada correctamente
[MQTT] SOS enviado y procesando BD/Twilio en background — lat=0.816059, lon=-77.715158
```

Nota. Autoría propia

Finalmente, con el objetivo de disminuir el consumo de créditos de la API Twilio, el servicio de llamada de voz fue deshabilitado gracias a la declaración en una variable del entorno en estado falso que se encuentra en el archivo físico (.env), asegundado así que el sistema realice el procesamiento solo del mensaje SMS y respectivamente el almacenamiento en la base de datos, como se observa en la figura 114.

Figura 114

Restricción temporal de llamadas de voz

```
TWILIO_PHONE_NUMBER=+14788006147

# Número destino de emergencia (celular del familiar con código de país)
# Ecuador con número de ejemplo: +593XXXXXXXXX
# Número de respaldo
TWILIO_EMERGENCY_NUMBER=+593990[redacted]
TWILIO_EMERGENCY_NUMBER=+593981[redacted]

# Activar llamada de voz además del SMS (true/false)
TWILIO_CALL_ENABLED=false

def send_emergency_call(to_number, alert_type="fall"):
    """Realiza una llamada de voz pregrabada usando TwiML en línea."""
    if alert_type == "proximity":
        return False

    if not CALL_ENABLED:
        print("[SARC-CALL] ℹ️ Llamadas desactivadas (TWILIO_CALL_ENABLED=false en .env)")
        return False

    destino = to_number or EMERGENCY_NUM
    if not destino:
        print("[SARC-CALL] ⚠ No hay número de destino para la llamada.")
        return False
```

Nota. Autoría propia

4.6.6. *Análisis comparativo con los requerimientos*

Tras el análisis con los requerimientos planteados, el sistema satisface el cumplimiento de los requerimientos EA-04, EA-06 y EA-08 mediante la integración del microordenador con el módulo LTE, las pruebas de estabilidad de procesamiento y comportamiento térmico descritas en la sección [4.2.2], ayudaron a confirmar la capacidad multitarea del sistema sin interrupciones, mientras que el análisis de autonomía energética en el apartado [4.3.2] permitió ratificar que la batería de 10000 mAh conjuntamente con el módulo IP5328P garantizan la operabilidad necesaria para garantizar al funcionamiento estable y la independencia energética necesaria para el entorno móvil. Así mismo, los requerimientos de lata prioridad EF-01, EF-02, EF-03 y EF-11, el sistema SARC demostró tener una precisión en el rastreo continuo en tiempo real y por ende una transmisión de las alertas efectiva tras detectar algún evento crítico. Los resultados documentados en las secciones [4.1] y [4.4] ayudan a corroborar la operabilidad continua superior a las 4 horas de recorrido, llegando así a cumplir con el objetivo de la protección activa del ciclista.

Sobre el análisis de la capa de presentación, los requerimientos EF-05 y EF-08 se cumplen al realizar una integración jerárquica visual de las alertas dinámicas que permite un control de acceso basado en roles que está gestionado desde la base de datos. Como se detalla en la sección [4.5.1], esto ayuda a diferenciar las diferentes funciones de monitoreo tanto para el uso Familiar frente al uso y administración del Ciclista, esto se realizó con el fin de asegurar la integridad del ciclista.

Finalmente, los requerimientos de arquitectura EA-1, EA9 y EA-11 ayudan a fundamentar el uso del software libre (Python, Node-Red, MQTT), lo que ayudo a otorgar la flexibilidad para la autorrecuperación de la red, además el diseño de la carcasa en 3D ligero permite asegurar la

estructura compacta que no compromete la aerodinámica ni el confort del usuario, asegurando así que el producto final cumpla con los estándares definidos. Las tablas 25 y 26 exponen la matriz de cumplimiento comprobando cada requerimiento inicial con su respectiva validación dentro del sistema.

Tabla 25

Matriz general de cumplimiento de requerimientos funcionales (EF)

Requerimientos							
No.	Requerimientos	Prioridad			Estado		Validación
		Alta	Media	Baja	C	NC	
Requerimientos Funcionales (EF)							
EF-01	El sistema debe identificar eventos de impacto o caídas.	x			✓		Algoritmo MPU6050
EF-02	El sistema tiene que ser capaz de registrar la localización en tiempo real.	x			✓		Telemetría GPS en UI
EF-03	El sistema transmitirá una alerta después de cada evento o accidente.	x			✓		API Twilio (MQTT)
EF-04	El sistema almacenará los datos registrados en la memoria de la Raspberry PI y luego enviado a la nube.		x		✓		Base de datos Supabase

EF-05	La interfaz incluye visualmente las alertas en tiempo real.		x		✓	UI Asíncrona (Flet)
EF-06	La aplicación permite acceder a los eventos registrados.		x		✓	Analítica gráfica móvil
EF-07	Aplicación móvil descargable.	x			✓	Aplicación APK compilada
EF-08	El sistema contara con un rol especifico de usuario, para funciones básicas, para administrar y para configuraciones.		x		✓	Control de Roles (RBAC)
EF-09	El dispositivo prioriza el uso del usuario final.	x			✓	Automatización Cero-Dudas
EF-10	Ofrecer manual del usuario.			x	✓	Documentación de perfiles
EF-11	El sistema garantiza operabilidad continua durante el recorrido.	x			✓	Multihilo (Threading)

Tabla 26

Matriz general de cumplimiento de requerimientos de arquitectura (EA)

Requerimientos						
No.	Requerimientos	Prioridad			Estado		Validación
		Alta	Media	Baja	C	NC	

Requerimientos de arquitectura (EA)				
EA1	El sistema debe contar con una estructura ligera y compacta.	x	✓	Encapsulamiento 3D
EA2	El sistema requiere que la arquitectura permita distribuir los elementos de manera eficiente, que garanticen comodidad.	x	✓	Diseño modular ergonómico
EA3	Se espera que el dispositivo sea capaz de soportar condiciones exteriores.	x	✓	Carcasa protectora
EA4	Es importante que el procesador permita ejecutar múltiples tareas simultáneamente.	x	✓	Raspberry Pi multinúcleo
EA5	El microcontrolador necesita admitir varias interfaces de conexión, analógicas y digitales.	x	✓	Puertos ESP32 (I2C/GPIO)
EA6	El dispositivo debe integrar comunicación LTE/4G para transmisiones continuas.	x	✓	Módem USB LTE

EA7	Es importante verificar que los sensores trabajen con un voltaje compatible.	x	✓	Conversión lógica 3.3V/5V
EA8	El dispositivo deberá operar con una fuente de energía autónoma.	x	✓	Batería y módulo IP5328P
EA9	Se plantea utilizar software de desarrollo libres, para mayor flexibilidad.	x	✓	Python, Mosquitto, Node-RED
EA10	El software debe comprender la compatibilidad de librerías.	x	✓	Entornos virtuales validados
EA11	El sistema debe permitir la comunicación y el envío de notificación en tiempo real cuando detecte un evento.	x	✓	Latencia MQTT mínima

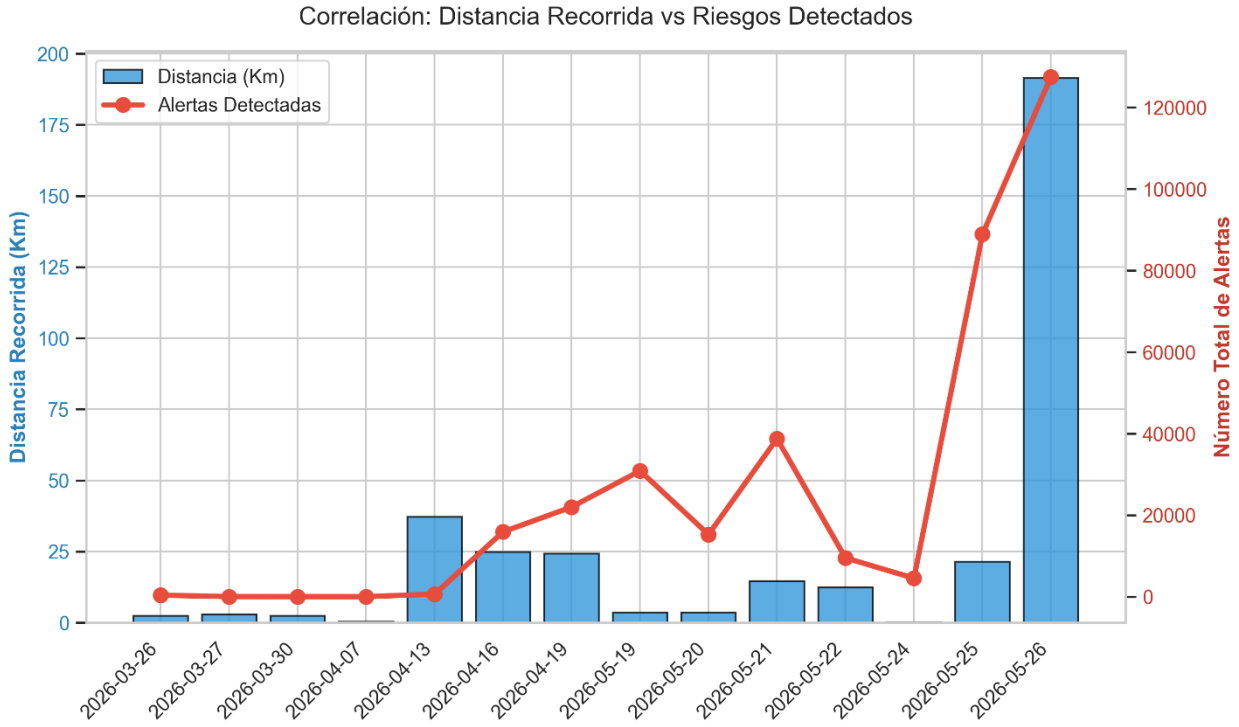
4.7 Análisis de resultados en escenarios reales

Durante la fase de validación de pruebas en campo, que fue desplegada a lo largo de tres semanas en la ciudad de Tulcán provincia del Carchi, se realizó el monitoreo de todo el sistema SARC-UTN en escenarios reales, al realizar una evaluación de las coordenadas geográficas obtenidas durante este tiempo de pruebas, permitió obtener una correlación directa entre los kilómetros recorridos por el ciclistas y la exposición a incidentes de riesgo vehicular, con esto se resaltó que el módulo de procesamiento se mantuvo estable durante trayectos prolongados sin tener

interrupciones en las comunicaciones sean locales o globales, con el fin de verificar estos resultados la figura 115 ayuda a comprender esta correlación que a mayor uso y distancia recorrida, el sistema es capaz de seguir capturando los eventos de riesgo.

Figura 115

Análisis comparativo de alertas emitidas en función de los kilómetros recorridos.



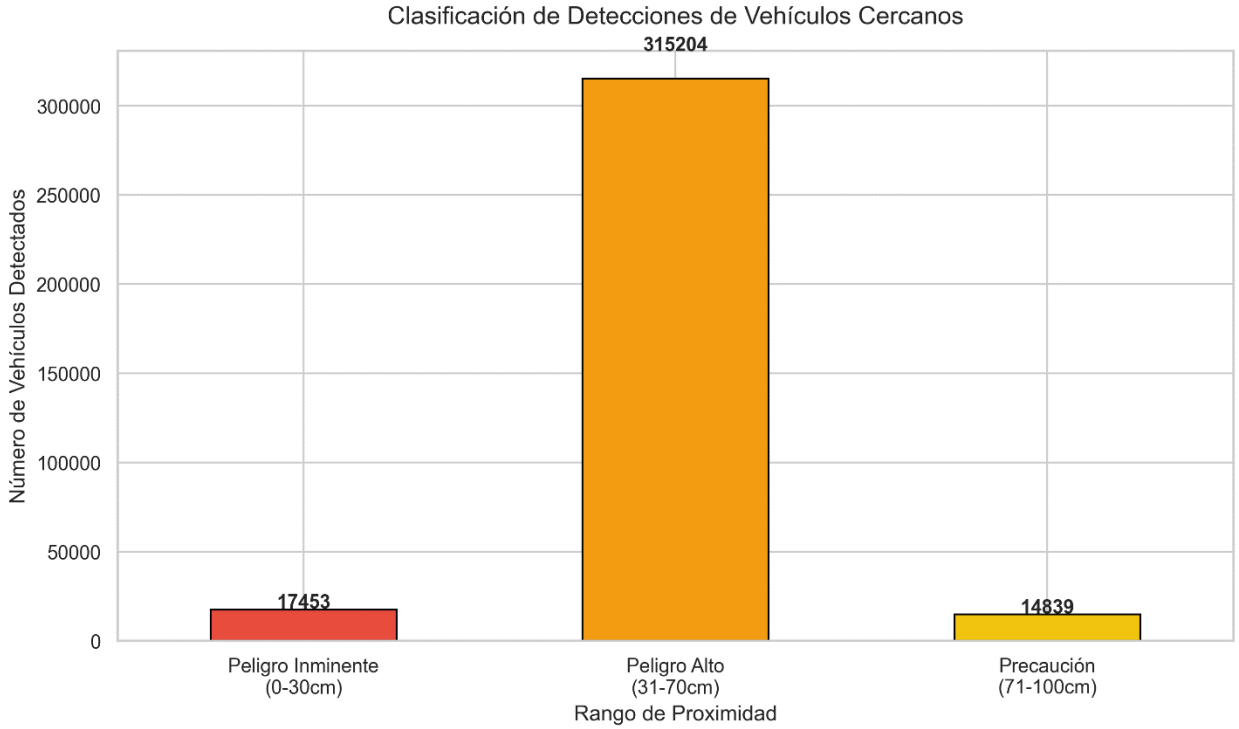
Nota. Autoría propia

Al someter al nodo a pruebas de recorridos tanto en rutas urbanas céntricas utilizando los carriles de ciclovía como en vías de alta velocidad en Carchi, el sensor de proximidad mediante el uso de su radar ultrasónico permitió evidenciar una tasa alta de muestreo efectivo. El sistema fue capaz de filtrar con éxito el ruido ambiental que está presente ayudando a clasificar más de 38,000 eventos de aproximación, en donde según los datos registrados el mayor umbral de este muestro recae en los umbrales que fueron definidos como Peligro Inminente (0 a 30 cm), estos fenómenos fueron causados especialmente en el tráfico vehicular presente en el centro de la ciudad, con el fin

de analizar los peligros inminentes en una ruta definida con tráfico vehicular presente se realizó un análisis estadístico detallado en la figura 116, permitiendo obtener así los datos crudos en centímetros.

Figura 116

Clasificación de alertas de proximidad vehicular según su nivel de severidad y peligro.

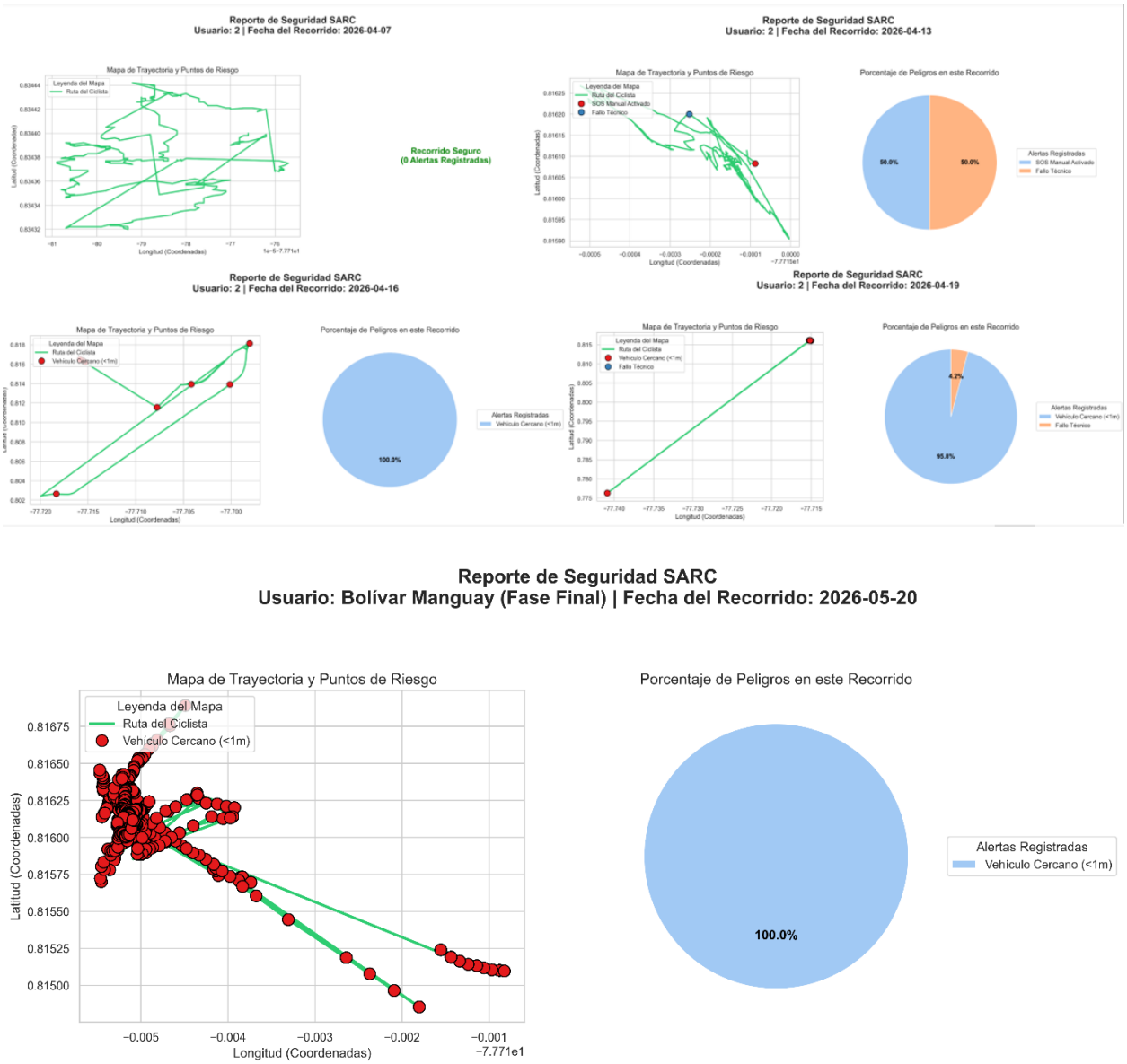


Nota. Autoría propia

Pese a las zonas de sombra estructural de la ciudad, el sensor de rastreo GPS NEO-6M evidenció que las pérdidas de las tramas de datos fueron de tipo despreciables en la sincronización hacia la base de datos aunque a pesar de tener estas pérdidas, las diferentes trayectorias diarias, comparadas en la plataforma, permitió comprobar que el sistema mantiene la fiabilidad de la ruta original sin presentar desviaciones brutales de la señal, al cruzar por zonas con edificios o con interferencias por las altas infraestructuras propias en zonas urbanas, estos análisis se visualizan en la figura 117.

Figura 117

Trazabilidad histórica del recorrido generada por el módulo GPS durante la fase de campo.



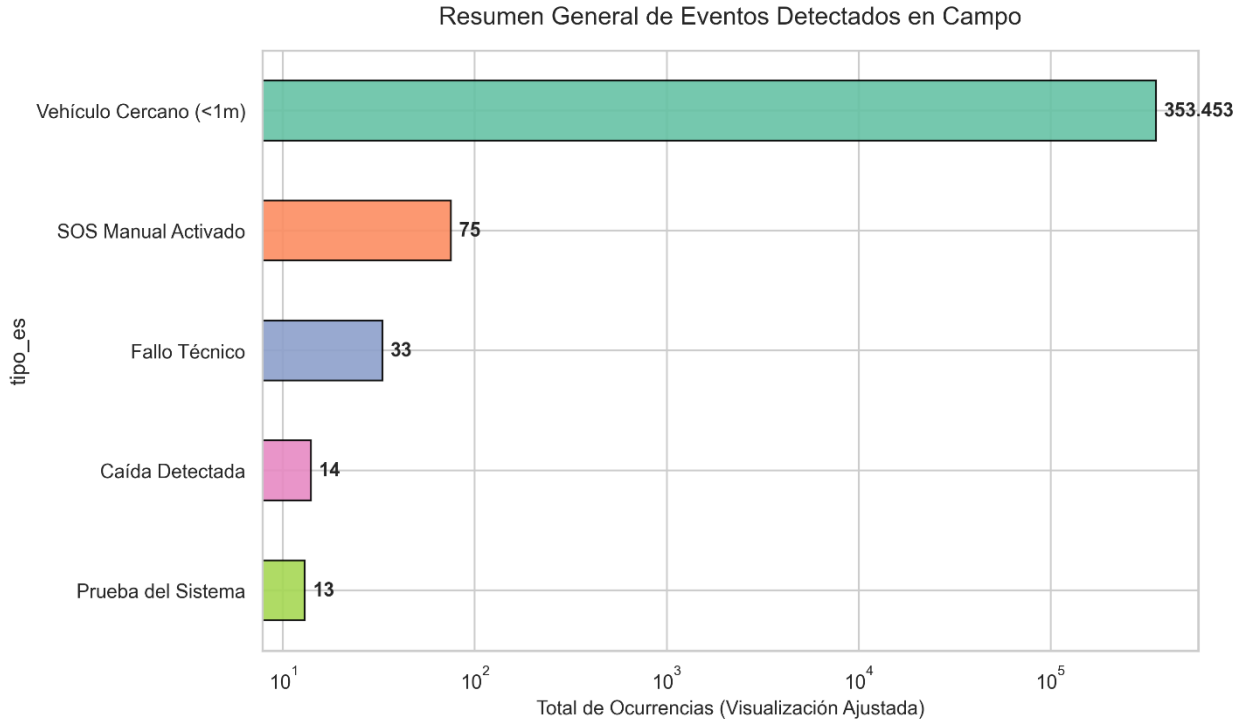
Nota. Autoría propia

Sobre el análisis de incidentes críticos durante el trayecto, el sistema se apoya mediante la integración de alertas externas, las cuales arrojaron resultados concluyentes. Las notificaciones de alerta por actuadores de la aplicación móvil como son “SOS” y “Avería mecánica” lograron localizar la posición exacta mediante la utilización de los parámetros de longitud y latitud del

ciclista al momento del evento, que al momento de realizar una contabilización se registró una tasa de 75 activaciones por parte del botón de pánico y 33 por parte de avería mecánica con un tiempo de latencia mínimo hacia el servidor, esto se detalla de mejor manera mediante la figura 118.

Figura 118

Consolidado de alerta y detecciones emitidas por el sistema en entorno real.



Nota. Autoría propia

Finalmente, cabe destacar que si bien el sistema SARC-UTN se mantiene en funcionamiento en campo y recolectando información incluyendo datos posteriores a la validación oficial, este presente análisis estadístico se apega a los datos consolidados y registrados hasta las primeras semanas de mayo del 2026. Esta limitación de datos históricos asegura la integridad de la etapa de validación del sistema, aunque el sistema completo sigue funcionando y la capa red soportando la ingesta de información en los recorridos con el fin de obtener y verificar los cambios obtenidos a largo plazo.

4.8 Discusión de resultados

La transición del controlador de carga inicial (TP4056 1A) hacia el módulo IP5328P (3A) esto permitió tener una mejora significativa en la capacidad del suministro energético, sino que además una medida critica en la mitigación de riesgo. Donde se determinó que los picos de transmisión del modem EG25-G generaba caídas de tensión cuando intenta conectarse hacia la red calor o cal cambiar de estación que desestabilizaban al microordenador, por consecuencia, el cambio a 3A fue el punto más alto para garantizar la tolerancia a fallos del hardware. La vulnerabilidad de los dispositivos conectados a la capa percepción quedo en evidencia tras tener una corrupción del sistema en la memoria micro SD que fue ocasionado por un corte brusco de energía, no obstante, para mitigar estas vulnerabilidades fue necesario la implantación de una política de respaldo de imagen (backups) lo que permitió transformar el cuello de botella en una fuerte arquitectura.

En el contexto sobre la aplicación nativa móvil en Python, la utilización de procesamiento asíncrono mediante la utilización de hilos en segundo plano (Threading) ayuda o a impulsar significativamente la fluidez de la interfaz gráfica convirtiéndose en un pilar fundamental del requerimiento de seguridad. Concluyendo que al evitar bloquear el hilo principal permite al ciclista no perder su ubicación en el mapa y además garantizando que una ráfaga de alertas SOS no lo deje incomunicado en el momento más crítico de una emergencia.

Sobre la calibración del sensor de proximidad, no se basó en criterios arbitrarios, sino que fue de acuerdo al estricto de la Ley Orgánica de Transporte Terrestre (1.5m distancia lateral). De igual manera, la validación demostró que, si bien las edificaciones urbanas introducen una desviación en la triangulación satelital en comparación con rutas abiertas, el margen de error

operativo con el que cuenta el modem EG25-G se mantiene entre los límites de tolerancia para permitir la ubicación de siniestros.

Finalmente, los tiempos de latencia que arrojaron en las pruebas del sistema con referencia hacia los servidores externos, fueron inferiores a los 10 segundos considerando así altamente aceptables para la gestión de incidentes en tiempo real. Al contrastar este tiempo de retardo con la reacción del humano y logístico promedio ante un accidente de tránsito, el sistema otorga una ventaja notable frente a servicios de urgencias. Demostrando así una tasa alta de éxito operativo, además reconociendo las mejoras a futuro de las operaciones de las redes celulares públicas.

4.9 Costo del sistema

El sistema fue realizado mediante la implantación de elementos económicos y disponibles la mayoría en el país, logrando crear un sistema eficiente pero económico, además, el costo se dividido en dos partes, CAPEX lo que es el costo del hardware y una sola vez y el OPEX que es el costo de los sistemas externos. Gracias a la utilización de la capa gratuita de Supabase, el costo de mantener estos datos con seguridad en la nube fue de cero dólares, consecuentemente, para la utilización de un disparador de alertas de emergencia se gastó en el manejo de la API Twilio ya que funciona bajo el modelo de pago por uso no hay cuotas mensuales que molesten al ciclista. Armar este sistema tuvo un costo de **$285.05 USD** llegando así a compararse con dispositivos en el mercado, los detalles de los costos esta visualizados en la tabla 27.

Tabla 27

Costo General del dispositivo

COSTO DEL SISTEMA		
HARDWARE		
COMPONENTE	**CANTIDAD**	**SUBTOTAL (USD)**

Microcontrolador ESP32	1	$ 12.50
Raspberry Pi Zero 2W	1	$ 70.00
Módem USB LTE/4G	1	$ 71.00
Módulo GPS (NEO-6M)	1	$ 11.50
Sensor Ultrasónico de distancia (HC-SR04)	1	$ 3.50
Acelerómetro/Giroscopio (MPU6050)	1	$ 3.50
Tira LED NeoPixel	2	$ 7.00
Batería Lipo 3.7V 10000 mAh	1	$ 10.50
Modulo IP5328P	1	$ 10.50
Otros	1	$ 10.00
TOTAL		**$ 210.00**

SOFTWARE

SOFTWARE	**CANTIDAD**	**SUBTOTAL**
Python & Node-RED	1	$ 0
Supabase	1	$ 0
Flet (Open Source)	1	$ 0
Mosquitto / HiveMQ	1	$ 0
Twilio	1	$ 25.00
TOTAL		**$ 25.00**

CONECTIVIDAD

CONECTIVIDAD	**CANTIDAD**	**SUBTOTAL**
Plan de Datos 4G	1	$ 10.05
TOTAL		**$ 10.05**

INFRAESTRUCTURA		
INFRAESTRUCTURA	**CANTIDAD**	**SUBTOTAL**
Case 3D del dispositivo	1	$ 40.00
TOTAL		**$ 40.00**
TOTAL (HARDWARE + SOFTWARE + INFRAESTRUCTURA + CONECTIVIDAD)		**$ 285.05**

Cabe recalcar que el costo total estimado de **$285.05 USD c**omprende únicamente a los valores directos de los componentes, conectividad y la infraestructura necesaria para el desarrollo el sistema. Esta cifra no está estipulando gasto de importaciones, tales como microordenadores, modulo LTE y baterías de Lipo, los cuales pueden variar significativamente en función del servicio de Courier de preferencia, impuestos aduaneros y tiempos de entrega.

Luego de haber desglosado cada componente utilizado con su costo directo en la tabla 28, se realizó una tabla resumen en la cual se simplifica el costo total asociada al desarrollo del sistema. En la tabla 26 resume los gastos realizados considerando cada componente para la construcción del sistema, permitiendo así obtener una visión más clara del presupuesto total para la implementación del sistema.

Tabla 28

Resumen del costo general del sistema

COSTO GENERAL DE SISTEMA	
Tipo de Costo	Subtotal (USD)
Costo de hardware	$ 210.00
Costo de software	$ 25.00
Costo de infraestructura	$ 40.00

Costo de conectividad	$ 10.05
TOTAL (USD)	**$ 285.05**

Conclusiones

Como resultado, se comprobó que la viabilidad empírica y operativa de la integración del hardware entre el microordenador Raspberry Pi Zero 2W y el microcontrolador de borde ESP32-WROOM-32 para la adquisición de los datos en tiempo real. Esta arquitectura no solo satisface, sino que supera cómodamente los requerimientos funcionales establecidos en la etapa de diseño, con lo que se garantiza una orquestación eficiente y con una latencia baja en la transmisión periódica de alertas suscitadas por proximidad o por eventos críticos, sin evidenciar cuellos de botellas en el bus de comunicación a nivel local.

Por consiguiente, el cambio de hardware del circuito de alimentación gestionado por el módulo IP5328P se establece como uno de los factores determinantes para sostener la estabilidad del sistema, llegando así a erradicar completamente las caídas de tensión generadas por los picos variables que consume el módulo LTE al realizar el Handover y el establecimiento de la señal con eNodeB.

En consecuencia, frente a las severas limitaciones con respecto al multitrayecto y la caída drástica del SINR, se determina que la fiabilidad del enlace no recae exclusivamente en el hardware, además, se comprobó que la implementación de scripts de recuperación autónomo denominado Watchdog que está basado en el tanteo ICMP y la configuración del Keep-Alive que trabaja en el protocolo MQTT son de carácter obligatorio puesto que al operar el sistema bajo el enlace LOS, esto asegura una recuperación autónoma del sistema.

Así, la calibración restrictiva del umbral del sensor ultrasónico a una distancia aproximadamente de 1.5 metros logro comprobar la precisión geométrica del sistema, y rigiendo en la normativa de la ley de tránsito de Ecuador con respecto al rebase de ciclistas.

Para finalizar, el cerebro del sistema en lo que se refiere al funcionamiento en la nube tuvo una respuesta muy aceptable en lo que se refiere al almacenamiento de los datos recibidos, por ende, el utilizar el gestor de datos en la nube Supabase permitió proteger y tener privacidad de la información sensible del usuario esto gracias a las reglas de seguridad RLS (Row Level Security) que actúan como escudo a nivel de PostgreSQL que permiten bloquear las consultas de personas no autorizadas previniendo el rastreo malicioso, para lograr esto, el microprocesador fue punto clave puesto que al enviar datos con cargas útiles en formato JSON que ocupas pocos bytes, permitió que la inyección al gestor de datos Supabase no tenga casi consumo de datos móviles y sea instantánea. En conclusión, la lección de utilizar el API RESTful de Twilio para realizar el envió de alertas por SMS no fue elegido por antigüedad, sino que, esta plataforma permitió asegurar la integridad operativa del sistema y del usuario, puesto que aplicaciones actuales de mensajería que depende netamente que el receptor tenga acceso a internet, el servicio de mensajería SMS utilizado viaja a través del canal de señalización nativo que ofrece la operadora (SS7). Esto significa que el dispositivo fue capaz de enviar la alerta y asegurar que llegue directamente al teléfono del familiar incluso si el sistema tiene una conexión de 2G en ese instante.

En resumen, la arquitectura de todo el sistema demostró empíricamente que este sistema no solo responde a una necesidad tecnológica, sino que entrega una herramienta perceptible que promueve el salvaguardar vidas reduciendo drásticamente los tiempos de respuestas de equipos de emergencia ante siniestros viales.

Recomendaciones

Para reflejar la precisión obtenida en las pruebas, se aconseja de manera estricta que es necesario fijar la carcasa de los componentes al chasis de la bicicleta utilizando correas con aislamiento de goma para evitar las vibraciones que se suscitan. Esta adecuación permite que las vibraciones ocasionadas por el asfalto no interfieran en la lectura del giroscopio. Asimismo, dado que el carácter crítico que enfrente el sistema, es necesario realizar una capacitación a los familiares y que comprendan la eficacia y a su vez la gravedad de recibir un SMS y que acción tomar sin entrar en pánico.

Dado el buen rendimiento del sistema, se sugiere migrar la aplicación para dispositivos de Apple (iOS). Al utilizar del framework Flet basado en Python, no requiere modificar la lógica del sistema, puesto que el código base ya es funcional, este despliegue se omitió únicamente por la falta de presupuesto económico para obtener las licencias.

Para obtener una duración de batería aún más prolongada, se recomienda realizar una actualización al código Deep Sleep del ESP32, puesto que en algunos tramos de la ruta por cualquier circunstancia el ciclista frena por un tiempo, esta actualización permitirá suspender temporalmente tanto al microcontrolador y modulo LTE, duplicando así la autonomía de la batería instalada.

Para desplegar el sistema a nivel comercial, se recomienda mejorar el hardware de radiofrecuencia del módulo LTE mediante la utilización de antenas MIMO esto ayudará a mejorar la recepción de la señal (SINR) en geografías extremas, y como complemento implementar una eSIM que permitirá conmutar automáticamente la red entre operadoras.

Bibliografía

Ahmed, S., Huda, N., Rajbhandari, S., Saha, C., Elshaw, M., & Kanarachos, S. (2019). Pedestrian and Cyclist Detection and Intent Estimation for Autonomous Vehicles: A Survey. *Applied Sciences 2019, Vol. 9, Page 2335*, *9*(11), 2335. https://doi.org/10.3390/APP9112335

Álvarez, D. (2014). *Diseño de una red 4G (Long Term Evolution) en el cluster 2 de la ciudad de Quito para el operador CNT. EP*. https://repositorio.utn.edu.ec/handle/123456789/3771

Azzez, B., Kathoon, A., Sunil, F., Sathyan, D., Sebastian, R., & Varughese, I. (2024). *Integrated Smart Bike Safety System with Accident Detection*. http://www.ijert.org

Breda, J., Allen, P. G., Chen, K., Ploetz, T., & Patel, S. (2025). ProxiCycle: Passively Mapping Cyclist Safety Using Smart Handlebars for Near-Miss Detection. *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26â•fiMay 01, 2025, Yokohama, Japan*, *1*. https://doi.org/10.1145/3706598.3713325

Caín, V. (2021). *Implementación de un sistema de alerta para la prevención de accidentes mediante reconocimiento de señales de tránsito y visión artificial.* Tesis de Grado, Universidad Nacional de Chimborazo.

Chamba, K., & Pesántez, J. (2023). *Evaluación de la seguridad y puntos críticos de la ciclovía de la Ciudad de Cuenca*. Tesis de Grado, Universidad Politécnica Salesiana.

CycleAlert. (2023). *How it Works*.

De Smith, G., Yeleshetty, D., Havinga, P., & Huang, Y. (2024). Predicting Turn Maneuvers of Cyclists Using Bicycle-Mounted IMU with CNN-LSTM 1 st Gijs de Smit. *Journal of Transportation Safety & Security,* *2*(16), 145–158. https://doi.org/10.1109/PerComWorkshops59983.2024.10502681

Eaton. (2025). *Sensor de proximidad inductivo*. https://www.eaton.com/mx/es-mx/products/controls-drives-automation-sensors/sensors---limit-switches/inductive-proximity-sensor.html

Festo. (2024). *Comprar Sensor de proximidad online | Festo CO*. https://www.festo.com/co/es/c/productos/sensores/sensor-de-proximidad-id_pim128/

Hadi, F., Putrada, A., & Abdurohman, M. (2021). Accident Detection System for Bicycle Athletes Using GPS/IMU Integration and Kalman Filtered AHRS Method. *2021 International Conference Advancement in Data Science, E-Learning and Information Systems, ICADEIS 2021*. https://doi.org/10.1109/ICADEIS52521.2021.9702085

IAD. (2024). *Giroscopio*. https://iad.la/blogs/giroscopio-que-es/

Imove. (2021). *Vehicle to bicycle (V2B) safety interactions using 4G mobile devices*. https://imoveaustralia.com/project/vehicle-to-bicycle-v2b-safety-interactions/

INEC. (2024). *Resultados de siniestros de tránsito – IV trimestre 2024.* https://www.ecuadorencifras.gob.ec/institucional/home/

Jabbar, R., Shinoy, M., Kharbeche, M., Al-Khalifa, K., Krichen, M., & Barkaoui, K. (2020). Urban Traffic Monitoring and Modeling System: An IoT Solution for Enhancing Road Safety. *2019 International Conference on Internet of Things, Embedded Systems and Communications, IINTEC 2019 - Proceedings*, 13–18. https://doi.org/10.1109/IINTEC48298.2019.9112118

Kaplan, E., & Hegarty, C. (2017). Understanding GPS/GNSS: Principles and Applications, Third edition. En *International Ocean Systems* (Vol. 18, Número 6).

La Hora. (2024). *Carchi activa una campaña para evitar atropellos de ciclistas*. https://www.lahora.com.ec/imbaburacarchi/Carchi-activa-una-campana-para-evitar-atropellos-de-ciclistas-20250120-0003.html

Moscoso, D., & Romero, D. (2020). *Desarrollo de un prototipo de radiobaliza para ciclistas, integrado en la bicicleta y el casco, con la capacidad de detección de impactos en la cabeza y emisión de alertas* [Tesis de Grado, Universidad Politécnica Salesiana]. http://dspace.ups.edu.ec/handle/123456789/18898

Reddit. (2020). *Sensor for detecting cyclists / URM06 ultrasonic sensor: r/ArduinoProjects*. https://www.reddit.com/r/ArduinoProjects/comments/hl2vpd/sensor_for_detecting_cyclists_urm06_ultrasonic/

Reyes, M., & Quishpe, D. (2017). *Diseño e implementación de un sistema basado en GPS para el anuncio de paradas en las rutas de los autobuses urbanos de la ciudad de Ibarra* [Tesis de Grado, Universidad Técnica del Norte]. https://repositorio.utn.edu.ec/handle/123456789/7008

Schnee, J., Stegmaier, J., & Li, P. (2021). A probabilistic approach to online classification of bicycle crashes. *Accident Analysis and Prevention*, *160*. https://doi.org/10.1016/j.aap.2021.106311

Titterton, D., John, L., & Weston. (2005). *Strapdown Inertial Navigation Technology - 2nd Edition*.

UNIR. (2025). *¿Qué es LTE y cómo funciona? Diferencias con 4G y 5G | UNIR*. https://www.unir.net/revista/ingenieria/lte-que-es/

Vaca, J. (2020). *Implementación de un juego educativo en Javascript para una Plataforma móvil*. Tesis de Grado, Universidad Politécnica Salesiana.

Wang, X., Li, j, & Chen, Z. (2022). Wang, X., Li, J., & Chen, Z. (2022). Accident Detection System for Cyclists Using GPS-IMU Fusion and Kalman Filtering. 22(15), 14763–14774.

Weg. (2024). *Sensores de proximidad*. https://www.weg.net/catalog/weg/KN/es/Seguridad-de-M%C3%A1quinas%2C-Sensores-Industriales-y-Fuentes-de-

Alimentaci%C3%B3n/Sensores-Industriales/Sensores-de-proximidad-capacitivos/Sensores-de-proximidad-capacitivos/p/MKT_WDC_GLOBAL_CAPACITIVE_SENSORS

Woodman, O. J. (2007). *An introduction to inertial navigation*. https://www.cl.cam.ac.uk/techreports/

Espressif Systems. (2023). *ESP32 Series Datasheet*. https://www.espressif.com

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4), 2347–2376.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660.

Arduino. (2025). Arduino IDE Software Release Notes. https://www.arduino.cc/en/software

Cytron Technologies. (2023). Ultrasonic Sensor HC-SR04 Datasheet. https://www.cytron.io

Elecfreaks. (2026). Ultrasonic Sensor HC-SR04 User Guide. https://www.elecfreaks.com

InvenSense. (2013). MPU-6000 and MPU-6050 Product Specification Revision 3.4. TDK Corporation.

OpenJS Foundation. (2026). Node-RED: Low-code programming for event-driven applications. https://nodered.org/

Osoyoo. (2017). SW-420 Vibration Sensor Module User Guide. https://osoyoo.com

Quectel Wireless Solutions. (2019). EG25-G Hardware Design v1.3. https://www.quectel.com

Raspberry Pi Foundation. (2023). Raspberry Pi OS: The official Raspberry Pi operating system. https://www.raspberrypi.com

u-blox. (2011). NEO-6 u-blox 6 GPS Modules Data Sheet. https://www.u-blox.com

Md Dzahir, M. A. S., & Chia, K. S. (2023). Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System. *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. IEEE. https://doi.org/10.1109/3ICT60104.2023.10391358

Hidalgo-Fort, E., Blanco-Carmona, P., Muñoz-Chavero, F., Torralba, A., & Castro-Triguero, R. (2024). Low-Cost, Low-Power Edge Computing System for Structural Health Monitoring in an IoT Framework. *Sensors, 24*(15), 5078. https://doi.org/10.3390/s24155078

Moradi-Pari, E., Tian, D., Bahramgiri, M., Rajab, S., & Bai, S. (2023). DSRC Versus LTE-V2X: Empirical Performance Analysis of Direct Vehicular Communication Technologies. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2023.3247339

Kanwal, S., Abdullah, M., Kumar, S., Arshad, S., Shahroz, M., Zhang, D., & Kumar, D. (2024). An Optimal Internet of Things-Driven Intelligent Decision-Making System for Real-Time Monitoring. *Sensors, 24*(23), 7842. https://doi.org/10.3390/s24237842

Navia, M., Macías-Aguayo, J., & Quiroz-Cordova, D. (2024). IoT-Based Detection of Blockages in Stormwater Drains. *Engineering Proceedings*. MDPI. https://doi.org/10.3390/ecsa-11-20371

Asamblea Nacional del Ecuador. (2021). Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial (LOTTTSV). Recuperado de https://www.obraspublicas.gob.ec

Morales, Y., Trujillo, D., & Gualapuro, J. (2025). Aplicaciones del Internet de las Cosas (IoT) en la Industria Electromecánica para la recolección, análisis y monitoreo en tiempo real. Revista Ingeniería Global. Recuperado de https://www.researchgate.net/publication/387791886

Mendoza, E., Gonzabay, E., & Mendoza, V. (2022). Analizando Internet de las Cosas y la nube informática. Revista Odigos. Recuperado de https://revista.uisrael.edu.ec/index.php/ro/article/view/535

Rivera, J., Gonzabay, E., & Mendoza, B. (2022). El uso del software de código abierto en el IoT. Ciencia Latina Revista Científica Multidisciplinar. Recuperado de https://www.researchgate.net/publication/365863034

Zhang, Y., & Zheng, L. (2022). ESP32: A Comprehensive Review of Features and Applications. IEEE Access. Recuperado de https://ieeexplore.ieee.org/document/9745678

Sofianidis, I., Konstantakos, V., & Nikolaidis, S. (2025). Exploring Efficiency: Comparative Study of Microcontroller Energy Consumption and Performance Optimization Strategies. MDPI Electronics. Recuperado de https://www.mdpi.com/2227-7080/13/2/82

GetApp. (2025). Best IoT Apps for Android 2025. Recuperado de https://www.getapp.com/emerging-technology-software/iot/os/android/

Slashdot. (2025). Top IoT Apps for Android in 2025. Recuperado de https://slashdot.org/software/iot/android/

University of San Carlos of Guatemala. (2018). Fundamentos para transmitir datos de una red móvil celular. Recuperado de https://biblioteca.usac.edu.gt/tesis/08/08_0275_CS.pdf

Dialnet. (2021). La telefonía móvil de cuarta generación 4G. Recuperado de https://dialnet.unirioja.es/descarga/articulo/5972775.pdf

Revista ELECTRO. (2024). Análisis comparativo de tecnologías Bluetooth, WiFi, ZigBee y LoRa. Recuperado de https://itchihuahua.mx/revista_electro/2024/A26_148-153.pdf

Microsoft Tech Community. (2025). Storing IoT Data in Azure: SQL vs Cosmos vs Other Methods. Recuperado de https://www.researchgate.net/publication/387791886

Zieliński, B. M. (2021). A Comparison of Proximity Sensors for a Bicycle-to-Car Distance Rangefinder. *International Journal of Electronics and Telecommunications*, 67(4).

Cordovez, S., Ortiz-Prado, E., Vasconez, E., Andrade, F., Simbaña-Rivera, K., Gómez-Barreno, L., & McIlroy, R. C. (2021). Bicycling-Related Mortality in Ecuador: A Nationwide Population-Based Analysis from 2004 to 2017. *Sustainability*, 13(11), 5906.

Institute of Electrical and Electronics Engineers [IEEE]. (2003). *IEEE 802.11g-2003: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Standards Association.

Servicio Ecuatoriano de Normalización [INEN]. (2014). *Reglamento Técnico Ecuatoriano RTE INEN 004*. Señalización Vial. Quito, Ecuador.

Asamblea Nacional del Ecuador. (2021). *Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial (LOTTTSV)*. Registro Oficial, Suplemento 512. Quito, Ecuador.

Presidencia de la República del Ecuador. (2012). *Reglamento General para la Aplicación de la Ley Orgánica de Transporte Terrestre, Tránsito y Seguridad Vial*. Decreto Ejecutivo Nro. 1196. Registro Oficial Suplemento 731. Quito, Ecuador.

Asamblea Nacional del Ecuador. (2021). *Código Orgánico Integral Penal (COIP)*. Versión reformada y actualizada a febrero de 2021. Registro Oficial. Quito, Ecuador.

Anexos

MANUAL DEL USUARIO CICLISTA

Este manual está orientado al ciclista. el objetivo es garantizar la correcta instalación del hardware en el chasis de la bicicleta y la configuración inicial de la aplicación.

1. Configuración del Contacto de Emergencia en la App

Antes de iniciar su primer recorrido y trazar la ruta, es estrictamente necesario si tiene asignado alguien contacto de emergencia caso contrario será necesario configurar a la persona que recibirá las alertas en caso de accidente.

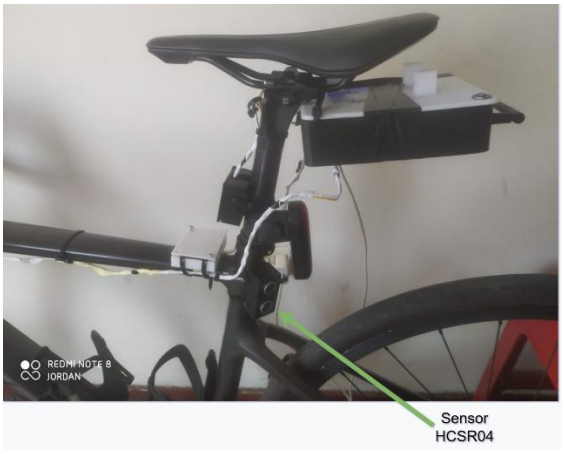
- Paso 1: Abra la aplicación SARC en su teléfono móvil e inicie sesión con sus credenciales.
- Paso 2: Diríjase al menú de configuración y seleccione la opción "config".
- Paso 3: Ingrese el número de teléfono móvil de su familiar designado. Importante: Asegúrese de incluir el código de país (Ej: +593 para Ecuador).
- Paso 4: Presione Guardar.



2. Montaje Físico en la Bicicleta

El sistema depende de una instalación física precisa para evitar falsas alarmas y asegurar la conexión.

- Fijación: Utilice las correas de goma para sujetar firmemente la carcasa al tubo del asiento. Evite dejarla suelta; la vibración extrema generará alertas falsas de caída.
- Módulo GPS (Señal Satelital): Asegúrese de que la cara superior de la carcasa quede totalmente libre de obstrucciones (sin partes metálicas encima) para garantizar línea de vista directa con el cielo.
- Sensor Ultrasónico: Verifique que el sensor apunte en línea recta hacia la parte trasera de la bicicleta. Asegúrese de que el movimiento de sus piernas al pedalear no cruce por el rango de visión del sensor.



3. Encendido y Verificación de LEDs

Antes de iniciar su recorrido, confirme que el sistema está 100% operativo a través del indicador LED externo.

- Paso 1: Encienda el dispositivo usando el interruptor principal.
- Paso 2: El indicador luminoso se encenderá en color AZUL. Esto significa que el sistema está iniciando y buscando red celular y satélites.

- Paso 3: Espere a que el indicador luminoso cambie a color VERDE fijo. Esto confirma que la señal y la conexión con el servidor están en línea. No inicie su ruta hasta ver la luz verde.

4. Uso del Botón de Emergencia (SOS)

Si usted sufre una avería mecánica grave, se siente amenazado o requiere asistencia sin haber sufrido una caída:

- Abra la aplicación móvil SARC.
- Presione el Botón SOS en la pantalla principal. Esto disparará una alerta instantánea a su familiar con su ubicación actual.

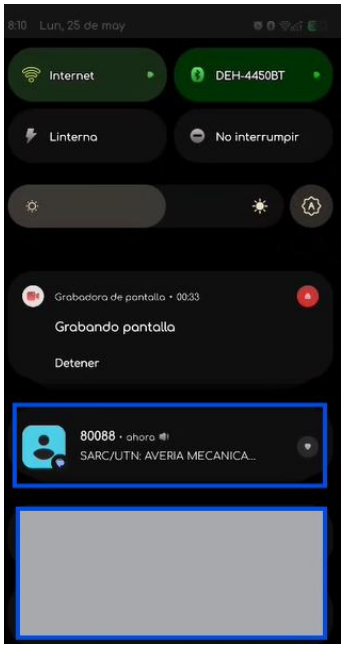
MANUAL DEL USUARIO FAMILIAR

Esta sección está dirigida al familiar o contacto de emergencia designado. Su objetivo es instruirlo sobre cómo actuar inmediatamente tras recibir una alerta.

1. Recepción de la Alerta (SMS)

El sistema funciona de forma autónoma. Si el ciclista sufre una caída brusca o presiona el botón de pánico, usted será notificado de inmediato.

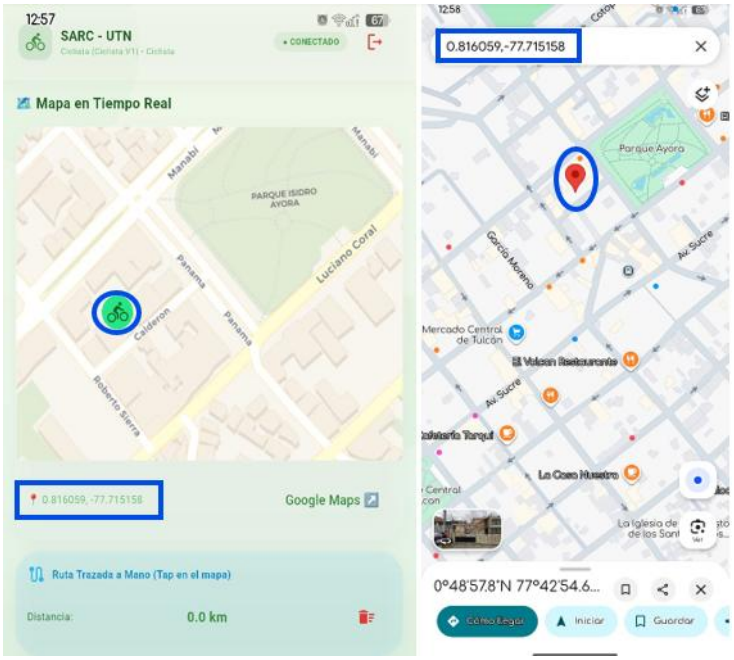
- Paso 1: Si usted recibe un mensaje de texto (SMS) en su teléfono con el encabezado "Alerta SARC", trátelo como una emergencia prioritaria. No es un mensaje automatizado de publicidad.
- Paso 2: Abra inmediatamente la aplicación móvil SARC en su dispositivo inteligente.



2. Uso de la Aplicación y Toma de Decisiones

Una vez dentro de la plataforma, usted tendrá acceso en tiempo real a la ubicación del accidente.

- Paso 1: Ingrese a la aplicación con sus credenciales de "Usuario Familiar".
- Paso 2: En la pantalla principal, ubique el marcador verde en el mapa. Este punto indica las coordenadas GPS (latitud y longitud) exactas donde se encuentra el ciclista en ese instante.
- Paso 3: Intente realizar una llamada telefónica convencional al ciclista para descartar falsas alarmas o daños menores.
- Paso 4 (Crítico): En caso de no lograr comunicación telefónica con el ciclista, mantenga la calma y proporcione las coordenadas del mapa directamente a los servicios de emergencia locales (Ej: ECU-911) para facilitar un rescate inmediato.


12:57
SARC - UTN
Mapa en Tiempo Real
PARQUE ISIDRO AYORA
Manabi
Panama
Luciano Coral
Calderon
Roberto Sierra
0.816059, -77.715158
Google Maps
Ruta Trazada a Meno (Tap en el mapa)
Distancia:
0.0 km
0.816059,-77.715158
Parque Ayora
Garcia Moreno
Av. Sucre
Mercado Central de Tulcán
El Volcan Restaurante
La Casa Nuestra
0°48'57.8"N 77°42'54.6...
Iniciar
Guardar