UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de geolocalización, telemetría y diagnóstico de unidades vehiculares bajo el concepto de IoT

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Angel Joshua Rosas Garcia

DIRECTOR DE TESIS

Dr. Saúl de la Rosa Nieves

Ciudad Universitaria, Cd. Mx., 2025

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado SISTEMA DE GEOLOCALIZACION, TELEMETRIA Y DIAGNOSTICO DE UNIDADES VEHICULARES BAJO EL CONCEPTO DE IOT. que presenté para obtener el titulo de INGENIERO ELÉCTRICO ELECTRÓNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

ANGEL JOSHUA ROSAS GARCIA
Número de cuenta: 314329170

Agradecimientos

A mis padres, quienes sembraron en mí los valores fundamentales que me convirtieron en la persona que soy hoy. Su apoyo incondicional a lo largo de mi etapa universitaria me permitió jamás sentirme solo. Este logro es tanto mío como suyo, fruto del esfuerzo y sacrificio de ambos. Gracias por inculcarme la bondad, la humildad y la disciplina, valores que día a día han guiado mi camino.

A mis hermanos, Emmanuel, Moisés y Daniel, quienes me regalaron la infancia más feliz que pude haber deseado. Junto a ellos aprendí el verdadero significado de la lealtad y el apoyo incondicional. Este triunfo también es para ellos, porque el esfuerzo y la dedicación de cada uno en sus respectivas áreas reflejan la gran familia que tenemos, una familia llena de amor y que me ha acompañado y ayudado a desarrollarme de la mejor manera.

A mis amigos, Ciro, Guillermo, Nicole, Vanesa, Quezni, Carvajal, Ocaña y Sharon, quienes son la familia que uno elige. Me siento afortunado de contar con un grupo de amigos sólidos, con quienes compartí hombro a hombro esta travesía académica. En ellos encontré apoyo, compañía y consuelo en los momentos difíciles.

A mi tutor, el Doctor Saul, quien me guió en el camino de la ingeniería y avivó mi pasión por la electrónica. A través de sus enseñanzas, consejos y experiencia, me ayudó a crecer tanto en el ámbito académico como en el profesional. De él aprendí que la constancia y el trabajo duro siempre dan grandes frutos. Como él mismo me dijo en más de una ocasión: «La suerte no es otra cosa que estar preparado cuando se te presenta la oportunidad. ».

A mis mentores, Aldair y Beca, cuyo ejemplo ha sido una fuente de inspiración para muchos estudiantes y amigos. De ellos aprendí que la constancia y la pasión por la investigación y el desarrollo abren puertas inimaginables, siempre acompañadas de calidad humana y un amor incondicional que trasciende lo académico

A mi universidad, mi segunda casa, donde viví experiencias inolvidables que me formaron tanto personal como profesionalmente. Más que un espacio de formación académica, fue un entorno donde aprendí sobre la sociedad, la colaboración y el pensamiento libre, aspectos que han enriquecido mi crecimiento.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

GNSS Sistema de Navegación por Satélite

GSM Sistema Global para Comunicaciones Móviles

GPRS Servicio General de Radio por Paquetes

HLR Registro de Localización Domiciliaria

VLR Registro de Ubicación de Visitantes

MCS Centro de Comunicación de Servicios Móviles

BSS Sistema de Estación Base

BTS Estación Transceptora Base

BSC Controlador de Estaciones Base

GMSC Puerta de Enlace MSC

SMSC Puerta de Enlace SMS

PSTN Red Telefónica Pública

PLMN Red de Telecomunicaciones Móviles Públicas

FDMA Frecuencia de División Múltiple

SCC Sistema Central de Control

CTS Red de Estaciones de Seguimiento y Control

ECU Unidad de Control Electrónico

ABS Sistema Antibloqueo de Frenos

ESC Control Electrónico de Velocidad Crucero

CAN Red de Control de Área

PS Señalización Física

LCC Control de Enlace Lógico

MAC Medio de Control de Acceso

NRZ No Retorno a Cero

CC Control de Climatización

SSS Sistema de Dirección y Suspensión

RTS Transmisión Remota

SSR Sustitución de Solicitud Remota

PID Identificación de Parámetro

DTC Códigos de Detección de Fallas

PCU Unidad de Control de Paquetes

SGSN Nodo de Soporte GPRS de Servicios

GGSN Nodo de Soporte GPRS de Puerta de Enlace

GPS Sistema de Posicionamiento Global

GLONASS Sistema de Navegación por Satélite en Órbita Global

OBD-II Diagnóstico a Bordo II

UART Receptor-Transmisor Asíncrono Universal

I2C Inter-Integrated Circuit

AT Comandos AT

CMI Módulo de Control del Sistema

GC Módulo de Localización y Telecomunicaciones

SV Módulo de Monitoreo de Conducción y Alertas

CUCM Módulo de Comunicación con la ECU

P Módulo de Gestión de Energía y Alimentación

Parte I

Planteamiento del proyecto

1 Introducción

La industria automotriz ha experimento un rápido crecimiento y evolución, con avances tecnológicos significativos y expansión global de las compañías del sector. Esto ha facilitado la adquisición y mantenimiento de flotas vehiculares, permitiendo que más empresas integren vehículos en sus operaciones.

Actualmente, el uso de flotas vehiculares es esencial para múltiples industrias, incluyendo logística, transporte personal, construcción, distribución de productos y servicios, turismo y transporte público. Sin embargo, la gestión eficiente de estas flotas representa un desafió, ya que factores como la optimización operativa, la seguridad vial, el mantenimiento preventivo y la reducción de costos son clave para maximizar la rentabilidad y eficiencia de las operaciones empresariales.

El uso indebido de los vehículos de la flota, como viajes no autorizados, desvíos de rutas y robo de combustible, pueden incrementar considerablemente los costos operativos y reducir la vida útil de los activos. Asimismo, una conducción inadecuada (exceso de velocidad, ralentí prolongado, maniobras bruscas, etc.) no solo compromete la seguridad del conductor y de terceros, si no también incrementa el consumo de combustible y desgaste del vehículo.

En respuesta a esta necesidad, han surgido diversas empresas especializadas en gestión de flotas vehiculares que ofrecen plataformas avanzadas para la localización, diagnóstico y control de vehículos a través de internet. Sin embargo, estas soluciones suelen ser muy costosas y tener limitaciones en la personalización de los servicios.

En el presente trabajo se muestra el desarrollo de un sistema de geolocalización, telemetría y diagnóstico, basado en la diseño y manufactura de hardware con capacidad de obtener coordenadas GPS y transmitir datos a través de internet. Este proyecto tiene como objetivo principal desarrollar e integrar tecnología de firmware y hardware que cumpla con características funcionales similares a las presentes en dispositivos comerciales ya posicionados en el mercado.

1.1. Objetivos generales

Integración del hardware y firmware de un sistema embebido para monitoreo, diagnóstico y geolocalización para flotas vehiculares basado en un microcontrolador e integrado con tecnologías Sistema de Navegación por Satélite (GNSS, por sus siglas en inglés) y Sistema Global para Comunicaciones Móviles/Servicio General de Paquetes por Radio (GSM/GPRS, por sus siglas en inglés) para la transmisión de datos a internet.

1.2. Objetivos específicos

- Diseño y construcción de un módulo Comando y Manejo de Información (CMI), basado en un microcontrolador programable, que incorpore un firmware de control encargado de gestionar la adquisición, procesamiento y envió de datos de telemetría, geolocalización y movimiento hacia una plataforma remota mediante conexión a internet.
- Implementación de un módulo GNSS para obtención de coordenadas geográficas con precisión suficiente para aplicaciones de monitoreo y gestión de flotas vehiculares.
- Implementación de un módulo GSM/GPRS que permita establecer conectividad con redes móviles para la trasmisión continua de datos.
- Desarrollo de un módulo de comunicación OBD, capaz de acceder a parámetros de diagnóstico vehicular, con el objetivo de obtener información relevante sobre el estado operativo del sistema automotriz.

- Implementación de una interfaz gráfica para la visualización de la información recopilada por el sistema.

1.3. Alcance

Este trabajo tiene como alances el diseño y fabricación de un módulo de Comando y Manejo de Información, basado en un microcontrolador programable, encargado de la gestión de la comunicación y del control de los diferentes subsistemas que conforman el sistema de geolocalización, telemetría y diagnóstico vehicular.

El sistema será compatible con tecnologías GNSS y GSM/GPR, permitiendo la adquisición de coordenadas geográficas y la transmisión de datos de telemetría a una plataforma remota a través de Internet. Además, se integrarán los elementos necesarios para la comunicación bajo el estándar OBD-II, con el fin de diagnosticar al vehículo.

Como parte importante del proyecto, se desarrollará un firmware embebido bien estructurado, basado en librerías propias para la gestión eficiente de los periféricos utilizados (UART, GPIO, CAN, I2C), así como una arquitectura de ejecución que permita el funcionamiento robusto y eficiente del sistema. Finalmente, se llevarán a cabo pruebas funcionales para validar la comunicación entre los módulos y la transmisión de la telemetría a través de Internet, con el objetivo de demostrar la viabilidad del diseño propuesto.

2 Metodología

La planificación adecuada, junto con una definición clara de los requerimientos, alcances y limitaciones, son fundamentales para el éxito del desarrollo de cualquier proyecto de ingeniería. En el presente trabajo se propone una metodología inspirada en el libro de diseño y desarrollo de productos de Karl T.Ulrich. La metodología utilizada en este proyecto consta de 5 fases, mostradas en la Figura 2.1.

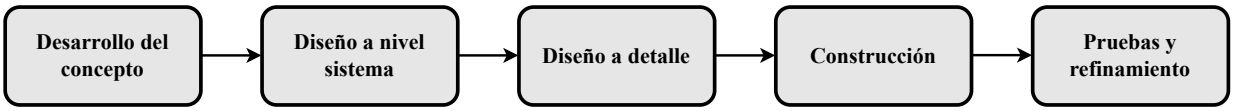


Figura 2.1: Metodología

En la fase del desarrollo del concepto se describen la función, las características y una lista clara de los requerimientos de la propuesta del proyecto con base en el estado del arte. En la fase del diseño a nivel sistema, se define la arquitectura del producto y la composición del proyecto en subsistemas y componentes.

En la fase del diseño a detalle se incluye la especificación completa de la geometría, circuitos, materiales, tolerancias del producto, documentación de control, especificaciones de piezas compradas, planes de proceso de fabricación y ensamblaje, así como las metas de desempeño y limitaciones. En la fase de construcción se fabrican prototipos de cada uno de los módulos para validar su funcionamiento y encapsular posibles errores de diseño.

Finalmente, en la fase de pruebas y refinamiento, se realizan pruebas de cada uno de los módulos y se verifica el funcionamiento individual, así como el funcionamiento del sistema integrado e instalado en la unidad vehicular. A su vez, se proponen modificaciones para mejorar el diseño y soluciones para los errores que se hayan detectado.

Parte II

Estado del arte

3 Sistemas de monitoreo, diagnóstico y geolocalización comerciales

El monitoreo de unidades vehiculares y de activos móviles es, por sí mismo, una industria con grandes empresas compitiendo en este mercado. Esto impulsa una constante evolución, donde se desarrollan avances en IoT, telemetría, inteligencia artificial y redes de comunicaciones.

Actualmente existen numerosas empresas que ofrecen soluciones que integran el Hardware (Dispositivo GPS/telemático) y la plataforma de gestión de datos que permite dar seguimiento al monitoreo de las unidades vehiculares, optimizar las operaciones logísticas, así como la reducción de costos operativos.

Las empresas que participan en este giro comercial se diferencian en tres grandes categorías:

- Proovedores de Hardware y software de la plataforma: Empresas que desarrollan tanto el dispositivo de rastreo y telemetría como la plataforma.
- Proveedores de plataforma de software: Empresas que diseñan plataformas compatibles con dispositivos GPS/-telemático genérico.
- Proovedores de Hardware GPS/telemático: Empresas que diseñan y comercializan dispositivos de rastreo y telemetría compatible con los formatos de datos más utilizados para esta aplicación.

3.1. Teletrac Navman

Empresa especializada en ofrecer soluciones de gestión de flotas de vehículos o equipos mediante GPS y datos celulares, su enfoque está en desarrollar soluciones telemáticas de última generación con Internet de las cosas, conexiones de alta velocidad transmisión de datos encriptados, hardware con los más altos estándares de seguridad y software enfocados en una experiencia de usuario intuitiva [1].

Su plataforma 'TN360' impulsada por IA es capaz de detectar irregularidades mucho antes que la capacidad humana y ayudar a las flotas a funcionar a su máximo potencial. La plataforma transforma grandes volúmenes de información en perspectivas empresariales, proporcionando visualizaciones claras de los datos sobre las mediciones más importantes para el cliente, resaltando automáticamente los patrones anómalos en los datos, tanto si el objetivo es aumentar la eficiencia del combustible y querer saber si el conductores conducen a exceso de velocidad o permanece en estado ralentí, para definir la frecuencia y los costos del mantenimiento por vehículo [1].

Hardware:

3.1.1. VT101

Dispositivo desarrollado para su uso con la plataforma TN360, es un dispositivo cableado que captura y envía información clave como la ubicación del vehículo, los kilómetros recorridos, los datos del motor del vehículo y datos de seguridad del conductor a la plataforma [2].

Figura 3.1: Dispositivo VT101 de Teletrac [2]

En la figura 3.1 se muestra uno de los dispositivos desarrollados por Teletrac equipado con una carcasa que permite una instalación sencilla, en la Tabla 3.1 se muestra un listado de las especificaciones funcionales que menciona el fabricante [2].

Tabla 3.1: Especificaciones funcionales de VT101

Conectividad	4G Global LTE CAT M1 / NBO / GPRS
Temperatura de funcionamiento	-40° a 185° F
Soportes	Alertas en directo (geocercas)
Rastreo	Datos segundo a segundo
Actualizaciones	Todas de manera inalámbrica
Tamaño	98H x 4.95W x 2.55D pulgadas
Peso	5.3 Onzas

En la Tabla 3.2 se muestran las especificaciones técnicas.

Tabla 3.2: Especificaciones técnicas de VT101

Bluetooth	BLE 5.0
Conectividad	4G Global LTE Cat M1 / NBO / GPRS
Acelerómetro	MEMS de 3 ejes
Antena	Celular interna y GNSS
Batería	LiPo de 100 mAh
GNSS	GLONASS / GALILEO / BEIDOU /GPS
Entradas / Salidas	3/3
LEDs	1
Protocolos OBDII	J1939 / ISO 9141 / KWP

3.1.2. SI201

Dispositivo desarrollado para uso con la plataforma TN360, el SI201 es un dispositivo flexible de auto instalación que captura y envía información directamente a la plataforma [3].

Figura 3.2: Dispositivo SI201 de Teletrac [3]

En la Figura 3.2 se muestra uno de los dispositivos desarrollado por Teletrac, diseñado para ser conectado directamente al puerto OBDII de los vehículos, en las Tablas 3.3 se muestran las especificaciones funcionales y técnicas [3].

Tabla 3.3: Especificaciones funcionales de SI201

Conectividad	4G Global LTE CAT M1
Instalación	Autoinstalable en el puerto OBDII
Red	AT&T
Temperatura de funcionamiento	-40 ° a 185 ° F
Soportes	alertas en directo (geocercas)
Rastreo	Datos segundo a segundo
Actualizaciones	Todas de manera inalámbrica
Tamaño	2.54 x 1.8 x 0.96 pulgadas
Peso	2.8 onzas

Tabla 3.4: Especificaciones técnicas de SI201

Bluetooth	BLE 5.0
Conectividad	G Global LTE Cat M1 / NBO / GPRS
Acelerómetro	Acelerómetro y giroscopio combinados (6-dof)
Antena	Celular interna y GNSS
Batería	LiPo de 220 mAh
GNSS	GLONASS / GALILEO / BEIDOU / GPS
Entrada/Salida	1/2
LEDs	3
Protocolos OBDII	SO 15765-4 / J1939

Plataforma TN360

En esta plataforma se gestiona el seguimiento de la ubicación en tiempo real, la captura instantánea de datos de las aplicaciones del conductor y los análisis de datos, para mantener la flota en la carretera y funcionando de manera efectiva. Cuenta con alertas en tiempo real, flujos de trabajo simplificados, registro automatizado e informes detallados para asegurar que los empleados están seguros y que la flota cumple con todos los estándares gubernamentales, se reportan las horas de manejo en tres campos; tiempo restante, tiempo restante de servicio y tiempo restante de ciclo [4].

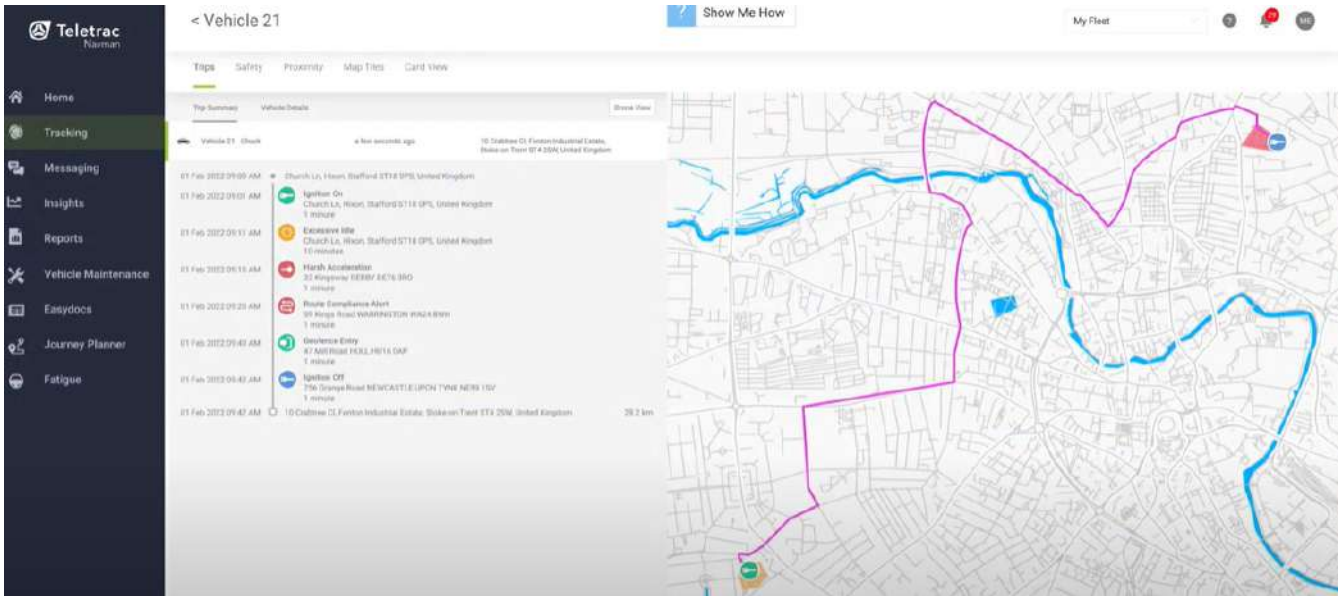


Figura 3.3: Plataforma TN360 [4]

En la figura 3.3 se muestra la ventana Tracking de la plataforma TN360, en ella se observa un mapa donde se visualiza la unidad vehicular así como la ruta asignada del vehículo 21, esta ruta es resaltada en el mapa de color morado, así como un historial los eventos que sucedieron durante el recorrido [4].

3.2. Ituran

Empresa líder en el área de tecnología de rastreo satelital, ofrece servicios como: localización, asistencia en la recuperación de vehículos robados, gestión de flotas, conectividad, movilidad, integración de plataformas, reducción de siniestros, análisis de información y una diversa variedad de accesorios que permiten tener mayor seguridad en tiempo real de personas, vehículos y mercancías [5].

Ofrecen varias versiones de su producto que contienen diferentes soluciones de acuerdo a la versión adquirida, las versiones más equipadas cuentan con localización, monitoreo, asistencia en caso de robo, servicio de call center, tracking cada 1 minuto, APP World Fleet para localizar tu unidad en el celular en tiempo real, así como consultas de reportes y recorridos históricos hasta de 4 días, al igual que envió de comandos de alertas de detenido y ralentí, algunas de las versiones son: [5].

Hardware

3.2.1. Ituran telematics

Dirigida a clientes que cuenten con flotillas vehiculares y necesiten información y diagnóstico remoto de los vehículos en tiempo real. El dispositivo de telemetría OBDII obtiene lecturas en tiempo real de gasolina, velocidad, kilometraje, RMP, temperatura, códigos de fallas DTC obtenida directamente del puerto OBDII/CANBUS [6].

Figura 3.4: Dispositivo Ituran 4G [6]

En la figura 3.4 se muestra Hardware desarrollado por Ituran para el producto Ituran telematics.

3.2.2. Ituran Tracker

Ofrece monitoreo de activos de personas, vehículos y mercancías, su versión **Car Tracker** no requiere instalación y cuenta con batería de 1 a 3 años según se configure, el monitoreo se visualiza en la APP y plataforma WORLDFLEET, con reportes de 1-3 veces al día, el equipo es aprueba de agua e intemperie [5].

En su versión **Personal Tracker:** Dispositivo pequeño y portátil, útil para un monitoreo continuo de personas, mercancías y autos, el reporte se realiza de 1 a 3 veces al día, con batería recargable y tecnología 4G.

3.3. Kosmos GPS

Empresa mexicana de rastreo satelital y telemetría con reconocimiento nacional que brinda servicios de calidad y soluciones de vanguardia con más 23 años de experiencia. su infraestructura integra sistemas redundantes para evitar perdida de datos y garantizar una comunicación continua ante cualquier eventualidad, utilizan dispositivos de rastreo y telemetría de alta tecnología como soluciones de geolocalización vehicular, seguridad y control de recursos logísticos [7].

Kosmos GPS desarrollo su propia plataforma tecnológica de última generación llamada DRACO, que permite garantizar un servicio continuo sin interrupciones, a su vez cuentan con una de las centrales de monitoreo más avanzadas del país y personal de monitoreo capacitado para brindar atención 24/7 los 365 días del año [7].

Hardware:

3.3.1. Kosmos Antares

Solución de rastreo satelital considerada como una de las herramientas más robustas en materia de seguridad satelital, una vez instalado el dispositivo en la unidad vehicular, transporte de carga u otro vehículo logístico, se podrá localizarlo en tiempo real y tomar el control vía remota ya sea desde una computadora o de un celular [8].

Figura 3.5: Dispositivo Kosmos Antares [8]

En la Figura 3.5 se muestra el hardware diseñado por Kosmos, es una de las soluciones más adaptables y seguras para implementar en: Tractocamiones, mudan ceros, rabones, camiones de carga y maquinaria pesada.

Especificaciones de Kosmos Antares

- Equipo 4G
- Detector de jammer
- Paro de motor
- Botón de pánico
- Voz en cabina (opcional)
- Bajo consumo de batería

3.3.2. Kosmos Taurus

Sistema de rastreo satelital, para seguimiento de automóviles particulares o utilitarios, también para el monitoreo de unidades de flotillas como: DiDi, Uber o Taxis. Compatible con la plataforma DRACO TELEMATICS SYSTEM para monitorear cualquier auto desde un ordenador o desde el celular, obtiene reportes de las rutas del conductor en tiempo determinado con la opción de activar un paro de motor con tan solo presionar un botón, evitando cualquier uso indebido del vehículo o robo [9].

Figura 3.6: Dispositivo Kosmos Taurus [8]

En la Figura 3.6 se muestra otro hardware diseñado por Kosmos, es una solución ideal para autos particulares, Auto Utilitario, Taxis o Flotilla de: DiDi, Uber.

Especificaciones de Kosmos Taurus

- Equipo 4G
- Fácil instalación
- Paro de motor vía remota
- Botón de pánico ante emergencias

Plataforma DRACO telematics system

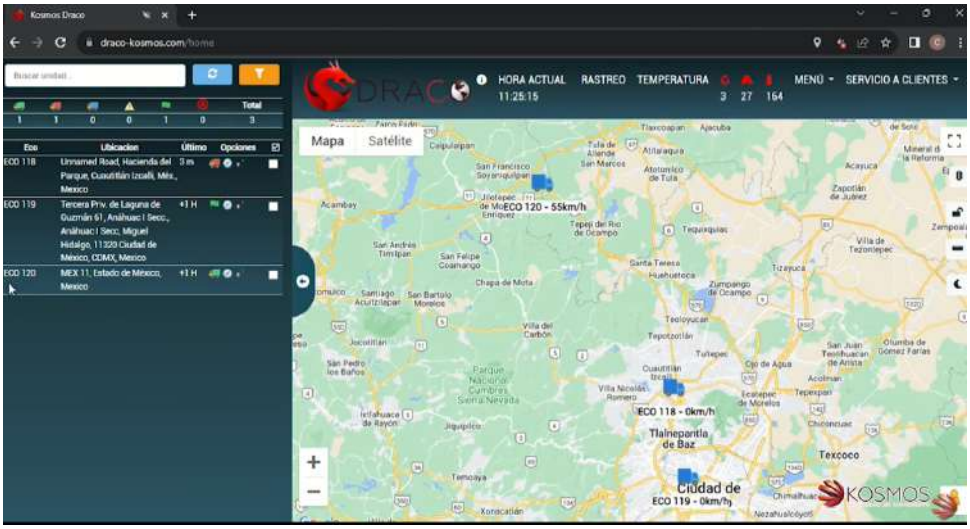


Figura 3.7: Plataforma DRACO telematics system [9]

En la Figura 3.7 se muestra la interfaz de una de las ventadas de la plataforma Draco, esta herramienta permite auxiliar en las labores logísticas diarias, optimiza considerablemente los tiempos de producción al ser una plataforma capaz de monitorear por si sola y mantener al tanto al usuario ante cualquier eventualidad a través de alarmas y correos electrónicos [10].

3.4. ORBCOMM

Empresa especializada en soluciones inteligentes de seguimiento, monitoreo y control de activos fijos y móviles en distintos sectores industriales. Sus dispositivos GPS telemáticos permiten la gestión remota de activos mediante tecnologías de conectividad satelital y celular [11].

- **Transporte:** Sus sistemas proporcionan visibilidad en tiempo real de la ubicación y estado de los activos dentro de la cadena de suministros. Están diseñados para flotas de diferentes tamaños y son compatible con la plataforma ORBCOMM.
- **Sector marítimo:** La tecnología aplicada en este sector permite el seguimiento integral de contenedores, embarcaciones comerciales y boyas oceánicas, facilitando la supervisión de flotas mercantiles y barcos pesqueros.
- **Industria pesada:** Los dispositivos están diseñados para operan en entornos extremos, brindando seguimiento y control en sectores como la construcción y minería.
- **Conectividad de IoT:** ORBCOMM suministra conectividad satelital y módulos de comunicación para el desarrollo de soluciones IoT industriales.
- **Recursos naturales:** La capacidad de recolección y transmisión de datos facilita la toma de decisiones en sectores como la agricultura, minería, petróleo y gas.
- **Gobierno:** Sus soluciones se emplean en comunicaciones de misiones críticas, que conecta las agencias gubernamentales con el personal, los vehículos, las embarcaciones y los equipos remotos.

Hardware:

3.4.1. BT 500

Ofrece seguimiento de vehículos por GPS, integración de cámaras, recopilación avanzada de datos desde CANbus, opciones de comunicación flexibles y una plataforma abierta y graduable a la que conectarse para aplicaciones de terceros y otros dispositivos. BT 500 permite a las flotas aprovechar el mayor rendimiento y la fiabilidad de las redes 4G LTE, así como HSPA y GSM, diseñado y fabricado por ORBCOMM para usarse en entornos operativos difíciles, cuenta con una carcasa robusta y duradera, que ofrece la máxima protección y la capacidad de soportar rangos de temperatura de -40 a +85 grados centígrados [12].

Figura 3.8: Hardware BT 500 desarrollado por ORBCOMM [12]

En la Figura 3.8 se muestra el dispositivo diseñado y fabricado por ORBCOMM, este dispositivo es altamente confiable y puede integrarse con cámaras de terceros, gracias a su diseño permite ser instalado de manera fácil y rápida, en la Tabla 3.5 se muestran las especificaciones del BT500 [12].

Tabla 3.5: Especificaciones del dispositivo BT500

Dimensiones	14 cm x 4 cm x 10.5cm
Peso	252 gr
Voltaje de entrada	9 VCC a 36 VCC
Potencia	3W promedio, 5W máximo
Protecciones	Descarga protegida / Gestión de potencia
Temperatura de funcionamiento	-40 °C A +85 °C
Clasificación	IP54

En la Tabla 3.5 se muestran las interfaces externas del dispositivo BT500

Tabla 3.6: Interfaces externas del dispositivo BT500

Bluetooth	2.1 +EDR, BLE 4.0
I/0 digital	1 x I/P, 3 x I/O, 5 x O/P
I/O serial	2 x RS232, OWIRE
USB	OTG, Host 2.0
GNSS	GPS, GLONASS, SBAS, QZSS, 56-ch
CAN	2 x ISO 11989-2/5
J1708	1
ISO 9141	1
WIFI	802.11 b/g/n
Movimiento	Acelerometro de 3 ejes +/-8g Giroscopio de 3 ejes +/-2000 °/S

En la Tabla 3.7 se muestra la cobertura del BT500

Tabla 3.7: Cobertura de bandas celulares del BT500

COBERTURA DE BANDAS CELULARES EN NORTEAMERICA	LTE: 2, 4, 5, 12 3G: 2,5 2G: No
COBERTURA DE BANDAS CELULARES EN EUROPA	LTE: 3,7,20 3G: No 2G: Edge E-GSM, DCS
COBERTURA DE BANDAS CELULARES EN AUSTRALIA Y NUEVA ZELANDA	LTE: 3, 8, 28 3G: 1 2G : No

3.4.2. IC 500

Captura en video eventos críticos con un testigo virtual para mejorar la seguridad del operador y ser usado como evidencia ante reclamos falsos, permite la reconstrucción de accidentes e identificar y corregir comportamientos de alto riesgo de los conductores [13].

La cámara en cabina IC 500 captura videos de incidentes como frenados bruscos, aceleraciones bruscas, vuelcos y eventos de alto impacto. Las imágenes generadas por eventos se almacenan automáticamente y se pueden revisar en

cualquier momento, el dispositivo también captura y almacena grabaciones completas de toda la actividad durante 7 días para permitir un análisis integral de las imágenes cuando sea necesario [13].

Figura 3.9: Hardware IC 500 desarrollado por ORBCOMM [13]

En la Figura 3.9 se muestra el modelo IC500, este dispositivo integrado con el BT 500 de ORBCOMM ofrecen una solución que en conjunto es más robusta, ofreciendo datos y evidencia de video de eventos que se pueden usar en litigios como prueba de lo que pasó.

En la Tabla 3.8 se muestra las especificaciones del dispositivo IC500.

Tabla 3.8: Especificaciones del IC500

Cámara	HD 720P, 120◦
DVR	30 fps, 730p
Almacenamiento	32 GB SD Card
Fuente de alimentación	12 VCC a 24 VCC
Dimensiones	4.3 x 2.72 x 2.05 pulgadas
Peso	0.5 libras
Montaje	Se monta en el parabrisas

Plataforma ORBCOMM

La plataforma ORBCOMM, permite el análisis y generación de informes de última generación única y unificada, que provee información rica en datos de sus activos para que pueda tomar decisiones más rápidas, precisas y mejor informadas.

3.5. Samsara

Empresa que ofrece soluciones avanzadas de hardware GPS y telemático diseñadas para mejorar la gestión eficiente de flotas y equipos industriales. Sus dispositivos proporcionan rastreo GPS en tiempo real, diagnóstico del motor, códigos DTC y monitoreo del comportamiento del conductor a su vez permite realizar un registro del historial de los viajes, excesos de velocidad y conducción imprudente [14].

Otras innovaciones de Samsara son sus cámaras con inteligencia artificial, diseñadas para supervisar en tiempo real el comportamiento del conducto, además de integrar sensores ambientales y de puertas que permiten monitorear condiciones como temperatura, humedad y estado de las puertas de los vehículos o almacenes. Estos dispositivos generan alertas en tiempo real para garantizar la integridad de la carga y mejorar la seguridad de los activos [14].

Hardware:

Figura 3.10: Hardware de las cámaras Samsara [14]

En la Figura 3.10 se muestran las cámaras impulsadas por IA que implanta samsara en sus soluciones. Sus cámaras inteligentes son tanto bidireccionales como frontales que se orientan tanto al conductor como a la carretea, incluso cuentan con un conector para usar 4 cámaras de alta definición de terceros para visibilidad lateral, trasera o interior.

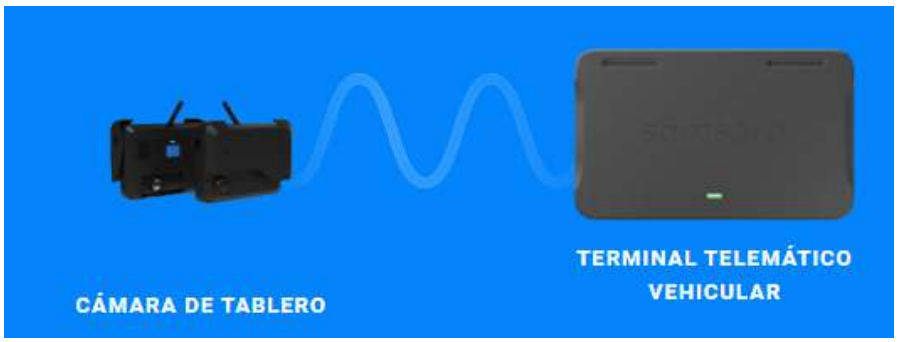


Figura 3.11: Conexión de las cámaras con el dispositivo telemático [14]

En la Figura 3.11 se muestra que la comunicación entre las cámaras y el dispositivo telemático se realiza de manera inalámbrica.

3.5.1. VG55 - Vehicle Gateway

Es un dispositivo avanzado de sensores que captura datos en tiempo real sobre el vehículo y del operador en la nube, cuenta con seguimiento GPS, diagnóstico remoto, capacidades de ELD. Proporciona a los operados información y análisis avanzado para planificar rutas, consumo de combustible, el mantenimiento y la gestión de los conductores. Esta terminal telemática es compatible con las cámaras, sensores y accesorios de Samsara [15].

Figura 3.12: Terminal telemático vehicular [15]

En la Figura 3.12 se observa el hardware desarrollado por Samsara, integra seguimiento GPS de alta precisión con actualizaciones cada segundo, obtiene datos clave del vehículo como el estado del motor, nivel de combustible, kilometraje y códigos de fallas con CAB BUS. Se conecta a la red celular 4G LTE e integra un punto de acceso Wifi para dispositivos móviles, en la Tabla 3.9 se muestran los protocolos soportados por el dispositivo [15].

Tabla 3.9: Protocolos de diagnóstico soportados

PROTOCOLOS	**VG55-NA**	**VG55-EU**
CAN - OBDII / ISO-15765	SI	SI
CAN - J-1939	SI	SI
J - 1708	SI	NO
Cable único CAN	SI	NO
CAN de alta velocidad secundario	SI	SI

En la Tabla 3.10 se muestran las especificaciones del hardware del dispositivo VG55

Tabla 3.10: especificaciones del hardware del dispositivo VG55

Material	Carcasa de policarbonato con un 50 % de contenido reciclado
Dimensiones	71mm x 117mm x 24mm
Peso	197g
Puerto USB	4 Puertos USB tipo A 2.0
Puertos de entrada / salidas auxiliares	Conector de 8 pines
Puerto de diagnóstico	Conector de 16 pines
Lineas auxiliares compatibles	3 entradas digitales, 2 entradas digitales o analógicas y 1 salida digital
Voltaje de entrada (Alimentación)	7 - 32 V
Voltaje de entrada (Auxiliar)	0 - 30 V
Rango de temperatura	-40° C - 80°C

En la Tabla 3.11 se muestra las especificaciones de conectividad celular

Tabla 3.11: Especificaciones de conectividad celular

REDES CELULARES ACCESIBLES NORTE AMERICA	AT & T y socios
REDES CELULARES ACCESIBLES EUROPA	vodafone y socios
SOPORTE DE GENERACION CELULAR	NA - 3G, 4G LTE EU - 4G LTE
COBERTURA DE BANDAS CELULARES EN NORTE AMERICA	LTE: 2, 4, 5, 12, 13 3G: 2,5 2G: No
COBERTURA DE BANDAS CELULARES EN EUROPA	LTE: 1,3,7,8,20, 28 3G: No 2G: No

En la Tabla 3.12 se muestra las especificaciones de la conectividad inalámbrica del dispositivo.

Tabla 3.12: Especificaciones de la conectividad inalámbrica

PROTOCOLOS WI-FI	802.11 a/b/g/n 2.4 GHz
PROTOCOLO DE CORTO ALCANCE	BLE 5.2
GNSS (SISTEMA GLOBAL DE NAVEGACIÓN POR SATÉLITE)	GPS L1, GLONASS G1 y Galileo E1

3.5.2. AG51 - Asset Gateway

El terminal telemático vehicular sin alimentación AG51 es un rastreador del tamaño de una cartera ideal para monitoreo de activos como contenedores intermodales, equipo de construcción, contenedores de basura, torres de luz y otros activos móviles. Cuenta con registro GPS personalizables, baterías AA reemplazables por el usuario que duran de 3 a 5 años y con carcasa resistente al agua e intemperie [16].

Figura 3.13: Dispositivo AG51 de samsara [16]

En la Figura 3.13 se muestra el hardware AG51 diseñado por Samsara, este dispositivo permite rastrear activos en tiempo real, detectar movimientos no autorizados mediante alertas de geocercas y facilitar la recuperación en caso de robo. a su vez, se sincroniza con la plataforma de Samsara y la nube, generando informes autorizados sobre la ubicación y uso de los activos. Su diseñado permite una instalación sencilla y eficiente sin necesidad de alimentación externa, a su vez en la Tabla 3.13 se listan las especificaciones generales del dispositivo AG51 [16].

Tabla 3.13: Especificaciones del dispositivo AG51

Material	Policarbonato estabilizado contra rayos UV.
Dimensiones	110 mm x 81mm x 31mm
Peso	168 g
Temperatura de operación	-40 ° C a 68 ° C
Resistencia al agua y polvo	IP67: Resistencia a la intemperie y sumergible hasta 1 metro IP69K: Máxima resistencia a lavado con alta presión y temperatura

En la Tabla 3.14 se muestran las especificaciones técnicas.

Tabla 3.14: Especificaciones técnicas del dispositivo AG51

Conectividad	LTE CAT M1 y NB-IoT
Areas de operación	Estados Unidos, Canadá, Reino Unido, México y Europa
Almacenamiento offline	Memoria Flash integrada para almacenar datos cuando no hay conexión a Internet
Seguridad	Conectividad a Internet mediante HTTPS con cifrado TLS
GPS	Sistema de posicionamiento que utiliza múltiples satélites (GPS, entre otros)
Batería	3 baterías AA reemplazables (Energizer L91 AA)
Tiempo de autonomía	3 a 5 años con 2 check-ins por día

3.6. GPS Tracker VT310

Este dispositivo es un producto de marca blanca producido por fabricantes en China y distribuido por distintas empresas bajo diferentes nombres, está enfocado para la venta a integradores de tecnología y empresas que ofrecen soluciones de rastreo GPS y Telemetría [17]. El dispositivo VT310 es un rastreador GPS/GSM/GPRS para vehículos, diseñado para monitorear en tiempo real la ubicación y estado de un vehículo mediante la red GSM y señales GPS, sus principales características son: seguimiento mediante SMS o GPRS (TCP/UDP), genera informes de ubicación actual, seguimiento por intervalos de tiempo, capacidad de almacenamiento de hasta 260,000 puntos de seguimiento, sensor de movimiento, integra un botón de pánico, control de geocercas, alerta de batería baja, alerta de acceso de velocidad, corte de motor remoto, así como alertas cuando se pierde la señal GPS y cunado el rastreador se enciende [17].

Figura 3.14: GPS Tracker VT310 [17]

En la Figura 3.14 se muestra el dispositivo de marca blanca VT310. En esta vista se puede apreciar las entradas de las antenas tanto GPS como GSM, así como leds indicadores y el botón de encendido [17].

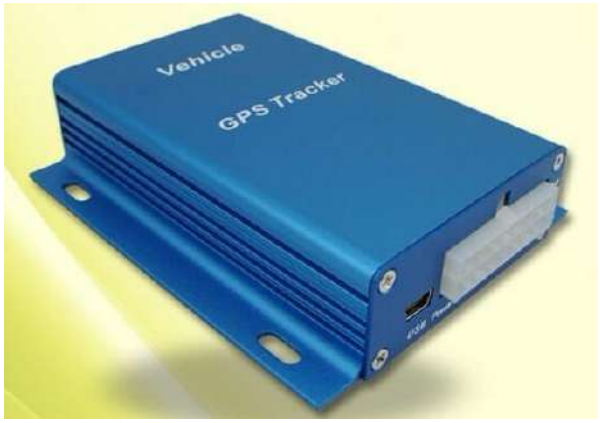


Figura 3.15: GPS Tracker VT310 vista tresera [17]

En la Figura 3.15 se observa el conector de 16 pines que utiliza para conectarse al vehículo y un conector USB ambos ubicados en la parte trasera del dispositivo, en las Tablas 3.15 y 3.16 y se muestran las especificaciones técnicas e interfaces de entrada/salida del dispositivo VT310 [17].

Tabla 3.15: Especificaciones técnicas del dispositivo VT310

Alimentación	DC9 ~35V / 1.5 A
Peso	700 g (sin accesorios adicionales)
Datum (Sistema de coordenadas)	WGS - 84
Temperatura de operación	-20 °C A +60 °C
Comunicación	Cuatribanda GSM: 850/900/1800/1900 MHz (Compatible con SMS, GPRS, TCP/UDP, CS Data)
Batería de respaldo	850 mAh
Dimensiones	104mm x 62mm x 24mm

Tabla 3.16: Interfaces de entrada y salida del dispositivo VT310

Entradas digitales	5 en total (3 con lógica negativa, 2 con lógica positiva)
Salidas digitales	5 en total
Entradas Analógicas	2 de 10 bits de resolución
Sensores compatibles	Sensor de temperatura, sensor de peso, sensor de nivel de combustible, sensor de presión del asiento (opcional)

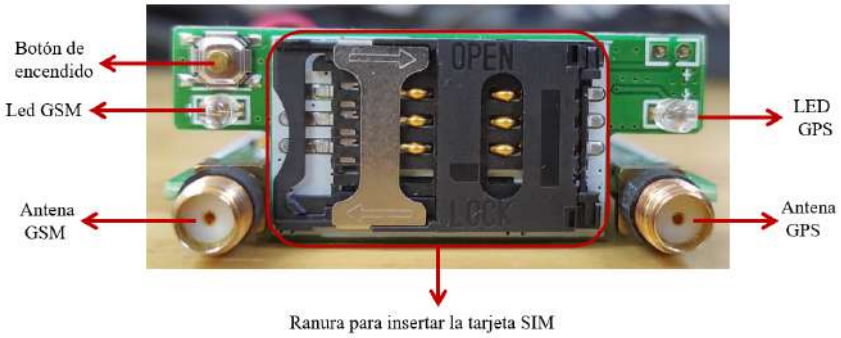


Figura 3.16: Conector SIM del VT310 [17]

En la Figura 3.16 se observa el conector SIM utilizado en este hardware, así como los leds indicadores, el botón de encendido, los conectores para las antenas GSM/GPRS y GPS.

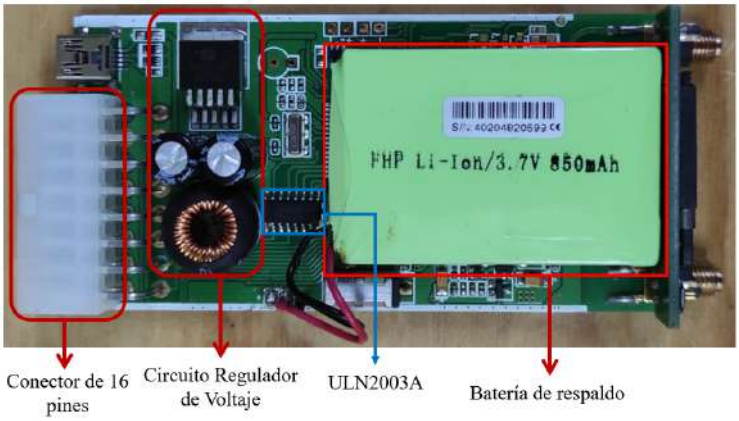


Figura 3.17: Placa PCB del hardware del dispositivo VT310 [17]

En la Figura 3.17 se resaltando el conector del dispositivo, el circuito regulador de voltaje, el integrado UNL2003A utilizado para conmutar cargas inductivas (relés), así como la batería de respaldo de 850 mAh a 3.7 V.

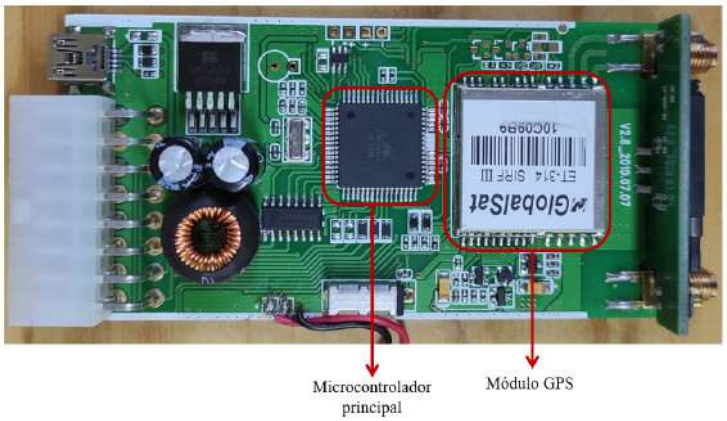


Figura 3.18: Placa PCB del hardware del dispositivo VT310 sin batería [17]

En la Figura 3.18 se muestra el hardware retirando la batería de respaldo. Se observan el microcontrolador principal ATMEGAG64A y el módulo GPS (GlobalSat ET-314 SIRF III) encargado de obtener la geolocalización y enviarla al microcontrolador principal para después ser transmitida a través de la red celular a Internet.

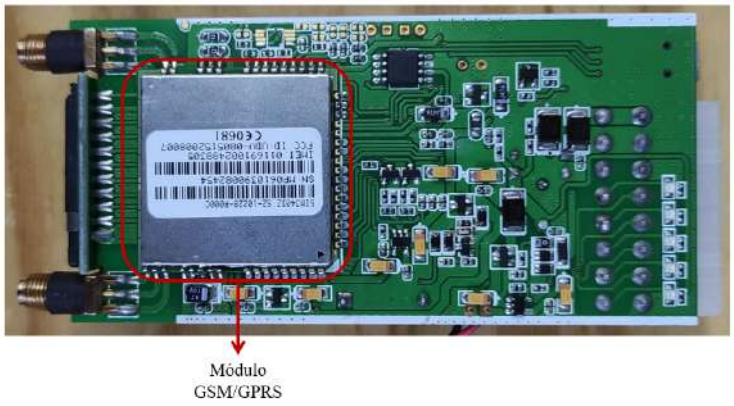


Figura 3.19: Placa PCB del hardware del dispositivo VT310 / módulo GSM/GPRS [17]

Finalmente en la Figura 3.19 se muestra el módulo GSM/GPRS (SIMCom SIM340DZ).

3.7. ELM327 OBDII

Dispositivo de diagnóstico vehicular OBDII, utilizado para leer datos y códigos de detección de errores a través del conector OBDII. Funciona para comunicarse con la computadora del vehículo utilizando una aplicación en el celular [18].

Este dispositivo no cuenta con antenas GPS ni GSM/GPRS pero si establece una comunicación directa con la compradora del vehículo, en este caso el interés del dispositivo es interpretar el hardware que se utiliza para conseguir este funcionamiento [18].

Figura 3.20: OBDII ELM327 [18]

En la Figura 3.20 se muestra la carcasa del dispositivo y el conector OBDII macho para una fácil instalación en el vehículo.

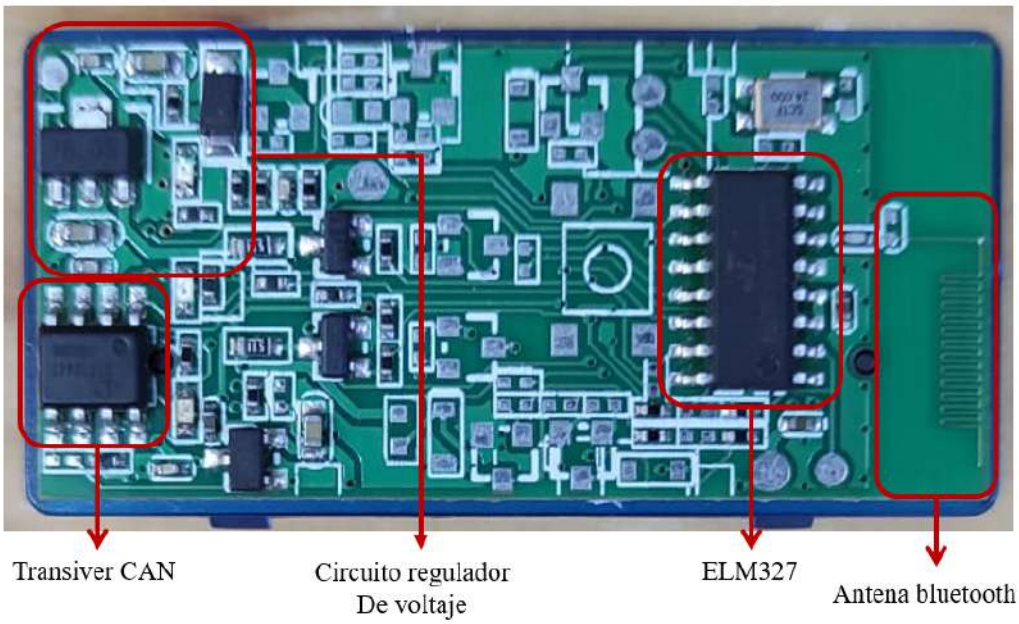


Figura 3.21: PCB del módulo OBDII ELM327 [18]

En la Figura 3.21 se muestra la PCB del módulo OBDII, se distingue el circuito regulador de voltaje, el dispositivo transductor para CAN, el integrado ELM327 y la antena Bluetooth que se utiliza para conectarse a un dispositivo Móvil.

3.8. Sistema IoT para la detección de fatiga y telemetría en vehículos de servicio mediante visión artificial y monitoreo de parámetros vehiculares

Se encontró esta investigación de tesis del año 2023, la cual aborda el diseño y desarrollo de un sistema integral de monitoreo, geolocalización y telemetría para vehículos de servicio, basado en el concepto de Internet de las Cosas (IoT). Su enfoque principal fue la detección de la fatiga de los conductores mediante cámaras y procesamiento de imágenes, utilizando algoritmos de reconocimiento facial y análisis de patrones como el parpadeo, los bostezos o la posición de la boca [19].

En la Figura 3.22 se muestra el diagrama de bloques propuesto para este trabajo de tesis.

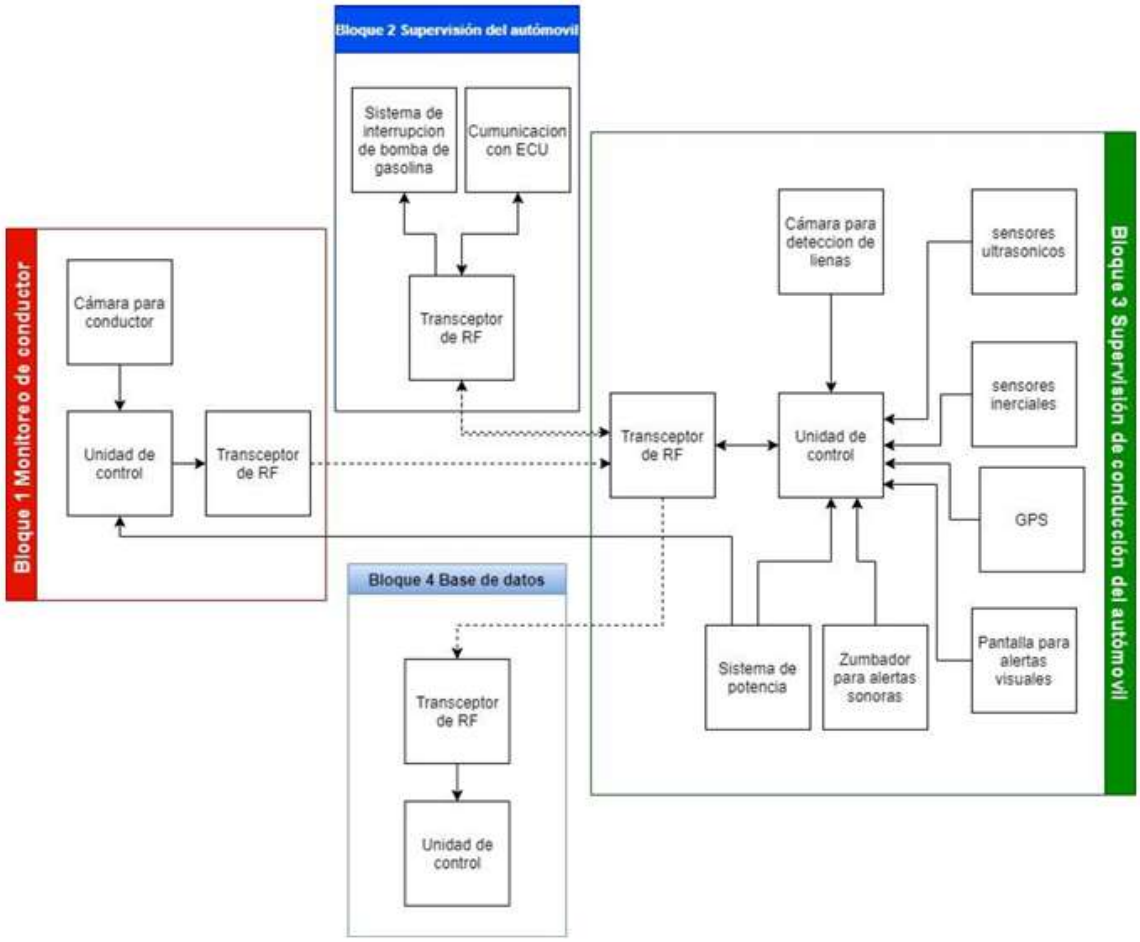


Figura 3.22: Diagrama de bloques del detección de fatiga [19].

El sistema desarrollado integra varios módulos que en conjunto permiten:

- Detectar el cansancio del conductor de forma no invasiva [19].
- Monitor de los patrones de conducción del vehículo, identificando cambios en la trayectoria o desvío del carril [19].
- Monitoreo de parámetros del vehículo como nivel de combustible, velocidad, geolocalización y estado de la batería [19].
- Genera aletas sonoras y visuales en caso de detectar cansancio, somnolencia y es capaz de activar un corte de combustible para prevenir accidentes [19].

- Registra datos en una base de datos para su análisis y seguimiento [19].

Los componentes y hardware utilizados en este trabajo corresponden a una integración de hardware propio, diseñado en EAGLE, junto con módulos comerciales. En la Tabla 3.17 se presentan los principales componentes empleados y su función dentro de esta propuesta.

Tabla 3.17: Componentes utilizados [19].

Categoría	Componente	Función
Microcontrolador	ESP32	Comunicación Wifi y Bluetooth
Módulo de Comunicación	ESP8266	Recolección de telemetría y transmisión de datos inalámbrica
Unidad de Procesamiento	Raspberry Pi 4	Procesamiento de imágenes, visión artificial y control de sensores
Comunicación CAN	MCP2515 / MCP2551	Protocolo CAN y conexión a la ECU del vehículo
Visión Artificial	Cámaras OV2640 / OV7670	Análisis facial y detección de líneas de la carretera
Sensores	Sensor ultrasónico HC-SR04	Medir distancia con otros vehículos
Sistema de Potencia	TP4056 / XL4015 /XL6009	Gestión de carga y regulación de voltaje
Actuadores	Transistor 2N2053 y relevador de 10 A	Sistema de corte de combustible
Alimentación	Batería Panasonic NCR-18650B	Batería de ion-litio 3.6V, 3250 mAh

3.8.1. Firmware y librerías

El proyecto hizo uso de Python como lenguaje principal de programación en la Raspberry Pi4, e integra varias librerías de terceros:

- **OpenCV:** Para la detección de rostro, ojos y boca con el algoritmo Viola-Jones, clasificadores Haar.
- **RPi.GPIO:** Para el control de hardware y actuadores desde la Raspberry.
- **Requests y Ubidots API:** Para él envió de datos a Internet para visualización y almacenamiento.

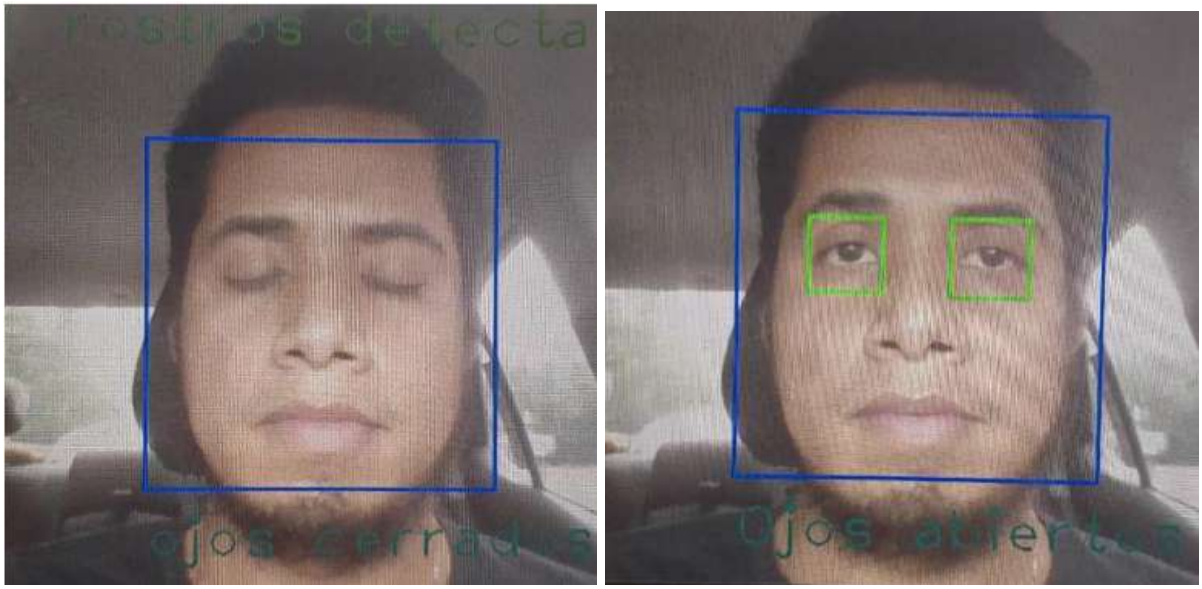
El uso de librerías de terceros facilitó el desarrollo del sistema de visión artificial, así como la gestión de hardware y la telemetría en tiempo real. Sin embargo, esto limita el desarrollo de firmware propio, ya que el sistema depende de código que no fue desarrollado completamente por el autor, sino que hace uso de herramientas y algoritmos externos.

Esto resta valor a la propuesta de diseño, al generar una dependencia de soluciones de terceros, lo cual afecta en la autonomía y originalidad del sistema desarrollado.

3.8.2. Limitaciones y Observaciones

Si bien el autor menciona la fabricación de las PCB personalizadas y el desarrollo de un prototipo funcional, tras revisar el documento no se presentan imágenes que confirmen dicha fabricación. Las imágenes incluidas corresponden únicamente a los diseños realizados en el software EAGLE, donde se muestran las placas electrónicas y sus respectivas rutas de conexión. Sin embargo, estas imágenes representan solo los esquemáticos y diagramas de las PCB, y en ningún momento se presentan fotografías reales de las placas ya fabricadas, ensambladas y en funcionamiento.

Asimismo, durante el desarrollo se identificaron problemas de cobertura y zonas muertas en la recepción de las señales inalámbricas. Además, si bien se menciona que el sistema de corte de combustible es seguro cuando el vehículo está detenido, se reconoce que para su implementación en carretera sería necesario un módulo adicional que permita realizar un frenado progresivo y seguro, evitando riesgos mayores.



(a) Detección de ojos cerrados (b) Detección de ojos abiertos

Figura 3.23: Detector de cansancio del conductor [19].

En la Figura 3.23 se muestran los resultados del monitoreo para detectar el cansancio del conductor.

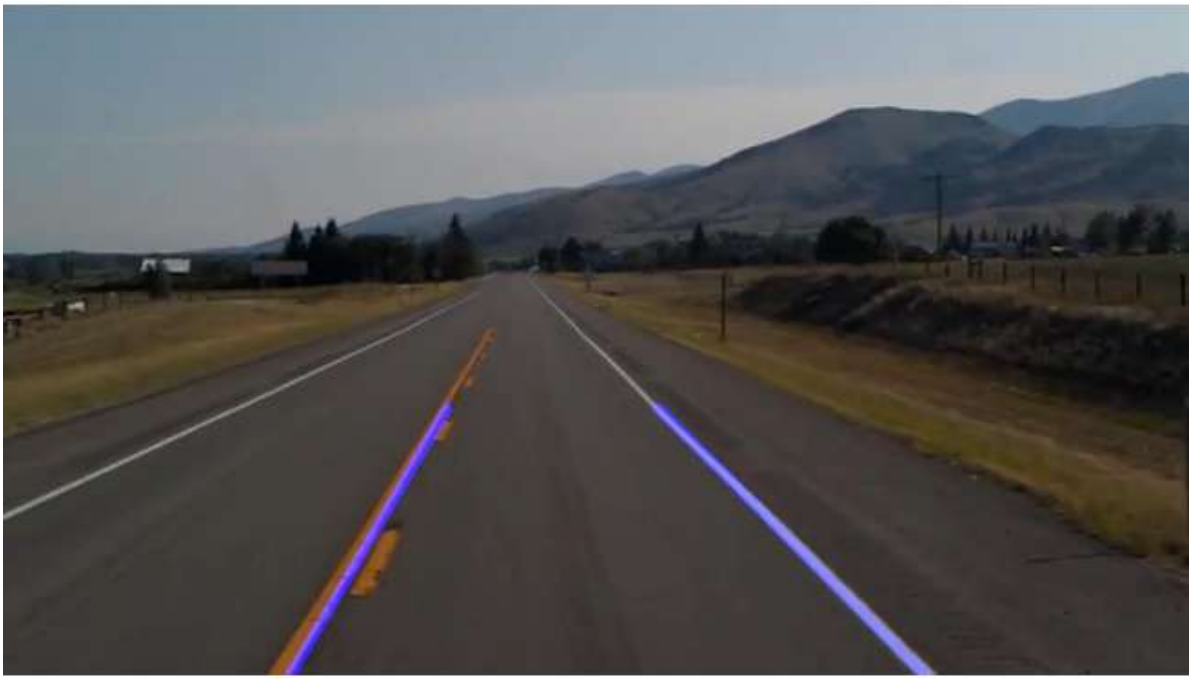


Figura 3.24: Detección de las líneas de carretera [19].

Finalmente en la Figura 3.24 se muestran los resultados de la detección de las líneas de carretera.

3.9. Conclusiones del estado del arte

La investigación de los sistemas de monitoreo, diagnóstico y geolocalización comerciales muestra una serie de dispositivos con diferentes niveles de integración y funcionalidades enfocados en la gestión de flotas, seguridad vehicular y transmisión de telemetría a una plataforma web.

Se identificaron características clave que definen los requerimientos mínimos viables para participar en este mercado como proveedor de hardware GPS/Telemático, en la Tabla 3.18 se muestra un listado de estas características.

Tabla 3.18: Resumen de características

Conectividad Móvil	Soporte GSM/GPRS/LTE para transmisión de datos en tiempo real
GNSS	Capacidad de rastreo con diferentes constelaciones de satélites: GPS / GLONASS / Galileo / BeiDou .. etc
Almacenamiento offline	Memoria Flash integrada para almacenar datos cuando no hay conexión a Internet
Interfaces de entrada y salida para interacción con el vehículo	Entradas Digitales y analógicas: Permiten integrar sensores externos, como apertura de puertas encendido del vehículo, temperatura y presión de asiento etc Salidas de control: Control remoto de motor
Almacenamiento de datos	Integran memoria flash externa que permite registrar eventos cuando la conexión celular no está disponible
Fuente de energía y batería de respaldo	Compatibles con sistemas eléctricos de vehículos (9∼40V). Baterías de respaldo integrada para asegurar el funcionamiento incluso cuando la fuente principal está desconectada
Patrones de conducción y sensores de movimiento	Analizar el comportamiento mediante sensores de movimiento y aceleración.
Adquisición de datos del vehículo mediante CAN/OBD-II	Comunicación con la computadora del vehículo a través del puerto OBD-II y bus CAN. Lectura de parámetros del motor como RPM, temperatura, velocidad, códigos de fallas.

Para diseñar una solución funcional que cumpla con las características identificadas en dispositivos GPS/Telemático para vehículos, es fundamental garantizar un hardware confiable y adecuado para condiciones del entorno automotriz. Esto implica la selección de componentes con características técnicas con certificación automotriz, así como el diseño de circuitos de alimentación y protección que aseguren un funcionamiento estable y seguro durante las pruebas funcionales.

Además, es esencial el diseño e integración del hardware adicional para implementar cada una de las funcionalidades del sistema, de esta manera se realiza un diseño modular que facilita la depuración y detección de posibles errores de diseño.

Finalmente, el desarrollo de un firmware robusto y bien estructurado es clave para garantizar un flujo de operación eficiente, con manejo adecuado de los periféricos, evitando fallos durante su ejecución y asegurando una respuesta estable en todo momento. La organización del código se basará en librerías propias, orientadas a facilitar el mantenimiento

y futuras actualizaciones.

Parte III

Marco teórico

4 Sistema Global para Comunicaciones Móviles(GSM)

El Sistema Global para Comunicaciones Móviles (GSM, por sus siglas en inglés) es un estándar desarrollado por ETSI(European Telecommunications Standards Institute) que define como se organizan y operan las redes móviles.

El GSM es una red en cuanto nos referimos a la infraestructura implementada en cada región que sigue el estándar, esta red inalámbrica está diseñada para brindar comunicaciones móviles utilizando canales de radio para la transferencia de información, tanto de voz como de datos. Para explicar los servicios móvil como está definido en el estándar es necesario explicar los subsistemas que conforman el sistema GSM [20].

4.1. Registro de Localización Domiciliaria(HLR)

Este subsistema es una base de datos encargada de la gestión de los suscriptores móviles. Una Red móvil de teléfono público (PLMN por sus siglas en inglés) puede contener uno o varios HLR dependiendo del número de suscriptores móviles, de la capacidad del equipo y de la organización de la red.

Todos los datos de suscripción se almacena en el HLR, principalmente se guardan datos relacionados con la ubicación de cada estación móvil, lo que permite enrutar llamadas a los suscriptores, además, el HLR almacena información clave asociada a cada suscriptor, esta información de muestra en la Tabla 4.1 [20].

Tabla 4.1: Datos que almacena el HLR

Dato almacenado	Descripción
IMSI	Identidad única del suscriptor en la red móvil. Se usa para autenticación
MSISDN	Número de teléfono asignado al suscriptor
Información de ubicación Número del VLR	Identifica la ubicación actual del suscriptor dentro de la red
Información sobre la suscripción	Indica si el usuario está registrado, activo o se encuentra utilizando la red
Restricciones del servicio	Restricciones de roaming o acceso a ciertos servicios

4.2. Registro de Ubicación de Visitantes (VLR)

El VLR es una base de datos temporal, dentro de la arquitectura de la red GSM, que almacena información sobre la ubicación de los suscriptores móviles que están actualmente en una determinada área de cobertura. Cuando una Estación Móvil (MS, por sus siglas en inglés) se conecta a la red celular en una zona distinta a su área de origen, inicia un procedimiento de actualización de localización y su información se registra en el VLR correspondiente a la zona donde se encuentra.

Para complementar los datos, el nuevo VLR consulta al HLR para obtener información adicional del suscriptor, y finalmente como parte de este procedimiento se liberar los recursos del anterior VLR. Gracias a este proceso, la MS puede moverse entre diferentes áreas de cobertura sin interrumpir la comunicación [20].

4.3. Centro de Comunicación de Servicios Móviles (MCS)

El MCS es una central de conmutación encarga de gestionar los recursos de radio y movilidad de los suscriptores en un zona geográfica especifica, conocida como área MCS.

Implementa las funciones de conmutación dentro de la red móvil, ya que controla las llamadas dirigidas a otros sistemas de telefonía o datos, como la Red Telefónica Pública (PSTN, por sus siglas en inglés), redes de datos públicas y privadas o incluso redes móviles no propias del operador [20].

4.4. Sistema de Estación Base (BSS)

El BSS es el encargado de gestionar las Estaciones Transceptoras Base (BTS, por sus siglas en inglés) y los controladores de estaciones base (BSC, por sus siglas en inglés). Este subsistema es responsable de controlar la activación y asignaciones de los canales de radio con base en lo establecido por el MSC [20].

- **BTS:** Componente que proporciona la interfaz de radio a los equipos móviles a través de distintos transceptores y antenas.
- **BCS:** Es un centro de conmutación de canales de alta capacidad que controla los aspectos de la asignación de frecuencias y la realización del handover, cada BSC controla varias BTS

4.5. Puerta de enlace MSC (GMSC)

En dado caso de llamadas entrantes a la PLMN, si la red fija no puede interrogar al HLR, la llamada se enruta a un GMSC y este GMCS interroga al HLR correspondiente y luego dirigirá la llamada al MSC donde se encuentra la MS [20].

4.6. Puerta de enlace SMS (SMS GMSC)

Es la interfaz entre la red móvil y la red que proporciona acceso al Centro de Servicios de Mensajes Cortos (SMSC, por sus siglas en inglés), permitiendo la entrega de mensajes cortos a las estaciones móviles [20].

4.7. Arquitectura de la red GSM

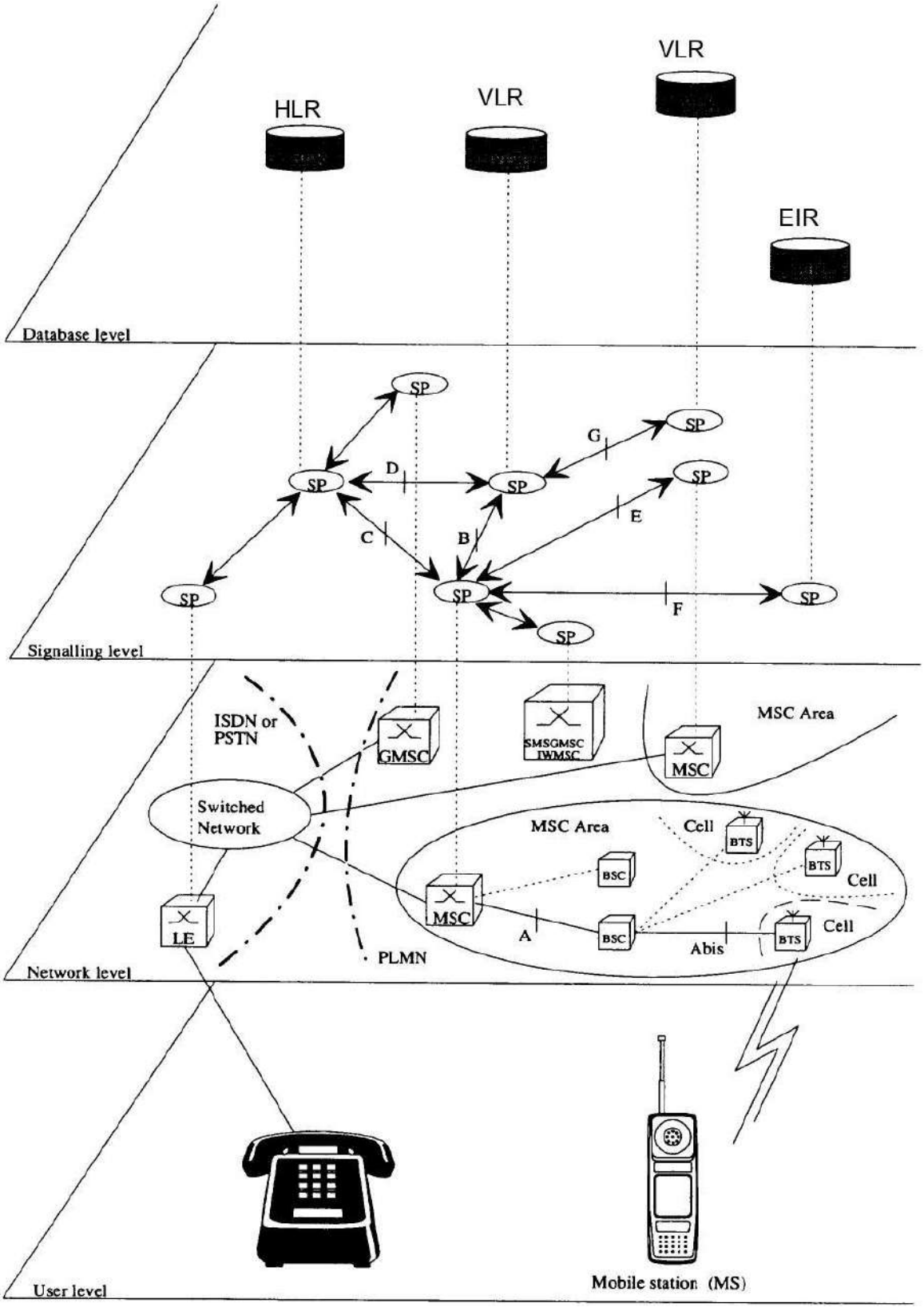


Figura 4.1: Arquitectura de la red inalámbrica GSM [20]

En la Figura 4.1 se muestra la arquitectura de la red GSM. Se divide en cuatro niveles.

- **Nivel de usuario:** En este nivel se muestran los dispositivos del usuario que se comunican con la red móvil, puede ser un teléfono móvil o teléfono fijo.
- **Nivel de red:** En este nivel se muestra la infraestructura de la red GSM, compuesta por: La Red de Telecomunicaciones Móviles Públicas (PLMN), los Centros de Conmutación Móvil (MSC), las antenas y los controladores, así como las puertas de enlace entre la red móvil y otras redes
- **Nivel de señalización:** Este nivel representa la conectividad y señalización entre los elementos de la red a través de puntos de señalización.
- **Nivel de Base de Datos:** En este nivel se muestran las bases de datos esenciales para la operación de la red GSM

En la Tabla 4.2 se muestra el flujo de la interacción entre los subsistemas del GSM.

Tabla 4.2: Flujo de iteraciones entre subsistemas

Interfaz	Origen —Destino	Descripción
A	MS — BTS/BSC	Establecimiento de una conexión desde el móvil hacia la red
B	BTS/BSC – MSC	Envió de la solicitud de una llamada o datos desde la MS
C	MSC – HLR	Consulta a la base de datos para obtener información del suscriptor
D	HLR – VLR	Transferencia de datos del usuario al VLR cuando entra a un área de cobertura
E	VLR – MSC	Confirmacion de registro del usuario y autorización para conectarse a la red
F	MSC – GMSC	Enrutamiento de llamadas hacia otra red o destino externo
G	GMSC – PSTN	Conexión de la llamada hacia la red pública
H	MSC – EIR	Verificación del dispositivo para permitir o bloquear el servicio
I	MSC – BTS/BSC	Establece la comunicación con el suscriptor móvil

Cuando una estación móvil inicia una llamada, la señal de radio llega a la BTS más cercana, que interpreta la petición y la envía al BSC, el BSC se encarga de gestionar los recursos de radios y establece comunicación con la MSC, finalmente el MSC valida la información del suscriptor consultando a las bases de datos y una vez autenticado y autorizado, establece un canal de comunicación con la estación móvil de destino [20].

5 Servicio General de Radio por Paquetes (GPRS)

A mediados de la década de 1980, las llamadas de voz eran el servicio más importante de las redes fijas e inalámbricas, por esta razón, el estándar GSM fue diseñado y optimizado inicialmente para la transmisión de voz y mensajes cortos, sin embargo, desde la década de 1990, la importancia de internet ha ido en constante aumento. El GPRS mejoró el estándar GSM para transportar datos de manera eficiente y permitió que los dispositivos inalámbricos accedieran a internet.

El Servicio General de Radio por Paquetes (GPRS) funciona de una manera muy diferente en comparación con la red GSM conmutada por circuitos, es por ello que se agregaron tres nuevos subsistemas a la red móvil GSM y se realizaron actualizaciones de software en algunos componentes existentes [21].

5.1. Unidad de Control de Paquetes (PCU)

El BSC de la GSM ha sido diseñado para conmutar canales de 16 *kbitss* con circuitos entre el MSC y los suscriptores. Como los suscriptores GPRS ya no tienen una conexión dedicada a la red GSM, por lo tanto, el BSC y su matriz de conmutación no son adecuados para gestionar el tráfico GPRS conmutado por paquetes, por lo tanto esta tarea la realiza el nuevo subsistema PCU [21].

5.2. El Nodo de Soporte GPRS de servicios (SGSN)

Se puede considerar como la contraparte conmutada por paquetes del MSC en la red GSM conmutada por circuitos. Es responsable de la gestión del plano del usuario y de la gestión de señalización.

Todas las tramas llegan al SGSN para un suscriptor son reenviadas al PCU responsable de la zona actual del suscriptor, de manera inversa, el PCU entrega las tramas de datos de un suscriptor al SGSN, que a su vez las reenviará al siguiente nodo de la red denominado Nodo de Soporte de Puerta de Enlace GPRS (GGSN, por sus siglas en inglés). [21]

5.3. Nodo de Soporte GPRS de Puerta de Enlace (GGSN)

Aunque el SGSN enruta los paquetes de datos del usuario entre la red de acceso radioeléctrico y la red central, el GGSN conecta la red GPRS a la red externa de datos. En la mayoría de los casos, la red externa será Internet. Para aplicaciones empresariales, el GGSN también puede ser la puerta de enlace a una intranet corporativa.

El GGSN también está involucrado en la configuración de un contexto PDP, de hecho, es el GGSN el responsable de asignar una dirección IP dinámica o estática al usuario. El usuario conserva esta dirección IP mientras se establece el contexto PDP [21].

5.4. Arquitectura de la red GPRS

En la Figura 5.1 se muestra cómo se integra GPRS a la red GSM para permitir que dispositivos móviles se comuniquen con servidores en internet, la principal diferencia entre ellos, es que mientras GSM establece un canal de comunicación dedicado entre ambos extremos, el GPRS divide los datos en paquetes y los envía a través de la red eligiendo la ruta más eficiente para cada paquete.

Cuando el suscripción inicia una comunicación de datos (GPRS) desde su estación móvil (MS), la señal viaja hacia la BTS, luego la BSC gestiona varias BTS y dirige la señal hacia el PCU encargado de la conmutación de paquetes

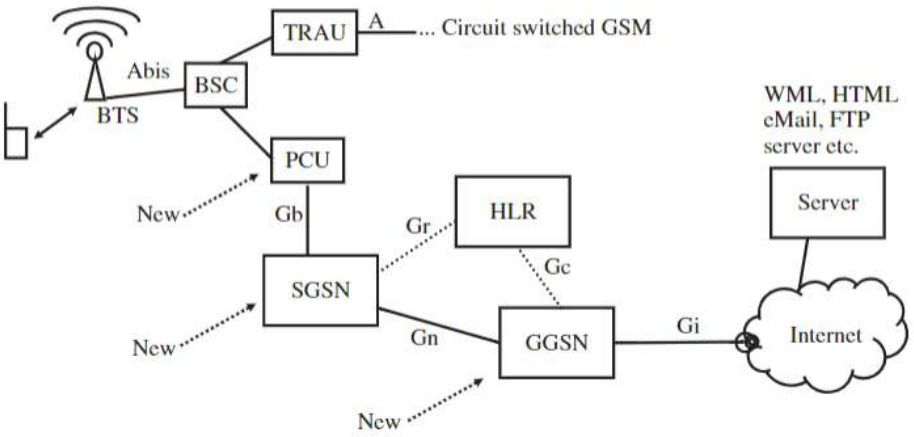


Figura 5.1: Red GPRS [21]

(GPRS) y separar el tráfico de datos del tráfico de los canales de voz y SMS (GSM) [21].

6 Sistemas de Navegación por Satélite (GNSS)

En el campo de la navegación, de la georreferenciación y del posicionamiento preciso en la Tierra, las técnicas apoyadas en satélites son actualmente las de mayor importancia, fundamentalmente en espacios abiertos. Los GNSS proporcionan un posicionamiento geoespacial y facilitan la navegación con cobertura global.

La cobertura global se logra mediante constelaciones de 24 a 27 satélites, ubicados en diferentes planos orbitales, típicamente circulares, con inclinaciones respecto al plano ecuatorial que van entre los 55° y los 65°, a una altura que oscila entre los 19,100 y los 28,000 km (orbitas MEO).

A nivel general, con aplicaciones en ciencias de la Tierra, las constelaciones de los satélites se ordenan en tres niveles, de acuerdo a las alturas sobre la Tierra: [22].

6.1. Principio básico del posicionamiento con GNSS

Conocidas las posiciones con gran precisión de los satélites en el espacio, para obtener las posición del receptor x, y, z, bastara con calcular la distancia entre el satélite y el receptor, el funcionamiento de los GNSS se basan en este principio, como se muestra en la Figura 6.1.

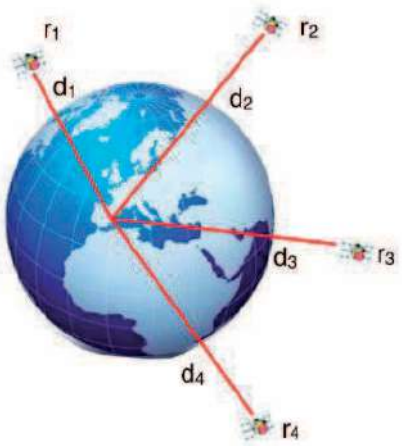


Figura 6.1: Medición GNSS [22]

Las señales de radio se propagan en el vació a la velocidad de la luz, aproximadamente de $300,000\frac{km}{s}$ y la distancia seria $d = v*t$, a este método se le conoce como pseudodistancia, para poder trabajar con esta idea se requiere sistemas de medición de tiempos muy precisos y por ello se utilizan en los satélites relojes u osciladores atómicos muy precisos, con una estabilidad de 10^{-13}, $-10^{-14}\frac{s}{dia}$ y en el caso de los receptores utilizan relojes de cuarzo [22].

6.2. Reloj u oscilador

El sistema de posicionamiento y navegación GPS se basa en la medida de la señal generada por un oscilador o reloj que genera las ondas portadores donde se monta toda la información. Un reloj atómico es un reloj que funciona activando un contador con una frecuencia de resonancia o vibración atómica, que mantiene una escala de tiempo continua y estable.

El principio de funcionamiento se basa en la transición de niveles de energía de átomos en concreto, la cual produce una frecuencia muy precisa que se utiliza para controlar por realimentación un oscilador piezoeléctrico de cuarzo [22].

6.3. Arquitectura de un sistema GNSS

En la arquitectura de los sistemas GNSS, distinguimos tres segmentos o sectores diferenciados: [22].

- **Segmento espacial:** compuesta por los satélites que forman el sistema tanto de navegación como de comunicación, así como las diferentes señales que se envían y reciben cada uno de los receptores.
- **Segmento de control:** formado por estaciones centrales de seguimiento que controlan a los satélites. Es el encargado de controlar y corregir las órbitas del segmento espacial, así como sus relojes u osciladores.
- **Segmento de usuario:** conformado por todos los equipos utilizados para la recepción de las señales emitidas por los satélites y empleados para el posicionamiento, ya sea estático o cinemático, navegación o para la determinación del tiempo con precisión.

7 Constelación NAVSTAR - Sistema GPS

7.1. Segmento Espacial GPS

El segmento está formado por una constelación mínima de 21 satélite y nominal de 24 satélites operativos que transmiten señales de radio a los usuarios. La constelación nominal de 24 satélites está ubicada en 6 planos orbitales que rodean la Tierra y el periodo orbital de los satélites es de 11h 58m, la mitad del día aproximadamente.

Cada plano orbital contiene 4 satélites en órbita MEO a una altitud de aproximadamente $20180km$ de altura sobre la superficie terrestre, cada plano orbital tiene una separación de 60° medidos con respecto ecuador con una inclinación con respecto al plano ecuatorial de 55°.

Estos satélites tienen antenas emisoras de señales de radios que trabajan en la banda L del espectro, y una antena emisora receptora para su control desde la Tierra, que trabaja en la banda S, además cada satélite está equipado con paneles solares que capturan la energía del sol y proporcionan la energía para el satélite durante su vida útil. Los paneles solares son realmente los principales indicadores de la vida útil del satélite.

Cada satélite a su vez cuenta con cuatro relojes atómicos, estos relojes tienen una precisión de menos de una millonésima parte de un segundo. El oscilador que implementan genera una frecuencia fundamental, pero a la salida un sistema multiplicador genera y transmite varios tipos de frecuencias de onda portadora L1, L2, y L5, en la Tabla 7.1 se muestra un resumen de las portadoras y códigos [23].

Tabla 7.1: Señales GPS del segmento espacial

Portadora	Frecuencia (Mhz)	Código
L1	1575.45	C/A P(Y) M
L2	1227.60	P(Y) M
L3	1176.45	I5 Q5

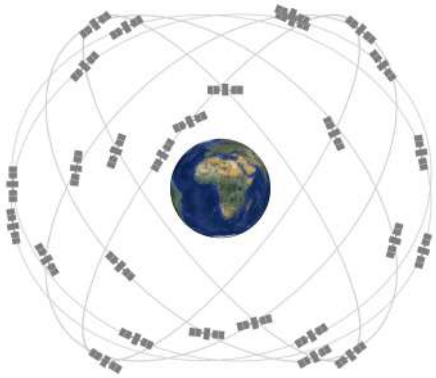


Figura 7.1: Constelación de satélites del sistema GPS [23]

En la Figura 7.1 se muestra la constelación de satélites del sistema GPS [23].

7.2. Segmento de Control GPS

El segmento de control lo forman las instalaciones en Tierra necesarias para dar soporte a la constelación GPS, sus elementos más importantes son la Estación de Control Maestra (MCS, por sus siglas en inglés), las estaciones de

monitoreo de la banda L y la antena de banda S, este segmento gestiona y vigila la señal de la banda L de navegación, actualiza los mensajes de navegación y resuelve anomalías de los satélites.

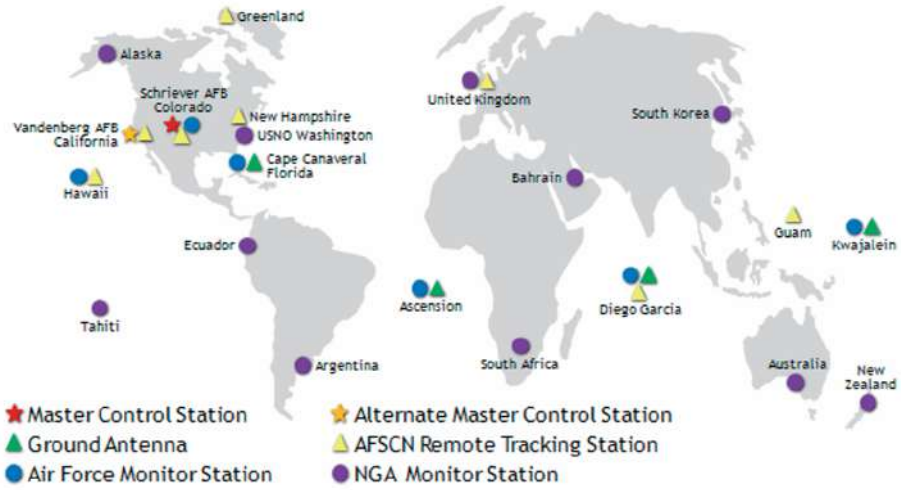


Figura 7.2: Red de control y seguimiento GPS [23]

En la Figura 7.2 se muestra el segmento de control operacional actual, que incluye una estación de control principal, una estación de control maestra, 12 antenas de mando y control y 16 sitios de monitoreo [23].

7.3. Segmento de Usuario GPS

Está constituido por todos los receptores GPS, desde el receptor más complejo hasta los chips instalados en un reloj o Smartphone cuya función principal es la recepción de señales emitidas por los satélites, las procesa para calcular su posición tridimensional y hora exacta.

El receptor obtiene las coordenadas en un sistema ECEF y procesa automáticamente todas las conversiones entre ECEF (X, Y, Z) y coordenadas latitud, longitud, altitud, o entre diversos sistemas de referencia. En la antena de los receptores se transforma la señal radioeléctrica, que capta de cada satélite, en señal eléctrica, y viceversa. En la antena se reciben las señales del satélite en línea de vista.

La índole gratuita, interrumpida y fiable del Sistema de Posicionamiento Global (GPS) ha permitido a los usuarios de todo el mundo desarrollar cientos de aplicaciones que afectan casi todas las facetas de la vida moderna en diferentes campos, así como, la agricultura, aviación, carreteras y autopistas, navegación marítima, recreación, rastreo, etc [23].

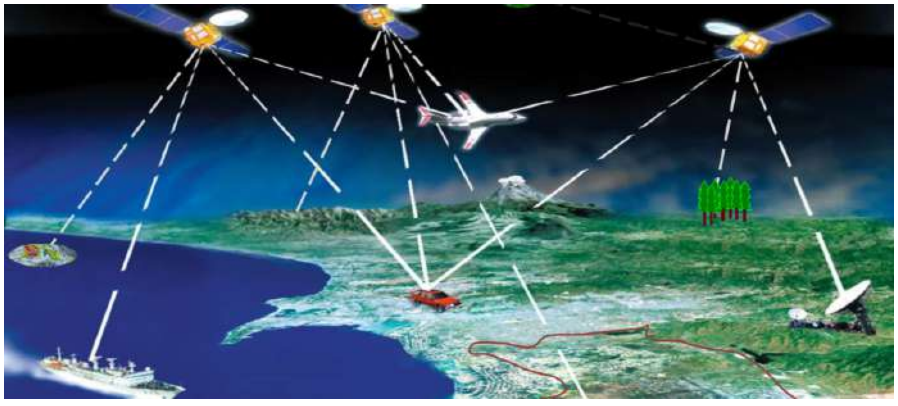


Figura 7.3: Usuarios del Sistema GPS [24]

8 Sistema de Navegación por Satélite en órbita Global - GLONASS

El Sistema de Navegación por Satélite en Órbita Global (GLONASS, por sus siglas en inglés) es muy similar a la GPS, su diferencia principal es que cada uno de los satélites emite en una frecuencia diferente, pero todos usan el mismo código, también utilizan un sistema propio sistema de referencia terrestres PZ-90, y un sistema de tiempo atómico y al igual que el sistema GPS, está formado por tres segmentos: el segmento espacial, el segmento de control y el segmento de usuario [23].

8.1. Segmento espacial

El sistema GLONASS opera con 24 satélites, pero en tres planos orbitales con 8 satélites cada uno e inclinaciones de 68,8° respecto al ecuador, a unos $19,100km$ de altitud, otra de las diferencias con el sistema GPS es que los satélites GLONASS utilizan la técnica de Frecuencia de División Múltiple(FDMA, por sus siglas en inglés), es decir, cada satélite emite una frecuencia diferente. En la siguiente tabla se pueden ver las frecuencias y los códigos para cada portadora. En la Tabla 8.1se muestra un resumen de las señales portadora, frecuencias y códigos [23].

Tabla 8.1: Señales GLONASS del segmento espacial

Portadora	**Frecuencia (MHz)**	**Código**
L1	1602 + 0.5625*n	C/A y P
L2	1246 + 0.4375*n	C/A y P
L3 en prueba y CDMA	1207.14	L30c

8.2. Segmento de control

Está conformado por un Sistema Central de Control (SCC, por sus siglas en inglés) en la región de Moscú y una Red de Estaciones de Seguimiento y Control (CTS, por sus siglas en inglés), compuesto por cinco estaciones de control y telemetría, tres equipos de navegación supervisores, un sistema sincronizador central y un sincronizador supervisor de la escala de tiempo a bordo, las estaciones supervisoras están distribuidas por toda la antigua Unión Soviética dando cobertura global.

El Segmento de control, al igual que el de GPS debe seguir y vigilar el estado de sus satélites, determinar las efemérides y errores de los relojes, así como también actualizar los datos de navegación de los satélites, estas actualizaciones se realizan dos veces al día.

9 Red de adquisición de datos en vehículos

Los vehículos modernos tienen subsistemas encargados de controlar el motor, los frenos, la dirección, el aire acondicionado, la información mostrada en el tablero, entre otros. Cada subsistema es gestionado por una Unidad de Control Electrónico (ECU, por sus siglas en inglés) con su propio sistema de adquisición de datos que consta de sensores, convertidores analógico-digitales y circuitos transceptor. Las ECU que controlan los subsistemas de los vehículos están conectadas a través de una red común donde se comparten mensajes entre sí, esta red se conforma de un bus de dos cables trenzados conocido como BUS-CAN que se distribuye a lo largo del vehículo y los subsistemas conectados a él [25].

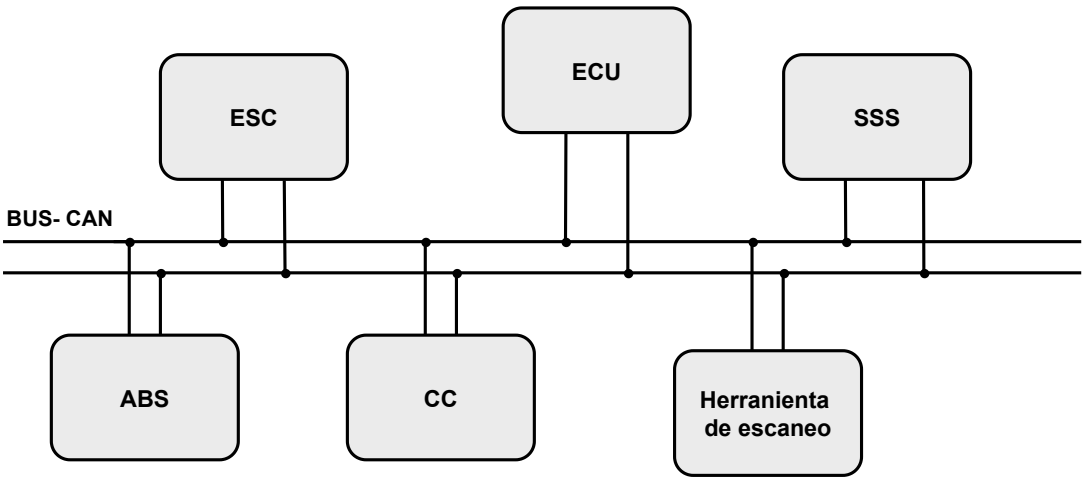


Figura 9.1: Red de adquisición de datos típica de un vehículo [26]

En la Figura 9.1 se muestran un ejemplo de cómo se conectan diferentes subsistemas de un automóvil usando el BUS-CAN, que se listan a continuación.

- Sistema Antibloqueo de Frenos (ABS, por sus siglas en inglés) [26].
- Control Electrónico de Velocidad Crucero (ESC, por sus siglas en inglés) [26].
- Control de Climatización (CC, por sus siglas en inglés) [26].
- Control Electrónico de Motor (ECU, por sus siglas en inglés) [26].
- Sistema de Dirección y Suspensión (SSS, por sus siglas en inglés) [26].

Para acceder a los datos del vehículo, se requiere de un módulo de escaneo capaz de conectarse al bus de la red, permitiendo la transmisión, recepción e interpretación de mensajes CAN. Este módulo actúa como una ECU más dentro de la red, utilizando el bus para obtener información disponible de las otras unidades de control.

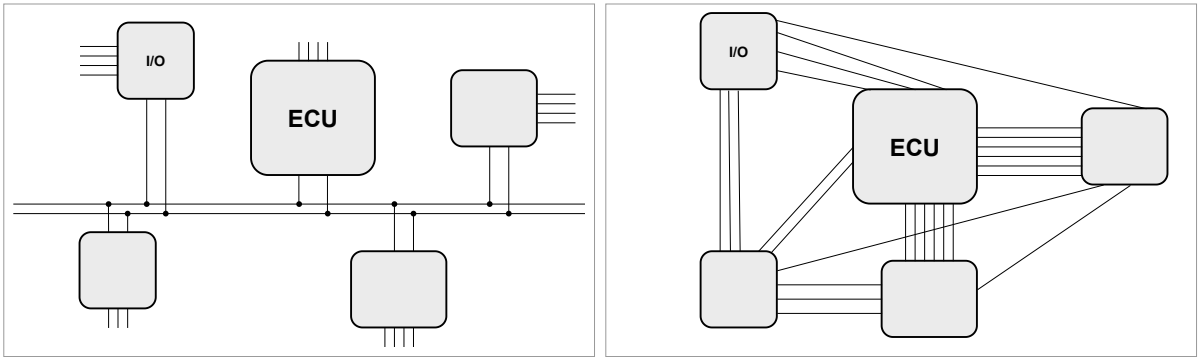
En el bus del vehículo existen dos tipos de mensajes:

- **Mensajes Normales:** Se transmiten de manera periódica sin necesidad de una solicitud y son fundamentales para el funcionamiento del vehículo, ya que permiten la comunicación entre los distintos subsistemas.
- **Mensajes solicitados:** Se envían únicamente cuando son requeridos y no son críticos para el funcionamiento normal del vehículo. Estos mensajes se emplean principalmente con fines de diagnóstico.

Todos los mensajes en el bus deben seguir la estructura definida por el protocolo de comunicación CAN y el protocolo de Diagnóstico a Bordo II (OBDII por sus siglas en inglés), para garantizar la compatibilidad de la comunicación en el sistema.

9.1. CAN en la industria automotriz

CAN es un protocolo de comunicación serial desarrollado por Bosch en 1985, originalmente considerado para ser usado como la red en la que los subsistemas de un vehículo se comunican entre sí. El avance tecnológico en la industria automotriz provocó que los fabricantes comenzaran a usar cada vez más dispositivos electrónicos en los vehículos y por lo tanto más subsistemas interactuando en esta red, el uso de esta red reemplazo el cableado punto a punto que antes se usaba en los vehículos, de esta manera, se disminuyó considerablemente la complejidad del cableado, en la Figura 9.2 se muestra una comparación del uso de la conexión punto a punto y el uso de la red CAN [27].



(a) Con CAN (b) Sin CAN

Figura 9.2: Ventajas de Red CAN [27]

La topología típica de una red CAN está conformada por un bus diferencial de dos líneas con resistencias en sus terminales, al cual se conectan nodos mediante dispositivos transceptor encargados de acondicionar las señales diferenciales a señales digitales, estas señales digitales son gestionadas por un controlador CAN, este controlador CAN puede ser un dispositivo externo o estar embebido en un microcontrolador, de tal manera que un nodo está formado por un microcontrolador, el controlador CAN y el transceptor, ver Figura 9.3

La comunicación entre nodos tiene como principio de funcionamiento la transmisión y recepción de mensajes, cada uno tiene una prioridad que depende del identificador que el controlador CAN le asigna a cada mensaje. Cualquier mensaje CAN transmitido a la red, es recibido por todos los nodos y cada uno decide si el mensaje que ha recibido lo gestiona o lo ignora, a este proceso se le conoce como filtrado de mensajes, mientras que si dos nodos transmiten un mensaje al mismo tiempo, los controladores CAN son capaces de detectar el mensaje de mayor prioridad y detener o continuar con la transmisión dependiendo de que nodo haya transmitido el mensaje con mayor prioridad, a este proceso se le conoce como arbitraje.

El uso de CAN se extendió rápidamente en la industria automotriz y, en 1993, se convirtió en un estándar internacional conocido como la ISO11898, en este estándar se describen la capa física y la capa de enlace de datos del modelo de Interconexión de Sistemas Abiertos (OSI, por sus siglas en inglés) [28].

CAN describe solo las dos primeras capas de las siete que tiene el modelo OSI. La capa física, donde se define la señalización física(PS, por sus siglas en inglés)que se encarga de la codificación de bits en señales (eléctricas o

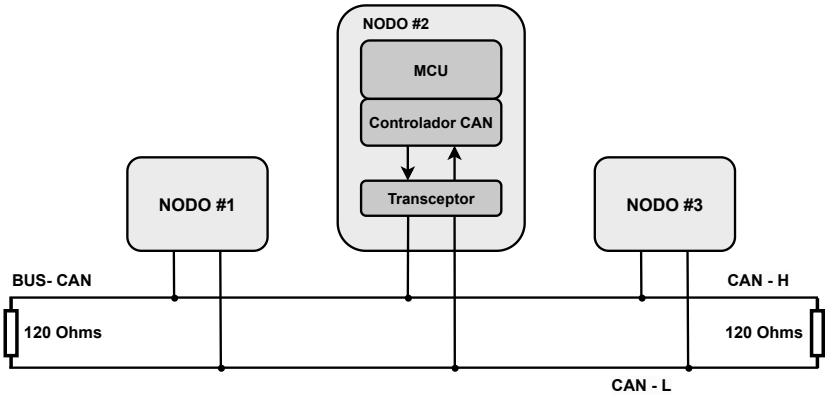


Figura 9.3: Red CAN típica [28]

electromagnéticas) con sus características físicas especificas así como de la temporización y sincronización de bits. El medio de transmisión eléctrica, los conectores y el dispositivo transceptor no son definidos por la capa física de CAN, por lo tanto, el diseñador del sistema debe elegir el circuito transceptor y medio de transmisión eléctrica, siempre y cuando se cumplan los requisitos de PS.

La capa de enlace de datos define el Control de Enlace Lógico (LCC, por sus siglas en inglés), encargado de gestionar el control, la notificación de sobrecarga, el filtrado de mensajes y funciones de recuperación, a su vez, esta capa también consta del Medio de Control de Acceso (MAC, por sus siglas en inglés), encargado de realizar la encapsulación de datos, detección y corrección de errores, y generar bits de relleno. Las versiones utilizadas hoy en día es CAN2.0 que consta de dos formatos, CAN2.0A que usa identificadores de 11 bits y CAN2.0B que usa identificadores de 29 bits [28].

En la Figura 9.4 se muestra la relación que existe entre el modelo OSI de siete capas y CAN.

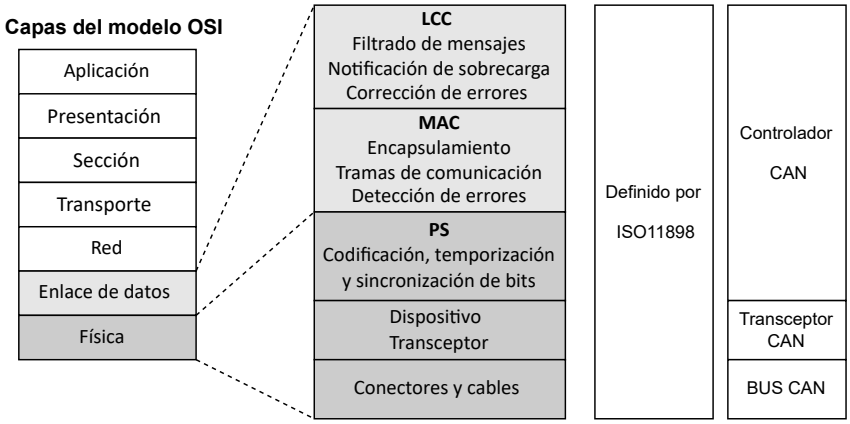


Figura 9.4: CAN y el modelo OSI [28]

9.2. Capa Física de CAN

9.2.1. Codificación de bits

La codificación de bits se realiza con el método No Retorno a Zero (NRZ, por sus siglas en inglés), con bits de relleno (bit-stuffing). Los bits se codifican en estados definidos como dominante y recesivo, se asume que el estado

dominante está asociado con el valor lógico 0 y el estado recesivo con el valor lógico 1. El estado dominante siempre prevalece sobre el recesivo en el bus CAN cuando varios nodos intentan transmitir al mismo tiempo[29].

Los bits de relleno son agregados en automático por el controlador CAN de cada nodo transmisor, después de haber transmitido cinco bits consecutivos del mismo valor, un bit de relleno toma el valor inverso al de los bits consecutivos, a su vez, los nodos receptores reconocen los bits de relleno y los eliminan antes de procesar el contenido de la trama. En la Figura 9.5 se muestra el ejemplo de cómo se agregan y eliminan los bits de relleno[29].

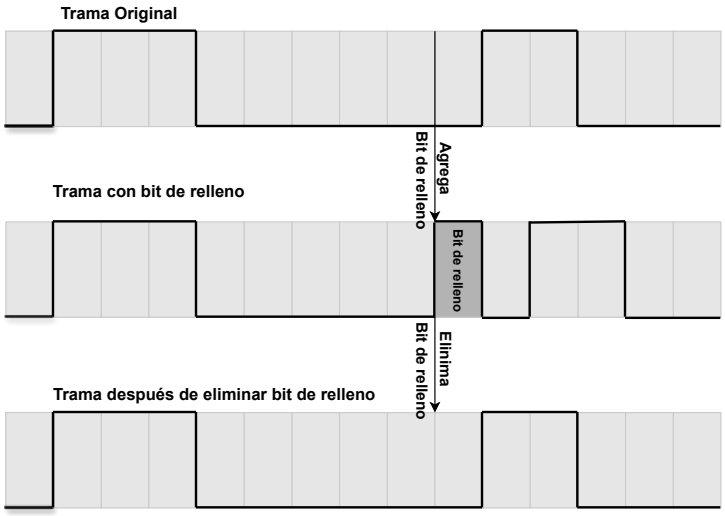


Figura 9.5: Bit de relleno [29]

9.2.2. Tiempo de bit

El tiempo de bit es el tiempo que dura la transmisión de un solo bit, cada bit transmitido tiene el mismo tiempo de bit y depende de la velocidad de transmisión definida para el bus CAN , este tiempo debe ser lo suficientemente grande para que la propagación de la señal llegue desde cualquier nodo emisor a cualquier nodo receptor y de regreso al emisor. En el tiempo de bit se tienen en cuenta retardos para compensar el tiempo que tarda la señal en propagarse a lo largo del bus [29].

La propagación de la señal en teoría depende de los dos nodos conectados al bus CAN más separados entre sí, en este caso el nodo A y el nodo B ver Figura 9.6. Cuando el nodo A transmite un bit a la red, este alcanza al nodo B después de propagarse a lo largo toda de la longitud del bus CAN desde A hasta B. En este punto el nodo B puede modificar el estado del bit de recesivo a dominante, el nodo A lee el cambio hecho por B hasta que la transición de recesivo a dominante se propaga desde el nodo B de vuelta al nodo A [29].

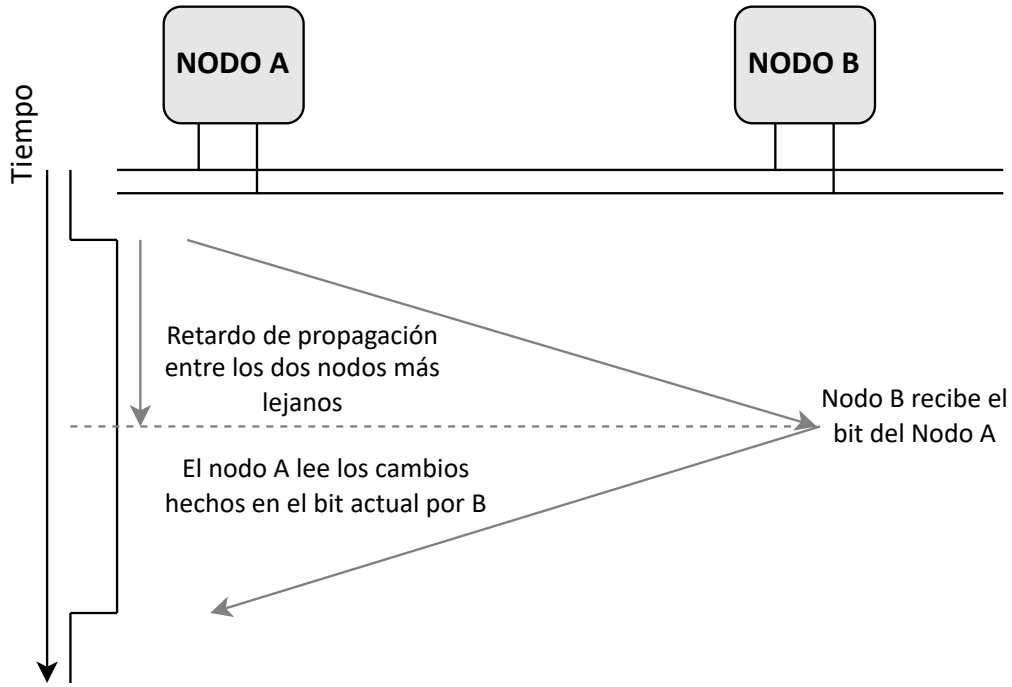


Figura 9.6: Propagación del Bit [29]

9.2.3. Sincronización de bits

Para garantizar la sincronización de bits en el bus CAN, es necesaria establecer una velocidad de transmisión común para todos los nodos conectados, esta velocidad a su vez define el tiempo de bit nominal de cada uno de los bits transmitidos. El tiempo de bit nominal es segmentado para definir el punto de muestreo, para sincronizar y para agregar retardos de propagación, en la Figura 9.7 se muestra el tiempo de bit nominal dividido en sus cuatro segmentos.

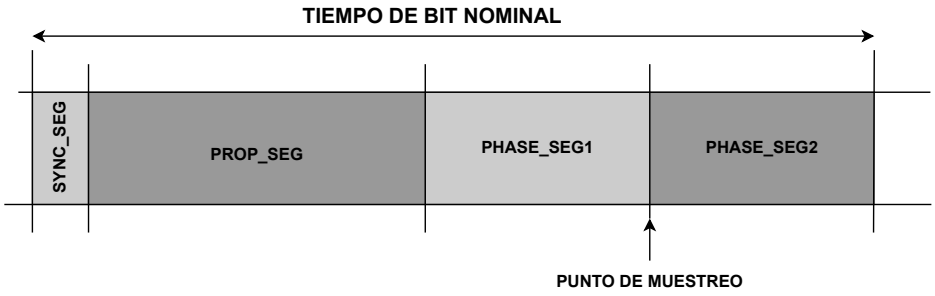


Figura 9.7: Segmentos del tiempo de bit [29]

El tiempo nominal de bit es la suma de los cuatro segmentos

$$t_{bit} = t_{SYNC\_SEG} + t_{PROP\_SEG} + t_{PHASE\_SEG1} + t_{PHASE\_SEG2}$$

El segmento de sincronización (**SYNC_SEG**) es el primer segmento en el tiempo de bit y es utilizado para sincronizar a todos los nodos del bus CAN, se espera que el primer flanco del bit se encuentre en este segmento. El

segmento de propagación (**PROP_SEG**) se utiliza para compensar el tiempo que tardan en propagarse los estados dominantes y recesivos a lo largo del bus CAN. Los segmentos de fase 1 y 2 (**PHASE_SEG1 y PHASE_SEG2**) son utilizados para compensar el error de fase en los flancos del bit, estos se pueden alargar o acortar para resincronizar la posición de **SYNC_SEG** con respecto al primer flanco del bit siguiente. El **Punto de muestreo** es el momento en el que se lee el estado del bus CAN y se interpreta el valor del bit respectivo.

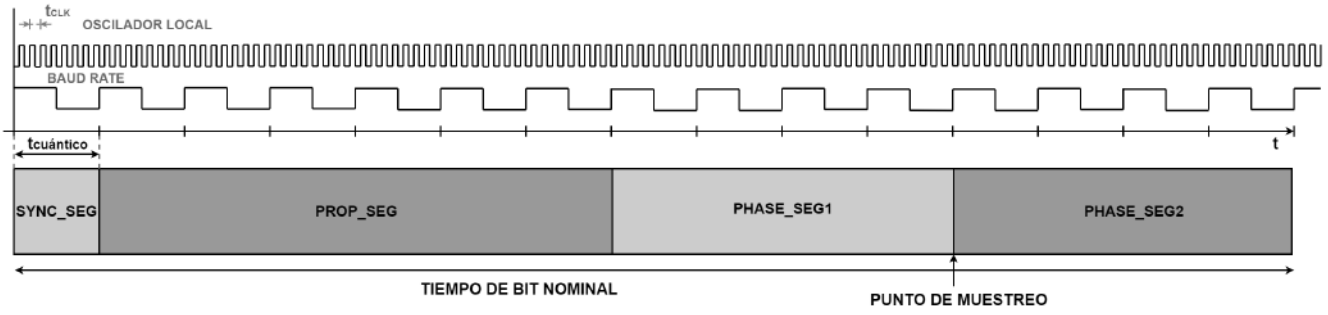


Figura 9.8: Tiempo cuántico [29]

Cada segmento es un múltiplo de una medida de tiempo denominada tiempo cuántico, que deriva del oscilador local al aplicarle un prescalador al reloj de CAN, en la Figura 9.8 se muestra al tiempo cuántico como unidad para definir la dimensión de los segmentos del CAN bit.

El ancho de cada segmento es definido de la siguiente manera: **SYNC_SEG** es igual a 1 tiempo cuántico, **PROP_SEG** y **PHASE_SEG1** están entre 1 y 8 tiempos cuánticos y el **PHASE_SEG2** tiene que ser menor o igual al valor del **PHASE_SEG1** + (tiempo de procesamiento), y el tiempo de procesamiento es menor o igual a dos tiempos cuánticos.

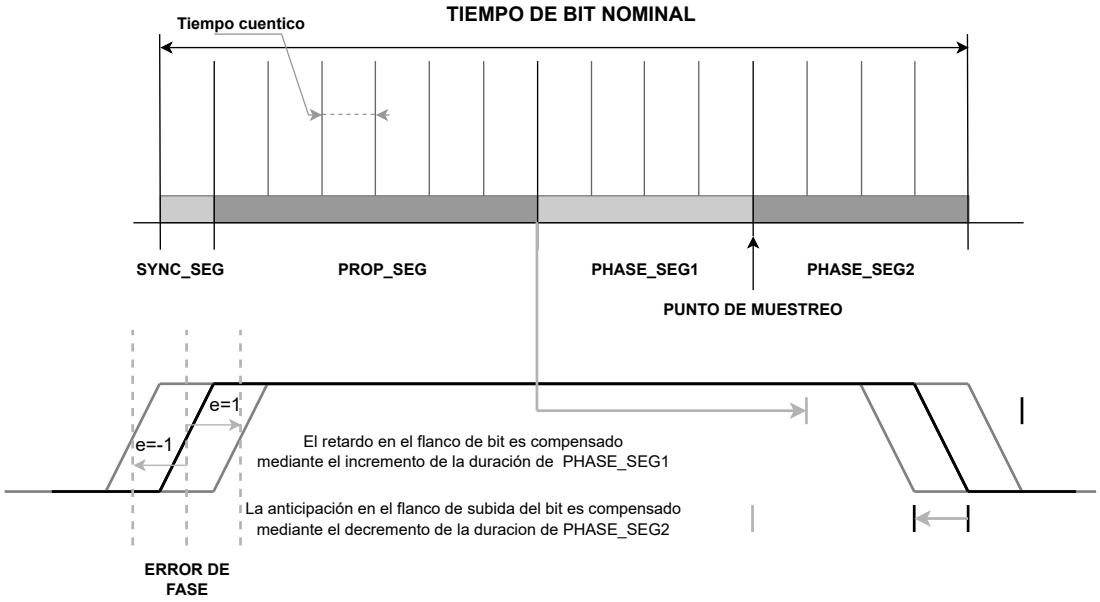


Figura 9.9: Sincronización de bit [29]

Resincronización: En la Figura 9.9 se muesra la resincronización, que se da cuando se modifican los segmentos **PHASE_SEG1** y **PHASE_SEG2** para que el siguiente bit de inicio de trama comience dentro del **SYNC_SEG** del siguiente tiempo de bit, el segmento **PHASE_SEG1** puede alargarse o el **PHASE_SEG2** puede acortarse para

compensar el error de fase.

La sincronización entre los nodos solo se puede realizar cuando el estado del bus CAN cambia debido a un cambio de bit. La resincronización durante la transmisión de una trama depende de la capacidad de detectar la transición del valor de un bit en cualquier intervalo de una trama CAN, debido a esto, se limita a que no más de 5 bits de la misma polaridad se transmitan a la vez, evitando esto mediante la adición de un bit de relleno.

9.2.4. Dispositivo transceptor

La interfaz entre el controlador CAN y el medio físico de transmisión, consiste en dispositivo que acondiciona las señales lógicas del controlador a señales analógicos llamadas CAN-H y CAN-L que en conjunto componen las señales diferenciales de los estados dominantes y recesivos que se propagan a lo largo del bus CAN.

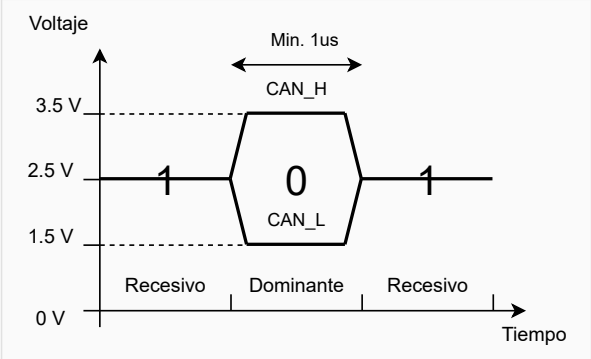


Figura 9.10: CAN BIT [30]

En la Figura 9.10 se muestra lo estados dominantes y recesivos. Durante el estado dominante, el valor digital 0 es acondicionado por el dispositivo transceptor a los valores analógicos 3.5V en la línea de CAN-H y 1.5V en la línea de CAN-L con un valor diferencial entre las dos líneas de 2V. Mientras que, en el estado recesivo, con valor digital 1, ambas líneas son acondicionadas a un valor analógico de 2.5 V, con un valor diferencial de 0V.

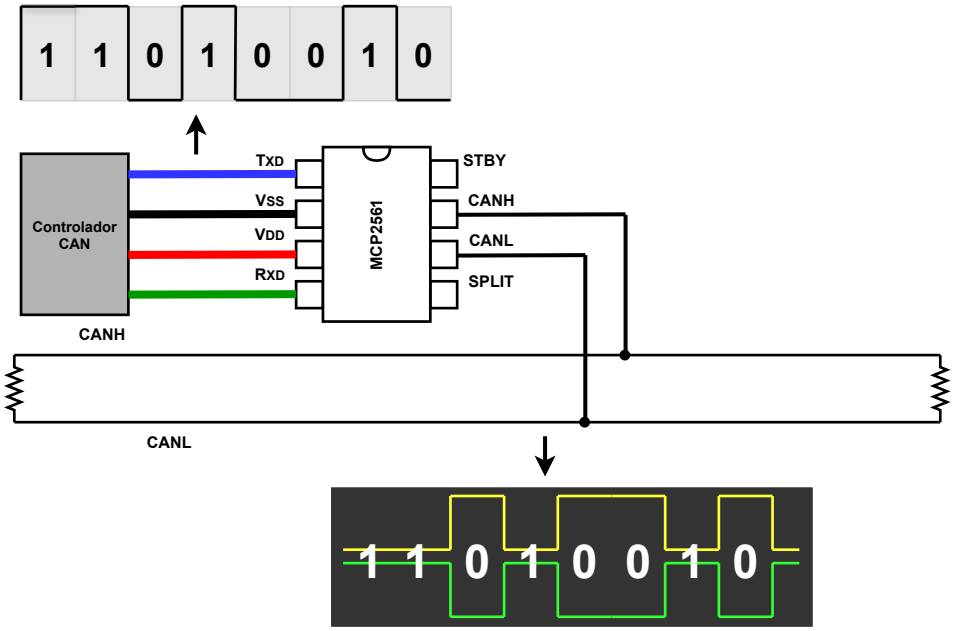


Figura 9.11: Interfaz entre controlador CAN y la capa física [30]

En la Figura 9.11 se muestra la conexión entre un controlador CAN y un dispositivo transceptor. El controlador CAN envía bits digitales (0 y 1), que luego son convertidos por el transceptor en las señales diferenciales correspondientes a CAN-H y CAN-L permitiendo la correcta transmisión de datos a través del bus CAN.

9.2.5. Cables y resistencia de terminación

El cableado utilizado en el protocolo se compone de un par de cables trenzados. Esto se debe a que el protocolo CAN utiliza bits diferenciales, es decir, en lugar de medir voltajes con referencia a Tierra, se mide la diferencia de potencial entre las dos líneas (CAN-H y CAN-L). Por lo tanto, cualquier ruido electromagnético inducido al par trenzado de cables afectará a ambas líneas de manera similar, manteniendo la diferencia de potencial constante garantizando la integridad del mensaje.

Además, para evitar reflexiones de señal en el bus, es necesario colocar resistencias de terminación de 120 ohmios en cada extremo del bus. Estas resistencias coinciden con la impedancia característica del cableado y permite disipar las señales reflejadas.

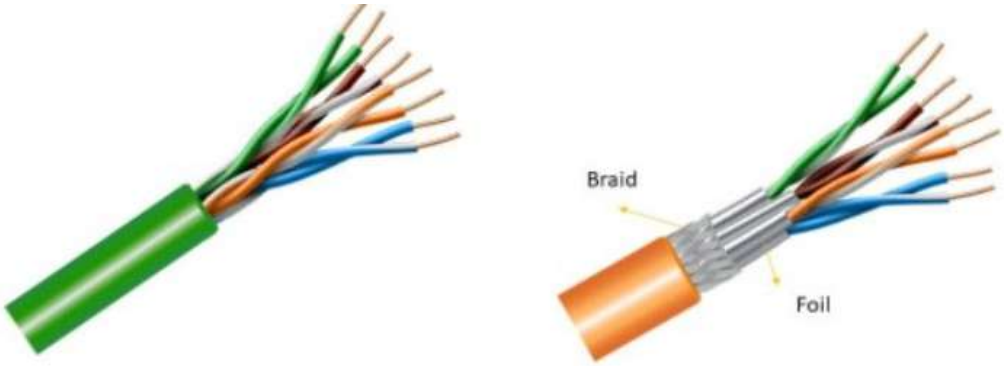


Figura 9.12: Cables de par trenzado [31]

En la Figura 9.12 se muestra el ejemplo de dos cables de par trenzado, el primer se trata de un par trenzado no apantallado que está compuesto por pares de hilos de cobre trenzados entre sí, sin ninguna protección adicional. El segundo se trata de un par trenzado que cuenta con una capa de blindaje que puede ser una malla o lamina de aluminio que protege contra interferencia electromagnética [31].

9.3. Capa de Enlace de Datos de CAN

En CAN, existen cuatro tipos de tramas, definidas de acuerdo a su contenido y función.

- **Tramas de Datos:** Contiene información de datos desde un transmisor hacia uno o varios receptores
- **Tramas remotas:** Son usadas para solicitar la transmisión de una trama de datos a un nodo con el mismo identificador de la trama remota.
- **Tramas de error:** Se transmiten cunado un nodo de la red detecta un error.
- **Tramas de sobrecargo:** Se usan para el control de flujo.

9.3.1. Trama de Datos

La trama de datos se utiliza para transmitir información entre uno o nodo transmisor y uno o más nodos receptores, cada nodo receptor decide qué mensaje recibir basado en el contenido de la información codificada en el campo de **Identificador** de la trama, existen dos formatos diferentes para los mensajes CAN. [29]

- **Tramas estándar:** CAN 2.0A
- **Tramas extendidas:** CAN 2.0B

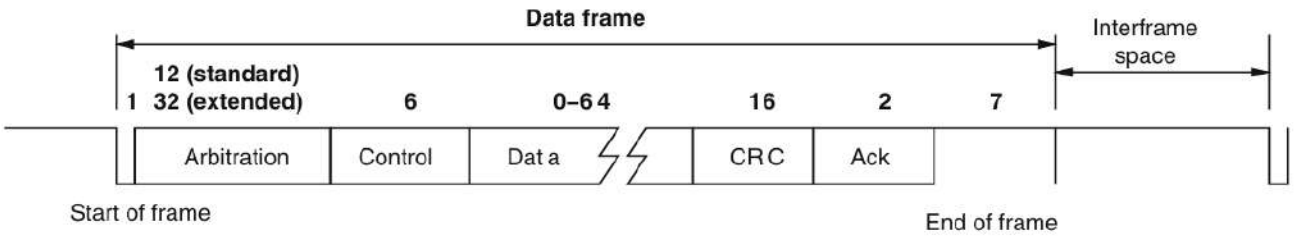


Figura 9.13: Trama de Datos [29]

En la Figura 9.13 se muestra el formato de la trama de datos, esta comienza con un bit dominante, que interrumpe el estado recesivo del bus CAN, después el campo del identificador define tanto la prioridad del mensaje en el arbitraje como el contenido del mensaje. Los otros campos son: el campo de control, que contine información sobre la longitud del mensaje; el campo de datos, que contiene la información real a trasmitir con un máximo de 8 bytes; checksum (CRC, por sus siglas en inglés), usado para verificar la integridad de los bits del mensaje; reconocimiento (ACK, por sus siglas en inglés), usado para confirmar la recepción del mensaje; fin de trama (ED), usado para definir el final del mensaje [29].

Campo Identificador

El campo de identificador consta de 11 bits en el formato estándar y 29 bits en el formato extendido, en ambos casos, el campo de identificador comienza con los 11 bits más significativos del identificador (en el formato extendido), seguido del bit de Solicitud de transmisión Remota (RTS, por sus siglas en inglés) en el formato estándar o por el bit Sustitución de Solicitud Remota (SSR, por sus siglas en inglés) en el formato extendido.

El bit RTR distingue las tramas de datos de las tramas remotas, es dominante en las tramas de datos y recesivo en las tramas remotas, mientras que el bit SRR es un marcador de posición (siempre recesivo) para garantizar la resolución entre tramas estándar y extendidas.

La trama extendida incluye un bit Extensión de Identificador (IDE, por sus siglas en inglés, siempre recesivo), seguido por los 18 bits menos significativos del identificador, y finalmente el bit RTR [29].

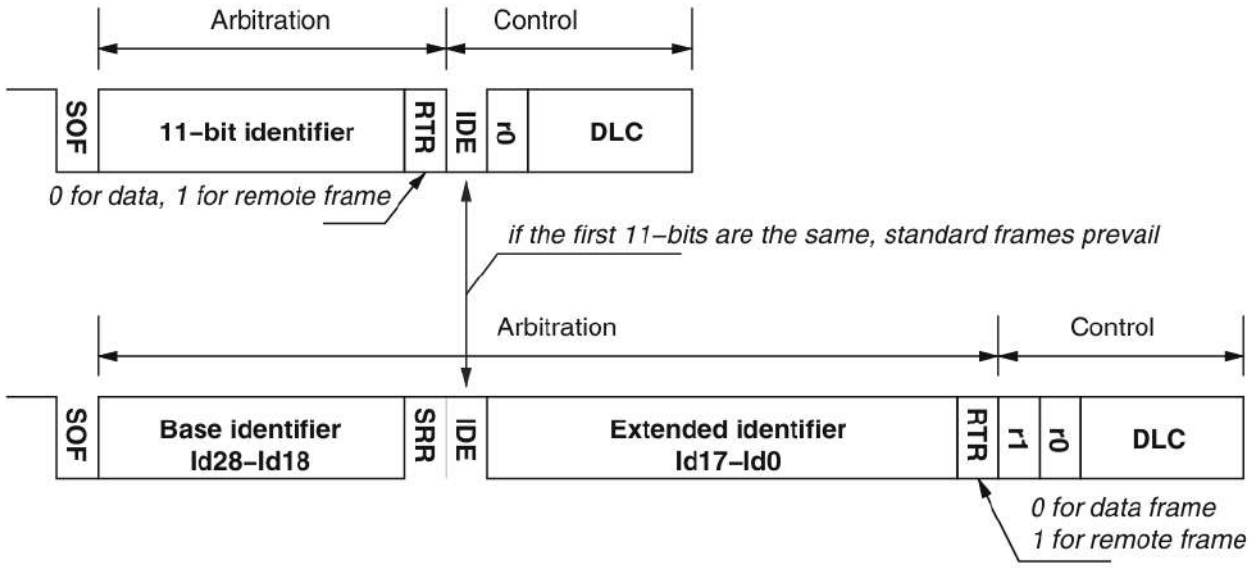


Figura 9.14: Campo identificador [29]

En la Figura 9.14 se muestra el formato de las tramas estándar de 11 bits y las tramas extendidas de 29 bits [29].

Campo de Control

El campo de control consta de 6 bits, los primeros bits están reservados y tienen un contenido predefinido, los siguientes cuatro bits en la trama de datos define la Longitud del Contenido de Datos (DLC, por sus siglas en inglés) en bytes, entonces los cuatro bits DLC representan la codificación brindar sin signo de la longitud.

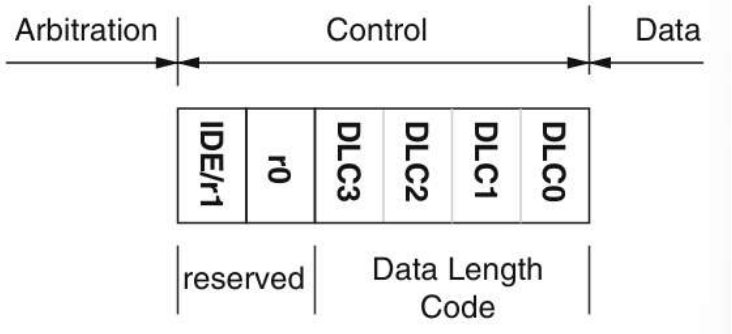


Figura 9.15: Campo de control [29]

En la Figura 9.15 se muestra el campo de control de la trama de datos [29].

Campo de CRC + ACK

Los campos CRC y ACK son los siguientes en el formato de la trama, el CRC de la trama se obtiene seleccionado el polinomio de entrada (que será dividido) como la secuencia de bits desde el bit de Inicio de Trama (SOF, por sus siglas en inglés) hasta el campo de datos (si está presente), seguido de 15 ceros.

Este polinomio se divide por el generador

$$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + X^1$$

El residuo de la división es la secuencia CRC de la trama, mientras que el campo de ACK consta de dos bits: el primer bit se utiliza para la confirmación de recibo de los receptores; el segundo bit es un delimitador en estado recesivo.

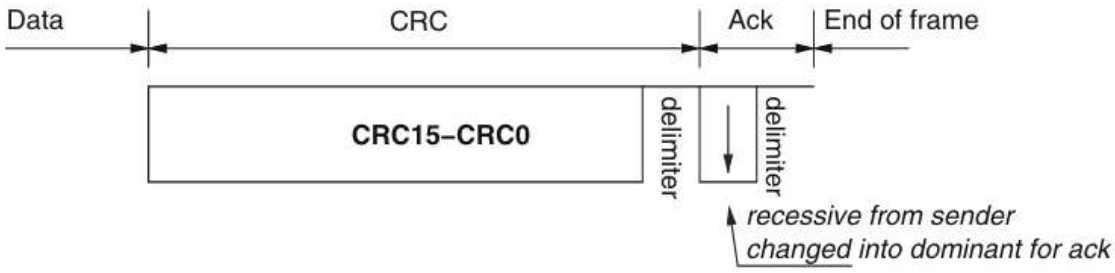


Figura 9.16: Campo de CRC + ACK [29]

En la Figura 9.16 se muestra el formato del campo del CRC + ACK [29].

9.3.2. Trama Remota

Una trama remota es usada para solicitar la transmisión de una trama de datos con un identificador especifico desde un nodo remoto. La trama remota tiene el mismo formato que una trama de datos, pero con las siguientes características: [29].

- El campo identificador se usa para indicar el identificador del mensaje solicitado.
- El campo de datos está siempre vació (0 bytes).
- El campo DLC indica la longitud de los datos del mensaje solicitado (no el transmitido)
- El bit RTR en el campo de arbitraje está siempre en estado recesivo.

9.3.3. Trama de error

La trama de error no es una trama en sí, pero es el resultado de la señalización de errores y las acciones de recuperación, esta trama, como era de esperar, tiene una forma fija compuesta por 6 bits dominantes o recesivos en secuencia y no incluye bit stuffing [29].

9.3.4. Trama de overfull

El propósito de la trama de sobrecargo es agregar un estado en el bus al final de la trasmisión de una trama para evitar el inicio de una nueva trasmisión, esta trama está compuesta por 6 bits dominantes consecutivos, su transmisión durante los primeros dos bits del espacio entre tramas, cerrando la trasmisión de la trama anterior [29].

9.4. Arbitraje

El protocolo está basado en prioridades, si un nodo desea transmitir, primero verifica que el BUS no este ocupado, esto mediante la verificación del estado recesivo del bus por más de 6 tiempos de bit CAN. Posteriormente se inicia la fase de arbitraje emitiendo el bit de inicio de trama SOF en estado dominante, en este punto, cada nodo puede comenzar la competencia por el BUS.

Todos los nodos se sincronizan con el borde del bit SOF y, se transmiten en serie los bits del identificador (que indica la prioridad del mensaje), las colisiones entre bits del identificador se resuelven de la siguiente manera:

- Si un nodo lee los bits de su identificador en el bus sin ningún cambio, se da cuenta que ha ganado la contienda por el bus y trasmite su mensaje, mientras los demás nodos pasan a modo escucha.
- Si un nodo detecta un cambio en el bit que envió, significa que hay un mensaje con mayor prioridad compitiendo por el bus, por lo que termina la transmisión y cambia a modo escucha

De acuerdo con este protocolo, el nodo con el identificador con el bit recesivo en la posición más significativa siempre gana la ronda de arbitraje y transmite su mensaje [29].

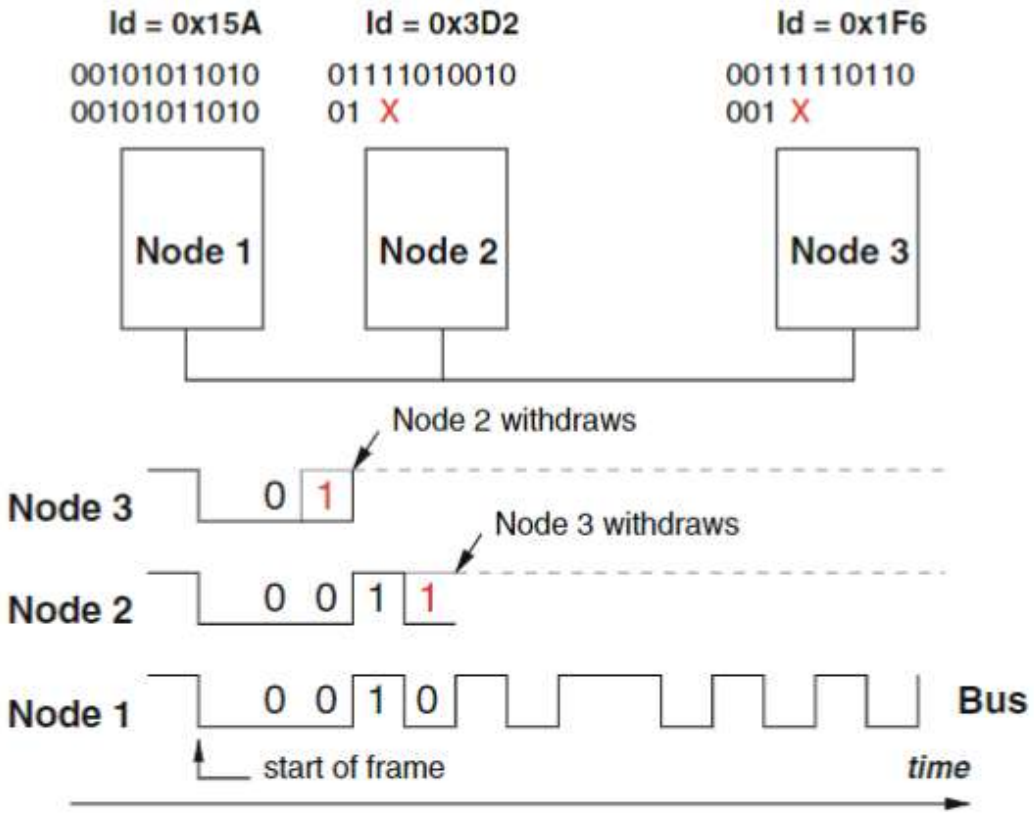


Figura 9.17: Proceso de Arbitraje [29]

En la Figura 9.17 se muestran tres nodos de la red (Nodo 1, Nodo 2 y Nodo 3) intentando enviar un mensaje con identificadores 0x15A, 0x3D2 y 0x1F6, respectivamente. La contienda comienza con el bit SFO, seguido de los bits del identificador, en el primer bit del identificador, todos los nodos envían un 0 (valor dominante), por lo que el estado del bus también es 0, como no hay diferencia con los valores transmitidos, todos los nodos continúan. En el siguiente bit, los nodos 1 y 3, envían un cero, mientras que el nodo 2 envía un 1, el estado del bus permanece en 0 (dominante) por lo que el nodo 2 se da cuenta que intento enviar un 1 y leyó un 0, por lo que se detiene la transmisión y cambia a modo escucha.

El siguiente bit, los nodos 1 y 3 envían un 1, el estado del bus es 1 y ambos continúan, mientras que en el siguiente bit, el nodo 1 envía un 0 y el nodo 3 envía un 1, como el estado resultante del bus es 0, el nodo 3 se retira de la contienda y finalmente, el nodo 1 gana el arbitraje y continúa trasmitiendo hasta el final del identificador, seguido por los campos restantes de la trama [29].

9.5. Filtrado de mensajes

Los controladores CAN tienen uno o más registros de datos que comúnmente son llamados RxObjects en los cuales se almacena el contenido de los mensajes, los nodos pueden definir uno o más filtros de mensajes asociado con cada RxObjeto y una o más máscaras de recepción para declarar los identificadores de mensajes que les interesan.

Una máscara de recepción especifica en que bits del identificador del mensaje recibido se deben operar los filtros para detectar una posible coincidencia. Un bit en 1 en el registro de máscara habilita la comparación entre los bits del filtro y los bits del identificador recibido. Un bit 0 se define como 'No importa', lo que significa que la comparación entre los bits del filtro y del identificador está deshabilitada [29].

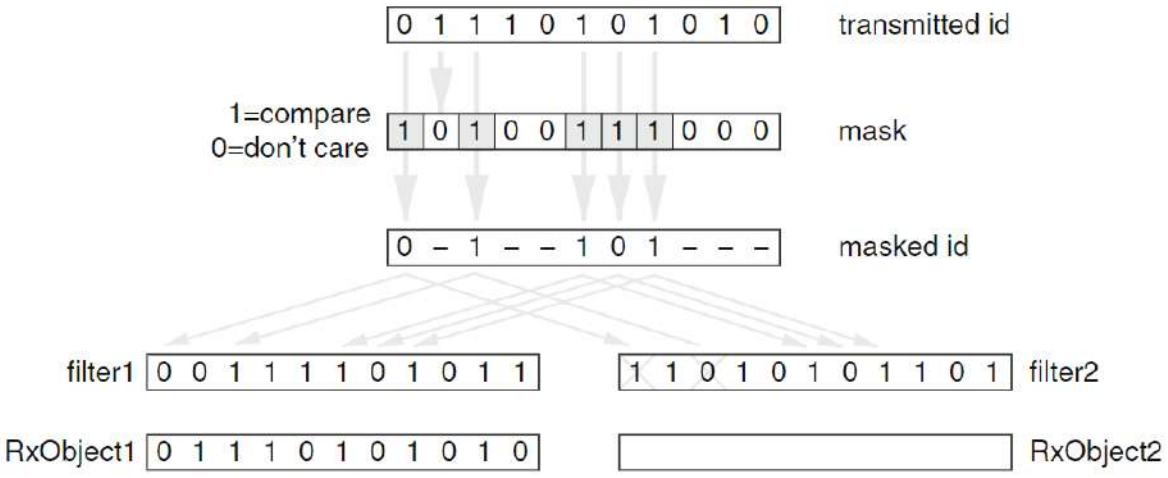


Figura 9.18: Proceso de filtrado de mensajes [29]

En la Figura 9.18 se muestra un ejemplo de filtrado de mensajes, donde el identificador transmitido en el bus es 0b01110101010 (0x3AA), dado que la configuración de la máscara, solo el primer, tercer, sexto, séptimo y octavo bit serán considerados para la comparación con los filtros de recepción, después de comparar el identificador enmascarado con los filtros de recepción, se encuentra que el filtro de RxObject coincide con los bits del ID enmascarado por lo tanto el mensaje entrante se almacena en el RxObject correspondiente [29].

10 OBD-II

El Diagnóstico a Bordo (OBD, por sus siglas en inglés), es un protocolo de capa superior (capa de aplicación) y usa como método de comunicación el protocolo CAN. En particular el estándar, especifica el conector OBDII, los protocolos de capa inferior, el formato de los mensajes, los identificadores de parámetros del vehículo que se pueden consultar como la velocidad0 RPM, temperatura, datos del motor, datos de emisiones. El estándar también define los Códigos de Diagnóstico de Problema (DTC, por sus siglas en inglés), utilizados para identificar un error y que subsistema está involucrado en el problema, los fabricantes implementan sus propios parámetros de datos personalizados OBDII específicos de cada modelo o versión del vehículo [32].

10.1. Conector OBDII [SAE J1962]

El conector OBDII de 16 pines que permite acceder a los datos del vehículo fácilmente, está especificado en el estándar SAE J1962/ISO 15031-3

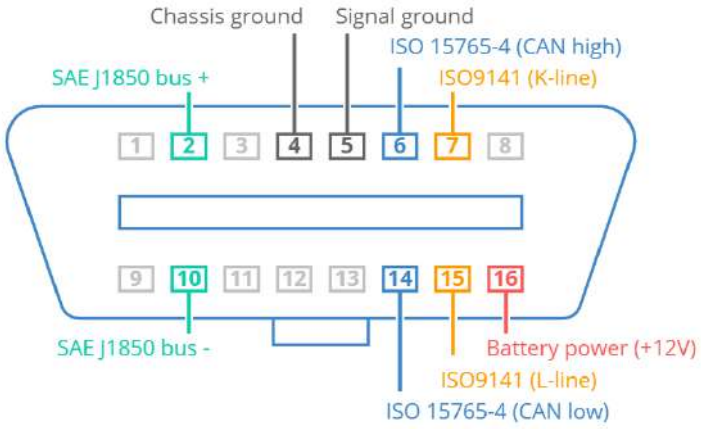


Figura 10.1: Connector OBDII [33]

En la Figura 10.1 se muestran los pines del conector OBDII, el conector se ubica por lo regular debajo del tablero y cerca del volante.

Desde 2008, el bus CAN ha sido el protocolo de capa inferior obligatorio para OBDII en todos los vehículos vendidos en Estados Unidos, según ISO 15765. La tasa de bits del bus CAN debe ser de 250K o 500K, utilizan identificadores CAN específicos para solicitudes y respuestas OBD, la longitud de los datos en el campo de datos de CAN debe ser de 8 bytes [33].

10.2. Identificadores CAN OBDII

Toda comunicación OBDII implica mensajes de solicitud y respuesta que son diferenciados por sus identificadores, los identificadores CAN OBDII pueden ser de 11 bits o de 29 bits, en la mayoría de los vehículos, se utilizan identificadores CAN de 11 bits para la comunicación, en este caso el identificador de 'direccionamiento funcional' es **0x7DF**, que corresponde a preguntar a todas las ECU compatibles con OBDII.

La ECU puede responder con identificadores de 11 bits **0x7E8 - 0X7EF**, el identificador más común es **0x7E8** [33].

10.3. Mensaje de Diagnóstico OBDII [SAE J1979, ISO 15031-5]

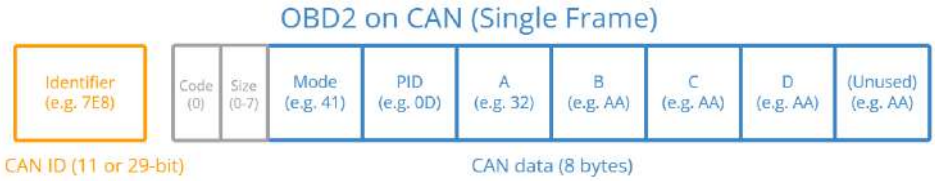


Figura 10.2: Formato de mensaje de Diagnóstico OBDII [33]

En la Figura 10.2 se muestra un mensaje OBDII, el cual se compone de un identificador, la longitud de datos. Los datos se dividen en Modo, ID de parámetro (PID, por sus siglas en inglés) y los bytes de datos.

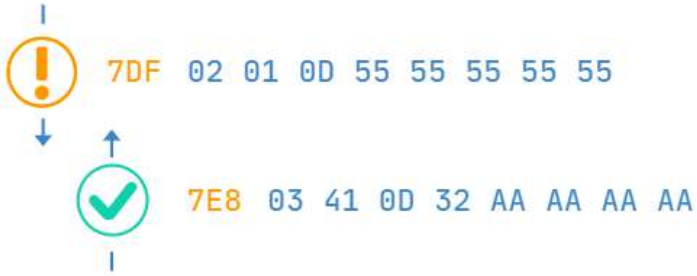


Figura 10.3: Ejemplo de solicitud y respuesta OBDII [33]

En la Figura 10.3 se muestra un ejemplo de solicitud y respuesta para el parámetro de 'Velocidad del vehículo', se define el mensaje de solicitud al automóvil con el ID CAN **0x7DF** y 2 bytes de carga útil: MODO **0x01** y PID **0x0D**. El automóvil responde a través del ID CAN **0x7E8** y tres bytes de carga útil, incluido el valor de la velocidad del vehículo en el cuarto byte, **0x32** (50 en decimal). Al consultar las reglas de decodificación para OBDII PID 0x0D determinamos que el valor físico es de $50\frac{km}{h}$ [33].

10.4. Los PID

Los parámetros PID son códigos para comunicarse con el sistema de diagnóstico a bordo de un vehículo. La Sociedad de Ingenieros Automotrices (SAE) definió códigos PID para cada modo de operación bajo el estándar J1939. El fabricante del vehículo no está obligado a implementar todos los modos de operación o códigos, y tiene la libertad de añadir sus propios. En la Tabla 10.1 se muestran los diez modos de operación que define el estándar OBD-II SAE J1979 [34].

Tabla 10.1: Modos de operación del estándar OBD-II

MODO (Hex)	Descripción
01	Muestra los parámetros disponibles
02	Muestra los datos almacenados por evento
03	Muestra los códigos DTC
04	Borra los datos almacenados, incluyendo los DTC
05	Este modo devuelve los resultados de las pruebas realizadas a los sensores de oxígeno para determinar el funcionamiento de los mismos.
06	Permite obtener los resultados de todas las pruebas de abordo
07	permite leer de la memoria de la ECU todos los DTC pendientes
08	Operaciones de control de los componentes y sistemas a bordo
09	Solicitud de información del vehículo
0A	Códigos DTC permanentes (borrados)

En la Tabla 10.2 se muestran algunos de los códigos PID más utilizados con su descripción, sus valores mínimos y máximos, sus unidades y la fórmula matemática que se le debe aplicar a los bytes de información para interpretar correctamente la respuesta, utilizando el modo 01 de operación [34].

Tabla 10.2: Comandos PID en MODO I

PID (hex)	Bytes de respuesta	Descripción	Valor mínimo	Valor máximo	Unidad	Fórmula
00	4	PIDs implementados [01-20]				Cada bit indica si los siguientes 32 PID están implementados (1) o no (0): [A7...D0] ==[PID 01...20]
04	1	Carga calculada del motor	0	100	%	A/2.55
05	1	Temperatura del líquido de enfriamiento del motor	-40	215	°C	A-40
0A	1	Presión del combustible	0	765	kPa	3A
0C	2	RPM del motor	0	16,383.75	rpm	(256A+B)/4
0D	1	Velocidad del vehículo	0	255	km/h	A
11	1	Posición del acelerador	0	100	%	A/2.55
13	1	Presencia de sensores de oxígeno (en dos bancos)				[A0...A3] == Banco 1, sensores 1-4 [A4...A7] == Banco 2
1C	1	Estándar: OBD implementado en este vehículo				Codificación de bits
1F	2	Tiempo desde que se puso en marcha el motor	0	65,535	sec	256A+B
21	2	Distancia recorrida con la luz indicadora de falla (MIL)	0	65,535	km	256A+B

11 UART

El protocolo de comunicación Receptor/Transmisor Asíncrono Universal (UART, por sus siglas en inglés) utiliza comunicación serial asíncrona con velocidad configurable y solo utiliza dos cables entre el transmisor y receptor, y un cable conectado a Tierra en ambas terminales [35].

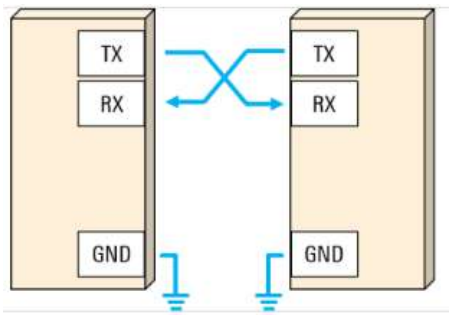


Figura 11.1: Conexión de dos terminales UART [35]

En la Figura 11.1 se muestra cómo se conectan dos terminales UART. La comunicación UART puede ser simplex (los datos solo se envían en una dirección), semidúplex (cada lado transmite, pero solo uno a la vez) o dúplex completo (ambos lados transmiten al mismo tiempo) [35].

La interfaz UART no utiliza una señal de reloj para sincronizar a los dispositivos transmisor y receptor, en lugar de una señal de reloj, se define la misma velocidad de transmisión en ambas terminales, y se utilizan los bits de protocolo de inicio y fin para la sincronización. La diferencia permitida de velocidad es baudios es de hasta 10 % antes de que la sincronización de los bits se desfase demasiado. En la Tabla 11.1 se muestra un resumen de las características de UART [36].

Tabla 11.1: Características de UART

Cables	**3 - (TX, RX, GND)**
Velocidad	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000
Método de trasmisión	Asíncrono
Número máximo de maestros	1
Número máximo de esclavos	1

11.1. Transmisión de datos

En UART el modo de transmisión es en forma de paquetes seriales, un paquete consta de un bit de inicio, una campo de datos, un bit de paridad y bits de parada [36].

11.1.1. Bit de inicio

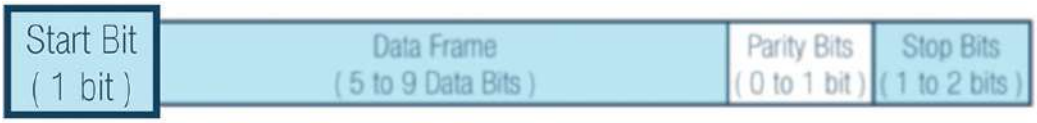


Figura 11.2: Bit de inicio [35]

En la Figura 11.2 se resalta el bit de inicio, la línea de transmisión de UART normalmente se mantiene en alto cuando no se están transmitiendo datos. Para iniciar la transferencia de datos, el UART transmisor tira la línea de trasmisión de alto a bajo durante un ciclo de reloj, en cuanto el UART receptor detecta la transición de voltaje alto a bajo, comienza a leer los bits en la trama de datos a la frecuencia de velocidad en baudios definida [36].

11.1.2. Campo de datos

Figura 11.3: Campo de datos UART [35]

En la Figura 11.3 se resalta el campo de datos de una transmisión UART, este campo contiene los datos reales que se transfieren y puede tener una longitud de 5 bits hasta 8 bits si se utiliza un bit de paridad. Si no se utiliza un bit de paridad, los datos pueden tener una longitud de 9 bits [36].

11.1.3. Paridad

El bit de paridad es la forma que tiene el UART receptor de saber si algún dato ha cambiado durante la transmisión, después de que el UART receptor lee la trama de datos, cuenta la cantidad de bits con el valor de 1 y verifica si el total de bits es un número par o impar. Si el bit de paridad es 0, indica paridad par, mientras que si el bit de paridad tiene un valor 1, indica paridad impar [36].

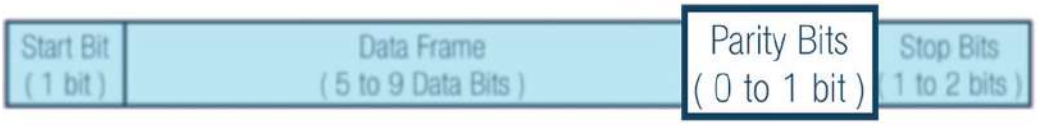


Figura 11.4: Bit de paridad de UART [36]

En la Figura 11.4 se resalta el bit de paridad de UART, cuando el bit de paridad coincide con los datos, el UART sabe que la transmisión no tuvo errores. Pero si el bit de paridad es 0 y el total es impar, o el bit de paridad es 1 y el total es par, el UART sabe que los bits en la trama de datos han cambiado [36].

11.1.4. Bits de parada

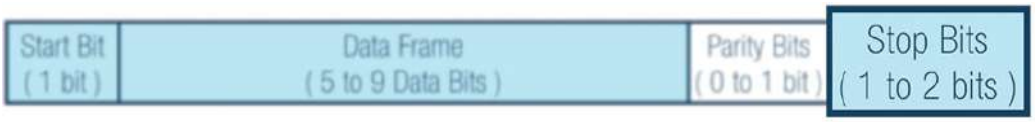


Figura 11.5: Bits de parada [35]

En la Figura 11.5 se resalta el final del paquete de datos, el UART de envió impulsa la línea de transmisión de datos de un voltaje bajo a un voltaje alto durante uno o dos bits según se defina [36].

12 Comandos AT

Los comandos AT o comandos 'ATtention' es un estándar de instrucciones universal para la comunicación entre cualquier módem celular (o RF) y su controlador host. Los comandos AT se utilizan principalmente para configurar y depurar módems, así como para habilitar la conexión de red a un operador GSM, GPRS y módems de Telefónica Móvil, así como para enviar y recibir parámetros de red y dispositivos desde el modem [37].

Los comandos AT para módulos GSM/GPRS IoT se pueden utilizar para acceder a algunos servicios de las redes móviles como: [38].

- Información del dispositivo y de la tarjeta SIM
- Servicios SMS
- Servicio de fax
- Servicios de voz y datos

Los comandos AT para módulos GNSS para acceder a algunos servicios son: [38].

- Configuración del módulo GNSS
- Información del dispositivo GNSS
- Servicio de posicionamiento
- Información de calidad de la señal

12.1. Sintaxis

Así como los lenguajes de programación, los comandos AT también tienen sintaxis que el usuario debe seguir para enviar comandos validado, la sintaxis general de comandos AT es 'AT <x><n>', donde '<x>' es el comando y '<n>' son los argumentos para ese comando, en la Tabla 12.1 se muestran los tipos de comando [39].

Tabla 12.1: Comandos AT

TIPO	Sintaxis	Descripción
Comando de prueba	AT+<x>=?	Este comando devuelve la lista de parámetros y rangos de valores establecidos por el comando de escritura o procesos internos
Comando de lectura	AT+<x>?	Devuelve el valor actualmente establecido de los parámetros
Comando de escritura	AT+<x>=<...>	Establece los valores de parámetros definidos por el usuario
Comando de ejecución	AT+<x>	Lee los parámetros no variables afectados por los procesos internos

Enseguida se muestra un ejemplo de un comando utilizado para configurar el modo SMS del módem.

- Sintaxis: AT+CMGF=<mode>
- Ejemplo: AT+CMGF=1

13 I2C

El protocolo I2C es un protocolo de comunicación serial desarrollador por Philips Semiconductors (ahora NXP Semiconductors) para comunicar diferentes dispositivos electrónicos dentro de un sistema.

Su diseño de dos hilos maximiza la eficiencia del hardware y la simplicidad del circuito, ya que todos los dispositivos conectados pueden comunicarse a través del mismo bus. Esto lo ha convertido en una solución ideal para aplicaciones en sistemas embebidos, sensores, memorias y microcontroladores [40].

13.1. Arquitectura y funcionamiento básico

El bus I2C implementa una arquitectura maestro-esclavo, en la cual el dispositivo maestro inicia la comunicación con los esclavos para enviar o recibir datos. Los esclavos no pueden iniciar la comunicación ni comunicarse entre ellos directamente. El maestro genera una señal de reloj (SCL) para sincronizar a todos los dispositivos en el bus, eliminando así la necesidad de que cada dispositivo tenga su propio reloj [40].

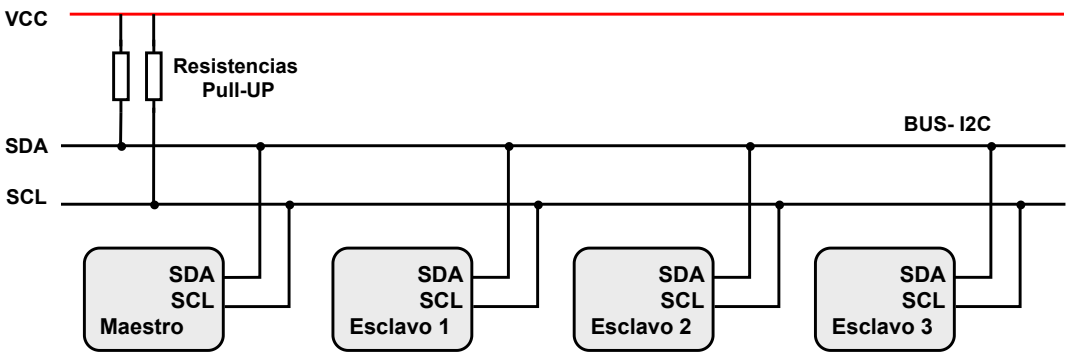


Figura 13.1: Bus I2C [40]

En la Figura 13.1 se muestra la topología de un Bus I2C, se conectan las líneas de datos (SDA) y reloj (SCL) de todos los dispositivos a las líneas correspondientes de maestro. Además, es necesario incluir resistencias de pull-up en ambas líneas para garantizar la transferencia de información en el bus debido a las componentes capacitivas y resistivas que presentan los dispositivos esclavos. Estas resistencias son esenciales para generar correctamente las condiciones de inicio (START) y fin (STOP) de la comunicación [40].

En un bus I2C, cada esclavo tiene una dirección única, que permite al dispositivo maestro comunicarse con esclavos específicos. Estas asignaciones de direcciones permiten que el maestro pueda acceder a la información de cada uno de los esclavos que están conectados al bus I2C. En la Tabla 13.1 se muestran los modos de operación que soporta el protocolo [40].

Tabla 13.1: Modos de operación de I2C

Modo de Operación	Velocidad máxima
Modo Estándar (SM)	100 kbps
Modo Rápido (FM)	400 kbps
Modo Rapido Plus (FM+)	1 Mbps
Modo de Alta Velocidad (HS)	3.4 Mbps
Modo Ultra Rápido (UFm)	5 Mbps

13.2. Estructura de la comunicación I2C

13.2.1. Condiciones de START y STOP

En I2C la transmisión de datos siempre comienza con la condición de START y finaliza con la condición de STOP.

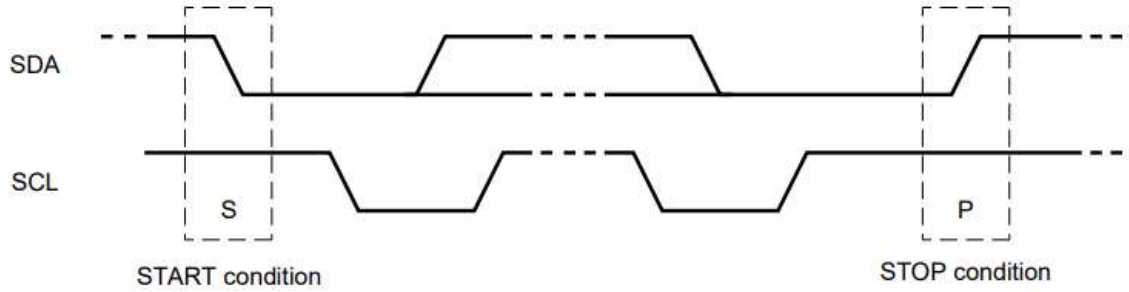


Figura 13.2: Condiciones de START y STOP [40]

En la Figura 13.2 se muestra las condiciones de START y STOP, estas condiciones son generadas por el controlador I2C [40].

La condición de START se define como la transición de ALTO a BAJO en la línea SDA mientras SCL está en alto, mientras que la condición de STOP se define como la transición de BAJO a ALTO en la línea SDA mientras SCL está en alto [40].

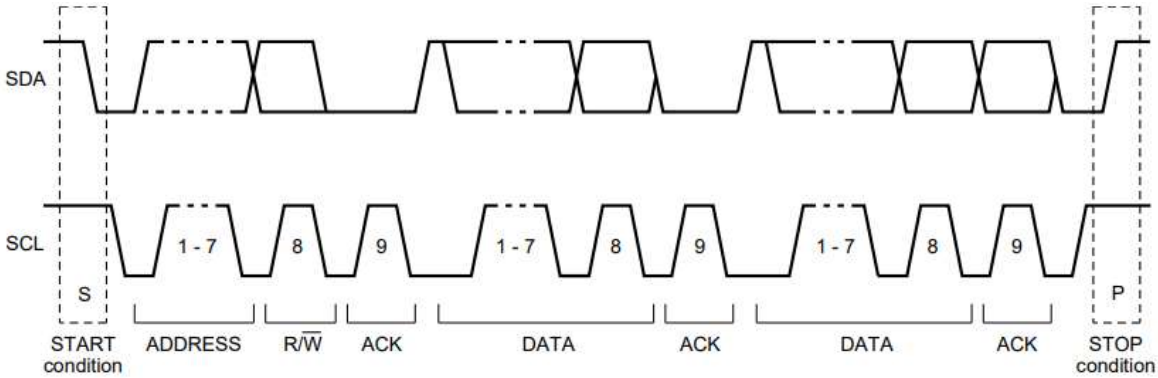


Figura 13.3: Transmisión de datos UART [40]

En la Figura 13.3se muestra una transmisión de datos de I2C, la comunicación comienza una vez generada la condición de START. Luego, se envía la dirección de esclavo del dispositivo en 7 bits, seguida de un octavo bit que indica la dirección de los datos (R/W). Un 0 en el bit R/W indica una escritura mientras que un 1 en el bit RW indica una lectura [40].

Si el mensaje es recibido por el esclavo con el identificador correspondiente, este valida la transmisión con un bit

de confirmación ACK, cada dato transmitido debe ir acompañado de su respectivo bit de ACK. Una vez transmitidos todos los datos, se genera la condición de STOP para finalizar la comunicación [40].

Parte IV

Diseño y construcción

14 Desarrollo del concepto

14.1. Definición de los requerimientos

Para el desarrollo del sistema de geolocalización, telemetría y diagnóstico de unidades vehiculares bajo el concepto de IoT, se definió una lista de requerimientos mínimos necesarios para cubrir las funcionalidades presentes en dispositivos comerciales en este mercado. Esta lista es el resultado del estado del arte de las empresas que ofrecen este tipo de dispositivos y servicios.

1. El sistema deberá ser capaz de obtener las coordenadas geográficas del vehículo.
2. El sistema podrá monitorear variables específicas e información de diagnóstico directamente de la unidad de control electrónico del vehículo
3. El sistema será capaz de detectar patrones de conducción y con base en ellos asignar una calificación numérica a la calidad de conducción del vehículo.
4. El sistema deberá tener un bajo consumo de energía tal que no comprometa la batería del vehículo.
5. El sistema tendrá la capacidad de comunicarse a la red celular y a través de ella conectarse a internet y actualizar la telemetría obtenida del vehículo en un servidor.

14.2. Definición de concepto

Con base en los requerimientos definidos para esta aplicación, se desarrolló el concepto del sistema de geolocalización, telemetría y diagnóstico de unidades vehiculares bajo el concepto de IoT. En la Figura 14.1 se muestra el diagrama del concepto, donde se observan los diferentes bloques que conforman el sistema y cómo se relacionan entre sí.

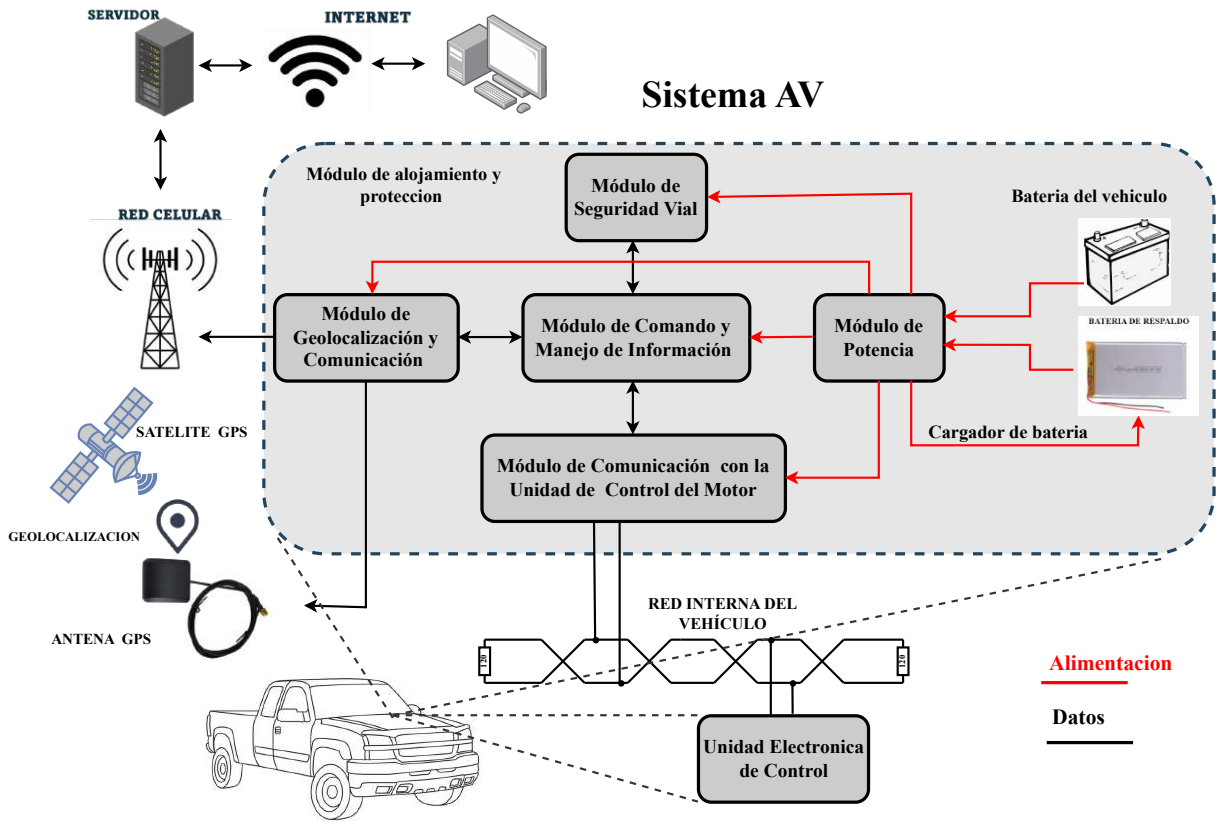


Figura 14.1: Definición de concepto

En la Figura 14.1 se muestran los módulos que conforman el sistema. El Módulo de Comando y Manejo de Información (CMI) es el encargado de gestionar todas las acciones de control y obtener la información de los módulos de Geolocalización y Comunicación (GC), de Comunicación con la Unidad de Control del Motor (CUCM) y del módulo de Seguridad Vial (SV). El módulo de potencia (P) es el encargado de acondicionar las entradas de voltaje de alimentación, en este caso la batería del vehículo y la batería de respaldo respectivamente, el módulo P carga la batería de respaldo y alimenta el sistema con la batería del vehículo cuando está conectada y usa la batería de respaldo para alimentar el sistema cuando la batería del vehículo está desconectada.

El módulo GC se encarga de obtener la geolocalización del vehículo y de conectarse a un servidor con acceso a internet a través de la red celular. El módulo CUCM hace el papel de interfaz entre el módulo CMI y la red interna del vehículo para comunicarse directamente con su unidad de control electrónico. El módulo SV es el encargado de monitorear las variables necesarias para determinar patrones de conducción y gestión de alertas. Finalmente, todo el sistema es protegido por el módulo de protección y alojamiento para garantiza su integridad física.

14.3. Requerimientos de los bloques conceptuales

Una vez planteada la definición del concepto en bloques funcionales ver Figura 14.1, se definen ahora los requerimientos específicos de cada uno de los bloques que conforman el sistema. En la Tabla 14.1se muestra un resumen de los requerimientos de cada uno de los módulos.

Tabla 14.1: Requerimientos de los bloques conceptuales

Módulo	Requerimientos
CMI	- Deberá generar la secuencia de comandos necesarios para solicitar la información de geolocalización al módulo GC. - Tendrá que generar los mensajes OBDII de peticiones de información específicos que recibirá el vehículo a través del módulo CUCM. - Deberá ser capaz de interpretar las respuestas de los mensajes OBDII del vehículo obtenidas a través del módulo CUCM. - Deberá asignar una puntuación numérica con base en la interpretación de la información obtenida del módulo SV. - Deberá generar la secuencia de comandos necesaria para subir la información recopilada a un servidor conectado a internet.
GC	- Deberá ser capaz de ofrecer una interfaz entre los comandos de control y las señales físicas que se trasmiten a la red celular para conectarse a internet - Será capaz de conectarse a un GNSS para obtener las coordenadas geográficas del módulo.
CUCM	- Se encargará de acondicionar las señales digitales provenientes del CMI a señales diferenciales que se usan en la red interna del vehículo.
SV	- Deberá contar con sensores que permitan determinar patrones de conducción.
P	- Deberá ser capaz de cargar una batería de respaldo cuando la batería del vehículo está conectada y usar la batería de respaldo en cuanto se detecte que la batería del vehículo está desconectada.
PA	- Podrá contener el hardware del sistema de tal manera que pueda ser colocado dentro del vehículo. - Debe contar con los materiales adecuados para soportar las condiciones donde sea colocado.

15 Diseño a nivel sistema

Con base en el desarrolló de concepto se realiza el diseño a nivel sistema, donde se describe a detalle la función de cada uno de los bloques que conforman el sistema, así como las interfaces que se utilizan para comunicarse entre ellos.

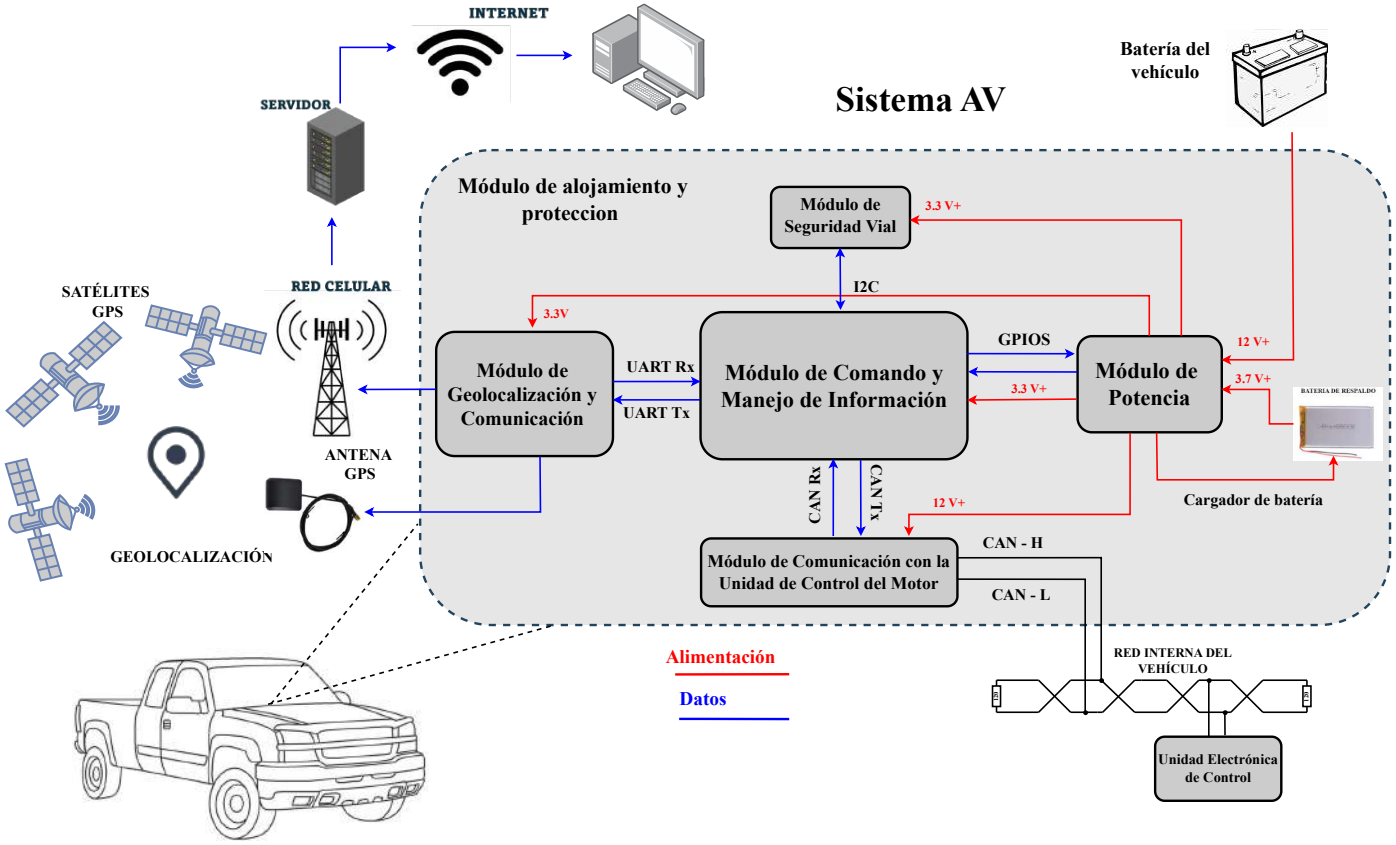


Figura 15.1: Diseño a nivel sistema

En la Figura 15.1 se muestra la arquitectura del sistema que consiste en primera instancia en dos fuentes de alimentación, la principal de 12V+ de la batería del vehículo y la batería de respaldo de 3.7V+ que es utilizada en ausencia de la batería principal. Las líneas de alimentación y su flujo están resaltadas en color rojo, mientras que las líneas de color azul son de las diferentes interfaces utilizadas por los módulos para comunicarse. En la arquitectura también se muestran las dos líneas del protocolo de comunicación que utilizan los vehículos para que dispositivos de procesamiento puedan acceder su información, estas líneas (CAN-H y CAN-L) salen del módulo CUCM y se conectan a la red interna del vehículo.

15.1. Módulo de Potencia

Las fuentes de alimentación son gestionadas por el módulo P mientras la batería del vehículo este conectada, el voltaje de la batería es regulador para suministrar la alimentación al sistema y cargar la batería de respaldo, sin embargo, cuando la batería del vehículo es desconectada, el módulo P regula entonces el voltaje de la batería de respaldo para el suministro de alimentación.

15.2. Módulo de comando y manejo de información

Este bloque es el encargado de gestionar todas las acciones de control del sistema y detecta la presencia o ausencia de la batería del vehículo. Cuando la batería del vehículo está conectada, todos los bloques del sistema están en funcionamiento y el CMI genera solicitudes de información con base en la interfaz de comunicación de cada bloque en particular, la información solicitada de cada bloque es la siguiente:

- **GC** - Coordenadas geográficas.
- **CUCM** - Velocidad del vehículo, RPM del motor, temperatura del líquido de enfriamiento y códigos de diagnóstico de error.
- **SV** - Aceleración lineal, Aceleración angular y temperatura.

El CMI analiza la información del SV para identificar patrones de conducción y asignar una calificación numérica que evalúa la calidad del manejo. Toda esta información conforma la telemetría del vehículo. Finalmente, el CMI solicita al módulo GC el envío de la telemetría cada 30 segundos. Sin embargo, cuando el sistema detecta que el vehículo está apagado, la solicitud de envío se realiza cada 5 minutos, sin considerar la información del módulo CUCM, ya que la ECU del vehículo solo opera cuando este está encendido.

Cuando el módulo P detecta que la batería del vehículo está desconectada, envía una señal al CMI indicando que el sistema se alimenta de la batería de respaldo. Esta batería debe tener una capacidad suficiente para garantizar una autonomía de al menos 24 horas. En este estado, el bloque CUCM no está disponible, por lo que la telemetría se compone únicamente de la información de los bloques GC y SV. Finalmente, el CMI solicita al módulo GC el envío de la telemetría cada 10 minutos.

15.3. Módulo de Geolocalización y Comunicaciones

Este bloque es el encargado de determinar las coordenadas geográficas a través del sistema de navegación por satélite GPS con una precisión de $\pm 2m$ y conectarse a la red celular para enviar telemetría determinada por el CMI a internet usando el protocolo HTTP.

15.4. Módulo de Seguridad Vial

El sistema de seguridad tiene como función determinar la temperatura, la variación de la aceleración en los ejes X, Y y Z, y la variación de la velocidad angular del sistema en los mismos ejes de tal manera que el CMI use esta información para detectar aceleración, frenados y giros bruscos para determinar la calidad de la conducción del vehículo.

15.5. Módulo de Comunicación con la Unidad de Control del Motor

Este módulo tiene como función acondicionar las señales digitales de los mensajes OBDII estructurados por el CMI a señales diferenciales para que los mensajes puedan ser interpretados correctamente por la ECU del vehículo, de igual manera el módulo CUCM acondiciona las respuestas estructuradas con señales diferenciales del vehículo a señales digitales que pueda interpretar el CMI.

16 Diseño a detalle

Con base en los requerimientos definidos para cada bloque en el diseño a nivel sistema, se proponen las arquitectura y tecnologías de las soluciones de cada uno de ellos, especificaciones completas de la geometría, circuitos, materiales, tolerancia del producto, documentación de control, especificaciones de componentes, planes de procesos de fabricación y ensamblaje, así como las metas de desempeño y limitaciones.

16.1. Diseño a detalle del módulo CMI

El CMI debe cumplir con las especificaciones planteadas en el desarrollo a nivel sistema, por lo tanto, el microcontrolador principal del sistema debe tener un bajo consumo de energía tal que no comprometa la integridad de la batería del vehículo, a su vez este microcontrolador debe tener las interfaces necesarias para controlar y gestionar a los demás bloques.

Para esta aplicación se seleccionó el microcontrolador STM32L452RE, de ultra baja potencia, basado en un núcleo ARM de 32 bits Cortex-M4 de alto rendimiento que funciona a una frecuencia de hasta 80MHz y soporta una temperatura de entre -40 a 150 grados centígrados. Este microcontrolador cuenta con interfaces de comunicación estándar y avanzadas, cuatro I2C, tres SPI, tres USART, un UART, un UART de bajo consumo, un SDMMC, un CAN y un dispositivo USB.

En la figura 16.1 se muestra a detalle el flujo de datos y las interfaces que el CMI utiliza para controlar y gestionar cada uno de los bloques. Se utiliza la interfaz CAN2.0B configurado a una velocidad de 500 kits/s para transmitir mensajes de solicitud y recibir mensajes de respuesta estructurados con base en el estándar de diagnóstico de a bordo OBDII utilizado para comunicarse con la Unidad Electrónica de control del vehículo a través del módulo CUCM.

Para la comunicación con el módulo GC se utiliza la interfaz UART configurada a una velocidad de 9600bits/s para transmitir una secuencia de comandos AT específicos para asegurar una conexión estable a la red celular y a diferentes sistemas globales de navegación por satélite, en esta aplicación el GNSS seleccionado es el GPS. Para la comunicación con el módulo SS la interfaz seleccionada es I2C configurada a una velocidad de 400 kits/s para obtener la información de la aceleración lineal y angular del módulo MPU6050 implementado en el módulo SV.

Cualquier microcontrolador requiere de hardware externo para su correcto funcionamiento, en la figura 16.2 se muestra el circuito que recomienda el fabricante para microcontroladores de la familia STM32L4XXXX.

Para describir las tareas que debe ejecutar el microcontrolador principal (STM32L452RE) para el funcionamiento del sistema, se dividen las interacciones que el CMI debe realizar con cada uno de los bloques y se describen en el diseño a detalle de cada bloque.

16.2. Diseño a detalle del módulo CUCM

En el diseño de este módulo se seleccionó el MCP2561 CAN de alta velocidad diseñado por Microchip como dispositivo transceptor encargado de acondicionar señales digitales a señales diferenciales propias del protocolo CAN, sus principales características son:

- Soporta los estándares ISO - 11898-2 y ISO - 11898-5 para redes CAN de alta velocidad.
- Soporta velocidades de hasta 1 Mbps.

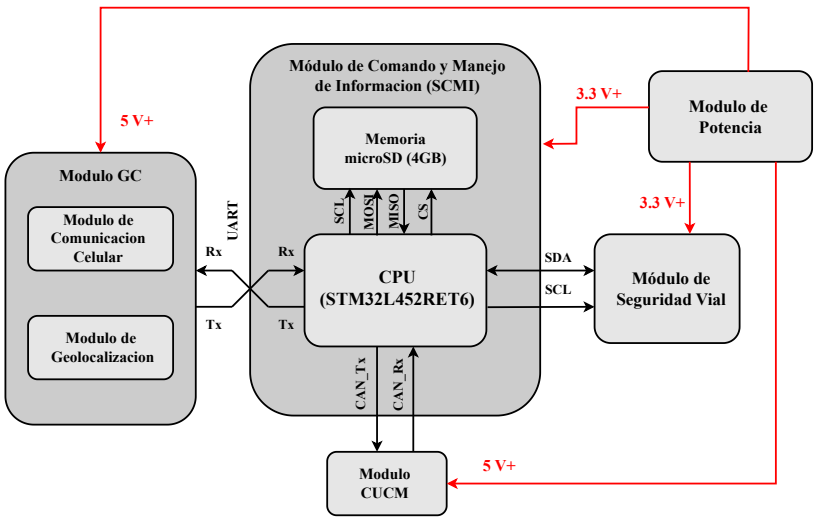


Figura 16.1: Interconexión del CMI con los módulos CUCM, GC, SS y P

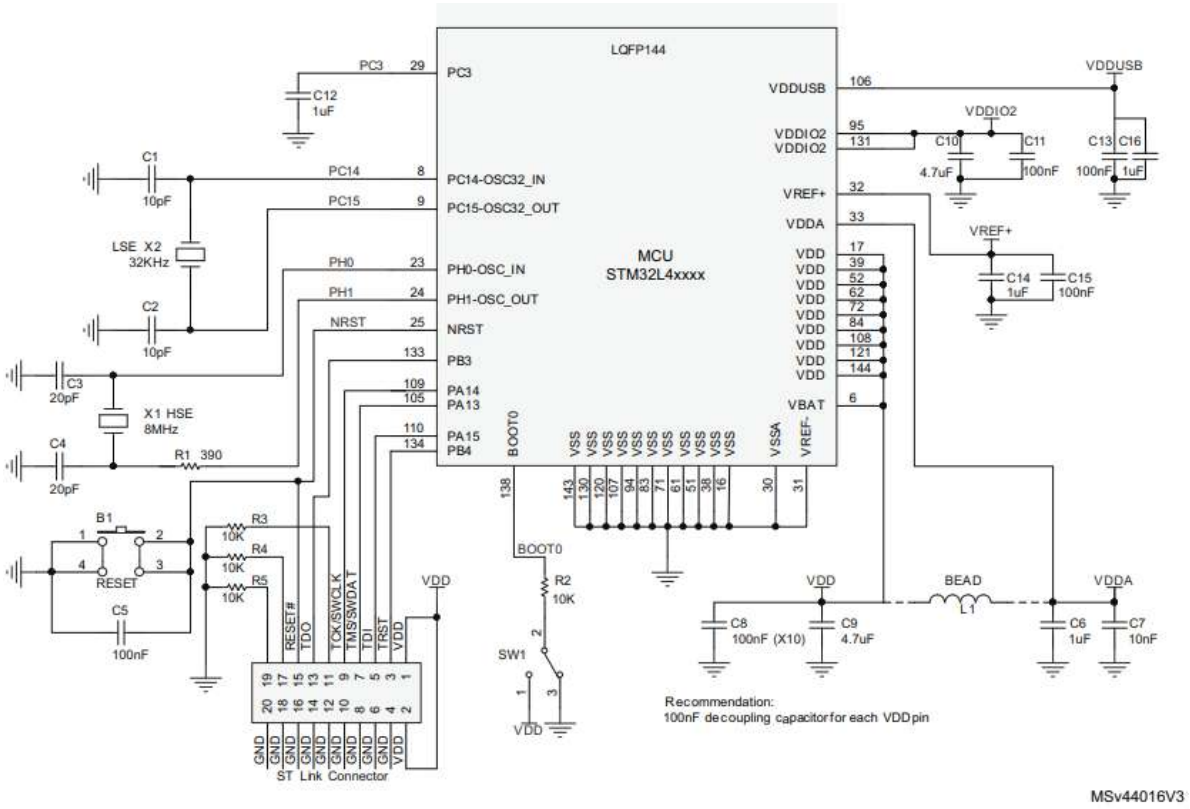


Figura 16.2: Esquemático para microcontroladores STM32L4XXXX [41]

- Bajo consumo en modo normal y un consumo ultra bajo de $5\mu A$ en modo stanby.
- Opera con un voltaje de 4.5V a 5.5V.
- Opera en el rango de temperaturas de $-40°C$ a $+150°C$.
- Protección contra corto circuito, contra transitorios de alto voltaje.
- Apagado térmico automático como protección.

- Ofrece alta resistencia al ruido electromagnético.

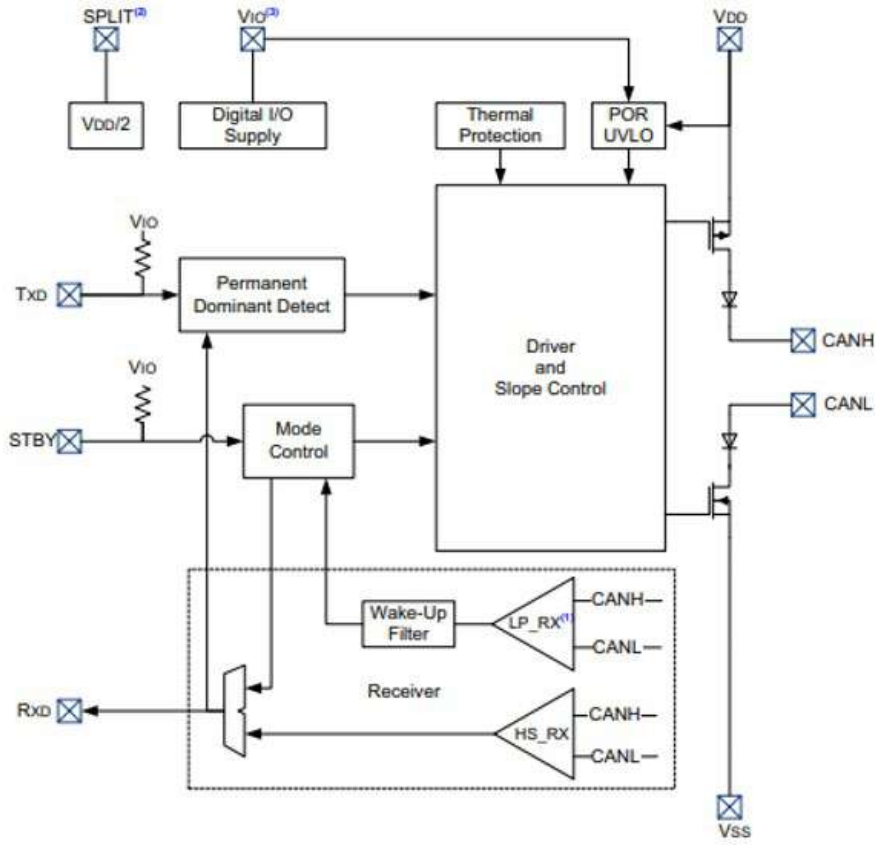


Figura 16.3: Diagrama de bloques MCP2561 [42]

Se seleccionó el empaqueto SOIC de montaje superficial, en la Figura 16.3 se muestra el diagrama de bloques que describe la arquitectura interna del MCP2561.

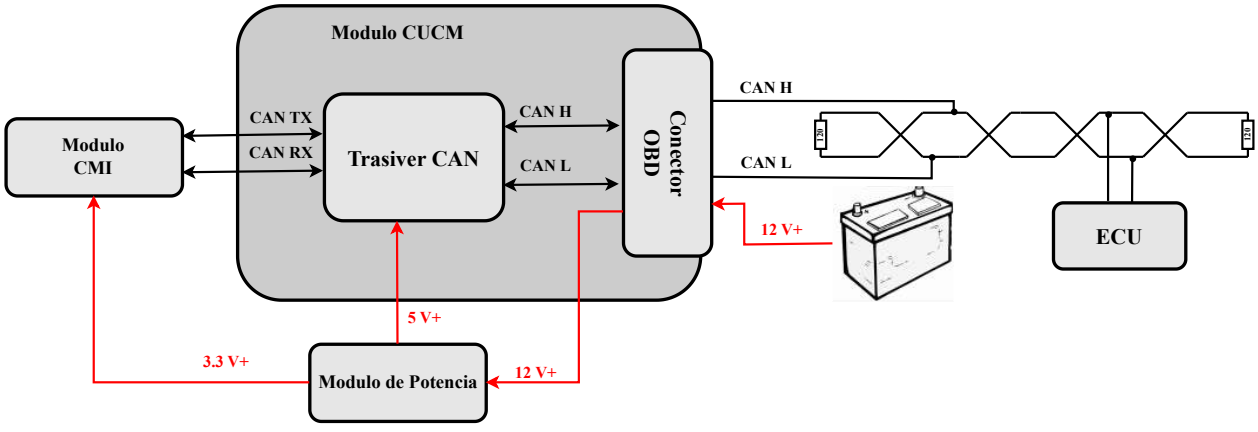


Figura 16.4: Diseño a detalle CUCM

El la Figura 16.4 se muestra el diagrama de bloques detallado del módulo CUCM, donde se resaltan las líneas CAN_TX y CAN_Rx que van del módulo CMI al módulo CUCM para ser acondicionadas a los niveles de voltajes adecuados que se propagan a lo largo de la red interna del vehículo a través del conector OBDII estandarizado por SAE J1962, así como las líneas de alimentación de cada uno de los bloques desde el módulo P.

La tarea principal del CMI con el módulo CUCM es obtener la telemetría del vehículo con base en los mensajes de petición y repuesta especificados en el estándar de diagnóstico y monitoreo OBDII. Todos los mensajes OBDII son transmitidos siguiendo las especificaciones del estándar CAN2.0B. En la Figura 16.5 se muestra la estructura de los mensajes de solicitud y de respuesta OBDII con base en la trama de datos de CAN2.0B vista en el marco teórico. Los identificadores que diferencian a los mensajes de solicitud y de respuesta son 0x7DF y 0x7E8 respectivamente, por lo tanto, el identificador 0x7DF pertenece al microcontrolador principal mientras que 0x7E8 pertenece al vehículo. El campo de datos se coloca el Parámetro ID (PID) referente a cada solicitud.

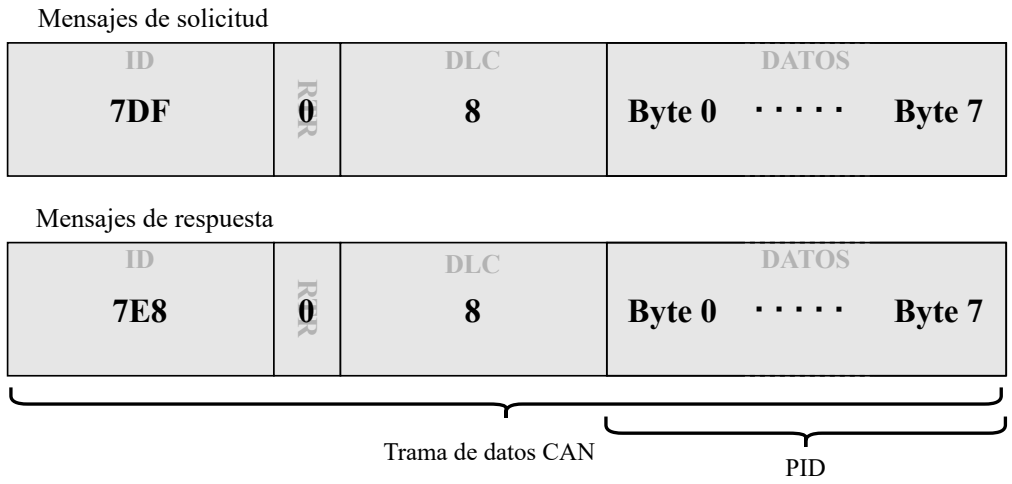


Figura 16.5: Mensajes de solicitud y de respuesta OBDII entre el CUCM y la ECU del vehículo.

En la Figura 16.6 se muestra la estructura del mensaje de petición de la velocidad usando comandos PID definidos por el estándar OBDII, este mensaje se divide en ocho bytes de información, el primer byte nos indica el tamaño del mensaje, por ejemplo, en este caso el Byte 0 tiene un valor de 02, lo que significa que el mensaje consta de dos Bytes de información, correspondientes al Byte 1 y Byte 2. El Byte 1 corresponde al modo que estamos utilizando ver [Marco teórico], en tanto que el Byte 2 corresponde al PID de la velocidad y del Byte 3 al 7 no son tomados en cuenta.

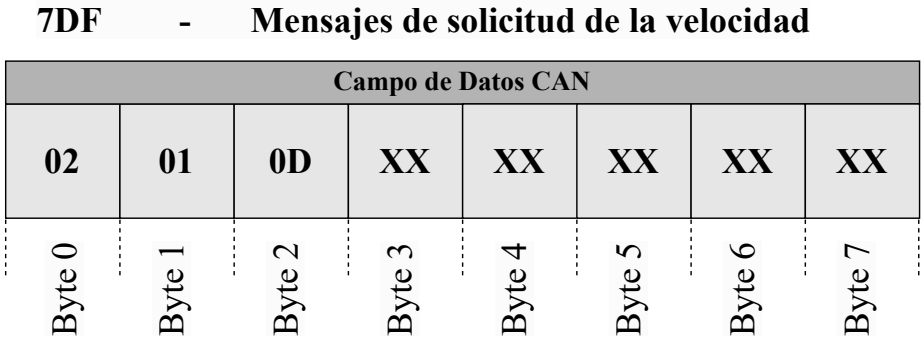


Figura 16.6: Mensajes de solicitud de la velocidad

En la Figura 16.7 se muestra la respuesta de la velocidad por parte del vehículo, esta respuesta se encuentra en el Byte 3, en este caso la información del Byte 3 no debe operarse para obtener la información de la velocidad a diferencia de otros mensajes de respuesta OBDII, por lo tanto, en este caso la velocidad es de 0x32 en hexadecimal y 50 en decimal, siendo la información interpretada de 50 km/h.

7DF - Mensajes de solicitud de la velocidad

Campo de Datos CAN							
03	**41**	**0D**	**32**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figura 16.7: Mensajes de respuesta de la velocidad

En la Figura 16.8, se muestra el mensaje de la solicitud y la respuesta de las RPM del motor, en este caso los operados de la fórmula para calcular las RPM son el Byte 3 y el Byte 4, la fórmula para interpretar las RPM es:

$$RPM = \frac{256A + B}{4}$$

7DF - Mensajes de solicitud de las RPM

Campo de Datos CAN							
02	**01**	**0C**	**XX**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

7E8 - Mensajes de respuesta de las RPM

Campo de Datos CAN							
04	**41**	**0C**	**A**	**B**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figura 16.8: Mensajes de solicitud y respuesta de las RPM

En la Figura 16.9 se muestra los mensajes para obtener la temperatura del líquido de enfriamiento del motor, en este caso la fórmula para interpretar esta información es: $Temperatura = [\, A - 40 \,]°C$

7DF - Mensajes de temperatura del vehículo

Campo de Datos CAN							
02	**01**	**05**	**XX**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

7E8 - Mensajes de respuesta de la temperatura del vehículo

Campo de Datos CAN							
03	**41**	**05**	**A**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figura 16.9: Mensajes de solicitud y respuesta de la temperatura del líquido de enfriamiento

En la Figura 16.10 se muestra los mensajes tanto de petición como de respuesta para obtener el nivel de combustible, en este caso la fórmula para interpretar la respuesta es : $Combustible = \frac{A}{2{,}55}\,\%$

7DF - Mensajes de Nivel de Combustible

Campo de Datos CAN							
02	**01**	**2F**	**XX**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

7E8 - Mensajes de respuesta del Nivel de Combustible

Campo de Datos CAN							
03	**41**	**2F**	**A**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figura 16.10: Mensajes de solicitud y respuesta del nivel de combustible

En la Figura 16.11 se muestra los mensajes para obtener los DTC, cada error es identificado por un código en específico y varía entre fabricantes de vehículos. Finalmente, toda la telemetría del vehículo es guardada en una memoria uSD con la interfaz SPI, la serie de comandos utilizados para la escritura es la siguiente:

7DF - Mensajes de Códigos de Detección de Error

Campo de Datos CAN							
02	**03**	**XX**	**XX**	**XX**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

7E8 - Mensajes de respuesta de los Códigos de Detección de Error

Campo de Datos CAN							
03	**41**	**P**	**04**	**20**	**XX**	**XX**	**XX**
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figura 16.11: Mensajes de solicitud y respuesta de los DTC

En la Figura 16.12 se muestra el circuito recomendado por el fabricante para su uso.

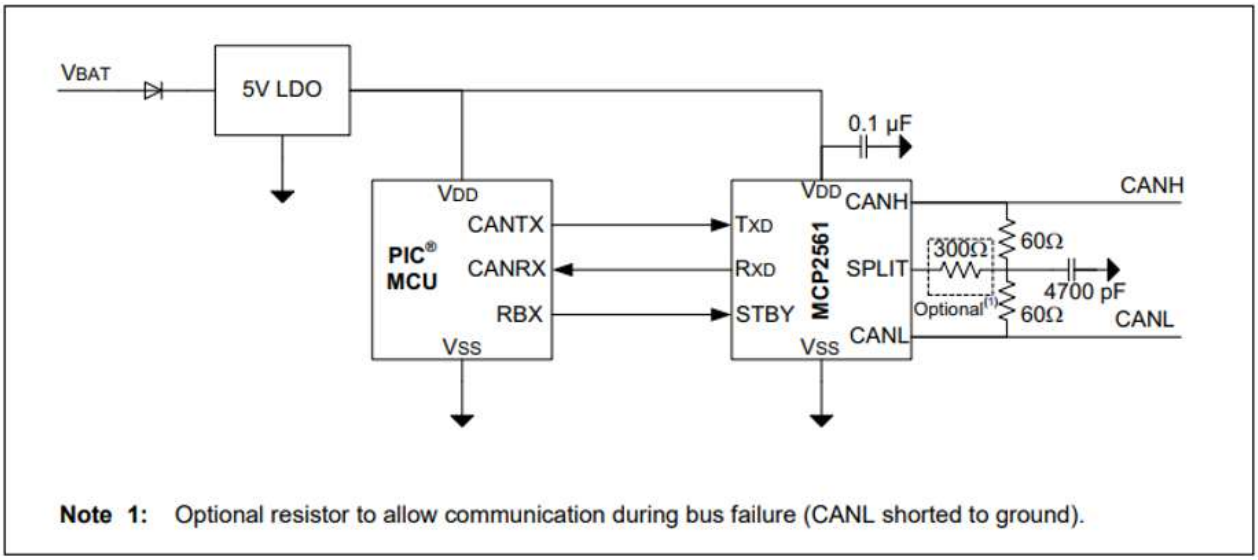


Figura 16.12: Circuito recomendado por el fabricante [42]

16.3. Diseño a detalle del módulo GC

Para el módulo GC se propone como solución utilizar un módulo LPWA BG95-M3 multi-modo que soporta LTE-Cat M1/Cat-NB2/EGPRS e integra sistemas GNSS. Ofrece velocidades máximas de datos de 588 Kbps en descarga y 1119 kbps en subida bajo LTE Cat M1. Cuenta con un consumo de energía ultra bajo al aprovechar la RAM/flash integrada, así como el procesador ARM Cortex A7, cuenta con un factor de forma SMT de 23,6 mm x 19,9 mm x 2,2 mm y un alto nivel de integración, BG95-M3 permite a los desarrolladores diseñar fácilmente sus aplicaciones y aprovechar el bajo consumo de energía.

El módulo BG95 integra un motor multi-GNSS compatible con los sistemas GPS, BeiDou, Galileo, GLONASS y QZSS, puede recibir señales simultaneas de un máximo de dos constelaciones (GPS y otra constelación). En la Figura se muestra el diagrama de bloques detallado del módulo GC.

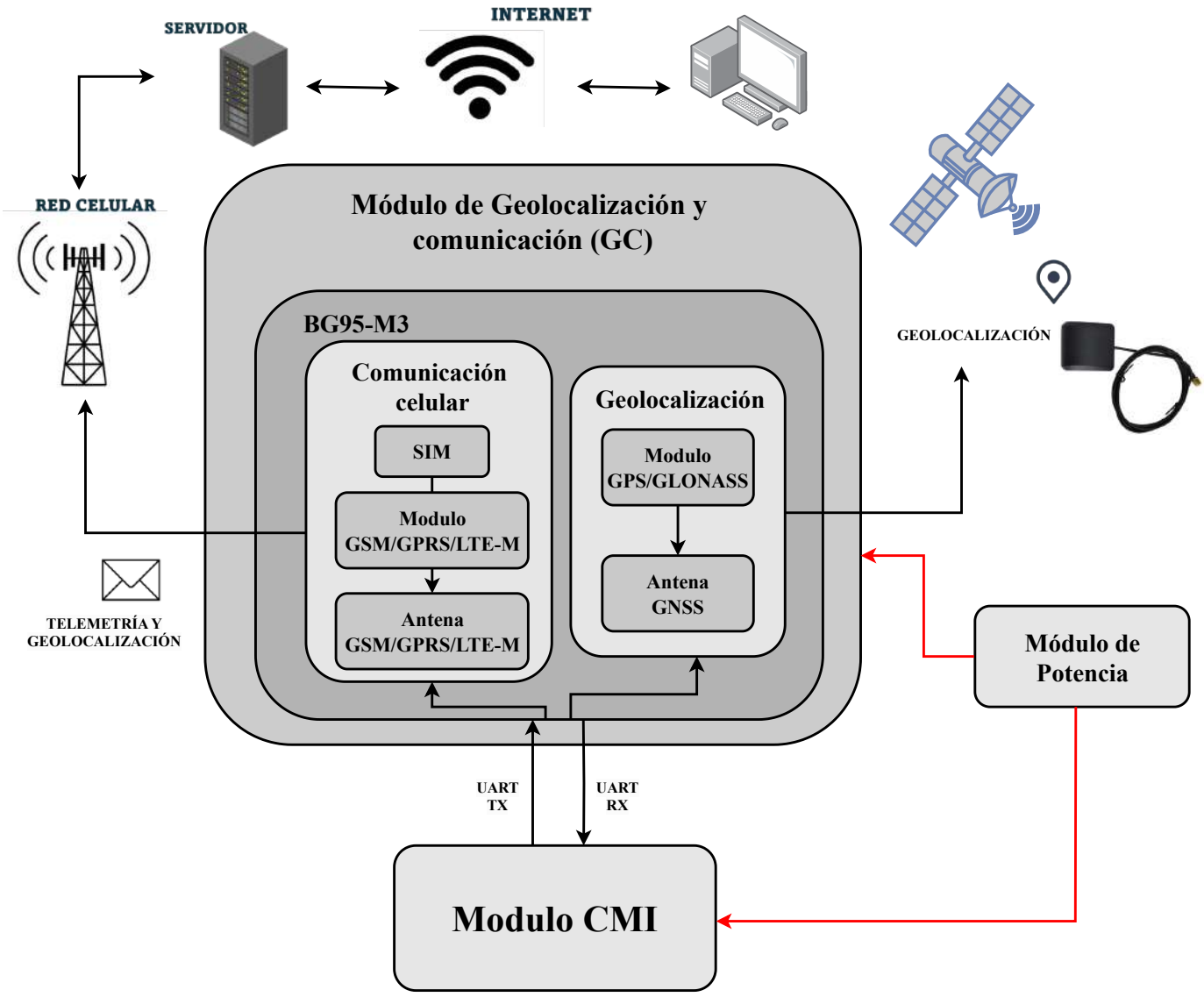


Figura 16.13: Módulo de Geolocalización y comunicación

La tarea principal del módulo GC es obtener la geolocalización del vehículo y gestionar la transmisión de la

telemetría a internet por medio de la red celular. El CMI debe controlar al módulo GC utilizando comandos AT que son transmitidos por medio de UART. En la Tabla 16.1 se muestran los comandos AT necesarios para obtener la geolocalización.

Tabla 16.1: Comandos AT para GNSS/GPS

Comando AT	Descripción
AT+GPS	Enciende el GNSS/GPS del módulo BG95-M3
AT+GPSLOC	Solicita la ubicación actual del módulo

La transmisión de la telemetría a Internet se realiza usando el protocolo HTTP, configurado mediante comandos AT. En la Tabla 16.2 se muestran los comandos AT necesarios para enviar información utilizando dicho protocolo.

Tabla 16.2: Comandos AT para envió de datos por HTTP

Comando AT	Descripción
AT+QICSGP	Configura el contexto Packet Data Protocol (PDP, por sus siglas en ingles) y el APN del proveedor de red
AT+QIACT=1	Activa el contexto PDP configurado
AT+QIACT?	Consula el estado actual del contexto PDP; devuelve IP asignada
AT+QHTTPCFG='contextid',1	Asigna el contexto PDP 1 al cliente HTTP
AT+QHTTPURL=XX	Define la longitud de la URL a trasmitir
AT+QHTTPGET	Realiza una petición HTTP GET al servidor

Para la trasmisión de la telemetría a internet se usa el protocolo 'Hypertext Transfer Protocol' (HTTP) para realizar peticiones de recursos a un servidor, siguiendo la arquitectura Cliente-Servidor. Existen más protocolos para enviar información a internet y se usan dependiendo la aplicación, el módulo GC es capaz de implementar otros protocolos para acceder a internet, como lo son: FTP, HTTPS, MQTT, SMTP.

16.4. Diseño a detalle del módulo SV

Para el módulo SV se propone usar el módulo MPU6050, este módulo es un dispositivo de monitoreo de movimientos físicos de 6 ejes, que combina un giroscopio de 3 ejes, un acelerómetro de 3 ejes con un Procesador de Movimiento Digital (DMP), integra obleas MEMS con electrónica CMOS a través de una unión a nivel oblea encapsulado en un empaquetado de $4\,mm \times 4\,mm \times 0{,}9\,mm$ (QFN).

El módulo MPU6050 cuenta con tres convertidores analógicos-digitales (ADC) de 16 bits para digitalizar las salidas del giroscopio, tres ADC de 16 bits para digitalizar las salidas del acelerómetro y un ADC para el sensor de temperatura. Cuenta con un rango de escala del giroscopio programable por el usuario ± 250, ± 500, ± 1000, $\pm 2000°/seg(dps)$, y un rango de escala del acelerómetro programable por el usuario de $\pm\, 2g$, $\pm 4g$, $\pm 8g$ *y* $\pm\, 16g$.

La comunicación con todos los registros del dispositivo se realiza utilizando I2C a 400kHz y opera con un rango de voltaje de fuente de alimentación VDD de 2.375V a 3.46V. En la Figura 16.14 se muestra el diagrama de bloques del módulo SV. El módulo CMI debe configurar la escala tanto del giroscopio como del acelerómetro y leer cada uno de sus registros utilizando la interfaz I2C, una vez obtenida la información el CMI la interpreta y se define umbrales límites a las variables de interés para determinar aceleraciones, frenados y giros bruscos, en este caso las variables son: aceleración lineal en el eje X, aceleración angular en el eje Z y temperatura respectivamente.

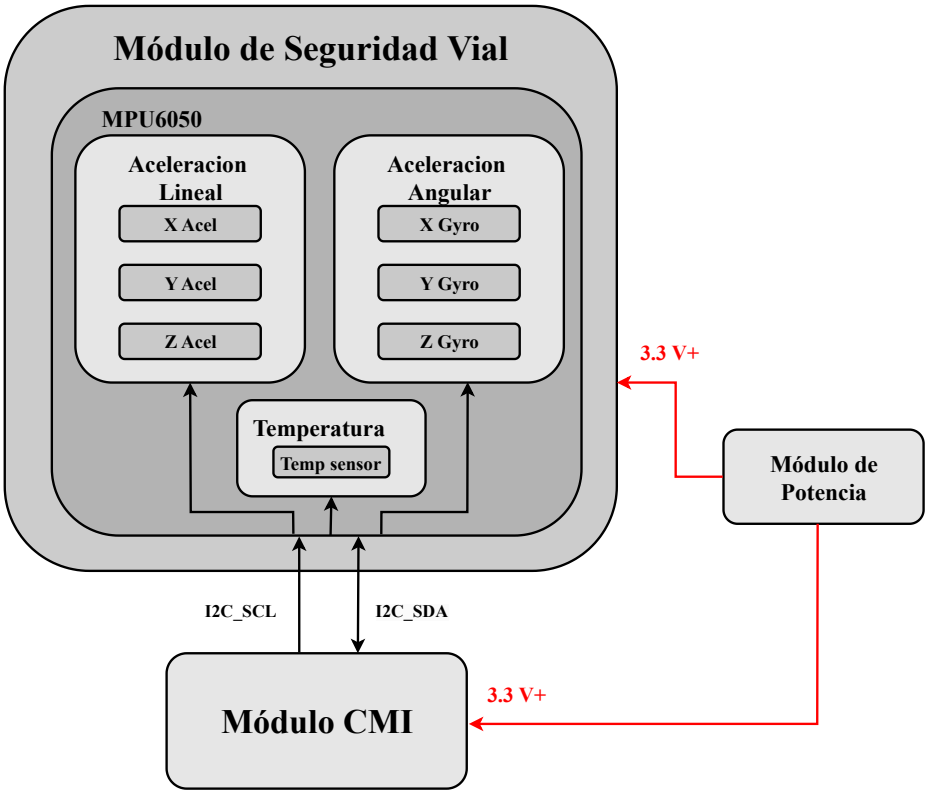


Figura 16.14: Diagrama de bloques del módulo SV

En la Figura 16.15 se muestra el diagrama de bloques que describe la arquitectura interna del módulo MPU6050, este diagrama de bloques es proporcionado por el fabricante.

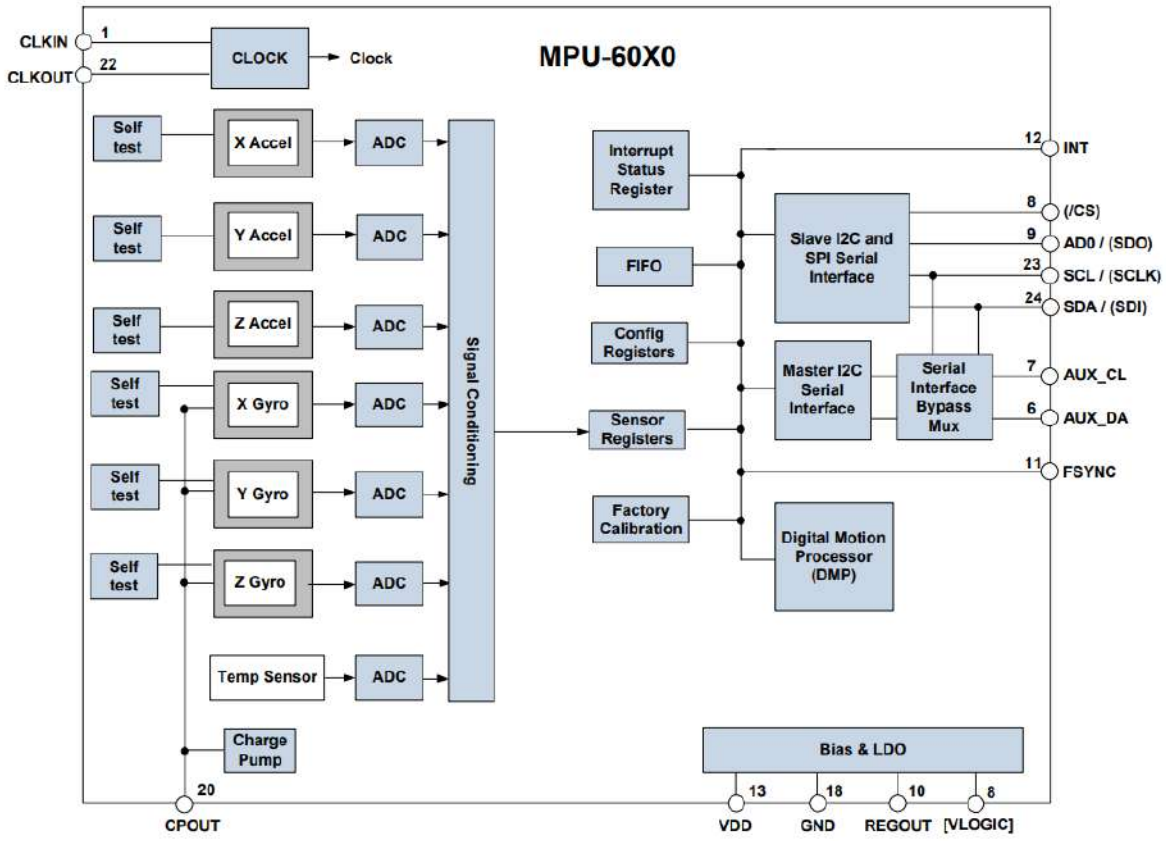


Figura 16.15: Diagrama de bloques del módulo MPU6050 [43]

De igual manera, el fabricante recomienda un circuito típico para usar este dispositivo, este circuito se muestra en la Figura 16.16.

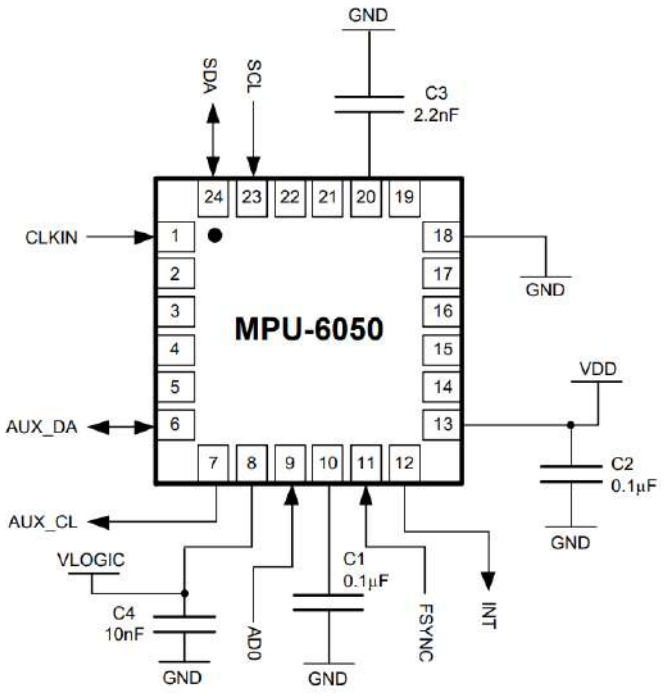


Figura 16.16: Circuito típico de módulo MPU6050 [43]

16.5. Diseño a detalle del módulo P

Para el módulo P se propone el diagrama de bloques mostrado en la Figura 16.17, la tarea principal de este módulo es detectar si la batería del vehículo está conectada o desconectada, cuando está conectada, la batería alimenta al regulador Step / Down para cargar la batería con el módulo cargador de batería. El conmutador PMOS detecta que la batería está conectada y conmuta la línea del regulador Step / Down al regulador Step / Up para generar las líneas de alimentación del sistema.

Cuando la batería del vehículo es desconectada el conmutador conecta la línea de alimentación de la batería de respaldo a el regulador Step / Up para generar las líneas de alimentación del sistema.

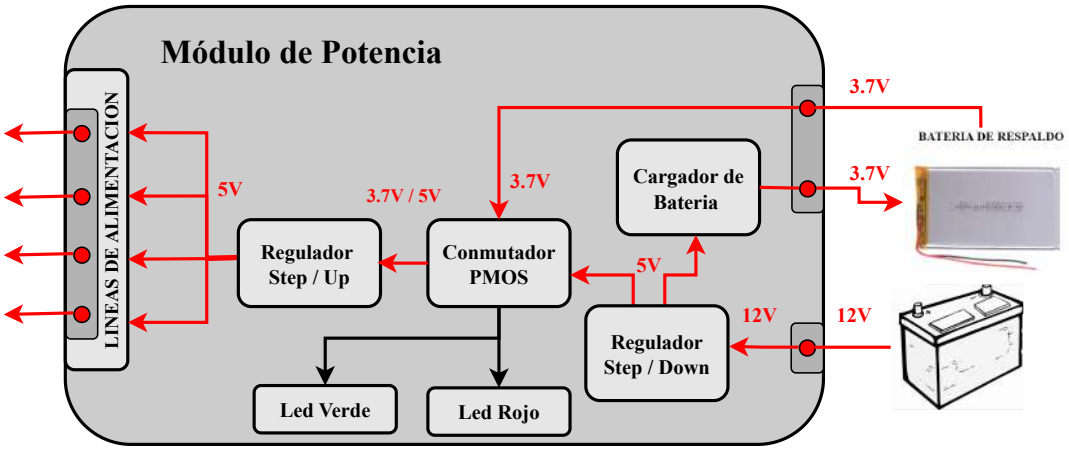


Figura 16.17: Diagrama de bloques de módulo P

Para el regulador Step Down se propone utilizar el módulo LM2596 en una placa de circuito impreso que contiene tanto al integrado LM2596 como la electrónica necesaria para que funcione, en la Figura 16.18 se muestra la PCB del módulo LM2596.

Figura 16.18: Regulador Step Down Módulo comercial LM2596

Este módulo tiene la capacidad de regular el voltaje de entrada de un circuito a partir de una fuente de alimentación con un voltaje mayor, soporta hasta 3A, temperaturas de -45°C a +85°C, voltajes de entrada de 4 a 35V y voltaje de salida de 2 a 28V, el voltaje de salida es regulado mediante un potenciómetro (tripod) multivuelta, el tamaño del módulo tiene las dimensiones de 43mm x 21mm x 14mm

Para el cargador de batería se usó el módulo TP4056 tipo C, este módulo permite cargar baterías de Li-Ion o LiPo de una celda 1S, está basado en el integrado TP4056, en la Figura 16.19 se muestra la placa PCB que contiene al integrado y la electrónica necesaria para que funcione.

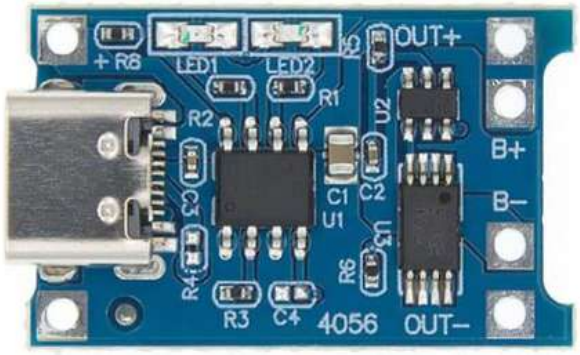


Figura 16.19: Cargador de batería Módulo comercial TP4056

Este módulo es un controlador de carga y descarga de baterías de iones de litio y de polímetro de litio, el voltaje típico de entrada es de 5V y máximo de +-8V, soporta una corriente de carga de 1A, voltaje de carga de 4.2 VDC, temperaturas de -40°C a 85°C y tiene un tamaño de 27.7mm x 17mm con un peso de 2g.

Para el regulador Step Up se propone utilizar el módulo "XL6019 Elevador de Voltaje Boost Step Up 20W 4", este módulo comercial es una placa de circuito impreso que tiene embebido el integrado XL6019, un trípode para ajustar el voltaje de salida, diodos, resistencias, inductores y capacitores que hacen funcionar al integrado adecuadamente, en la Figura 16.20 se muestra el módulo XL6019.

Figura 16.20: Módulo comercial XL6019

Este módulo es capaz de obtener voltajes entre -0.3V a 60V DC a partir de una fuente de voltaje inferior de entre -0.3V a 45V, puede manejar una carga de hasta 4A máximo, temperaturas de -40°C a 150°C y tiene dimensiones de 53mm x 26mm con un peso de 17g. En la Figura 16.21 se muestra cómo se conectan cada uno de los módulos para generar las líneas de alimentación del sistema.

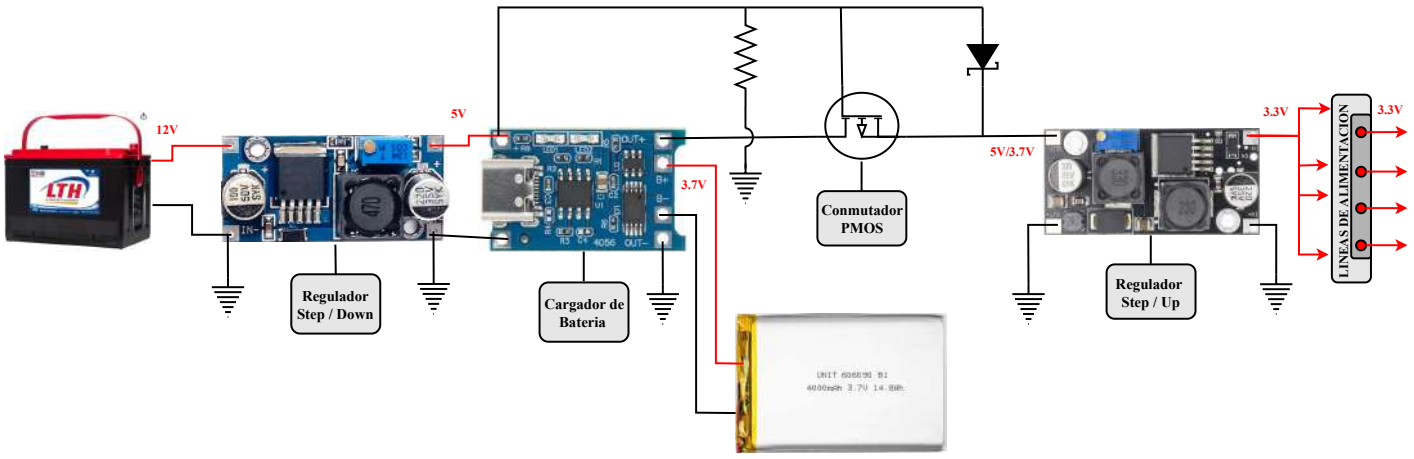


Figura 16.21: Integración del módulo P

16.6. Diseño a detalle del Firmware

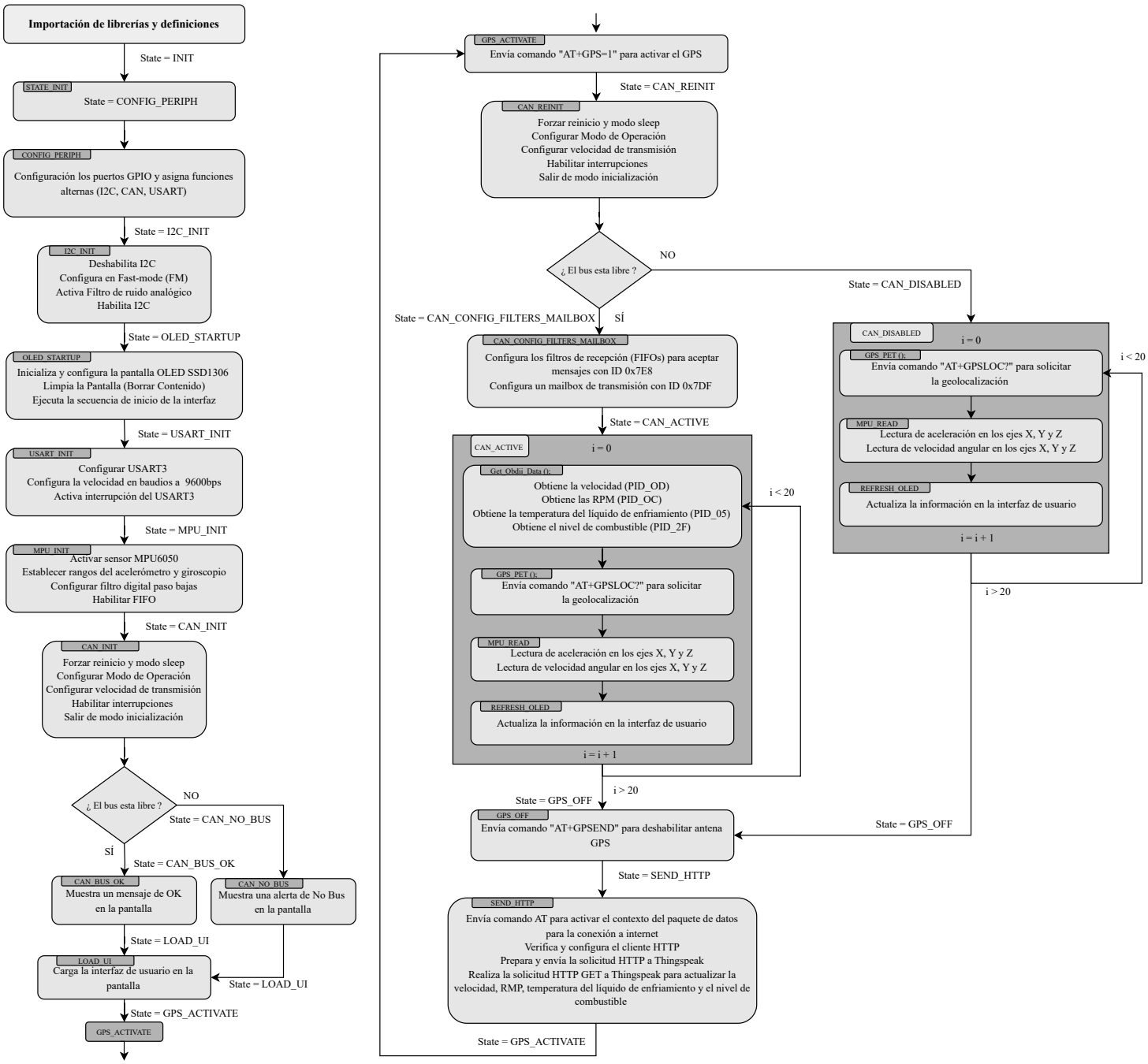


Figura 16.22: Diagrama de flujo del sistema

En la Figura 16.22 se muestra el diagrama de flujo del programa principal del archivo “main.c” del firmware. El flujo sistema está gestionado por una máquina de estados finita (FMS, por sus siglas en inglés), esto permite la ejecución ordenada y secuencial de la operación de los distintos periféricos.

En el diagrama de flujo mostrado, cada bloque representa un estado en específico del sistema, en la esquina superior derecha de cada bloque, se indica el nombre del estado correspondiente, tal como aparece en el código fuente, mientras que el cambio de estados está representado mediante flechas y la modificación de la variable “State”.

16.6.1. void Finite_State_Machine(SystemState *State);

En esta sección se muestra el código fuente en lenguaje C correspondiente a la implementación de la máquina de estados finita (FMS), que gestiona la lógica de operación del sistema, la función Finite_State_Machine recibe como argumento un puntero a una variable de tipo SystemState_t, la cual representa el estado actual de la máquina.

Dentro de la funcion, se emplea una estructura de control switch (*State), para acceder al valor actual de cada estado, cada case dentro del switch representa un estado definido, y contiene las acciones específicas que deben ejecutarse en dicho estado. Al finalizar, cada bloque modifica el valor apuntado por *State, para pasar hacia el siguiente estado correspondiente, de acuerdo a la lógica del programa.

```
1  void Finite_State_Machine(SystemState_t *State) {
2      switch (*State) {
3          case INIT:
4              *State = CONFIG_PERIPH;
5              break;
6
7          case CONFIG_PERIPH:
8  /***** INICIALIZACIÓN Y CONFIGURACIÓN DE PUERTOS Y PERIFÉRICOS *****/
9              Init_Peripheral();    // Inicializa Periféricos del sistema
10             Conf_Peripheral();    // Configura los puertos GPIO y asigna funciones
                   alternas (I2C, CAN, USART)
11             *State = I2C_INIT;
12             break;
13
14         case I2C_INIT:
15             I2C_Init(I2C3);  // Inicializa el periférico I2C3 (comunicación con
                   MPU6050 y OLED SSD1306)
16             *State = OLED_STARTUP;
17             break;
18
19         case OLED_STARTUP:
20             SSD1306_Init();       // Inicializa y configura la pantalla OLED SSD1306
21             SSD1306_Clear();      // Limpia la pantalla (borra el contenido)
22             SecuanciaInicio();    // Ejecuta la secuencia de inicio de la interfaz
                   en pantalla
23             *State = USART_INIT;
24             break;
25
26         case USART_INIT:
27             USARTx_CONF(USART3, USARTx_BRR_MSI_16MHz, USART3_IRQn, 7);    //
                   Configura USART3 a 16 MHz, habilita interrupciones y prioridad NVIC =
                    7
28             I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0); //
                   Selecciona la página 0 del display
29             SSD1306_PosCom(0);    // Posiciona el cursor en la columna 0
30             SSD1306_WriteString("!!␣USART␣OK␣!!");
31             for (int i = 0; i < 4; i++) { delay(1000000);} // Ejecuta 4 retardos
                   consecutivos para dar tiempo a que el usuario visualice el mensaje
32             SSD1306_Clear();
33             *State = MPU_INIT;
34             break;
35
36         case MPU_INIT:
37             MPU6050_Init();
38             I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0); //
                   Selecciona la página 0 del display
```

```
39                SSD1306_PosCom(0);    // Posiciona el cursor en la columna 0
40                SSD1306_WriteString("!! MPU OK !!");
41                for (int i = 0; i < 4; i++) { delay(1000000);} // Ejecuta 4 retardos
                      consecutivos para dar tiempo a que el usuario visualice el mensaje
42                SSD1306_Clear();
43                *State = CAN_INIT;
44                break;
45
46            case CAN_INIT:
47                if (CANx_Init(CAN1)) {*State = CAN_BUS_OK;}
48                else {*State = CAN_NO_BUS;}   // Timeout en INIT
49                break;
50
51            case CAN_NO_BUS:
52                I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0); //
                      Selecciona la página 0 del display
53                SSD1306_PosCom(0);    // Posiciona el cursor en la columna 0
54                SSD1306_WriteString("!! CAN BUS NO !!");
55                for (int i = 0; i < 4; i++) { delay(1000000);} // Ejecuta 4 retardos
                      consecutivos para dar tiempo a que el usuario visualice el mensaje
56                SSD1306_Clear();
57                *State = LOAD_UI;
58                break;
59
60            case CAN_BUS_OK:
61                I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0); //
                      Selecciona la página 0 del display
62                SSD1306_PosCom(0);    // Posiciona el cursor en la columna 0
63                SSD1306_WriteString("!! CAN BUS OK !!");
64                for (int i = 0; i < 4; i++) { delay(1000000);} // Ejecuta 4 retardos
                      consecutivos para dar tiempo a que el usuario visualice el mensaje
65                SSD1306_Clear();
66                *State = LOAD_UI;
67                break;
68
69            case LOAD_UI:
70                SSD1306_LoadUI();
71                *State = GPS_ACTIVATE;
72                break;
73
74            case GPS_ACTIVATE:
75                delay(100000);        // Pequeño retardo para asegurar la correcta
                      inicialización del sensor
76                sendStringUARTx(USART3, "AT+QGPS=1\r\n"); // Envía comando AT por USART3
                       al módulo GSM/GPRS para activar el GPS interno
77                *State = CAN_REINIT;
78                break;
79
80            case CAN_REINIT:
81                if (CANx_Init(CAN1)) {*State = CAN_CONFIG_FILTERS_MAILBOX;}
82                else {*State = CAN_DISABLED;}
83                break;
84
85            case CAN_CONFIG_FILTERS_MAILBOX:
86        /****** CONFIGURACIÓN DE FILTROS DE LAS FIFOs DE RECEPCIÓN *****/
87                CAN_FilterInit(CAN1, List_mode, Single_32bit, Fifo_0, 0);  // Configura
                      Filtro 0 en modo lista, escala 32 bits, asignado a FIFO 0
88                CAN_SetFilterValue(CAN1, 0x7E8, 0x00, 0);                  // Establece
```

```
                filtro 0 para recibir mensajes con ID 0x7E8 (respuesta ECU)
89              CLEAR_BIT(CAN1->FMR, CAN_FMR_FINIT);                    // Sale del
                modo de inicialización de filtros y habilita los filtros
90
91      /***** CONFIGURACIÓN DE FILTROS DE LOS MAILBOX DE TRANSMISIÓN *****/
92              CAN_MailboxConfig(CAN1, false, 0x7DF, false, 0x0);      // Configura
                el Mailbox de transmisión: ID 0x7DF (broadcast OBDII), estándar
93              CAN_SendData(CAN1, 8, 0x000D0102, 0x0);                 // Envía
                mensaje CAN de 8 bytes al Mailbox 0 con el dato 0x000D0102
94              *State = CAN_ACTIVE;
95              break;
96
97          case CAN_ACTIVE:
98              for (int i = 0; i < 20; i++) {
99                  GPS_PET();                  // Obtener ubicación GNSS
100                 Get_Obdii_Data();           // Solo si hay bus CAN
101                 MPU6050_Read_Ace_Giro();    // IMU
102                 SSD1306_Refresh();          // Actualizar pantalla
103             }
104             *State = GPS_OFF;
105             break;
106
107         case CAN_DISABLED:
108             for (int i = 0; i < 20; i++) {
109                 GPS_PET();                  // GNSS sigue funcionando
110                 MPU6050_Read_Ace_Giro();    // IMU
111                 SSD1306_Refresh();          // Pantalla también se actualiza
112             }
113             *State = GPS_OFF;
114             break;
115
116         case GPS_OFF:
117             delay(200000);
118             sendStringUARTx(USART3, "AT+QGPSEND\r\n"); //Comando AT para apagar GPS
119             delay(200000);
120             *State = SEND_HTTP;
121             break;
122
123         case SEND_HTTP:
124         sendMessage_http_field1(Vel_Km); //Función para actualizar el campo de
                velocidad con HTTP
125         sendMessage_http_field2(RPM_motor); //Función para actualizar el campo de
                RPM con HTTP
126         sendMessage_http_field3(Temp_liquido); //Función para actualizar el campo de
                 Temperatura con HTTP
127         sendMessage_http_field4(Combustible); //Función para actualizar el campo de
                Combustible con HTTP
128         *State = GPS_ACTIVATE;    // Vuelve al ciclo principal
129         break;
130         default:
131             *State = INIT;
132             break;
133     }
134 }
```

16.6.2. Estructura y organización del proyecto

El firmware del sistema fue desarrollado en C utilizando el entorno de desarrollo STM32CubeIDE, la organización del proyecto sigue una estructura modular, separando los archivos en dos carpetas.

La carpeta **Inc/** contiene los archivos de encabezado .h necesarios para la declaración de funciones, estructuras y constantes. Dentro de esta carpeta se encuentran tres tipos principales de archivos:

Archivos CMSIS

- **core_cm4.h, mpu_armv7.h** : Definen el núcleo ARM Cortex-M4 y sus registros internos.
- **cmsis_gcc.h, cmsis_version.h, cmsis_compiler.h** : Configura compilador, vesiones y macros estándares.

Drivers de periféricos

- **CANx.h** : Define las funciones y macros para la configuración y manejo del periférico CAN
- **GPIOx.h** : Define las funciones y macros para la configuración de puertos GPIO
- **I2Cx.h** : Define las funciones y macros para I2C
- **USARTx.h** : Define las funciones y macros de USART
- **MPU6050.h** : Define las funciones y constantes para leer los datos del acelerómetro y giroscopio, así como la definición de las direcciones internas de sus registros.
- **SSD1306.h** : Define los comandos, direcciones y funciones de control para la pantalla OLED SSD1306 mediante I2C

Archivos específicos del microcontrolador

- **stm32l4xx.h, stm32l452xx.h** : Definición de registros específicos del microcontrolador STM32L452
- **System_stm32l4xx.h** : Define funciones de arranque y configuración del sistema

La carpeta **Src/** contiene los archivos fuente .c donde se redacta la lógica de la implementación de las funciones

- **CANx.c** : Manejo del periférico CAN, inicializaron, filtros y transmisión de datos
- **GPIOx.c** : Configuración de entradas y salidas así como asignar funciones alternas a los puertos
- **MPU6050.c** : Lectura de los datos del acelerómetro y giroscopio.
- **NVIC.c** : Funciones para el control de prioridades e interrupciones
- **RCC.c** : Activación de periféricos mediante el reloj del sistema.
- **USARTx.c** : Transmisión y recepción serial.
- **main.c** : Función principal del programa, implementa la máquina de estados del sistema y gestiona la ejecución de las tareas.

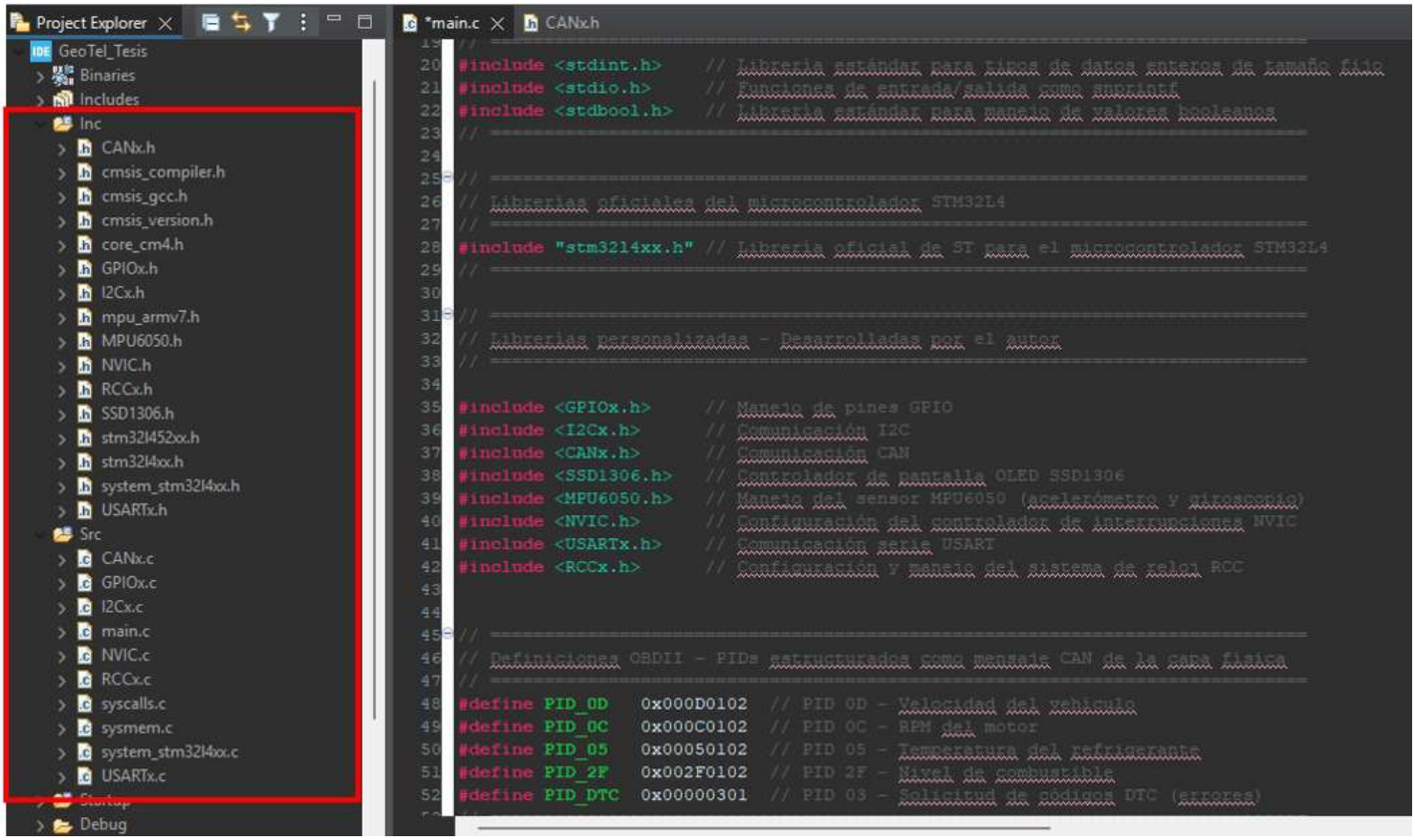


Figura 16.23: Estructura del firmware

En la Figura 16.23 se muestra el entorno de desarrollo utilizado, STM32CubeIDE, junto con la estructura de carpetas que conforman el firmware del proyecto, en el recuadro resaltado en rojo se observan las dos carpetas principales, **Inc/** que contiene los archivos de encabezado .h y **Scr/** que contiene los archivos fuente .c, con la implementación de las funciones.

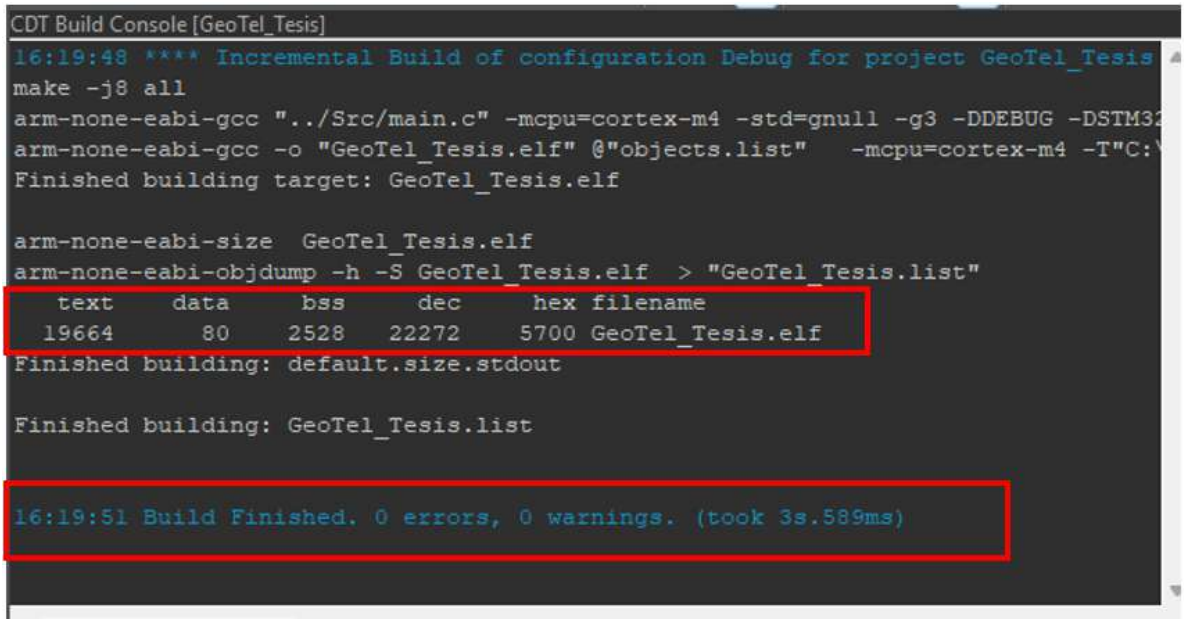


Figura 16.24: Compilación del firmware del proyecto

En la Figura 16.24 se muestran los resultados de la compilación del proyecto en STM32CubeIDE, en el primer recuadro se resalta un resumen del uso de memoria del microcontrolador, donde se indican los recursos utilizados por el programa.

En la primera columna del reporte corresponde a la seccion **.text**, la cual representa el tamaño del código ejecutable que se almacena en la memoria Flash del microcontrolador. Las columnas **.data** y **.bss** corresponden al uso de memoria RAM, finalmente en la Tabla 16.3 se muestra una comparacion entre los recursos utilizados y los recursos disponibles del microcontrolador STM32L452RE, el cual cuenta con 512 kB de memoria Flash y 160 kB de memoria RAM.

Tabla 16.3: Uso de memoria del sistema

Recursos	Utilizado	Total disponible	Porcentaje utilizado
Flash	19,664 B	512 kB	3.84 %
RAM	2,608 B	160 kB	1.63 %

Este análisis permite validar que el firmware desarrollado se encuentra dentro de las capacidades de memoria del microcontrolador STM32L452. El código resultante es muy liviano y utiliza menos del 5 % de la memoria Flash y menos del 2 % de la memoria RAM disponible, dejando mucho espacio libre para agregar más funcionalidades en futuras actualizaciones. Lo demás del código correspondiente el archivo main.c se muestra de forma completa en el Anexo.

17 Construcción

Siguiendo con la metodología de diseño propuesta, se realiza el prototipo de las PCB de cada uno de los módulos que conforman el sistema, empezando por el diseño del diagrama esquemático, seguido de la placa de circuito impreso PCB junto con sus dimensiones y finalmente el modelo 3D. Se utilizó el Software Fusión 360 (Utilizado con licencia educativa UNAM) que cuenta con la capacidad para el diseño electrónico y de PCB, ofreciendo a los desarrolladores las herramientas y librerías de componentes necesarias para facilitar el proceso de diseño, desde el desarrollo del concepto del circuito hasta la visualización de la integración del producto final.

Para la construcción y validación del sistema se propone un diseño modular, donde a la placa PCB del CMI se le pueden conectar las placas de los demás módulos, de esta manera los errores de diseño son encapsulados solamente en la placa donde se originen, facilitando la depuración y mejora del sistema.

17.0.1. Esquemático del módulo CMI

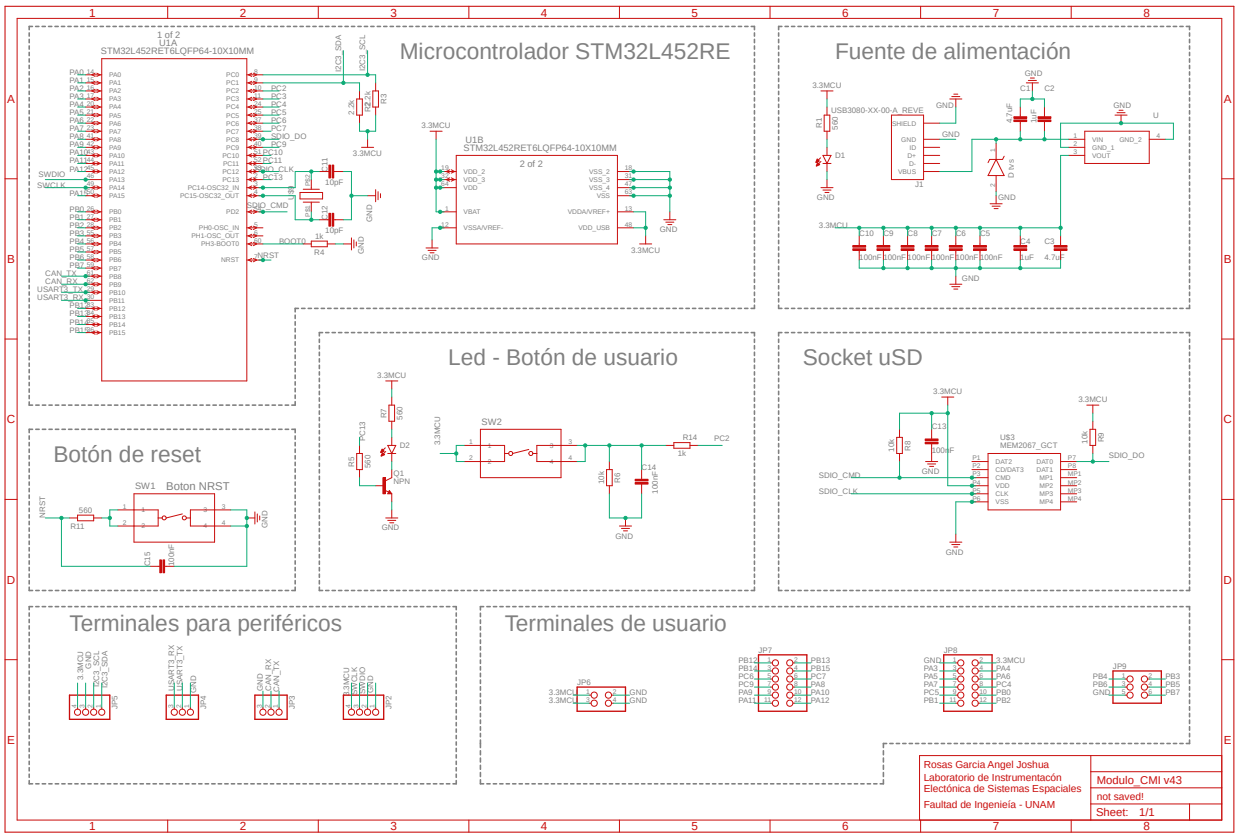


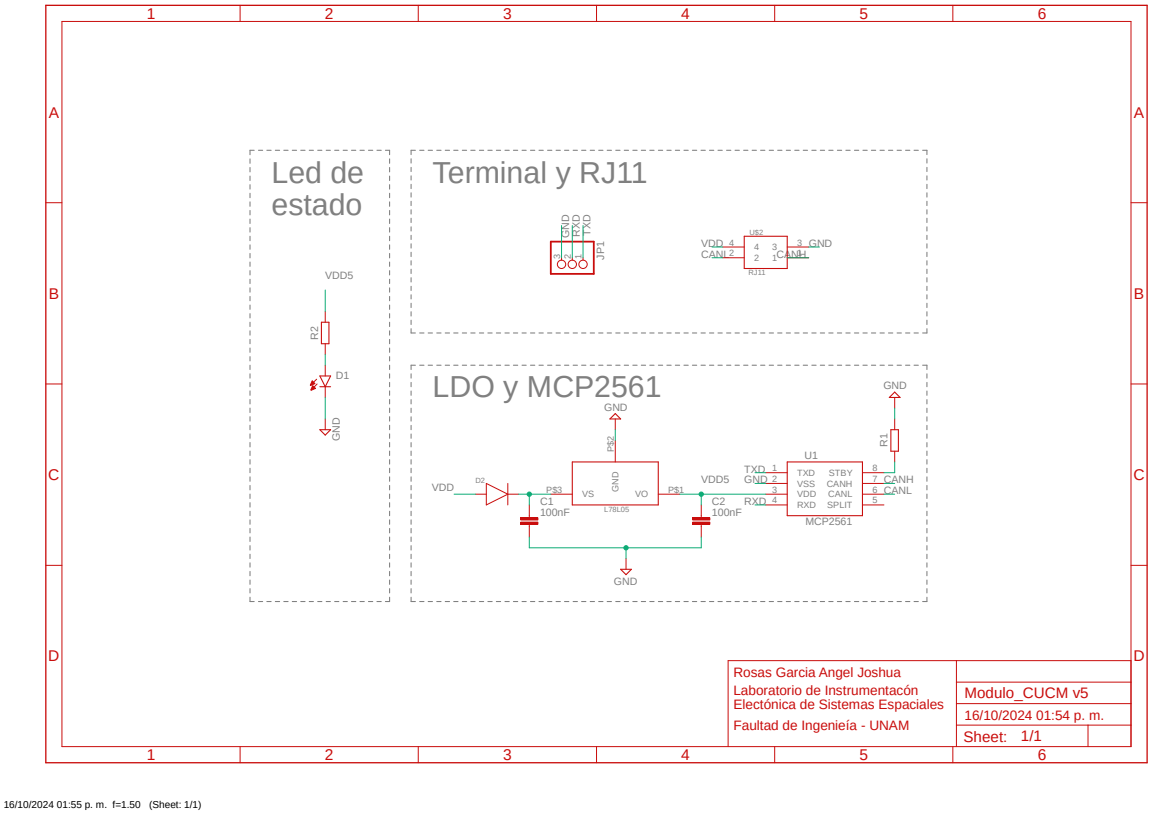
Figura 17.1: Esquemático del módulo CMI

En la Figura 17.1 se muestra el diagrama esquemático correspondiente a la placa PCB del módulo CMI, el microcontrolador es representado en dos bloques, el primero corresponde a los puertos y el segundo a los pines de alimentación, en este apartado también se muestra el cristal de 32.768kHz, resistencias de pull up para la interfaz I2C y una resistencia que asegura un nivel lógico 0 en el pin BOOT0 para que el microcontrolador arranque desde su

memoria flash.

En la fuente de alimentación se muestra el circuito correspondiente a la regulación del voltaje a 3.3V para alimentar al microcontrolador, un led indicador y el conector USB micro-b, de igual manera se muestran el circuito del Botón Reset y del Botón de usuario, así como el Socket uSD. Finalmente se muestran las terminales para los periféricos, en los cuales se utilizan en el sistema y las terminales de los puertos libres que pueden ser configurados por el usurario con base en las necesidades del sistema.

17.0.2. Esquemático del módulo CUCM



16/10/2024 01:55 p. m. f=1.50 (Sheet: 1/1)

Figura 17.2: Esquemático del módulo CUCM

En la Figura 17.2 se muestra el esquemático correspondiente a la placa PCB del módulo CUCM, cuenta con un led de estado, terminales utilizadas para conectarse a la placa del módulo CMI usando la interfaz CAN, un conector RJ11 hembra que recibe un adaptador RJ11 macho con cable UTP en un extremo y un conector macho OBDII de 16 pines genérico al otro extremo para conectarlo al puerto OBDII del vehículo, finalmente se muestra el circuito de regulación de voltaje a 5V del MCP2561 y se agrega una resistencia conectada a GND en el pin 8 para mantenerlo activado.

17.0.3. Esquemático del módulo SV

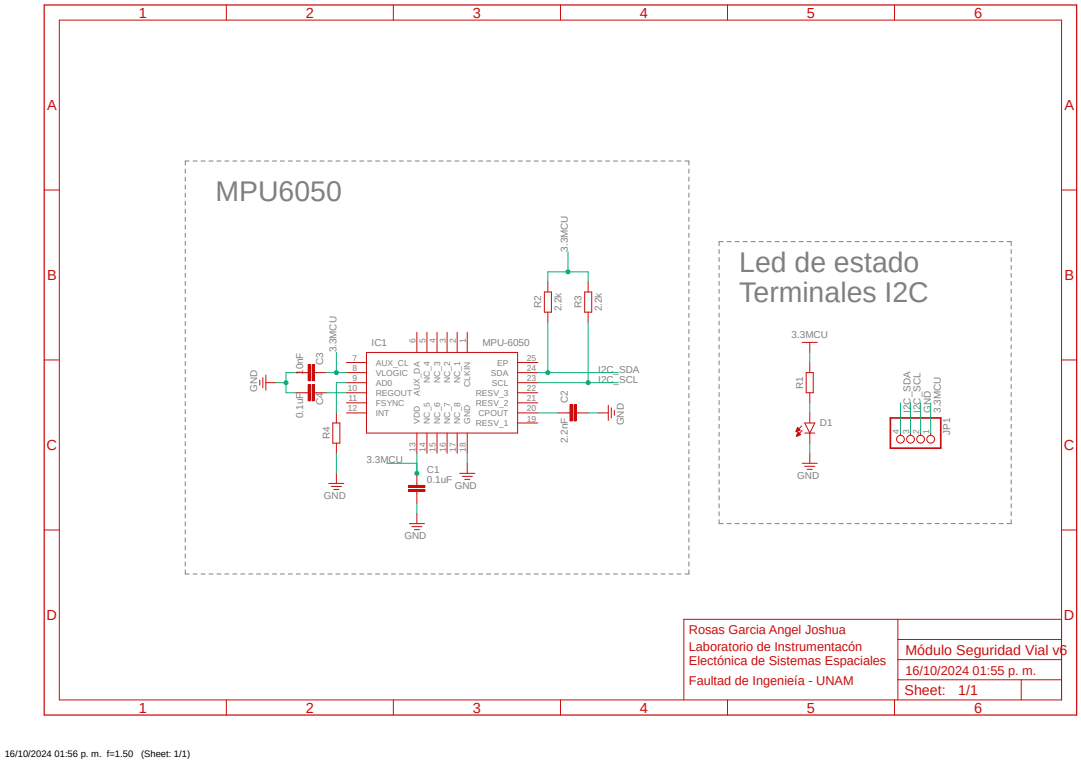


Figura 17.3: Esquemático del módulo SV

En la Figura 17.3 se muestra el esquemático correspondiente a la placa PCB del módulo SV, integra un módulo MPU6050 que cuenta con tecnología MEMS y electrónica CMOS junto con el circuito recomendado por el fabricante necesaria para su funcionamiento, resistencias de pull up en los pines SDA y SCL de la interfaz I2C, así como un led de estado y las terminales para conectarse a la placa PCB del módulo CMI.

17.1. Diseño de la placa de circuito impreso

Siguiendo con la construcción se definen las dimensiones y forma de cada una de las placas PCB de los módulos, serigrafía de ambas capas, así como los empaquetados de los componentes seleccionados. En el diseño de la placa de circuito impreso de los diferentes módulos se implementaron buenas prácticas de diseño de PCB, como colocar capacitores de desacoplo cerca de los pines de alimentación del microcontrolador para minimizar el ruido y estabilizar la línea de alimentación, se definió un plano de tierra (GND) y un plano de alimentación (VCC), evitar rutas con quiebres de 90 grados, colocación estratégica de componentes para minimizar el espacio, selección de componentes con certificación automotriz AEC-Q100, aplicar técnicas en el par diferencial utilizado (CAN-H y CAN-L) para asegurar que ambas líneas sean de la misma longitud.

17.1.1. Circuito impreso PCB del módulo CMI

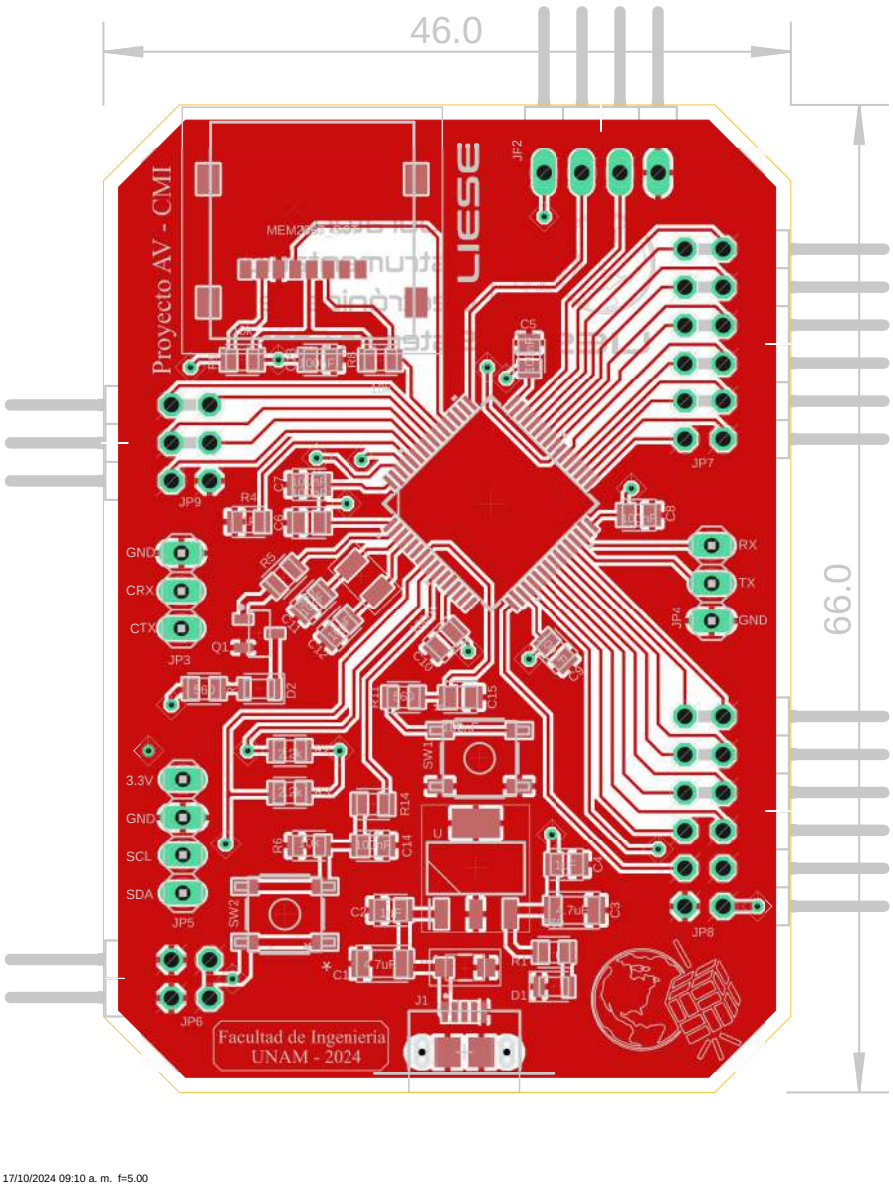


Figura 17.4: Capa Top de la placa de circuito impreso del módulo CMI

En la Figura 17.4 se muestra en color rojo la capa top de la PCB del módulo CMI, así como los pads de cada

uno de los componentes pasivos con las dimensiones específicas de sus empaquetados, en este diseño se utilizaron en su mayoría empaquetados de montaje superficial de 0805 y 1206, en color amarillo se muestra el borde de la placa y en verde los pads de las terminales true hole estándar con un pitch de 2.54 mm así como las vías utilizadas para conectar una ruta de una capa a otra.

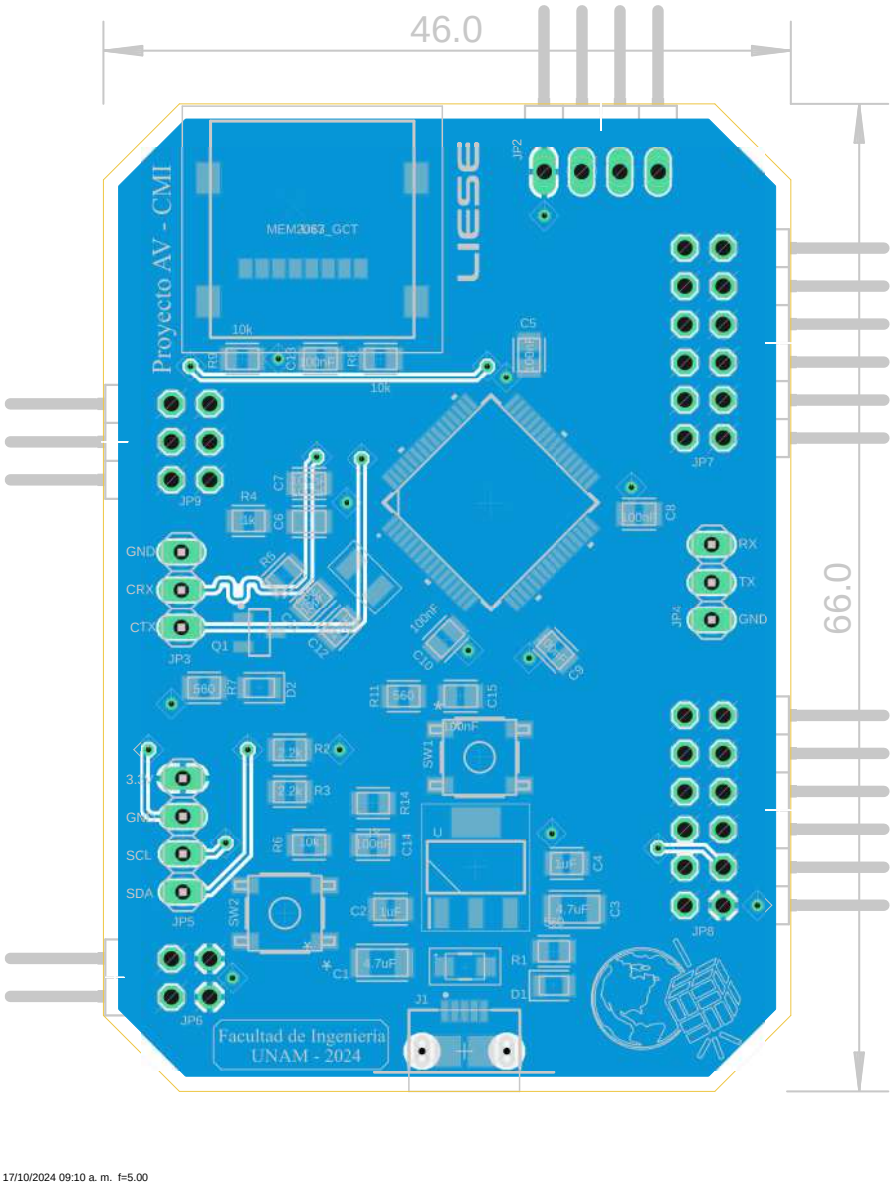
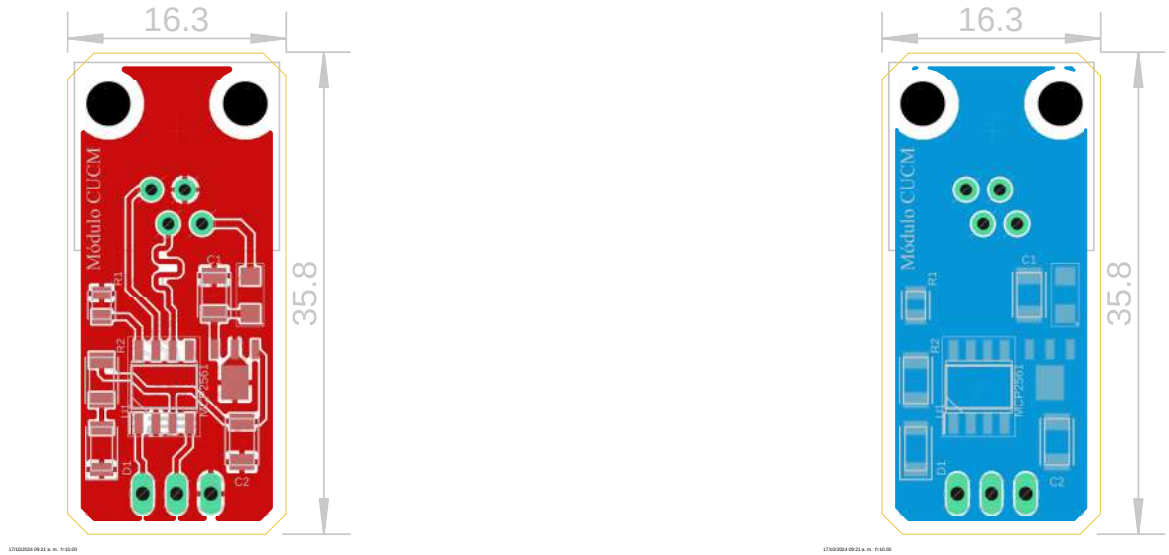


Figura 17.5: Capa Bottom de la placa de circuito impreso del módulo CMI

En la Figura 17.5 se muestra en color azul la capa bottom de la PCB del módulo CMI, así como las rutas que se realizaron sobre esta capa, gracias a la colocación estratégica de los componentes se logró reducir el tamaño hasta obtener unas dimensiones de 46.0 mm x 66.0 mm, de esta manera se logra tener un producto compacto altamente confiable.

17.1.2. Circuito impreso PCB del módulo CUCM



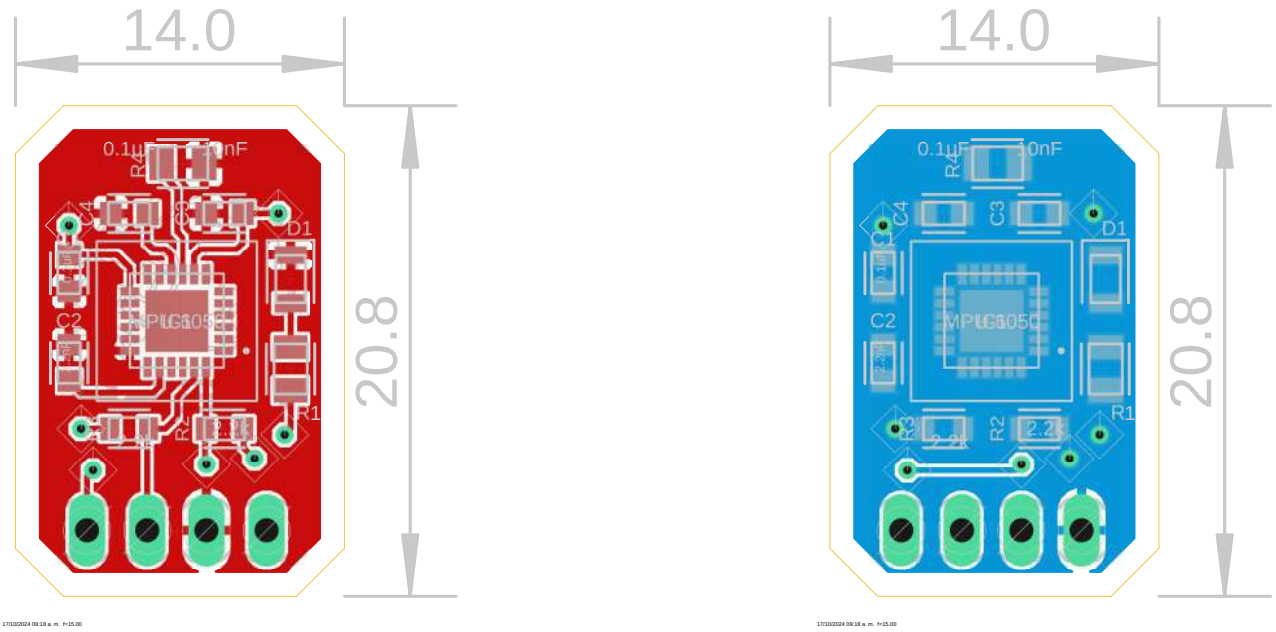
(a) Placa de circuito impreso (Top)

(b) Placa de circuito impreso (Bottom)

Figura 17.6: Capa Top y Bottom módulo CUCM

En la Figura 17.6a se muestra en color rojo la capa superior de la placa PCB del módulo CUCM, en este diseño se utilizaron componentes pasivos con empaquetados 1206 y 0805, empaquetado SOIC para el MCP2561, empaquetado SOT-89 para el LDO, conector RJ11 y terminales estándar con un pitch de 2.54 mm. En la Figura 17.6b se muestra en color azul la capa inferior de la placa PCB, en amarillo el borde, las dimensiones de esta PCB que se consiguieron son de 16.3 mm x 35.8 mm.

17.1.3. Circuito impreso PCB del módulo SV



(a) Placa de circuito impreso (Top)

(b) Placa de circuito impreso (Bottom)

Figura 17.7: Capa Top y Bottom módulo SV

En la Figura 17.7 se muestra en rojo y azul las capas superior e inferior respectivamente de la placa PCB del módulo SV, los componentes pasivos de este diseño son de empaquetados 0805 y 0603 mientras que el MPU6050 es de empaquetado QFN-24, cuenta con terminales estándar con un pitch de 2.54 mm.

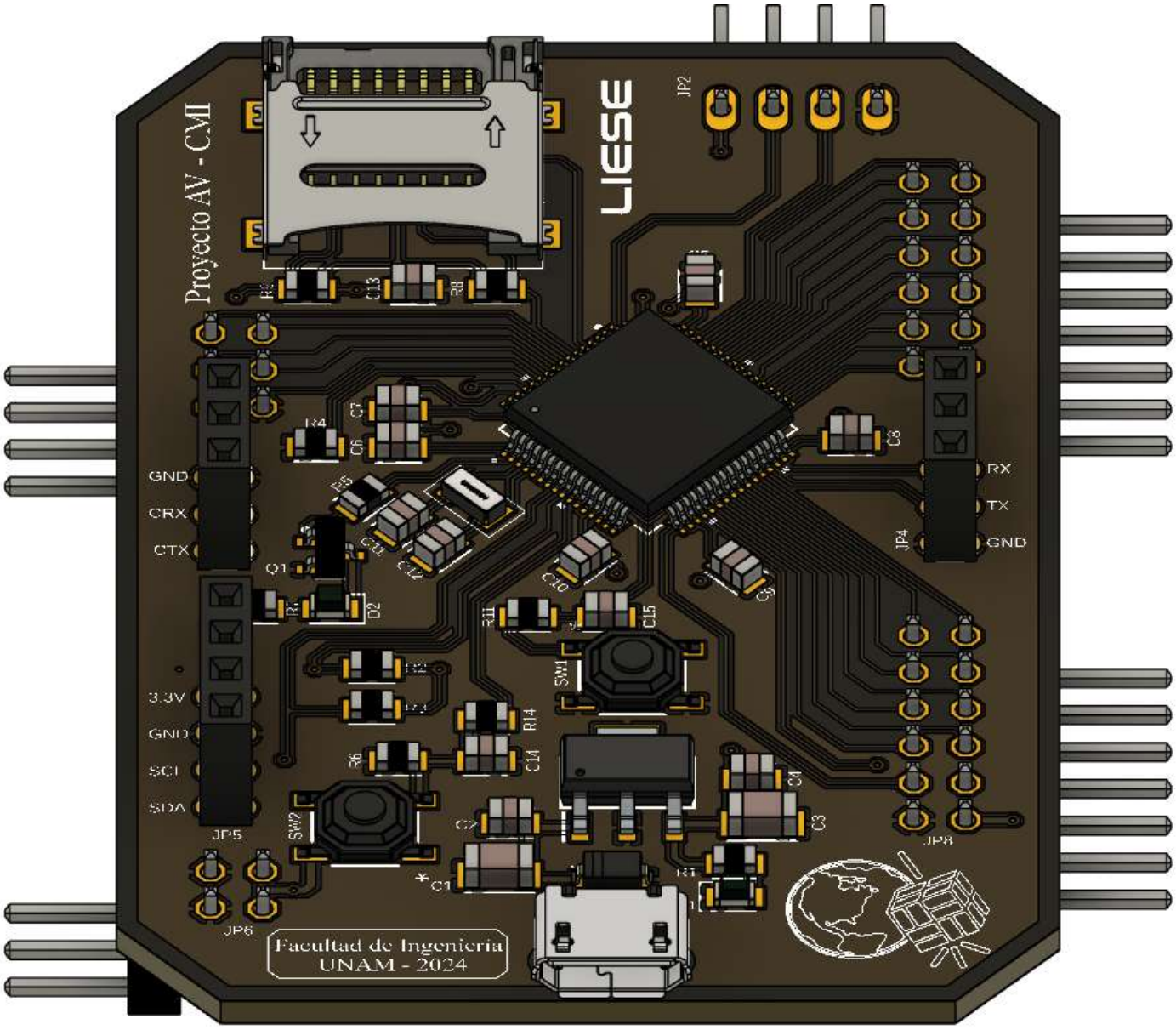
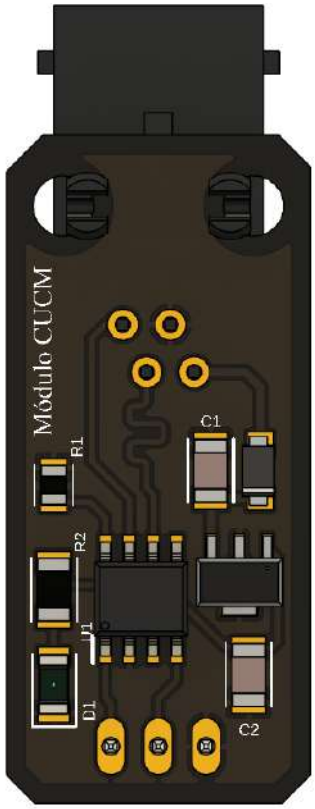


Figura 17.8: Modelo 3D de la PCB del módulo CMI (Top).

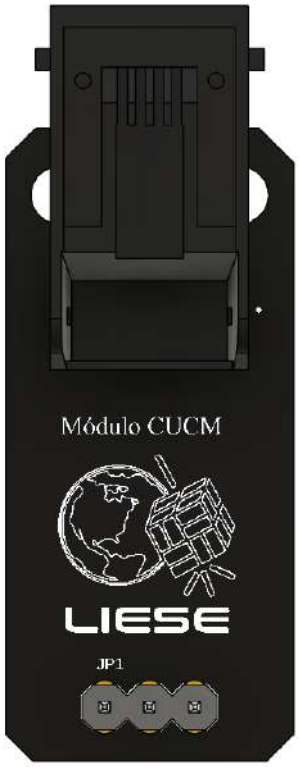
En la Figura 17.8 se muestra la capa superior del modelo 3D de la placa PCB del módulo CMI, donde se aprecia de mejor manera la serigrafía y una representación gráfica fiable en dimensiones y componentes de cómo se vería el producto final.

Figura 17.9: Modelo 3D de la PCB del módulo CMI (Bottom).

En la Figura 17.9 se muestra la capa inferior del modelo 3D de la placa PCB correspondiente al módulo CMI, en esta capa se puede observar la serigrafía del logotipo del LIESE-UNAM, así como información correspondiente a la UNAM.



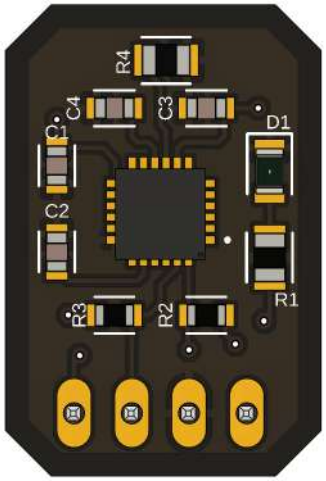
(a) Placa de circuito impreso (Top)



(b) Placa de circuito impreso (Bottom)

Figura 17.10: Modelo 3D de la PCB del módulo CUCM

La Figura 17.10 se muestra la capa superior e inferior del modelo 3D de la PCB del módulo CUCM.



(a) Placa de circuito impreso (Top)

(b) Placa de circuito impreso (Bottom)

Figura 17.11: Modelo 3D de la PCB del módulo SV

En la Figura 17.11 se muestra la capa superior e inferior del modelo 3D de la PCB del módulo SV.

17.1.4. Módulo LPWA BG95-M3 & EVB

Para facilitar las pruebas con el circuito integrado BG95-M3 se utilizó una tarjeta de desarrollo, o Evaluation Board(EVB, por sus siglas en inglés) ya que proporciona los voltajes y corrientes necesarias para su operación, además de facilitar la conexione con las antenas GPS y GSM, este kit de desarrollo propiedad de Quectel y se programa utilizando comandos AT enviados al módulo por medio de la interfaz UART.

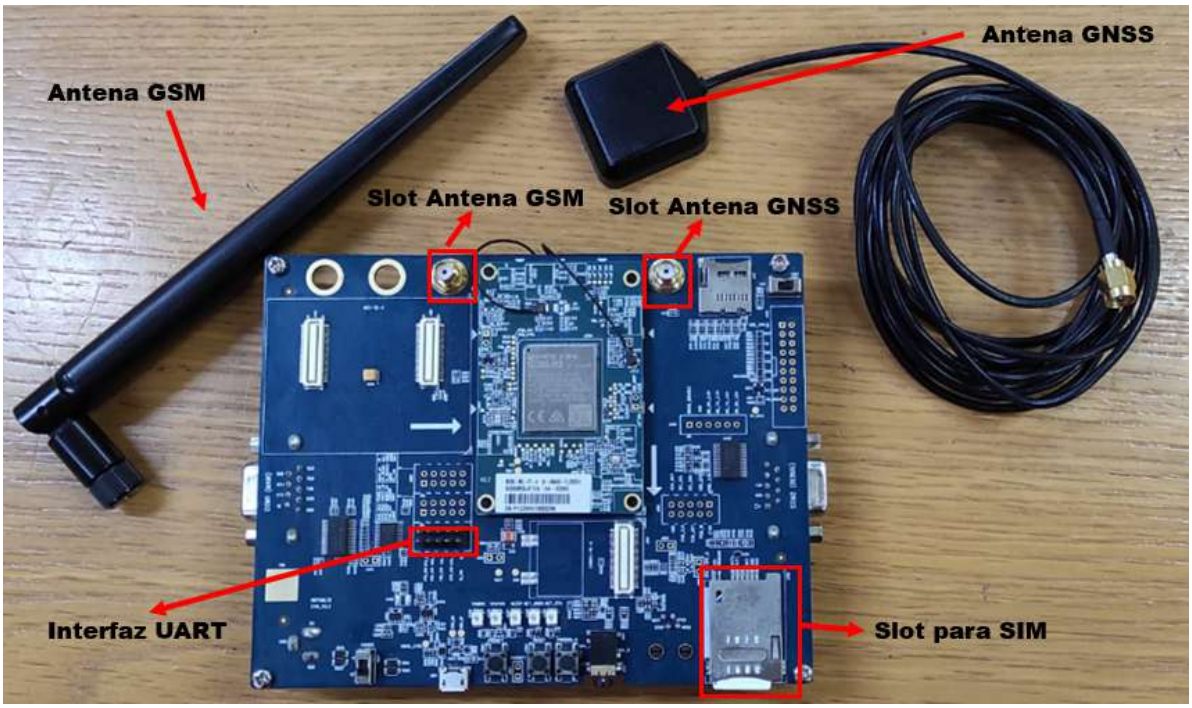


Figura 17.12: EVB LPWA BG95-M3

En la Figura 17.12 se muestra la tarjeta EVB BG95-M3 donde se resalta en rojo los slots para la SIM y las antenas, asi como las terminales de la interfaz UART para programarla.

17.1.5. Integración del sistema

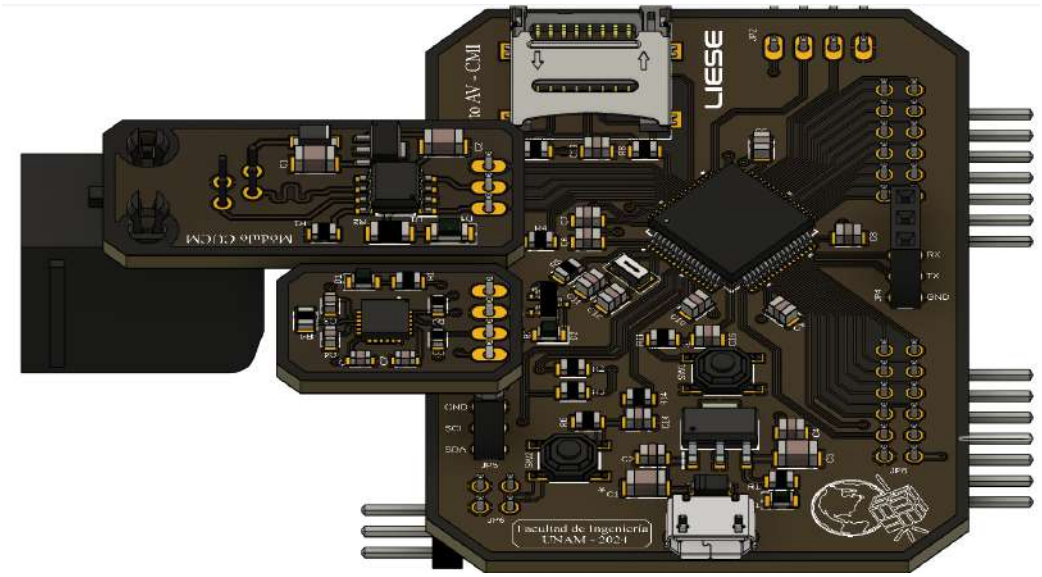


Figura 17.13: Integración del sistema

En la Figura 17.13 se muestra el renderizado 3D de la integración de las placas diseñadas en Fusion 360, las placas PCB de los módulo CUCM y módulo SV se conectan a la placa PCB del módulo CMI y se dejan las terminales libres para conectar el EVB LPWA BG95-M3.

Figura 17.14: Cable RJ11 - OBDII

En la Figura 17.14 se muestra la placa PCB del módulo CUCM con el cable RJ11 - OBDII, utilizado para acceder al puerto OBDII del vehículo.

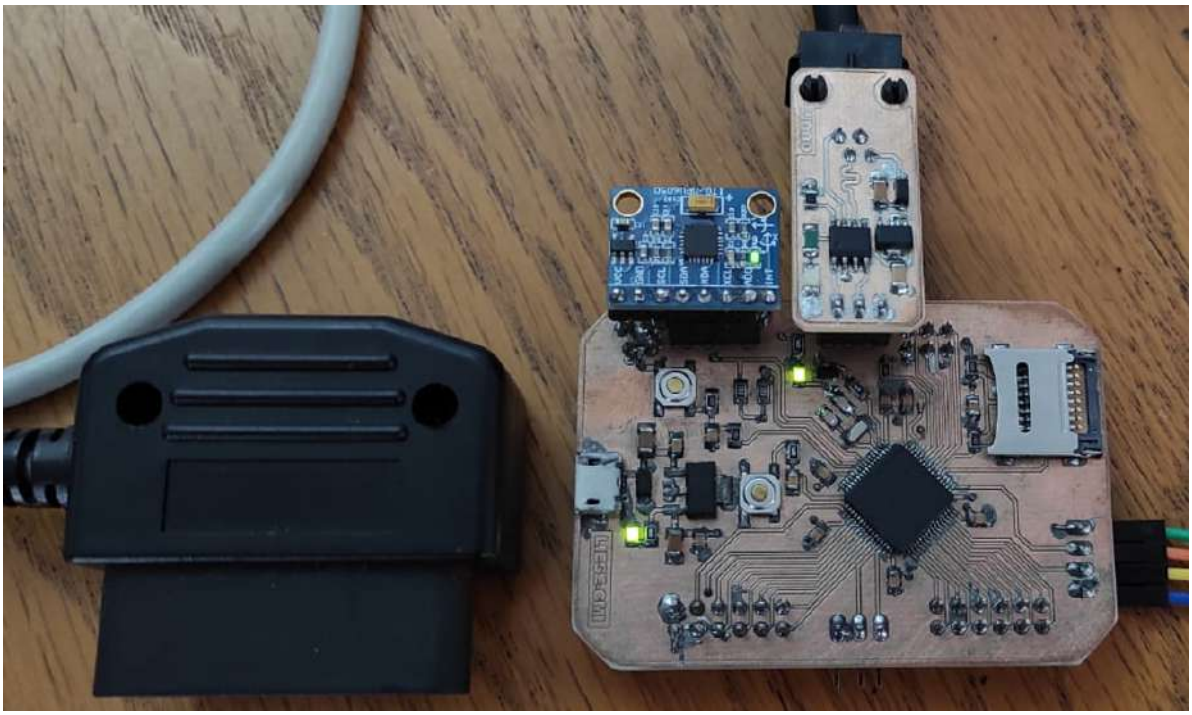


Figura 17.15: Hardware manufacturado, populado y alimentado

En la Figura 17.15 se muestra las placas PCB manufacturadas del sistema con ayuda de una maquina CNC de la marca LPKF modelo E44 y se usaron placas fenólicas FR4 como material de las PCB. La PCB del módulo CUCM se encuentra conectado a la PCB del módulo CMI. La placa PCB seleccionada para el módulo SV fue un módulo comercial que ya tiene embebido el giroscopio y acelerómetro MPU6050, se decidió por este módulo comercial ya que la tecnología de manufactura con la que se contaba no hizo posible soldar el empaquetado de dicho sensor.

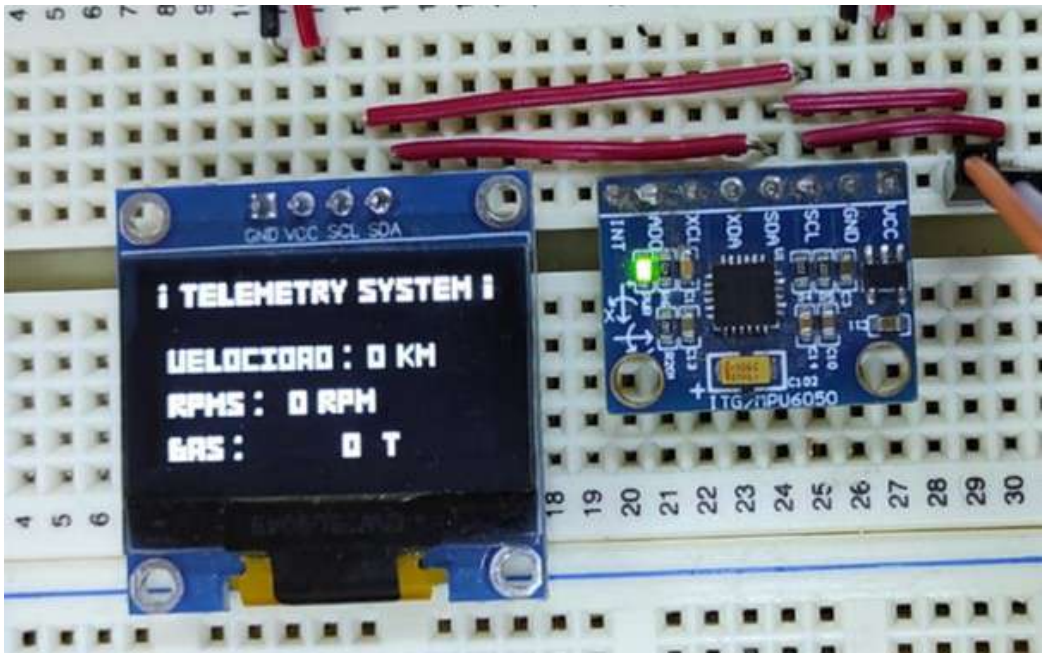


Figura 17.16: Interfaz de usuario del sistema

En la Figura 17.16 se muestra la pantalla utilizada como interfaz visual, con el propósito de validar el funcionamiento del firmware y hardware durante las pruebas.

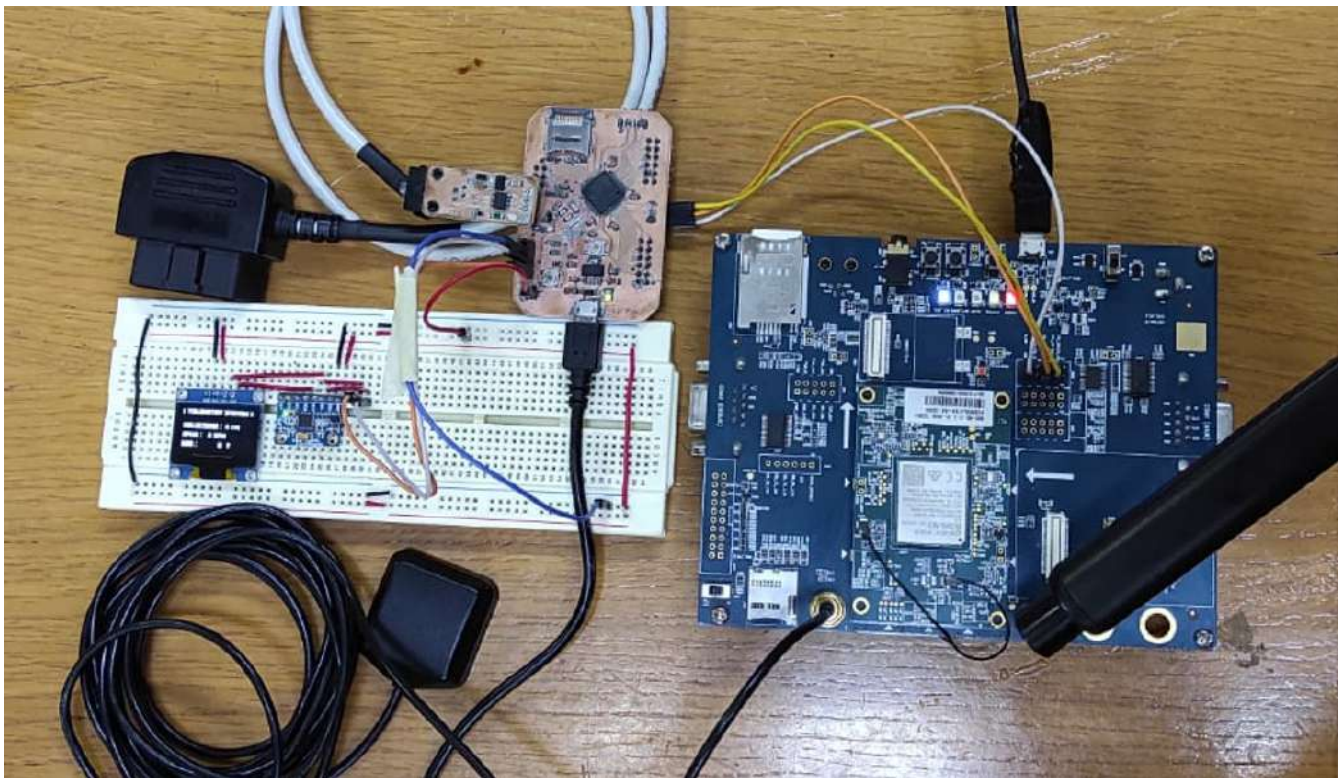


Figura 17.17: Hardware manufacturado e integrado

Finalmente, en la Figura 17.17 se muestra la integración del sistema utilizada durante la validación del diseño.

18 Pruebas y refinamiento

En este capítulo se describen las pruebas que se realizaron para comprobar el funcionamiento del Hardware y Firmware del sistema.

18.1. Prueba No 1

La prueba No1 se utilizó para verificar que la manufactura de la PCB del módulo CMI se realizó correctamente y que esta se puede programar.

- Verificar con un multímetro en modo de continuidad que no existan cortocircuitos entre las pistas de señal y el plano de tierra en la PCB ensamblada.
- Conectar en la entrada micro USB de la PCB del CMI a una fuente de alimentación de 5v y comprobar que el led1 se encienda.

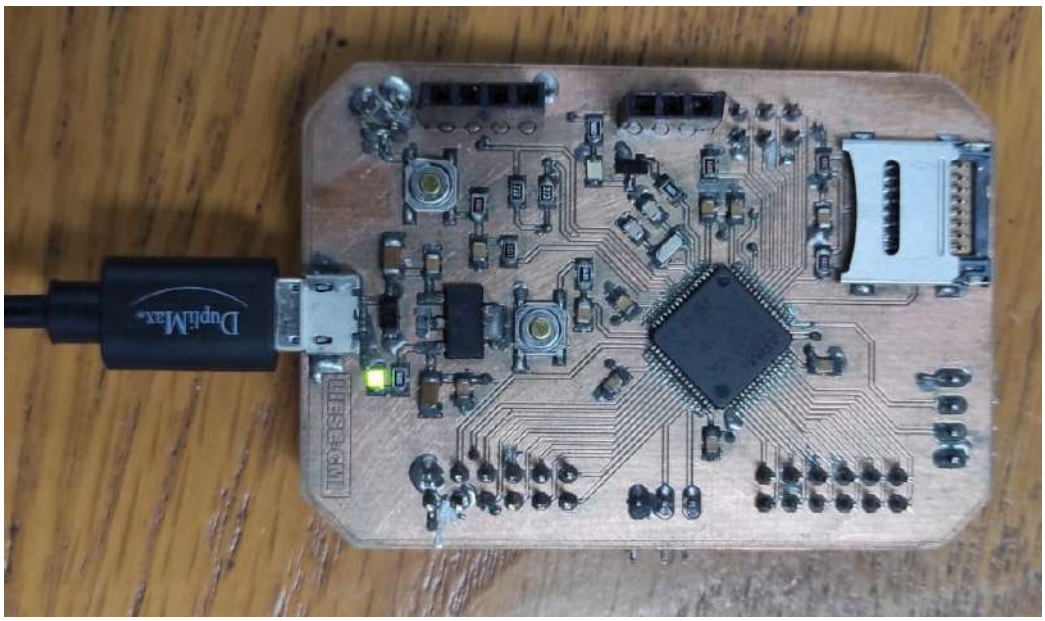


Figura 18.1: Pruebas de alimentación

- Verificar los niveles de voltaje de los pines VDD.
- Conectar la herramienta STLINK a los pines correspondientes de protocolo SWD y verificar la información de la tarjeta en el Software STM_Programer.
- Programar el microcontrolador STM32L452RE para verificar el funcionamiento del botón y led de usuario, que están embebidos en la PCB.

  En la Figura 18.3 se muestra la consola de registro donde se proporciona información del estado de programación, diagnóstico de errores y seguimiento del proceso.

  En la Figura 18.4 se muestra el led de usuario encendido, comprobando que el código y el hardware funcionan correctamente.

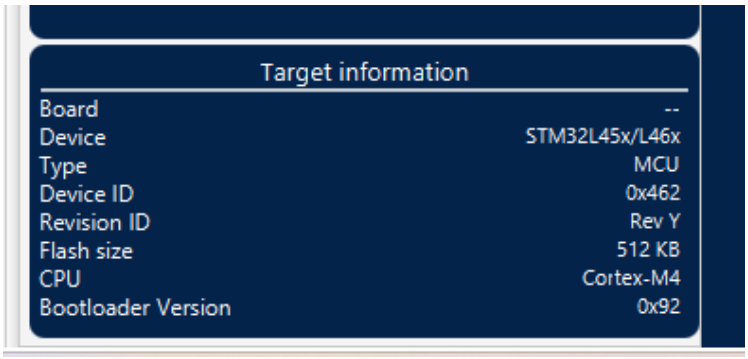


Figura 18.2: Información de la tarjeta STM32L452RE

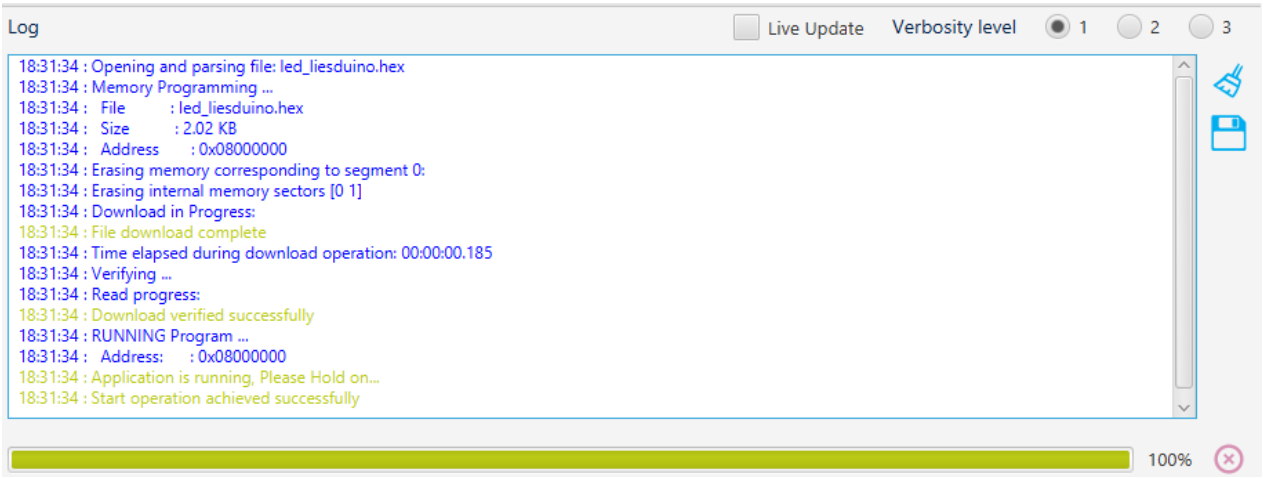


Figura 18.3: Consola de registro de STM Programer

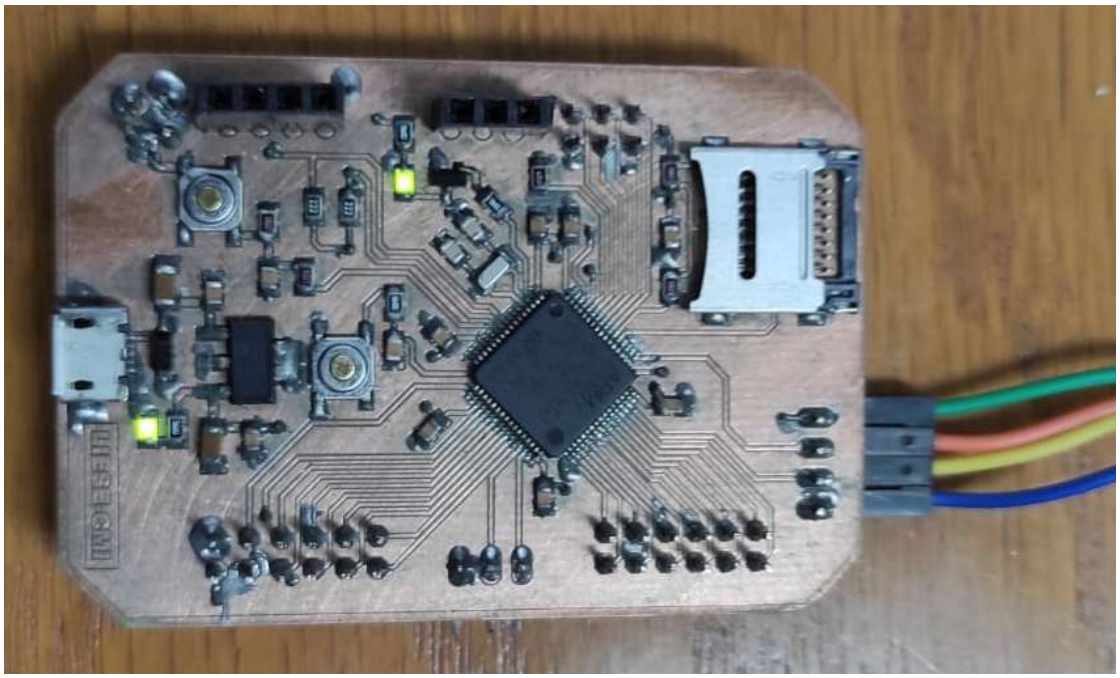


Figura 18.4: Led de usuario encendido tras presionar el botón de usuario

18.2. Prueba No 2

La Prueba No 2 se utilizó para verificar la inicializaron y configuración de los periféricos que se utilizaron en el sistema.

- Programar la PCB para generar las señales del protocolo I2C de la secuencia de escritura del módulo MPU6050.

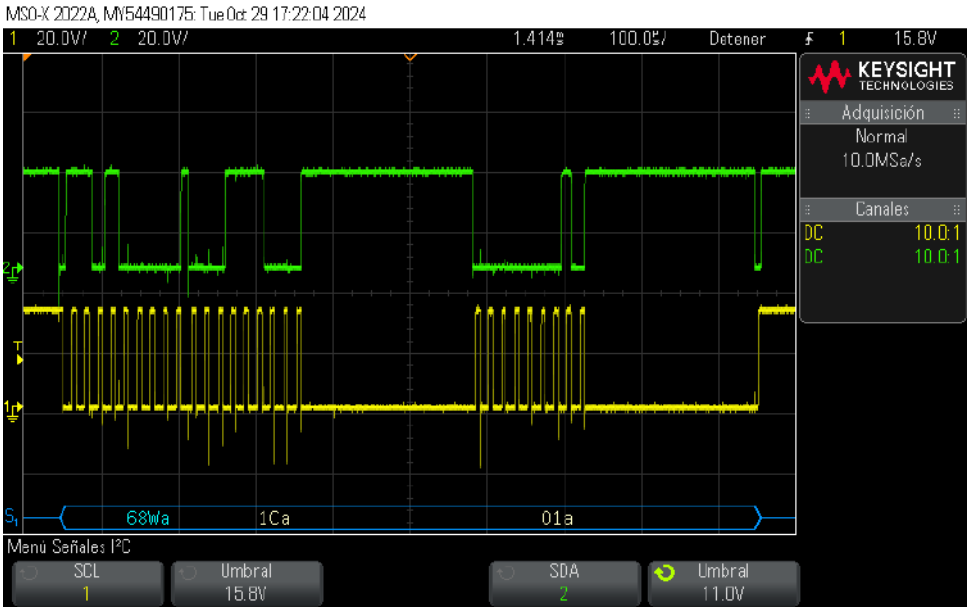


Figura 18.5: Trama I2C para configurar el acelerómetro

- Programar la PCB para generar las señales del protocolo I2C de la secuencia de lectura del módulo MPU6050.

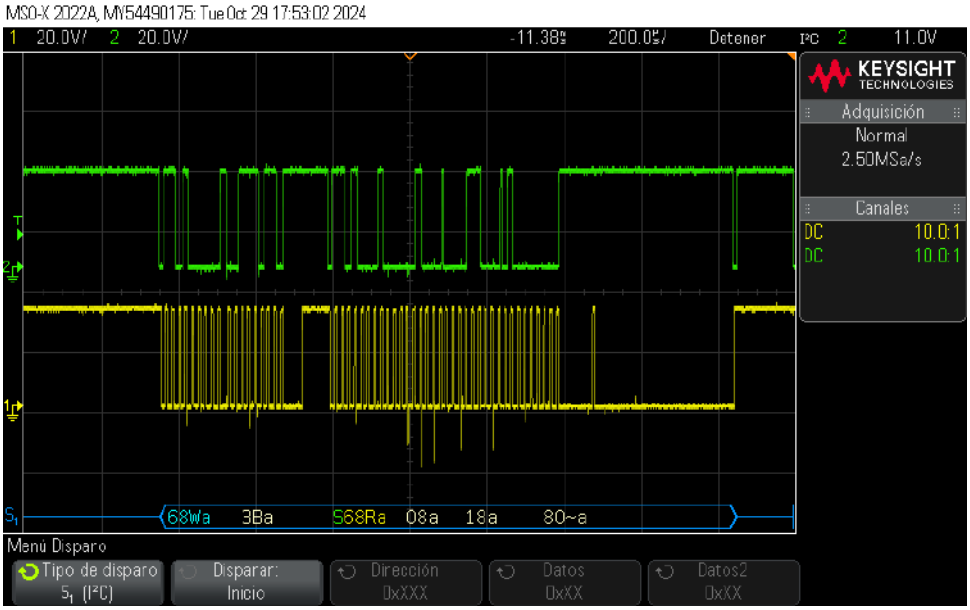


Figura 18.6: Secuencia de lectura establecida por el fabricante

- Programar la PCB para generar las señales del protocolo UART de la cadena de caracteres correspondiente al comando AT que enciende el GPS del módulo GC.

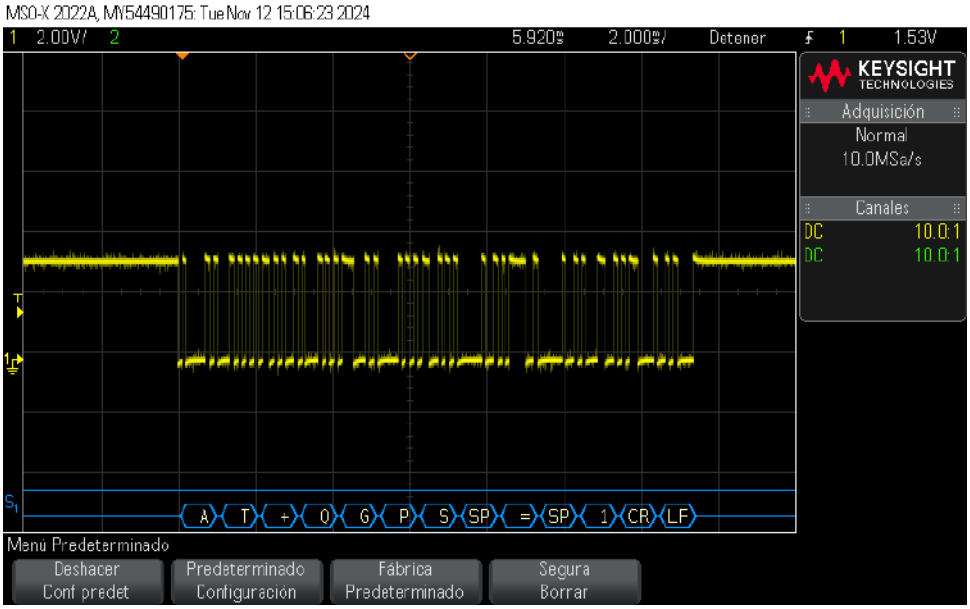


Figura 18.7: Comando AT para encender el GPS

- Programar la PCB para generar las señales del protocolo CAN del mensaje OBDII para obtener la velocidad del vehículo y verificar que el módulo CUCM genere los bits diferenciales propios del protocolo.

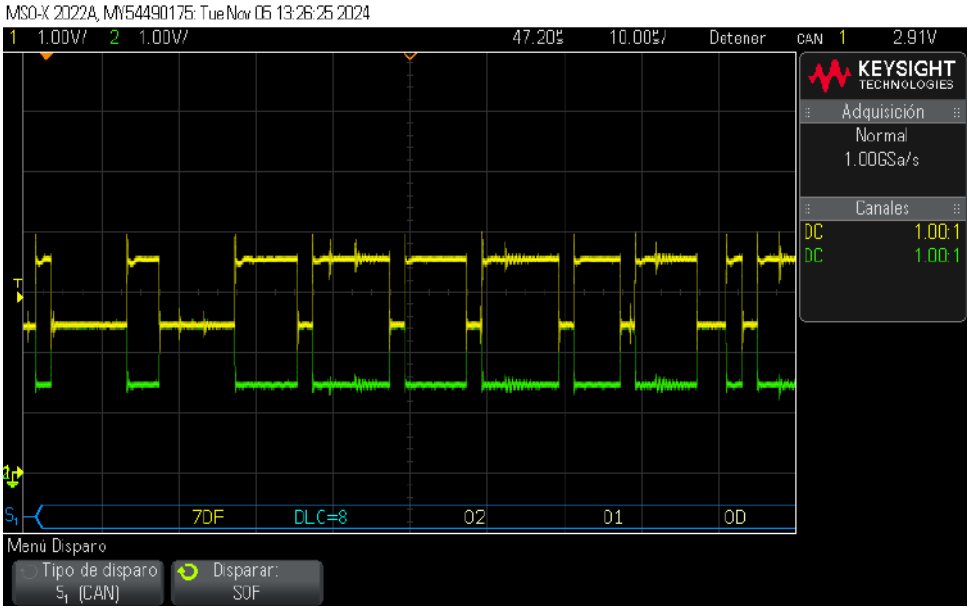


Figura 18.8: Primera parte del mensaje de petición de la velocidad

- Verificar que la velocidad de transmisión sea de 500Kbits por segundo

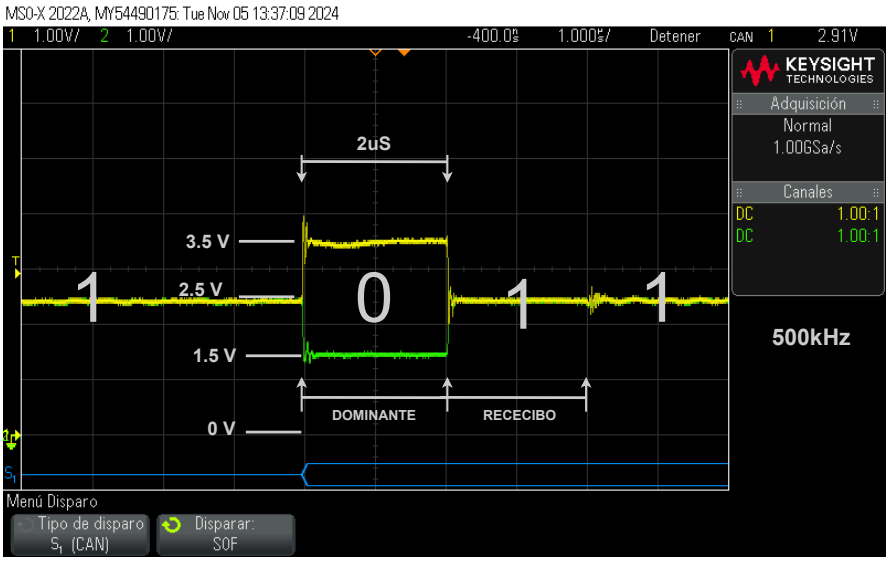


Figura 18.9: Codificación de bits generados por STM32L452RE (Dominante/Recesivo)

18.3. Prueba No 3

La prueba No 3 se utilizó para comprobar la conexione con los satelites GPS para obtener la geolocalización

- Implementar en la PCB la lógica necesaria para enviar a través de UART el comando AT encargado de solicitar le geolocalización del módulo BG95-M3

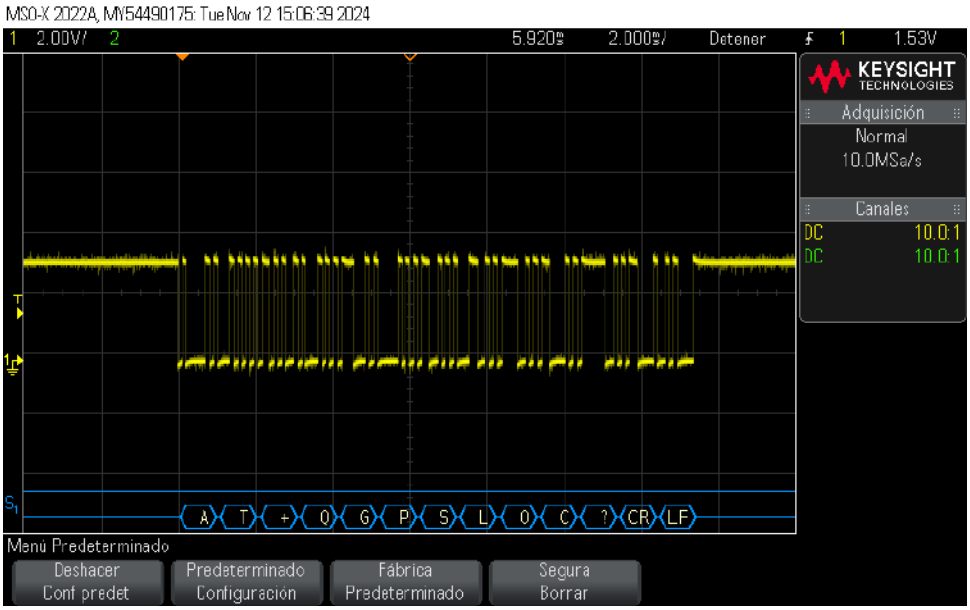


Figura 18.10: Trama UART para solicitar la geolocalización

- Guardar los datos geolocalización en un arreglo previamente definido en el firmaware para guardar las coordenadas.

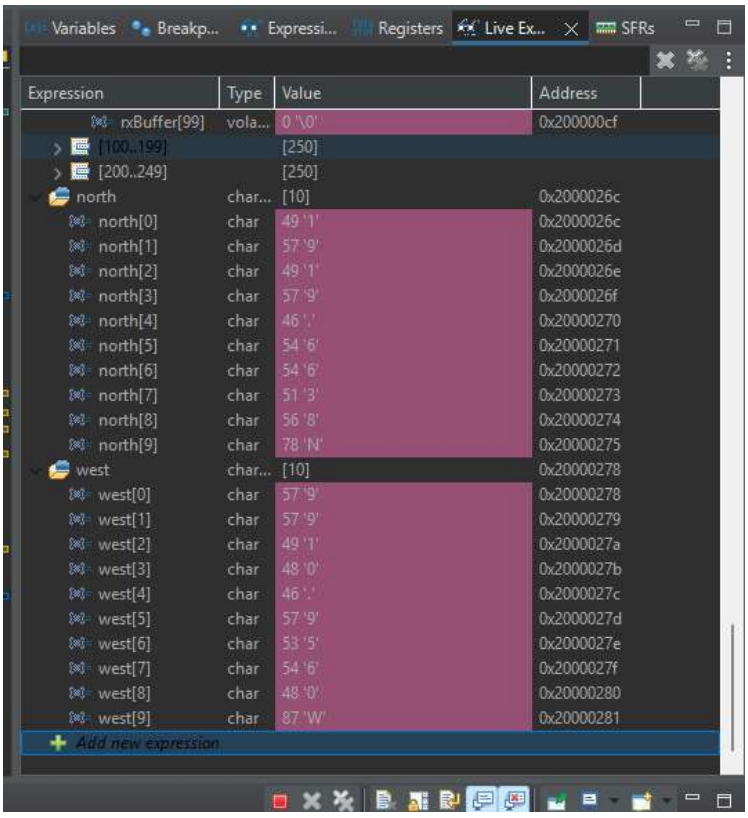


Figura 18.11: Datos de geolocalización guardados

- Interprepar los datos obtenidos en el formato de la Asociación Nacional de Electrónica Marina (NMEA, por sus siglas en inglés) a formato decimal para poder ingresar las coordenadas a un software de navegación (Google Maps).

  El primer dato corresponde a la latitud: $1919{,}6638N$ que corresponde a 19 grados con 19,6638 minutos, por lo tanto la conversion a formato decimal seria:

$$Latitud = 19 + \frac{19{,}6638}{60} = 19 + 0{,}32773 = 19{,}3277^{\circ}N$$

  El segundo dato corresponde a la longitud: $9910{,}9560W$ que corresponde a 99 grados con 10,9560 minutos, por lo tanto la conversion a formato decimal seria:

$$Longitud = 99 + \frac{10{,}9560}{60} = 99 + 0{,}1826 = -99{,}1826^{\circ}N$$

  se usa el signo negativo ya que se trata del hemisferio oeste.

- Ingresa los datos de geolocalización en un software de navegación para validar la ubicación obtenida.

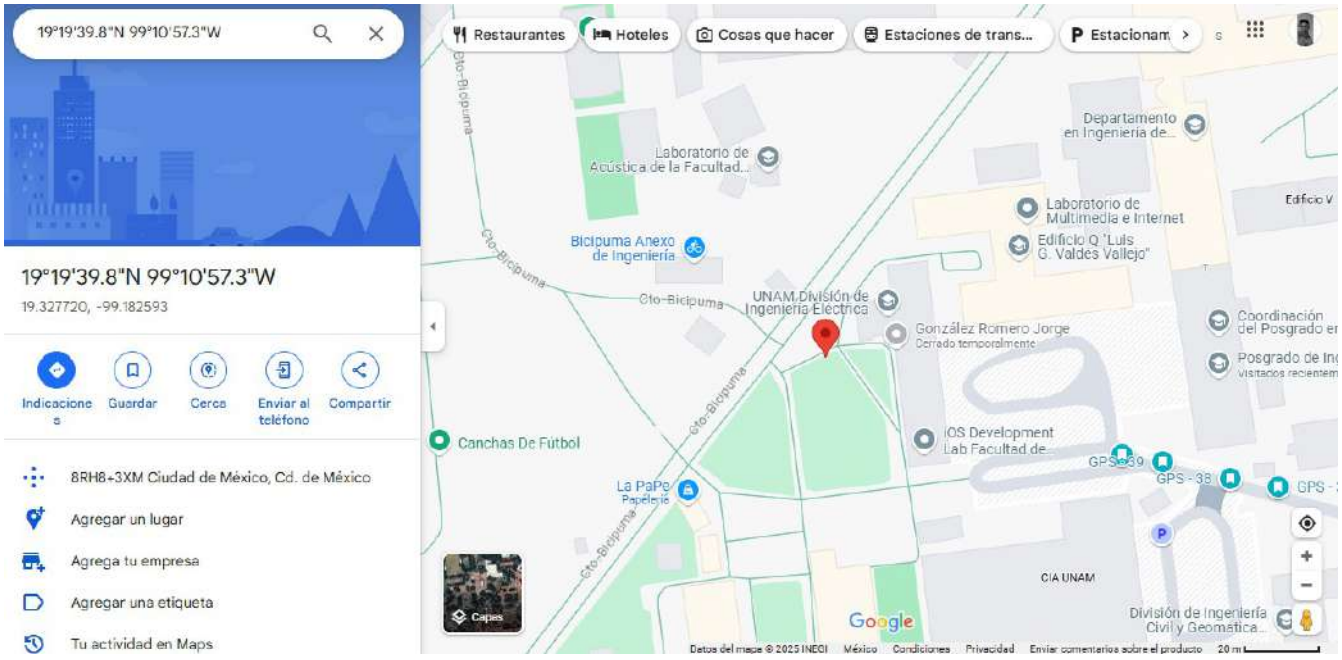


Figura 18.12: Geolocalización en Google Maps

Parte V

Resultados y conclusiones

19 Resultados

Como último paso del presente trabajo, se analiza el comportamiento del sistema bajo condiciones reales de operación, abordo de un vehículo. En este capítulo se muestran los resultados de las pruebas realizadas considerando un viaje común de un vehículo.

19.1. Plataforma de pruebas

Thingspeak es una plataforma basada en la nube que permite agregar, visualizar y analizar flujo de datos en vivo, se pueden enviar datos a Thingspeak desde dispositivos IoT conectados a Internet.

El Hardware y Firmware del dispositivo manufacturado realiza la actualización de los datos, los datos son mostrados en interfaz gráfica de un velocímetro y en una gráfica donde se guardan y visualizan cada una de las actualizaciones que se detecten en la plataforma.

En la Figura 19.1 se muestran los datos obtenidos de velocidad de un viaje de aproximadamente 20 minutos realizado en el circuito exterior de Ciudad Universitaria, en la gráfica se pueden visualizar cada uno de los datos guardados que corresponden a una conducción convencional donde los primeros datos registrados son de una velocidad de 0 km para después registrar aumentos, disminución y mantenimiento de la velocidad para finalizar nuevamente en 0 km que corresponde a cuando el automóvil termino el recorrido y esperamos estacionados para finalizar la obtención de los datos.

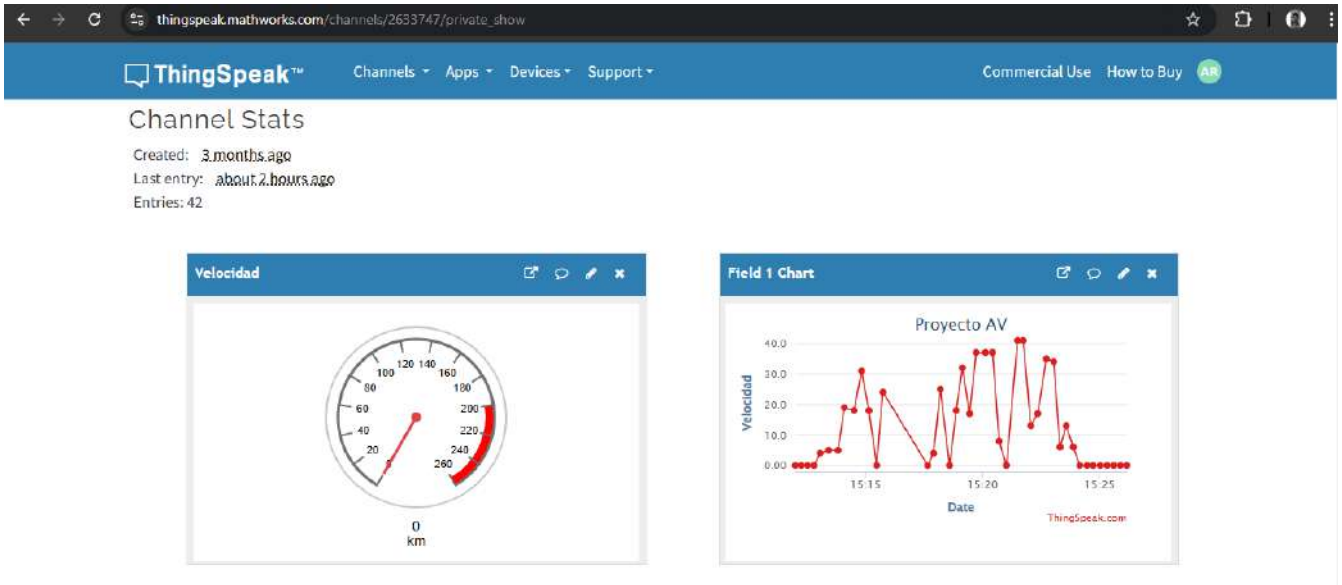


Figura 19.1: Plataforma de pruebas ThingSpeaK mostrando datos de Velocidad

La información mostrada en la página de pruebas está limitada a una tasa de refresco de 15 segundos, propia de ThingSpeak, pero funciona como prueba de concepto, para una solución más robusta se debe proponer la infraestructura de un servidor junto con desarrollo propio de Frontend y Backend para ofrecer una solución con presencia en Internet.

19.2. Mapeo de la geolocalización

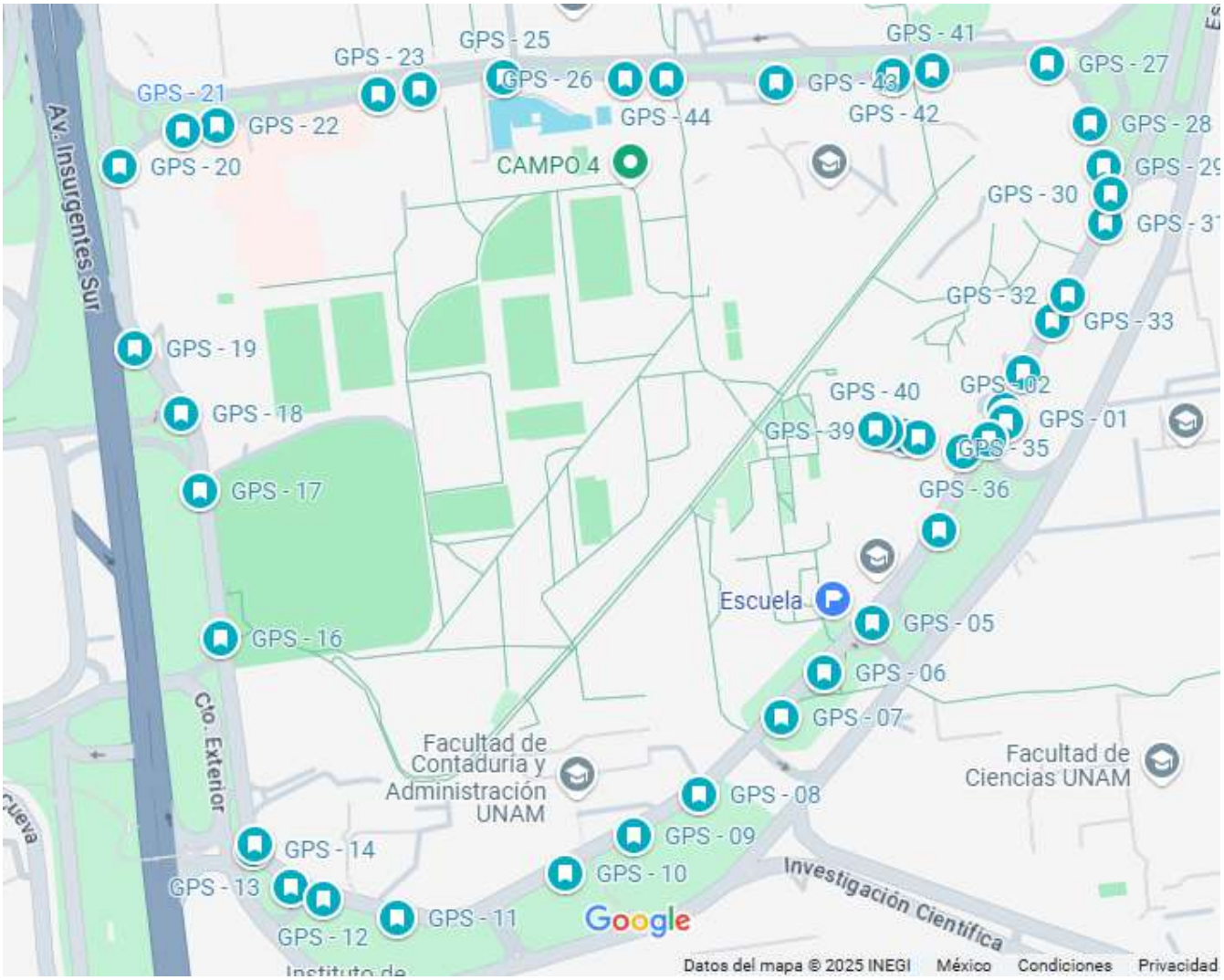


Figura 19.2: Mapeo de recorrido de prueba

Finalmente en la Figura 19.2 se muestra el mapeo de las coordenadas geográficas que se obtuvieron durante el recorrido antes mencionado, cada etiqueta corresponde a unas coordenadas interpretadas de tal manera que puedan ser visualizadas en Google Maps.

20 Conclusiones

El prototipo del sistema logró establecer una comunicación estable con módulos comerciales GNSS y GSM/GPRS, validando su capacidad para la transmisión de datos de geolocalización y telemetría a Internet. El sistema utilizó GPS como sistema GNSS, y el protocolo de comunicación HTTP para la transferencia de datos a Internet. La utilización de componentes de montaje superficial fue clave para la miniaturización de la electrónica del sistema, permitiendo reducir el tamaño del hardware sin comprometer su funcionalidad, lo que facilita su integración en aplicaciones en espacios reducidos.

El firmware fue desarrollado con una arquitectura modular y en capas, lo que favorece su mantenimiento y escalabilidad. La primera capa comprende los drivers propios, diseñados para inicializar y configurar los periféricos empleados en el sistema. La segunda capa corresponde a la capa de aplicación, donde se implementaron funciones específicas que hacen uso de los drives. Para la gestión eficiente del hardware, se utilizó el mapeo de registros de memoria basado en el Estándar de Interfaz de Software del Microcontrolador Cortex (CMSIS, por sus siglas en ingles) de STMicroelectronics, optimizando el control y manejo del microcontrolador.

El sistema fue validado mediante pruebas de campo en un Toyota Corolla, conectándose directamente al puerto OBDII del vehículo, durante un recorrido por el circuito exterior de Ciudad Universitaria, se recopilaron datos en tiempo real, verificando la funcionalidad y estabilidad de la comunicación. Se confirmo la capacidad del prototipo para leer información del vehículo y enviarla a la plataforma de monitoreo en línea, confirmando su viabilidad como una herramienta para la gestión de flotas vehiculares o monitoreo de activos móviles.

Cabe destacar que este dispositivo es complemente escalable y adaptable a diferentes aplicaciones que requieran la integración de sistemas de monitoreo, diagnóstico y geolocalización en vehículos o activos móviles. En conclusión, este trabajo sienta las bases para el desarrollo de un sistema embebido funcional y versátil en el campo del IoT vehicular, con un alto potencial de evolución y aplicaciones en entornos comerciales y de investigación.

El diseño de este prototipo representa el primer paso hacia una solución más robusta en el ámbito de la geolocalización, telemetría y diagnóstico vehicular, con el potencial de evolucionar hasta convertirse en un producto que pueda participar en este mercado. La implementación del hardware del CMI demostró su correcto funcionamiento, dejando una base sólida para la escalabilidad del sistema en versiones futuras, donde se realice la integración total del sistema en una sola PCB.

21 Trabajo a futuro

El desarrollo de este sistema validó la prueba de concepto, pero se abre la posibilidad de mejoras que pueden abordarse en futuras versiones del dispositivo. Entre las principales tareas del trabajo a futuro se identifican los siguientes:

- **Optimización del sistema de potencia:** si bien el sistema de alimentación fue planteado en la fase de diseño, su implementación no se llevó a cabo. Futuras versiones deben integrar un módulo de potencia complemente funcional, que permita la autonomía del dispositivo sin depender exclusivamente de la batería del vehículo.
- **Fabricación del módulo SV:** aunque se diseñó el módulo SV para la adquisición de datos de aceleración y orientación, este no fue fabricado por el tipo de empaquetado del sensor. Su implementación permitirá mejorar la caracterización del comportamiento del vehículo.
- **Incorporación de una memoria externa con SIDO:** la PCB actual cuenta con el slot para una tarjeta SD, pero no se realizó la implementación del sistema de almacenamiento. Agregar una memoria externa permitirá registrar datos en una bitácora local.
- **Caracterización y optimización del algoritmo de análisis de conducción:** las pruebas realizadas fueron enfocadas en la estabilidad y funcionalidad del sistema, sin realizar pruebas exhaustivas que incluyeran manejo irresponsable o condiciones extremas. Es necesario realizar pruebas más robustas que permitan caracterizar con mayor precisión los eventos relacionados con conducción irresponsable.
- **Implementación de un sistema operativo en tiempo real (RTOS):** actualmente, el firmware está diseñado en una arquitectura de capas, integrar un RTOS permitirá mejorar la gestión de las tareas repetitivas y optimizar el uso de los recursos del microcontrolador.

Parte VI

Anexo

22 Código fuente del archivo main.c

```c
    /**
 ******************************************************************************
 * @file           : main.c
 * @author         : Rosas Garcia Angel Joshua
 * @brief          : Firmware para dispositivo de geolocalización y telemetría
 *
 * @details
 * Proyecto de titulación - UNAM (Ingeniería)
 * Este firmware está diseñado para recolectar datos de sensores (MPU6050, GPS),
 * comunicarse por CAN con el vehículo y enviar telemetría por GPRS.
 *
 * 2024 Rosas Garcia Angel Joshua. Todos los derechos reservados.
 * Incluye componentes licenciados por STMicroelectronics.
 ******************************************************************************
 */

// ============================================================================
// Librerías estándar
// ============================================================================
#include <stdint.h>    // Librería estándar para tipos de datos enteros de tamaño fijo
#include <stdio.h>     // Funciones de entrada/salida como snprintf
#include <stdbool.h>   // Librería estándar para manejo de valores booleanos
// ============================================================================

// ============================================================================
// Librerías oficiales del microcontrolador STM32L4
// ============================================================================
#include "stm32l4xx.h" // Librería oficial de ST para el microcontrolador STM32L4
// ============================================================================

// ============================================================================
// Librerías personalizadas - Desarrolladas por el autor
// ============================================================================

#include <GPIOx.h>     // Manejo de pines GPIO
#include <I2Cx.h>      // Comunicación I2C
#include <CANx.h>      // Comunicación CAN
#include <SSD1306.h>   // Controlador de pantalla OLED SSD1306
#include <MPU6050.h>   // Manejo del sensor MPU6050 (acelerómetro y giroscopio)
#include <NVIC.h>      // Configuración del controlador de interrupciones NVIC
#include <USARTx.h>    // Comunicación serie USART
#include <RCCx.h>      // Configuración y manejo del sistema de reloj RCC


// ============================================================================
// Definiciones OBDII - PIDs estructurados como mensaje CAN de la capa física
// ============================================================================
#define PID_0D   0x000D0102  // PID 0D - Velocidad del vehículo
#define PID_0C   0x000C0102  // PID 0C - RPM del motor
#define PID_05   0x00050102  // PID 05 - Temperatura del refrigerante
#define PID_2F   0x002F0102  // PID 2F - Nivel de combustible
```

```
52 #define PID_DTC  0x00000301  // PID 03 - Solicitud de códigos DTC (errores)
53 // ============================================================================
54
55 /*
56  * COMANDOS OBDII PID - Descripción de cada parámetro
57  * ---------------------------------------------------
58  * 0D -> Velocidad del vehículo
59  * 0C -> RPM del motor
60  * 05 -> Temperatura del líquido de enfriamiento
61  * 2F -> Nivel de combustible
62  * 03 -> Solicitud de códigos DTC (códigos de falla)
63  *
64  */
65 /*********DECLARACION DE LAS FUNCIONES DEL MAIN PRINCIPAL***********/
66 typedef enum {
67     INIT,                          // Estado inicial del sistema
68     CONFIG_PERIPH,                 // Configura puertos y periféricos
69     I2C_INIT,                      // Inicializa el bus I2C (MPU6050 y OLED)
70     OLED_STARTUP,                  // Muestra secuencia de inicio en OLED
71     USART_INIT,                    // Configura USART3 para comunicación AT
72     MPU_INIT,                      // Inicializa el sensor MPU6050
73     CAN_INIT,                      // Intenta iniciar el bus CAN
74     CAN_NO_BUS,                    // CAN falló en iniciar (sin bus)
75     CAN_BUS_OK,                    // CAN inició correctamente
76     LOAD_UI,                       // Carga la interfaz gráfica en OLED
77     GPS_ACTIVATE,                  // Activa el GPS mediante comando AT
78     CAN_REINIT,                    // Reintenta inicializar el CAN
79     CAN_CONFIG_FILTERS_MAILBOX,   // Configura filtros y mailbox del CAN
80     CAN_ACTIVE,                    // Estado activo: lectura de sensores y envío de datos
           OBD
81     CAN_DISABLED,                 // Sin bus CAN, solo GPS e IMU activos
82     GPS_OFF,                       // Apaga el GPS
83     SEND_HTTP                      // Envía datos a servidor vía HTTP
84 } SystemState_t;
85
86 SystemState_t State = INIT;          // Variable que almacena el estado actual del
       sistema; inicia en el estado 'INIT'
87
88 /********* DECLARACIÓN DE LAS FUNCIONES DEL MAIN PRINCIPAL **********/
89 void Init_Peripheral(void);                       // Inicializa todos los periféricos (
       GPIO, I2C, CAN, USART, etc.)
90 void Conf_Peripheral(void);                       // Configura parámetros específicos de
        los periféricos
91 void SecuanciaInicio(void);                       // Ejecuta la secuencia de inicio del
       sistema
92 void MPU6050_Read_Ace_Giro(void);                 // Lee datos del acelerómetro y
       giroscopio (MPU6050)
93 void delay(uint32_t n);                           // Genera un retardo en milisegundos (
       bloqueante)
94
95 void sendMessage_http_field1(uint16_t fieldValue); // Envía el valor al campo 1 de
       ThingSpeak (GSM/GPRS - AT)
96 void sendMessage_http_field2(uint16_t fieldValue); // Envía el valor al campo 2 de
       ThingSpeak (GSM/GPRS - AT)
97 void sendMessage_http_field3(uint16_t fieldValue); // Envía el valor al campo 3 de
       ThingSpeak (GSM/GPRS - AT)
98 void sendMessage_http_field4(uint16_t fieldValue); // Envía el valor al campo 4 de
       ThingSpeak (GSM/GPRS - AT)
```

```
99
100 void Obdii_Message(uint32_t Obdii_pid);           // Construye y envía el mensaje CAN
    con el PID OBDII recibido
101 void Get_Obdii_Data(void);                        // Solicita datos OBDII al vehículo,
    pide todos los PIDs configurados
102
103 void SSD1306_LoadUI(void);                       // Prepara la interfaz estática de la
    pantalla OLED, cargando los títulos y etiquetas donde luego irán los datos dinámicos.
104 void SSD1306_Refresh(void);                      // Refresca la pantalla OLED SSD1306 (
    actualiza visualización)
105
106 void Finite_State_Machine(SystemState_t *State); // Máquina de estados principal del
    sistema: gestiona el flujo entre cada etapa del firmware
107
108 /*********FIN DE LAS FUNCIONES DEL MAIN PRINCIPAL***********/
109
110
111 /********* DECLARACIÓN DE VARIABLES GLOBALES **********/
112
113 /***** Variables de aceleración lineal (MPU6050 - eje X, Y, Z) *****/
114 int16_t AC_X = 0;                 // Aceleración cruda eje X (raw del sensor)
115 int16_t AC_Y = 0;                 // Aceleración cruda eje Y (raw del sensor)
116 int16_t AC_Z = 0;                 // Aceleración cruda eje Z (raw del sensor)
117
118 float AX = 0;                     // Aceleración procesada eje X (en g o m/s^2)
119 float AY = 0;                     // Aceleración procesada eje Y
120 float AZ = 0;                     // Aceleración procesada eje Z
121
122 /***** Variables de aceleración angular / giroscopio (MPU6050 - eje X, Y, Z) *****/
123 int16_t G_X = 0;                  // Giroscopio crudo eje X (raw del sensor)
124 int16_t G_Y = 0;                  // Giroscopio crudo eje Y
125 int16_t G_Z = 0;                  // Giroscopio crudo eje Z
126
127 float GX = 0;                     // Giroscopio procesado eje X (en grados/s o rad/s)
128 float GY = 0;                     // Giroscopio procesado eje Y
129 float GZ = 0;                     // Giroscopio procesado eje Z
130
131 /***** Variables auxiliares (revisar si se usan activamente) *****/
132 uint32_t receivedData0_L, receivedData0_H;  // Datos recibidos CAN (parte baja y alta)
133 uint32_t receivedData1_L, receivedData1_H;  // Datos recibidos CAN (parte baja y alta)
134 uint8_t PID;                      // PID actual a solicitar o procesar
135
136 /***** Variables OBDII - datos obtenidos del vehículo *****/
137 int Vel_Km = 0;                   // Velocidad del vehículo en Km/h
138 int RPM_motor = 0;                // Revoluciones por minuto del motor
139 int Temp_liquido = 0;             // Temperatura del refrigerante
140 int Combustible = 0;              // Nivel de combustible (%)
141 int Variable_Prueba = 0;
142 /***** Variables de control y conteo de mensajes *****/
143 int messageCounter = 0;           // Contador de mensajes enviados
144 uint8_t ByteA, ByteB, ByteC, ByteD;  // Bytes individuales posiblemente usados en
    empaquetado CAN
145
146 /***** Punteros a las estructuras FIFO del hardware CAN *****/
147 CAN_FIFOMailBox_TypeDef *FIFOmailBox0, *FIFOmailBox1;
148
149 // ========================================================================
150 //       FUNCION PRINCIPAL - PUNTO DE ENTRADA DEL SISTEMA EMBEBIDO
```

```
151 // ============================================================================
152 int main(void)
153 {
154     while (1) {
155         Finite_State_Machine(&State);   // Ejecuta la máquina de estados principal
156     }
157 }
158 
159 // ============================================================================
160 //                         FIN DE FUNCION PRINCIPAL
161 // ============================================================================
162 
163 
164 // ============================================================================
165 //          IMPLEMENTACIÓN DE FUNCIONES DEFINIDAS POR EL USUARIO
166 // ============================================================================
167 
168 
169 void Init_Peripheral(void){
170       /*CONFIGURACIÓN DEL RELOJ*/
171       SystCLK_SetMSI(MSI_16MHz);     //  Configura 16MHz como  reloj
172       RCC_EnPort(GPIOB);             //  Configura reloj en puerto B Para CANX
173       RCC_EnPort(GPIOC);             //  Configura reloj en puerto C para I2C
174       RCC_En_I2C(I2C3);              //  Habilitar reloj al periférico I2C.
175       RCC_En_CANx(CAN1);             //  Habilitar reloj al periferico CAN
176       RCC_En_USARTx(USART3);
177 }
178 
179 void Conf_Peripheral(void) {
180     GPIOx_InitIO(GPIOC, 13, GPIO_MODER_INPUT, false);                    // Botón de
        usuario en la Nucleo (PC13 como entrada)
181     GPIOx_InitIO(GPIOB, 13, GPIO_MODER_OUTPUT, false);                   // LED de
        usuario en la Nucleo (PB13 como salida)
182 
183     // Configuración de pines para I2C3 (MPU6050, SSD1306 OLED)
184     GPIOx_InitAF(GPIOC, 0, 1, 2, 4, false);                               // PC0 -
        I2C3_SCL (Open Drain, High Speed, AF4)
185     GPIOx_InitAF(GPIOC, 1, 1, 2, 4, false);                               // PC1 -
        I2C3_SDA (Open Drain, High Speed, AF4)
186 
187     // Configuración de pines para CAN1
188     GPIOx_InitAF(GPIOB, 8, GPIO_OTYPER_PP, GPIO_OSPEEDR_HS, 9, false);   // PB8 -
        CAN1_RX (Push-Pull, High Speed, AF9)
189     GPIOx_InitAF(GPIOB, 9, GPIO_OTYPER_PP, GPIO_OSPEEDR_HS, 9, false);   // PB9 -
        CAN1_TX (Push-Pull, High Speed, AF9)
190 
191     // Configuración de pines para USART3 (comunicación con módulo GSM/GPRS)
192     GPIOx_InitAF(GPIOB, 10, GPIO_OTYPER_PP, 0, 7, false);                 // PB10 -
        USART3_TX (Push-Pull, AF7)
193     GPIOx_InitAF(GPIOB, 11, GPIO_OTYPER_PP, 0, 7, true);                  // PB11 -
        USART3_RX (Push-Pull, AF7, con Pull-Up)
194 }
195 
196 void MPU6050_Read_Ace_Giro(void) {
197     // Lectura de la aceleración en el eje X (dato crudo)
198     AC_X = MPU6050_Read(MPU6050_ACCEL_XOUT_H);
199     AX = (AC_X / 16384.0);                    // Conversión a 'g' según la escala del
        sensor
```

```
200     // Lectura de la aceleración en el eje Y (dato crudo)
201     AC_Y = MPU6050_Read(MPU6050_ACCEL_YOUT_H);
202     AY = (AC_Y / 16384.0);                  // Conversión a 'g'
203     // Lectura de la aceleración en el eje Z (dato crudo)
204     AC_Z = MPU6050_Read(MPU6050_ACCEL_ZOUT_H);
205     AZ = (AC_Z / 16384.0);                  // Conversión a 'g'
206     // Lectura de la velocidad angular (giroscopio) en el eje X (dato crudo)
207     G_X = MPU6050_Read(MPU6050_GYRO_XOUT_H);
208     GX = (G_X / 131.0);                     // Conversión a grados/s (escala + - 250
            grados/s)
209     // Lectura de la velocidad angular en el eje Y (dato crudo)
210     G_Y = MPU6050_Read(MPU6050_GYRO_YOUT_H);
211     GY = (G_Y / 131.0);                     // Conversión a grados/s
212     // Lectura de la velocidad angular en el eje Z (dato crudo)
213     G_Z = MPU6050_Read(MPU6050_GYRO_ZOUT_H);
214     GZ = (G_Z / 131.0);                     // Conversión a grados/s
215 }
216
217 void SecuanciaInicio(void) {
218     // Detiene cualquier scroll de la pantalla OLED
219     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0x2E); // Scroll OFF
220     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA5); // Display ALL ON (test)
221     delay(1000000);
222
223     // Alterna entre mostrar encendido y volver a modo normal varias veces
224     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA4); // Display normal
225     delay(1000000);
226     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA5); // Display ALL ON
227     delay(1000000);
228     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA4); // Display normal
229     delay(1000000);
230     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA5); // Display ALL ON
231     delay(1000000);
232     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA4); // Display normal
233     delay(1000000);
234
235     // Muestra el nombre del laboratorio
236     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE1);
237     SSD1306_PosCom(0);
238     SSD1306_WriteString("␣␣!␣LIESE␣LAB␣2025␣!␣");
239     delay(2000000);
240
241     // Limpia y luego muestra "TELEMETRY SYSTEM"
242     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE3);
243     SSD1306_PosCom(0);
244     SSD1306_WriteString("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"); // Limpia línea
245     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE3);
246     SSD1306_PosCom(0);
247     SSD1306_WriteString("!␣TELEMETRY␣SYSTEM␣!");
248     delay(2000000);
249
250     // Limpia y luego muestra "SCANNER OBDII"
251     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE5);
252     SSD1306_PosCom(0);
253     SSD1306_WriteString("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣"); // Limpia línea
254     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE5);
255     SSD1306_PosCom(0);
256     SSD1306_WriteString("␣␣!␣SCANNER␣OBDII␣!␣");
```

```
257     delay(2500000);
258
259     // Efecto visual de inversión de pantalla varias veces
260     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA7); // Invert display ON
261     delay(1000000);
262     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA6); // Invert display OFF
263     delay(1000000);
264     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA7); // Invert display ON
265     delay(1000000);
266     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA6); // Invert display OFF
267     delay(1000000);
268     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA7); // Invert display ON
269     delay(1000000);
270     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, 0xA6); // Invert display OFF
271     delay(1000000);
272     SSD1306_Clear(); // Limpia la pantalla al finalizar la secuencia de inicio
273 }
274
275
276 /*****************************************************************************
277  * @brief   Carga la interfaz de usuario estática en la pantalla OLED SSD1306
278  *          Establece las etiquetas base donde luego se mostrarán los datos dinámicos
279  *****************************************************************************
        */
280 void SSD1306_LoadUI(void) {
281     // Escribe el título principal en la primera página de la pantalla OLED
282     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0);   // Selecciona la pá
            gina 0 del display
283     SSD1306_PosCom(0);                                               // Posiciona el
            cursor en la columna 0
284     SSD1306_WriteString("!␣TELEMETRY␣SYSTEM␣!");                     // Escribe el título
             en la pantalla
285
286     // Configura la pantalla para mostrar la etiqueta "VELOCIDAD" en la página 3
287     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE3);   // Selecciona la pá
            gina 3 del display
288     SSD1306_PosCom(0);                                               // Posiciona el
            cursor en la columna 0
289     SSD1306_WriteString("␣VELOCIDAD␣:␣␣␣␣␣␣␣");                     // Escribe la
            etiqueta para la velocidad
290
291     // Configura la pantalla para mostrar la etiqueta "RPMS" en la página 5
292     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE5);   // Selecciona la pá
            gina 5 del display
293     SSD1306_PosCom(0);                                               // Posiciona el
            cursor en la columna 0
294     SSD1306_WriteString("␣RPMS␣:␣␣␣␣␣␣␣␣␣␣␣␣␣␣");                 // Escribe la
            etiqueta para las RPMs
295
296     // Configura la pantalla para mostrar la etiqueta "GAS" en la página 7
297     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE7);   // Selecciona la pá
            gina 7 del display
298     SSD1306_PosCom(0);                                               // Posiciona el
            cursor en la columna 0
299     SSD1306_WriteString("␣GAS␣:␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣");                 // Escribe la
            etiqueta para el nivel de combustible
300 }
301
```

```
302 // Función de retardo simple: realiza una espera bloqueante según el valor de 'n'
303 void delay(uint32_t n)
304 {
305     while(n--);
306 }
307 
308 void CAN1_RX0_IRQHandler(void) {
309     NVIC_ClearPendingIRQ(CAN1_RX0_IRQn);
310     CAN1_Rx0_FIFOx(CAN1, FIFOmailBox0);                                  // Lee el mailbox 0
311     receivedData0_L = CAN1->sFIFOMailBox[0].RDLR;                        // Captura datos low
312     receivedData0_H = CAN1->sFIFOMailBox[0].RDHR;                        // Captura datos
        high
313     PID = (receivedData0_L & 0x00FF0000) >> 16;                          // Extrae PID del
        dato recibido
314 }
315 
316 void CAN1_RX1_IRQHandler(void) {
317     NVIC_ClearPendingIRQ(CAN1_RX1_IRQn);
318     CAN1_Rx1_FIFOx(CAN1, FIFOmailBox1);                                  // Lee el mailbox 1
319     receivedData1_L = CAN1->sFIFOMailBox[1].RDLR;                        // Captura datos low
320     receivedData1_H = CAN1->sFIFOMailBox[1].RDHR;                        // Captura datos
        high
321     PID = (receivedData1_L & 0x00FF0000) >> 16;                          // Extrae PID del
        dato recibido
322 }
323 
324 /* ***************************************************************************
325  * Función para enviar datos a través de HTTP con el valor del campo 1
326  * ***************************************************************************/
327 void sendMessage_http_field1(uint16_t fieldValue) {
328     // Configura el contexto PDP con el APN de Telcel (México)
329     sendStringUARTx(USART3, "AT+QICSGP=1,1,\"internet.itelcel.com\",\"webgprs\",\"
        webgprs2002\",1");
330     delay(200000);
331     // Activa el contexto PDP para poder realizar transmisiones de datos
332     sendStringUARTx(USART3, "AT+QIACT=1\r\n\0");
333     delay(200000);
334     // Consulta el estado del contexto PDP (opcional, para verificar que está activo)
335     sendStringUARTx(USART3, "AT+QIACT?\r\n\0");
336     delay(200000);
337     // Configura el contexto ID que usará el cliente HTTP
338     sendStringUARTx(USART3, "AT+QHTTPCFG=\"contextid\",1\r\n\0");
339     delay(200000);
340     // Prepara el módulo para recibir la URL del servidor (Thingspeak)
341     sendStringUARTx(USART3, "AT+QHTTPURL=74,30\r\n\0");  // 71 = longitud de la URL, 30
        = timeout
342     delay(200000);
343     // Construye la URL con el valor de 'field1' para enviar a Thingspeak
344     char url[100];
345     snprintf(url, sizeof(url), "https://api.thingspeak.com/update?api_key=
        JQ5F91V9QFHPK4R8&field1=%d\r\n", fieldValue);
346     sendStringUARTx(USART3, url);  // Envía la URL al módulo
347     delay(200000);
348     // Ejecuta la petición GET al servidor con un timeout de 15 segundos
349     sendStringUARTx(USART3, "AT+QHTTPGET=15\r\n\0");
350     delay(200000);
351     // Repite la petición GET por seguridad / asegurando la transmisión
352     sendStringUARTx(USART3, "AT+QHTTPGET=15\r\n\0");
```

```
353     delay(200000);
354 }
355 
356 /* ***************************************************************************
357  * Función para enviar datos a través de HTTP con el valor del campo 2
358  * ***************************************************************************/
359 void sendMessage_http_field2(uint16_t fieldValue){
360     sendStringUARTx(USART3,"AT+QICSGP=1,1,\"internet.itelcel.com\",\"webgprs\",\"
        webgprs2002\",1");
361     delay(100000);
362     sendStringUARTx(USART3, "AT+QIACT=1\r\n\0");
363     delay(100000);
364     sendStringUARTx(USART3, "AT+QIACT?\r\n\0");
365     delay(100000);
366     sendStringUARTx(USART3, "AT+QHTTPCFG=\"contextid\",1\r\n\0");
367     delay(100000);
368     sendStringUARTx(USART3, "AT+QHTTPURL=71,30\r\n\0");
369     delay(100000);
370     char url[100];  // Asegúrate de que el tamaño sea adecuado
371     snprintf(url, sizeof(url), "https://api.thingspeak.com/update?api_key=
        D2SHDYLUTSW1PFRW&field2=%d\r\n", fieldValue);
372     sendStringUARTx(USART3, url);
373     delay(100000);
374     sendStringUARTx(USART3, "AT+QHTTPGET=30\r\n\0");
375     delay(100000);
376     sendStringUARTx(USART3, "AT+QHTTPGET=30\r\n\0");
377     delay(100000);
378 }
379 
380 /* ***************************************************************************
381  * Función para enviar datos a través de HTTP con el valor del campo 3
382  * ***************************************************************************/
383 void sendMessage_http_field3(uint16_t fieldValue){
384     sendStringUARTx(USART3,"AT+QICSGP=1,1,\"internet.itelcel.com\",\"webgprs\",\"
        webgprs2002\",1");
385     delay(100000);
386     sendStringUARTx(USART3, "AT+QIACT=1\r\n\0");
387     delay(100000);
388     sendStringUARTx(USART3, "AT+QIACT?\r\n\0");
389     delay(100000);
390     sendStringUARTx(USART3, "AT+QHTTPCFG=\"contextid\",1\r\n\0");
391     delay(100000);
392     sendStringUARTx(USART3, "AT+QHTTPURL=71,30\r\n\0");
393     delay(100000);
394     char url[100];  // Asegúrate de que el tamaño sea adecuado
395     snprintf(url, sizeof(url), "https://api.thingspeak.com/update?api_key=
        D2SHDYLUTSW1PFRW&field3=%d\r\n", fieldValue);
396     sendStringUARTx(USART3, url);
397     delay(100000);
398     sendStringUARTx(USART3, "AT+QHTTPGET=30\r\n\0");
399     delay(100000);
400     sendStringUARTx(USART3, "AT+QHTTPGET=30\r\n\0");
401     delay(100000);
402 }
403 
404 /* ***************************************************************************
405  * Función para enviar datos a través de HTTP con el valor del campo 4
406  * ***************************************************************************/
```

```
407 void sendMessage_http_field4(uint16_t fieldValue) {
408     sendStringUARTx(USART3, "AT+QICSGP=1,1,\"internet.itelcel.com\",\"webgprs\",\"
        webgprs2002\",1");
409     delay(100000);
410     sendStringUARTx(USART3, "AT+QIACT=1\r\n\0");
411     delay(100000);
412     sendStringUARTx(USART3, "AT+QIACT?\r\n\0");
413     delay(100000);
414     sendStringUARTx(USART3, "AT+QHTTPCFG=\"contextid\",1\r\n\0");
415     delay(100000);
416     sendStringUARTx(USART3, "AT+QHTTPURL=71,30\r\n\0");
417     delay(100000);
418     char url[100];
419     snprintf(url, sizeof(url), "https://api.thingspeak.com/update?api_key=
        D2SHDYLUTSW1PFRW&field4=%d\r\n", fieldValue);
420     sendStringUARTx(USART3, url);
421     delay(100000);
422     sendStringUARTx(USART3, "AT+QHTTPGET=30\r\n\0");
423     delay(100000);
424     sendStringUARTx(USART3, "AT+QHTTPGET=30\r\n\0");
425     delay(100000);
426 }
427 
428 /* ***************************************************************************
429  * Función para enviar un mensaje OBD-II con un PID específico
430  * ***************************************************************************/
431 void Obdii_Message(uint32_t Obdii_pid) {
432     CAN_MailboxConfig(CAN1, false, 0x7DF, false, 0x0); // Configuración del buzón CAN
433     CAN_SendData(CAN1, 8, Obdii_pid, 0x0);             // Enviar mensaje CAN
434     CAN_RequestTransmission(CAN1, 0x0);                // Solicitar transmisión CAN
435 }
436 
437 /* ***************************************************************************
438  * Función para obtener datos OBD-II (velocidad, RPM, temperatura, combustible)
439  * ***************************************************************************/
440 void Get_Obdii_Data(void) {
441 
442     // Petición de la velocidad
443     //Limpiar Bytes
444     ByteA = 0;
445     ByteB = 0;
446     ByteC = 0;
447     ByteD = 0;
448 
449     delay(100);
450     Obdii_Message(PID_0D);    //Enviar mensaje de peticion OBDII de la velociad
451     delay(10000);
452 
453     //Filtrar cada Byte del mensaje de respuesta
454     ByteA = (CAN1->sFIFOMailBox[0].RDLR & 0xFF000000) >> 24;
455     ByteB = (CAN1->sFIFOMailBox[0].RDHR & 0x000000FF);
456     ByteC = (CAN1->sFIFOMailBox[0].RDHR & 0x0000FF00) >> 8;
457     ByteD = (CAN1->sFIFOMailBox[0].RDHR& 0x00FF0000) >> 16;
458 
459     Vel_Km = ByteA; //Formlula para interpertar la velocidad
460 
461     // Petición de las RPM
462     //Limpiar Bytes
```

```
463     ByteA = 0;
464     ByteB = 0;
465     ByteC = 0;
466     ByteD = 0;
467 
468     delay(100);
469     Obdii_Message(PID_0C);    //Enviar mensaje de peticion OBDII de las RPM
470     delay(10000);
471 
472     //Filtra cada Byte del mensaje de repuesta
473     ByteA = (CAN1->sFIFOMailBox[0].RDLR & 0xFF000000) >> 24;
474     ByteB = (CAN1->sFIFOMailBox[0].RDHR & 0x000000FF);
475     ByteC = (CAN1->sFIFOMailBox[0].RDHR & 0x0000FF00) >> 8;
476     ByteD = (CAN1->sFIFOMailBox[0].RDHR& 0x00FF0000) >> 16;
477 
478     RPM_motor = (256 * ByteA + ByteB) / 4; //Formula para interpretar las RPM del motor
479 
480     // Petición del liquido de enfriamiento
481     //Limpiar Bytes
482     ByteA = 0;
483     ByteB = 0;
484     ByteC = 0;
485     ByteD = 0;
486 
487     delay(100);
488     Obdii_Message(PID_05);    //Enviar mensaje de peticion OBDII para el liquido de
        enfriamiento
489     delay(10000);
490 
491     //Filtra cada Byte del mensaje de la respuesta del liquido de enfriamiento
492     ByteA = (CAN1->sFIFOMailBox[0].RDLR & 0xFF000000) >> 24;
493     ByteB = (CAN1->sFIFOMailBox[0].RDHR & 0x000000FF);
494     ByteC = (CAN1->sFIFOMailBox[0].RDHR & 0x0000FF00) >> 8;
495     ByteD = (CAN1->sFIFOMailBox[0].RDHR& 0x00FF0000) >> 16;
496 
497     Temp_liquido = ByteA - 40; //Formula para interpretar la temperatura del liquido de
        enfriamiento
498 
499     // Petición del nivel de combustible
500     //Limpiar Bytes
501     ByteA = 0;
502     ByteB = 0;
503     ByteC = 0;
504     ByteD = 0;
505 
506     delay(100);
507     Obdii_Message(PID_2F);  //Enviar mensaje de peticion OBDII del nivel de combustible
508     delay(10000);
509 
510     //Filtra cada Byte del mensaje de la respuesta del nivel de combustible
511     ByteA = (CAN1->sFIFOMailBox[0].RDLR & 0xFF000000) >> 24;
512     ByteB = (CAN1->sFIFOMailBox[0].RDHR & 0x000000FF);
513     ByteC = (CAN1->sFIFOMailBox[0].RDHR & 0x0000FF00) >> 8;
514     ByteD = (CAN1->sFIFOMailBox[0].RDHR& 0x00FF0000) >> 16;
515 
516     Combustible = (ByteA * 100) / 255; //Formula para interpretar el nivel de
        combustible
517 }
```

```
518
519 void SSD1306_Refresh(void) {
520     char buffer[10]; // Espacio para almacenar la representación en cadena
521
522     // Mostrar velocidad en la página 3
523     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE3);
524     SSD1306_PosCom(80);
525     SSD1306_WriteString("␣␣␣␣␣␣␣␣␣␣␣␣␣");
526     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE3);
527     SSD1306_PosCom(80);
528     sprintf(buffer, "%d", Vel_Km);
529     SSD1306_WriteString(buffer);         // Escribe la cadena en la pantalla OLED
530     SSD1306_WriteString("␣KM");
531
532     // Mostrar RPM en la página 5
533     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE5);
534     SSD1306_PosCom(50);
535     SSD1306_WriteString("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣");
536     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE5);
537     SSD1306_PosCom(50);
538     sprintf(buffer, "%d", RPM_motor);
539     SSD1306_WriteString(buffer);         // Escribe la cadena en la pantalla OLED
540     SSD1306_WriteString("␣RPM");
541
542     // Mostrar nivel de combustible en la página 7
543     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE7);
544     SSD1306_PosCom(70);
545     SSD1306_WriteString("␣␣␣␣␣␣␣␣␣␣␣␣␣␣");
546     I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE7);
547     SSD1306_PosCom(70);
548     sprintf(buffer, "%d", Combustible);
549     SSD1306_WriteString(buffer);         // Escribe la cadena en la pantalla OLED
550     SSD1306_WriteString("␣␣T");
551 }
552
553
554
555 /*****************************************************************************
556  * @brief   Máquina de estados principal del sistema (modifica el estado por referencia)
557  * @param   currentState - Puntero al estado actual del sistema
558  *****************************************************************************
     */
559 void Finite_State_Machine(SystemState_t *State) {
560     switch (*State) {
561         case INIT:
562             *State = CONFIG_PERIPH;
563             break;
564
565         case CONFIG_PERIPH:
566 /*************************** INICIALIZACIÓN Y CONFIGURACIÓN DE PUERTOS Y PERIFÉRICOS
    ***************************/
567 /*
    *****************************************************************************
    */
568             Init_Peripheral();                                          //
                Inicializa Perifericos del sistema
569             Conf_Peripheral();                                          // Configura
                los puertos GPIO y asigna funciones alternas (I2C, CAN, USART)
```

```
570
571             *State = I2C_INIT;
572             break;
573
574         case I2C_INIT:
575
576             I2C_Init(I2C3);                                                 //
                Inicializa el periférico I2C3 (comunicación con MPU6050 y OLED SSD1306)
577
578             *State = OLED_STARTUP;
579             break;
580
581         case OLED_STARTUP:
582             SSD1306_Init();                                                 // Inicializa
                y configura la pantalla OLED SSD1306
583             SSD1306_Clear();                                                // Limpia la
                pantalla (borra el contenido)
584             SecuanciaInicio();                                              // Ejecuta la
                secuencia de inicio de la interfaz en pantalla
585             *State = USART_INIT;
586             break;
587
588         case USART_INIT:
589             USARTx_CONF(USART3, USARTx_BRR_MSI_16MHz, USART3_IRQn, 7);      // Configura
                 USART3 a 16 MHz, habilita interrupciones y prioridad NVIC = 7
                                                                    // (Utilizado para
                enviar comandos AT al módulo GSM/GPRS)
590             I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0);   //
                Selecciona la página 0 del display
591             SSD1306_PosCom(0);                                              // Posiciona
                 el cursor en la columna 0
592             SSD1306_WriteString("!!␣USART␣OK␣!!");
593             for (int i = 0; i < 4; i++) {
594                 delay(1000000); // Ejecuta 4 retardos consecutivos para dar tiempo a que
                     el usuario visualice el mensaje
595             }
596             SSD1306_Clear();
597             *State = MPU_INIT;
598             break;
599
600         case MPU_INIT:
601             MPU6050_Init();
602             I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0);   //
                Selecciona la página 0 del display
603             SSD1306_PosCom(0);                                              // Posiciona
                 el cursor en la columna 0
604             SSD1306_WriteString("!!␣MPU␣OK␣!!");
605             for (int i = 0; i < 4; i++) {
606                 delay(1000000); // Ejecuta 4 retardos consecutivos para dar tiempo a que
                     el usuario visualice el mensaje
607             }
608             SSD1306_Clear();
609             *State = CAN_INIT;
610             break;
611
612         case CAN_INIT:
613             if (CANx_Init(CAN1)) {
614                 *State = CAN_BUS_OK;
```

```
615             } else {
616                 *State = CAN_NO_BUS;      // Timeout en INIT
617             }
618 
619             break;
620 
621         case CAN_NO_BUS:
622             I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0);   //
                   Selecciona la página 0 del display
623             SSD1306_PosCom(0);                                                 // Posiciona
                    el cursor en la columna 0
624             SSD1306_WriteString("!! CAN BUS NO !!");
625             for (int i = 0; i < 4; i++) {
626                 delay(1000000); // Ejecuta 4 retardos consecutivos para dar tiempo a que
                        el usuario visualice el mensaje
627             }
628             SSD1306_Clear();
629             *State = LOAD_UI;
630             break;
631 
632         case CAN_BUS_OK:
633             I2C_Tx_2Bytes(I2C3, AdreSSD1306, ControlbC, SSD1306_SETPAGE0);   //
                   Selecciona la página 0 del display
634             SSD1306_PosCom(0);                                                 // Posiciona
                    el cursor en la columna 0
635             SSD1306_WriteString("!! CAN BUS OK !!");
636             for (int i = 0; i < 4; i++) {
637                 delay(1000000);                                          // Ejecuta 4
                    retardos consecutivos para dar tiempo a que el usuario visualice el
                    mensaje
638             }
639             SSD1306_Clear();
640             *State = LOAD_UI;
641             break;
642 
643         case LOAD_UI:
644             SSD1306_LoadUI();
645             *State = GPS_ACTIVATE;
646             break;
647 
648         case GPS_ACTIVATE:
649                 delay(100000);                                       // Pequeño retardo
                    para asegurar la correcta inicialización del sensor
650             sendStringUARTx(USART3, "AT+QGPS=1\r\n\0");          // Envía comando AT
                    por USART3 al módulo GSM/GPRS para activar el GPS interno
                                                 // Ejecuta 4 retardos consecutivos para dar
                    tiempo a que el usuario visualice el mensaje
651 
652             *State = CAN_REINIT;
653             break;
654 
655         case CAN_REINIT:
656             if (CANx_Init(CAN1)) {
657                     *State = CAN_CONFIG_FILTERS_MAILBOX;
658                 } else {
659                     *State = CAN_DISABLED;
660                 }
661             break;
```

```
662
663         case CAN_CONFIG_FILTERS_MAILBOX:
664             /************************* CONFIGURACIÓN DE FILTROS DE LAS FIFOs DE RECEPCIÓ
                    N *************************/
665             /*
                    ********************************************************************************
                    */
666             CAN_FilterInit(CAN1, List_mode, Single_32bit, Fifo_0, 0);   // Configura
                    Filtro 0 en modo lista, escala 32 bits, asignado a FIFO 0
667             CAN_SetFilterValue(CAN1, 0x7E8, 0x00, 0);                   // Establece
                    filtro 0 para recibir mensajes con ID 0x7E8 (respuesta ECU)
668             CLEAR_BIT(CAN1->FMR, CAN_FMR_FINIT);                        // Sale del modo
                     de inicialización de filtros y habilita los filtros
669
670             /************************* CONFIGURACIÓN DE FILTROS DE LOS MAILBOX DE
                    TRANSMISIÓN **********************/
671             /*
                    ********************************************************************************
                    */
672             CAN_MailboxConfig(CAN1, false, 0x7DF, false, 0x0);          // Configura el
                    Mailbox de transmisión: ID 0x7DF (broadcast OBDII), estándar
673             CAN_SendData(CAN1, 8, 0x000D0102, 0x0);                     // Envía mensaje
                     CAN de 8 bytes al Mailbox 0 con el dato 0x000D0102
674             *State = CAN_ACTIVE;
675             break;
676
677         case CAN_ACTIVE:
678             for (int i = 0; i < 20; i++) {
679                 GPS_PET();                // Obtener ubicación GNSS
680                 Get_Obdii_Data();         // Solo si hay bus CAN
681                 MPU6050_Read_Ace_Giro();  // IMU
682                 SSD1306_Refresh();        // Actualizar pantalla
683             }
684             *State = GPS_OFF;
685             break;
686
687         case CAN_DISABLED:
688             for (int i = 0; i < 20; i++) {
689                 GPS_PET();                // GNSS sigue funcionando
690                 MPU6050_Read_Ace_Giro();  // IMU
691                 SSD1306_Refresh();        // Pantalla también se actualiza
692             }
693             *State = GPS_OFF;
694             break;
695
696         case GPS_OFF:
697             delay(200000);
698             sendStringUARTx(USART3, "AT+QGPSEND\r\n\0");          //
699             delay(200000);
700             *State = SEND_HTTP;
701             break;
702
703
704         case SEND_HTTP:
705             sendMessage_http_field1(Vel_Km);
706             sendMessage_http_field2(RPM_motor);
707             sendMessage_http_field3(Temp_liquido);
708             sendMessage_http_field4(Combustible);
```

```
709             *State = GPS_ACTIVATE;   // Vuelve al ciclo principal
710             break;
711         default:
712             *State = INIT;
713             break;
714     }
715 }
```

Referencias

[1] T. N. México. «Soluciones de Administración de Flotas y Rastreo GPS.» Consultado el 28 de febrero de 2025. dirección: https://www.teletracnavman.com.mx/.

[2] T. Navman. «VT101.» Consultado el 28 de febrero de 2025. dirección: https://www.teletracnavman.com/product-resources/vt101.

[3] T. Navman. «SI201.» Consultado el 28 de febrero de 2025. dirección: https://www.teletracnavman.com/product-resources/si201.

[4] T. N. México. «TN360: Plataforma de Gestión de Flotas con Inteligencia Artificial.» Consultado el 28 de febrero de 2025. dirección: https://www.teletracnavman.com.mx/software/tn360.

[5] I. México. «Rastreo Satelital y Telemetría: Potencia el Control de tus vehículos.» Consultado el 28 de febrero de 2025. dirección: https://www.ituran.com.mx/.

[6] I. México. «Ituran Telematics.» Consultado el 28 de febrero de 2025. dirección: https://www.ituran.com.mx/ituran-telematics.html.

[7] K. GPS. «Monitoreo y Control de Flotas.» Consultado el 28 de febrero de 2025. dirección: https://kosmosgps.com/monitoreo-y-control-de-flotas/.

[8] K. GPS. «Kosmos Antares: Solución de Rastreo Satelital para Transporte.» Consultado el 28 de febrero de 2025. dirección: https://kosmosgps.com/rastreo-satelital/kosmos-antares/.

[9] K. GPS. «Kosmos Taurus: Sistema de Rastreo Satelital para Automóviles Particulares y Utilitarios.» Consultado el 28 de febrero de 2025. dirección: https://kosmosgps.com/rastreo-satelital/kosmos-taurus/.

[10] K. G. S. de C.V. «Draco Telematics System.» Consultado el 28 de febrero de 2025. dirección: https://draco-kosmos.com/.

[11] ORBCOMM. «Seguimiento de remolques por GPS.» Consultado el 28 de febrero de 2025. dirección: https://go.orbcomm.com/gps-trailer-tracking-es/.

[12] ORBCOMM, *BT 500: Gestión de camiones rápida, fiable y en tiempo real*, Consultado el 28 de febrero de 2025, 2021. dirección: https://www.orbcomm.com/PDF/datasheet/spanish/bt-500-spa.pdf.

[13] ORBCOMM, *IC 500: Cámara en cabina para captura completa de eventos críticos*, Consultado el 28 de febrero de 2025, 2023. dirección: https://www.orbcomm.com/PDF/datasheet/spanish/ic-500-SPA.pdf.

[14] S. México. «Líder en tecnología industrial IoT.» Consultado el 28 de febrero de 2025. dirección: https://www.samsara.com/mx.

[15] Samsara, *Vehicle Gateway VG55: Advanced Sensor Platform for Fleet Management*, Consultado el 28 de febrero de 2025, 2025. dirección: https://www.samsara.com/pdf/Datasheet-Vehicle-Gateway-VG55.pdf.

[16] Samsara. «AG51: Solución de problemas de Gateway de activos sin alimentación (AG).» Consultado el 28 de febrero de 2025. dirección: https://samsara1675703105.zendesk.com/hc/es/articles/20345215804557-AG51-Soluci%C3%B3n-de-problemas-de-Gateway-de-activos-sin-alimentaci%C3%B3n-AG.

[17] Fiesa. «VT310 Rastreador Meitrack.» Consultado el 28 de febrero de 2025. dirección: https://www.fiesa.com.ar/DETALLE/VT310-Rastreador-Meitrack/ITEM_ID=251/fiesa.aspx.

[18] Orgontec. «Escáner Automotriz OBDII ELM327 Bluetooth.» Consultado el 28 de febrero de 2025. dirección: https://www.orgontec.mx/product-page/escaner-automotriz-obdii-elm327-bluetooth.

[19] J. A. Rodríguez Sánchez, *Desarrollo de un sistema de monitoreo, geolocalización y telemetría en vehículos de servicio bajo el concepto IoT*. Universidad Nacional Autónoma de México (UNAM), 2023, Tesis de licenciatura, Facultad de Ingeniería. dirección: http://132.248.9.195/ptd2023/marzo/0836521/Index.html.

[20] E. TC-SMG, «GSM 09.02: Digital Cellular Telecommunications System (Phase 2+); Mobile Application Part (MAP) Specification,» ETSI, inf. téc., ver. 5.3.0, ago. de 1996, Consultado el 15 de febrero de 2025. dirección: https://www.etsi.org/deliver/etsi_gts/09/0902/05.03.00_60/gsmts_0902v050300p.pdf.

[21] M. Sauter, *From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*. 2010, Consultado el 14 de febrero de 2025. dirección: https://mitmecsept.wordpress.com/wp-content/uploads/2017/04/from-gsm-to-lte.pdf.

[22] J. L. B. Valero, A. B. A. Julián y N. G. Villén, *GNSS. GPS: fundamentos y aplicaciones en Geomática*. Editorial Universitat Politècnica de València, 2014, ISBN: 978-84-9048-261-2. dirección: https://www.academia.edu/34699683/GNSS_GPS_fundamentos_y_aplicaciones_en_Geom%C3%A1tica.

[23] J. L. B. Valero, N. G. Villén y R. C. Romá, *GNSS. Geodesia espacial y Geomática*. Editorial Universitat Politècnica de València, 2023, ISBN: 978-84-1396-072-2. dirección: https://gdocu.upv.es/alfresco/service/api/node/content/workspace/SpacesStore/cb35c468-7927-4ddc-a502-219654f3bc8c/TOC_6494_01_01.pdf?guest=true.

[24] M. S. Solar. «Satélites Geoestacionarios.» Consultado el 2 de abril de 2025. dirección: https://misistemasolar.com/satelites-geoestacionarios/.

[25] R. Walter y E. Walter, «Chapter 1 Benefits and Applications of the In-Vehicle Network for Data Acquisition,» en *Data Acquisition from HD Vehicles Using J1939 CAN Bus*. 2016, págs. 1-6.

[26] N. del autor, «Título del artículo,» *Nombre de la revista*, Año de publicación. dirección: https://autosoporte.com/sistemas-controlados-por-la-ecu-automotriz/.

[27] National Instruments. «Seamlessly Connect to Third-Party Devices and Supervisory System: Controller Area Network (CAN) Overview. »dirección: https://www.ni.com/es/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/controller-area-network--can--overview.html.

[28] I. O. for Standardization. «ISO 11898-1:2015 — Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling.» Consultado el 2 de enero de 2025. dirección: https://www.iso.org/obp/ui/#iso:std:iso:11898:-1:ed-3:v1:en.

[29] M. Di Natale, *Understanding and using the controller area network communication protocol : theory and practice*. Springer, 2012, ISBN: 9781461403142. dirección: http://pbidi.unam.mx:8080/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat02025a&AN=lib.MX001001501634&lang=es&site=eds-live.

[30] Texas Instruments. «Introduction to the Controller Area Network (CAN).» Consultado el 2 de enero de 2025. dirección: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf.

[31] CRXCabling. «El cable de par trenzado: conceptos básicos para comprender qué es y cómo funciona.» Consultado el 10 de febrero de 2025. dirección: https://cabling.crxconec.com/es/crx-blog/CRX-Blog-10.html.

[32] Kvaser. «Introduction to the OBD-II Standard.» Consultado el 15 de febrero de 2025. dirección: https://kvaser.com/about-can/can-standards/introduction-to-obd-ii/.

[33] M. Falch. «OBD2 Explained - A Simple Intro.» Consultado el 20 de febrero de 2025. dirección: https://www.csselectronics.com/pages/obd2-explained-simple-intro.

[34] D. A. D. Tools. «Manual OBD-II.» Consultado el 20 de febrero de 2025. dirección: https://es.slideshare.net/slideshow/manual-obd-ii/16743003.

[35] T. Instruments, *KeyStone Architecture: Universal Asynchronous Receiver/Transmitter (UART) - User Guide*, Consultado el 20 de febrero de 2025, 2025. dirección: https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf?ts=1740734247470.

[36] E. Peña y M. G. Legaspi. «UART: A Hardware Communication Protocol.» Consultado el 20 de febrero de 2025. dirección: https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html.

[37] C. Wireless. «An Introduction to Cellular AT Commands.» Consultado el 12 de febrero de 2025. dirección: https://www.cavliwireless.com/blog/nerdiest-of-things/an-introduction-to-cellular-at-commands.html.

[38] 330ohms. «Guía sintetizada de comandos AT GSM+BT.» Consultado el 12 de febrero de 2025. dirección: https://www.330ohms.com/blogs/blog/guia-definitiva-de-comandos-at-gsmbt.

[39] Quectel, *Quectel BG95&BG77&BG600L Series AT Commands Manual V2.0*, Consultado el 12 de febrero de 2025, 2022. dirección: https://www.quectel.com/download/quectel_bg95bg77bg600l_series_at_commands_manual_v2-0/.

[40] N. Semiconductors, *UM10204: I²C-bus Specification and User Manual*, Rev. 7.0, Consultado el 22 de febrero de 2025, oct. de 2021. dirección: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[41] STMicroelectronics. «AN4555: Getting started with STM32L4 Series and STM32L4+ Series hardware development.» Consultado el 2 de abril de 2025. dirección: https://www.st.com/resource/en/application_note/an4555-getting-started-with-stm32l4-series-and-stm32l4-series-hardware-development-stmicroelectronics.pdf.

[42] Microchip Technology Inc. «MCP2561/2 Data Sheet.» Consultado el 2 de abril de 2025. dirección: https://ww1.microchip.com/downloads/en/devicedoc/20005167c.pdf.

[43] InvenSense Inc. «MPU-6000/MPU-6050 Register Map and Descriptions.» Document Number: RM-MPU-6000A-00, Revision 4.2, consultado el 2 de abril de 2025. dirección: https://www.invensense.com/.