Diseño e Implementación de una interfaz de comunicaciones mediante tecnología LPWAN Sigfox para la monitorización de parámetros de funcionamiento de un vehículo, orientado en IoT.

Armas Bolaños, Jordan Andrés

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

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

Ing. Castro Carrera, Alejandro Fabián

03 de Febrero de 2023

Informe de originalidad

NOMBRE DEL CURSO
Tesis Jordan Armas

NOMBRE DEL ALUMNO
JORDAN ANDRES ARMAS BOLAOS

NOMBRE DEL ARCHIVO
Tesis_Armas_Jordan_Final

CREACIÓN DEL INFORME
3 feb 2023

Resumen

Pasajes marcados	71	7 %
Pasajes citados/entrecomillados	9	0.8 %

Departamento de Eléctrica, Electrónica Y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Certificación

Certifico que el trabajo de titulación: "**Diseño e Implementación de una interfaz de comunicaciones mediante tecnología LPWAN Sigfox para la monitorización de parámetros de funcionamiento de un vehículo, orientado en IoT**" fue realizado por el señor **Armas Bolaños, Jordan Andrés** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 03 de Febrero de 2023

Firma

Firmado electrónicamente por:
ALEJANDRO FABIAN
CASTRO CARRERA

.....

Castro Carrera, Alejandro Fabián

C.C. 171104696-9

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica Y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Responsabilidad de Autoría

Yo, **Armas Bolaños, Jordan Andrés**, con cédula de ciudadanía N° 0401983903, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Diseño e Implementación de una interfaz de comunicaciones mediante tecnología LPWAN Sigfox para la monitorización de parámetros de funcionamiento de un vehículo, orientado en IoT"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos Intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 03 de Febrero de 2023

Firma

.....

Armas Bolaños, Jordan Andrés

C.C. 0401983903

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica Y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Autorización de Publicación

Yo, **Armas Bolaños, Jordan Andrés**, con cédula de ciudadanía N° 0401983903, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Diseño e Implementación de una interfaz de comunicaciones mediante tecnología LPWAN Sigfox para la monitorización de parámetros de funcionamiento de un vehículo, orientado en IoT"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 03 de Febrero de 2023

Firma

.....

Armas Bolaños, Jordan Andrés

C.C. 0401983903

Dedicatoria

Dedico este trabajo de titulación a mi madre Patricia por su apoyo incondicional y ser mi principal motivación para no rendirme y salir adelante, por formarme con buenos hábitos y valores y educarme con amor.

A mi padre Hernán, participe en mi formación como ser humano, y ser quien supo inculcar en mí los valores de la responsabilidad y honestidad.

A mi hermana y familiares por ser parte de mi crecimiento y motivación para cumplir mis metas y objetivos.

A mi abuelita Aura, en paz descanse, quien supo cuidarme desde mi niñez, y que con su gran amor y entrega supo demostrarme el valor del trabajo y esfuerzo.

A mis amigos, por compartir momentos y apoyarnos mutuamente durante la carrera, además de ser una de las razones por la cual la Universidad es un lugar único y hermoso.

Agradecimiento

Agradezco a mis padres por ser pieza fundamental en mi formación personal y profesional, por ser ejemplo de dedicación, constancia y responsabilidad.

A la Universidad de las Fuerzas Armadas "ESPE" por acogerme y brindarme la oportunidad de ser parte de su comunidad universitaria. A sus docentes por compartir sus conocimientos, tiempo y experiencias con mi persona.

A mis compañeros de cátedra por ser un gran apoyo y compañía en este largo proceso de educación.

Agradezco a mi Director de Tesis, el Ingeniero Alejandro Castro, por ser el principal participe en el desarrollo de este proyecto, ya que gracias a su conocimiento en el tema, aclaraciones y recomendaciones, fui capaz de solucionar todos los problemas e inconvenientes que se presentaron en este proceso.

Índice De Contenido

Índice de Tablas

Índice de Figuras

Resumen

Toda revolución implica un cambio en la noción de nuestro mundo. Por supuesto, tecnologías disruptivas como el Internet de las cosas, uso de la nube y *Big Data*, marcan una gran diferencia en el ámbito profesional y personal. Ahora es el momento perfecto, porque la mayoría de las empresas de hoy no están implementando nuevas tecnologías. Las redes de área extendida de baja potencia son tecnologías de comunicación inalámbrica capaces de transmitir datos entre un dispositivo y una estación base-*Gateway* que pueden estar separados por varios kilómetros con un muy bajo consumo energético. Por su naturaleza, estas tecnologías están posibilitando la implementación de las mayores iniciativas IoT.

Al ser diseñadas específicamente para este entorno, permiten instalar decenas o centenares de nodos distribuidos por una gran área sin requerir una infraestructura significativa o costosa.

El presente proyecto tiene como finalidad el desarrollo de una interfaz de comunicaciones para la monitorización de las variables propias de un vehículo, mediante la utilización de la tecnología Sigfox. Esta interfaz permite al operador saber cuándo las piezas del vehículo necesitan servicio o revisión, para evitar daños prematuros en los componentes y así mejorar la fiabilidad y seguridad del vehículo. A su vez la información obtenida notifica al usuario acerca de la velocidad, revoluciones por minuto y la posición en tiempo real del vehículo mediante un sistema de geolocalización, monitoreo y control de la actividad del vehículo, que permite hacer un seguimiento de la ruta y la posición actual del automóvil. Los datos se captaron a través de una tarjeta de adquisición de señales Arduino, se almacenaron en una base de datos y son visualizados a través de una aplicación móvil que presenta la información recibida de los diferentes sensores incorporados en la unidad de control electrónico (ECU). Este proyecto permite al usuario conocer los errores de funcionamiento del vehículo que pueden surgir, además de, contar con un plan de mantenimiento preventivo con el fin de evitar futuros fallos o anomalías.

Palabras clave: IOT, LPWAN, OBD, SIGFOX.

Abstract

Every revolution implies a change in the notion of our world. Of course, disruptive technologies such as the Internet of Things, use of the cloud and Big Data, make a big difference in the professional and personal field. Now is the perfect time, because most companies today are not implementing new technologies. Low Power Wide Area Networks are wireless communication technologies capable of transmitting data between a device and a base station-Gateway that can be separated by several kilometers with very low energy consumption. By their nature, these technologies are enabling the implementation of the largest IoT initiatives.

Being specifically designed for this environment, they allow the installation of tens or hundreds of nodes distributed over a large area, powered by batteries that last for years, and without requiring significant or expensive infrastructure.

The purpose of this project is the development of a communications interface for monitoring the variables of a vehicle, through the use of Sigfox technology. This interface allows the operator to know when vehicle parts need service or overhaul, to prevent premature damage to components and thus improve vehicle reliability and safety. In turn, the information obtained notifies the user about the speed, revolutions per minute and the real-time position of the vehicle through a geolocation, monitoring and control system of the vehicle's activity, which allows monitoring of the route and the current position of the car. The data was captured through an Arduino signal acquisition card, stored in a database and displayed through a mobile application that presents the information received from the different sensors incorporated in the electronic control unit (ECU). This project allow the user to know the vehicle's operating errors that may arise, in addition to having a preventive maintenance plan in order to avoid future falls or anomalies.

Key Words: IOT, LPWAN, OBD, SIGFOX.

Capítulo 1

Definición del Proyecto

Introducción

Hace 20 años, Internet se usaba principalmente como un instrumento para buscar y obtener información. En la década pasada hemos sido testigos de una nueva forma de usar de Internet, donde todo se ha convertido en social, transaccional y móvil.

El concepto de Internet de las cosas (IoT, del inglés *Internet of Things*) se concibe con la iniciativa de generar soluciones por medio de la interconexión de objetos o cosas con acceso a Internet y de esta manera enviar datos de interés con el propósito de reconocer y controlar el estado del medio ambiente. Con la expansión y el crecimiento global de IoT, se debe utilizar redes de área amplia de bajo consumo (LPWAN, del inglés *Low Power Wide Area Networks*).

En cuanto a las ventajas de IoT, está la utilización de tecnologías que ocupan bajos recursos por lo que las redes LPWAN pueden entrar en consideración como recurso principal para enfrentar muchos desafíos, como la creciente preocupación por la seguridad y privacidad de los datos, por ser redes que tienen una cobertura amplia y un bajo consumo de potencia, siendo así el IoT un factor fundamental para las organizaciones que pretendan ser líderes, en cualquier sector. Su uso ofrece un gran valor, particularmente en relación con los costes o el aumento de la productividad.

La integración de las comunicaciones, el control y el procesamiento de la información en varios sistemas de transporte es solo una de las muchas áreas en las que IoT puede ayudar. Las aplicaciones del IoT se extienden a todos los aspectos y propiedades de los sistemas de transporte, como ejemplo, el mismo vehículo en sí, la infraestructura, así como el usuario. La interacción dinámica entre cada uno de los componentes del sistema de transporte facilita las comunicaciones inter e intra vehiculares, *Smart parking, Smart traffic control*, sistemas de peaje electrónico, gestión de flotas y logística, control del vehículo, asistencia en carretera y predicción de fallas.

Es así que la propuesta del presente proyecto está orientado al monitoreo y gestión de datos propios del automóvil. Los vehículos tienen un puerto de entrada y salida llamado OBD (del inglés *On Board Diagnostics*), que se conecta a una unidad de control (ECU, del inglés *Engine Control Unit*) responsable de verificar y controlar los parámetros del vehículo.

Antecedentes

En general, la tecnología LPWAN funciona bien cuando los dispositivos deben transmitir pocos datos en un área grande y requieren una batería de larga duración durante un largo período de tiempo. Esto diferencia a LPWAN de otros protocolos propios de redes inalámbricas como RFID, Bluetooth, M2M celular y ZigBee.

En la actualidad distintas empresas y demás entidades se han establecido con el propósito de brindar servicios de electrónica automotriz para la detección de fallas, los dispositivos utilizados para realizar el diagnóstico tienen valores económicos muy altos, siendo así menos accesibles para los consumidores. Por ello se optó por el uso de software y componentes cuyos valores económicos son relativamente bajos puedan adaptarse a las necesidades de las personas y empresas, evidenciándose en el presente trabajo o proyecto de grado.

El uso y distribución de estas redes ha experimentado un crecimiento exponencial año tras año debido a la implementación de aplicaciones ampliamente relacionadas al Internet de las Cosas. Dado a su gran adaptabilidad, en este contexto es posible plantear y desplegar todo tipo de aplicaciones de monitoreo y gestión de datos.

Existen varios trabajos relacionados con el monitoreo y control de variables, como es el de Ramirez (2020) quién desarrolló una red de sensores conformada por dos estaciones de monitorización de grados de contaminación, orientadas a dos metodologías distintas (muestreo pasivo y activo) para la adquisición de las muestras, definiendo los contaminantes a monitorizar y evaluando los resultados que se presentan en una aplicación móvil y una página web.

Además, la monitorización de los objetos es indispensable sobre todo en el sector industrial en donde se busca tener todos los elementos y maquinaria controlados y en un buen estado; por ello la geolocalización en el sector del transporte es fundamental, al ser la seguridad uno de los motivos para su implementación. Bajo esta premisa Villavicencio et al. (2020) fueron capaces de desplegar un sistema que cuenta con un módulo de lectura y envío de datos que está integrado por un dispositivo OBD-II que incorpora una antena GPS, tarjeta SIM y un puerto OBD usado para conectar al vehículo. Este módulo de gestión y visualización de datos permite configurar todos los parámetros relacionados a la monitorización de los vehículos tales como eventos, alertas, rastreó y seguimiento del módulo de geo posicionamiento que forma parte del prototipo.

Otro trabajo de investigación relacionado con esta temática es el de Caizatoa y Méndez (2014) donde se diseñó un prototipo de monitoreo de vehículos empleando el estándar OBDII basado en el programa BASCOM AVR.EI/ELM327 que interpreta toda la información del vehículo y la convierte en datos seriales con el fin de informar al conductor de las averías del automóvil y que este pueda comprender los códigos de fallos relacionados con la ECU.

En el trabajo de investigación de Quito y Sarango (2021) se diseñó y construyó un prototipo de Internet de la Cosas de adquisición de datos para un sistema de diagnóstico a bordo de vehículos OBD-II que, es capaz de entregar información a un usuario final sobre el comportamiento y rendimiento en tiempo real y los códigos de fallas que pueden presentarse en un automóvil, información que puede ser consultada de forma local mediante una Tablet o remota mediante un servidor web.

Este prototipo fue diseñado con el objetivo de contar con un dispositivo IoT de bajo costo y largo alcance para que el usuario conozca el funcionamiento óptimo de su vehículo sin la necesidad de tener demasiados conocimientos técnicos automotrices y así poder anticiparse a cualquier daño o avería que pueda presentarse a largo plazo. De esta manera, este prototipo es implementado en un Raspberry PI-4 que cuenta con comunicación Bluetooth y GPRS, donde esta tecnología se usa para consultar los

datos propios del vehículo inmediatamente a la Unidad de Control del Motor a través del módulo ELM-327.

Justificación e Importancia

El mantenimiento del vehículo es esencial para la seguridad y el funcionamiento normal de los componentes del mismo. Además, mantener el vehículo en las mejores condiciones, no solo reduce el riesgo de accidentes debido al mal estado de los componentes del automotor, sino que también ayuda a reducir averías graves y costosas. Priorizar la seguridad vehicular en carretera parte de contar con automotores que se encuentren en óptimas condiciones, para ello, es recomendable realizar una revisión semanal del estado de sus componentes.

Si bien reconocemos la importancia del mantenimiento, a menudo nos olvidamos de integrar el mantenimiento como otra rutina empresarial que exige la seguridad de los equipos, vehículos y empresas, lo cual constituye una práctica fundamental.

Sin embargo, el mantenimiento adecuado de los vehículos sigue siendo una cuestión abierta para varias empresas que a menudo no realizan el mantenimiento preventivo y revelan una serie de fallas que amenazan tanto la seguridad del conductor como la del resto de usuarios en carretera.

Debido a lo expuesto con anterioridad, el presente proyecto cumple un papel importante ya que contribuye a la recolección de datos de ciertas variables propias de un vehículo, para lo cual se ha propuesto un sistema de monitoreo a bordo que permita a los conductores familiarizarse con los periodos correctos para reemplazar o inspeccionar ciertas partes de un vehículo. Por ello se consideró implementar la interfaz de comunicaciones con la tecnología LPWAN Sigfox y una aplicación móvil.

Alcance

El proyecto en mención busca recolectar la información de las variables propias de un vehículo, mediante la interfaz OBD cuyo sistema verifica el estado de todos los sensores involucrados en las emisiones del vehículo y se implementa una conexión alámbrica hacia la placa de desarrollo Thinxtra la

cual se conecta a la red de Sigfox y envía datos del estado del vehículo al Backend de Sigfox y posteriormente a una base de datos en la nube.

Se usaron los pines de energía que tiene la interfaz OBD para alimentar el dispositivo, para ello se conectó el conector OBDII correctamente al terminal del vehículo y se puso el coche en ignición. Esta operación se puede realizar sin ningún dispositivo externo que alimente al Arduino, dado que este se alimenta directamente de la batería del vehículo.

Los datos enviados son respaldados en el Backend de Sigfox y luego se almacenan en una base de datos de telemetría que está disponible en la nube.

Se diseñó una aplicación web y una aplicación para dispositivos móviles Android capaz de conectarse a la base de datos a través de Internet, y recuperar el estado actual del vehículo, sus variables y la ubicación del mismo. Las aplicaciones cuentan con una interfaz gráfica de usuario y proporcionan información sobre el trayecto, recorrido y su ubicación a través de Google Maps.

La recolección de datos es fundamental para llevar a cabo el análisis de los mismos en esta investigación. Parte del análisis consiste en evaluar la totalidad de los datos que se tienen para posteriormente reducirlos y satisfacer los objetivos de este proyecto.

Una vez se haya recopilado la información es de gran importancia cerciorarse de que el análisis de los datos permita alcanzar los objetivos propuestos al inicio de la investigación. Así los objetivos más importantes a considerar a partir el análisis de la información obtenida serian: Planificar periodos de mantenimiento, realizar operaciones de diagnóstico, verificar niveles máximos permisibles de contaminación, proponer un sistema de mantenimiento preventivo y realizar un análisis comportamental del conductor por motivos de seguridad.

Objetivos

Objetivo General

- Diseñar e implementar una interfaz de comunicaciones con tecnología LPWAN Sigfox que

permita la monitorización y análisis de los parámetros de funcionamiento de un vehículo y gestionar los datos obtenidos en una plataforma IoT.

Objetivos Específicos

- Diseñar una aplicación web y una aplicación para dispositivos móviles Android, con el fin de interactuar con una plataforma dedicada para el usuario que permita realizar diagnósticos en vehículos equipados con la unidad de control con interfaz OBD.
- Integrar la tecnología LPWAN al sistema de diagnóstico a bordo en vehículos OBD.
- Desarrollar una base de datos en la Internet para almacenar la información reportada por el sistema de diagnóstico OBD.
- Implementar el sistema mediante la tecnología Sigfox a un vehículo y presentar la información obtenida por la interfaz OBD, adicional generar una base de datos para registrar los parámetros.
- Analizar el desempeño del sistema de diagnóstico a partir de los resultados obtenidos del sistema OBD y de las mediciones de las variables del vehículo

Capítulo 2

Marco Teórico

Internet of Things

El Internet de las Cosas (Iot, del inglés Internet of Things), también conocido como Internet de los Objetos y más comúnmente abreviado como IoT, es un supraconcepto que, en opinión de Barrio (2018), caracteriza la próxima gran etapa en el desarrollo de las aplicaciones de Internet, su expansión más allá de la comunicación entre las personas, o entre las personas y todo tipo de contenido digital, que se extiende a millones de objetos cotidianos.

Los sistemas IoT incorporan en sus prestaciones: la adquisición de datos de sensores y la entrega de órdenes y mandatos a dispositivos que interactúan y están asociados o forman parte del mundo real. También pueden reconocer eventos y cambios, y a su vez son capaces de reaccionar a estos de forma autónoma y apropiada. El IoT se equipara mayoritariamente con electrodomésticos y bienes de consumo, como los automóviles inteligentes. Por lo tanto, muchas de las necesidades iniciales a cubrir se han centrado en los productos de consumo.

Figura 1

Esquema del IoT

Nota. Representación de como el IoT es capaz de ofrecer servicios mediante la interconexión de objetos (Físicos y virtuales). Tomado de *Internet de las cosas*, por Barrio, M., 2018, Editorial Reus.

Con la denominación “inteligente” presente en cada vez más dispositivos, no es de extrañar que el IoT se haya podido posicionar como una de las tecnologías y tendencias más relevantes para la industria TI y su futuro.

Dispositivos IoT

Cuando hablamos de dispositivos IoT, a menudo describimos cosas como sensores ambientales, electrodomésticos conectados, rastreadores de vehículos o incluso maquinaria de línea de montaje. Si bien un dispositivo IoT puede ser cualquier dispositivo electrónico capaz de comunicarse con Internet, generalmente no nos referimos a un teléfono móvil o una computadora de propósito general.

Muchos de estos dispositivos no se construyeron originalmente teniendo en cuenta la conectividad a Internet y deben ser modificados con soluciones de postventa para conectarse. Sin embargo, las capacidades de IoT se incorporan cada vez más a los nuevos dispositivos, donde pueden reducir significativamente los costos y mejorar la funcionalidad (Buyya & Vahid, 2016).

Todos los dispositivos IoT deben comunicarse. Algunos dispositivos solo se encargan de enviar información; muchos otros envían y reciben. Si bien algunas comunicaciones con dispositivos similares son directas, las comunicaciones remotas a menudo necesitan pasar por una puerta de enlace para llegar a su destino.

CAN Bus

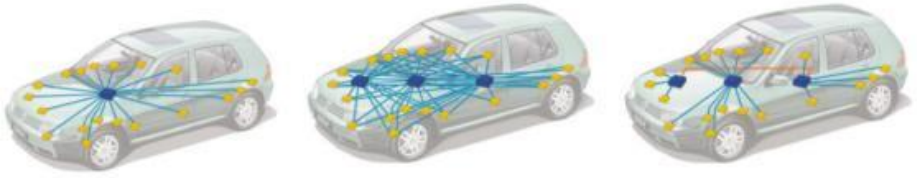
Se conoce a la tecnología CAN como un protocolo de comunicaciones habitualmente usado en sistemas y entornos con requisitos de uso en tiempo real. CAN surgió por la necesidad de conectar cada vez más dispositivos electrónicos en el interior de los vehículos, debido a sus garantías, es utilizado frecuentemente en el sector de la industria automotriz, donde la fiabilidad de las comunicaciones es de suma importancia para el buen funcionamiento de los sistemas (Martínez, 2017).

Debido a que el protocolo CAN utiliza un único bus de comunicación compartido por todos los periféricos y dispositivos internos, eliminó la necesidad de instalar una conexión punto a punto con

cada dispositivo. El uso y ejecución de este modelo de comunicaciones se ha traducido en una reducción de la cantidad de conexiones entre dispositivos internos que se requerían para mantener todos los elementos en constante comunicación.

Figura 2

Disminución de las conexiones y cableado necesario.



Nota. La figura representa como el protocolo CAN reduce la cantidad de conexiones necesarias. Tomado de *Introducción a CAN bus: Descripción, ejemplos y aplicaciones de tiempo real,* por Martínez, A., 2017.

Ventajas del Protocolo CAN Bus

Una de las mayores fortalezas de la tecnología CAN, que le ha permitido mantenerse en el tiempo a pese al surgimiento de otros protocolos de comunicación capaces de transmitir datos a mayor distancia o con mayor velocidad, son las garantías de comunicación que brinda, garantías que son fundamentales a la hora de desarrollar todo tipo de sistemas con características de tiempo real y alta integridad (Martínez, 2017).

- Tiene la capacidad de retransmitir automáticamente tramas de datos erróneas mediante herramientas especializadas en la detección de errores de envío de datos.
- La capacidad de diferenciar entre errores de transmisión específicos o fallas provocadas por la falla de un nodo, agregando la capacidad de desconectarse para evitar la saturación de la red.
- Priorizar los mensajes y asegurar la latencia en su entrega. Esta cualidad posibilita que el protocolo CAN sea ampliamente requerido en el campo de los sistemas en tiempo real.
- Asegurar consistencia en los datos.

- La configuración de la red cuenta con una gran flexibilidad, tanto en número de nodos como en su disposición, se puede agregar o quitar nodos dinámicamente sin afectar el protocolo.
- Es posible conectar a la red CAN hasta 110 nodos.

Estandarización del protocolo CAN

Se define al protocolo CAN en el estándar ISO 11898, este estándar comprende diferentes normas específicas para diversos tópicos del protocolo y sus distintos tipos de funcionamiento. De manera que, la norma la norma ISO 11898-3 se encarga de estandarizar el protocolo CAN de baja velocidad tolerante a fallos, y la norma ISO 11898-2 se ocupa del protocolo CAN de alta velocidad, capaz de alcanzar velocidades máximas de 1 Mbps (ISO, 2016).

Tabla 1

Velocidades para la transmisión del CAN bus de acuerdo a la longitud.

Longitud del bus	Velocidad	Tiempo máximo de transmisión*
25 m	1 Mbps	129 µs
100 m	500 Kbps	258 µs
500 m	125 Kbps	1032 µs
1000 m	50 Kbps	2580 µs

Nota. *mensajes de 129 bits de longitud. Tomado de *ISO Car Informatics*, por ISO, 2016.

Un módulo CAN está compuesto de los siguientes elementos básicos:

Controlador. Es el encargado de gestionar el montaje de las tramas CAN, detección de colisiones y comprobación de errores en la transmisión o en otros nodos.

Transmisor-Receptor. Conocido también como transceptor. Este módulo se encarga de la sincronización, codificación/decodificación de los mensajes en el bus, control de los niveles de la señal o del control de acceso al medio.

Cable CAN bus de datos. Es el canal por donde la información fluye y está conformado por dos tipos de cables entrelazados (CAN-*High* o H y CAN-*Low* o L).

Elementos finales del bus de datos. Resistencias presentes en los extremos de los cables H y L.

Debido a su completa independencia de los otros nodos, el transceptor y el controlador CAN no necesitan asignar ningún recurso para controlar las gestión de comunicaciones, el acceso al medio o colisiones. Aun cuando algunos microcontroladores cuentan con módulos CAN en un único encapsulado, internamente son circuitos independientes en su gran mayoría.

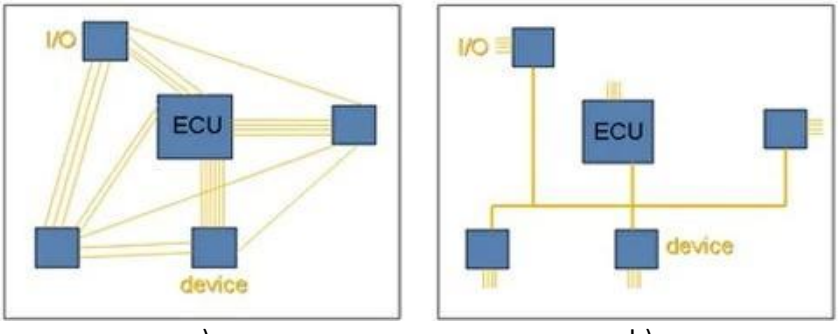
Todos los dispositivos conectados al bus son capaces de enviar mensajes y todos los nodos conectados al mismo podrán receptarlos. Para distinguir los tipos de mensajes, estos están asociados a un identificador único. De esta forma, cada nodo puede procesar o rechazar los mensajes.

Beneficios del protocolo CAN

El mayor beneficio del CAN bus es que el cable de datos funciona como un gran canal en el que se transmite la información proveniente de todos los dispositivos propios del automóvil. Es así que este sistema reemplaza a los modelos tradicionales de comunicación que utilizaban gran cantidad de cables, que en consecuencia propiciaba que los componentes del vehículo y su funcionamiento sean más complicados de diagnosticar, configurar y de reparar.

Figura 3

Esquema de un sistema con y sin CAN



a) b)

Nota. El gráfico muestra un esquema donde las conexiones de los dispositivos del automóvil se ven reducidas al usar el protocolo CAN. En a) Sistema sin CAN, en b) Sistema con CAN. Tomado de *Introducción a CAN bus: Descripción, ejemplos y aplicaciones de tiempo real,* por Martínez, A., 2017.

Como resultado, el vehículo es más ligero y económico, y la comunicación de los sensores es más rápida y directa gracias al sistema CAN Bus.

Capa 1 de Nivel físico del protocolo CAN

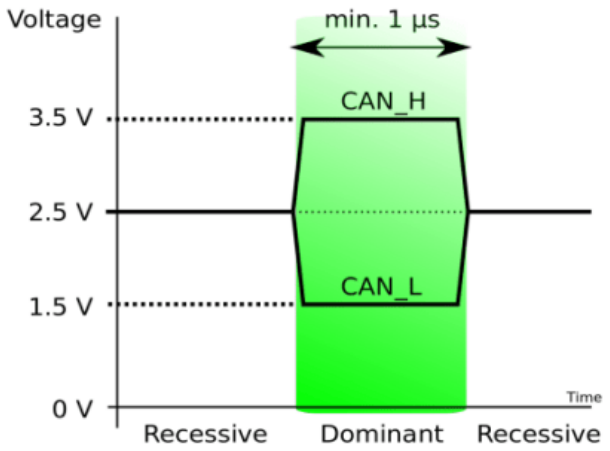
Su topología o configuración tiene un bus único donde se conectan todos los dispositivos en una misma red, en la que solamente se requiere dos cables trenzados con una impedancia de 120 ohms, para establecer esta conexión.

Las señales de estos cables se nombran CAN *low* y CAN *high* y dependiendo de la tensión de cada señal, se determina el modo en el que el bus se encuentra trabajando, en modo dominante, cuando existe una diferencia de potencial entre los cables de 1.5 V y en modo recesivo, cuando ambos cables tienen el mismo nivel de voltaje. El objetivo principal de este método de comunicación es ofrecer una mejor protección contra las interferencias electromagnéticas (García, 2020).

Debido a que la lectura de los bits depende de la diferencia de potencial eléctrico entre los dos cables trenzados, esta protección se utiliza para conseguir que, aunque exista alguna variación de señal en los cables, la tensión entre ambos se mantenga constante.

Figura 4

Modos de operación de acuerdo a la tensión en el CAN Bus



Nota. Modos de trabajo de acuerdo a los niveles de tensión del CAN bus. Tomado de *Diagnóstico de fallas a través del CAN BUS,* por García, H., 2020.

Control de acceso al medio (MAC)

Entre las características del protocolo CAN existe una que se denomina arbitraje, mediante la cual es posible controlar el acceso al medio por parte de los nodos, evitando así posibles colisiones en las comunicaciones. Esta característica junto a otras constituyen el control de acceso al medio establecido por el protocolo CAN.

Formato de tramas CAN

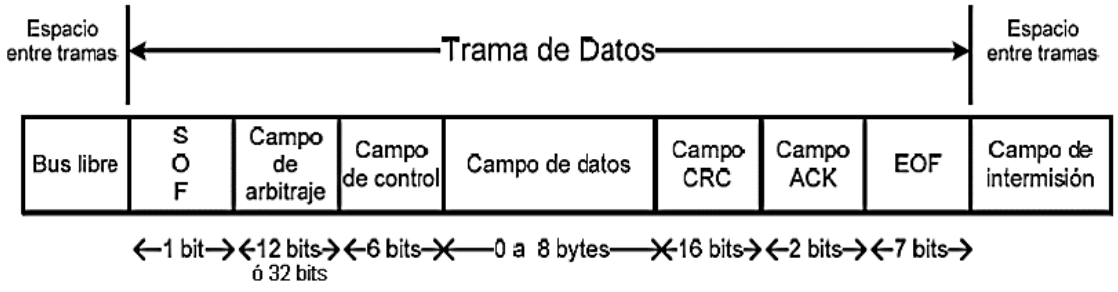
El protocolo CAN incorpora en su operación distintos formatos de tramas, cada una de ellas está orientada a un propósito definido dentro del funcionamiento del protocolo CAN (Encinas et al., 2009). Las siguientes son las más importantes:

Trama de datos. Son tramas que son enviadas y recibidas por uno o más nodos y se utilizan en el flujo de información entre ellos.

Una trama de datos CAN puede tener un formato extendido con un identificador de 29 bits o formato base con identificador de 11 bits.

Figura 5

Campos de la Trama de Datos



Nota. Formato de un mensaje o trama de datos y los campos q la conforman. Tomado de *Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores*, por Encinas et al., 2009.

A continuación se detalla la función de cada campo que conforma la trama de datos.

Campos SOF (Start of Frame) y EOF (End of Frame). Son delimitadores de la trama.

Campo de Arbitraje. Se encarga de definir la prioridad de un mensaje cuando dos o más nodos pretenden acceder al bus. En el formato estándar, el campo comprende un identificador de 11 bits y un bit RTR (Remote Transmission Request) con el que se identifica a la trama como de datos o remota. En el formato extendido, el campo comprende un identificador de 29 bits, un bit SRR (Substitute Remote Request), un bit IDE (Identifier Extension) y un bit RTR.

Campo de Control. Indica el número de octetos que comprenderá el campo de Datos.

Campo de Datos. Contiene los datos de la trama cuya la longitud viene dada por el código de longitud de datos o DLC.

El campo CRC. También conocido como campo de verificación por redundancia cíclica, verifica que los datos fueron transmitidos correctamente.

Campo de confirmación (ACK). Es donde el nodo receptor indica recepción correcta del mensaje, colocando un bit dominante en el flag ACK de la trama.

Trama de error. Tramas utilizadas cuando uno de los nodos de la red encuentra un error en alguno de los mensajes transmitidos por otros nodos. Infringen las reglas y estándares establecidos para el formato de tramas del protocolo CAN.

Trama de petición remota. Se utilizan para realizar solicitudes de envío de información a otro nodo. Para ello se remite una trama cargada con el identificador del nodo del que se necesita el envío de información, solicitándola, éste recibe el ID y envía esta información en una trama de datos.

Trama de Sobrecarga. Obliga a que los nodos prolonguen el tiempo de transmisión entre tramas sucesivas (Remotas y de Datos).

Espacio entre trama. Las tramas Remotas y de Datos se separan entre sí por esta secuencia predefinida. Está conformada por tres campos: Bus libre, Intermisión y Transmisión suspendida.

Estándar OBD II

OBD II es el protocolo de segunda generación de diagnóstico a bordo de vehículos (del inglés *On*

Board Diagnostics) que, en relación a sus predecesores el OBD I y OBD, incorpora la estandarización y perfecciona el sistema de diagnóstico del automóvil utilizado para detectar fallos mecánicos, químicos y eléctricos, que en consecuencia pueden provocar emisiones de gases contaminantes al entorno y para notificar al usuario de cualquier daño, deterioro o anomalía en los componentes que pueda presentar el vehículo.

Cuando los niveles de emisión de gases sobrepasan los umbrales establecidos, el sistema OBD II notificara al usuario. Asimismo, este sistema de diagnóstico se encarga de verificar que todos los sensores relacionados con la inyección de combustible al motor y la admisión funcionen correctamente. De manera que es necesario que los sensores localizados antes y después del sistema de escape (específicamente el catalizador) presenten un rendimiento químico óptimo.

Si el sistema OBD II detecta una falla en el automóvil, encenderá instantáneamente la luz indicadora de falla “MIL” en el instrumento combinado. Para este propósito, existen cientos de protocolos de comunicación en el campo automotriz, sin embargo, a partir de la última década se estandarizó el manejo del conector de diagnóstico de OBD-II con la normativa J1962 (Jhou et al., 2013).

Las normativas que rigen en el sistema de diagnóstico OBD II son mucho más extensas que las que se han definido en el sistema OBD y están estrechamente relacionadas a otras normativas como las SAE e ISO.

Conector DLC OBD II

Es un dispositivo que permite establecer la comunicación con todas las computadoras y módulos del vehículo. Mediante este conector (DLC OBD II, del inglés *Data Link Connector OBD II*) se establecerá la comunicación con cada uno de los módulos presentes en el automóvil, y se rige de acuerdo a la normativa ISO-15031-3 (ISO, 2016).

El conector DLC OBD II además de ser el traductor entre el automóvil y el técnico automotriz, se convierte también en una fuente crucial de información que reciben los módulos cuando son

programados ya que ahí generalmente es donde se hacen los ajustes de la mariposa de aceleración y la programación de llaves que permitan abrir y cerrar los cilindros electrónicos de frenos EPB.

Mayoritariamente está localizado en el compartimiento del conductor por debajo del tablero o en el compartimento del acompañante.

Figura 6

Conector OBD II

Nota. La figura muestra los pines y la distribución del conector OBD II. Tomado de *Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores*, por Encinas et al., 2009.

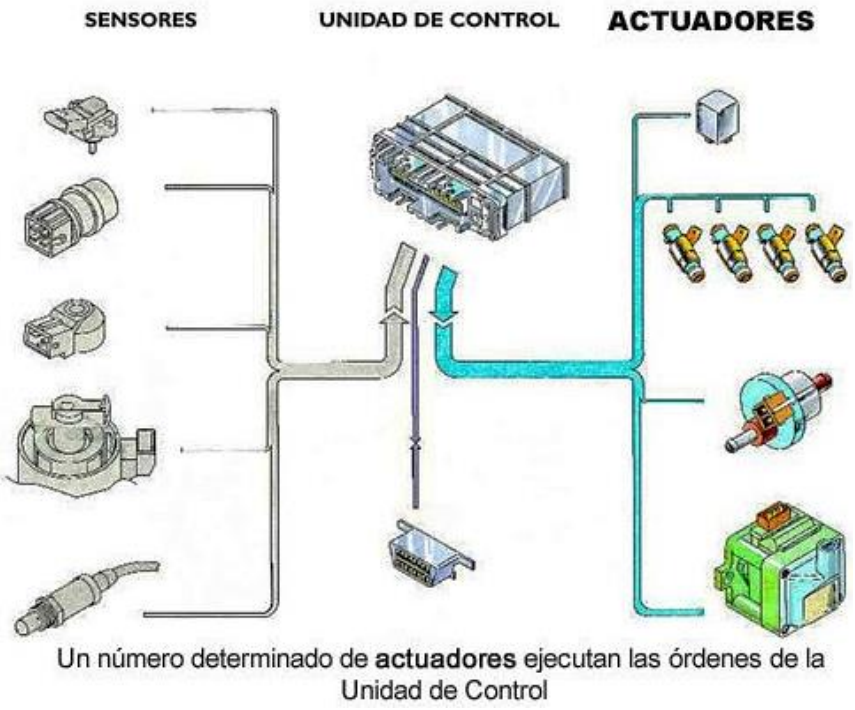
Unidad de Control Electrónico

La ECU es la unidad de control electrónico que regula el motor cuyas siglas en inglés significan “Engine control Unit”. Es un aparato electrónico que usualmente está conectado a una serie de sensores capaces de brindar información a la unidad central, que luego envía las instrucciones necesarias a los actuadores para transformar la información inicial y ejecutar sus comandos (Paulino, 2022).

ECU gestiona varios aspectos del funcionamiento del motor como los de combustión interna del mismo.

Figura 7

ECU automotriz y su conexión con los sensores y actuadores



Nota. Es posible evidenciar el comportamiento de la Unidad de Control, sensores y actuadores encargados de ejecutar las órdenes dispuestas por la ECU, y cómo están relacionados unos con otros. Tomado de *Encendiendo tu motor con tu ECU Programable,* por Paulino, N., 2022, Alphax.

Existen diversos tipos de Unidades de control entre ellas:

Las ECU más sencillas que solamente se encargan de controlar la cantidad de carburante que es inyectado en cada uno de los cilindros durante los ciclos del motor y las ECU más avanzadas que se encargan de recibir, encausar y enviar toda la información propia de los sensores con el propósito de valorar:

- La relación de aire/combustible.
- El punto de ignición del combustible
- Tiempo variable de cierre y apertura de todas las válvulas de admisión de aire o escape de gases.

- Velocidad en régimen ralentí o marcha mínima.
- Control del sistema de encendido o encendido electrónico (inyectores, encendido transistor, bobinas).
- Revoluciones por minuto que el motor puede alcanzar en su máximo desarrollo.
- Regulador de baja presión en el combustible: La utilidad del regulador de presión de combustible es básicamente controlar y mantener constante la presión del combustible aumentando el tiempo en el que la bujía actúa para compensar una pérdida en la presión del combustible.
- El control de la sincronización entre los movimientos de árbol de levas o árboles de levas con el cigüeñal

GPS

Conocido como “Sistema de Posicionamiento Global”, es una constelación de al menos 24 satélites que sirve como un sistema de navegación por satélite. La características de este sistema permiten que funcione en cualquier lugar del mundo, condición climática, a todas horas, y sin cargos de de configuración, uso y suscripción.

Los satélites GPS que componen este sistema circundan la Tierra dos veces al día en una órbita precisa. Los satélites envían una señal única y parámetros orbitales que permiten a los dispositivos GPS decodificar y estimar la ubicación exacta de un satélite. Los receptores GPS determinan la ubicación precisa de un individuo utilizando estos datos y la trilateración, una técnica matemática que utiliza la geometría de triángulos para determinar las posiciones relativas de los objetos. Básicamente, el receptor GPS es el encargado de medir la distancia existente a cada satélite de acuerdo al tiempo que tarda recibir una señal transmitida (Sánchez, 2012).

Para determinar la posición bidimiensional (longitud y latitud) y el movimiento de pista de un objeto, un receptor GPS debe estar bloqueado a la señal de por lo menos tres satélites, para que el

receptor sea capaz de determinar la posición tridimensional (longitud, latitud y altitud) de un objeto debe tener en línea de vista cuatro o más satélites. Habitualmente, un receptor GPS puede rastrear 8 o más satélites, pero eso depende del lugar en el que se encuentre el objeto y la hora.

Una vez que se ha calculado su posición, la unidad GPS puede estimar otro tipo de información, como por ejemplo:

- Velocidad del objeto
- Rango de movimiento y Recorrido
- Distancia hasta el destino

Los satélites que conforman la constelación GPS emiten por lo menos dos señales de radio de baja potencia, estas señales viajan por su línea de vista y contienen esta información:

- Código pseudoaleatorio. Permite identificar qué satélite de la constelación está transmitiendo información.
- Datos de efemérides. Datos necesarios para definir la posición de cualquier satélite, además de proporcionar información considerable acerca de la fecha y la hora actual, y la salud del satélite.
- Datos del almanaque. Muestran la información orbital de cualquier satélite de la constelación, además de indicarle al receptor GPS dónde se encuentra cada uno de los satélites del sistema en cualquier instante.

Redes de área amplia de baja potencia

Una Red de Área amplia de baja potencia corresponde a un tipo de red de telecomunicaciones inalámbricas diseñada para crear redes de sensores inalámbricos, esta red ofrece servicio de infraestructura y puede ser de carácter público o privado.

Como su nombre lo indica LPWAN es una red de área amplia de baja potencia. Es un protocolo de transmisión inalámbrica de datos que hoy en día es la base para la mayoría de las implementaciones de IoT.

Entre las ventajas del uso de este tipo de redes está su bajo consumo energético, por lo que pueden implementarse en proyectos que utilizan baterías para alimentar dispositivos que deben tener una autonomía medida en meses o años (Sampaulo, 2020).

En su mayoría, las redes LPWAN son usadas para suministrar una transferencia de datos con una baja velocidad entre un servidor central y puntos finales, mejor conocidos como nodos. El flujo de información va en la dirección del enlace ascendente (de los nodos hacia el servidor central) con confirmaciones de recibo, configuración y actualizaciones del sistema utilizando la conectividad del enlace descendente o downlink. Mayoritariamente las aplicaciones LPWAN se centran en entregar información sobre objetos estáticos (autos estacionados o electrodomésticos) o cuasiestáticos (animales) (Wireless, 2015).

Sigfox

Es una red propietaria, ofrecida por una compañía originaria de Francia. A día de hoy, es considerada una de las redes LPWAN más grandes del mundo y se caracteriza por usar una frecuencia sin licencia en las bandas de 868 MHz o 902 MHz. Se distingue por usar una transmisión de radio de banda ultra estrecha y su cobertura es capaz de cubrir grandes distancias, con una tasa de transferencia o velocidad real de transporte de datos baja (Sampaulo, 2020).

Sigfox es una plataforma que puede enviar 12 bytes de información y es capaz de receptar 8 bytes. Cantidad suficiente para comunicar detalles acerca de la geolocalización, temperatura, uso de sensores, o un heartbit (0 bytes). Entre las características de funcionamiento y tecnologías que usa:

- Usa codificación de Desplazamiento de Fase Binaria Diferencial “DBPSK”.
- Usa Codificación de Desplazamiento de Frecuencia Gaussiana “GFSK”.
- Tiene tecnología UNB.
- Baja frecuencia
- Es propietario, por lo que es necesario contratar con los proveedores.

- No hay circuitos del receptor, por tanto es necesario menos energía.

Existen distintas aplicaciones que requieren este tipo de tecnología basada en comunicaciones inalámbricas. Necesita un punto final de radio económico y una estación base más sofisticada para gestionar la red. Está dirigida principalmente a las aplicaciones que requieren de una baja velocidad de transmisión de datos y un bajo consumo de energía. Necesita una cantidad mucho menor de antenas en contraste con las redes celulares tradicionales como GSM-CDMA. Sigfox ha creado un protocolo ligero a medida para manejar los mensajes pequeño donde enviar menos datos significa un menor consumo energético y, por ende, una vida útil de la batería más larga.

Al utilizar la modulación UNB, Sigfox funciona en los 200 kHz de la banda sin licencia disponible públicamente para así poder intercambiar mensajes de radio por el aire donde cada uno de los mensajes tiene un ancho de 100 Hz y es enviado a una velocidad de 100 o 600 bps dependiendo de la región. Por consiguiente, alcanza largas distancias y es muy resistente al ruido (Sigfox, 2017).

La densidad de celdas de la red Sigfox se basa en un rango promedio de 30 a 50 km en áreas rurales y de 3 a 10 km en áreas urbanas donde hay más obstáculos y el ruido existente es mayor. Las distancias pueden ser mayores para los nodos al aire libre donde los mensajes de Sigfox pueden viajar distancias mayores a 1000 km (Samsung, 2016).

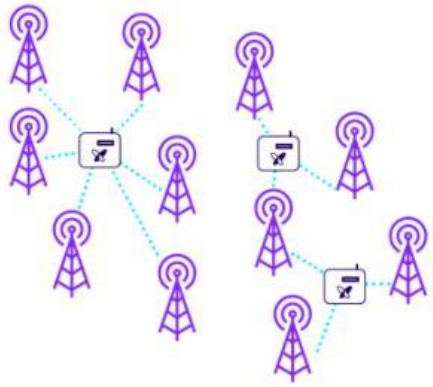
Arquitectura de Sigfox

Sigfox emplea una infraestructura de estaciones base, cada una con una antena correspondiente y otros componentes típicos de la red. El mensaje puede ser recibido por cualquier estación base dentro del rango (típicamente tres estaciones base), es decir, se usa una recepción cooperativa, demostrando que los dispositivos no solo se encuentran asociados a una estación base.

Los objetos se comunican con las estaciones base, que escuchan permanentemente el espectro, e interpretan todas las señales que reciben, para enviarlas a los sistemas de soporte de Sigfox.

Figura 8

Recepción Cooperativa



*Nota. R*ecepción cooperativa de estaciones base. Tomado de *Sigfox Techinical Overiew*, por Sigfox, 2017, Sigfox (https://www.Sigfox.com/).

En contraste con distintas redes móviles, para la red Sigfox, los objetos no están ligados solamente a una estación base específica, sino que un mismo paquete de datos es enviado tres veces (diversidad en tiempo y frecuencia) a distintas estaciones base que monitorizan el espectro y buscan las señales UNB para demodularlas.

La comunicación es bidireccional aunque por lo general son los nos los que emiten información a la red. Esto significa que se puede configurar al módulo Sigfox para que al enviar un mensaje de "uplink" sea pueda solicitar información desde los servidores de Sigfox, a esto se le denomina mensaje de enlace descendente. Sigfox (2017) detalla la función de algunos componentes dentro de la arquitectura como:

EP (End Point). Se encarga de transmitir mensajes a la red Sigfox.

BS (Estación Base). Son instalaciones directamente conectadas con el servidor o Backend por medio de conexiones punto a punto y receptan los paquetes de datos de los End Point. Tambien son los encargados de modular y demodular los mensajes para posteriormente enviarlos al Backend.

Backend Sigfox. Transmite todos los paquetes mediante Callbacks hacia una base de datos para su presentación y visualización en plataformas externas o aplicaciones del usuario.

Plataforma del Usuario. Recibe los mensajes de los End Point a través de la red de Sigfox.

Figura 9

Arquitectura de la red Sigfox

Nota. El gráfico representa la arquitectura de una red Sigfox especializada en la obtención y gestión de información propia de vehículos. .

La transmisión de datos se caracteriza por ser asíncrona entre el objeto y la red y no requiere ninguna inicialización previa (emparejamiento). Es necesario que un dispositivo tenga un módulo de comunicación compatible con Sigfox (tarjetas, chips o kits) para que funcione con esta red.

El consumo de energía usado al enviar mensajes desde un nodo hacia el servidor se encuentra en el rango de 10 a 50 mA, mientras que en estado de reposo es aproximadamente 0 mA, lo que permite una autonomía de la batería entre 10 y 15 años.

Modos de Operación de Sigfox

Los dispositivos transmiten información utilizando frecuencias en la banda ISM sin licencia, de forma que, se utiliza la frecuencia de 920 MHz para América del sur o 868 MHz para Europa. Otros países y regiones también cuentan con bandas de frecuencia de banda ISM sin licencia semejantes, es así que para la región América del norte se maneja una frecuencia de 902 MHz. En la tabla 2 podemos ver la clasificación de las configuraciones de radio así como la frecuencia en la que trabajan las

configuraciones destinadas exclusivamente a los países donde red Sigfox se encuentra activa y trabajando. A partir de esta información es posible deducir que Ecuador pertenece a la configuración RC4= 920 MHz.

Tabla 2

Configuraciones de radio y sus respectivas frecuencias

Configuración de Radio	Frecuencias	Regiones
RC1	868 MHz	Europa, Ultramar Francia, Oriente Medio y África
RC2	902 MHz	Canadá, México, USA, Brasil, Puerto Rico.
RC3	923 MHz	Japón
RC4	920 MHz	América Latina y Oceanía
RC5	923 MHz	Corea del Sur
RC6	865 MHz	India
RC7	868 MH<	Rusia

Nota. Frecuencias y regiones para cada configuración de Radio. Tomado de *Sigfox Techinical Overiew*, por Sigfox, 2017, Sigfox (https://www.Sigfox.com/).

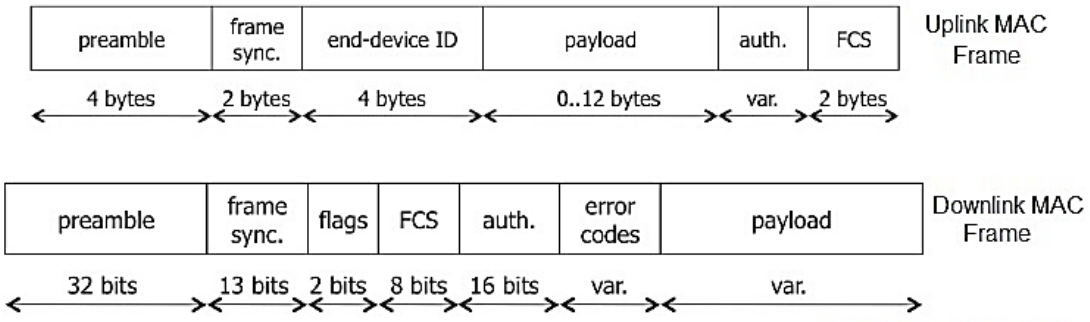
Mensajes Sigfox

Los mensajes de Sigfox están diseñados para ser relativamente pequeños, optimizados para la adquisición de información de sensores y necesitar solamente una pequeña cantidad de energía para poder transmitirlos. Sin contar su encabezado, la carga útil de Sigfox está limitada a 12 bytes.

Los mensajes enviados (uplink) no pueden tener más de 12 bytes y pueden ser enviados durante 6 segundos a una velocidad de 100 bps, por el contrario los mensajes recibidos (downlink) no pueden sobrepasar los 8 bytes. Es posible enviar hasta 140 mensajes por día, es decir, 6 mensajes por hora y recibir hasta 4 paquetes por día, garantizando así una ocupación de la banda del 1% máximo.

Figura 10

Tramas uplink y downlink de la red Sigfox

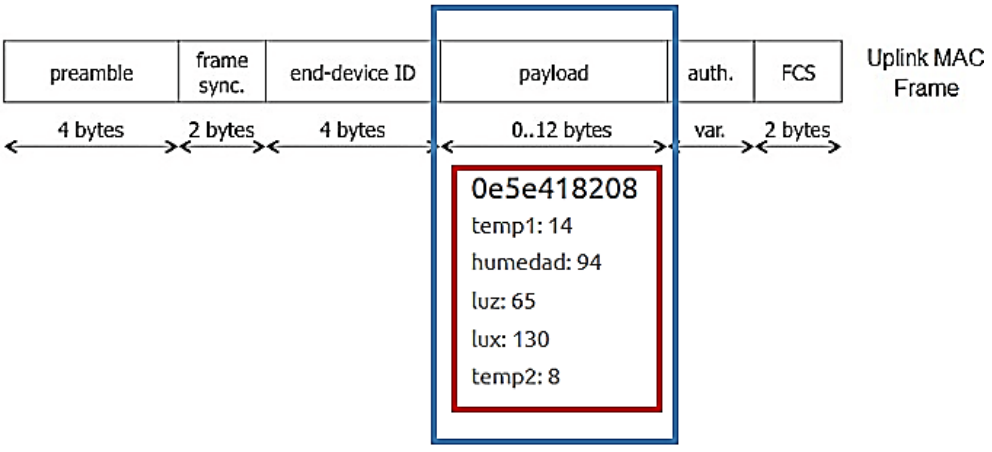


Nota. La figura muestra cómo están definidas las tramas de enlace ascendente y descendente de la red Sigfox. Tomado de *LPWAN, las redes del* IoT*,* por Sampaulo, P., 2020, RedGPS.

El preámbulo (preamble) de las tramas se compone de símbolos predefinidos que normalmente se utilizan para la sincronización de la transmisión. Los tipos de tramas transmitidas se definen mediante campos de sincronización de tramas. El FCS es la secuencia usada en la verificación de tramas (FCS: Frame Check Sequence) útil en la detección de errores.

Figura 11

Trama de mensaje compuesta por una carga útil enviada al Backend



Nota. La figura muestra cómo está compuesta la carga útil o payload. Tomado de *LPWAN, las redes del* IoT*,* por Sampaulo, P., 2020, RedGPS.

Adicionalmente, como se muestra en la Figura 11, el mensaje uplink que nos

interesa enviar tiene un lugar en el campo payload, que tiene un rango de tamaño de 0 a 12 Bytes. El payload contiene las variables que queremos administrar en nuestra aplicación, como se muestra en el cuadro rojo, donde se concatenan en pares de esta manera: **"0d6d527309" = 0d-6d-52-73-09**

Para este caso cada dupla representa una variable, que tomándolas de derecha a izquierda (formato little-endian) y convirtiéndolas a números enteros nos quedaría de la siguiente manera:

- temp2: 9 → dupla: x09
- lux: 115 → dupla: x73
- luz: 82 → dupla: x52
- humedad: 109 → dupla x6d
- temp1: 13 → dupla: x0d

Figura 12

Payload cargado de diferentes parámetros

0	1	2	3	4	5	6	7	8	9	10	11
Lat.	Lat.	Lat.	Lat.	Lon.	Lon.	Lon.	Lon.	Vol.	Sats.	Acq.	Spd.

Nota. Ejemplo de una estructura de 12 bytes es posible enviar un conjunto de parámetros donde se detallan: coordenadas GPS, velocidad, hora y voltaje de batería. .

Cada variable tiene un tamaño diferente, por lo que es importante definir su tamaño así:

Tabla 3

Tamaños de paquetes de acuerdo a su área de aplicación

Tipo	Tamaño
Coordenadas GPS, 3 metros de precisón	6 bytes
Temperatura de 0˚C a 200˚C	2 bytes
Velocidad hasta 200 km/h	1 bytes
Estado de un objeto	1 bytes

Nota. La tabla muestra algunos ejemplos de tamaños de paquetes de acuerdo a su área de aplicación. Tomado de *Particularidades de la red Sigfox*, por Martres, P., 2019.

Como se mencionó anteriormente, la recepción es cooperativa (diversidad espacial), donde tres

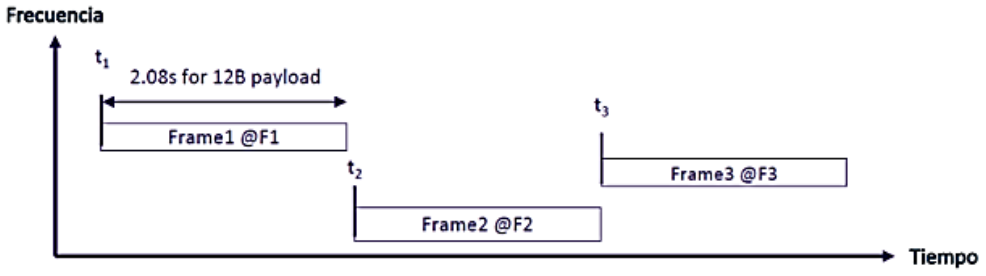
estaciones base reciben el mismo mensaje y el acceso a la red es aleatorio, es decir está habilitado para frecuencias diferentes.

Recepción de Mensajes. Cada uno de los dispositivos en una red Sigfox posee un ID único. El ID es utilizado para firmar y enrutar mensajes, así como autenticar el dispositivo Sigfox. Una de las características de la comunicación Sigfox es que utiliza el modelo disparo y olvido. Es decir los mensajes no son reconocidos por el receptor, al contrario, el dispositivo envía un mensaje tres veces en tres frecuencias diferentes en tres momentos diferentes como se muestra en la figura 13, esto garantiza la integridad de la transmisión de un mensaje.

El modelo disparo y olvido puede asegurar que el mensaje se recibió realmente, por lo que depende del emisor hacer lo posible para garantizar su transmisión, no existe tráfico de sincronización ni señalización entre la red y el dispositivo para ahorrar ancho de banda y consumo de energía.

Figura 13

Paquete triplicado para garantizar su integridad



Nota. Frame enviado tres veces en tres frecuencias diferentes. Tomado de *Particularidades de la red Sigfox*, por Martres, P., 2019.

Backend de Sigfox

En el proceso de comunicación de la red Sigfox, existen dos extremos de comunicación: los dispositivos o nodos transmisores, que están conectados a la red, y la plataforma Backend o punto final de la comunicación, encargada de recibir mensajes y procesarlos para generar resultados (Sigfox, 2017).

Es así que entre los servicios que ofrece Sigfox, existe Sigfox Cloud, que comprende una

aplicación web conocida como Sigfox Backend. Desde esta plataforma, es posible gestionar los nodos transmisores conectados a la red y visualizar los mensajes transmitidos por los mismos, además configurar la integración de los datos. Este servicio permite redirigir todo el volumen de información que llega al Backend a cualquier aplicación de un servidor externo o centro de procesamiento de datos.

Para tratar los datos que recoge el Backend de Sigfox existen dos maneras, la primera es utilizando la API que proporciona el Backend, basada en HTTP REST (GET o POST) la cual, en función del recurso solicitado, devuelve un resultado concreto, con un payload o carga útil con formato JSON.

La segunda manera es utilizando una URL de Callback, identificando dicha URL asociada a la aplicación web encargada de recibir los mensajes. Por ende, se registraría dicha URL en el Backend, definiendo los atributos que le interese recibir y cada vez que llegase un mensaje al Backend, éste será capaz de reenviar los valores requeridos en un mensaje en el mismo formato de la aplicación (Json).

Como se explicó en la arquitectura de la red Sigfox y la recepción cooperativa que esta incorpora, el módem de radio de Sigfox envía ráfagas de datos a las antenas de la estación base. Por lo que, para un caso ideal, una señal será captada por más de una antena. El paquete de datos se demodula en la estación base y luego es enviado al centro de datos de Sigfox. Posteriormente, el centro de datos envía los datos recibidos a los usuarios del servicio a través de servicios web de Callback.

URL Callback. Método utilizado para leer y comprobar la recepción de los datos ya que Sigfox permite recoger y mostrar de manera sencilla los datos que están en la nube. Un Callback es conocido como la devolución de llamada y es importante subrayar que es posible configurarlo en la sección DEVICE TYPE de la plataforma Backend, la cual permite añadir, modificar y eliminar estas retro-llamadas.

Cobertura

Latinoamérica es una de las regiones donde Sigfox se mantiene activo, sin embargo solamente se mantuvo en operación para Ecuador hasta finales del año 2022.

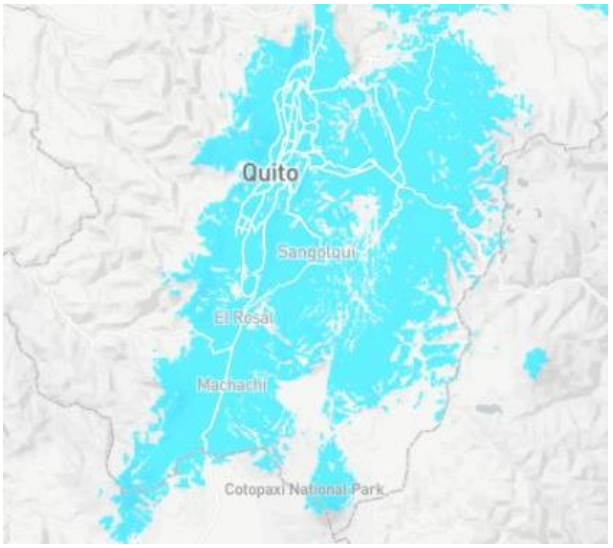
Figura 14

Mapa de la cobertura de la red Sigfox en Ecuador

Nota. Cobertura de la red Sigfox para Ecuador. Tomado de *Sigfox Techinical Overiew*, por Sigfox, 2017, Sigfox (https://www.Sigfox.com/coverage/).

Figura 15

Mapa de la cobertura de la red Sigfox en Quito



Nota. Cobertura de la red Sigfox para Quito y sus inmediaciones. Tomado de *Sigfox Techinical Overiew*, por Sigfox, 2017, Sigfox (https://www.Sigfox.com/en/coverage/).

Plataformas y servicios

Empresas multinacionales como Amazon, IBM y Microsoft dedicadas al sector de software y hardware ofrecen gran variedad de servicios y aplicaciones para el procesamiento, gestión, control y visualización de la información obtenida por los sensores de tecnología IoT.

Los servicios permiten al usuario supervisar, conectar y controlar miles de millones de recursos y se acoplan a la necesidad del usuario, la mayoría de estos aplicativos son de pago y las plataformas incorporan un sistema de periodo de prueba gratuito limitado. Asimismo, se señala que la mayoría aloja sus funciones en la nube, pero también es posible instalar el software de forma local (Morales & Altamirano, 2016).

Ubidots. Es una plataforma de IoT que permite construir nuevos productos y aplicaciones a partir de objetos conectados a Internet. Ubidots permite al usario enviar datos de sensores a la nube, conectarse con otras plataformas, configurar tableros y alertas, usar herramientas de analítica y arrojar mapas de datos en tiempo real. Ubidots permite crear aplicaciones para el “IoT” en poco tiempo y cuenta con una interfaz amigable con el usuario (Espinosa & Orellana, 2021).

Es un servicio gratuito que permite a los desarrolladores crear un entorno de computación en la nube fiable, asequible y utilizable en un ecosistema de plataformas “IoT”. Ubidots se especializa en soluciones de software y hardware cuyo fin va de la mano con monitorizar, automatizar y controlar procesos remotamente en el ámbito de la salud, fabricación, servicios públicos, transporte, industria y energía (Espinosa & Orellana, 2021).

Figura 16

Plataforma de Ubidots

Nota. Dashboard creado en Ubidots. Tomado de *Desarrollo de aplicaciones de monitoreo y control basadas en* IoT *a través de la plataforma Ubidots,* por Espinosa, B., & Orellana, M., 2021.

Algunos de los beneficios de la plataforma Ubidots son:

- Es posible configurar las variables de forma automática, su apariencia, características y periodo de tiempo.
- Permite la supervisión para el análisis de datos de las aplicaciones con integraciones (Api, del inglés *Application Programming Interface*).
- Conversión de datos de origen nativos mediante UbiFunctions en información con variables sintéticas, solo disponible mediante versión paga.
- Consta con una interfaz de diseño para cuadros de mando y análisis en tiempo real para el análisis de datos y control en los dispositivos

- Mejora la función compartir datos con enlaces públicos o integrando cuadros de mando o *widgets* en aplicaciones web privadas y móviles.
- Cuenta con permisos y funciones para cualquier usuario que pueda interactuar con los comandos, dispositivos, variables y eventos.

Capítulo 3

Materiales y Métodos

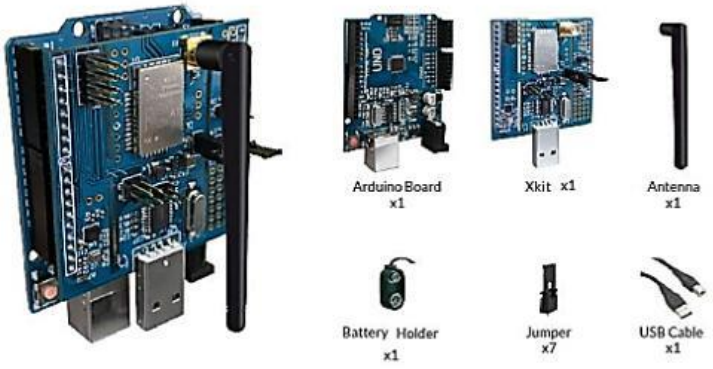
Para el monitoreo y gestión de parámetros de vehículos, el mercado ofrece una serie de recursos que posibilitan la implementación de nuevos proyectos, por lo que existen diferentes tipos de sensores y plataformas especializadas, por lo que se debe seleccionar adecuadamente el material más adecuado para el desarrollo de la aplicación.

Módulo Thinxtra Sigfox

Sigfox es una compañía especializada en telecomunicaciones que provee a sus usuarios distintas soluciones de IoT. Como se muestra en la Figura 17, el módulo comprende una placa compatible con Arduino UNO que permite enviar y recibir mensajes a nivel mundial bajo la red de Sigfox; este módulo permite a los estudiantes, programadores y emprendimientos crear soluciones de IoT sin la necesidad de tener experiencia previa y cuenta con un año de garantía para su uso y despliegue (Thinxtra, 2019).

Figura 17

Componentes Kit de desarrollo Thinxtra



Nota. Kit de desarrollo Xkit Sigfox Thinxtra y sus componentes. Tomado de *Thinxtra Sigfox Developer Xkit*, por Thinxtra, 2019.

Sigfox Thinxtra RCZ4

Se maneja con una frecuencia de 920MHz, comprende tres sensores embebidos, a uno de ellos

se lo conoce como acelerómetro de tres ejes, el mismo que es micromaquinado con una resolución de 14 bits y su función es medir fuerza ya sea dinámica de aceleración o estática, también posee un sensor digital de temperatura que se maneja en 300 a 1100 hPa y otro de presión que está en grados de -40˚ a 85˚ C, sus sensores de luz poseen fotodiodo y un circuito amplificador de corriente, además posee interruptores de láminas, leds, entre otros (Thinxtra, 2019).

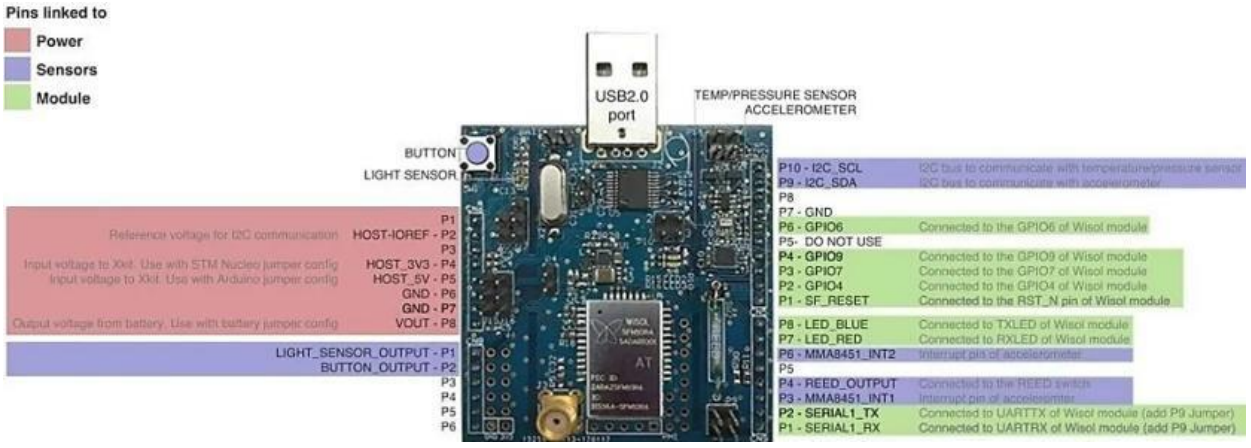
Brinda un alto rendimiento, alta potencia de radiofrecuencia, alta calidad y el software incluye adicionalmente un gestor de arranque que facilita el desarrollo de aplicaciones y las actualizaciones de software, todo esto a bajo costo como se caracteriza Sigfox (M2Communication, 2018).

Esta placa posee una tasa de datos de 600 bps, una potencia máxima de 24 dBm por dispositivo, con una duración de batería extendida dependiendo completamente de la cantidad de conexiones que se originen y del tipo de sensor usado.

Ofrece una gran seguridad dado que todos sus mensajes son autenticados y codificados bajo una palabra clave y están basados en mecanismos sólidos como autenticación, privacidad, integridad y confidencialidad.

Figura 18

Disposición de los pines de entrada y salida del módulo Thinxtra Xkit

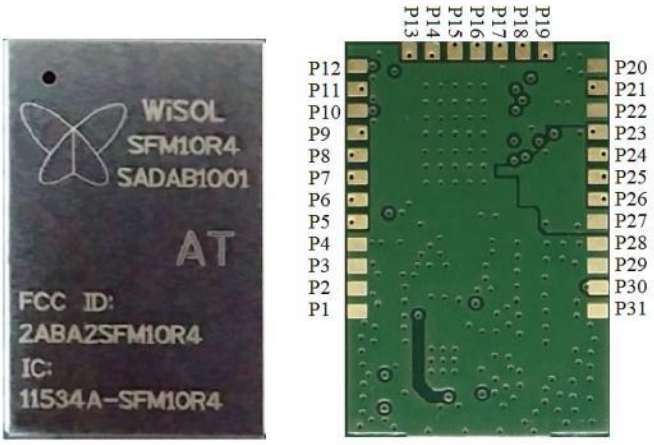


Nota. La figura se muestra la disposición y la función de cada uno de los pines de entrada y salida del módulo. Tomado de *Thinxtra Sigfox Developer Xkit*, por Thinxtra, 2019.

El chip que utiliza el Xkit Thinxtra es el WISOL/WSSFM10R4AT correspondiente a la región RCZ4, que funciona con ciertas regulaciones locales, cada zona maneja diferentes frecuencias (RCZx) y canales. Es así que se determinó que la placa a usar en el trabajo de titulación es la RCZ4 debido a la ubicación en la que se ha desarrollado el proyecto de Investigación (Ecuador), además de que se maneja un protocolo inalámbrico WSSFM10.

Figura 19

Configuración de los pines de entrada y salida del módulo Wisol



*Nota. D*isposición de los pines de entrada y salida del módulo Wisol, izquierda (vista frontal) y derecha (vista trasera). Tomado de *WISOL/WSSFM10R4AT Datasheet,* por Ji, S., 2019.

Tabla 4

Función de los pines del módulo Wisol

Pin	Función
1,2,3,4	GND
5	NC3/SYSCLK
6	GPIO6
7	GPIO7
8	GPIO8
9	GPIO5
10	GPIO4

Pin	Función
11	CPU LED
12	RADIO LED
13	GPIO9
14	UART TX
15	UART RX
16	RXLED/DBG DATA
17	TXLED/DBG CLK
18	NC4/DBG EN
19	RST N
20, 22	GND
21	VDD IO
23,24,25,26	GPIO 0,1,2,3
27,28,29,31	GND
30	RF IO

Nota. La tabla especifica la configuración de pines o pinout del módulo Wisol y la función de cada uno de ellos. Tomado de *WISOL/WSSFM10R4AT Datasheet,* por Ji, S., 2019.

En cuanto se refiere a especificaciones técnicas de este módulo, la tabla 5 muestra los rangos máximos absolutos del chip WISOL y la tabla 6 muestra las características DC del mismo.

Tabla 5

Rangos máximos absolutos del chip WISOL

Símbolo	Parámetro	Rango	Unidad
VCC	Voltaje de entrada del módulo	-0.5 a 5.5	V
OT	Temperatura de Funcionamiento	-30 a 85	˚ C
ST	Temperatura de almacenamiento	-40 a125	˚ C

Nota. Tomado de *WISOL/WSSFM10R4AT Datasheet*, por Ji, S., 2019.

Tabla 6

Características DC del chip WISOL

Símbolo	Parámetro	Mínimo	Típico	Máximo	Unidad
VCC	Voltaje de entrada	27	3.3	3.6	V
	Corriente de Tx	-	200	-	mA
Corriente	Corriente de Rx	-	32	-	mA
	Corriente en modo “Sleep”		2.5		µA

Nota. Tomado de *WISOL/WSSFM10R4AT Datasheet*, por Ji, S., 2019.

Tabla 7

Especificaciones RF

Parámetro		Mínimo	Típico	Máximo	Unidad
Frecuencia RF	Tx	-	920.8	-	MHz
	Rx		922.3	-	MHz
Potencia de Salida TX		-	22.5	-	dBm
Tolerancia de errores de frecuencia		- 2.5	-	+2.5	Ppm
Sensibilidad de Rx (600bps, GFSK)		-129	-	-	dBm
Emisión Espuria de Rx		-	-	-54	dBm

Nota. Se describe algunas de las especificaciones RF del módulo. Tomado de *WISOL/WSSFM10R4AT Datasheet*, por Ji, S., 2019.

Arduino

Es una plataforma de hardware con un modelo de desarrollo de software basado en la colaboración abierta, se constituye de una placa de circuito impreso que comprende un microcontrolador de marca ATMEL que dispone de entradas y salidas, digitales y analógicas, en un

entorno de desarrollo fundamentado en el lenguaje de programación “processing”. Los dispositivos de la familia Arduino son capaces de conectar el mundo físico con un entorno virtual controlando alarmas, sensores, actuadores físicos, sistemas de comunicaciones y motores (Tapia & Manzano, 2013).

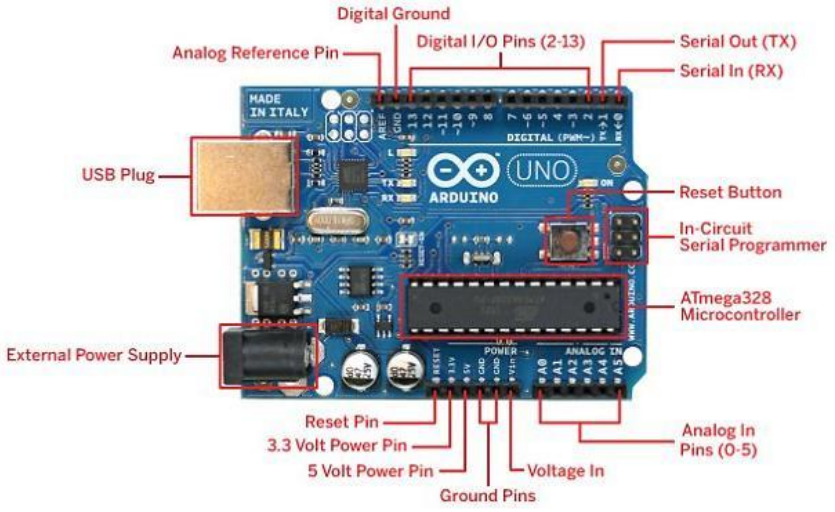
Arduino UNO

Arduino UNO es una de las placas más empleadas en la industria y en proyectos tecnológicos de robótica y electrónica, contiene un microcontrolador ATmega328 con una memoria flash de 32 KB para almacenar el código de los cuales 0,6 KB se destinan para el uso del gestor de arranque. Para la SRAM se dispone de 2 KB y 1 KB de EEPROM, dispone de 6 entradas análogas, 14 entradas y salidas digitales, entre las cuales, 6 son utilizados como salidas PWM (Modulación por ancho de pulso), por separado comprende 6 entradas analógicas, un cristal de 16 MHz oscilador, una conexión USB, una cabecera ICSP y un botón de reinicio, todos estos componentes son mostrados en la figura 20.

Este diseño hace posible que el microcontrolador se pueda alimentar por medio de un cable USB y una PC, una fuente de alimentación AC/DC o mediante una batería externa (Tapia & Manzano, 2013).

Figura 20

Componentes de una placa Arduino Uno



Nota. Tomado de *Evaluación de la plataforma Arduino e Implementación de un sistema de control de posición horizontal*, por Tapia, C., & Manzano, H., 2013.

Tabla 8

Características básicas de una placa Arduino Uno

Parámetro	Valor
Voltaje de operación	5 V
Voltaje de Entrada (recomendado)	7-12 V
Número de pines digitales	14 (6 proporcionan salida PWM)
Número de pines de entrada analógica	6
Corriente DC	40 mA
Corriente continua	3.3 V Pin 50 mA
Memoria Flash	32 KB (ATmega328) de los cuales 0,5 KB utilizado por gestor de arranque
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad del reloj	16 MHz

Nota. Tomado de *Evaluación de la plataforma Arduino e Implementación de un sistema de control de posición horizontal*, por Tapia, C., & Manzano, H., 2013.

Microcontrolador ATMEGA328. Es un circuito integrado que contiene las partes funcionales de una Pc, como: memorias (RAM) para datos, CPU, memorias (PROM, ROM y EPROM) para escribir el programa, pines de entrada y salida para la comunicación con el mundo exterior y algunos periféricos (comunicación serial, convertidor A/D, temporizador).

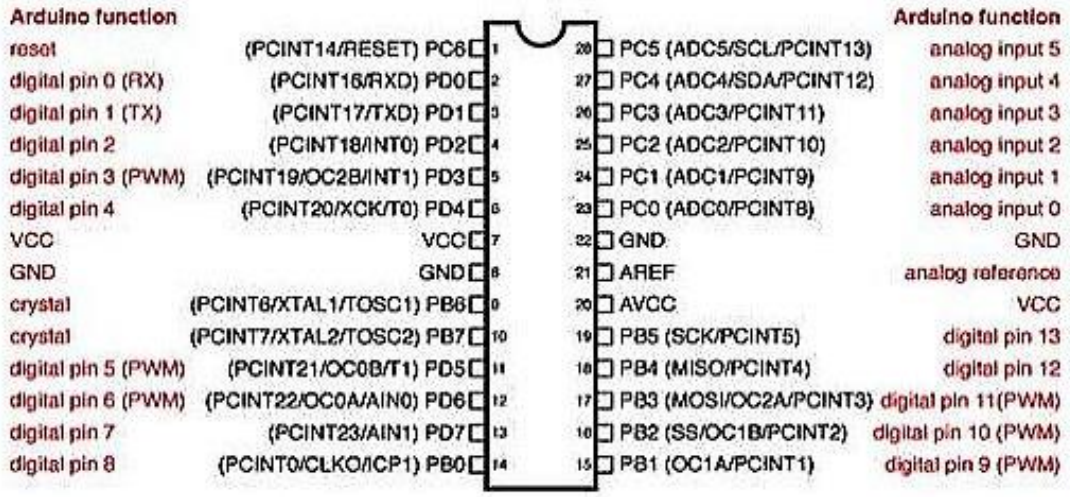
La familia de microcontroladores AVR son producidos por ATMEL, y cada microcontrolador es un chip con memoria flash reprogramable. Una de las características clave de los microcontroladores ATMEL es su estructura interna, que incluye 28 pines, 16k bytes de In-flash, un sistema programable con lectura y escritura, 1k bytes de SRAM, 512 bytes de EEPROM, 32 registros para trabajo general, 23 líneas para E/S de propósito general, un temporizador y contadores (Atmel, 2015).

El usuario puede detener o apagar módulos no utilizados en el microcontrolador utilizando uno de sus diversos modos de suspensión, lo que reduce significativamente el consumo de energía.

Entre los más importantes destacan el modo de apagado que detiene todos los relojes generados y permite solo la operación de módulos asíncronos, y el modo ahorro de energía, este modo de suspensión es similar al modo de apagado, solo con una excepción, es decir, si el temporizador/contador está habilitado, este permanecerá en estado de ejecución incluso en el momento de la suspensión.

Figura 21

Mapa de pines microcontrolador ATmega 328



Nota. Disposición de los pines analógicos y digitales del microcontrolador. Tomado de *Inspiring Smart and Secure Connected Design,* por Atmel, 2015.

Tabla 9

Parámetros y Especificaciones del microcontrolador ATmega328P

Parámetro	Especificación
I/O y paquetes	23 líneas de I/O programables 32-lead TQFP, y 32-pad QFN/MLF

Parámetro	Especificación
Voltaje Funcionamiento	2.7V a 5.5V
Rango de temperatura	Rango de temperatura automotriz: –40°C to +125°C
Grado de Velocidad	0 a 16 MHz a 4,5 a 5,5 V (rango de temperatura automotriz: –40 °C a +125 °C)
Bajo consumo de energía	Modo activo: 1.5mA a 3V - 4MHz Modo de apagado: 1 µA a 3 V

Nota. Tomado de *Evaluación de la plataforma Arduino e Implementación de un sistema de control de posición horizontal*, por Tapia, C., & Manzano, H., 2013.

Arduino MEGA

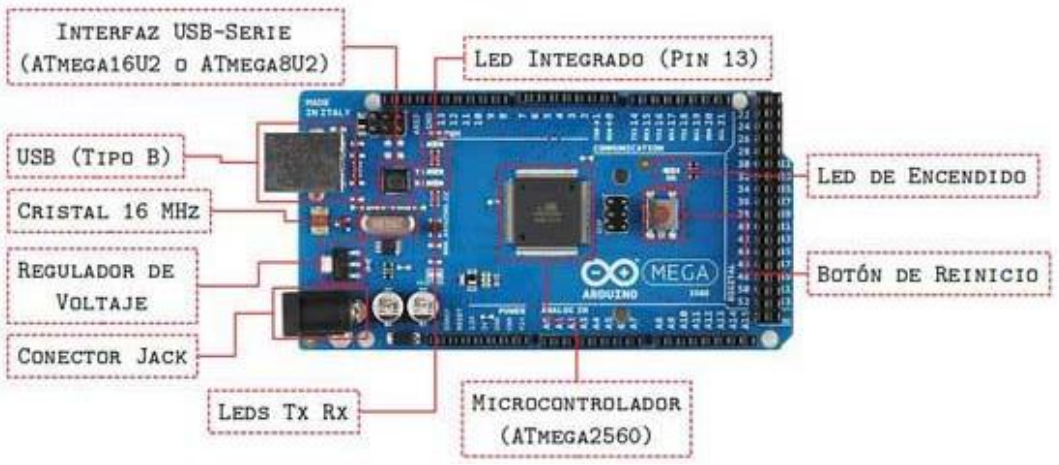
Arduino MEGA 2560 es una placa de desarrollo cuyo funcionamiento está basado en el microcontrolador ATmega2560 (de donde proviene su nombre). Esta placa pertenece a una vasta familia de placas Arduino, siendo una de las más representativas junto con la placa Arduino UNO. Sus componentes se muestran en la figura 22.

La placa Arduino MEGA 2560 está compuesta, básicamente, por distintos elementos, descritos a continuación:

- El microcontrolador ATmega2560 con la configuración de “sistema mínimo”, es decir, solo se utilizan los componentes indispensables para el correcto funcionamiento de microcontrolador ATmega.
- Una interfaz USB/Serie que otorga la capacidad de re-programar el microcontrolador usando un cable USB, un ordenador, y el software Arduino IDE.
- Un conjunto de cabezales montados que permiten conectar los pines de entrada/salida, ya sea con cualquier sistema externo o placa complementaria que permita ampliar las capacidades de la placa Arduino Base (Shields).

Figura 22

Descripción de los componentes de la placa Arduino MEGA



Nota. Tomado de *Evaluación de la plataforma Arduino e Implementación de un sistema de control de posición horizontal*, por Tapia, C., & Manzano, H., 2013.

Tabla 10

Características básicas de la placa Arduino MEGA

Parámetro	Valor
Voltaje de operación	5 V
Voltaje de Entrada (recomendado)	7-12 V
Voltaje de Entrada (límites)	6-20 V
Número de pines digitales	54(de los cuales 15 proporcionan salida PWM)
Número de pines de entrada analógica	16
Corriente DC	40 mA
Corriente continua	3.3 V Pin 50 mA
Memoria Flash	256 KB de los cuales 8 KB utilizado por gestor de arranque
SRAM	8 KB

Parámetro	Valor
EEPROM	4 KB
Velocidad del reloj	16 MHz

Nota. Tomado de *Evaluación de la plataforma Arduino e Implementación de un sistema de control de posición horizontal*, por Tapia, C., & Manzano, H., 2013.

Microcontrolador ATMEGA2560. El ATmega2560 es el cerebro de la placa Arduino MEGA (se encarga de grabar, almacenar y ejecutar el código programado) y el resto de los componentes, de manera qué garantiza el correcto funcionamiento de la placa (Williams, 2015). Sus características más importantes son:

- 256 kB de memoria FLASH (espacio disponible para almacenar el sketch o programa).
- 8 kB de memoria SRAM (espacio donde se crean las variables declaradas en el programa).
- 4 kB de memoria EEPROM (almacena los datos y permite que se conserven aunque falle la alimentación o se reinicie).
- Frecuencia de CPU Máxima: 16MHz.
- Voltaje de Operación máximo: 6.0V

CAN Bus Shield

Esta shield permite a una placa Arduino comunicarse mediante el bus CAN. Proporciona la capacidad para sondear la ECU para obtener información, incluida la posición del acelerador, la temperatura del refrigerante, la velocidad del vehículo y las rpm del motor. También puede almacenar estos datos o enviarlos al tablero para mostrarlos. Emplea el controlador CAN-MCP2515 de Microchip con el transceptor CAN-MCP2551. La conexión CAN se realiza a través de un conector sub-D estándar de 9 pines para utilizar con un cable OBD-II ideal para aplicaciones en automóviles. El CAN Bus shield también tiene un soporte para tarjetas microSD, conexión para serial LCD y conector para un módulo GPS EM506. Estas características hacen que este shield sea ideal para aplicaciones de registro de datos.

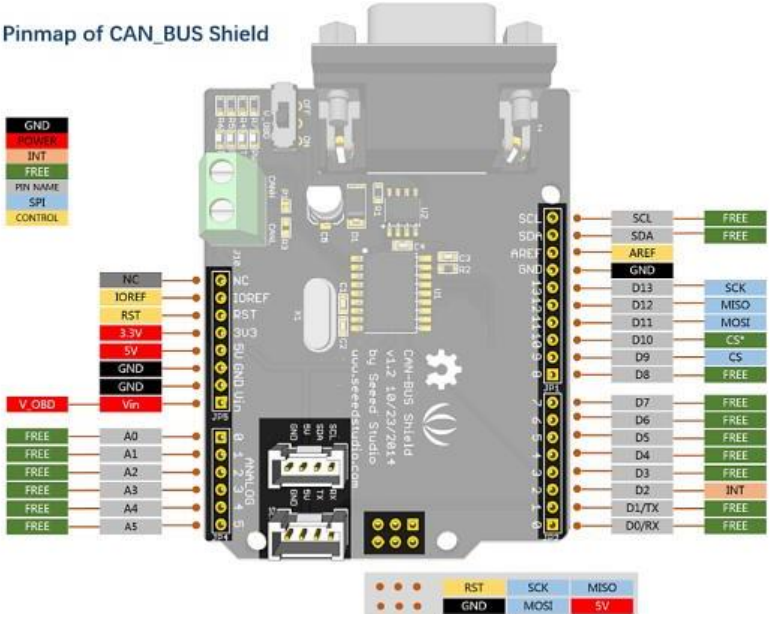
Características del CAN Bus Shield

- Implementa velocidad CAN V2.0B de hasta 1 Mb/s
- Velocidad de interfaz SPI de hasta 10 MHz
- Datos extendidos (29 bits) y marcos remotos
- Dos búferes de recepción con almacenamiento de mensajes priorizados
- Conector DB-9 estándar industrial

CAN BUS Shield funciona bien con Arduino UNO (ATmega328), Arduino Mega (ATmega1280/2560) y Arduino Leonardo (ATmega32U4).

Figura 23

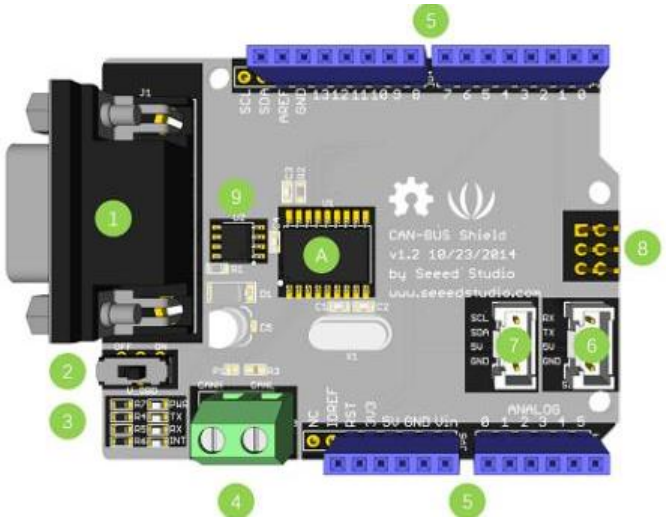
Configuración de los pines del CAN Bus Shield



Nota. El gráfico representa el mapa de pines del CAN Bus Shield, donde los pines libres están disponibles para otros usos, los pines señalados de azul corresponden al protocolo de comunicación SPI, los pines rojos a la alimentación y los pines amarillos son de control. Tomado de *CAN-BUS shield V2.0*, por Zuo, B., 2022, Seedstudio (https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/).

Figura 24

Descripción de los componentes del CAN Bus Shield



Nota. Se muestra una descripción general del hardware del CAN Bus Shield. Tomado de *CAN-BUS shield V2.0*, por Zuo, B., 2022, Seedstudio (https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/).

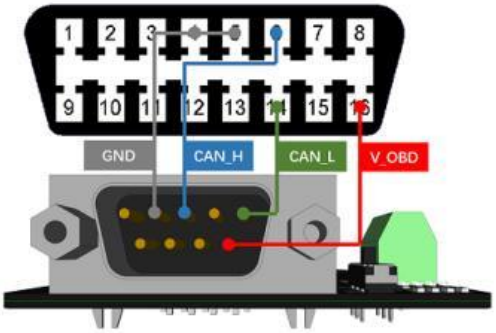
1. **DB9 Interface**. Para conectarse a la interfaz OBDII a través de un cable DBG-OBD.
2. **V_OBD**. Obtiene energía de la interfaz OBDII (de DB9).
3. **Indicadores Led**
    - **PWR**. Poder.
    - **TX**. parpadea cuando los datos se están enviando.
    - **RX**. parpadea cuando los datos se están recibiendo.
    - **INT.** interrupción de datos.
4. **Terminal. CAN-H y CAN-L**
5. **Asignación de pines para Arduino UNO**
6. **Serial Grove connector**
7. **I2C Grove connector**
8. **ICSP pins**

9. **IC MCP2551.** Transceptor CAN de alta velocidad.

10. **IC MCP2515.** Controlador CAN independiente con interfaz SPI.

Figura 25

Interfaz DB9 y OBD II



Nota. El gráfico muestra la interfaz DB9-OBDII, Tomado de *CAN-BUS shield V2.0*, por Zuo, B., 2022, Seedstudio (https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/).

Módulo GPS NEO-6M

Es un módulo receptor cuyo uso están relacionado con aplicaciones de geolocalización, posee una memoria EEPROM para guardar datos, además cuenta con una antena de gran potencia, y una batería utilizada para respaldar la configuración del módulo, recibiendo así las señales de diferentes satélites que circundan el planeta tierra.

El GPS NEO-6M es compatible con AVR, Arduino, PIC, y otros microcontroladores disponibles en el mercado y puede entregar información precisa, así como ser configurado mediante el puerto UART.

El módulo receptor GPS NEO-6M se comunica con el terminal PC o microcontrolador a utilizar mediante comunicación USART. Recepta información como la altitud, hora UTC, latitud, longitud, etc. de los satélites en forma de cadena NMEA. Esta cadena debe ser analizada para extraer la información deseada. Las siguientes son algunas características del módulo GPS NEO-6M:

Interfaz: RS232 TTL

- Fuente de alimentación: 3V a 5V

- Velocidad de transmisión predeterminada: 9600 bps
- Funciona con frases NMEA estándar

Figura 26

Módulo GPS NEO-6M

Nota. Tomado de *Sistema posicionamiento global (GPS) y teorías de la relatividad*, por Sanchez, G., 2012.

Sensores

La adquisición de datos es el proceso mediante el cual se toman variables físicas de un proceso, se miden mediante sensores que las convierten en señales eléctricas, se muestrean para ser procesadas, almacenadas o visualizadas. Para el propósito de este proyecto, algunos de los sensores usados son:

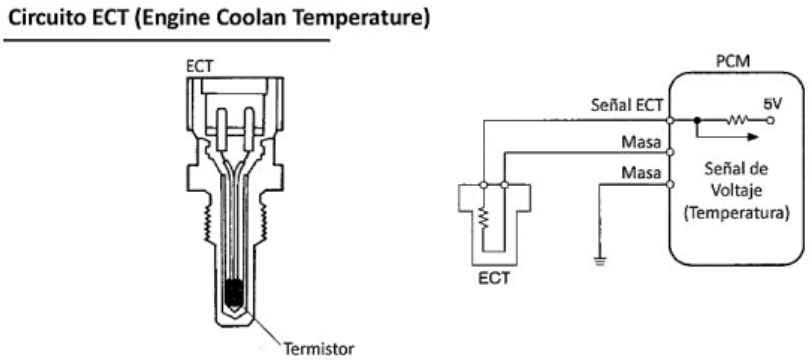
Sensor ECT

El sensor de temperatura refrigerante o ECT es un dispositivo emplazado en el motor del vehículo, está regularmente enroscado dentro del bloque del motor en el múltiple de la toma inferior o en el cabezal del cilindro para proporcionar un contacto directo con el refrigerante (Orozco, 2020).

Este sensor permite estimar la temperatura del motor, en virtud de ser un sensor termistor, es decir, posee una resistencia que va a cambiar de acuerdo a la variación de la temperatura, cuanto más se expone a altas temperaturas, la resistencia será mucho menor. De este modo, el sensor envía una señal a la unidad de control para que esta pueda recibir la información, decodificar los valores y transformarlos en un índice de temperatura del refrigerante. Véase figura 27.

Figura 27

Sensor ECT



Nota. El gráfico muestra un esquema del sensor ECT y sus pines. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

Códigos OBD-II relacionados al Sensor ECT. El Sensor ECT se encarga de enviar información del sistema de inyectores a la ECU del vehículo. De esta manera, el motor actúa de forma normal. Sin embargo, cuando ocurren fallas, se deben diferenciar unas de otras para poder atacar el problema. Es así que, se han establecido los códigos OBD II, que permiten identificar los síntomas de falla, como:

- **P0117.** Es generado cuando hay bajo voltaje.
- **P0118.** Respuesta a alto voltaje.
- **P0070.** Indica que existe una falla en el circuito del sensor.

Sensor IAT (Temperatura de aire de entrada)

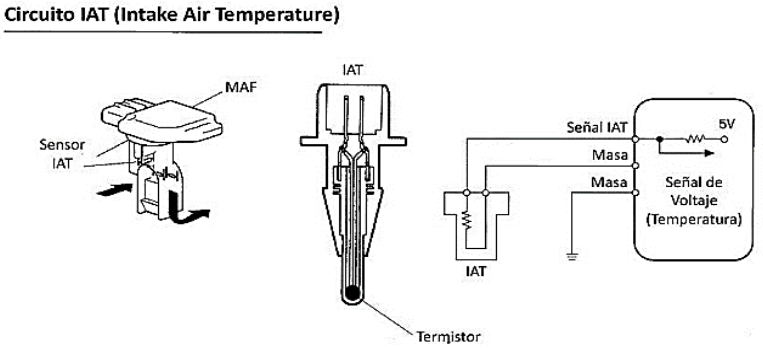
El Sensor IAT es un termistor o resistencia térmica capaz de censar la temperatura del colector de admisión y ajustarla en una medida variable. Estas mediciones se realizan durante un arranque en frío o cuando el motor calienta el aire de admisión; cuanto mayor sea el calor, menor será la resistencia del termistor, estas mediciones son tomadas durante un arranque en frío o cuando el motor calienta el aire de admisión. Esta información es enviada a la computadora central del vehículo para que se pueda corregir el tiempo de inyección y ajustar la mezcla de combustible.

El sensor IAT es un termistor NTC, que se caracteriza por ser un sensor de temperatura por resistencia, que varía su valor con la temperatura con un coeficiente negativo. Este concepto establece que a un aumento de temperatura se le atribuye una disminución de la resistencia térmica.

Es posible instalar este sensor por separado o como parte del sensor MAF. En el primer escenario, el diagrama de circuito tiene dos terminales: el cable de tierra o masa y el cable de señal (5 Voltios).

Figura 28

Sensor IAT



Nota. El gráfico muestra un esquema del sensor IAT y sus pines. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

Códigos OBD-II relacionados al Sensor IAT. Los códigos OBD-II son descripciones alfa numéricas que permiten detectar los gases contaminantes emitidos al medio ambiente por los vehículos automotores. Al existir fallas en el Sensor IAT, los códigos reportan el siguiente significado:

- **P0112.** Se produce si la entrada está por debajo del rango estimado. De manera que, si es menor a 0,18 voltios genera la falla.
- **P0113.** Indica entrada alta al circuito. Puede darse por daño en el cableado o del mismo sensor.
- **P0127.** La tensión de voltaje o la temperatura del aire son muy altas.

Sensor de presión absoluta del múltiple (MAP)

El Sensor MAP (del inglés *Manifold Absolute Pressure*) es un componente de los vehículos de

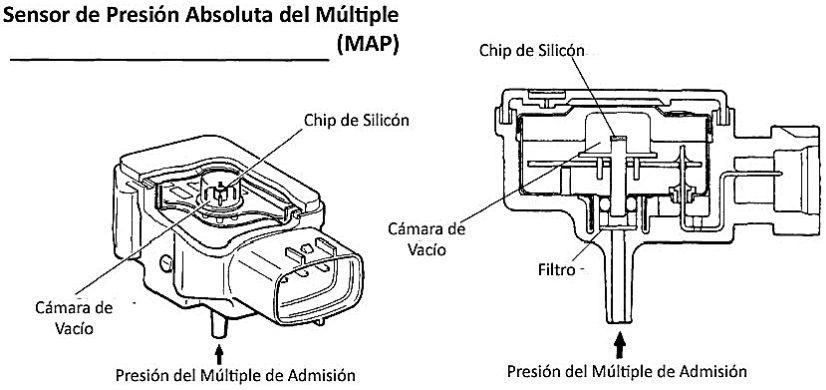
combustión interna se encarga de medir la presión de aire que pasa a través del cuerpo del acelerador o la presión del colector de admisión ubicado detrás del acelerador.

La función del Sensor MAP es comparar la presión múltiple de admisión del vehículo con la presión de la atmósfera, generando una señal de voltaje existente y enviándola a la unidad de control del vehículo. A su vez, la ECU evalúa las magnitudes adquiridas, las compara con otros parámetros internos predefinidos y decide si se requiere una inyección de combustible adicional.

La ECU usa esta señal para calcular el volumen de aire de admisión en la fórmula capaz de determinar la relación entre la presión del colector de admisión y los rpm.

Figura 29

Sensor MAP



Nota. El gráfico muestra un esquema del sensor MAP y sus pines. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

Códigos OBD-II relacionados al Sensor MAP. Si el Sensor MAP presenta fallas bajo el escáner puede reportar el código P0107. El significado de este código va relacionado con una entrada baja de voltaje en la unidad de control del motor o ECU. Generalmente, el valor se sitúa en un valor menor a 5 voltios, rango donde hay un buen rendimiento del motor. Otros códigos de fallas son:

- **P0109.** Alerta que el Módulo de Control del Tren de Potencia o PCM ha detectado que el sensor de presión absoluta del múltiple ha dado una lectura intermitente.

- **P0108.** Indica que existe un problema en el circuito del sensor MAP y que hay una entrada de voltaje demasiado elevada para el PCM.

Sensor de posición del acelerador (TPS)

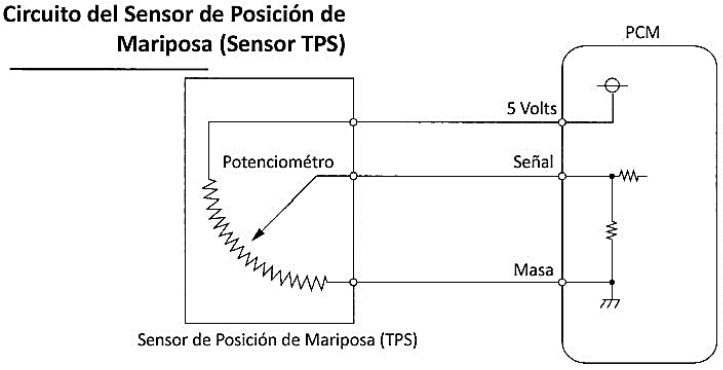
El Sensor TPS es un transmisor que se encarga de controlar la inyección de combustible mediante una señal que se envía a la computadora. Al abrirse la mariposa o cuerpo de acelerador, se permite la entrada del aire a la cámara de combustión del motor. Acción que es posible gracias a una guaya accionada por el acelerador o pedal.

Bajo este principio, el sensor es capaz de verificar la posición de la mariposa que se encuentra justo a la entrada del motor del vehículo, encargándose de hacerle saber a la unidad de control electrónico o ECU en qué estado se encuentra la mariposa de aceleración. Si no se acciona, no podrá responder a las órdenes del usuario (Orozco, 2020).

Dependiendo de la aceleración del vehículo el Sensor TPS puede moverse hasta 100 grados. De manera que, si el vehículo no acelera, la mariposa se encuentra cerrada y el sensor está a 0 grados. El Sensor TPS tiene injerencia sobre las funciones: Dosifica la cantidad de combustible, desconecta el aire acondicionado cuando hay aceleración brusca y controla la marcha en mínimo.

Figura 30

Sensor TPS



Nota. El gráfico muestra un esquema del sensor TPS y sus pines. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

Códigos OBD2 relacionados al Sensor TPS. El Sensor TPS puede presentar fallas que se diagnostican mediante el protocolo OBD-II, mediante un escáner. Para este sensor, los códigos que pueden ser arrojados más comunes son:

- **P0122.** Se reporta cuando el voltaje de salida del sensor se encuentra por debajo del rango correcto.
- **P0123.** Al contrario que el código anterior. Es reportado cuando el módulo de control del motor o ECM supera el voltaje esperado por el sensor.

Sensor de velocidad del Vehículo (VSS)

Este dispositivo es el encargado de avisarle a la unidad central automotriz a qué velocidad se está desplazando el vehículo. El VSS es capaz de captar el giro o revoluciones de un componente mecánico del vehículo y posteriormente enviar la señal a la ECU. Gracias al sensor de velocidad del vehículo, la computadora interna del vehículo puede controlar otras funciones relacionadas al desplazamiento.

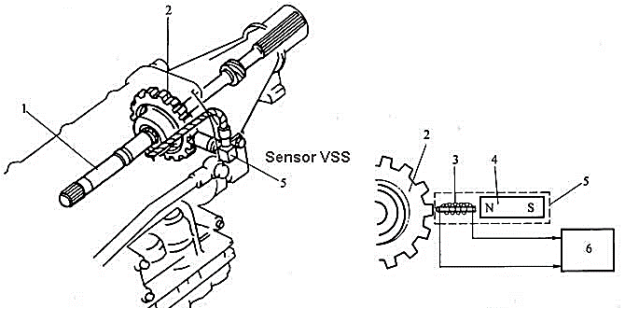
Además de indicar a qué velocidad marcha el vehículo y la variación de la misma con respecto al tiempo, el sensor VSS pasa a controlar otros indicadores como el odómetro, el velocímetro, y el convertidor de fuerza.

Cuando el sensor VSS envía la señal proporcional a la velocidad de giro, la ECU puede sincronizar el accionar de otros sistemas del vehículo. Como, por ejemplo, la inyección de combustible, el tiempo de encendido, el control de tracción, y el sistema ABS.

Varios de estos sensores de velocidad tienen un extremo magnetizado, el cual capta el movimiento rotativo, por generación de pulsos. Para otros casos su accionar está basado en el efecto HALL (Orozco, 2020).

Figura 31

Sensor VSS



Nota. El gráfico muestra un esquema del sensor VSS y su estructura. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

Códigos OBD2 relacionados al Sensor de Velocidad. Al presentarse una falla en el sensor de velocidad pueden presentarse alguno de los siguientes códigos ODBII:

- **P0500.** Se encontró una falla en el sensor de VSS.
- **P0501.** La velocidad del vehículo se encuentra fuera del rango normal de trabajo.
- **P0502.** La señal de entrada es muy baja.
- **P0503.** El voltaje es intermitente.

Sensor RPM

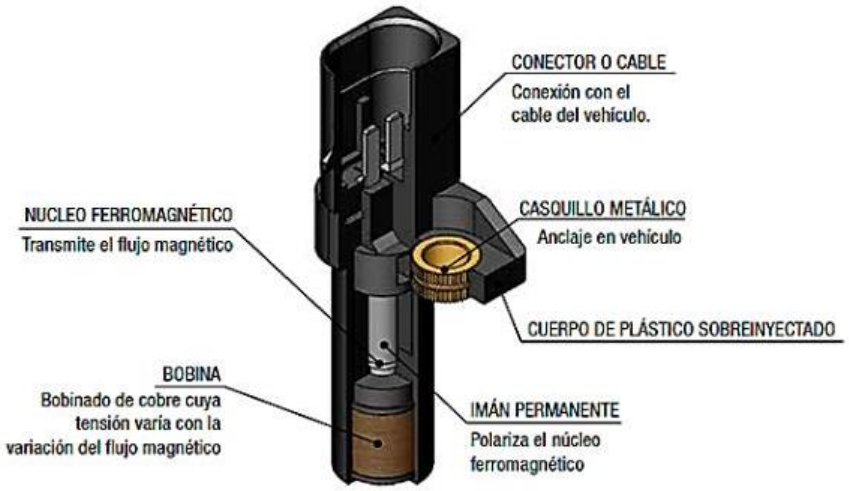
Un componente fundamental para el funcionamiento típico del motor es el sensor RPM o de revoluciones por minuto del cigüeñal. Está a cargo de determinar las velocidades de rotación del cigüeñal y del árbol de levas del motor y transmitir los datos a la unidad de control para su procesamiento adicional. El principio de funcionamiento del sensor de velocidad de rotación es detectar el cambio magnético generado entre la captación del sensor y el elemento giratorio, considerando que el cambio magnético variará según el tipo de sensor.

La información adquirida del sensor RPM se transforma en señales eléctricas y es enviada a la ECU para definir la cantidad de combustible que debe inyectarse en el momento idóneo. Esta señal

facilita que la unidad del motor ajuste la recirculación de los gases de escape y la presión de sobrealimentación, además de reflejar la velocidad del motor en el tablero del conductor, y en algunos casos, la posición de la caja de cambios de acuerdo a la configuración del fabricante. Véase figura 32.

Figura 32

Sensor Revoluciones por Minuto (Efecto Hall)



Nota. El gráfico muestra un esquema del sensor RPM y sus componentes. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

Manejo de servicios de Ubidots

Ubidots maneja cuatro términos necesarios para la gestión de datos en la nube: Data Source, Value, Variable y Event. Se detallará cada uno de estos términos a continuación:

- **Data Source.** Manipula un conjunto de información haciendo referencia a un dispositivo, en donde cada data source utiliza una o más variables.
- **Value.** Es el valor actual de una variable en un instante determinado.
- **Variable.** Maneja un conjunto de datos que varían de acuerdo al tiempo.
- **Event.** Permite tomar una acción en un momento determinado y facilita la interacción con cada elemento de ubidots a través de su API, por lo que, cada elemento que presenta puede ser creado, modificado o eliminado.

Elementos de la Interfaz de Ubidots

Ubidots no solo proporciona un back-end de ingesta para recibir y almacenar datos, sino también un front-end amigable y fácil de usar para crear Dahsboards o funciones de gestión de usuarios para administrar cómo y quién debe tener acceso a los datos, es así que esta interfaz de usuario está conformada por los siguientes elementos:

Dispositivos. Comprenden los datos almacenados en variables. Para acceder a uno se requiere una key API o token, que permite la lectura y modificación de las variables propias de cada dispositivo.

Variables. El servidor almacena los datos en variables, variables que pueden ser sintéticas o crudas. Las variables sintéticas son variables con las que se ha realizado cálculos dentro de las funciones de Ubidots por lo que son variables cuyo valor ha sido modificado con respecto a un cálculo o fórmula, mientras que las variables crudas son variables obtenidas desde un entorno externo, en este caso valores y parámetros propios de un vehículo, donde los datos llegan sin ser procesados.

***Dashboard* (Tablero).** Conocido por su nombre en español como tablero, son herramientas que permiten al usuario ordenar y visualizar la información procesada mostrándola en *widgets*.

***Widgets*.** Son herramientas que permiten visualizar el comportamiento de las variables al extraer los datos de una variable almacenada anteriormente para un instante o periodo de tiempo determinado. Existen varios *widgets* para distintos propósitos, lo que facilita crear un ambiente llamativo para el usuario al momento de gestionar y operar datos.

Eventos. Ubidots permite remitir alarmas y notificaciones al usuario cuando este lo necesite, estas alarmas activan los eventos y se disparan cuando los valores de las variables a observar exceden los umbrales máximos antes preestablecidos.

Capítulo 4

Diseño e Implementación

El diseño consiste en un prototipo IoT que permite adquirir los datos del sistema de diagnóstico OBD-II que comprende información de los valores químicos, eléctricos y mecánicos adquiridos por los sensores que componen la unidad de control electrónico “ECU”.

Gracias a este prototipo fue posible adquirir la información recibida de los sensores para que pueda ser interpretada, manipulada y tratada para un posterior análisis de detección de fallas, es preciso señalar que al utilizar el sistema OBD-II el usuario puede recibir datos en tiempo real ya sea de rpm, temperatura y velocidad directamente de la unidad de control del vehículo también conocida como “ECU”.

Una vez adquirida esta información es enviada al Backend de Sigfox para posteriormente realizar la interpretación de los datos obtenidos y mostrarlos en la interfaz del usuario propia de la plataforma Ubidots.

La plataforma IoT Ubidots es la encargada de almacenar y mostrar toda la información del vehículo en tiempo real mediante una base de datos y un *dashboard* respectivamente, así como también las consultas históricas de los múltiples eventos y errores que pudieron presentarse en las pruebas de ruta, mismos que, han sido configuradas anteriormente como eventos globales dentro de esta plataforma. También se cuenta con un plan de mantenimiento preventivo y un historial de las posibles fallas que puedan suceder durante el uso del vehículo y en las pruebas de ruta a realizarse.

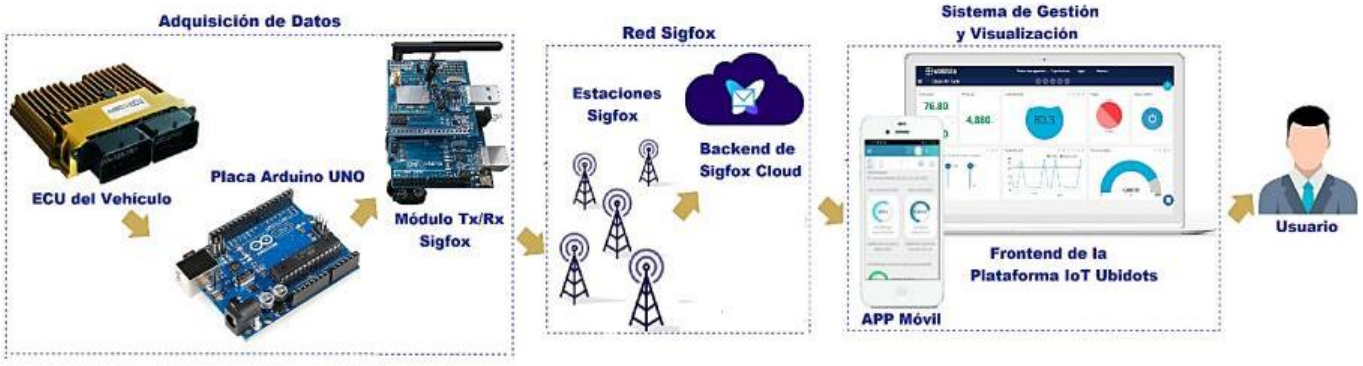
El frontend de Ubidots, así como también la aplicación móvil estan configurados para que el usuario pueda observar el comportamiento de los distintos sensores de la ECU, mediante la creación de un *dashboard* y distintos *widgets* que permiten visualizar la información y a partir de esta realizar un mantenimiento preventivo del vehículo.

En la figura 33 se detalla el diagrama de bloques simplificado del prototipo IoT para la

adquisición de datos del vehículo y el sistema de gestión de la información que ha sido descrito en esta sección, de manera visual.

Figura 33

Diagrama de Bloques simplificado del sistema de adquisición y monitoreo de parámetros de un Vehículo



Nota. Diagrama de Bloques simplificado. .

Diagrama de Bloques

Este diagrama está compuesto principalmente por tres bloques importantes: Adquisición de datos, Red Sigfox y el Sistema de Gestión y Visualización. El sistema ofrece únicamente una comunicación unidireccional ya que el flujo comunicacional solo es transmitido desde el emisor hacia el receptor considerando que los sensores que componen el ECU se encargan únicamente de la adquisición de la información para su posterior envío al Backend de Sigfox.

Bloque de Adquisición de Datos

Para el nodo 1 este bloque se compone de la Unidad de Control del Vehículo, el cable adaptador de puerto serie OBD II de 16 pines a DB9 que se encarga de conectar la interfaz de diagnóstico y el vehículo con la certificación OBD II, también se compone de un CAN-BUS Shield que incorpora el controlador de bus CAN MCP2515 con interfaz SPI y transceptor CAN MCP2551, mismo que se conecta a la tarjeta Arduino para así proporcionarle capacidades de CAN-BUS.

Una vez conectado el Can-Bus Shield y la tarjeta de desarrollo Arduino UNO esta será capaz de

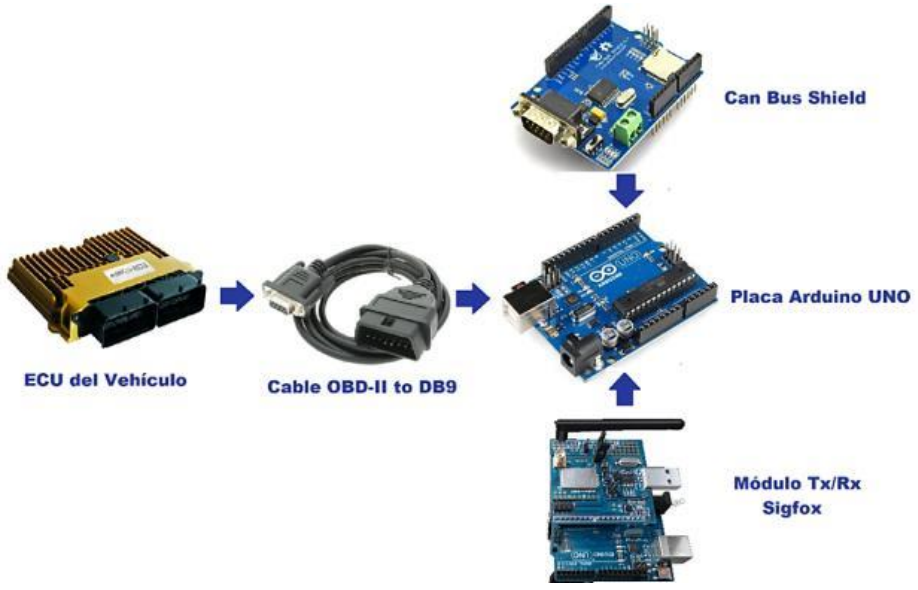
interpretar la información recolectada por los sensores, para posteriormente, mediante el módulo de desarrollo Thinxtra empaquetar los datos en un mensaje y enviarlos por la red de comunicaciones Sigfox para un análisis y gestión de los mismos. Los esquemas en Proteus son los anexos 5 y 6.

Para el nodo 2 este bloque se compone de una batería de 9V para la alimentación de un Arduino Mega, al cual se conecta un módulo GPS NEO-6M mediante el puerto serie Tx1 y Rx1 (pines 18 y 19 respectivamente), ya que el Rx0 y Tx0 (puerto serie por Hardware) estará siendo ocupado por la conexión existente entre el Arduino Mega y el módulo de desarrollo Thinxtra. Se realizan las conexiones que se describen a continuación:

- Conectar el pin 3.3V del Arduino MEGA al pin Vcc del módulo GPS.
- Conectar el pin GND del Arduino MEGA al pin GND del módulo GPS.
- Conectar el pin 18 Tx1 del Arduino UNO al pin Rx del módulo GPS.
- Conectar el pin 19 Rx1 del Arduino MEGA al pin Tx del módulo GPS

Figura 34

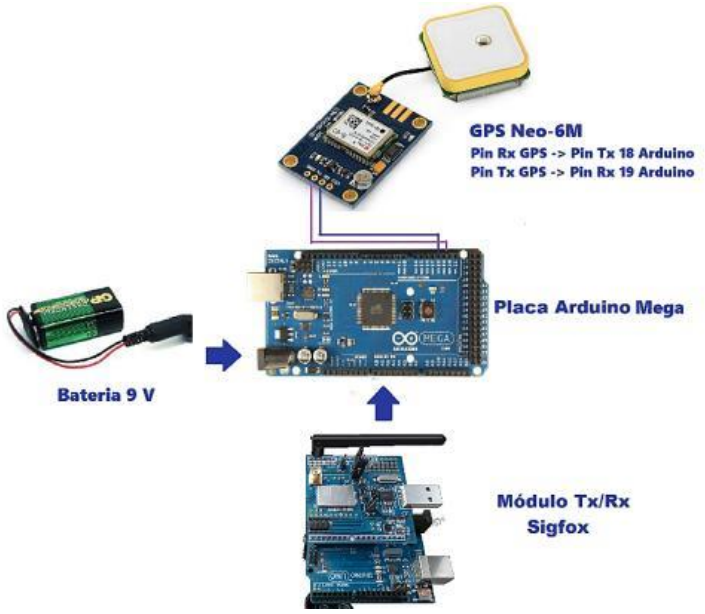
Bloque de adquisición de datos y sus elementos para el Nodo 1



Nota. Bloque de adquisición de datos del nodo 1 con mayor detalle y cada uno de sus elementos.

Figura 35

Bloque de adquisición de datos y sus elementos para el Nodo 2



Nota. El gráfico muestra el bloque de adquisición de datos del nodo 2 con mayor detalle y cada uno de sus elementos.

Bloque Red de Sigfox

Para el envío y recepción de la información obtenida mediante el bloque de adquisición de datos, es necesario utilizar una o varias estaciones base a través de la diversidad espacial con la que cuenta la Red Sigfox, lo cual significa que cada mensaje emitido por un dispositivo es recibido por múltiples estaciones base que se encargan de recibir dicho mensaje y los redirijen hacia el Sigfox Backend. Una vez los datos han llegado al Sigfox Backend este proporcionará al usuario una interfaz de aplicación web para la gestión de dispositivos y la configuración de la integración de datos, así como el API web basado en distintos estándares para automatizar la gestión de dispositivos e implementar la integración de datos.

Bloque del Sistema de Gestión y Visualización

La gestión de los datos entre Sigfox y Ubidots pasa por un "Callback" que permite redireccionar

los mensajes recibidos por el Backend hacia servidor de Ubidots para la creación de alarmas, notificaciones, gráficos y estadísticas en tiempo real.

El dispositivo Sigfox estará a cargo de leer los sensores y enviar los valores a Sigfox, el mensaje se decodificará utilizando la configuración de carga útil personalizada en las configuraciones de devolución de llamada para generar la solicitud para vincular el mensaje a Ubidots.

Base de Datos. Ubidots cuenta con una base de datos en la nube administrada que se proporciona como parte los servicios de Ubidots y permite organizar y almacenar la información de los dispositivos y variables registradas en Ubidots de forma permanente.

Ubidots Explorer. Ubidots cuenta con Ubidots Explorer, una aplicación móvil diseñada para permitir a los usuarios existentes de Ubidots crear y explorar aplicaciones de IoT que convierten los datos de los sensores en información práctica. Esta app está equipada con una API compatible con los protocolos HTTP, MQTT, TCP o UDP. Mediante esta aplicación, Ubidots proporciona una conexión simple y segura que es capaz de enviar y recuperar los datos que el usuario requiera (Espinosa & Orellana, 2021).

Figura 36

Bloque de Gestión y Visualización de datos

Nota. La figura representa un *Dashboard* creado en Ubidots y una aplicación móvil que permite la gestión y visualización de los datos.

Figura 37

Diagrama Esquemático de la configuración de un Callback de Sigfox a Ubidots Cloud

Nota. La figura muestra la configuración de un Callback y el uso de Ubidots por parte de los usuarios para visualizar y gestionar los datos obtenidos. Tomado de *Desarrollo de aplicaciones de monitoreo y control basadas en* IoT *a través de la plataforma Ubidots,* por Espinosa, B., & Orellana, M., 2021.

Despliegue e Instalación de los nodos

Las pruebas de ruta del prototipo fueron realizadas en la ciudad de Sangolquí, provincia de Pichincha, campus de la Universidad de Las Fuerzas Armadas “ESPE”.

Se optó por cubrir los dos nodos sensores por filamento PLA para que estén protegidos ante factores externos, algunas ventajas del uso de este material son: resistente a golpes, resistente al rayado, biodegradable y de fácil impresión.

Para el nodo GPS se imprimió en 3D una carcasa de 98 mm de ancho, 32 mm de largo y de alto. Esta carcasa cuenta con tres orificios, uno por donde se podrá conectar o desconectar la alimentación del dispositivo, otro por el cual será visible la antena del GPS Neo-6M debido a que esta necesita tener línea de vista y estar descubierta para que el módulo GPS sea más exacto, y el último orificio destinado a la antena de la placa Thinxtra.

Para el nodo CAN Bus se imprimió en 3D una carcasa de 64 mm de ancho, 88 mm de largo y de 39 mm alto. Esta carcasa cuenta con dos orificios, uno diseñado para la antena de la placa Thinxtra y el otro orificio está diseñada para poder conectar y desconectar el conector DLC OBD II, cuyo extremo

estará conectado a nuestro prototipo y el otro al terminal OBD II de un automóvil Kia Rio 2020, automóvil que se usará para las pruebas de ruta.

Figura 38

Nodo GPS con y sin tapa

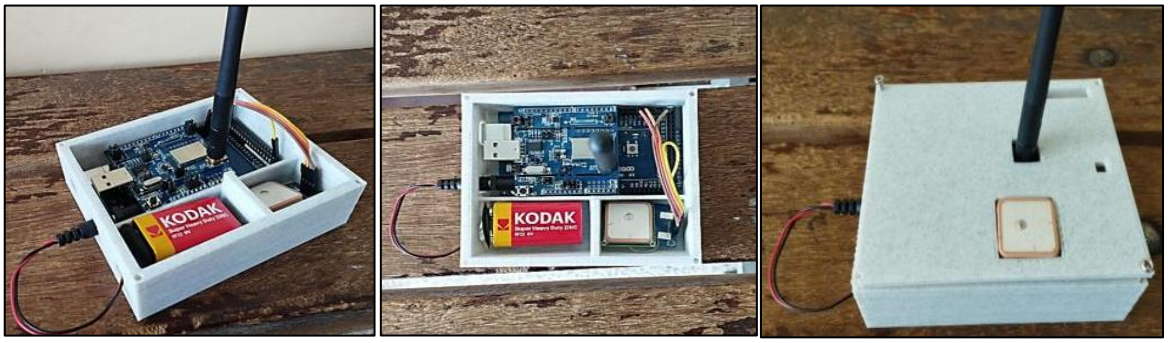
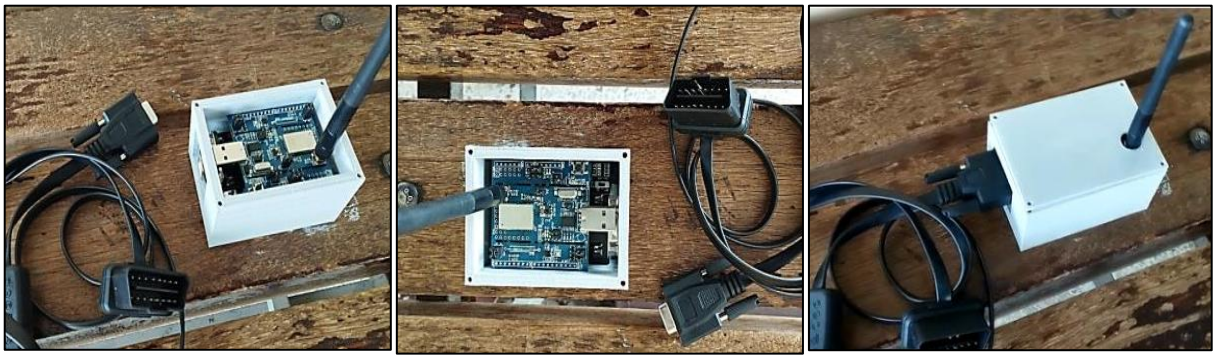


Figura 39

Nodo CAN BUS con y sin tapa



Nota. Los planos de estas carcasas se encuentran en los Anexos 1, 2, 3 y 4, correspodientes a las medidas y materiales.

Caracterización de vehículo

Para este trabajo de investigación se optó por usar un Kia Rio 2020 para las pruebas de ruta, ya que este vehículo es compatible con el protocolo de comunicaciones CAN y se rige bajo la normativa SAE J1979 que permite identificar los PIDs propios de OBD II.

Tabla 11

Características Kia Rio 2020

Característica	Valor
Motor	1.6 litros de 4 cilindros
Potencia Motor	6300 rpm
Transmisión	Manual
Cilindrada	1.396 cc
Sistema de Válvulas	DDHC 16 válvulas
Combustible	Gasolina

Nota. La tabla describe las características más importantes de un automóvil Kia 2020 de acuerdo a su fabricante. Tomado de *Student Learning Guide and Workbook,* por Kia, 2020.

Figura 40

Automóvil Kia Rio 2020 y su conector OBD II

Figura 41

Prueba de ruta del Sistema

Programación de los Nodos

En este capítulo se denota la implementación de los componentes que constituyen la propuesta, teniendo en cuenta el hardware y software a utilizar. Asimismo, se detalla las características principales de las librerías necesarias para la adquisición de datos y los algoritmos de programación realizados para cada etapa de la solución propuesta.

El entorno de desarrollo de Arduino IDE permite el uso de librerías, las cuales facilitan en gran medida el manejo y recolección de datos cuando se trabaja con sensores, también permiten optimizar los algoritmos al reducir la extensión del código.

Mayoritariamente estas librerías se pueden buscar e instalar desde el gestor de librerías, si está disponible. Si no está disponible en el gestor de librerias es necesario descargar el .zip del repositorio de Github, este es el caso de las librerías de Sigfox: Tsensor y Wisol que requieren del importe directo a partir de un archivo .zip disponible en el siguiente enlace: https://github.com/Thinxtra/Xkit-Sample.

A continuación se describen las librerías usadas en la programación de los nodos:

- **Tsensor.** Permite la lectura de los sensores internos de Xkit Thinxtra.
- **Wisol.** Controla del módulo Wisol del Xkit de thinxtra.
- **Wire.** Comunicación con dispositivos I2C/TWI.
- **SimpleTimer.** Permite trabajar con el tiempo.
- **Math.** Para realizar cálculos matemáticos
- **Avr/wdt.** Se encarga de manejar el watchdog de dispositivos AVR.
- **SPI.** Esta biblIoTeca le permite al Arduino comunicarse con dispositivos SPI.
- **TinyGPS.** Convierte los datos de posicionamiento global que se encuentran en formato NEMA en variables fáciles de usar para longitud, latitud, tiempo y altitud.

Diagramas de Flujo

El primer diagrama de flujo correspondiente al nodo 1 muestra gráficamente los pasos o procesos a seguir que están reflejados en el código, en él se puede apreciar la inicialización de los filtros y máscaras, el envío de la petición relacionado a cada PID (Ect, Iat, Tps, Etc, Map, Rpm), el tiempo de envió de mensajes, el proceso de lectura de los sensores de la ECU y el envío de esta información al Backend de Sigfox.

El segundo diagrama de flujo correspondiente al nodo 2 especifica los procesos propios de la adquisición y lectura de los datos (latitud y longitud) provenientes del módulo GPS Neo-6M, su activación, funcionamiento y el envío del payload de 8 bytes al Backend correspodiente a las variables de Ubicación en tiempo real del Vehículo.

Figura 42

Diagrama de flujo del Nodo 1

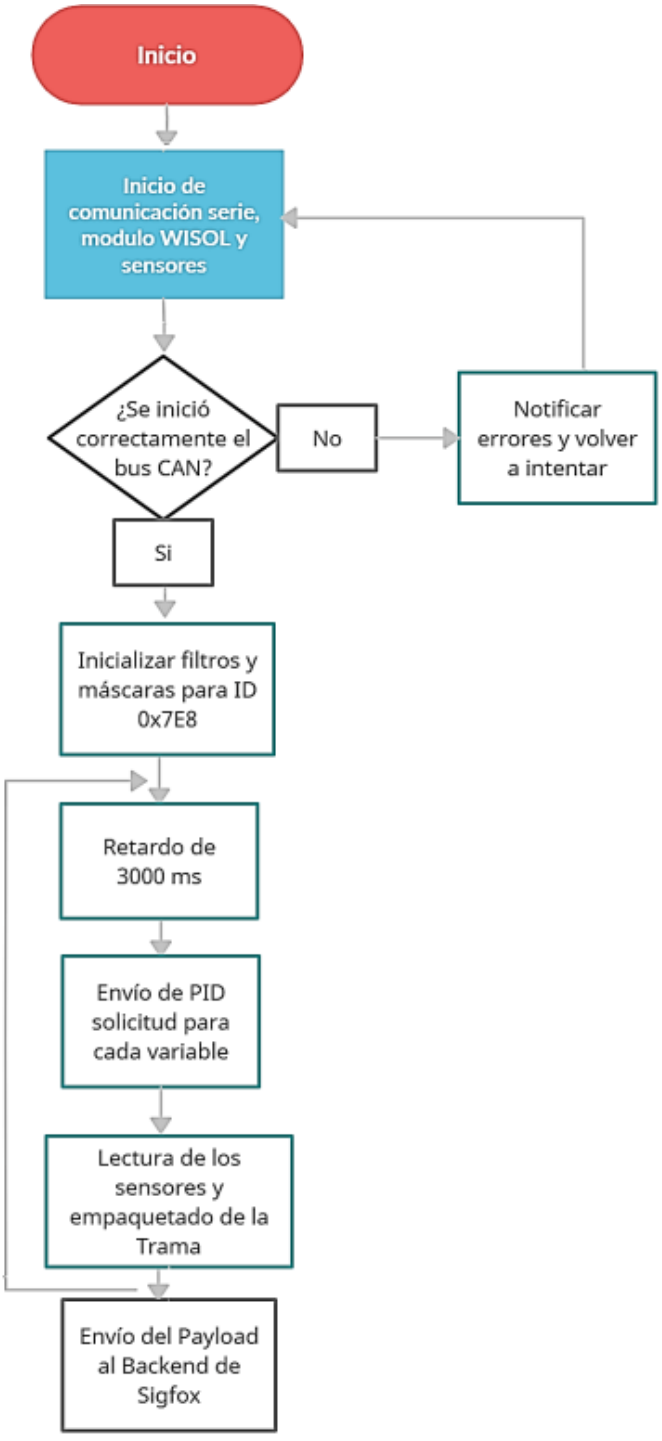
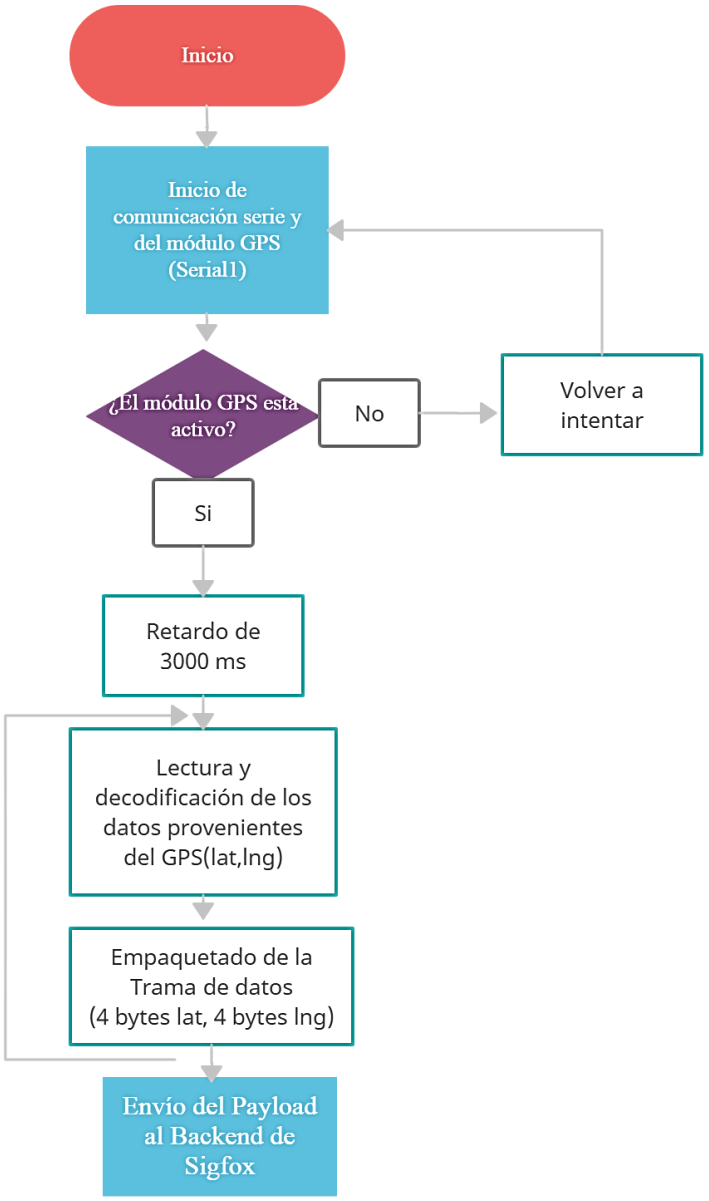


Figura 43

Diagrama de flujo del Nodo 2



Nota: En los diagramas de flujo se omiten subprocesos, ya que se ejecutan en los bloques principales.

Configuración del nodo 1 CAN Bus

La escritura del código de arduino comienza por establecer el SPI CS Pin de acuerdo con el hardware con el que se está trabajando.

```
const int SPI_CS_PIN = 9;
```

```
const int CAN_INT_PIN = 2;
#include "mcp2515_can.h"
mcp2515_can CAN(SPI_CS_PIN);
#define CAN_ID_PID        0x7DF
```

En este caso al trabajar con un MCP2515 hat se conoce que el pin cs de la versión posterior a v1.1 está predeterminado en D9, para v0.9b y v1.0 es D10 por defecto. Es así que para el Canal 0 Pin SPI_CS el pin declarado es 9 y el Pin de interrupción será 2.

A continuación, se inicializa los sensores internos del Xkit, las variables a usar y el módulo WISOL que se encarga de realizar la comunicación y envío de mensajes a la red Sigfox,

```
ISigfox *ISigfox = new WISOL();
Tsensors *tSensors = new Tsensors();
SimpleTimer timer;
```

Los nodos CAN Bus se pueden utilizar como receptor o transmisor de datos. Entonces llamamos al nodo CAN Bus un "transceptor", no existe el concepto de "Dirección" en el protocolo CAN, sino que cada dispositivo se distingue por una ID, es decir, cada dispositivo tiene una identificación especial.

El nodo CAN Bus se basa en su función de prueba de hardware para implementar una recepción selectiva de la trama de datos desde una identificación especial en el momento de la inicialización.

Inmediatamente después de comenzar a recibir los datos, se percibe la recepción de muchos mensajes con identificadores diferentes a los esperados. En el caso del diagnóstico, se deben enviar con el ID predefinido para la encuesta 0x7DF y se espera recibir respuestas identificadas por 0x7E8 o 0x7E9, que corresponden a los módulos más importantes dentro del sistema (ISO 15765-4).

Por este motivo se debe aplicar, dentro de esta función, el filtrado y enmascaramiento de mensajes, para así poder asegurar la recepción de los mensajes de diagnóstico únicamente. Las funciones que permiten implementar esto y configurar el ID especial son init_Mask() e init_Filt(), descritas a continuación.

Init_Mask(Masker_t Masker_num, INT8U ext, INT32U Data). Esta función se encarga de inicializar el registro de máscara.

Parámetros:

- **Masker_num.** Nombre de registro de máscara.
- **Ext.** "ext=0", configurar marco estándar con configuración de registro de máscara; "ext = 1", configure el marco extendido con la configuración de registro de máscara.
- **Data.** Escriba estos datos en el registro de máscara, para configurar qué registro se bloqueará.

Ejemplo:

Init_Mask(MCP_RXM0, 0, 0x3ff); Implementa un marco de filtro estándar de 0 a 9 bits, 10 bits en total, porque la forma binaria de "0x3ff" es "11 1111 1111".

Init_Filter(Filter_t Filter_num, INT8U ext, INT32U Data). Inicializa el registro del filtro de aceptación de mensajes.

Parámetros:

- **Filter_num.** Número de filtro de aceptación de mensajes.
- **Ext.** Si ext=0, denota que el filtro de aceptación de mensajes recibe solo el mensaje de marco de datos estándar; si ext=1, significa que el filtro de aceptación de mensajes solo recibe mensajes de marco de datos extendidos.
- **Data.** ID de mensaje filtrado. El controlador CAN solo puede recibir el marco de datos con identificación filtrada. Por lo tanto, si se puede recibir un próximo marco de datos, depende del valor en masker_num en la función init_Mask(), el valor en masker_num se registra en la función init_Filter() y el ID del identificador del mensaje próximo.

Estos tres valores se pueden consultar en la siguiente tabla. Si se reciben todos los resultados verdaderos, el controlador CAN recibirá el mensaje. De lo contrario, será descartado.

Tabla 12

Tabla de verdad de registro de filtro/ máscara

Bit Mascara	Bit Filtro	Bit de identificador de mensaje	Bit de aceptación o rechazo n
0	X	x	Aceptado
1	0	0	Aceptado
1	0	1	Rechazado
1	1	0	Rechazado
1	1	1	Aceptado

Nota. x es una variable aleatoria. Tomado de *Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores,* por Encinas et al., 2009.

La tabla 12 permite comprender el concepto de la máscara, que es permitir que se acepte un rango de ID de mensaje.

Ejemplo:

init_mask(MCP_RXM0, 0, 0x7ff);

init_Filter(MCP_RXF0, 0, 0x04); Trabajando dentro de la función init_Mask(). Establece 0x04 como ID de recepción del controlador CAN.

En cuanto a la programación de Arduino y en base a los conceptos ya expuestos, se establece ambas máscaras en 0x3ff y se establece el filtro de la siguiente manera para así poder recibir id de 0x04 a 0x09.

```
void set_mask_filt()
{
 CAN.init_Mask(0, 0, 0x7FC);
 CAN.init_Mask(1, 0, 0x7FC);
 CAN.init_Filt(0, 0, 0x7E8);
 CAN.init_Filt(1, 0, 0x7E8);
```

```
 CAN.init_Filt(2, 0, 0x7E8);
 CAN.init_Filt(3, 0, 0x7E8);
 CAN.init_Filt(4, 0, 0x7E8);
 CAN.init_Filt(5, 0, 0x7E8);
}
```

sendMsgBuf(INT32U id, INT8U ext, INT8U len, INT8U *buf). Envía un conjunto de tramas de datos.

Parámetros:

- **Id.** ID del marco de datos.
- **Ext.** "ext = 0", es un marco estándar; "ext = 1", es un marco extendido.
- **Len.** Longitud de datos, len < 8
- **buf.** Punto de búfer de datos

Ejemplo:

```
unsigned char data[8]={'R','O','B','O','T', '!'};
sendMsgBuf(0x06, 0, sizeof(data), data);
```

De manera que en el código de arduino se coloque las siguientes líneas, correspondientes al conjunto de trama de datos a enviar:

```
void sendPid(unsigned char __pid)
{
 unsigned char tmp[8] = {0x02, 0x01, __pid, 0, 0, 0, 0, 0};
 CAN.sendMsgBuf(CAN_ID_PID, 0, 8, tmp);
}
```

Dentro de la función void setup() se inicializa Xkit Thinxtra mediante el código Wire.begin(), también se realiza un testeo del módulo WISOL, que indica si se encuentra activo y entrega el ID de la placa mediante la función GetDeviceID().

En este apartado también se inicia la comunicación serie del arduino con la placa Thinxtra con

una velocidad de comunicación serial de 9600 baudios, se toma una velocidad del Can bus de 500k baudios y se define la frecuencia de envío de mensajes que será cada tres mil ms, además se llama a la función set_mask_filt(), cuya utilidad ya ha sido explicada.

```
void setup()
{
//Inicialización Xkit Thinxstra
Wire.begin();
Serial.begin(9600);
unsigned long tiempo_envio = 3000;
timer.setInterval(tiempo_envio,enviar_OBDII);
// WISOL test
 flagInit = -1;
 while (flagInit == -1)
    {
  Serial.println(""); //reset
  PublicModeSF = 0;
  flagInit = ISigfox->initSigfox();
  ISigfox->testComms();
  Serial.println("Sigfox Activo");
  GetDeviceID();
    }
while (CAN_OK != CAN.begin(CAN_500KBPS))
            { delay(100);
            }
 set_mask_filt();
}
```

A continuación se definen las funciones que se ejecutarán indefinidamente en la función void loop(), de manera que, en este apartado se especifican los identificadores asociados a cada una de las variables del vehículo y se llama a la función taskCanRecv() y la función senPid() para enviar la tramas de datos que permitan hacer la solicitud de la información requerida

Además se usa la función timer.run() que permite inicializar todos los temporizadores utilizados en el código y la función wdt.reset(), encargada de proteger la placa Thinxtra en caso de incidentes o fallos que pueden producirse al ejecutar el código.

```
void loop()
{
 timer.run();
 wdt_reset();
 watchdogCounter = 0;
  //1 rpm
  taskCanRecv();
  pid=0xC;
  sendPid(pid);
  delay(100);

  //2 velocidad del auto
  taskCanRecv();
   pid=0xD;
  sendPid(pid);
  delay(100);

  //3 iat
  taskCanRecv();
  pid=0x0F;
  sendPid(pid);
  delay(100);

  //4 tps
  taskCanRecv();
  pid=0x11;
  sendPid(pid);
  delay(100);
```

```
//5 temperatura refrigerante
taskCanRecv();
pid=0x05;
sendPid(pid);
delay(100);

//6 map
taskCanRecv();
pid=0x0B;
sendPid(pid);
delay(100);
}
```

Para leer el campo de datos perteneciente a la Trama CAN, que contiene los datos esta, es necesario conocer los PID OBD-II (ID de parámetros de diagnóstico a bordo), qué son códigos generalmente usados para solicitar datos de un vehículo, que se emplean mayoritariamente como herramienta de diagnóstico.

Muchos PID OBD-II están definidos en el estándar SAE J1979. Todos los vehículos y camiones de carretera comercializados y distribuidos en Norteamérica deben cumplir con un subconjunto de estos códigos, principalmente para las inspecciones de emisiones exigidas por el estado.

Asimismo, los fabricantes se encargan de definir PID adicionales específicos para sus vehículos. No todos los vehículos son capaces de admitir todos los PID y pueden existir PID personalizados previamente definidos por el fabricante que no están definidos en el estándar OBD-II.

El campo de datos de la trama Can está conformado por 8 bytes, de los cuales dependiendo de la longitud de la variable, se tomaran el byte 3 para variables que requieran de un solo byte para ser agregados al payload a enviar al Backend de Sigfox o el byte 3 y 4 para aquellas que necesitan de dos bytes.

Las figuras 44 y 45 muestran el campo de datos y la correspondiente interpretación del valor físico de los PID de revoluciones por minuto y velocidad del vehículo de 16 y 8 bits de longitud respectivamente

Figura 44

Campo de datos propio del PID 0C (Revoluciones por minuto)

PID	Name	Bit start	Bit length	Scale	Offset	Min | Max	Unit
0C	Engine speed	31	16	0.25	0	0 | 16384	rpm

	CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Request	7DF	02	01	0C	AA	AA	AA	AA	AA
Response (example)	7E8	04	41	0C	12	34	AA	AA	AA

Physical value (DEC)	=	0	+	0.25	*	4660	=	**1165**	**rpm**

Nota. Esta variable tiene una longitud de 16 bits o 2 bytes, por lo que se tomara en cuenta el byte 3 y 4 del campo de datos. Tomado de *OBD2 Explicado: Una Introducción Simple,* por CssElectronics, 2022.

Figura 45

Campo de datos propio del PID 0D (Velocidad de vehículo)

PID	Name	Bit start	Bit length	Scale	Offset	Min | Max	Unit
0D	Vehicle speed	31	8	1	0	0 | 255	km/h

	CAN ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Request	7DF	02	01	0D	AA	AA	AA	AA	AA
Response (example)	7E8	03	41	0D	12	AA	AA	AA	AA

Physical value (DEC)	=	0	+	1	*	18	=	**18**	**km/h**

Nota. Esta variable tiene una longitud de 8 bits o 1 byte, por lo que se tomara en cuenta solo el byte 3 del campo de datos. Tomado de *OBD2 Explicado: Una Introducción Simple,* por CssElectronics, 2022.

La siguiente tabla presenta los datos más importantes de cada PID valorado en este trabajo de investigación, además, proporciona la información sobre cómo traducir la respuesta en datos esperada

para cada PID, donde para cada fórmula se asigna el byte 3 como la variable A y al byte 4 como la variable B.

Tabla 13

Características de los PID

PIDs (hex)	PID (Dec)	Data bytes returned	Descripción	Min valor	Max valor	Unidad	Fórmula
0C	12	2	Revoluciones por minuto	0	16,383.75	Rpm	$\frac{256A+B}{4}$
0D	13	1	Velocidad del Vehículo	0	255	km/h	A
0F	15	1	Sensor IAT (Intake air temperatura)	-40	215	°C	$A-40$
11	17	1	Sensor TPS (Throttle position)	0	100	%	$\frac{100A}{25}$
05	5	1	Sensor ECT (Engine coolant temperature)	-40	215	°C	$A-40$
0B	11	1	Sensor MAP (Intake manifold absolute pressure)	0	255	KPa	A

Nota. La tabla indica los PIDs a analizar, su tamaño, descripción, unidad y fórmula. Tomado de *OBD2 Explicado: Una Introducción Simple*, por CssElectronics, 2022.

La Función taskCanRecv() se encarga de llamar a los siguientes comandos para leer los datos recibidos:

checkReceive(void). Verifica la validez del marco de datos recibido.

Parámetros: Ninguno

Retorno: Si el shield recibe el marco de datos válido, devuelve "CAN_MSGAVAIL"; si no, devuelve "CAN_NOMSG".

readMsgBuf(INT8U *len, INT8U *buf). Lee datos del búfer de recepción MCP2515.

Parámetros:

- **Len.** Guarda la longitud de los datos de recepción
- **buf.** Guarda los datos de recepción

De manera que, el código a escribir comprenderá los comandos readMsgBuf() y checkReceive() además de la correspondiente traducción de la información de cada PID de acuerdo a la fórmula especificada en la tabla 11, como se puede apreciar en el siguiente fragmento del código.

```
void taskCanRecv()
{
  unsigned char len = 0;
  unsigned char buf[8];
  if (CAN_MSGAVAIL == CAN.checkReceive())
 {
   CAN.readMsgBuf(&len, buf);// read data
   if(pid==0x0C) //rpm
    {
    numero=(buf[3]*256)+buf[4];// conversión información
    rev=numero*0.25;
    word rev1=rev;
    OBDII[0]=highByte(rev1);
    OBDII[1]=lowByte (rev1);
    }
if(pid==0x0D)//speed
    {
    numero=buf[3];
    vel=numero*1;
    OBDII[2]=vel;
    }
  if(pid==0x0F)//IAT
    {
    numero=buf[3];
    IAT=(numero-40);//restar 40
```

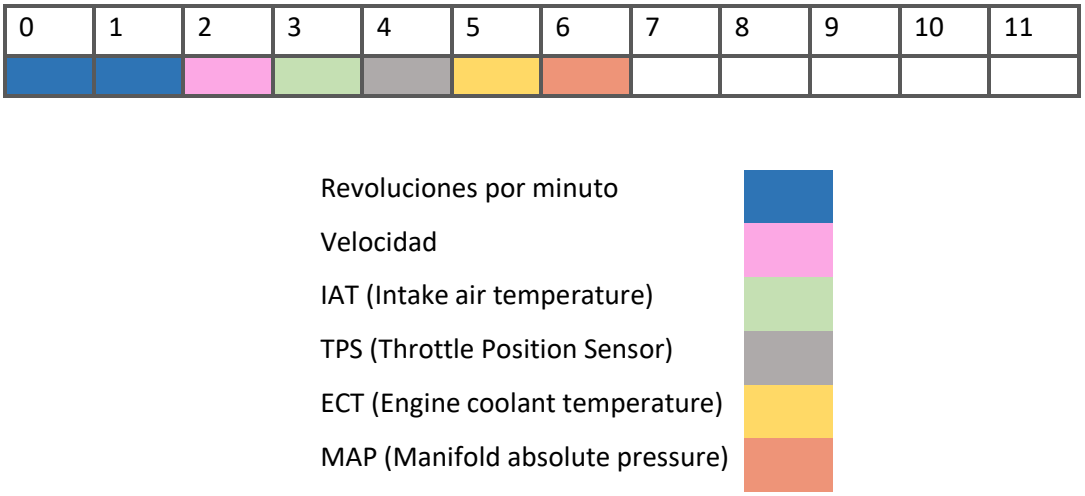
```
      OBDII[3]=IAT;
      }

    if(pid==0x11)//TPS
      {
      numero=buf[3];
      TPS=(numero*0.4);
      OBDII[4]=TPS;
      }
     if(pid==0x05)//Temperatura refrigerante
      {
      numero=buf[3];
      ECT=(numero);//restar 40
      OBDII[5]=ECT;
      }
     if(pid==0x0B)//MAP
      {
      numero=buf[3];
      MAP= numero;
      OBDII[6]=MAP;
      }
    }
   }
```

A partir de la lectura y conversión de los valores las variables descritas en la tabla en un tipo de dato de tamaño de 2 bytes o 1 byte dependiendo de la variable, es posible construir el payload o trama de 12 bytes que posteriormente será enviada al Backend de Sigfox. En este nodo el valor del sensor de rpm ocupa 2 bytes y los valores de los otros 5 sensores ocupan 1 byte cada uno, de manera que, en su totalidad se ocupará 7 bytes de payload, a continuación se muestra el formato del payload construido con los datos de cada sensor.

Figura 46

Payload construido, a partir de los datos capturados de los sensores de la ECU



Debido a que los mensajes del Backend de Sigfox son decodificados en tramas de un tamaño de 12 bytes es necesario convertir las variables utilizadas anteriormente a variables tipo UINT8 que tienen un tamaño de 1 byte u 8 bits, esto se especifica en el argumento de la función envio_dato(). UINT8 es un tipo de variable, denominada así por ser para enteros sin signo, permite un máximo de 8 bits de almacenamiento, por el contrario UINT16 permite un máximo de 16 bits.

```
void enviar_OBDII()
{
 envio_dato(OBDII,12);
}
void envio_dato(uint8_t *sendData, const uint8_t len)
        {
    // No se requiere de mensaje de enlace descendente
    recvMsg *RecvMsg;
    RecvMsg = (recvMsg *)malloc(sizeof(recvMsg));
    ISigfox->sendPayload(sendData, len, 0, RecvMsg);
    for(int i = 0; i < RecvMsg->len; i++)
            {
            Serial.print(RecvMsg->inData[i]);
```

```
                }
        free(RecvMsg);
              }
```

La función SendPayload() envía el mensaje empaquetado de 12 bytes con el valor de cada uno de los sensores de la ECU al Backend de Sigfox para una posterior decodificación.

Configuración del nodo 2 GPS

La escritura del código de arduino comienza por establecer la estructura de coordenadas GPS, considerando un paylaod de 12 bytes, donde cada variable tiene un tamaño de 32 bits o 4 bytes de espacio.

```
struct gpscoord
{
 float a_latitude;  // 4 bytes
 float a_longitude; // 4 bytes
};
gpscoord coords = {lat,lon};
```

Como se explicó en la configuración del nodo 1, el proceso para inicializar los sensores internos del Xkit y el módulo WISOL es el mismo.

```
ISigfox *ISigfox = new WISOL();
Tsensors *tSensors = new Tsensors();
SimpleTimer timer;
```

De la misma forma en la función void setup() se inicializa Xkit Thinxtra mediante el código Wire.begin(), se testea el módulo WISOL y se define la frecuencia de envío de mensajes.

Se inicializan los pines Tx1 y Rx1 (pines 18 y 19 respectivamente) para que puedan ser usados como puerto serial y así sea posible realizar la conexión del Arduino con el Módulo GPS, ya que el Rx0 y Tx0 (puerto serie por Hardware) estará siendo ocupado por la conexión existente entre el Arduino Mega y la placa Thinxtra.

```
void setup()
```

```
{
 Wire.begin();
 Serial.begin(9600);
//Serial GPS Pin TX1 18 RX1 19
 Serial1.begin(9600);
 unsigned long tiempo_envio = 3000;
 timer.setInterval(tiempo_envio,enviar_OBDII);
}
```

Dentro de las funciones ejecutadas indefinidamente se encuentra Serial1.available(), que retorna el número de bytes disponibles para su lectura desde el puerto serie, comprobando la llegada de datos y su almacenamiento en el buffer de recepción serie, en este caso verificando si los datos del GPS están disponibles y han sido recibidos correctamente.

Otra de las funciones a usar son:

- **Gps.encode().** Indica cuando los datos nuevos se han decodificado por completo y se pueden usar.
- **Gps.f_get_position().** Obtiene la posición, donde la latitud y la longitud son variables de tipo flotante, y se devuelven los valores reales de las coordenadas requeridas. Los tipos flotantes son más fáciles de usar, pero dan como resultado un código más grande y más lento (Hart, 2013).

```
void loop()
{
   if(Serial1.available())     // verificación de datos del GPS
{
    Serial.println("Activo GPS");
    if(gps.encode(Serial1.read()))       //  codificación datos
{
    gps.f_get_position(&lat,&lon);       // obtención de latitud y longitud
              }
     }
 gpscoord coords = {lat,lon};
```

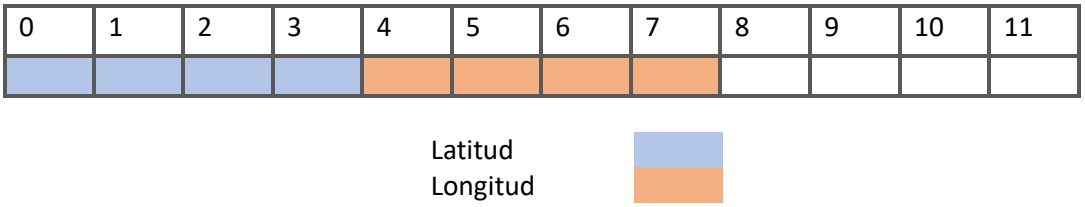
```
uint8_t* bytes = (uint8_t*)&coords;

}
```

Una vez se adquiere las variables de latitud y longitud, se procede a almacenar las coordenadas obtenidas en la estructura dedicada ya establecida al inicio del código y se convierte estos datos flotantes de GPS en datos del tipo UINT8, para así construir la trama que posteriormente será enviada al Backend mediante la función envío_dato().

Figura 47

Payload construido, a partir de los datos obtenidos del módulo GPS



Nota. Se representa la carga útil personalizada para latitud y longitud.

La función envío_dato() toma como argumento los bytes ya transformados en datos UINT8 y el tamaño de los mismos de acuerdo a la estructura que estos tienen.

```
void enviar_GPS()
{
 envio_dato(bytes[12],sizeof(gpscoord));
}
void envio_dato(uint8_t *sendData, const uint8_t len)
        {
   recvMsg *RecvMsg;
   RecvMsg = (recvMsg *)malloc(sizeof(recvMsg));
   ISigfox->sendPayload(sendData, len, 0, RecvMsg);
   for(int i = 0; i < RecvMsg->len; i++)
            {
            Serial.print(RecvMsg->inData[i]);
            }
```

```
free(RecvMsg);
```

El comando Sendpayload() envía el mensaje empaquetado de 12 bytes donde 4 bytes corresponden a la latitud y 4 a la longitud para un total de 8 bytes al Backend de Sigfox para una posterior decodificación.

Registro de los Equipos

Los dispositivos deben estar registrados antes de que puedan conectarse a la red Sigfox. Para registrarse y crear una cuenta para poder usarlos, para lo cual es necesario acceder a la página de Sigfox Backend. Este registro consiste en ingresar la información personal del usuario y lo más importante el ID del dispositivo. En la figura 48 se visualiza la creación del perfil nuevo y el registro del Callback en el Backend de Sigfox.

Figura 48

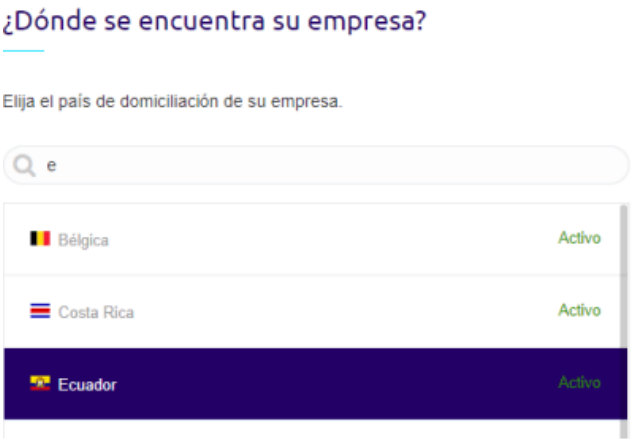
Interfaz para la activación de Devkit

Nota. Pantalla de creación de cuenta de usuario de Ubidots. Tomado de *Sigfox Techinical Overiew*, por Sigfox, 2022, Sigfox (https://www.Sigfox.com/en/login/).

Se debe seleccionar la opción Activate my DevKit ya que este registro permite obtener un año de servicio de forma gratuita y a su vez el envío de hasta 140 mensajes por día, cada uno de estos mensajes con un máximo de 12 bytes.

Figura 49

Selección del país de uso de la placa



El dispositivo seleccionado para el desarrollo del proyecto es el módulo Thinxtra Xkit RCZ4 y cuenta con un ID único y un PAC (código de uso único) que deberán ser ingresados para su registro en la plataforma oficial de Sigfox. Estos datos están disponibles en el empaque de fábrica de la placa de desarrollo. Además, en esta interfaz se deberá ingresar obligatoriamente una breve descripción y el propósito del proyecto.

Figura 50

Registro del ID y PAC de la placa

Otros datos adicionales que se solicitan en el registro son algunos datos personales del usuario y un correo electrónico donde posteriormente se enviaran los respaldos de la información cuando sean requeridos por el usuario.

Sigfox Backend

Inmediatamente después del registro de los equipos, es posible acceder a la cuenta y por lo tanto al Backend de Sigfox ingresando el usuario y contraseña. Aquí se muestra el listado de dispositivos registrados.

Figura 51

Dispositivo con ID: 4127BD registrado en el Backend correspondiente al nodo 1

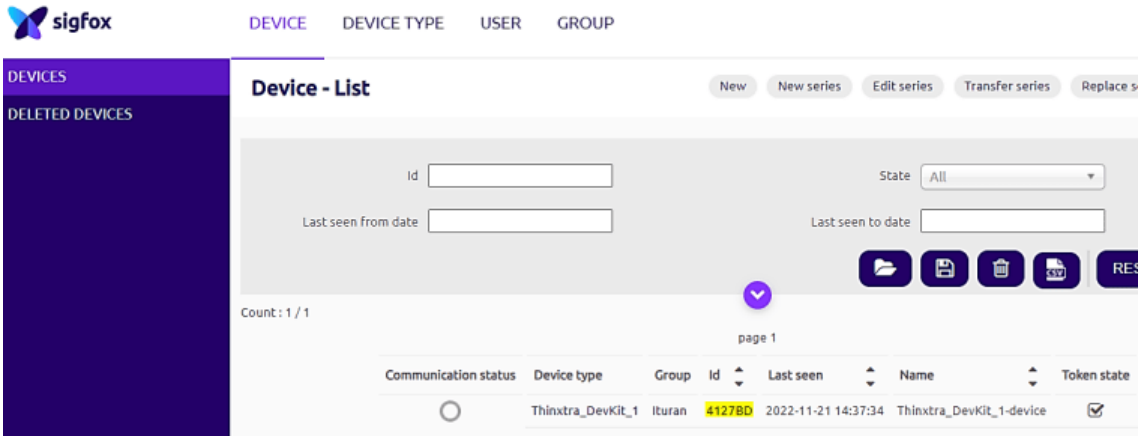
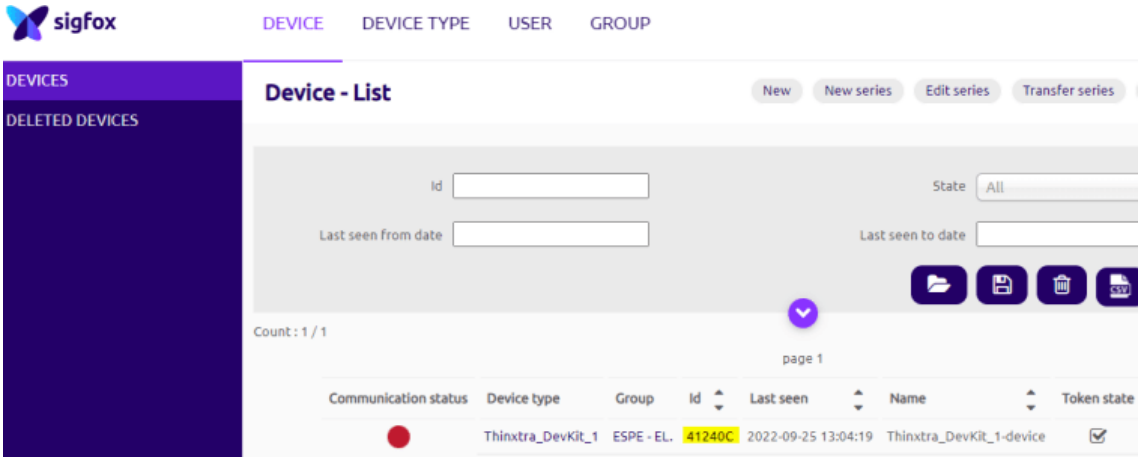


Figura 52

Dispositivo con ID: 41240C registrado en el Backend correspondiente al nodo 2



Al dar clic en el hipervínculo ID se redirige al usuario a una nueva ventana que presenta información general del dispositivo en varios parámetros, los más importantes son:

1. **Información.** Muestra la información general más relevante del dispositivo como el nombre, PAC (código de uso único), tipo de dispositivo, protocolo, estado, indicador de calidad de señal, estado de comunicación, fecha de activación y expiración del token.
2. **Mensajes.** Evidencia el listado de mensajes enviados por el dispositivo y cada uno cuenta con distintos parámetros como:

2.1 **Time.** Indica la hora y fecha que fue enviado el mensaje.

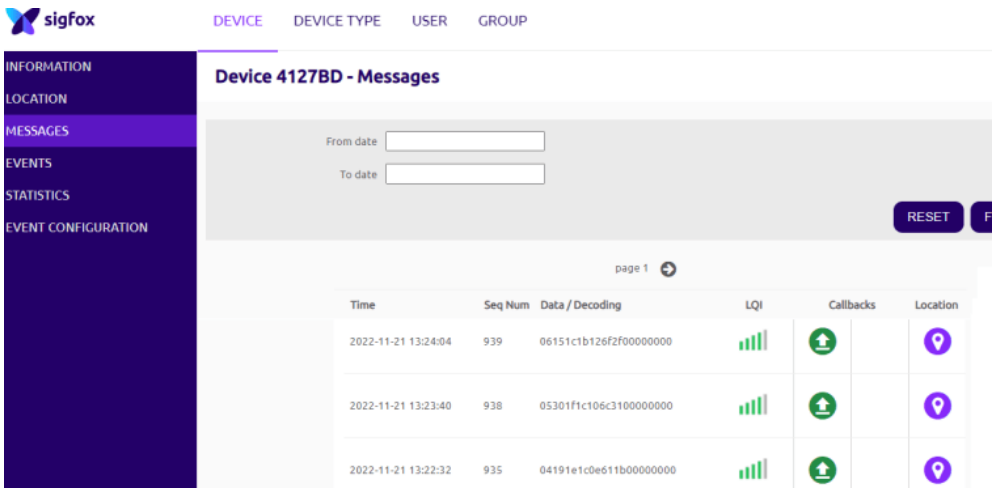
2.2 **LQI (Link Quality Indicator).** Es un indicador de la intensidad de la señal de la red en el momento que el mensaje fue enviado.

2.3 **Seq number.** Es un contador que va aumentando cada vez que se envía un dato al Backend, este conteo de datos es independiente de la intensidad de la señal de la red, es decir cuenta los intentos para enviar de forma exitosa un Callback.

3. **Callback.** Notifica cuando un Callback fue enviado correctamente, al darle click se puede comprobar la información decodificada de acuerdo a su configuración.

Figura 53

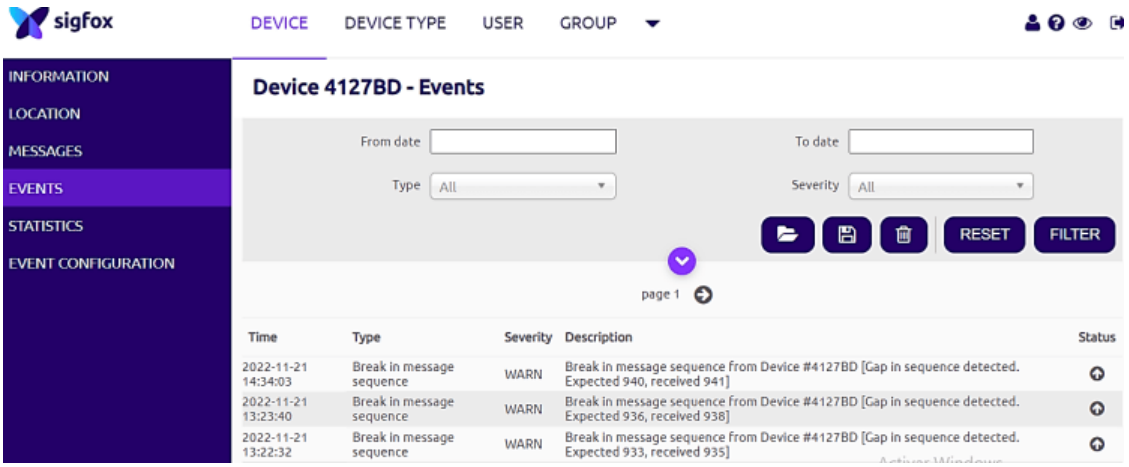
Apartado de Mensajes



4. **Eventos.** Muestra un listado con cada uno de los errores existentes al enviar mensajes, indica cada SEQ NUMBER que no pudo ser registrado en el apartado de Mensajes visto anteriormente. Estos errores pueden producirse por fallas en la cobertura de la red o una cobertura irregular.

Figura 54

Apartado de Eventos

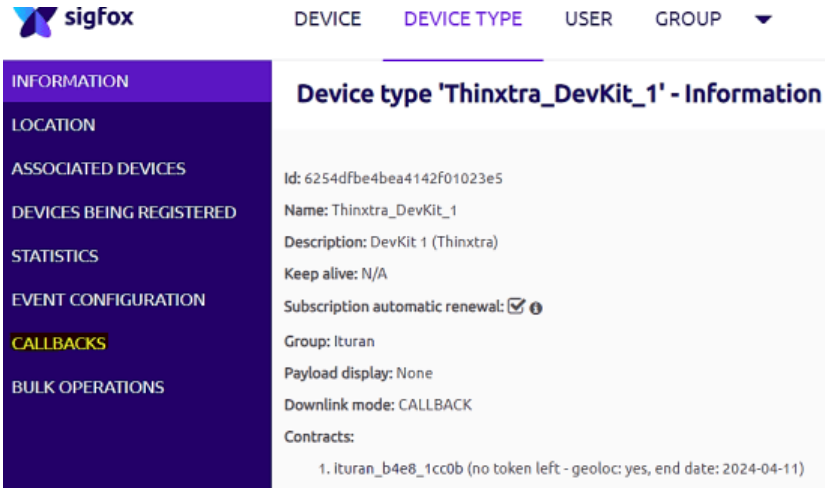


Creación del Callback

En el tipo de dispositivo al dar clic sobre el hipervínculo que corresponde al nombre del dispositivo se desplegará una nueva ventana donde es posible configurar un Callback.

Figura 55

Ventana Device Type



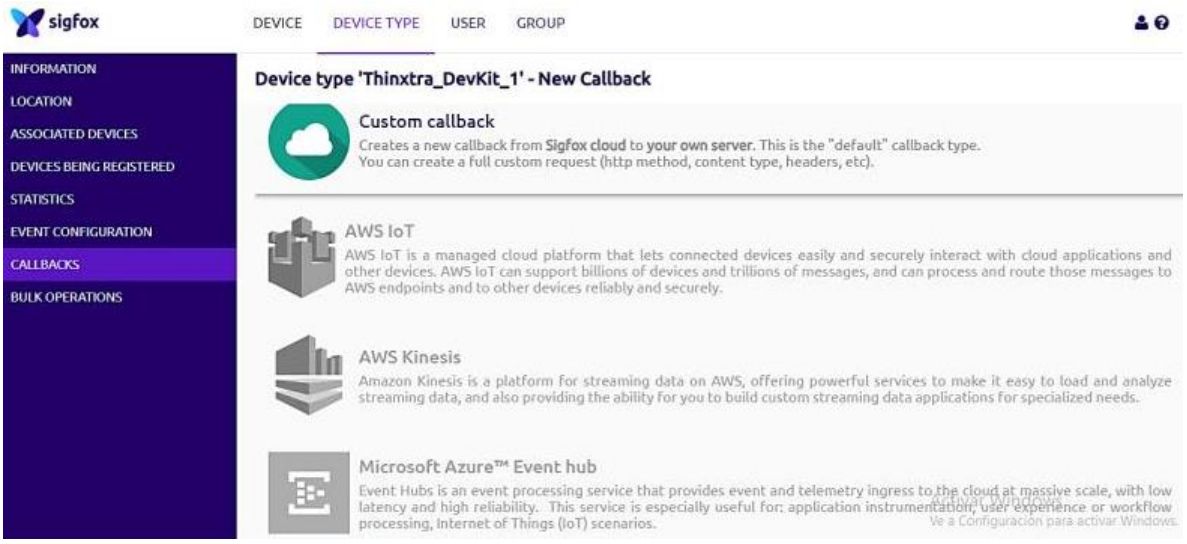
En un Callback se configura que hacer con los mensajes una vez que han llegado al Backend de Sigfox, es decir cómo manejarlos, de manera que permite redireccionarlos por medio de solicitudes HTTP con mensajes tipo GET o POST para el uso de un servidor externo o una transmisión directo de datos mediante el uso de plataformas cuyo funcionamiento esté basado en la nube.

En esta ventana se visualiza las diferentes opciones de creación de un nuevo Callback para distintos entornos IoT como por ejemplo AWS Kinesis, Azure, IBM Watson.

Para el desarrollo de este proyecto se usará la plataforma Ubidots para diseñar la aplicación web que funcionara como servidor, por lo que seleccionamos Custom Callback para configurar un nuevo Callback.

Figura 56

Ventana de creación de Callbacks



Decodificación de la trama de datos

Todos los valores sensados son transmitidos desde la placa Thinxtra y son recibidos en el Backend de Sigfox como una trama hexadecimal, por este motivo es necesario realizar una decodificación del payload para poder configurar correctamente el Callback.

Esta decodificación se realiza en el Backend de Sigfox seleccionando la opción “Custom

Callback", donde se despliega una nueva ventana que permite configurar ciertos parámetros relacionados con la personalización de la carga útil y el método de petición HTTP a usar. Se creará dos Callbacks diferentes, uno para cada nodo, es decir la configuración del nodo CAN Bus será diferente a la configuración del Callback para el nodo GPS.

A pesar de que en la ventana de mensajes el payload se muestre como una trama en hexadecimal, la decodificación de los datos se toma como bytes y cuenta con una gramática propia de decodificación para cada tipo de mensaje personalizado, es así que el campo se compone por los siguientes elementos:

Nombre. Es el identificador en el cual puede incluir números, letras y caracteres.

Índice de bytes. Es el desplazamiento en el búfer de mensajes desde donde se va a leer el campo, comenzando en cero. Si se omite, para el primer campo, una posición omitida significa cero (inicio del búfer de mensajes).

Posteriormente va el nombre del tipo de la variable y los parámetros que cambian según el tipo. Para la decodificación de nuestra trama hexadecimal se utilizaran dos tipos de datos (float y uint), cuyos parámetros se describen a continuación:

- **Flotante (float).** Los parámetros son la longitud en bits del dato, pueden ser de 32 o 64 bits.
- **Entero sin signo (uint).** Los parámetros son directamente la cantidad de bits a introducir en el valor.

A partir de los parámetros explicados en esta sección, la configuración personalizada de la carga útil del primer nodo correspondiente al nodo can bus es la siguiente:

rpm::uint:16 vel:2:uint:8 iat:3:uint:8 tps:4:uint:8 ect:5:uint:8 map:6:uint:8

Donde solamente la variable rpm utilizará dos bytes o 16 bits de la trama, y las variables velocidad, iat, tps, ect y map utilizaran un byte U 8 bits cada una, todas estas variables manejan tipos de datos uint.

La configuración personalizada de la carga útil del segundo nodo correspondiente al nodo GPS es la siguiente:

lat::float:32 lng::float:32

Donde las dos variables manejan tipos de dato float debido a su naturaleza, además, longitud y latitud utilizaran 4 bytes o 32 bits de la trama cada una.

Configuración del Callback

Sigfox permite al usuario redireccionar todo el volumen de los datos que ingresan al Backend hacia cualquier aplicación que se ejecute en un servidor o una plataforma de procesamiento de información. La adquisición de los datos del Backend de Sigfox se efectúa a través del reconocimiento de una URL (canal) hacia la aplicación web a la cual se desea transmitir los mensajes.

Puesto que desde el Backend de Sigfox no es posible configurar una alarma y la información adquirida es temporal, ya que en este apartado solamente se puede redireccionar los datos del mensaje mandado por el dispositivo Sigfox, estos datos se reenvían mediante dos Callbacks a Ubidots.

La misión de Ubidots es analizar el mensaje, almacenar los datos, representarlos gráficamente y finalmente arrojar un evento de alarma cuando corresponda, es decir, al superar el umbral máximo de funcionamiento de cada variable. Estas son tareas que el Backend de Sigfox no puede realizar. La ventaja de usar un servidor/plataforma como Ubidots es que la información es almacenada en nuestra propia infraestructura y es accesible para siempre, considerando que Ubidots cuenta con una base de datos incorporada. Para configurar los Callback es necesario conocer el significado de cada uno de los siguientes campos y como rellenarlos:

- **Canal.** Se debe seleccionar que permite enviar los datos a una sola URL de destino.
- **Configuración de carga personalizada.** Esta opción permite al usuario decodificar la carga útil en variables distintas y simples dentro de la GUI de Backend, es importante colocar en este apartado la carga personalizada ya descrita para cada nodo.

- **URL Pattern.** Define la solicitud HTTP que se enviará a su servidor, incluidas las variables disponibles. Es importante definir el puerto específico que es compatible con el canal URL, este puerto esta predefinido por la plataforma o servidor (Ubidots).
- **Método HTTP.** Pueden ser utilizados tres métodos http distintos (Get, post, put), en este caso se utiliza un método post que permite enviar información al servidor.
- **Encabezado.** Se coloca un token propio de Ubidots, que funciona como una clave única que permite al dispositivo ingresar datos a la plataforma.
- **Tipo de contenido.** Se coloca application/json debido a que Ubidots admite ese formato de texto para el intercambio de datos.
- **Body.** Se debe colocar las variables que se enviarán en el cuerpo considerando la notación que debe usarse.

Figura 57

Configuración final del Callback correspondiente al nodo CAN BUS

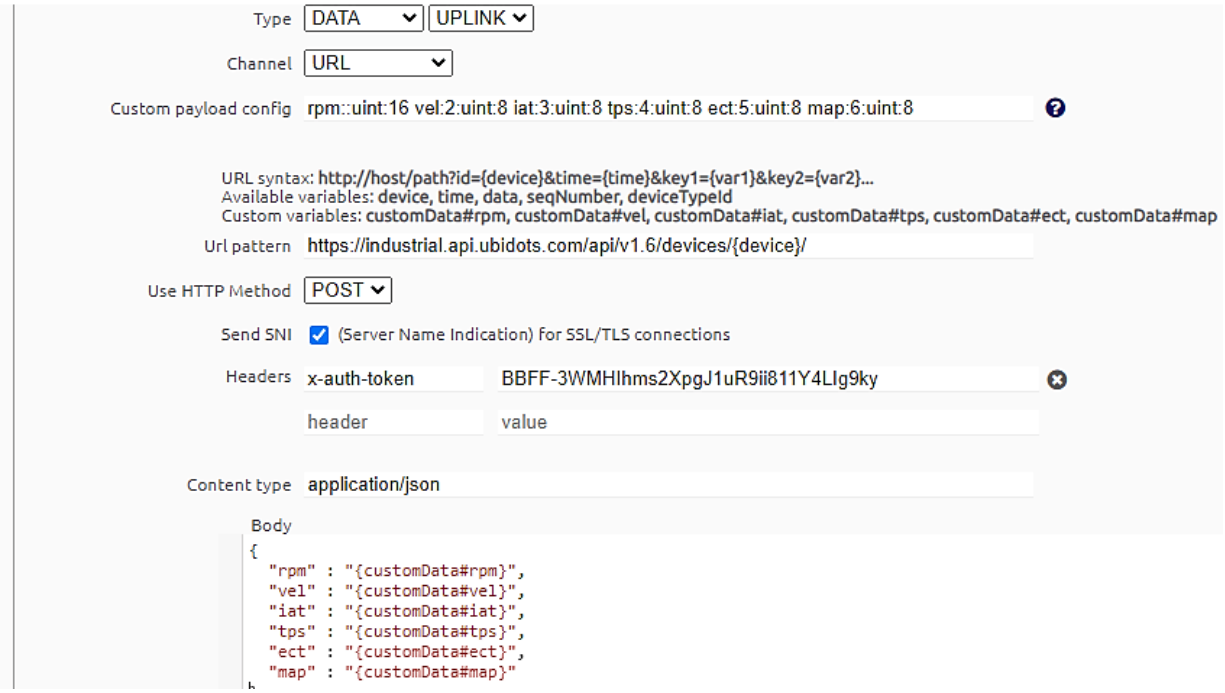
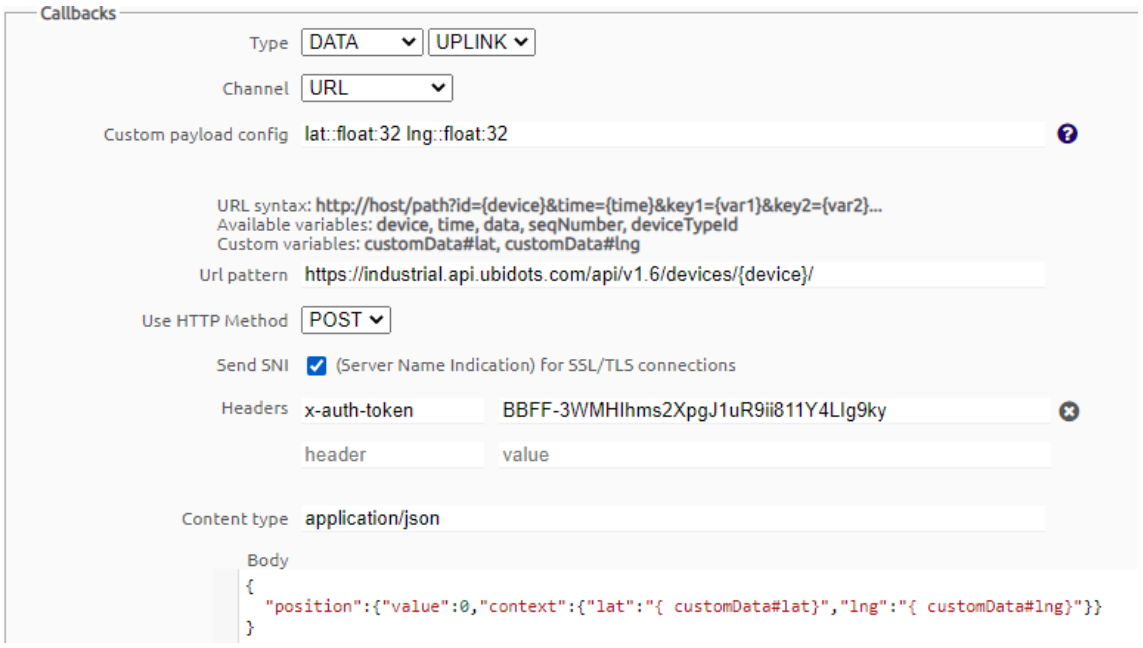


Figura 58

Configuración final del Callback correspondiente al nodo GPS



Plataforma de gestión y manejo de datos

Se seleccionó como plataforma gestión, manejo y visualización de datos a Ubidots, que al ser una plataforma certificada con Sigfox, puede comunicarse a través de una URL o una URL por lotes con Sigfox y le permite al usuario desarrollar aplicaciones web con la integración de dispositivos IoT.

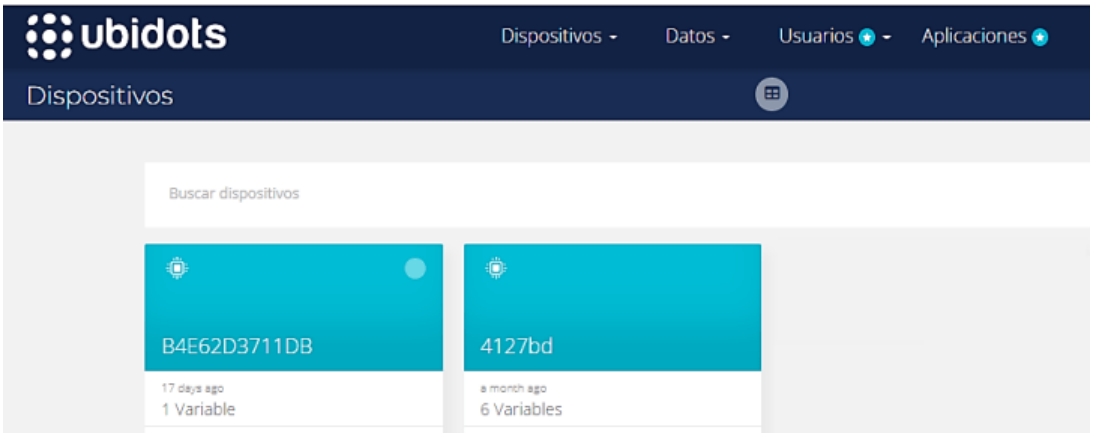
Para usar Ubidots es necesario crear una cuenta gratuita, una vez creada una cuenta podremos visualizar y gestionar las variables en tableros mediante *widgets* y otras herramientas, sin embargo, al ser una cuenta gratuita existen ciertas limitaciones como: un máximo de tres tableros para usar y diez *widgets* por tablero. A pesar de esto y debido a sus virtudes, se considera a esta plataforma como la adecuada para los requerimientos de este trabajo de investigación en la monitorización de variables y parámetros propios de un vehículo y su geolocalización.

Una vez ingresamos a Ubidots accederemos a la pestaña dispositivos, en este apartado podremos observar los dispositivos registrados y sus variables, cuya creación se producirá automáticamente cuando los dispositivos IoT comiencen a enviar mensajes al Backend de Sigfox, todo

esto gracias a la previa configuración de los dos Callbacks para cada dispositivo y la recepción de un punto del token ingresado.

Figura 59

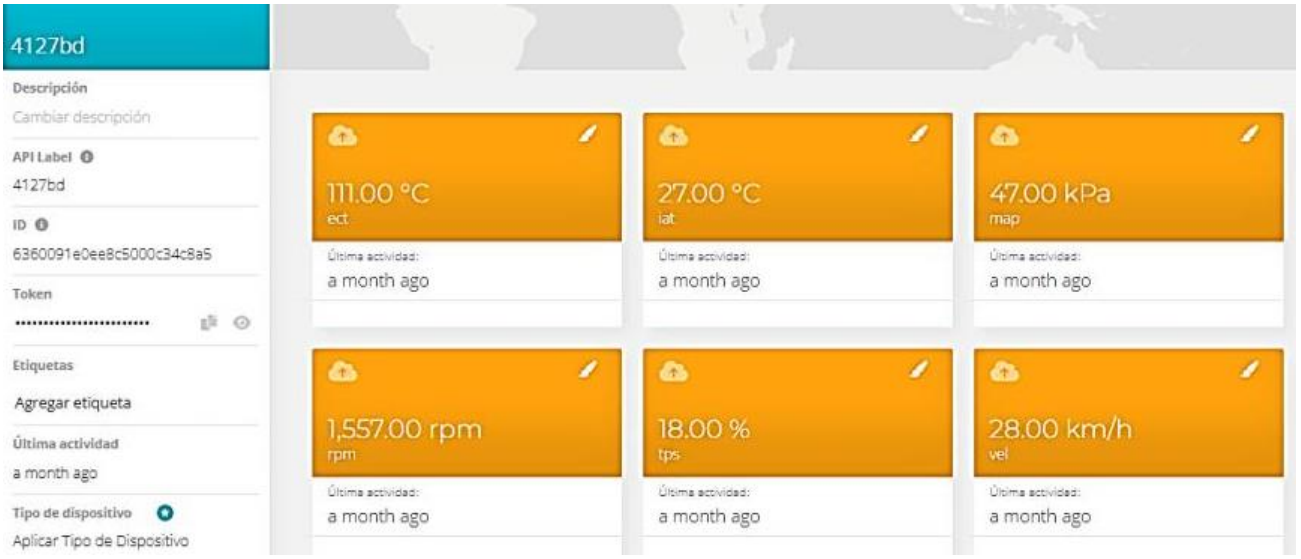
Dispositivos vinculados a Ubidots



En las siguientes figuras podemos ver las variables asociadas al nodo CAN Bus, y como los valores propios de cada una son almacenados en la base de datos que incorpora Ubidots.

Figura 60

Variables asociadas al dispositivo del nodo CAN Bus



Como ejemplo podemos observar el almacenamiento de los valores numéricos propios de la variable “ECT”, donde también se encuentra la hora y fecha de llegada de cada mensaje y su valor.

Figura 61

Valores almacenados de la variable ECT

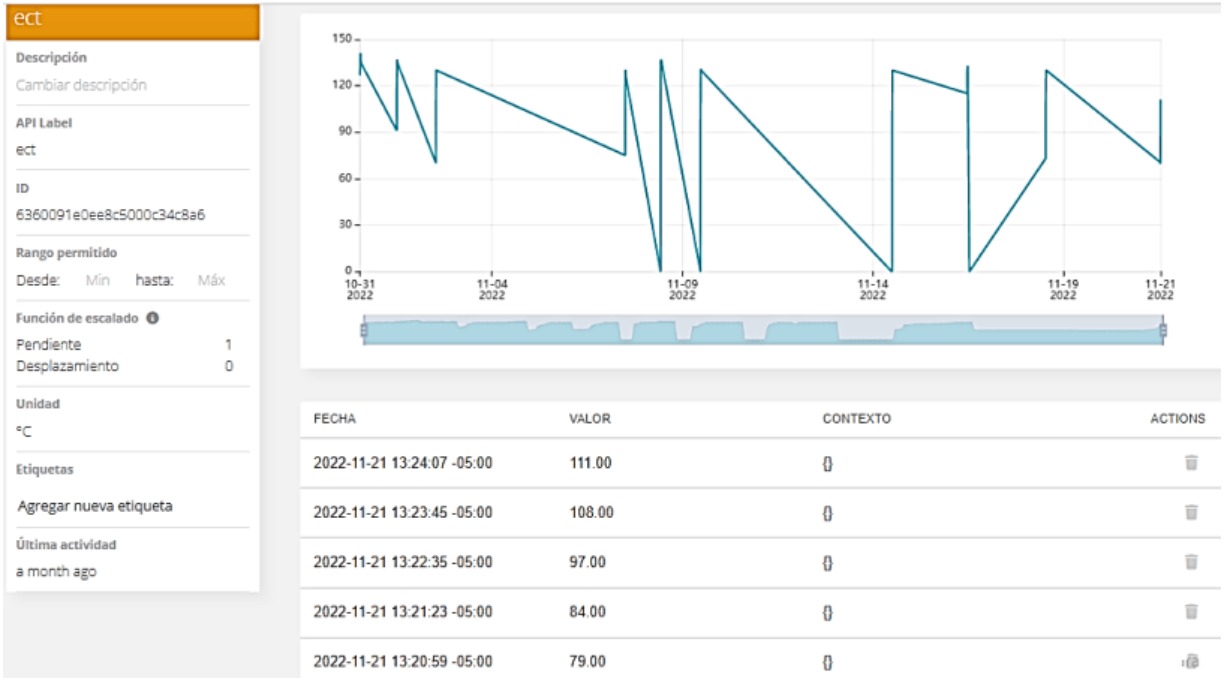
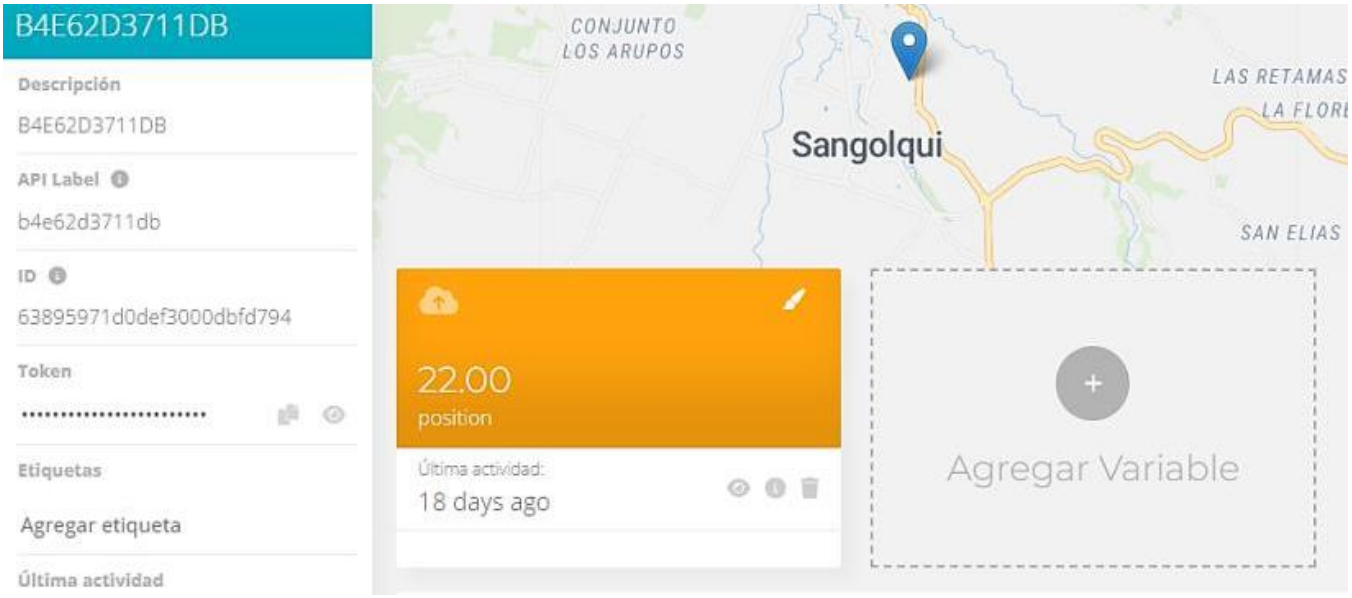


Figura 62

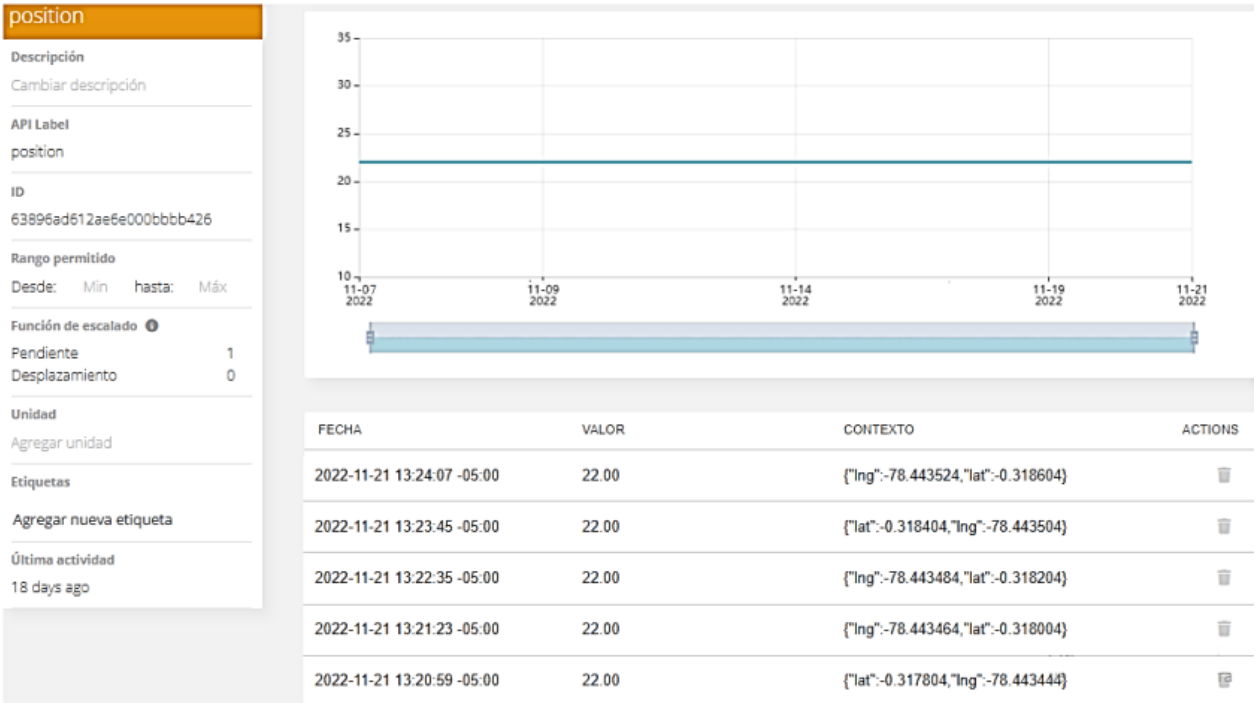
Variables asociadas al dispositivo del nodo GPS



Los valores numéricos no son el único tipo de dato admitido por esta plataforma, también es posible almacenar el tipo de datos “cadena” dentro de lo que denominamos contexto. De manera que la variable “position” asociada al nodo GPS usará un contexto, que al ser un objeto clave-valor, permite almacenar no solo los valores numéricos sino también de cadena, siendo posible enviar la latitud y longitud en una sola variable y luego recuperarlas.

Figura 63

Valores almacenados de la variable position y su contexto (latitud y longitud)

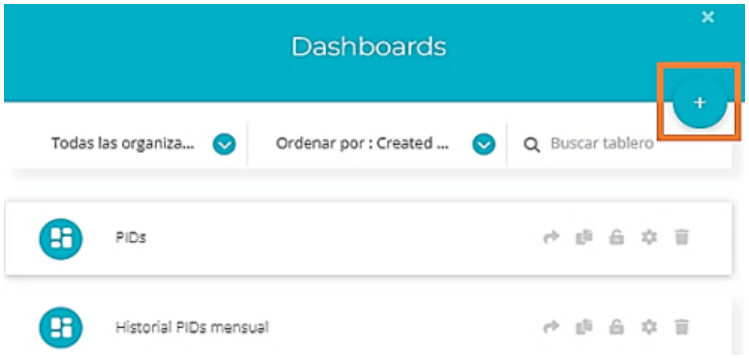


Creación del Tablero

Los tableros son las interfaces hombre-máquina donde podemos visualizar los datos fácilmente, Ubidots permite crear un número limitado de paneles y *widgets*. Estos tableros permiten realizar un seguimiento de nuestro proceso de adquisición y gestión de datos, y monitorearlo. Para crear un tablero estático o dinámico, nos dirigimos a “Datos”->”Tableros” y hacemos clic en el ícono “+” que añadirá un nuevo tablero a la interfaz.

Figura 64

Creación de tableros

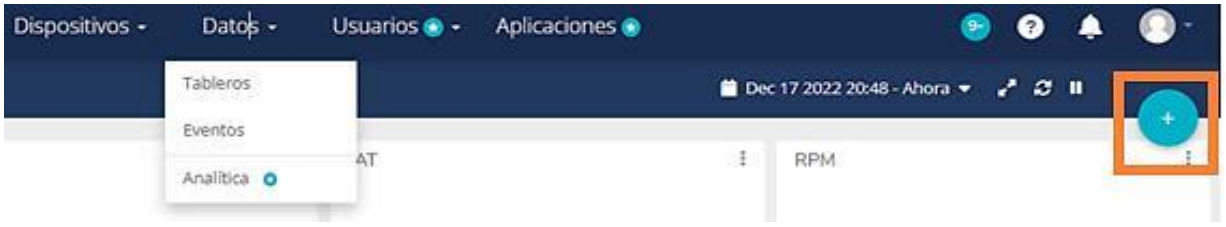


Después de definir un nombre para nuestro tablero, procedemos a crear los *widgets* que conformaran el tablero, estos *widgets* permiten trazar los datos según nuestras necesidades, y cada uno de ellos tiene un conjunto diferente de opciones de configuración, es así que para nuestro nodo CAN Bus crearemos tres tipos diferentes de *widgets*: Métrica, medidor y gráfico de líneas.

Para crear un nuevo widget ya sea estático o dinámico, hacemos clic en el ícono "+" que se encuentra en la esquina superior derecha de la interfaz de usuario de nuestro panel de trabajo.

Figura 65

Creación de Widgets



Posteriormente, seleccionamos el tipo de widget que necesitemos entre las opciones disponibles, su comportamiento (estático o dinámico) y su apariencia.

Para propósito de este trabajo de investigación, en primera instancia crearemos *widgets* que permitan conocer los valores de todas las variables en tiempo real, para este caso usaremos los *widgets*

"métrica", "medidor" y "termómetro", en segunda instancia crearemos *widgets* que sean capaces de mostrar un historial mensual que mostrará la evolución de las variables ECT, TPS, IAT y MAP con respecto al tiempo, para esto usaremos *widgets* "gráfico de líneas".

Figura 66

Selección del tipo de widget a crear

Figura 67

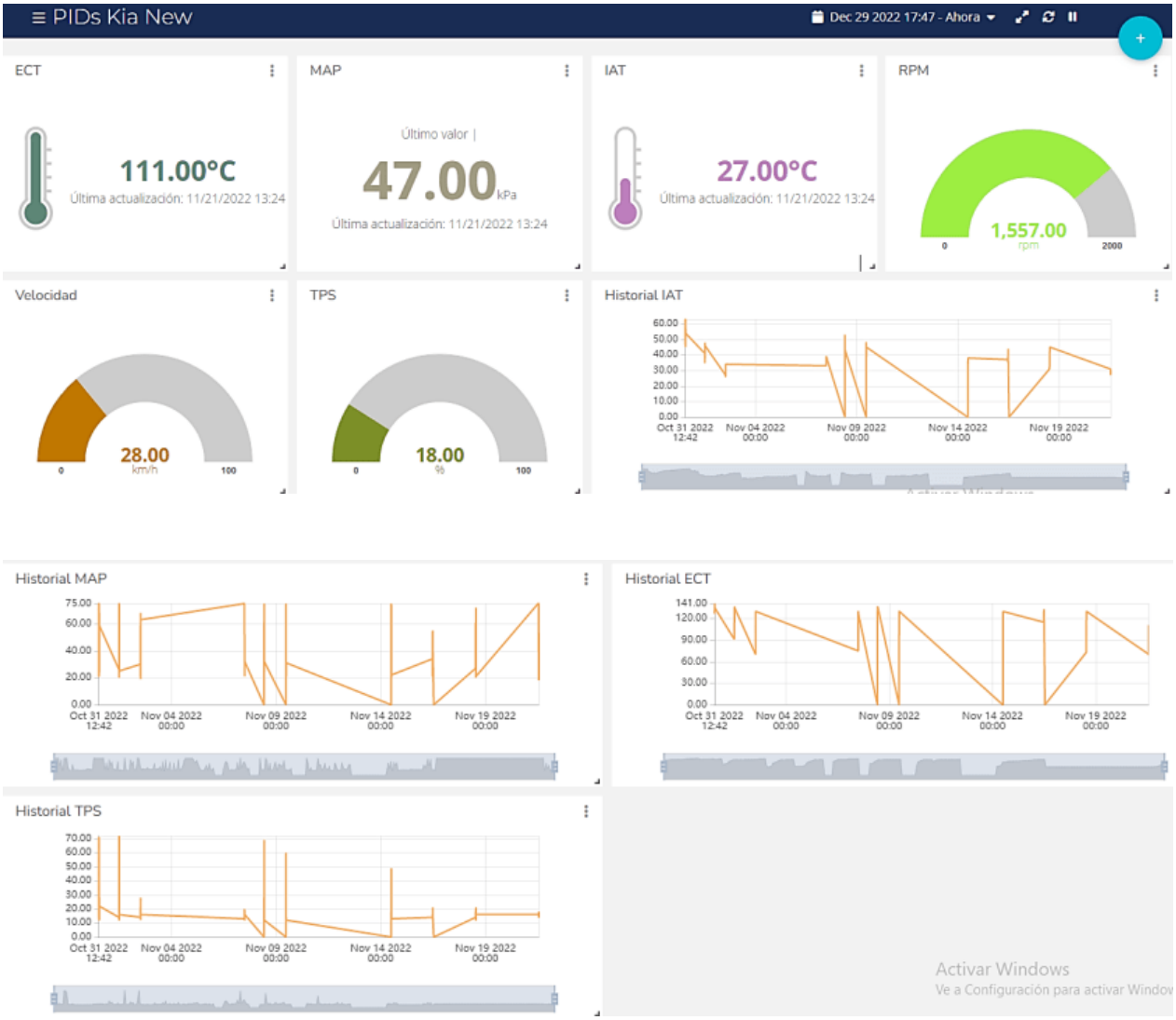
Selección de la variable asociada al nuevo widget

Para crear un widget también es necesario escoger el dispositivo y la variable de este que se desee representar, como se muestra en la figura anterior.

Una vez creados los *widgets*, estos sumarán automáticamente todos los valores dentro del intervalo de tiempo especificado en la barra de configuración del tablero, de manera que nuestro tablero CAN Bus estará dispuesto de la siguiente manera:

Figura 68

Dashboard perteneciente al nodo CAN Bus

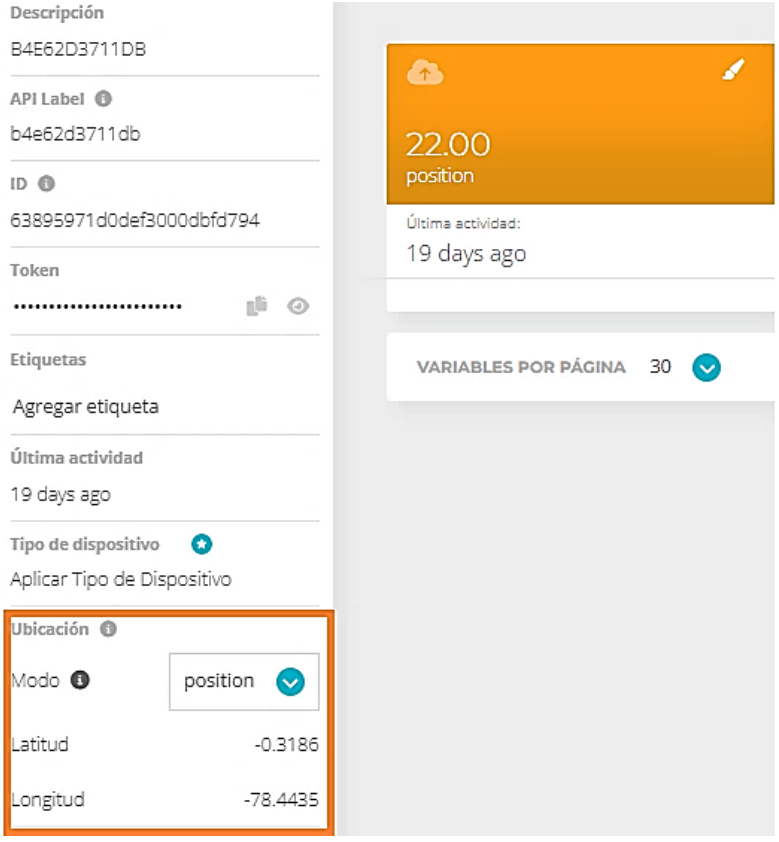


Nota. Dashboard compuesto de varios *widgets* usados para la representación de las variables del vehículo.

Para conocer la información de nuestro nodo GPS y representarla en un mapa crearemos un nuevo panel, ya que Ubidots solamente permite la creación de diez *widgets* por panel. Antes de crear un widget mapa, Ubidots necesita saber dónde encontrar los datos de ubicación del dispositivo encargado de proveer esta información. Cada dispositivo tiene una propiedad predefinida llamada “Ubicación”, que se encuentra en la parte final de la columna izquierda de la vista del dispositivo.

Figura 69

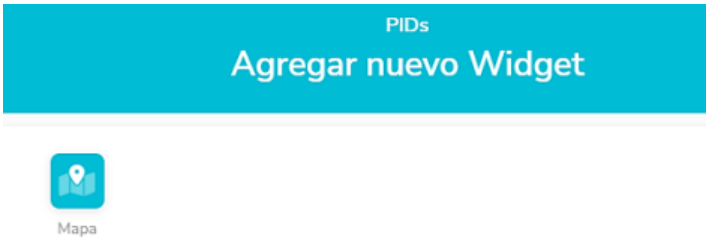
Parámetro de ubicación del dispositivo



En este aparatado especificaremos que la variable position tiene los valores “lat” y “lng” en su contexto. Es importante enviar los valores “lat” y “lng” al “contexto” de la variable de ubicación para permitir que Ubidots lea los datos de ubicación de nuestro dispositivo. Una vez realizado esto, creamos un nuevo widget del tipo mapa, para crearlo debemos seleccionar que dispositivo cuenta con la variable “position” y configuramos aspectos básicos de apariencia de nuestro mapa.

Figura 70

Creación del widget mapa



Una vez creamos nuestro wigdet podremos conocer la ubicación en tiempo real de nuestro dispositivo mediante un ícono azul y la ruta que ha marcado el vehículo al desplazarse.

Figura 71

Dashboard perteneciente al nodo GPS

Nota. La figura presenta un *Dashboard* que incluye un widget “mapa” que da seguimiento a la posición y movimiento del vehículo en tiempo real.

Aplicación móvil

Ubidots provee una conexión sencilla y segura para enviar y recuperar datos desde y hacia su servicio en la nube en tiempo real mediante “Ubidots Explorer”, una aplicación móvil desarrollada para sistema operativo Android que está diseñada para permitir a los usuarios existentes de Ubidots el manejo y visualización de toda la información procesable que se encuentre en su plataforma. Al abrir la aplicación, se debe ingresar usuario y contraseña para poder acceder a la ventana principal.

Ubidots Explorer permite al usuario administrar los dispositivos, variables y la aplicación en general ya ensamblada, para así conectar de manera más eficiente los datos del usuario con la arquitectura del sistema.

Figura 72

Ubidots Explorer en Google Play

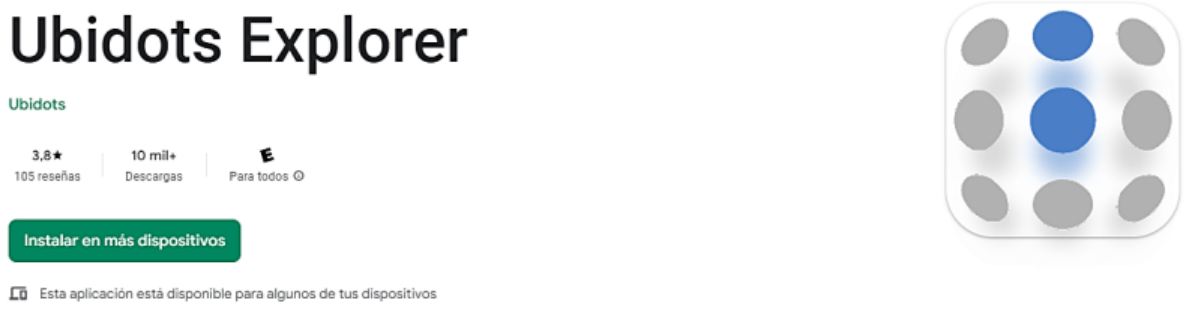


Figura 73

Login de Ubidots Explorer

Figura 74

Dispositivos, tableros y variables de Ubidots Explorer

Figura 75

Dashboards en Ubidots Explorer

Mantenimiento preventivo del Vehículo

El objetivo de un buen mantenimiento preventivo es prevenir cualquier tipo de inconveniente para que cada componente del vehículo siga funcionando de manera adecuada, su propósito es comprobar el correcto funcionamiento del vehículo y realizar los cambios de piezas desgastadas si se requiere.

Para realizar este mantenimiento, revisiones y ajustes, y además llevar un registro del número de fallas y errores es necesario conocer los valores máximos y mínimos de operación normal de las variables que tenemos a disposición.

Tabla 14

Valores máximos y mínimos permitidos de las variables

Variable	Valores mínimos y máximos	
ECT	-40° C mínimo y 130° C máximo	
IAT	-40° C mínimo y 170° C máximo	
MAP	0 kpa mínimo y 255 kpa máximo	
RPM	0 rpm mínimo y 5000 rpm máximo	
VEL	0 km/h mínimo y 120 km/h	Carretera
	0 km/h mínimo y 20 km/h	Campus Universitario
TPS	0 % mínimo a 100 % máximo	

Nota. Tomada de *Tecnicatura en Automotores*, por Cipolleti, C., 2018.

Al tener conocimiento de los errores y fallas que presenta el vehículo y para evitar daños se proyecta un plan de mantenimiento basado en el kilometraje para realizar los ajustes que se requieran y proveer de las garantías necesarias al propietario del automóvil.

Tabla 15

Plan de mantenimiento Kia Rio 2020

Elemento	Inspección	Reemplazo
Aceite de motor	-	10000 km
Filtro de aceite	-	10000 km
Aceite de transmisión	-	60000 km
Bujías	25000 km	75000 km
Filtro de aire	-	30000 km
Refrigerante de motor	10000 km	30000 km
Pastillas de freno	20000 km	60000 km
Forros de freno	20000 km	80000 km
Líquido de freno y embrague	-	30000 km
Neumáticos	5000 km	Según el uso
Batería	4 meses	4 años

Nota. Tomada de *Tecnicatura en Automotores*, por Cipolleti, C., 2018.

Eventos

En Ubidots, los eventos son mensajes activados y entregados a través de correos electrónicos o notificaciones propias de la aplicación móvil. Estas notificaciones se activan cuando las lecturas de un sensor alcanzan o superan un valor específico.

Para notificar al usuario creamos eventos de acuerdo a la tabla de valores máximos de las variables, estableciendo el rango de funcionamiento normal a cada variable. Para crear un evento accedemos a la pestaña “Eventos” y seleccionamos el dispositivo y la variable que deseamos controlar como se muestra en la siguiente figura.

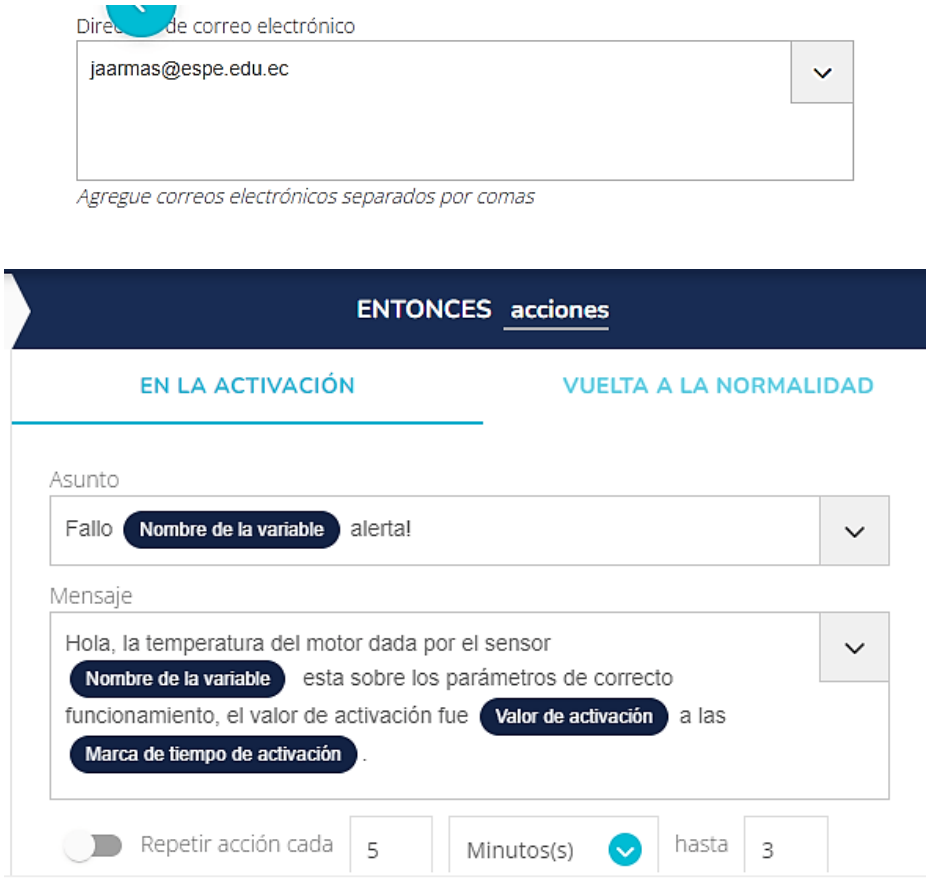
Figura 76

Creación de un evento y sus condiciones

Como ejemplo tomamos la creación de un evento para la variable “ECT” cuyo funcionamiento normal está dado por un rango donde la temperatura máxima es de 130 grados y la mínima es de -40 grados, después de establecer este rango se debe especificar la acción y el mensaje a enviar.

Figura 77

Creación del mensaje/notificación que recibirá el usuario al dispararse el evento

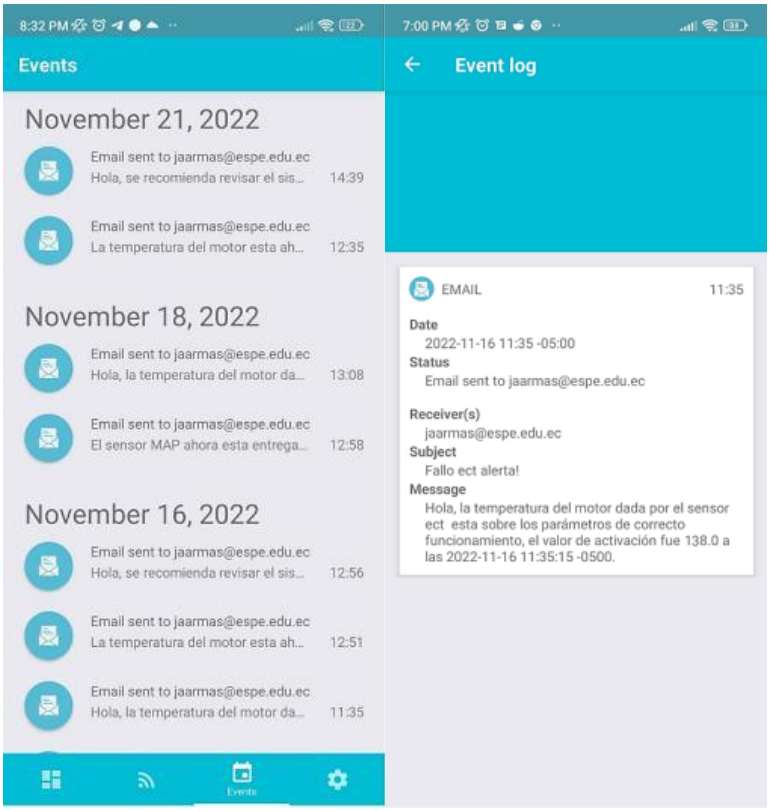


También se crearon eventos que notifiquen al usuario cuando el vehículo se está desplazando a exceso de velocidad, esto cuando la velocidad es mayor a 120 km/h en carretera y 20 km/h dentro del campus universitario de la ESPE.

Estas notificaciones serán visibles en la aplicación móvil, donde se llevara un registro de cada notificación, además serán enviadas al correo que fue especificado en el momento de su creación.

Figura 78

Notificaciones presentes en la app móvil "Ubidots Explorer"



Capítulo 5

Análisis de Resultados

Periodo de Recolección de datos

El periodo de pruebas y recolección de datos para el posterior análisis de los mismos fue realizado desde el 31 de Octubre al 21 de Noviembre de 2022, las pruebas de rutas fueron realizadas de lunes a viernes, ya que la Universidad de las Fuerzas Armadas "ESPE" no se encuentra abierta para estudiantes los días sábados y domingos.

Mensajes enviados y Calidad de enlace

Sigfox permite al usuario exportar los parámetros de rendimiento de los mensajes almacenados, además del número de mensajes enviados por día. Para poder realizar un análisis general de la calidad del enlace, Sigfox proporciona un indicador de calidad de enlace (LQI, del inglés *Link Quality Indicator*) basado en un indicador de intensidad de la señal recibida (RSSI, del inglés *Received Signal Strength Indicator*) y un indicador cualitativo de la calidad del enlace.

A continuación se muestra una tabla que detalla el número de mensajes enviados por día, la RSSI máxima y el indicador de calidad del enlace.

Tabla 16

Calidad de enlace y mensajes para el nodo CAN Bus

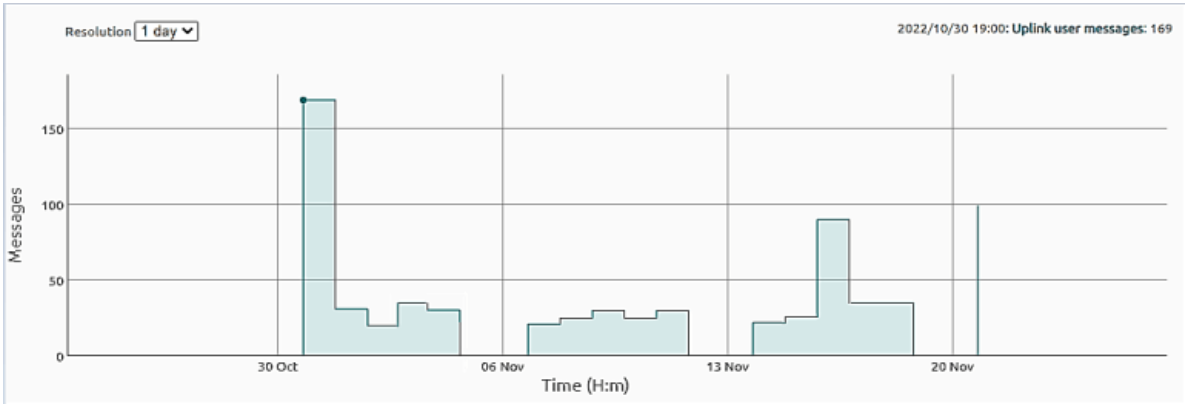
Fecha	Mensajes Recibidos	RSSI Máxima	Link Quality
31/10/2022	37	-56	Good
01/11/2022	31	-62	Good
02/11/2022	20	-60	Good
03/11/2022	33	-62	Good
04/11/2022	31	-59	Good
05/11/2022	0	n/a	Good
06/11/2022	0	n/a	Good
07/11/2022	21	-59	Good

Fecha	Mensajes Recibidos	RSSI Máxima	Link Quality
08/11/2022	25	-60	Good
09/11/2022	30	-61	Good
10/11/2022	26	-65	Good
11/11/2022	29	-57	Good
12/11/2022	0	n/a	Good
13/11/2022	0	n/a	Good
14/11/2022	22	-62	Good
15/11/2022	24	-60	Good
16/11/2022	90	-63	Good
17/11/2022	35	-62	Good
18/11/2022	35	-65	Good
19/11/2022	0	n/a	Good
20/11/2022	0	n/a	Good
21/11/2022	99	-56	Good

Nota. La columna RSSI comprende el valor máximo obtenido por día y la columna mensajes indica el total por día.

Figura 79

Mensajes enviados por el nodo CAN Bus al Backend de Sigfox



A continuación se detalla una muestra de los datos recolectados por el nodo CAN Bus correspondientes al 2 de Noviembre de 2022, se tomó los datos de esta prueba, al ser de corta duración.

Tabla 17

Valores correspondientes a los sensores del nodo CAN Bus

TPS	VEL	RPM	ECT	IAT	MAP	Fecha y Hora
16.0	0.0	503.0	130.0	34.0	63.0	02/11/2022 12:48:52
15.0	0.0	819.0	130.0	33.0	41.0	02/11/2022 12:48:29
12.0	7.0	607.0	130.0	33.0	28.0	02/11/2022 12:48:05
13.0	19.0	1217.0	129.0	32.0	23.0	02/11/2022 12:47:42
21.0	33.0	1445.0	130.0	29.0	68.0	02/11/2022 12:46:57
14.0	34.0	1455.0	130.0	30.0	27.0	02/11/2022 12:46:34
14.0	30.0	1311.0	128.0	30.0	21.0	02/11/2022 12:45:22
18.0	22.0	1237.0	127.0	30.0	54.0	02/11/2022 12:44:58
14.0	34.0	1457.0	128.0	29.0	21.0	02/11/2022 12:44:36
13.0	25.0	1115.0	129.0	28.0	21.0	02/11/2022 12:44:11
18.0	32.0	1352.0	123.0	27.0	57.0	02/11/2022 12:43:03
15.0	33.0	1396.0	121.0	27.0	29.0	02/11/2022 12:42:38
13.0	16.0	764.0	112.0	27.0	25.0	02/11/2022 12:40:41
28.0	33.0	1829.0	108.0	27.0	67.0	02/11/2022 12:40:17
14.0	36.0	1532.0	103.0	26.0	19.0	02/11/2022 12:39:30
21.0	30.0	1652.0	99.0	26.0	59.0	02/11/2022 12:39:07
14.0	15.0	1304.0	73.0	28.0	21.0	02/11/2022 12:36:46
15.0	11.0	916.0	73.0	28.0	39.0	02/11/2022 12:36:29
15.0	6.0	1026.0	71.0	28.0	28.0	02/11/2022 12:36:23
14.0	0.0	973.0	70.0	27.0	30.0	02/11/2022 12:36:17

Nota. La Tabla comprende los valores obtenidos para todos los sensores del sistema de inyección.

Las pruebas de ruta correspondientes al nodo GPS fueron realizadas con la misma duración y en simultáneo con las pruebas del nodo CAN Bus a partir del 7 de noviembre hasta el 21 de Noviembre de 2022. Se procuró que el tiempo de encendido y apagado de los dos nodos sea específicamente el mismo, para obtener el mismo número de mensajes en los dos nodos.

Tabla 18

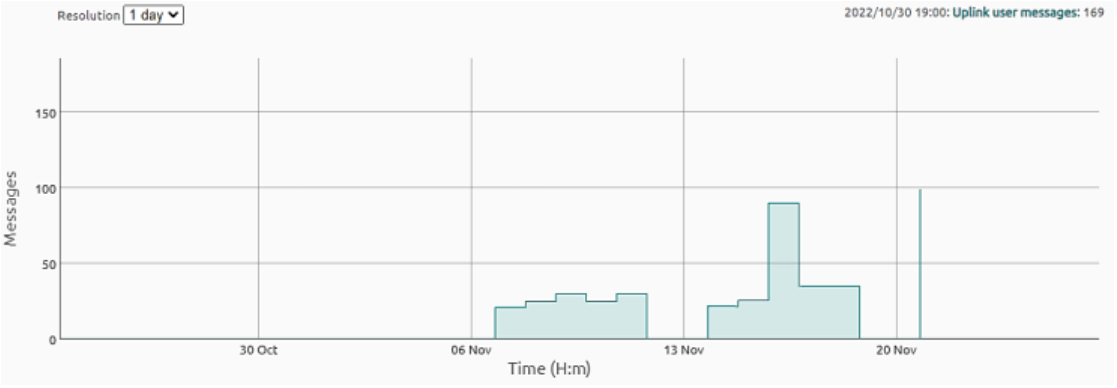
Calidad de enlace y mensajes para el nodo GPS

Fecha	Mensajes Recibidos	RSSI Máxima	Link Quality
07/11/2022	21	-66	Good
08/11/2022	25	-59	Good
09/11/2022	30	-61	Good
10/11/2022	26	-65	Good
11/11/2022	29	-59	Good
12/11/2022	0	n/a	Good
13/11/2022	0	n/a	Good
14/11/2022	22	-62	Good
15/11/2022	24	-59	Good
16/11/2022	90	-60	Good
17/11/2022	35	-62	Good
18/11/2022	35	-62	Good
19/11/2022	0	n/a	Good
20/11/2022	0	n/a	Good
21/11/2022	99	-59	Good

Nota. La columna RSSI comprende el valor máximo obtenido por día.

Figura 80

Mensajes enviados por el nodo GPS al Backend de Sigfox



Una muestra de los datos recolectados por el nodo GPS correspondientes al 14 de Noviembre de

2022. Es importante mencionar que el valor de la variable “Position” es simbólico e irrelevante, mientras que el contexto de esta variable es realmente la información que necesitamos.

Tabla 19

Valores correspondientes a la variable “Position”

Contexto de la variable Position	Fecha y Hora
{'lat': -0.333004, 'lng': -78.444964}	14/11/2022 12:17:11
{'lng': -78.444944, 'lat': -0.332804}	14/11/2022 12:16:48
{'lat': -0.332604, 'lng': -78.444924}	14/11/2022 12:16:02
{'lng': -78.444904, 'lat': -0.332404}	14/11/2022 12:15:40
{'lat': -0.332204, 'lng': -78.444884}	14/11/2022 12:14:43
{'lng': -78.444864, 'lat': -0.332004}	14/11/2022 12:14:18
{'lat': -0.331804, 'lng': -78.444844}	14/11/2022 12:13:56
{'lng': -78.444824, 'lat': -0.331604}	14/11/2022 12:12:22
{'lat': -0.331404, 'lng': -78.444804}	14/11/2022 12:11:34
{'lng': -78.444784, 'lat': -0.331204}	14/11/2022 12:09:38
{'lng': -78.444764, 'lat': -0.331004}	14/11/2022 12:08:45
{'lat': -0.330804, 'lng': -78.444744}	14/11/2022 12:05:39
{'lat': -0.330604, 'lng': -78.444724}	14/11/2022 12:04:52
{'lat': -0.330404, 'lng': -78.444704}	14/11/2022 12:04:29
{'lat': -0.330204, 'lng': -78.444684}	14/11/2022 12:04:05
{'lng': -78.444664, 'lat': -0.330004}	14/11/2022 12:03:41
{'lat': -0.329804, 'lng': -78.444644}	14/11/2022 12:03:24
{'lat': -0.329604, 'lng': -78.444624}	14/11/2022 12:03:18
{'lat': -0.329404, 'lng': -78.444604}	14/11/2022 12:03:12
{'lat': -0.329204, 'lng': -78.444584}	14/11/2022 11:42:53

Para determinar la calidad del enlace podemos guiarnos en la siguiente tabla que detalla los diferentes parámetros a tomar en cuenta para las zonas RC2 y RC4, dependiendo del número de estaciones base que recibieron los mensajes que cumplen con los umbrales RSSI.

Tabla 20

Calidad de enlace para RC2 y RC4

RSSI	Número de estaciones base	Indicador de calidad del enlace
-114 dBm < RSSI	3	Excelente
-127 dBm < RSSI ≤ -114 dBm	3	Bueno
-114 dBm < RSSI	1 o 2	Bueno
-127 dBm < RSSI ≤ -114 dBm	1 o 2	Promedio
RSSI ≤ -127 dBm	Ninguna	Límite

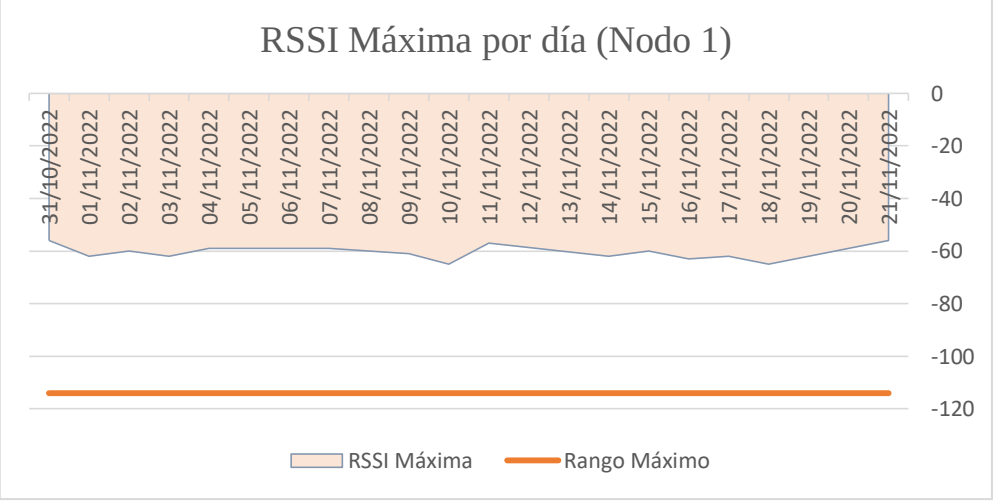
Nota. Tomado de *Sigfox Techinical Overiew,* por Sigfox, 2017, Sigfox (https://www.Sigfox.com/).

Debido a que las pruebas de ruta fueron realizadas en el anillo vial del campus de la ESPE, se utilizó una estación base para él envió de los mensajes, es decir para el Nodo 1 y 2 todos los mensajes fueron recibidos únicamente por una estación base.

En las figuras podemos observar los valores de RSSI máxima por día para cada nodo y comprobar si alguno sobrepaso el rango de -114dBm < RSSI, rango de RSSI para una única estación base receptora.

Figura 81

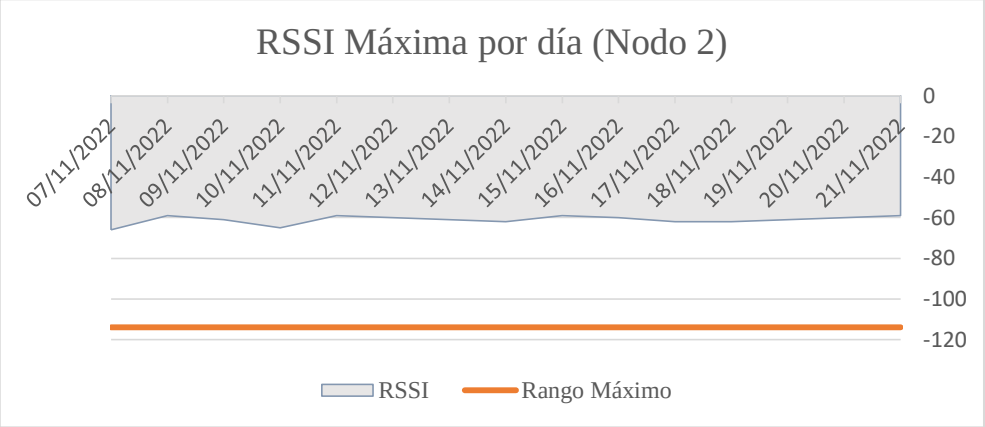
RSSI máxima por día para el Nodo 1



Nota. Se considera el RSSI máximo, el rango específico para la región y el número de estaciones.

Figura 82

RSSI máxima por día para el Nodo 2



Nota. Se considera el RSSI máximo, el rango específico para la región y el número de estaciones.

El ID de la Estación base Sigfox que recibió todos los mensajes es PICH 0014. No existe ningún mensaje recibido con un indicador de calidad del enlace que se encuentre en Limit (limite) o Average (promedio). Según lo indicado se concluye que la calidad del enlace para todos los mensajes recibidos por la estación base es BUENA ya que el RSSI máximo siempre se encuentra en el rango de -114dBm < RSSI, sin llegar a ser excelente debido a que solamente se trabajó con una estación base.

Figura 83

Radiobase PICH 0014 instalada en el Campus "ESPE Matriz"

Mensajes perdidos

Dado que las pruebas fueron realizadas en el Campus “ESPE” Matriz Sangolquí, la transmisión de los datos depende de la cobertura que pueda proveer la radio base instalada en el campus. Esta radio base dota de cobertura a la mayor parte del campus, sin embargo hay zonas donde esta cobertura se ve limitada, lo que ocasiona que existan brechas en la secuencia de mensajes y en consecuencia paquetes perdidos, esta cobertura se especifica en el siguiente mapa.

Figura 84

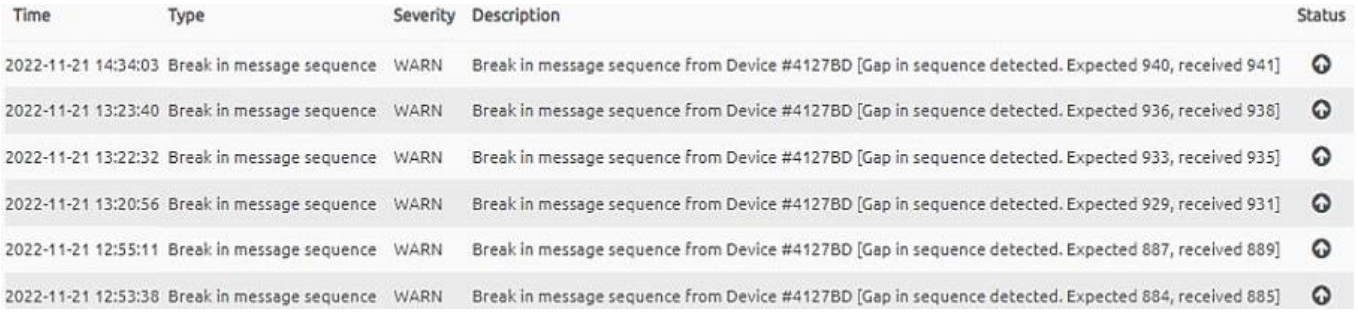
Cobertura en Campus ESPE

Sigfox cuenta con alertas predeterminadas que se envían cuando ocurren ciertos eventos, de manera que podemos llevar un conteo de los mensajes perdidos observando si existen interrupciones o brechas en la secuencia del mensaje.

Para determinar los mensajes perdidos por problemas de cobertura es necesario analizar y llevar un registro de los eventos y las alertas que han ocurrido en las pruebas de ruta.

Figura 85

Interrupciones en la secuencia del mensaje



Time	Type	Severity	Description	Status
2022-11-21 14:34:03	Break in message sequence	WARN	Break in message sequence from Device #4127BD [Gap in sequence detected. Expected 940, received 941]	
2022-11-21 13:23:40	Break in message sequence	WARN	Break in message sequence from Device #4127BD [Gap in sequence detected. Expected 936, received 938]	
2022-11-21 13:22:32	Break in message sequence	WARN	Break in message sequence from Device #4127BD [Gap in sequence detected. Expected 933, received 935]	
2022-11-21 13:20:56	Break in message sequence	WARN	Break in message sequence from Device #4127BD [Gap in sequence detected. Expected 929, received 931]	
2022-11-21 12:55:11	Break in message sequence	WARN	Break in message sequence from Device #4127BD [Gap in sequence detected. Expected 887, received 889]	
2022-11-21 12:53:38	Break in message sequence	WARN	Break in message sequence from Device #4127BD [Gap in sequence detected. Expected 884, received 885]	

Nota. Para conocer la totalidad de los mensajes perdidos se debe contabilizar las interrupciones en la secuencia del mensaje.

Los mensajes que llegaron exitosamente y los que se perdieron se muestran en las tablas:

Tabla 21

Mensajes recibidos y perdidos para el Nodo 1

Recibidos			**Perdidos**			**Total**		
Mensajes	Bytes	Porcentaje	Mensajes	Bytes	Porcentaje	Mensajes	Bytes	Porcentaje
588	7056	88.95 %	73	876	11.04%	661	7932	100 %

Nota. El porcentaje fue obtenido de acuerdo al total de mensajes enviados.

Tabla 22

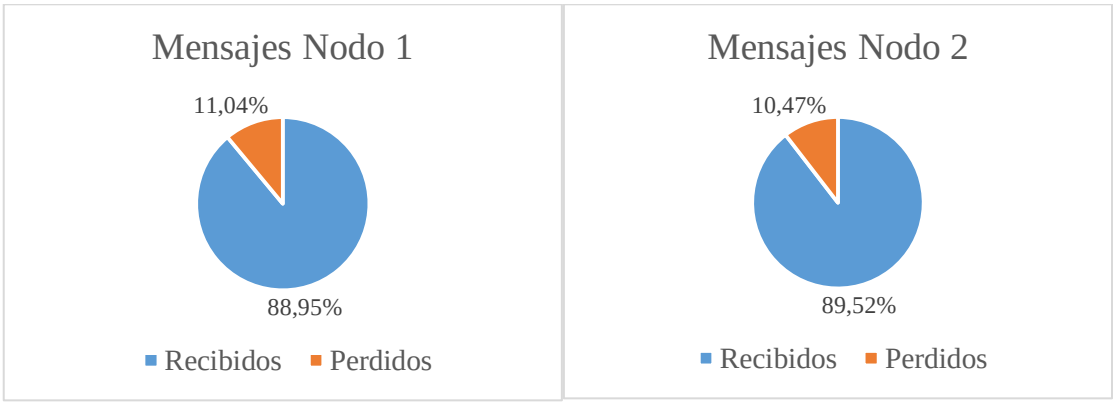
Mensajes recibidos y perdidos para el Nodo 2

Recibidos			**Perdidos**			**Total**		
Mensajes	Bytes	Porcentaje	Mensajes	Bytes	Porcentaje	Mensajes	Bytes	Porcentaje
436	5232	89.52 %	51	612	10.47%	487	584	100 %

Nota. El porcentaje fue obtenido de acuerdo al total de mensajes enviados.

Figura 86

Porcentajes de mensajes recibidos y perdidos para el nodo 1 y 2



Análisis de latencia de Callback

El retraso que se muestra en la ventana de estadísticas de Callback es el tiempo transcurrido entre que la estación base recibe el mensaje y se activa el Callback. Es importante llevar un registro de latencia ya que es posible observar un retraso alto en los mensajes del dispositivo si este está cubierto por una sola estación base y esta última tiene un problema de conectividad temporal.

Sigfox permite exportar esta información además de exponer una representación gráfica de los retardos de activación de Callback por día, como se observa a continuación.

Figura 87

Representación de la latencia de activación de Callback

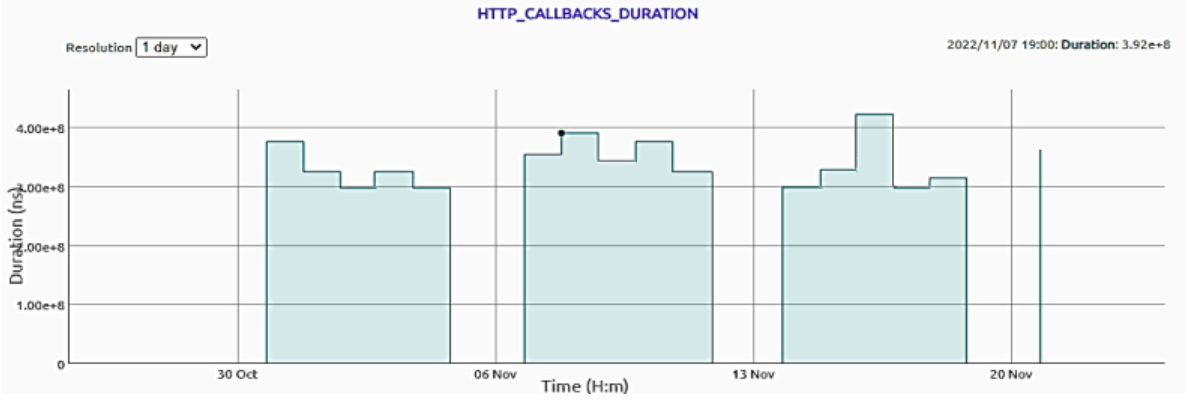


Tabla 23

Latencia para la activación de Callback por día

Fecha	Retardo Callback Nodo 1	Retardo Callback Nodo 2
31/10/2022	377 ms	n/a
01/11/2022	326 ms	n/a
02/11/2022	299 ms	n/a
03/11/2022	325 ms	n/a
04/11/2022	299 ms	n/a
05/11/2022	n/a	n/a
06/11/2022	n/a	n/a
07/11/2022	355 ms	671 ms
08/11/2022	392 ms	430 ms
09/11/2022	345 ms	353 ms
10/11/2022	374 ms	380 ms
11/11/2022	325 ms	389 ms
12/11/2022	n/a	n/a
13/11/2022	n/a	n/a
14/11/2022	300 ms	361 ms
15/11/2022	324 ms	502 ms
16/11/2022	424 ms	397 ms
17/11/2022	299 ms	438 ms
18/11/2022	316 ms	341 ms
19/11/2022	n/a	n/a
20/11/2022	n/a	n/a
21/11/2022	363 ms	384 ms
Promedio	340.1875 ms	422.3636 ms

Para estos resultados, el retardo de activación de Callback promedio es de apenas 340.18 ms para el Nodo 1 y de 422.36 ms para el Nodo 2, tiempos considerados en la activación de cada uno de los Callback.

Aunque los Callbacks de datos son casi instantáneos, algunos pueden activarse después de aproximadamente 30 segundos.

Análisis de latencia entre el Backend de Sigfox y Ubidots

La conexión entre el servidor de Ubidots y el Backend de Sigfox puede contener un retardo en la llegada de información, este retardo corresponde al tiempo en el que se establece la conexión con el servidor y se ejecutan las acciones de procesamiento y almacenamiento de los datos.

Para conocer esta información debemos comparar el tiempo de llegada de los paquetes al Backend de Sigfox con la marca temporal de cada mensaje presente en la base de datos de Ubidots y calcular el promedio de latencia de mensajes por día.

Para ejemplificar este proceso se tomará como muestra los mensajes adquiridos el 8 de Noviembre de 2022, se calculará su latencia promedio y se replicara el mismo proceso para todos mensajes de cada día en el que se realizó pruebas.

Tabla 24

Comparación de las marcas temporales del Backend de Sigfox y Ubidots

Fecha	Marca Temporal Backend	Marca Temporal Ubidots	Latencia (s)
08/11/2022	10:28:32	10:28:35	0:00:03
08/11/2022	10:28:09	10:28:11	0:00:02
08/11/2022	10:27:45	10:27:48	0:00:03
08/11/2022	10:26:59	10:27:01	0:00:02
08/11/2022	10:25:49	10:25:52	0:00:03
08/11/2022	10:25:03	10:25:05	0:00:02
08/11/2022	10:24:39	10:24:42	0:00:03
08/11/2022	10:24:33	10:24:35	0:00:02
08/11/2022	10:22:29	10:22:32	0:00:03
08/11/2022	10:20:55	10:20:57	0:00:02
08/11/2022	10:19:21	10:19:23	0:00:02

Fecha	Marca Temporal Backend	Marca Temporal Ubidots	Latencia (s)
08/11/2022	10:18:57	10:18:59	0:00:02
08/11/2022	10:17:47	10:17:52	0:00:05
08/11/2022	10:17:24	10:17:26	0:00:02
08/11/2022	10:16:37	10:16:39	0:00:02
08/11/2022	10:15:03	10:15:05	0:00:02
08/11/2022	10:13:06	10:13:09	0:00:03
08/11/2022	10:12:44	10:12:46	0:00:02
08/11/2022	10:12:20	10:12:22	0:00:02
08/11/2022	10:11:55	10:11:58	0:00:03
08/11/2022	10:11:31	10:11:35	0:00:04
08/11/2022	10:11:08	10:11:11	0:00:03
08/11/2022	10:10:50	10:10:53	0:00:03
08/11/2022	10:10:45	10:10:48	0:00:03
08/11/2022	10:10:39	10:10:42	0:00:03
	Promedio		0:00:03

Tabla 25

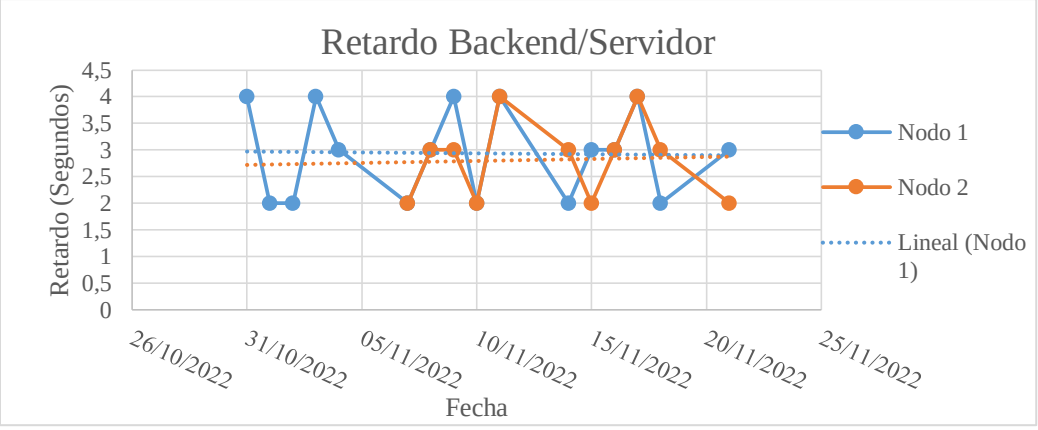
Latencia de recepción de mensajes del servidor

Fecha	Retardo Backend/Servidor Nodo 1	Retardo Backend/Servidor Nodo 2
31/10/2022	4 seg	n/a
01/11/2022	2 seg	n/a
02/11/2022	2 seg	n/a
03/11/2022	4 seg	n/a
04/11/2022	3 seg	n/a
05/11/2022	n/a	n/a
06/11/2022	n/a	n/a
07/11/2022	2 seg	2 seg
08/11/2022	3 seg	3 seg

Fecha	Retardo Backend/Servidor Nodo 1	Retardo Backend/Servidor Nodo 2
09/11/2022	4 seg	3 seg
10/11/2022	2 seg	2 seg
11/11/2022	4 seg	4 seg
12/11/2022	n/a	n/a
13/11/2022	n/a	n/a
14/11/2022	2 seg	3 seg
15/11/2022	3 seg	2 seg
16/11/2022	3 seg	3 seg
17/11/2022	4 seg	4 seg
18/11/2022	2 seg	3 seg
19/11/2022	n/a	n/a
20/11/2022	n/a	n/a
21/11/2022	3 seg	2 seg
Promedio	2,9375 seg	2,8181 seg

Figura 88

Comportamiento de la latencia promedio en la recepción de mensajes de Ubidots



Nota. La figura muestra latencia promedia por día para ambos nodos y su línea de tendencia.

El tiempo de retardo conocido como latencia puede crear ineficiencias, especialmente en las operaciones en tiempo real que dependen de datos propios de sensores. El promedio de latencia de

recepción de los mensajes por el servidor es de 2.9375 segundos para el nodo 1 y 2.8181 segundos para el nodo 2. Estos tiempos de respuesta son relativamente bajo y cumplen con los requerimientos del estudio de caso, por lo que no representa ningún inconveniente en el envío y recepción de datos.

Análisis de los datos de los Sensores

Los sensores de temperatura ECT e IAT trabajan en conjunto con otros sensores, como el TPS, MAP, MAF y otros más. Estos sensores proporcionan información idónea a la ECU del automóvil para llevar un buen funcionamiento y rendimiento.

Especificaciones de precisión de los Sensores de la ECU

Los automóviles modernos cuentan con un sistema electrónico que utiliza toda una red moderna de sensores de alta precisión que miden en todo momento condiciones de temperatura, presión y posición. Para comprobar que los resultados obtenidos mediante el prototipo son fidedignos se tomó como valores de referencia los obtenidos a partir de un escáner automotriz con el régimen de motor a 2000 rpm estables con un tiempo de encendido de 40 segundos, tiempo necesario para que el motor llegue a su temperatura normal de operación.

Tabla 26

Valores obtenidos con el escáner automotriz

Nombre	Valor	Unidad
Apertura de la mariposa	8,24	%
Presión del colector de admisión (MAP)	228,2	kPa
Temperatura de admisión de aire	68,25	˚ C
Temperatura del agua	99	˚ C
Velocidad del motor	2182	Rpm

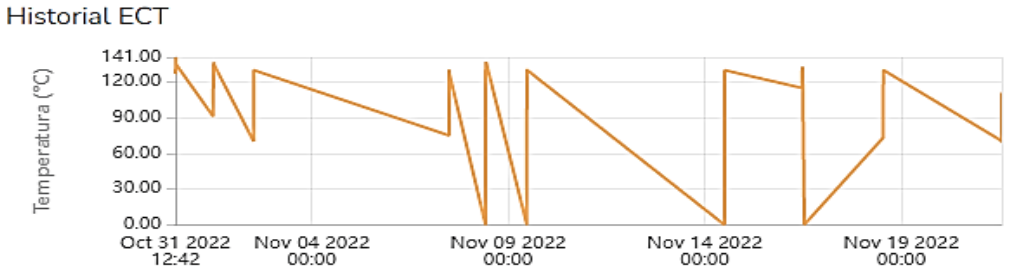
Nota. Los resultados obtenidos por el escáner automotriz fueron captados cuando el vehículo alcanzo las 2000 rpm y concuerdan con los valores que el prototipo devolvió.

Análisis ECT

Es importante conocer la información provista por el sensor ECT ya que la ECU gestiona la misma para controlar la inyección de la mejor manera posible y ajustar la mezcla aire /combustible. Para analizar esta información, los datos fueron compilados y comparados con respecto al rango de valores máximos y mínimos de funcionamiento, para así conocer su comportamiento y determinar si los valores de la variable se encuentran dentro de los límites de operación.

Figura 89

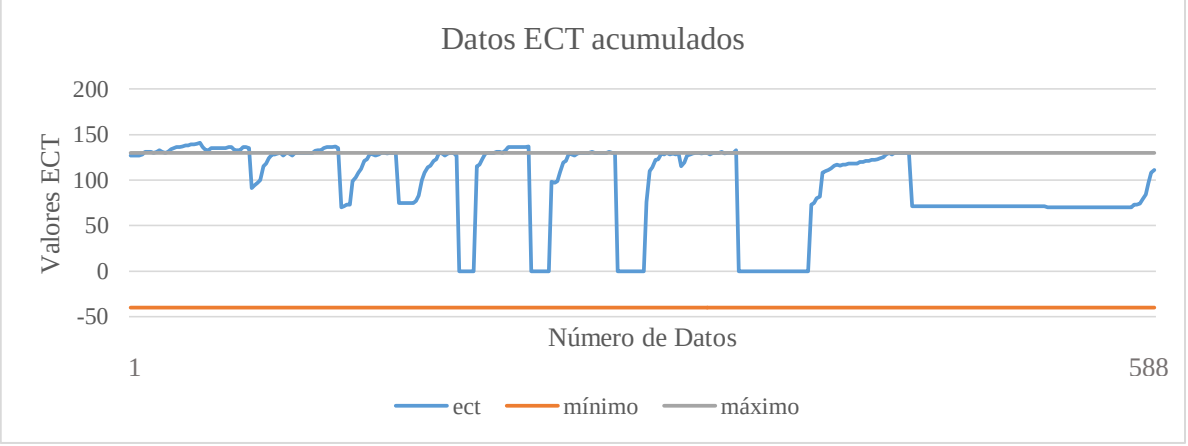
Representación de la variable ECT en Ubidots



Para conocer si se encuentran dentro de los límites determinados y cuantos valores están dentro o fuera de ellos, se realizó la siguiente representación visual de los datos, donde es posible distinguir si existen o no valores que exceden o superen los umbrales críticos.

Figura 90

Datos ECT Acumulados



De acuerdo a la figura anterior podemos concluir que existen valores que se encuentran fuera del rango establecido, por lo que para realizar un conteo de estos empleamos la siguiente fórmula en Excel, donde especificamos los límites superiores e inferiores.

=CONTAR.SI(B2:B589;">130")

=CONTAR.SI(B2:B589;"<-40")

Tabla 27

Análisis de porcentajes de la variable ECT

	Datos Totales	Valores ECT que superan el Umbral de 130° C y -40° C	Valores ECT que se mantienen en el rango de correcto funcionamiento
Número	588	119	469
Porcentaje	100 %	20.23 %	79.76 %

Nota. La Tabla muestra los umbrales máximos y mínimos de funcionamiento del sensor ECT y los porcentajes que cumplen o no con estos rangos.

El sensor ECT permite verificar que la temperatura en el circuito de refrigeración por agua no supere el umbral crítico, según las mediciones realizadas, este umbral fue superado en un 20.23 % de los datos adquiridos, lo que produce el encendido del ventilador situado delante del radiador cada vez que se supera este límite.

Al superarse el umbral crítico de funcionamiento algunos de los problemas que pueden presentarse son un consumo excesivo de combustible, niveles de CO_2 muy altos y problemas de sobrecalentamiento.

Análisis IAT

El propósito del sensor IAT es medir la temperatura y determinar la cantidad de masa de aire que entra en el colector de admisión. Considerando que la densidad del aire es inversamente proporcional a su temperatura.

Figura 91

Representación de la variable IAT en Ubidots

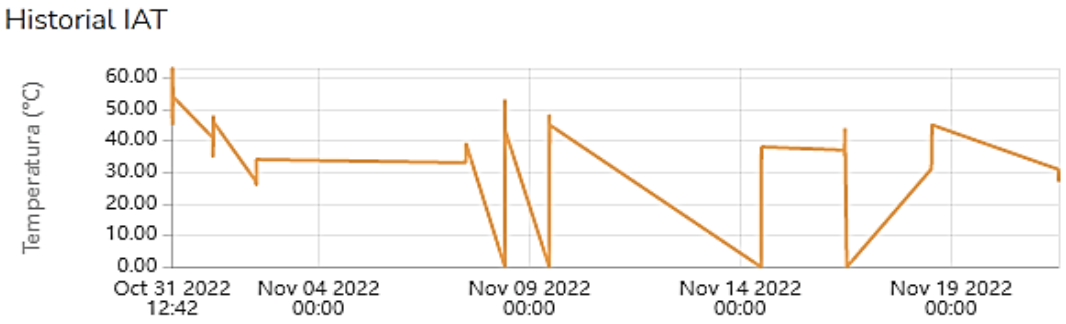


Figura 92

Datos IAT acumulados

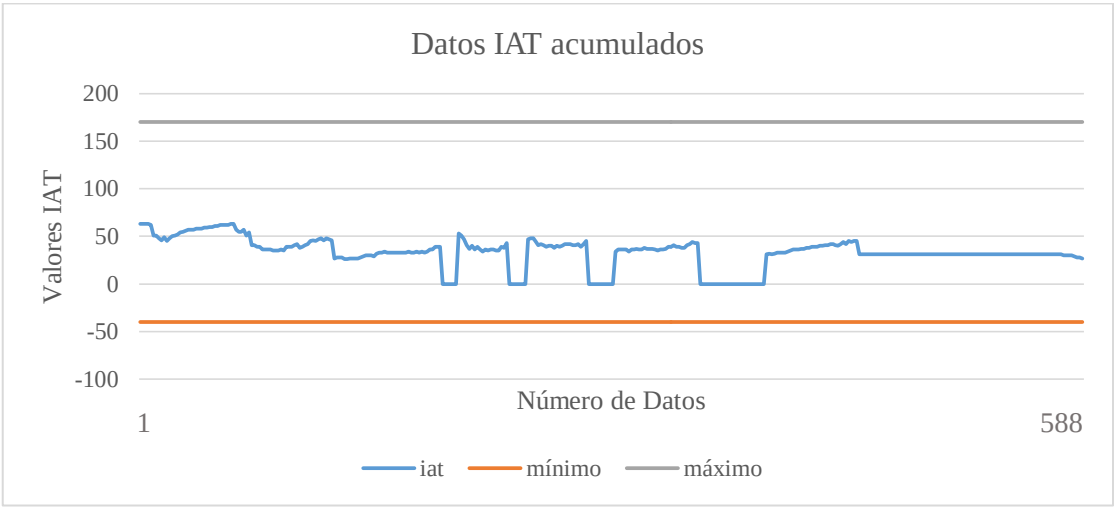


Tabla 28

Análisis de porcentajes de la variable IAT

	Datos Totales	**Valores IAT que superan el Umbral de 170̊ C y -40̊ C**	**Valores IAT que se mantienen en el rango de correcto funcionamiento**
Número	588	0	588
Porcentaje	100 %	0 %	100 %

De acuerdo a las mediciones realizadas, la temperatura siempre se mantiene dentro del rango

de correcto funcionamiento, por lo que no se espera que pueda producirse algún fallo, sin embargo, es importante llevar un registro de esta información ya que si el umbral se ve superado algunos de los fallos que pueden producirse son: Alto consumo de combustible, aumento en las emisiones de gases, dificultad para arrancar en frío y que la ECU sea incapaz de controlar el tiempo de encendido.

Análisis MAP

Para que el vehículo tenga una alta eficiencia de consumo, la mezcla oxígeno/combustible debe ser la adecuada. El objetivo del sensor MAP es controlar la cantidad de aire que podrá entrar para realizar la mezcla con el combustible.

Figura 93

Representación de la variable MAP en Ubidots

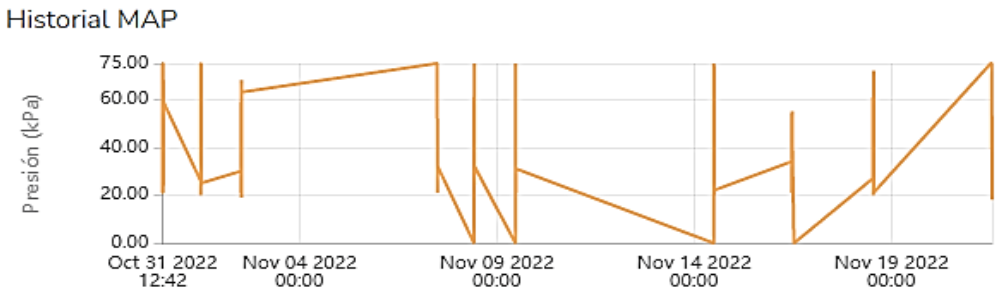


Figura 94

Datos MAP acumulados

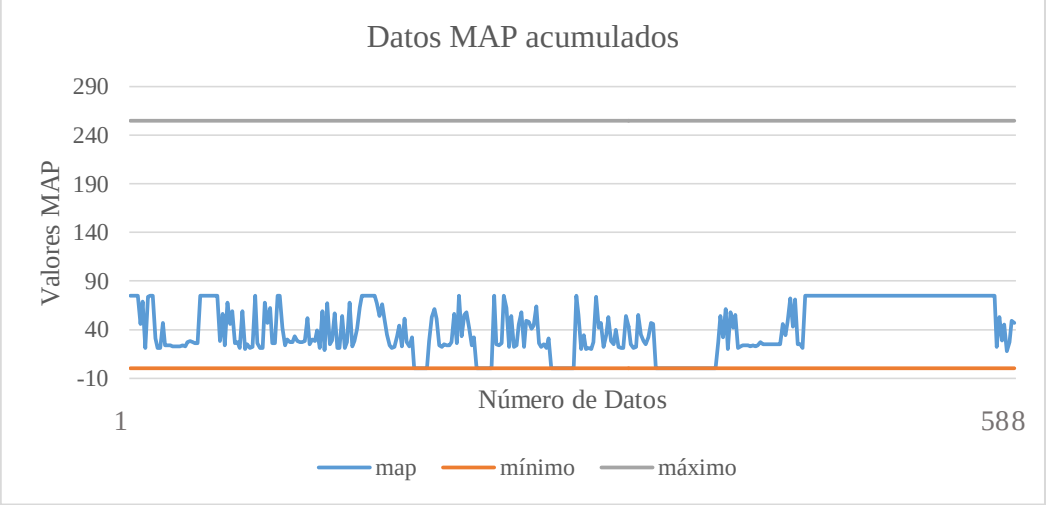


Tabla 29

Análisis de porcentajes de la variable MAP

	Datos Totales	Valores MAP que superan el Umbral de 255 kpa y 0 kpa	Valores MAP que se mantienen en el rango de correcto funcionamiento
Número	588	52	536
Porcentaje	100 %	8.84 %	91.15 %

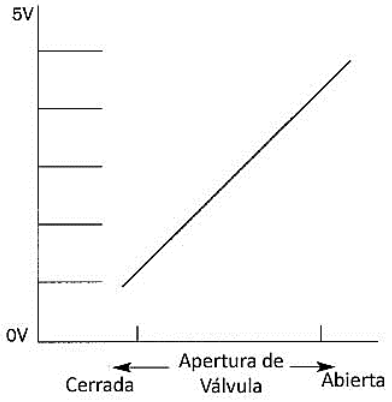
Según los datos recolectados, un 8.84 % de los valores están fuera de un rango de óptimo funcionamiento, de manera qué, algunos de los posibles fallos que podrían ocasionarse al existir un mal funcionamiento en esta área son fugas de vacío en el colector de admisión y daños por humedad en el sistema de admisión. Esto afecta otros componentes del auto, como el catalizador.

Análisis TPS

El sensor TPS está incorporado en el cuerpo de aceleración del vehículo y transforma el ángulo de la mariposa del cuerpo de aceleración en una señal eléctrica. En cuanto el papalote o mariposa se abre, el voltaje de la señal se incrementa.

Figura 95

Curva funcionamiento del sensor TPS



Nota. Comportamiento del sensor TPS de acuerdo a su curva característica. Tomado de *Manuales de computadoras y módulos automotrices,* por Orozco, M., 2020, Toyota Hiace.

La ECU usa la información entregada por el sensor TPS correspondiente a la posición de la mariposa de aceleración para conocer:

- Modo del motor: aceleración parcial, aceleración total y ralentí.
- Correcciones necesarias a realizar del incremento de potencia del motor.
- Correcciones necesarias a realizar de la proporción de ratio aire/combustible.

Figura 96

Representación de la variable TPS en Ubidots

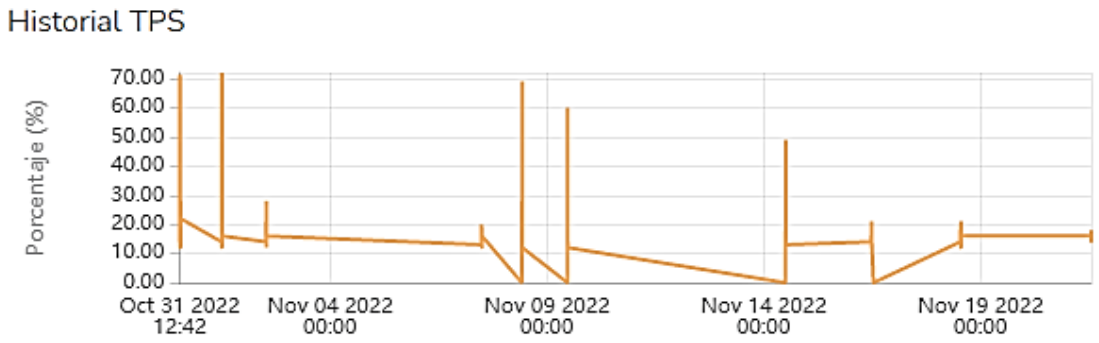


Figura 97

Datos TPS acumulados

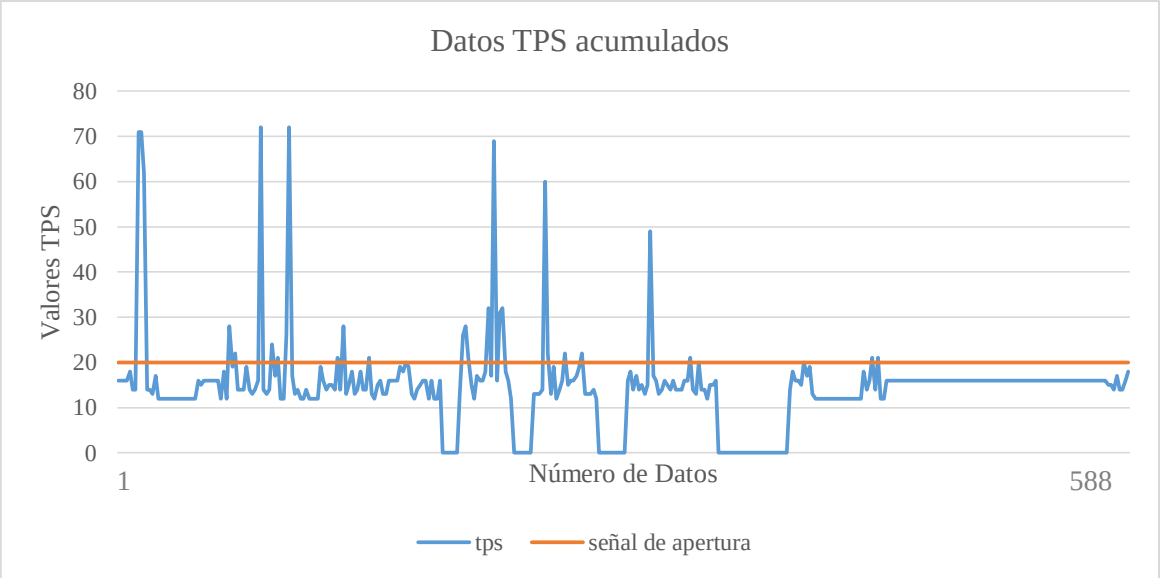


Tabla 30

Análisis de porcentajes de la variable TPS

	Datos Totales	Valores cuando el plato de la mariposa está cerrado (<20 %)	Valores cuando el plato de la mariposa está abierto (> 20 %)
Número	588	521	67
Porcentaje	100 %	88.60 %	11.39 %

En el modo del motor “ralentí”, el voltaje de la señal del sensor es entre 0.6 y 0.9 Volts, lo que equivale a el rango de 0 a 20 % de variación. En este voltaje, la ECU reconoce que el plato de la mariposa está cerrado. A partir de 1 Volt o valores mayores a 20 % la ECU reconoce que el plato de la mariposa está abierto y que el modo del motor en el que el vehículo está trabajando es aceleración parcial o total.

Es importante analizar la información provista por este sensor para evitar problemas relacionados con el motor como una marcha ralentí inestable, motor muy acelerado y falta de potencia en el motor.

Análisis Velocidad

El sensor de velocidad cumple la función de informar la velocidad que desarrolla un vehículo, y envía esta señal a la ECU. La ECU utiliza esta señal para determinar el momento exacto en el que la transmisión automática debe cambiar la marcha, además de calcular el tiempo de encendido y el tiempo de inyección del combustible.

Figura 98

Representación de la variable velocidad en Ubidots

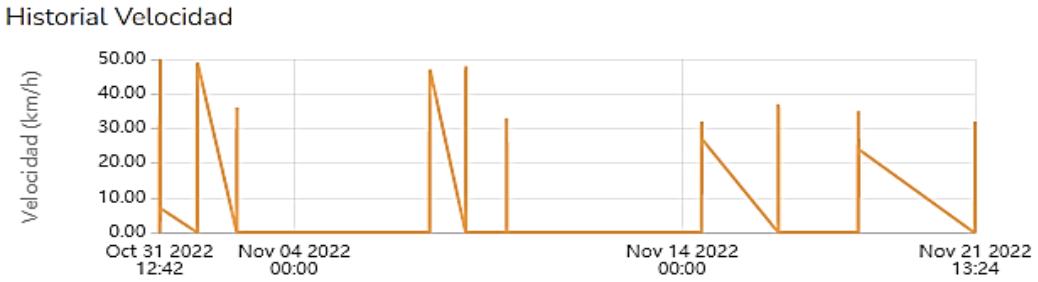


Figura 99

Datos Velocidad acumulados

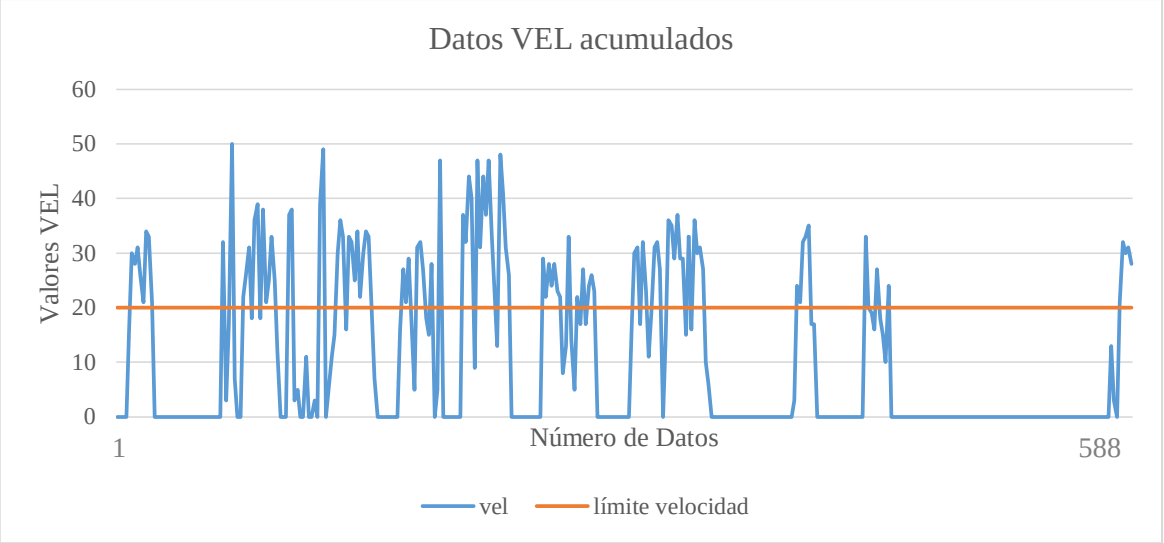


Tabla 31

Análisis de porcentajes de la variable Velocidad

	Datos Totales	Valores cuando el vehículo no supero el límite de velocidad	Valores cuando el vehículo excedió el límite de velocidad
Número	588	167	421
Porcentaje	100 %	28.40 %	71.60 %

Nota. El análisis de la tabla está relacionado a los parámetros de los límites de velocidad establecidos en el Campus Universitario “ESPE”.

Para analizar los datos provistos por este sensor se consideró que las pruebas de ruta fueron realizadas en el Campus Universitario “ESPE” donde el límite de velocidad para los vehículos es de 20 km/h, por lo que el sistema informa cada vez que el usuario supera este límite, siendo superado en un 71.60 % de las veces que se realizaron las pruebas, para un 28.40 % restante donde se respetó el límite de velocidad.

Análisis RPM

Un sensor RPM determina las revoluciones a las que gira el árbol de levas y el cigüeñal del motor, y envía esta información a la ECU para que determine el caudal de carburante que debe ser inyectado en el momento apropiado. Además, esta señal permite a la unidad de motor ajustar la presión de soplado del turbo y la recirculación de los gases del tubo de escape.

Figura 100

Representación de la variable RPM en Ubidots

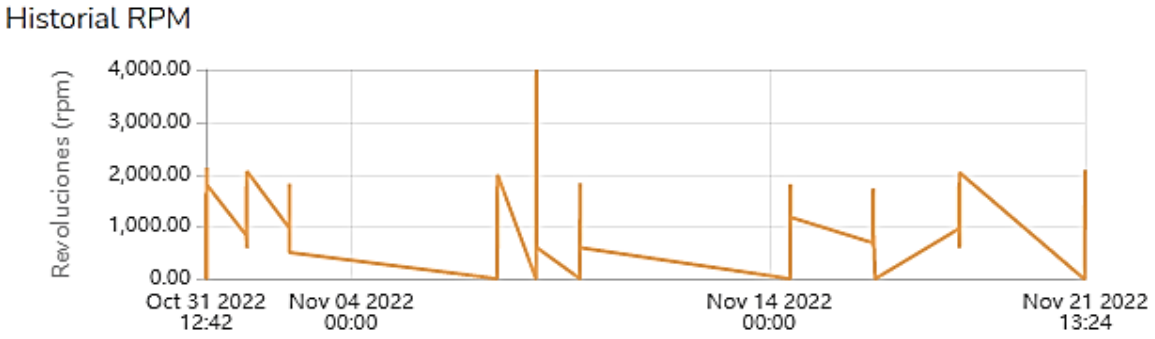


Figura 101

Datos RPM acumulados

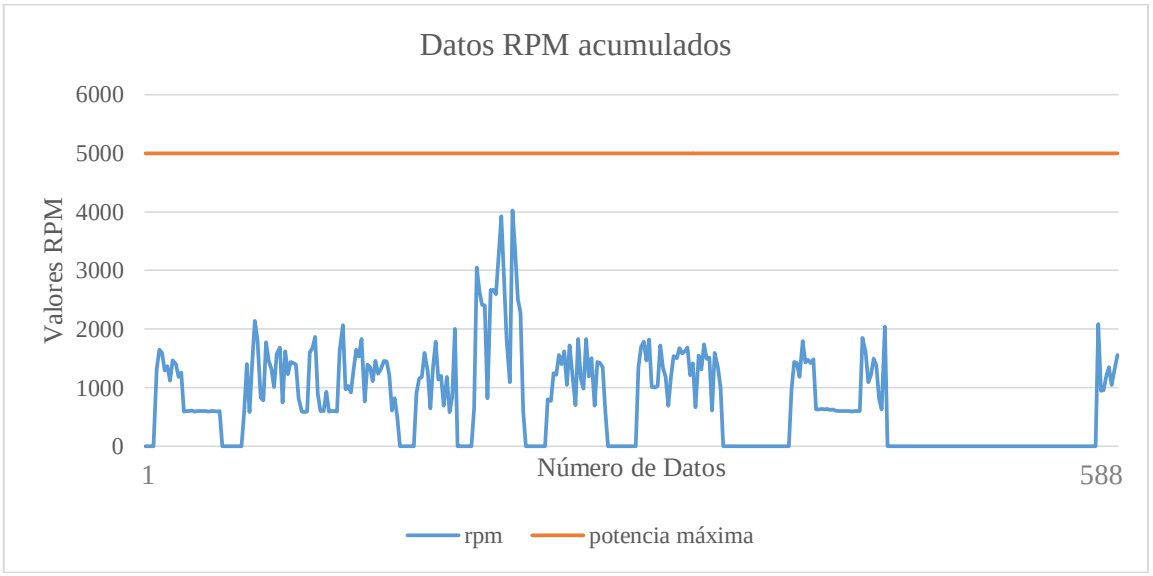


Tabla 32

Análisis de porcentajes de la variable RPM

	Datos Totales	Valores cuando el vehículo no desarrollo la potencia máxima	Valores cuando el vehículo alcanzo la potencia máxima
Número	588	588	0
Porcentaje	100 %	100 %	0 %

A partir de los datos recogidos, se determina si el vehículo logro alcanzar o no una potencia máxima de uso del motor. Esta potencia máxima se alcanza a diferentes velocidades del cigüeñal. La potencia máxima para vehículos que funcionan con gasolina generalmente se logra a 5 mil rpm, por lo que para las pruebas de ruta realizadas, no fue posible alcanzar dicha potencia.

En caso de que un vehículo presente un sensor de revoluciones en mal estado, el vehículo puede verse afectado con una pérdida de potencia en el motor y con un desgaste por el colapso de los pistones, las válvulas y el cigüeñal.

Análisis de la trayectoria recorrida por el vehículo

Además de estimar la ubicación en tiempo real del vehículo (latitud y longitud), es posible determinar el movimiento de pista de un objeto. Una vez que se ha calculado la posición mediante sus coordenadas, a partir de esta información es posible estimar otro tipo de información, como la trayectoria recorrida y la distancia hasta el destino de la prueba de ruta.

Para trazar la trayectoria recorrida por el vehículo es necesario importar una hoja de cálculo que contenga las coordenadas de latitud y longitud a Google Earth mediante un archivo .kml, para graficar la pista GPS es necesario especificar otros elementos propios de esta como: el color de la línea que unirá las coordenadas, la numeración del ícono a usar, su color y su tamaño.

Tabla 33

Hoja de cálculo para la creación del archivo .kml

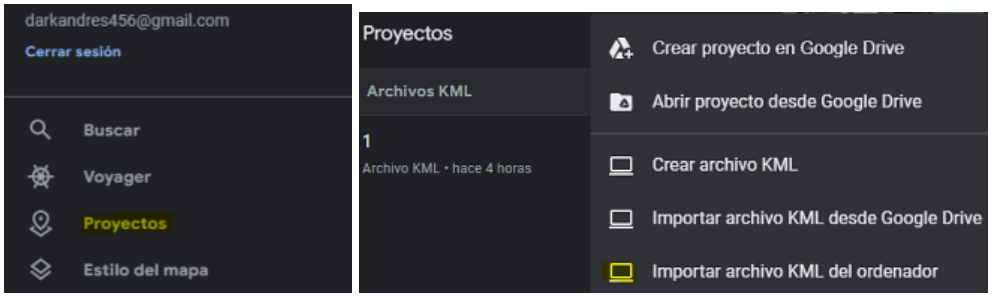
Latitude	Longitude	LineStringColor	Icon	IconColor	IconHeading	IconScale
-0,316751491	-78,44383617	cian	186	amarillo	line -180	0,1
-0,316590561	-78,44378253	cian	186	amarillo	line -180	0,1
-0,31571081	-78,44339629	cian	186	amarillo	line -180	0,1
-0,314198067	-78,44310661	cian	186	amarillo	line -180	0,1
-0,313586533	-78,44323536	cian	186	amarillo	line -180	0,1
-0,313211029	-78,44363232	cian	186	amarillo	line -180	0,1
-0,312856983	-78,44411512	cian	186	amarillo	line -180	0,1
-0,312352735	-78,4448125	cian	186	amarillo	line -180	0,1
-0,312095247	-78,44516655	cian	186	amarillo	line -180	0,1
-0,311859216	-78,44571372	cian	186	amarillo	line -180	0,1
-0,311880673	-78,44625016	cian	186	amarillo	line -180	0,1
-0,312245448	-78,44666859	cian	186	amarillo	line -180	0,1
-0,312921355	-78,44704409	cian	186	amarillo	line -180	0,1
-0,312921355	-78,44704409	cian	186	amarillo	line -180	0,1
-0,313071556	-78,44707628	cian	186	amarillo	line -180	0,1
-0,313296858	-78,44715138	cian	186	amarillo	line -180	0,1
-0,314273168	-78,44701191	cian	186	amarillo	line -180	0,1
-0,314852516	-78,44664713	cian	186	amarillo	line -180	0,1
-0,315474779	-78,44627162	cian	186	amarillo	line -180	0,1
-0,316612019	-78,44631453	cian	186	amarillo	line -180	0,1

Para transformar el archivo xls o csv correspondiente a la hoja de cálculo que comprende las coordenadas a un archivo .kml existen varios sitios que pueden ayudarnos, como es el caso de “Earth Point”, para esto ingresaremos al siguiente enlace: https://www.earthpoint.us/exceltokml.aspx.

Una vez hemos generado el archivo .kml debemos crear un proyecto en Google Earth, donde podremos importar el archivo .kml creado desde nuestro ordenador, así:

Figura 102

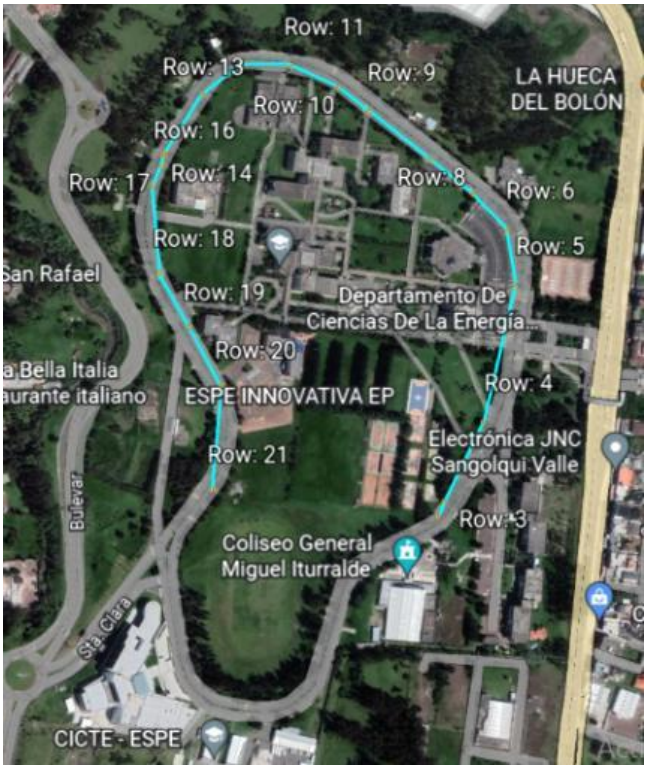
Creación de un nuevo proyecto en Google Earth



Al cargar este archivo inmediatamente se trazara la ruta que recorrió el vehículo y los puntos de coordenadas presentes en la hoja de cálculo.

Figura 103

Ruta trazada por Google Earth



Al analizar la pista trazada por Google Earth podemos comprobar que el trazo de esta depende de la cobertura que pueda proveer la radio base, ya que en locaciones donde disminuye la cobertura se pierden paquetes de información que contienen las coordenadas de esa ubicación, y la ruta no puede

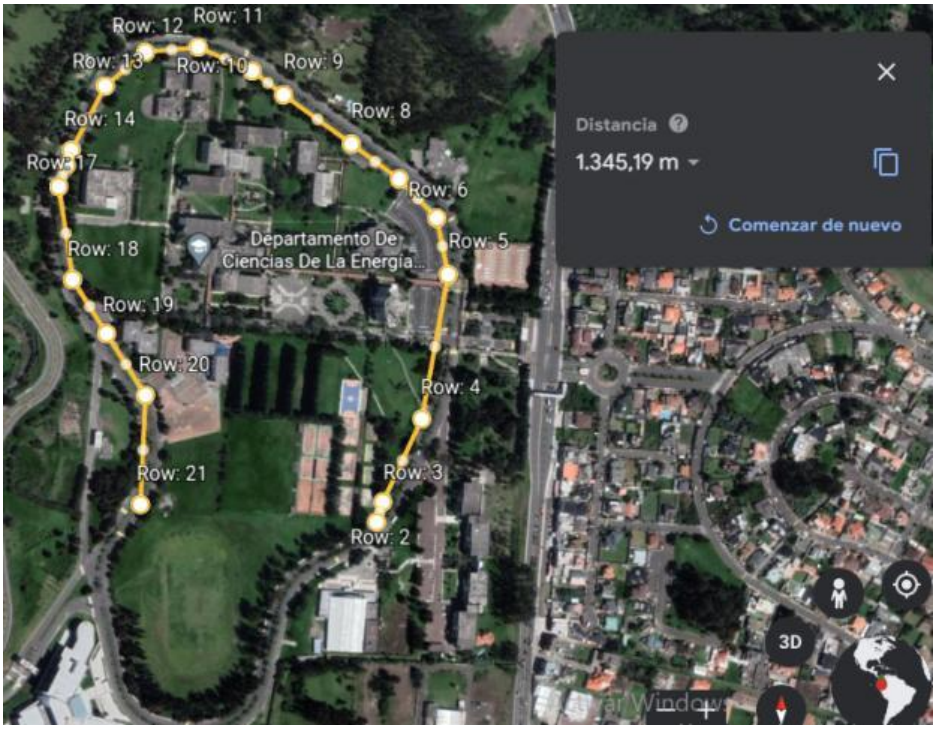
trazarse al 100 %, es así que para zonas donde no existe una buena cobertura como se observa en la parte inferior del mapa, no será posible obtener la información correspondiente a la ubicación y ruta recorrida.

Para un buen trazado es necesario considerar también otros parámetros importantes como la exactitud y nivel de recepción del módulo GPS Neo-6M y que exista una buena cantidad de paquetes de coordenadas en distancias cortas, para realizar así una representación fidedigna de la trayectoria recorrida en las pruebas de ruta.

Google Earth también permite realizar mediciones de la distancia recorrida a partir del punto de salida hasta punto de llegada, para este caso, la prueba de ruta realizada tuvo una distancia recorrida por el vehículo de 1345.19 m, esta información es valiosa ya que será relevante para cumplir con el plan de mantenimiento del vehículo de acuerdo al kilometraje.

Figura 104

Mediciones realizadas de la distancia recorrida



Es necesario definir un plan de mantenimiento del sistema de inyección en base al kilometraje y estado de los componentes para que un vehículo pueda tener una vida útil duradera, al no obtener los datos correspondientes al PID odómetro se optó por usar los datos provistos por el GPS (coordenadas, trayectoria y distancia recorrida) como alternativa para registrar la distancia total o parcial recorrida por el vehículo.

Costos del Sistema

La implementación del proyecto tuvo gastos relacionados con la instalación y despliegue del prototipo y otros relacionados con la operación y pruebas de ruta realizadas, es importante señalar que al usar plataformas gratuitas de servicios en la nube no hubo gastos en este apartado.

Tabla 34

Materiales usados en el proyecto

Orden	Detalle	Cantidad	Valor Unitario USD	Valor Total USD
1	Arduino UNO	1	27	54
2	Arduino Mega	1	48,40	48,40
3	Módulo Thinxtra	2	36,25	72,50
4	Shield CAN Bus	1	34	34
5	Cable OBD II	1	12	12
6	Módulo GPS Neo-6M	1	13	13
7	Bateria 12 V	1	3	3
8	Cinta Autofundente	1	8	8
	Total			244,9

Tabla 35

Gastos en pruebas de ruta

Orden	Detalle	Cantidad	Valor Unitario USD	Valor Total USD
1	Combustible	33 galones	2,40	79,2
	Total			79,2

Tabla 36

Costo total proyecto

Orden	Detalle	Valor Total USD
1	Materiales	244,9
2	Pruebas	79,2
	Total	324,1

Trabajos Futuros

Como continuación a este proyecto de investigación, existen diversas líneas de investigación que quedan abiertas y en las que es posible continuar trabajando. A continuación se presentan algunos trabajos futuros que pueden desarrollarse como resultado de esta investigación:

- Implementar servicios de seguridad y comunicación, relacionados con direccionamiento vial, localización y recuperación del vehículo en caso de robo.
- Desarrollar un sistema de atención al cliente las 24 horas del día que proporcione al usuario asistencia médica o mecánica cuando este lo requiera.
- Crear comandos de apertura y cierre remoto de puertas en caso de olvido de llaves o control parental.

Conclusiones

- Se diseñó e implementó una interfaz de comunicaciones con tecnología LPWAN Sigfox que permite la monitorización y análisis de los parámetros básicos de funcionamiento de un vehículo Kia Rio 2020 mediante una plataforma IoT que gestiona los datos obtenidos.
- Se diseñó una aplicación web en la plataforma Ubidots, que incorpora una aplicación para dispositivos móviles Android, para que el usuario pueda realizar un diagnóstico automotriz de su vehículo y pueda recibir alertas de fallas del funcionamiento del vehículo.
- Al integrar la tecnología LPWAN Sigfox con el sistema de diagnóstico a bordo se creó un sistema

de monitoreo y gestión que permite al usuario verificar las variables y su estado en tiempo real.

- Se desarrolló una base de datos almacenada en Ubidots que guarda toda la información reportada por el sistema de diagnóstico OBD II, la unidad de control del motor y los sensores que la conforman (TPS, MAP, VEL, IAT, RPM y ECT).
- Se presentó la información obtenida por la interfaz OBD en un *dashboard* que permite al usuario visualizar el estado de las variables, además de permitir seleccionar los datos que se desee analizar de acuerdo al periodo de tiempo y variable de interés.
- Se analizó el desempeño del sistema de diagnóstico a partir de los resultados obtenidos para cada variable, verificando los valores de acuerdo a los rangos de rendimiento y funcionamiento óptimo.
- Respecto al nivel de RSSI de la red se verificó que los niveles de señal de la red Sigfox en todo momento fueron superiores a -114 dBm, considerando que solamente se usó una estación base. Sin embargo, se evidenció la perdida de paquetes debido a la disminución de cobertura.
- Con el análisis de los valores obtenidos de los sensores que conforman la ECU y el sistema de inyección del vehículo en base a sus umbrales críticos se pudo definir el procedimiento para realizar un cambió o inspección de los mismos.

Recomendaciones

- Para realizar una comparación entre el funcionamiento de distintas plataformas que permitan desarrollar aplicaciones IoT, es recomendable hacer una validación de la herramienta Ubidots con otras plataformas como TTN networks, PowerBI.
- Analizar el sistema electrónico de vehículos fabricados en años anteriores, con el objetivo de verificar la compatibilidad o la realización de modificaciones en el prototipo actual de manera que pueda funcionar en dichos automotores.
- Se debe ubicar los nodos en el vehículo de manera que la antena del módulo Thinxtra tenga

línea de vista y el módulo GPS tenga buena cobertura, para lo cual lo mejor es situarlos sobre el tablero cerca del parabrisas.

- Para el despliegue de otros proyectos similares se recomienda el uso de otras tecnologías LPWAN, 3G y 4G, para comparar el desempeño con la red de comunicaciones utilizada en este trabajo.

Bibliografía

Atmel. (2015). *Inspiring Smart and Secure Connected Designs.*

Barrio, M. (2018). *Internet de las cosas.* Editorial Reus.

Buyya, R., & Vahid, A. (2016). *Principios y paradigmas de Internet de las Cosas.* Editorial Morgan Kaufmann.

Caizatoa, M., & Méndez, X. (2014). *Diseño e implementación de un prototipo de monitoreo de automóviles empleando el estándar OBD-II.* [Tesis de Pregrado].Universidad de las Fuerzas Armadas ESPE. doi:http://repositorio.espe.edu.ec/handle/21000/8517

Cipolletti, C. (2018). *Tecnicatura en Automotores.*

CssElectronics. (2022). *OBD2 Explicado: Una Introducción Simple*. Obtenido de https://www.csselectronics.com/pages/obd2-explained-simple-intro

Encinas, D., Meilan, P., Bava, A., & Naiouf, M. (2009). *Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores.* Vehículos Espaciales de Nueva Generación S. A. (VENG S. A.). Obtenido de http://sedici.unlp.edu.ar/handle/10915/21218

Espinosa, B., & Orellana, M. (2021). *Desarrollo de aplicaciones de monitoreo y control basadas en* IoT *a través de la plataforma Ubidots. Aplicaciones a sistemas de automatización bajo entornos de simulación.* [Tesis de Maestría]. Universidad Politécnica Salesiana sede Cuenca. doi:http://dspace.ups.edu.ec/handle/123456789/20298

García, H. (2020). *Diagnóstico de fallas a través del CAN BUS.* Podibooks.

Hart, M. (2013). *TinyGPS Library. Electronic Projects Components Available Worldwide*. Obtenido de https://www.pjrc.com/teensy/td_libs_TinyGPS.html

ISO. (2016). *ISO Car informatics. On board computer systems. Including navigation systems, car radio, etc.*

Jhou, J., & Chen, S. (2013). *The Implementation of OBD-II Vehicle Diagnosis System Integrated with Cloud*

Computation Technology. Second International Conference on Robot, Vision and Signal Processing. doi:https://doi.org/10.1109/RVSP.2013.55

Ji, S. (2019). *WISOL/WSSFM10R4AT Datasheet.*

Kia. (2020). *Student Learning Guide and Workbook.*

M2Communication. (2018). *Sigfox 0G Technology.*

Martínez, A. (2017). *Introducción a CAN bus: Descripción, ejemplos y aplicaciones de tiempo.* [Tesis de Pregrado]. E.T.S.I. de Sistemas Informáticos (UPM). doi:https://oa.upm.es/48061/

Martres, P. (2019). *Particularidades de la red Sigfox.* [Tesis de Pregrado]. UDE Universidad de la Empresa. doi:https://ude.edu.uy/particularidades-de-la-red-Sigfox/

Morales, E., & Altamirano, F. (2016). *Computación en la Nube con Google Drive.* [Tesis de Pregrado]. ESPOCH.

Orozco, M. (2020). *Manuales de computadoras y módulos automotrices.* Toyota Hiace.

Paulino, N. (2022). *Encendiendo tu motor con tu ECU Programable.* Alphax.

Quito, F., & Sarango, P. (2021). *Diseño e implementación de un prototipo* IOT *de adquisición de datos de OBD-II con monitoreo web server para análisis de detección de fallas.* [Tesis de Pregrado].Universidad Politécnica Salesiana. doi:http://dspace.ups.edu.ec/handle/123456789/20772

Ramirez, G. (2020). *Desarrollo una red de sensores que permita la monitorización de los niveles de contaminación mediante tecnología LPWAN.* [Tesis de Pregrado]. Universidad de las Fuerzas Armadas. doi:http://repositorio.espe.edu.ec/handle/21000/22595

Sampaulo, P. (2020). *LPWAN, las redes del* IoT. RedGPS.

Samsung. (2016). La conectividad según SIGFOX. *Samsung Newsroom España*.

Sánchez, G. (2012). *Sistema posicionamiento global (GPS) y las teorías de la relatividad*. Obtenido de http://web.usal.es/guillermo

Sigfox. (2017). *Sigfox Techinical Overiew*. Obtenido de https://ismacnc.net/wp/wpcontent/uploads/2017/08/Sigfoxtechnicaloverviewjuly2017-170802084218

Tapia, C., & Manzano, H. (2013). *Evaluación de la plataforma Arduino e Implementación de un.* [Tesis de Pregrado]. Universidad Politécnica Salesiana sede Guayaquil. doi:http://dspace.ups.edu.ec/handle/123456789/5522

Thinxtra. (2019). *Thinxtra Sigfox Developer Xkit.* Obtenido de Thinxtra The IoT Telco: https://thinxtra.com/IoT-connectivity/xkit/

Villavicencio, I., Verduzco, J., García, N., Figueroa, P., González, J., & Ortiz, A. (2020). *Plataforma* IoT *para el rastreo y monitoreo remoto de parámetros de vehículos.* [Tesis de Pregrado]. Instituto Tecnológico de Colima. doi:https://doi.org/10.23857/dc.v6i3.1276

Williams, E. (2015). *Arduino SRL to Distributors: "We're the Real Arduino"*. Obtenido de Hackaday: https://hackaday.com/2015/03/28/arduino-srl-to-distributors-were-the-real-arduino/

Wireless. (2015). *A comparative study of LPWAN technologies for large-scale* IoT*. Wireless & RF Magazine*.

Zuo, B. (2022). *CAN-BUS shield V2.0*. Obtenido de Seeedstudio: https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/

Apéndice