



UNIVERSIDAD DON BOSCO

VICERRECTORÍA DE ESTUDIOS DE POSTGRADO

TRABAJO DE GRADUACIÓN

**COMPUTACIÓN EN LA NUBE E INTERNET DE LAS COSAS.
PROPUESTA DE DISEÑO DE ARQUITECTURA PARA PROTOTIPO DE
HARDWARE PARA LA EXTRACCIÓN DE DATOS DE LOS VEHÍCULOS CON
SISTEMA DE OBDII.**

**PARA OPTAR AL GRADO DE
MAESTRO EN ARQUITECTURA DE SOFTWARE**

ASESOR:

IVÁN ORLANDO ALVARADO NIÑO

PRESENTADO POR:

JORGE ERNESTO ALMENDÁREZ

JOSÉ FERNANDO DURÁN

EDWIN ALBERTO MORALES

Antiguo Cuscatlán, La Libertad, El Salvador, Centroamérica

Enero 2019

Tabla de contenido

1	Introducción.....	7
2	Estado del arte	8
2.1	Definiciones de IoT.....	8
2.1.1	Historia de la IoT.....	8
2.1.2	Como inciden las IoT en la industria y en los sistemas empresariales	9
2.1.3	Elementos de IoT	11
2.1.4	Como funciona las IoT	12
2.1.5	Seguridad en las IoT	13
2.2	OBD	14
2.2.1	Definiciones de On Board Diagnostic System OBD.....	14
2.2.2	Historia de la OBD	14
2.2.3	Extracción de códigos PID.....	15
2.3	Sistemas embebidos.....	17
2.3.1	Microcontroladores	17
2.4	Computación en la nube	18
2.4.1	Definiciones de computación en la nube	18
2.4.2	Modelos de servicio de computación en la nube	19
2.4.3	Serverless.....	20
2.4.4	Tipos de computación en la nube (Cloud).....	20
2.4.5	Seguridad en la nube.....	20
2.4.6	Web Services	21
2.4.7	Tipos de mensajes.	21
3	Diagnóstico	22
3.1	Historia de la empresa	22
3.2	Contexto actual.....	22
4	Metodología.....	24
4.1	Fase de la investigación	25
4.1.1	Fase 1	25
4.1.2	Fase 2	25
4.1.3	Fase 3	25
4.1.4	Fase 4	25
4.1.5	Fase 5	26

4.2	Metodología para la selección del prototipo de hardware	26
4.2.1	Características y criterios de evaluación	26
4.3	Perfil para la selección de expertos.....	28
4.4	Experto automotriz – OBDII:.....	28
4.5	Experto en integración de sistemas:	28
4.6	Experto en IoT – Cloud Computing:.....	28
4.7	Entrevistas	28
4.8	Técnico de taller	28
4.9	Gerencia de TI (Tecnologías de la Información)	29
4.10	Experto Automotriz.....	29
4.11	Experto en Cloud computing.....	29
4.12	Alcances y limitaciones.....	29
4.12.1	Alcances	29
4.12.2	Limitaciones.....	29
4.13	Entregables.....	29
4.14	Plan de trabajo.....	30
5	Diseño de la API.....	32
6	Requisitos.....	32
6.1	Lista de requisitos de software.....	32
7	Selección de hardware para la extracción de la información	33
7.1	Matriz de evaluación de prototipos de hardware.....	33
8	Diseño de algoritmo para la extracción de la información	35
8.1	Diagrama de clase	35
8.2	Diagrama de componentes.....	36
8.3	Diagrama de secuencia	37
8.4	Diagrama de Estado	38
8.5	Diagrama de actividades.....	39
8.6	Descripción de librerías y funciones utilizadas.	40
9	Diseño de Arquitectura de servicios en la nube para la recepción y almacenamiento de la información.....	40
9.1	Descripción de Servicios	40
9.2	Diagrama de servicios expuesto para el envío de la información.....	42
9.3	Comunicación con la API.....	42

9.4	Diagrama de componentes.....	43
9.5	Diagrama de actividades.....	44
10	Requisitos mínimos.....	45
11	Resultados.....	46
11.1	Pruebas iniciales: Algoritmo de extracción.....	46
11.2	Pruebas Iniciales: Computación en la Nube.....	48
12	Conclusiones	50
13	Recomendaciones	51
14	Bibliografía	52
15	Catálogo de Tablas.....	54
16	Catálogo de Ilustraciones.....	54
17	Anexos	56
17.1	Anexo 1 – Tabla de principales PID para OBD	56
17.2	Anexo 2 – Algoritmo de extracción de datos.....	58
17.3	Anexo 3 - Serverless.....	62
17.4	Anexo 4 – Instrumentos de recolección de datos cuantitativos	67
17.5	Anexo 5 – Códigos de diagnostico	68

Resumen

Este trabajo presenta un diseño de arquitectura para un prototipo de hardware, para la extracción de información de los vehículos automotores que poseen el sistema de diagnóstico a bordo, para que luego, esta información sea recolectada por servicios de internet de las cosas y computación en la nube. Posteriormente la información es expuesta para su análisis en herramientas de inteligencia de negocio o también puede ser manipulada en sistemas empresariales. Este diseño es una propuesta para el caso de estudio Gx, al cual se realizaron entrevistas, análisis de los requisitos funcionales y no funcionales; adicional se realizó investigación sobre las tecnologías IoT y computación en la nube. Ya que el diseño está basado sobre un caso de estudio es fácilmente adaptable a otros casos de estudio que cumplan con los requisitos mínimos.

Términos

IoT: Internet of things – Internet de las cosas.

RFID: Radio Frequency Identification.

AWS: Amazon Web Services.

OBD: On Board Diagnostics System.

ERP: Enterprise Resource Planning.

M2M: Machine To Machine.

PID: Parameter ID.

ROM: Read Only Memory.

RAM: Random Access Memory.

DTC: Data Trouble Code – Código de Diagnóstico de Error.

CAPITULO I: Estado del arte

1 Introducción

Dado el avance, impacto y crecimiento que ha tenido la Computación en la Nube, y específicamente, las tecnologías de IoT, se ve la oportunidad de aportar acerca del ecosistema alrededor de ellas, de manera teórica, así como también de manera práctica por medio de un diseño arquitectónico y un prototipo funcional.

La comunicación con diferentes dispositivos que forman parte de la vida diaria ha pasado de ser un pensamiento futurista a una realidad. En el año 1999 Kevin Ashton daba vida por primera vez al término “Internet de las Cosas” en una presentación sobre tecnologías RFID, para el cual concibió un sistema de sensores ubicuos que conectan el mundo físico con Internet; aunque las cosas, Internet y la conectividad son los tres componentes principales de IoT, el valor está en cerrar la brecha entre el mundo físico y digital en los sistemas (AWS, 2018).

Se ve la oportunidad de aportar con la investigación de IoT por la poca exploración en el campo académico en El Salvador, previamente se realizó una investigación bibliográfica en las universidades y se encontró poco avance en este tema, por lo que esta investigación abona un material útil acerca de estas tecnologías que, según muchos autores y conferencistas, tomará un rol importante en nuestras vidas en un futuro cercano.

Con la evolución de las tecnologías se buscan formas de simplificar el desarrollo de actividades y tener un mejor control sobre aspectos del día a día. Con esto en mente se toma como caso de estudio una empresa salvadoreña dedicada a la comercialización y arrendamiento financiero (leasing) de vehículos automotores, comercialización de repuestos, servicios de reparación de vehículos, entre otros.

Tomando este caso de estudio, la empresa, en el área de arrendamiento financiero, no cuenta con métodos formales o procesos establecidos para obtener la información de manera oportuna del estado de las diferentes unidades arrendadas que se encuentran en circulación. En la actualidad los vehículos traen incorporado un sistema de diagnóstico a bordo también conocido como OBD. Este sistema permite conocer el estado del vehículo y si este posee fallas a nivel mecánico, químico o eléctrico. El sistema realiza un escaneo de todos los sensores del vehículo para determinar su condición; teniendo esta información el sistema lo traduce a códigos estándares que son definidos por el fabricante y pueden ser extraídos por medio del puerto OBD.

Se realizará una investigación del funcionamiento de este sistema a bordo de los vehículos y el cómo integrarlo con IoT para el manejo de los datos y que posteriormente la empresa en estudio pueda utilizar dicha información para sus fines operativos.

2 Estado del arte

2.1 Definiciones de IoT.

Internet de las cosas no es un concepto nuevo en la industria, pero aún sigue evolucionando en distintas áreas. Algunos autores definen IoT como escenarios en los que la conectividad de red y la capacidad de cómputo se extienden a objetos, sensores y artículos de uso diario que habitualmente no se consideran computadoras, permitiendo que estos dispositivos generen, intercambien y consuman datos (Karen Rose, 2015), este mismo autor hace referencia que para IoT no existe una definición única y universal, pero algo en los que muchos coinciden es en la conectividad de los dispositivos por medio de internet para lograr uno o varios objetivos específicos.

Según Google Cloud, IoT es un extenso conjunto de tecnologías y casos de uso que no tienen una clara y única definición. Un punto de vista viable enmarca a IoT como el uso de dispositivos conectados a la red, integrados en el entorno físico, para mejorar algunos procesos existentes o para habilitar un nuevo escenario que no era posible anteriormente.

Por otra parte, Microsoft define a IoT, en ocasiones llamada Internet de Todo, como una red gigante de objetos que se conectan entre sí e intercambian y analizan datos.

Héctor Cuesta (experto en IoT y Ciencia de Datos), en su presentación “Seguridad en Dispositivos IoT” en 2016, donde define las IoT como una red universal de objetos interconectados y direccionables basada en protocolos de comunicación estándar.

2.1.1 Historia de la IoT

El origen de los objetos conectados se remonta a los comienzos tecnológicos del siglo XIX, siendo el primer experimento de telemetría de la historia (del que se tiene constancia), llevado a cabo en 1874 por científicos franceses. Para ello se instalaron dispositivos de información meteorológica y de profundidad de nieve en la cima del Mont Blanc. Con esto, y mediante un enlace de radio de onda corta, transmitían los datos a París.

Posteriormente, se realizaron otros experimentos similares en el siglo XX por iniciativas originadas en países como Rusia o Estados Unidos, ayudando al crecimiento de la telemetría y llevándola a un uso extensivo impulsado por la evolución de diversas tecnologías de comunicación.

Este tipo de ideas sobre poder conectar los objetos y que estos fuesen inteligentes, se podía encontrar ya plasmadas en los pensamientos y escritos de científicos como Nikola Tesla y Alan Turing.

En 1926, Nikola Tesla ya se anticipaba de forma sorprendente al crecimiento de la conectividad a nivel global y la miniaturización tecnológica, y fue uno de los padres de las comunicaciones inalámbricas.

En 1950, Alan Turing avanzó la necesidad futura de dotar de inteligencia y capacidades de comunicación a los dispositivos sensores, pero a pesar de sus postulados y la visión tan temprana

que supieron transmitir varios científicos, incluyéndole, la inmadurez tecnológica de la época hizo que todo esto quedase como propuestas irrealizables.

Entre los años 60 y 70 nace la ARPANET, lo que sería la base de lo ahora conocido como Internet y es hasta mediados de los 90 que el Internet comercial y universal comenzó su expansión definitiva. Luego de esto y en el año 1990, John Romkey creó el primer objeto conectado a Internet, el cual fue una tostadora que se podía encender o apagar de forma remota.

Es en 2009 cuando Kevin Ashton, profesor del MIT en ese entonces, usó por primera la expresión “Internet of Things” (IoT) de forma pública y desde entonces el crecimiento y la expectación alrededor del término ha venido aumentando de forma exponencial.

A finales del 2012, ya había 8,700 millones de dispositivos conectados en el mundo y, a medida va avanzando, Cisco apunta que para el 2020 se alcanzará alrededor de 50 billones de dispositivos conectados (Brono Cendón, 2017).

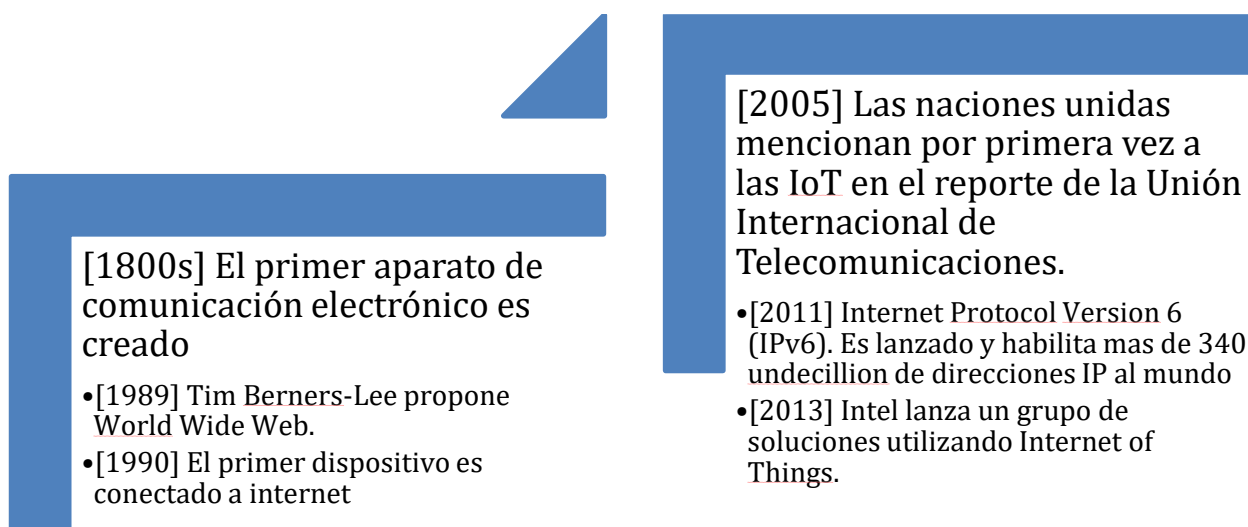


Ilustración 1 Evolución de IoT
Elaboración propia

2.1.2 Como inciden las IoT en la industria y en los sistemas empresariales

El IoT ha venido a hacer un impacto en diferentes sectores, tanto tecnológicos como empresariales, esto debido a que hay una enorme cantidad de ventajas y un sin fin de posibilidades al tener diferentes dispositivos conectados a internet y los sensores que estos contienen, generando una gran cantidad de datos potencialmente útiles para ser analizados. Dichos datos crecen día a día con el incremento de los dispositivos conectados, lo cual afecta a los distintos procesos de los negocios, tales como la planificación de recursos empresariales.

Como es bien conocido, los sistemas ERP pueden llegar a proveer de múltiples funciones críticas como la mejora en la toma de decisiones en las empresas, la eficiencia, informes mejorados, etc. Lo cual combinado con la integración de IoT se puede añadir mayor ventaja.

Entre los impactos que pueden causar IoT en los ERP son:

Expansión de Mercado de los ERP: desde hace algunas décadas los ERP han tenido evolución en el mercado, y es bien conocido que se invierten miles de millones en la implementación de estos sistemas. Según datos publicados por **Allied Market Research** en el reporte de **Mercado de Software ERP (Shrikant Chaudhari, 2013)**, las proyecciones indican que el mercado global de ERPs espera superar un valor de \$41.69 mil millones para 2020, con la adopción de los ERP por parte de las empresas contribuyendo con una tasa de crecimiento anual de casi un 8%. Por parte, diferentes organizaciones en diferentes sectores buscan implementar soluciones IoT con el fin de tener un respaldo en la toma de decisiones basadas en datos actualizados. **La Guía de Gastos (International Corporation Data, 2016)** publicada por **International Data Corporation (IDC)**, menciona que el gasto mundial en IoT crecerá a una tasa anual del 17%, en 2015 aproximadamente \$700 billones a casi \$1.3 trillones en 2019. Y también hace una predicción en que el 40% de todos los datos vendrán como resultado de máquinas y de otros sistemas inteligentes para el año 2020, y que hasta 50 billones de dispositivos contribuirán a la información generada por la tecnología.

Con los datos anteriores podemos apreciar que ambas soluciones (IoT y ERP) siguen incrementando los beneficios que ofrecen a grandes y pequeñas empresas, y dichos beneficios llegan a una intersección. **Research and Markets** en (Research and Market, 2017) señala que IoT-ERP se convertirá en una oportunidad considerable para las empresas y proveedores de servicios por igual, y se espera que este mercado alcance \$49.9 billones para 2022. **Research and Markets** también menciona que los ERP proporcionan un repositorio central para toda la información empresarial y permiten la mejora del flujo de datos a través de toda la organización, por otra parte, el software y sistemas de IoT lo transforman aún más, ya que las soluciones ERP habilitadas para IoT conectan personas, procesos, datos y cosas en un repositorio.

Beneficios comerciales por integración IoT-ERP: como menciona **Research and Markets** en (Research and Market, 2017), la intersección del crecimiento entre IoT y ERP parece natural, ya que los datos proporcionados por los sensores y sistemas IoT ayudarán a informar mejor la disponibilidad de los datos de los ERP. Por lo que se espera que IoT mejore la eficiencia de los ERP, que permita y facilite la creación de nuevos modelos comerciales y que provea de una alineación de las operaciones físicas con los activos digitales en tiempo real.

La integración IoT-ERP también permitirá una mejora en la toma de decisiones por parte de la alta dirección, en especial si los sistemas IoT se utilizan en los procesos más importantes de la empresa. Por ejemplo, los diferentes sensores conectados pueden comunicar detalles sobre la disponibilidad de inventario, lo que permitiría a los supervisores gestionar mejor el pedido y reabastecimiento, y a su vez, eliminan la necesidad de interacción humana.

Por otra parte, los sistemas empresariales tienen un papel importante en las organizaciones para el control de la información de sus procesos, clientes, proveedores, productos, finanzas, etc. pero

también es impactado con las IoT por medio de la información que los dispositivos pueden recolectar; esta información puede ser utilizada para estrategias de mercado, mejora de sus productos, conocer las tendencias o preferencias de sus clientes, proveer productos a tiempo y más. Los restaurantes, hoteles, parques, estadios, etc. son puntos de enfoque para dispositivos con dispensador de productos ej. Máquinas dispensadoras de soda, toallas, alimentos, productos de sanidad; estos dispositivos alertan a la central (ES) del proveedor con la cantidad de producto disponible; el sistema empresarial analiza la información y por medio de los procesos de negocio o flujos de toma de decisión se agendan rutas para la entrega de nuevo producto a sus clientes. Esto mejora el proceso de cadena de suministro ya que el cliente no tiene la necesidad de revisar los dispensadores periódicamente, realizar los pedidos manualmente al proveedor y no enfocarse en su negocio (Jorge Alvarado, 2017).

Manufactura inteligente: El servicio o función de una máquina o una parte de ella pueden mejorar antes de que se presente una falla, eliminando así los costosos tiempos de inactividad y eventualidades no previstas.

Cadenas inteligentes de suministro: Proporcionando información en tiempo real de la oferta, demanda y envíos a los clientes. Las entregas pueden ser rastreadas y recuperadas si son extraviadas o robadas.

Infraestructuras inteligentes: Las oficinas inteligentes contribuyen a generar ahorros de energía, mejorar la auto sustentabilidad y potenciar la colaboración entre los empleados.

Las IoT integrado con los sistemas empresariales busca lo siguiente:

- Velocidad en la respuesta ya que está orientado a servicios
- Infraestructura de comunicación
- Toma de decisiones en tiempo real
- Optimiza los procesos del negocio
- Análisis de información del mercado
- Datos centralizados en sistemas empresariales

2.1.3 Elementos de IoT

Las IoT deben considerar ciertos elementos para su aplicación; es decir elementos que soportan la comunicación entre los dispositivos e informar de alguna forma al usuario que ha iniciado o finalizado su tarea o función designada. A continuación, se presentan 6 elementos que ImapaTech considera importantes para la aplicación de IoT (ImapaTech, 2017).

- **Transmisión de Datos:** Las aplicaciones de IoT acumulan más información de lo que el procesamiento tradicional puede administrar; tener capacidad para transmitir esta información continuamente es clave para alimentar de forma fiable, y en tiempo real, procesos de negocios y extraer información oportuna.

- **Plataformas IoT:** Una plataforma IoT permite desarrollar, implantar y administrar aplicaciones IoT y M2M: automatizar procesos y conexiones de red, almacenar y administrar datos de sensores, conectar y controlar los dispositivos y analizar la información.
- **Aplicaciones IoT:** Las aplicaciones IoT deben ser capaces de capturar, recolectar, interpretar y actuar en grandes cantidades de información, detectando brechas de conectividad, manejando interrupciones y cumpliendo con requisitos específicos del negocio y la industria.
- **IoT Cloud:** Las soluciones IoT en la nube proveen acceso asequible a redes de datos de alta velocidad para ampliar significativamente el alcance y la usabilidad de sus aplicaciones. También puede ofrecer almacenamiento de datos procesamiento, análisis y administración remota de dispositivos.
- **Datos IoT:** Las tecnologías de gestión de datos IoT garantizan la recopilación de los datos de forma correcta y en el momento oportuno, incluso cuando la conectividad se interrumpe.
- **Seguridad IoT:** Los enormes volúmenes de datos de dispositivos en IoT deben protegerse de robo o alteración desde el momento de su extracción; la seguridad tiene que continuar mientras los datos están en tránsito a través de la red, en reposo en los almacenes de datos y cuando las aplicaciones los utilizan.

2.1.4 Como funciona las IoT

IoT funciona básicamente por medio de dispositivos que se conectan a través de WIFI, Bluetooth, 3G/4G u otros protocolos de red. Estos dispositivos están equipados con diferentes sensores de detección como, por ejemplo: sensores de movimientos, cambios de luz, gravedad, sonido, etc. El dispositivo está programado para realizar ciertas actividades o también puede ser capaz de recibir órdenes de tareas a ejecutar por medio de la comunicación inalámbrica (Amidata S.A, 2017).. Adicional a esto debe existir un sistema que recolecta esta información y la envía a la nube, donde la información recolectada será analizada y procesada por los servicios de IoT (IBM Think Academy, 2015).

La comunicación de estos dispositivos debe tener ciertas características que deben cumplir como la conectividad entre dispositivos y protocolos de seguridad, esto con el fin de proteger la integridad de la información. A continuación, se muestra un diagrama del funcionamiento y componentes que permiten la concepción de IoT (Amidata S.A, 2017).

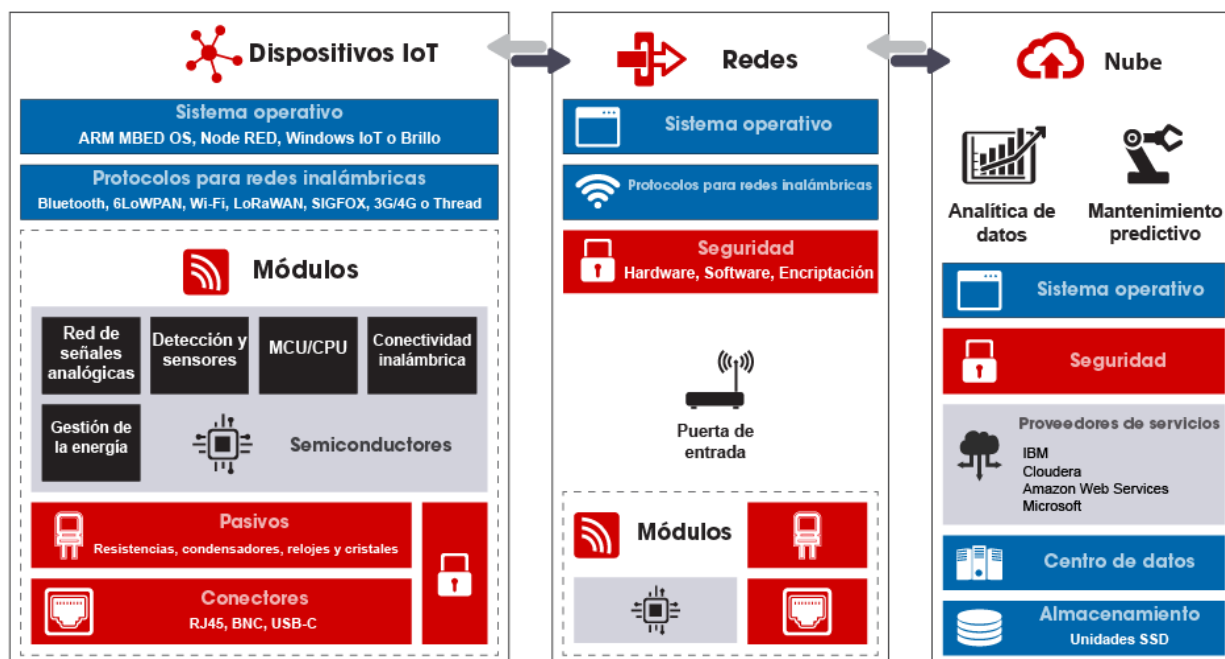


Ilustración 2 Arquitectura IoT

Referencia: <http://img-europe.electrocomponents.com/>

El diagrama muestra la interconexión de dispositivos con la nube u otros dispositivos mediante algún tipo de red (bluetooth, wifi, etc.) y por la cual pueden transferir o solicitar información uno al otro, todo esto con el fin de centralizar la información obtenida de los dispositivos para su posteriormente analizarlos o disponer de ellos.

2.1.5 Seguridad en las IoT

La seguridad de la información es un tema sensible desde todos los ámbitos y es por lo que IoT también debe ser seguro para proteger la integridad, confidencialidad, disponibilidad e identidad de las personas que utilizan estos servicios. La organización OWASP (Open Web Application Security Project) es una organización que se dedica a la generación de buenas prácticas de seguridad y considera un enfoque crítico para sistemas IoT (Héctor Cuesta, 2017). Algunos enfoques críticos para considerar son:

- Interfaz Web Insegura
- Autenticación insuficiente
- Servicios de red inseguros
- Falta de encriptación en el medio de transporte / integridad de la verificación
- Preocupaciones de seguridad
- Interfaz de Nube insegura
- Interfaz Móvil insegura
- Configuración de seguridad insuficiente
- Software o Firmware inseguro

- Seguridad Física pobre

Por ello IoT debe contener protocolos de seguridad suficientes para asegurar la transmisión de la información hacia la nube sin ser vulnerada por personas maliciosas que buscan robar la información que es transmitida vía internet.

2.2 OBD

Cada vez existen más dispositivos conectados a internet y muchas de las brechas de seguridad no son solventadas y puede llegar ser más críticos con el uso que se le puede dar; según el informe de Julio de 2014 de HP FORTIFY el 80% de los dispositivos tienen fallos en la autenticación y 6 de cada 10 dispositivos con interfaz de usuario son vulnerables (Marco Arenas, 2016).

2.2.1 Definiciones de On Board Diagnostic System OBD

OBD es un sistema para el monitoreo y diagnósticos de fallas de los vehículos automotores, consiste en recolectar información de los componentes del vehículo y así alertar al usuario de posibles fallas mecánicas, eléctricas, entre otras. El acceso al sistema OBD se realiza por medio de un puerto también conocido por el mismo nombre OBD que en muchos modelos se encuentra en la parte baja del control de mando o bajo el capo del vehículo, este permite el acceso a las fallas detectadas como por ejemplo la temperatura del motor, nivel de aceite, nivel de voltaje de batería, falla en sistemas ABS, emisión de gases, etc.

2.2.2 Historia de la OBD

A principios de la década de 1980, los fabricantes de automóviles comenzaron a usar productos electrónicos y a bordo, computadoras para controlar las funciones del motor. La creciente complejidad de la tecnología de los vehículos llevó a los fabricantes a desarrollar formas de diagnosticar eficazmente problemas en componentes del vehículo. Por lo tanto, la forma más temprana de OBD del vehículo fue desarrollada para disminuir el tiempo de inactividad dedicado a diagnosticar vehículos.

En 1989, CARB emitió regulaciones que requieren la segunda generación de regulaciones OBD, a menudo conocido como OBDII. CARB requirió sistemas OBDII en 1994 y más tarde en vehículos livianos, camiones y vehículos con motores de servicio mediano vendidos en California. En 1990, el Congreso finalizó la Enmienda a la Ley de Aire Limpio que incluyen un mandato a la Agencia de Protección Ambiental para desarrollar regulaciones que requieren sistemas OBD en todos los vehículos de 1994 y posteriores, vendidos a nivel nacional y conocidos como OBD federal. Estas regulaciones expandieron la lista de componentes que fueron monitoreados para incluir componentes relacionados con la emisión y se agregó una función de autodiagnóstico que evaluó el componente de condición más allá de la simple conectividad y verificaciones de falla que anteriormente existían en la primera generación de vehículos OBD u OBD I. Esto aumentó aún más la complejidad de la tecnología del vehículo, pero también agregó cantidades significativas de información disponible para diagnosticar problemas del vehículo (Arvon L. Mitcham, 2000).

2.2.3 Extracción de códigos PID

Para poder realizar extracción de datos de los vehículos se debe comprender el funcionamiento del sistema OBDII, este funciona por medio de un conjunto de sensores instalados en el vehículo que están constantemente enviando señales del estado de sus componentes y se puede trabajar en nueve modos, comunes entre todos los vehículos que implementan la tecnología, que van desde extraer datos para su verificación, extraer códigos de averías, borrados de datos y realizar pruebas dinámicas. El modo en que estos datos se presentan se manejan bajo un estándar y de igual forma se maneja el tamaño de trama de consulta OBDII.

Byte #0	Byte #1	Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7
---------	---------	---------	---------	---------	---------	---------	---------

Ilustración 3 Trama estándar OBDII
Elaboración propia

El contenido de cada “Byte” depende de si la trama es de petición o respuesta y el modo en que se está accediendo, a excepción del #0 que contiene el número de bytes que poseen información.

Este trabajo de investigación utiliza específicamente 2 modos, los cuales son el “modo 01”, por medio del cual se pueden consultar parámetros actuales del vehículo y el “modo 03” el cual consulta los códigos de averías detectados por la computadora del vehículo.

2.2.3.1 Modo 01 – datos de diagnóstico actuales.

Este modo sirve para consultar datos en tiempo real sobre el estado del vehículo, como la temperatura del motor o las revoluciones por minuto. Para realizar una consulta en este modo, se debe enviar en la trama de OBDII el valor PID (por sus siglas en inglés) a extraer; puede consultarse más de un valor PID en la misma trama hasta un máximo de 6, la forma en que la trama debe llenarse es la siguiente:

Tabla 1 Trama de consulta modo 1
Elaboración propia

Byte	Parámetro	Valor Hex.
#1	Indica que se está realizando una petición en el Modo 01	01
#2	Valor hexagesimal del PID requerido	XX
#3	(Opcional) Valor hexagesimal del segundo PID requerido	XX
#4	(Opcional) Valor hexagesimal del tercer PID requerido	XX
#5	(Opcional) Valor hexagesimal del cuarto PID requerido	XX
#6	(Opcional) Valor hexagesimal del quinto PID requerido	XX
#7	(Opcional) Valor hexagesimal del sexto PID requerido	XX

2.2.3.2 Modo 03 – Códigos de diagnóstico

Este modo permite extraer todos los códigos de diagnóstico de error (o DTC por sus siglas en inglés) que se encuentran en la computadora del vehículo. Estos códigos contienen la siguiente estructura:

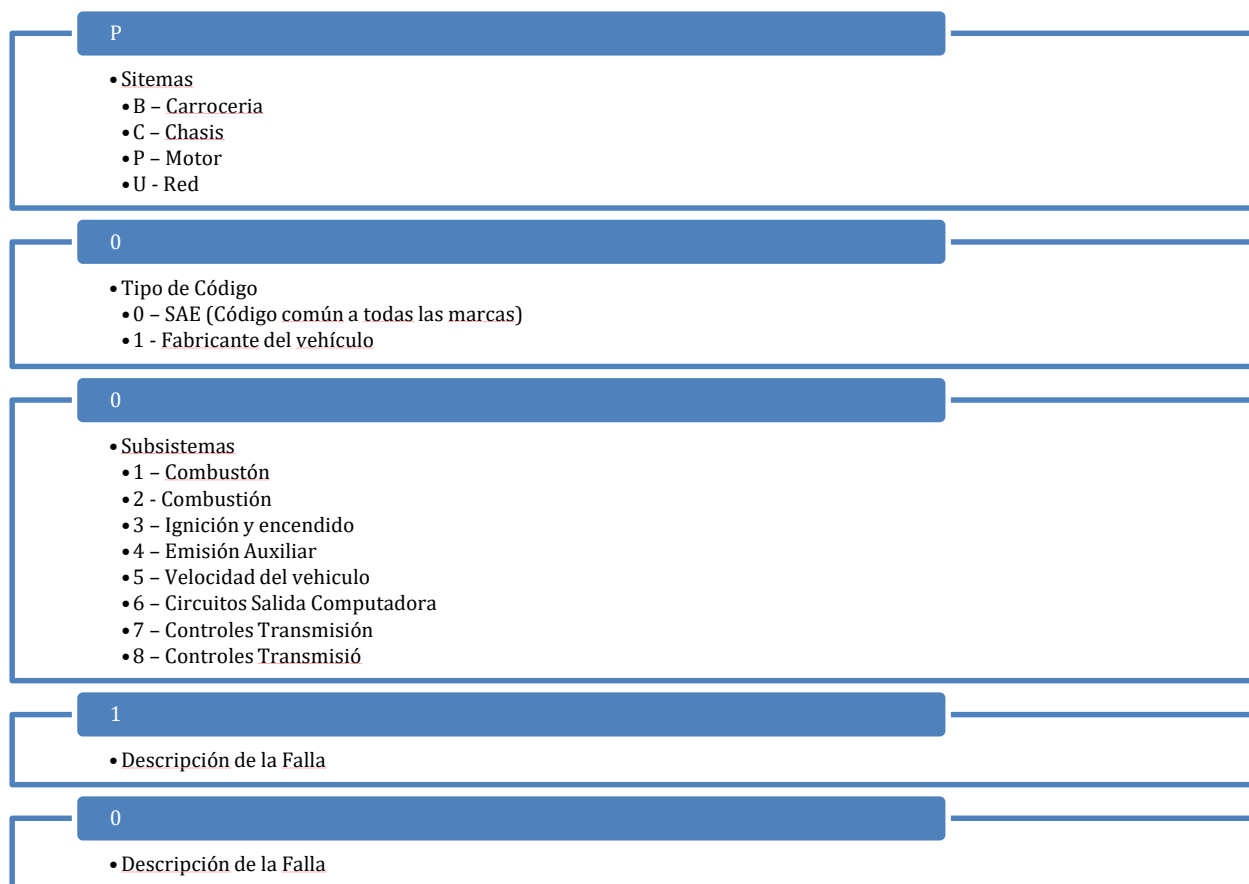


Ilustración 4 Formato de DTC
Elaboración propia

Los códigos que se pueden obtener mediante este modo se pueden categorizar en dos tipos; según el valor del segundo carácter (Tipo de código), estos pueden ser códigos estandarizados o propios del fabricante del vehículo.

Para poder realizar una consulta mediante este modo, se debe llenar en la trama con la siguiente información:

Tabla 2 Trama de consulta modo 3
Elaboración propia

Byte	Parámetro	Valor Hex.
#1	Indica que se está realizando una petición de los DTCs almacenados en Modo 03	03

2.3 Sistemas embebidos

Se conoce como sistema embebido a un circuito electrónico computarizado que está diseñado para cumplir una labor específica en un producto; lo cual lo diferencia de los sistemas computacionales tradicionales, tales como las computadoras personales, servidores o teléfonos inteligentes. Estos sistemas están presentes en casi cualquier elemento electrónico de uso cotidiano y es muy común encontrarlos en electrodomésticos, equipo médico, vehículos, etc.

El número de aplicaciones soportadas por los sistemas embebidos empezó su crecimiento gracias a la llegada de los microcontroladores “single-chip”, los que permiten tener dispositivos de menor tamaño, menor consumo de energía, entre otras facilidades.

El sistema embebido single-chip contiene en una sola pastilla de silicio todo el esquema de un sistema convencional con su procesador en el centro, sistema de decodificación, sistema de multiplexación, memoria ROM, memoria RAM, entradas análogas y digitales. (Gustavo Galeano, 2009).

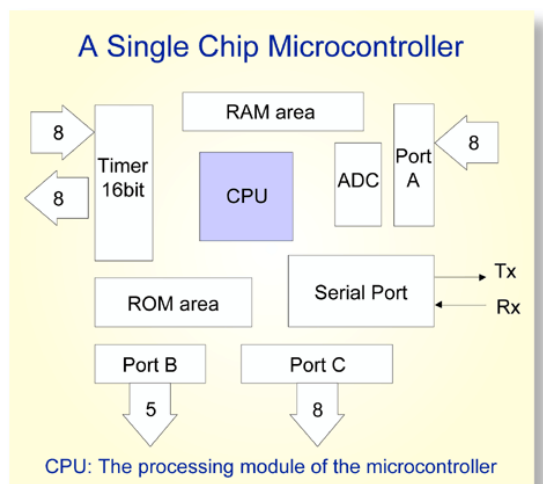


Ilustración 5 Microcontrolador single chip
Referencia Peter Minns, Northumbria University

2.3.1 Microcontroladores

Usualmente se confunden los microcontroladores con los microprocesadores, sin embargo, ambos difieren el uno del otro en muchos sentidos, su diferencia más importante es la funcionalidad. Para utilizar un microprocesador en una aplicación real, se requieren un conjunto de componentes conectados, tales como memoria, componentes de transmisión de datos, etc. Aunque el microprocesador se considera una máquina de computación poderosa, no está preparado para la

comunicación con dispositivos periféricos que se le conectan; para que el microprocesador se comunique con algún dispositivo periférico, es necesario utilizar circuitos especiales.

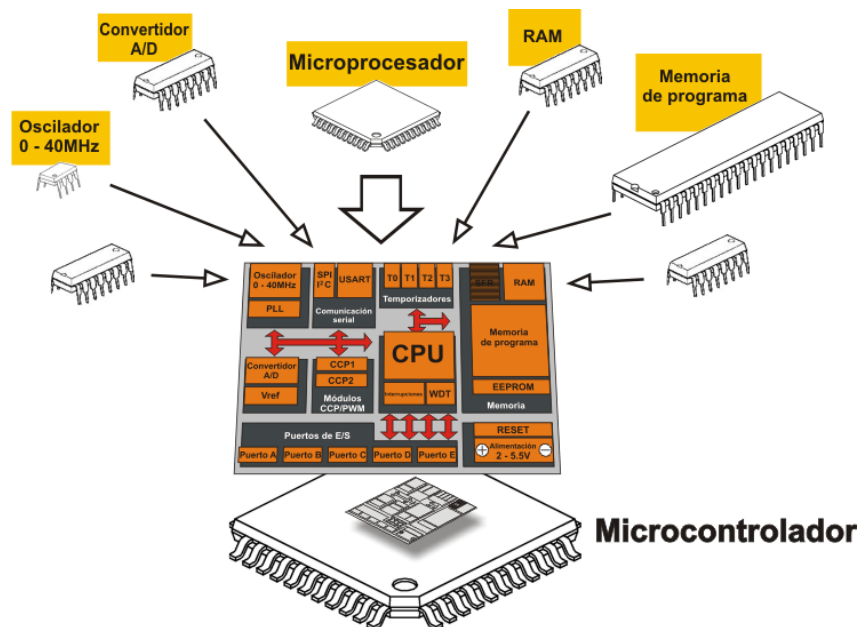


Ilustración 6 Arquitectura de microcontrolador
Referencia: Mikroelectronica

Por otra parte, el microcontrolador es diseñado de tal forma que tenga todos los componentes integrados, no necesita de otros componentes especializados para su aplicación, ya que todos los circuitos necesarios para la comunicación con otros componentes ya se encuentran incorporados (Milan Verle, 2009).

2.4 Computación en la nube

La computación en la nube es una solución integrada basada en servicios por medio de internet y no es necesario tener dichos servicios dentro de la organización, esto facilita que los recursos sean escalables y de alta disponibilidad desde cualquier lugar con acceso a internet (SAP, 2018).

2.4.1 Definiciones de computación en la nube

Computación en la nube o también conocido como “Cloud computing” es una capa de servicios publicados en internet y está disponible todo el tiempo y en cualquier lugar, el único requisito es tener una conexión a internet para poder acceder a los servicios. Beka Kezherashvili en su investigación del 2011 define la computación en la nube como un modelo de presentación de servicios de negocio y tecnología, que permite al usuario acceder a un catálogo de servicios estandarizados y responder a las necesidades de su negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado.

2.4.2 Modelos de servicio de computación en la nube

La computación en la nube incluye varios modelos de servicios, los cuales son ofertados individualmente como servicios en la nube, estos pueden ser Software as a Service, Platform as a Service o Infraestructura as a Service. Estos servicios comúnmente son desarrollados en conjunto para tener un aprovechamiento del aprovisionamiento de software en la nube. A continuación, se muestran algunos ejemplos de los modelos en la nube.

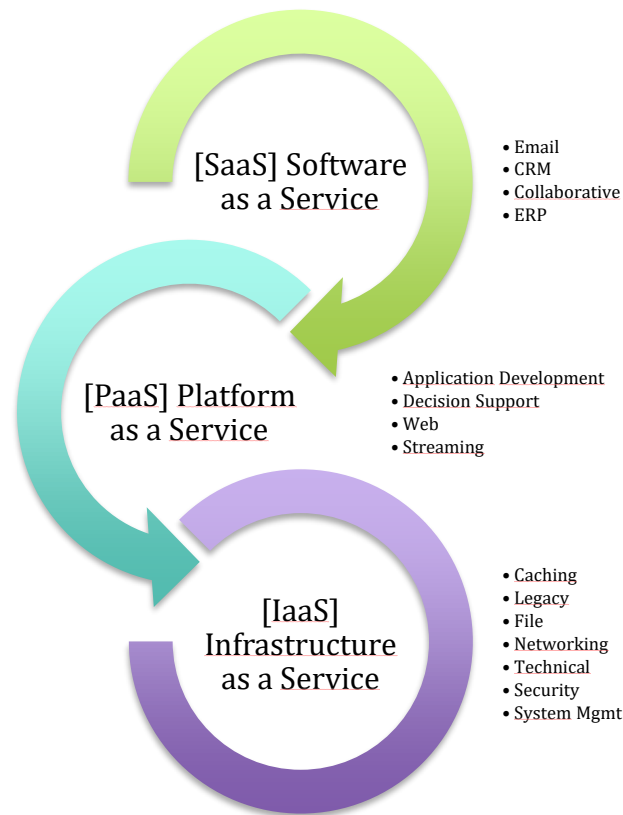


Ilustración 7 Modelos de servicios en la nube
Elaboración propia

2.4.2.1 SaaS

SaaS (Software as a Service) es una forma de distribución de software en la nube, con un régimen de pago normalmente mensual o anual, aunque existen otros que son muy utilizados para este tipo de modelo, donde el proveedor contiene sus aplicaciones en servidores propios permitiendo a sus clientes acceder por medio de internet, estando disponible en todo momento y desde cualquier lugar.

2.4.2.2 PaaS

PaaS (Platform as a Service) son herramientas de software que son utilizadas para el desarrollo de aplicaciones y que está alojada en la nube, permitiendo a los desarrolladores ingresar desde cualquier computadora y comenzar a construir, analizar, desarrollar, probar, documentar y hasta poner en marcha la aplicación, sin tener que instalar algún software en el equipo local.

2.4.2.3 IaaS

IaaS (Infrastructure as a Service) se refiere a la virtualización de servidores. Usualmente la contratación de estos servicios se realiza en base a procesador, memoria y espacio de almacenamiento; siendo escalable según el negocio lo requiera y no es necesario adquirir equipos o actualización de estos.

2.4.3 Serverless

Se refiere a la arquitectura nativa de la nube sin el uso de un servidor, con la cual se le permite trasladar más responsabilidades operativas a los servicios en la nube aumentando la agilidad y la innovación. La informática sin servidor (Serverless), puede crear y ejecutar aplicaciones y servicios sin tener que preocuparse por los servidores. Permite eliminar las tareas de administración de infraestructura, como el aprovisionamiento de servidores o clústeres, los parches, el mantenimiento del sistema operativo y la capacidad de escalamiento de recursos del mismo. Pueden crearse para prácticamente cualquier tipo de aplicación o servicio de backend. Además, administra todo lo necesario para ejecutar y escalar la aplicación con alta disponibilidad (SAP, 2015).

2.4.4 Tipos de computación en la nube (Cloud)

El departamento de seguridad de los Estados Unidos de América clasifica los tipos de computación en la nube como pública, privada, comunitaria e híbrida (Alexa Huth, 2011).

- Pública: Puede ser accedido por cualquier suscriptor que tenga acceso a internet.
- Privada: Se establecen accesos a un grupo específico y es limitado únicamente a ellos.
- Comunidad: Es compartida por dos o más grupos que poseen requerimientos similares.
- Híbrida: La combinación es esencial de al menos dos nubes donde exista un mixto de público, privado o comunidad.

2.4.5 Seguridad en la nube

La seguridad de información es el desafío que muchas empresas tienen al momento de adoptar sistemas y almacenamiento en la nube, ya que todos requieren que su información sea segura y con la garantía de no sufrir pérdida parcial y mucho menos total de la información, así también que la información no pierda su integridad o robo de información. Las empresas que se dedican a servicios en la nube poseen herramientas, recursos, conocimiento y adquieren experiencia para garantizar el resguardo de la información; es una ventaja para tener la confianza de adoptar servicios en la nube poniendo de lado el mantenimiento de recursos, respaldos, restauraciones y la disponibilidad de los servicios y de la información en todo momento.

Preguntas importantes que las empresas deben realizarse antes de implementar soluciones en la nube:

- ¿Quién tendrá acceso a la información?

- ¿Cuáles serán las políticas al acceso a la información?
- ¿La información estará encriptada, en los almacenes de datos, con un eficiente algoritmo de encriptación?
- ¿En caso de desastres, el “data center” o proveedor de la nube mantendrá una replicación de la información?
- ¿Cuál es el proceso que se aplicara para el respaldo de la información y cuál es el proceso para la restauración del respaldo en caso exista una catástrofe?

2.4.6 Web Services

Es un sistema de comunicación entre diferentes sistemas informáticos para compartir documentos estándar (XML, JSON, etc.) y que permite ser desarrollado en distintos lenguajes de programación ejecutado sobre cualquier plataforma. El intercambio de información se realiza a través de internet (Aros, 2009).

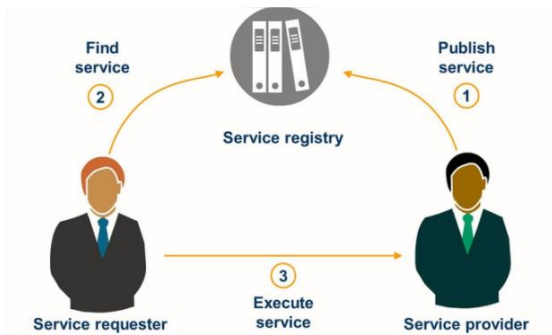


Ilustración 8 Trama de consulta modo 3
Referencia Handbook SAP Integration Overview

2.4.7 Tipos de mensajes.

El intercambio de información se realiza por medio de mensajes estándar que definen un formato previamente definido entre ambos sistemas, los más utilizados son:

XML (Extensible Markup Language) es un lenguaje de marcado muy parecido a HTML que se utiliza para el diseño de páginas web, el lenguaje XML es el recomendado por W3C para el intercambio de información y/o almacenamiento de información por alta compatibilidad entre sistemas, seguridad y sencillez.

JSON (JavaScript Object Notation) es un lenguaje de notación moderno utilizado para el intercambio de información vía Web services por su facilidad de leer, escribir y convertir para los desarrolladores y sistemas informáticos. Este es compuesto por medio de una colección de nombres y valores generando así una lista de valores y objetos.

3 Diagnóstico

El presente trabajo contempla utilizar como referencia de investigación a la empresa Grupo Gx, la cual cuenta con presencia en Centroamérica y se dedica a la distribución de vehículos automotores de distintas marcas y ofrece varios productos como lo son:

- Venta de vehículos: es la venta de vehículos al contado o con financiamiento con bancos o la financiera de Gx.
- Arrendamiento de vehículos: es el arrendamiento de un vehículo para uso particular o comercial; con un pago mensual pagado al arrendante (Gx).
- Servicios de mantenimiento y reparación de vehículos: corresponde al servicio de taller.
- Restauración y pintura de vehículos: Es el servicio de pintura personalizado para el vehículo.
- Financiamiento de vehículos: ofrece servicios financieros como préstamos para la compra de un vehículo automotor, este puede ser de Grupo Gx o de un tercero, incluso vehículos usados en buen estado.
- Venta de repuesto de vehículos: la sucursal cuenta con venta de repuestos nuevos y originales de los vehículos, lubricantes, llantas, etc.

3.1 Historia de la empresa

La empresa Grupo Gx fue fundada en 1952 con el sueño del visionario de S.Q. en El Salvador, empresa que evolucionaría, para 50 años más tarde convertirse en Grupo Gx, una corporación regional que actualmente cuenta con presencia en todos los países de Centroamérica.

Hoy en día, Grupo Gx, distribuye una serie de marcas de vehículos en el país como lo son Nissan, Mazda, Hyundai, Porsche, entre otras. También ofrece distintos productos y servicios como:

- Repuestos originales de vehículos
- Servicios de taller
- Restauración y pintura
- Financiamientos de vehículos, etc.

3.2 Contexto actual

La investigación se enfoca en uno de los productos que la empresa ofrece llamado “leasing” o arrendamiento financiero de vehículos. El producto es otorgado a clientes naturales y jurídicos. Con el fin de arrendar un vehículo, el cliente tiene la opción de que al final del arrendamiento el vehículo pueda ser adquirido por el arrendante, es decir que el vehículo pase a ser de su propiedad o tenencia según la entidad de otorgar el derecho de circulación. Por ser un servicio de arrendamiento el bien (vehículo) pertenece legalmente a la empresa arrendadora (Gx) por lo que debe mantener su activo en las mejores condiciones posibles; para ello existen recursos con la asignación de administrar dichos bienes por medio de mantenimientos (preventivos y correctivos) y adicional la parte

administrativa del servicio creando excelencia en el servicio brindado a los clientes. La administración del vehículo está conformada principalmente por las siguientes actividades:

- Negociación (oferta)
- Seguimiento con aseguradoras
- Garantías de vehículos
- Pagos
- Emisión de créditos fiscales
- Cobranzas
- Recepción y seguimiento de los pagos
- Seguimiento de mantenimientos y reparaciones
- Reportes por accidentes
- Reportes por defectos del vehículo
- Mantenimiento preventivo

La administración de las flotas vehiculares es una problemática con la que Gx se enfrenta, debido a que, conforme la cantidad de vehículos arrendados es mayor en comparación hace unos años (y se pronostica aumentar la cartera de clientes con este producto en unos cuantos años) la administración del producto se vuelve difícil de soportar ya que los procesos de administración de los vehículos son manuales y es casi imposible tener un panorama claro de las fallas que el vehículo puede estar presentando o la administración de los mantenimientos preventivos. Los administradores de este producto no tienen la visibilidad del momento de la falla, puede llegar hasta el punto de pérdidas totales de las unidades por irresponsabilidad del arrendante o por mantenimientos que pudieron alargar la vida útil del vehículo según las recomendaciones de cada fabricante. La administración de los vehículos se realiza por medio de correos electrónicos, llamadas telefónicas, hojas de electrónica, documentos impresos, etc. la administración se vuelve inmanejable por los administradores del producto. Para el caso de programar citas a taller aun siendo de la misma empresa la visibilidad o rastreo de las reparaciones que se realizan a los vehículos, tiempos de entrega, costos, y automatización de citas no existen, esto ocasiona reprocesos en la gestión y administración del producto.

La excelencia por el servicio es uno de los valores que Gx impulsa y por ello los productos y servicios deben ser presentados oportunamente y con la calidad que los clientes se merecen es por ello que se busca sistematizar la gestión y administración del producto de arrendamiento de vehículos para evitar que los vehículos circulen en mal estado o sin las recomendaciones del fabricante, la gestión automatizada e integrada con los sistemas de talleres es una solución que ayudará a los administradores llevar un registro adecuado del historial de los vehículos.

CAPITULO II: Metodología

4 Metodología

Teniendo en cuenta el problema descrito en el caso de estudio, la investigación se basa en el diseño de una arquitectura, para lo cual se utilizarán los conceptos antes vistos sobre IoT, Computación en la Nube y demás elementos profundizados en el estado del arte. El desarrollo de este diseño se llevará a cabo en cinco fases.

En la siguiente tabla se muestran los objetivos utilizados en la investigación, así como los problemas que estos pretenden solventar.

Tabla 3 Matriz de congruencia
Elaboración propia

#	Objetivos específicos	Problemas	Conceptos
1	Seleccionar un prototipo de hardware para la extracción de información del vehículo a partir de una evaluación de los prototipos existentes en el mercado.	¿De qué forma se extraerá los datos de diagnóstico y de estado del vehículo, en tiempo real?	Códigos de diagnóstico Componentes electrónicos Señales GPRS On-Board Diagnostic System
2	Diseñar y desarrollar un algoritmo de extracción para el prototipo de hardware que permita obtener la información de diagnóstico de vehículos automotores por medio del puerto OBDII.		Librerías freematics Sketch Arduino
3	Diseñar la arquitectura IoT de comunicación para el registro y consumo de la información de diagnóstico proveniente del prototipo de hardware en un repositorio central.	¿De qué forma se transferirá los datos, obtenidos por el algoritmo de extracción, desde el prototipo de hardware hacia el repositorio?	Seguridad IoT Cloud database
4	Diseñar la arquitectura en la nube de los servicios de recolección, procesamiento y revisión de datos	¿Cómo se administrarán los datos extraídos por el prototipo de hardware?	Cloud Computing Protocolos de seguridad Servicios Web – SOAP o REST Sistemas empresariales Transferencia de información

4.1 Fase de la investigación

La metodología comprende las siguientes fases para el cumplimiento de cada uno de los objetivos propuestos:

4.1.1 Fase 1

Investigación bibliográfica. Esta fase tiene por objetivo enriquecer los conocimientos y estos a la vez nos brindaran un punto de partida en el marco conceptual que se enfocara en las herramientas de IoT, sus características, proveedores que brindan servicios integrales; también información técnica de componentes que utilizan los prototipos de hardware a evaluar con el fin de comprender y tomar criterios de evaluación según los requisitos funcionales y no funcionales.

4.1.2 Fase 2

Seleccionar un prototipo de hardware para la extracción de información del vehículo. Se debe presentar un prototipo de hardware que cumpla con un mínimo de características necesarias para la recolección de información de diagnóstico del vehículo y la transferencia de estos datos hacia la API. Para cumplir con este objetivo se realizarán las siguientes actividades:

- Investigar prototipos en el mercado y realizar un modelo comparativo.
- Describir la información de los componentes que utilizan los prototipos de hardware y así determinar el que más se adapta a las necesidades de la investigación.

4.1.3 Fase 3

Diseñar y desarrollar un algoritmo de extracción para el prototipo de hardware que permita la extracción de la información de diagnóstico de vehículos automotores por medio del puerto OBDII. Este será el interlocutor lógico entre el hardware y la computadora central del vehículo y que estará programado para obtener información del vehículo y ser enviada a la API de manera que los datos puedan ser utilizados posteriormente para ellos se realizarán las siguientes actividades:

- Seleccionar el protocolo de intercambio de datos para crear la API, en base a la investigación realizada.
- Crear una estructura de datos a utilizar.
- Diseñar un modelo para el registro de datos.
- Realizar el diseño de la API en base a los requisitos funcionales y no funcionales.
- Integrar los componentes involucrados (sistemas, datos, dispositivo, protocolos).
- Crear un algoritmo de extracción de la información con pasos lógicos de extracción y envío de la información hacia la API publicada en los servicios de IoT.

4.1.4 Fase 4

Diseñar una API para la administración de los datos. La función principal de la API de administración el traslado de la información extraída del vehículo hacia un repositorio central. Para lograrlo se realizarán las siguientes actividades:

- Generar pruebas de configuración de los servicios de comunicación.

- Generar casos de uso.
- Realizar un diseño de los servicios de comunicación.

4.1.5 Fase 5

Diseñar la arquitectura IoT de comunicación para el registro y consumo de la información de diagnóstico proveniente del prototipo de hardware en un repositorio central. Para realizar este objetivo se realizarán las siguientes fases:

- Evaluarán proveedores de IoT que más se adapten a las necesidades.
- Evaluar proveedores de Cloud Computing.
- Evaluar los componentes que se necesitan para generar una arquitectura funcional que cumpla con los requisitos.
- Diseñar una base de datos que almacene de forma central los registros extraídos de los vehículos.
- Diseñar un modelo de autenticación y autorización para el acceso a la información.

Generar los diagramas necesarios, que complementen el diseño de la arquitectura IoT.

Para lograr el diseño de un sistema automatizado para la extracción y centralización de los datos de los vehículos, se requiere la ejecución de una serie de pasos en un orden lógico. Cada objetivo específico para este proyecto tiene una metodología a seguir y para ello, se cuenta con las siguientes herramientas:

Entrevistas realizadas a personal operativo y gerencia en la empresa de estudio. Estas incluyen información como:

- Descripción del proyecto.
- Requisitos funcionales y no funcionales.
- Perfiles de expertos.

4.2 Metodología para la selección del prototipo de hardware

4.2.1 Características y criterios de evaluación

Para la selección del prototipo de hardware, se cuenta con un cuadro de evaluación, elaborado específicamente para este proyecto, en el que están plasmadas las características a evaluar y el nivel de cumplimiento de cada opción evaluada, como se presenta en la tabla 2 y tabla 3, donde los puntajes van desde 1 (No cumple con el requisito) hasta 5 (Supera el nivel requerido).

Tabla 4 Tabla de puntajes
Elaboración propia

Puntaje	Descripción
1	No cumple con el requisito
2	Cumple parcialmente

3	Cumple con el mínimo requerido
4	Cumple con el estándar requerido
5	Supera el nivel requerido

Tabla 5 Matriz de evaluación
Elaboración propia

Característica	Descripción	Peso (%)	Opciones				
			1	2	3	4	5
Procesador programable	Capacidad de programar el dispositivo para modificar su funcionalidad.						
Entorno de desarrollo	Lenguaje y herramientas para la programación del dispositivo (lenguaje de programación, librerías prefabricadas, IDE de programación, etc.)						
Módulo OBDII	Componente para adaptar el dispositivo a puerto OBDII.						
Módulo Wifi	Componente para utilizar redes Wifi.						
Módulo Bluetooth	Componente para utilizar tecnología Bluetooth						
Módulo SIM	Componente para utilizar datos móviles mediante una tarjeta SIM.						
I/O externo	Puertos de entrada y salida (micro-usb, display, etc.).						
Espacio para programa	Espacio total para el programa del dispositivo, de esto depende la complejidad que pueda tener el programa.						
Espacio almacenamiento	Espacio para almacenamiento de datos, esto puede ser por memoria interna o insertando un medio externo como tarjeta microSD.						
Precio	Valor promedio del dispositivo en el mercado						
Tiempo en mercado	Tiempo de dispositivo en el mercado.						
Marca popular	Popularidad del dispositivo y/o influencia en el mercado.						
Total:		100					

4.3 Perfil para la selección de expertos

Para el cumplimiento de los objetivos se necesita el apoyo de expertos durante el transcurso de la investigación para acreditar las fases antes mencionadas.

4.4 Experto automotriz – OBDII:

- Conocimientos avanzados en automotriz.
- Experiencia de al menos 5 años en diagnóstico de vehículos utilizando dispositivos OBDII.
- Conocimientos avanzados de los protocolos que utiliza los fabricantes en sus vehículos.
- Conocimientos de datos importantes del vehículo que puede ser extraído por el dispositivo OBDII.

4.5 Experto en integración de sistemas:

- Experiencia de al menos 5 años como administrador de integración de sistemas de información.
- Conocimientos de tecnologías para las transferencias de información, JSON, SOAP, RESTful, FTP, Socket.
- Experiencia de al menos 5 años en administración y desarrollo de base de datos.
- Conocimientos en importación y exportación de datos.
- Conocimientos de optimización de procesos de bases de datos.
- Experiencia de al menos 2 años en el desarrollo de interfaces de datos.
- Experiencia en desarrollos de procesos automatizados.

4.6 Experto en IoT – Cloud Computing:

- Experiencia de al menos 5 años en tecnologías IoT.
- Experiencia de al menos 5 años en tecnologías Cloud Computing.
- Conocimientos en seguridad informática.
- Experiencia en implementación de proyectos con tecnologías IoT y Cloud Computing.

4.7 Entrevistas

Para el cumplimiento de los objetivos será necesario realizar entrevistas con los siguientes implicados en la empresa seleccionada para el caso de uso y los expertos descritos en el inciso anterior:

Operativo del negocio: por medio de la entrevista con los operativos del negocio se pretende identificar los requisitos funcionales y no funcionales, conociendo la operatividad y documentando sus procesos para su análisis y sistematización.

4.8 Técnico de taller

por medio de la entrevista con el personal técnico se dará soporte a las entrevistas realizadas con los operativos del negocio. Adicional a esto se identificarán los tipos de problemas que suceden con frecuencia y generan ineficiencia en la operatividad del negocio, especialmente para el producto en estudio.

4.9 Gerencia de TI (Tecnologías de la Información)

Con la entrevista a la gerencia de TI se pretende conocer la infraestructura actual de la organización y en base a esto realizar la propuesta de solución a nivel de infraestructura es decir un diseño basado en tecnología “on-premise” o “cloud”. Esto también ayudara a tomar en cuenta los sistemas que la organización actualmente posee para su integración con la solución propuesta.

4.10 Experto Automotriz

Esta entrevista se realizará con el objetivo de ampliar la información recolectada en la investigación acerca de los dispositivos, con la experiencia en el campo automotriz; así como también el apoyo directo en asesoría para la creación de una herramienta de evaluación para la selección del dispositivo OBDII.

4.11 Experto en Cloud computing

Esta entrevista pretende respaldar el conocimiento adquirido en la investigación de modo que la selección de la arquitectura y las tecnologías que mejor se adapten al caso de estudio.

Experto de integración de sistemas: Esta entrevista logra asesorar sobre las técnicas de integración de sistemas propuestas para la solución.

4.12 Alcances y limitaciones

4.12.1 Alcances

- El diseño de la recolección de los datos del vehículo a un almacén de datos.
- El presente trabajo se contempla la selección de un prototipo hardware para la recolección de datos.
- El estudio no contempla la implementación de la solución.
- No contempla los diseños de comunicación entre sistemas de apoyo a la operación.
- No contempla el diseño y/o implementación de los Sistemas de Información Empresarial de la organización.

4.12.2 Limitaciones

- Disponibilidad de dispositivos en el país.
- Disponibilidad de expertos locales.
- Los servicios utilizados de computación en la nube son de paga.

4.13 Entregables

A continuación, se presenta la descripción de cada uno de los entregables del proyecto:

- Estado del arte: es el resultado de la fase II de la investigación bibliográfica sobre tecnologías IoT y sistemas embebidos; los cuales proveerán las bases teóricas para la realización, tanto del prototipo, como de la arquitectura IoT y las mejores prácticas para su uso.
- Prototipo de hardware: tiene como resultado un hardware seleccionado, según criterios a investigar y analizar.
- Algoritmo de extracción: consiste en un algoritmo (sketch) de extracción de datos a ser implementado en el prototipo de hardware.
- Arquitectura IoT y computación en la nube: consiste en documentar e identificar proveedores en el mercado y comparar sus características como lo son capacidad en repositorios de datos, interfaces de comunicación, extracción de la información, y “dashboards”.
- Trabajo de investigación: es el informe final que contiene toda la investigación realizada y describe todos los componentes de los entregables.

4.14 Plan de trabajo

A continuación, se presenta el tiempo estimado para la ejecución de las actividades a realizar para el desarrollo del proyecto; tomando de referencia la fecha de inicio Junio 2018:

Tabla 6 Calendario del proyecto.
Elaboración propia

Actividad	Duración (días)
Cronograma del proyecto	101
Preparación de Tesis	19
Inicio del proyecto	1
Entrevistas con GX para el levantamiento de requisitos	15
Identificación de requisitos funcionales	3
Identificación de requisitos no funcionales	3
Identificar expertos del negocio	1
Evaluar el prototipo de hardware para obtener los datos por medio del puerto OBDII	27
Se determinará los datos que será necesario recolectar a través del prototipo de hardware, mediante entrevistas con el personal técnico de la empresa.	5
Se evaluará con expertos las características que determinan la selección del prototipo de hardware para cumplir con los requisitos del negocio.	5
Se creará una herramienta de evaluación de los dispositivos y sus características, previamente determinadas.	3
Se seleccionará una muestra intencionada de 5 prototipos de hardware, los cuales se determinarán en base al cumplimiento de las características evaluadas con expertos.	1
Se evaluarán los prototipos de hardware que cumplan con las características técnicas para el soporte del negocio, en la herramienta previamente diseñada.	3

Se realizarán pruebas con el dispositivo mejor evaluado.	15
Diseñar y crear servicios en la nube para registrar la información extraída por el prototipo de hardware	36
Realizar investigación bibliográfica sobre computación en la nube	4
Seleccionar el protocolo de intercambio de datos, en base a la investigación realizada.	2
Crear una estructura de datos a utilizar.	2
Diseñar un modelo para el registro de datos	10
Realizar el diseño de servicios base a los requisitos funcionales y no funcionales	5
Integrar los datos extraídos por el prototipo de hardware y almacenarlo en la nube	5
Crear un algoritmo de extracción del sistema de diagnóstico abordado para la recolección de la información.	15
Definir el estilo arquitectónico a utilizar para el sistema.	14
Investigación documental de los estilos de arquitectura.	10
Definir los parámetros operativos del sistema.	4
Seleccionar el modelo de servicios a utilizar.	4
Documentación de Casos de uso	15
Creación de diagramas de secuencia	5
Creación de diagramas de comunicación	2
Creación de diagramas de componentes	4
Creación de diagramas de casos de uso	6
Generación de prototipos.	10
Documentar el proyecto	5
Generar los documentos de la solución	5

CAPITULO III: *DISEÑO*

5 Diseño de la API.

En este capítulo realizará la elaboración de los diseños necesarios para el desarrollo de la arquitectura los cuales están analizados en base a los requisitos funcionales y no funcionales que se generaron en base a entrevistas con los usuarios de Gx. El diseño constara con los diagramas UML necesarios como lo son diagramas de clases, secuencia, actividades, etc. Estos describirán los componentes físicos y lógicos que son necesarios para el correcto funcionamiento de la arquitectura; desde la extracción de la información por medio del prototipo de hardware hasta el registro y consulta de la información en Cloud Computing. Cabe mencionar que la metodología a seguir para este trabajo de investigación es RUP (Relation Unified Process), esta es una metodología estándar para el análisis, implementación y documentación combinando perspectivas estáticas y dinámicas en un único diagrama (Krutchen, 2003).

6 Requisitos.

Los requisitos son la base fundamental de la arquitectura en base a estos se realiza el diseño arquitectónico para la extracción de información de vehículos por medio de un prototipo de hardware. Estos requisitos se despliegan como lista y se clasifican como requisitos funcionales y no funcionales.

6.1 Lista de requisitos de software.

A continuación, se muestran todos los requisitos funcionales y no funcionales que se recopilaron en la investigación en la empresa Gx.

Tabla 7 Lista de requisitos
Elaboración propia

Descripción del Requisito	Funcional/No funcional
Obtener información de la computadora del vehículo, relevante para GX.	Funcional
Administrar la información obtenida del vehículo.	No Funcional
Almacenar la información obtenida del vehículo, para ser consultada posteriormente.	Funcional
La información debe estar disponible en tiempo real.	No Funcional
Los servicios de consulta deben ser fácilmente accesibles para la integración en otros sistemas.	No Funcional
Las consultas deben devolver la información de forma estructurada.	Funcional

Se debe generar una estructura estándar para el manejo de la información. (JSON o XML).	Funcional
Se debe manejar seguridad para cada dispositivo con acceso al sistema por medio de una autenticación, asignando usuario y clave a cada dispositivo.	Funcional
En el caso de códigos de diagnóstico, el servicio debe devolver tanto el código, como una descripción de este.	Funcional
Tener la posibilidad de vincular los dispositivos (prototipos de hardware) a empresas o flotas.	No Funcional
Cada dispositivo deberá tener “Plug&Play”	Funcional
Prototipo de hardware que sea capaz de leer información de los vehículos por medio del puerto OBD	Funcional
Diseñar una arquitectura basada en cloud computing	No Funcional
Diseñar servicios web para el consumo de información desde los sistemas empresariales u otro canal autorizado.	Funcional
Los servicios deben estar disponible a todo momento	No Funcional
Debe ser escalable	No Funcional
El prototipo de hardware debe ser portátil	No Funcional
La conexión debe ser directa entre el vehículo y el repositorio central y no por medio de intermediarios	No Funcional
La transmisión de los datos debe pasar por medio de protocolos seguros	Funcional

7 Selección de hardware para la extracción de la información

7.1 Matriz de evaluación de prototipos de hardware

Para este proyecto se ha planteado la necesidad de la evaluación de dispositivos en el mercado que cumplan con los requisitos descritos en la matriz de evaluación, presentada entre las herramientas a utilizar, y se le dará una puntuación a cada característica del dispositivo, basado en la tabla de puntajes presentada en la misma sección.

Para esta evaluación se tomaron los siguientes dispositivos a evaluar:

- **Opción 1:** Freematics One +
- **Opción 2:** Freematics One
- **Opción 3:** OBDLink MX+
- **Opción 4:** IDS Home

- **Opción 5:** BlueDriver LSB2

Tabla 8 Matriz comparativa de prototipos en el mercado
Elaboración propia

Característica	Descripción	Peso (%)	Opciones				
			1	2	3	4	5
Procesador programable	Capacidad de programar el dispositivo para modificar su funcionalidad.	15	4	4	2	3	1
Entorno de desarrollo	Lenguaje y herramientas para la programación del dispositivo (lenguaje de programación, librerías prefabricadas, IDE de programación, etc.)	5	4	4	4	4	1
Módulo OBDII	Componente para adaptar el dispositivo a puerto OBDII.	15	5	5	5	5	5
Módulo Wifi	Componente para utilizar redes Wifi.	5	5	3	1	3	3
Módulo Bluetooth	Componente para utilizar tecnología Bluetooth	5	5	3	5	5	5
Módulo SIM	Componente para utilizar datos móviles mediante una tarjeta SIM.	8	5	4	1	4	5
I/O externo	Puertos de entrada y salida (micro-usb, display, etc.).	7	4	4	3	3	4
Espacio para programa	Espacio total para el programa del dispositivo, de esto depende la complejidad que pueda tener el programa.	9	4	3	2	3	1
Espacio almacenamiento	Espacio para almacenamiento de datos, esto puede ser por memoria interna o insertando un medio externo como tarjeta microSD.	5	3	3	4	3	5
Precio	Valor promedio del dispositivo en el mercado	10	1	4	1	2	1
Tiempo en mercado	Tiempo de dispositivo en el mercado.	8	3	4	3	4	4
Marca popular	Popularidad del dispositivo y/o influencia en el mercado.	8	4	4	3	4	4
Total:		100	3.90	3.91	2.80	3.59	3.11

La fórmula utilizada para darle puntaje a cada dispositivo fue la siguiente:

$$\frac{\sum(Puntaje\ Característica) * (Peso\ Característica)}{100} = Puntaje\ Total$$

En base a las necesidades del negocio y los requisitos listados, se seleccionará como prototipo de hardware “Freematics ONE” ya que cubre las necesidades y soporta otros módulos inalámbricos para la transmisión de datos que se pueden adaptar para las necesidades de otras organizaciones con requisitos similares al caso de estudio.

8 Diseño de algoritmo para la extracción de la información

Esta fase de diseño tiene como objetivo documentar y describir por medio de diagramas UML la capa del algoritmo de extracción, entiéndase como el programa que ejecutara la secuencia de sentencias en orden para poder extraer la información del vehículo y posteriormente enviarla al almacén de datos en la nube por medio de servicios web.

8.1 Diagrama de clase

El algoritmo de extracción y transmisión de datos hacia el servicio web en la nube, está compuesto por clases de inicialización de configuración que permite al prototipo de hardware funcione correctamente esto contiene procesos de verificación de memoria, conexión a la computadora central del vehículo, entre otros.

La extracción y transmisión de la información es por medio de una clase que realiza consultas directas al vehículo (sistema OBD) por medio de PID y posteriormente se inicializa la segunda clase de transmisión para enviar la información a los servicios en la nube.

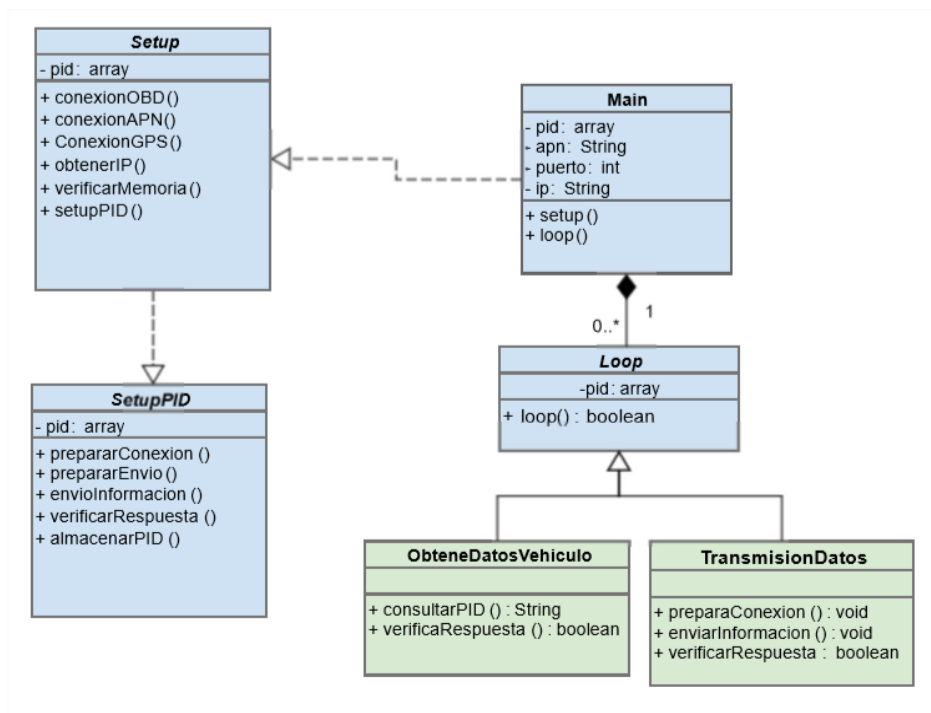


Ilustración 9 Diagrama de clases del algoritmo de extracción.
Elaboración propia

8.2 Diagrama de componentes

Los componentes que se verán involucrados en el sketch o algoritmo de extracción serán los listados:

- **Prototipo de hardware:** prototipo de hardware seleccionado con base a las necesidades de extraer la información del vehículo y que sea capaz de abrir y enviar peticiones por medio del protocolo http.
- **Cloud Computing:** se refiere a todos los componentes necesarios para poder disponer de servicios web y/o servicios IoT intercomunicado con el prototipo de hardware.
- **Librerías:** hace referencia a las librerías utilizadas en el sketch para la extracción de la información del vehículo.
- **Puerto OBD:** este es el punto de conexión del prototipo de hardware al vehículo normalmente están situados en la parte baja del volante, es decir bajo la consola del vehículo.
- **Computadora central:** este componente es el que centraliza toda la información desde el sistema OBD del vehículo.
- **Algoritmo:** también conocido como sketch es el que de forma lógica y secuencia extrae y envía la información del vehículo.

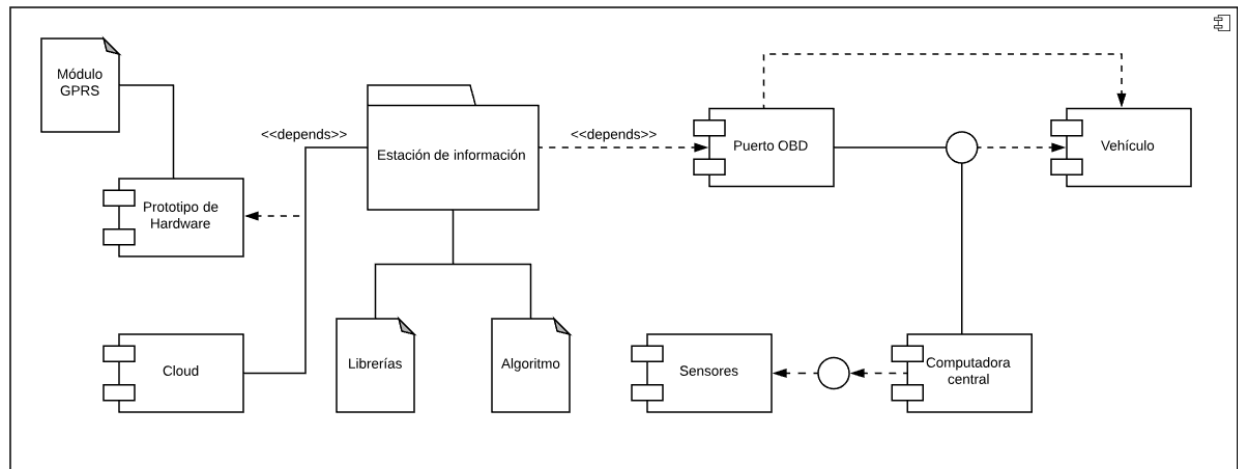


Ilustración 10 Diagrama de componentes de algoritmo de extracción
Elaboración propia

8.3 Diagrama de secuencia

El diagrama de secuencia define los pasos a seguir en el algoritmo de extracción desde el prototipo de hardware, por medio del puerto OBDII del vehículo. Como se presenta en el diagrama todo inicia desde el encendido del dispositivo, en este caso se asume que el prototipo está conectado al vehículo en el puerto OBDII correctamente. El encendido del vehículo es el que inicializa el algoritmo entrando en un proceso obligatorio de configuración y pruebas de comunicación; una vez el dispositivo se encuentra listo, se procede a obtener los PID que son necesarios para la extracción de la información; esto con el objetivo de extraer información únicamente necesaria para el análisis de información de Gx y no generar información “basura” en el almacén de datos. Teniendo la lista de PID disponibles, se procede a consultar estos directamente al sistema OBDII del vehículo, para luego transmitir la información obtenida al flujo de procesamiento de información en la nube.

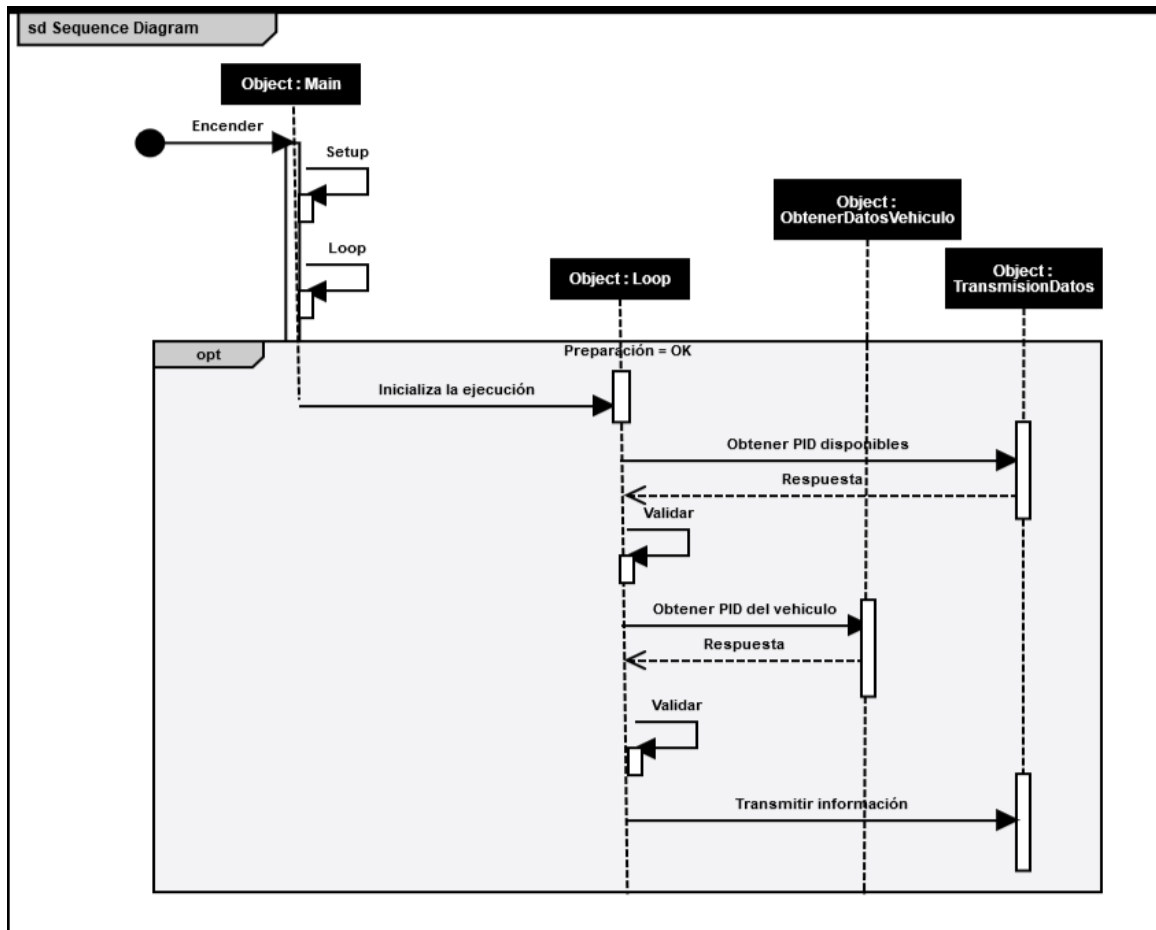


Ilustración 11 Diagrama de secuencia de algoritmo de extracción

Elaboración propia

8.4 Diagrama de Estado

En el presente diagrama se muestran los estados en que el algoritmo de extracción permanece desde que el prototipo es conectado al vehículo, este inicializa los estados de memoria, conexión al puerto OBDII, y conexión a las redes inalámbricas GPRS; esto es necesario para que posteriormente entre a un estado de recolección y envío de información, y se mantiene así hasta tener cambio de estado; dicho de otra forma el algoritmo no puede comenzar o continuar su flujo mientras los estados cargados inicialmente no estén listos.

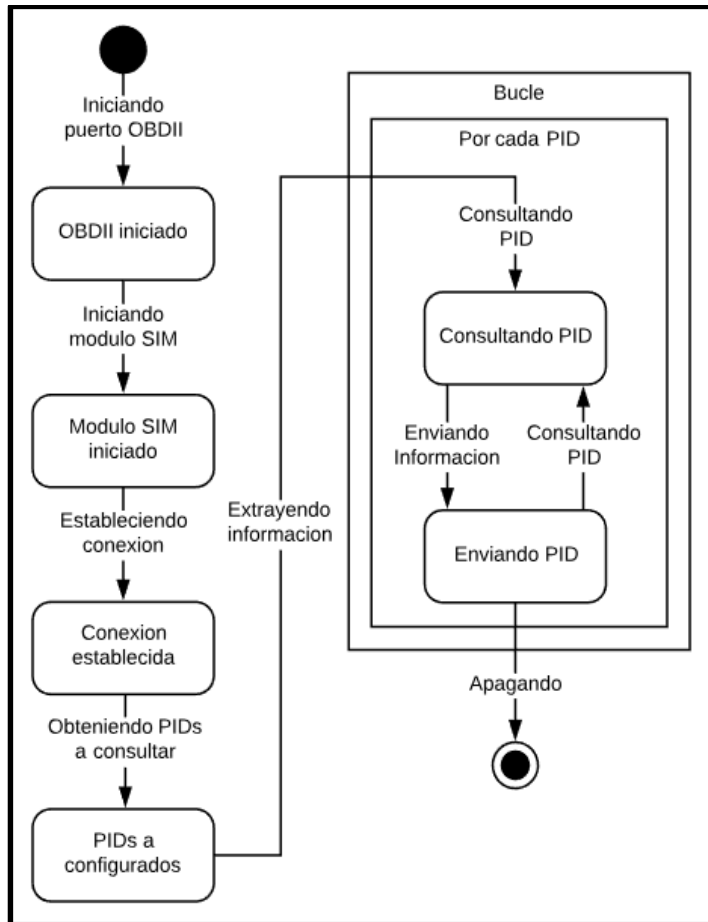


Ilustración 12 Diagrama de estado de algoritmo de extracción

Elaboración propia

8.5 Diagrama de actividades

Las actividades para realizar en el algoritmo de extracción de datos se dividen principalmente en dos grandes funcionalidades.

- Preparación de módulos:** se refiere a preparar los estados de memoria, conexiones al puerto OBDII y garantizar que las conexiones inalámbricas GPRS están disponibles. A todo esto, le llamamos configuración inicial, posteriormente se tiene la validación de los estados para verificar que todo esté listo, y proseguir con la actividad de extracción de los datos.
- Extracción de datos:** son actividades invocadas constantemente mediante un bucle condicionado por los estados de configuración inicial, para garantizar que no exista un cambio durante la extracción de los PID y/o el conjunto de PIDs haya terminado de procesarse por medio de las actividades de consulta de PID, validación de los datos y por último enviar la información a un servicio web.

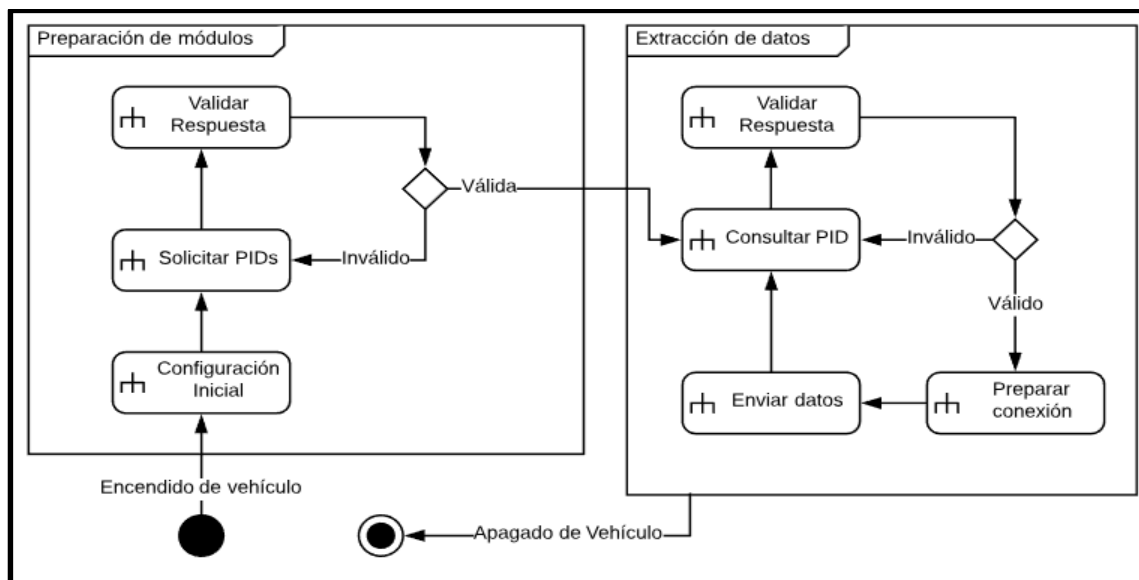


Ilustración 13 Diagrama de actividades de algoritmo de extracción

Elaboración propia

8.6 Descripción de librerías y funciones utilizadas.

El desarrollo del algoritmo de extracción de información del OBD está basada en las librerías de freemantics las cuales utilizan diferentes funciones para la conexión a redes inalámbricas, conexión a la computadora central del vehículo por medio de PID predefinidos de forma estándar para obtener la información de sensores del vehículo y funciones de envío de información hacia servicios web.

9 Diseño de Arquitectura de servicios en la nube para la recepción y almacenamiento de la información.

Este diseño es otra fase del proyecto que involucra todos aquellos componentes en la nube y se integran con el prototipo de hardware, por medio de servicios de computación en la Nube. A continuación, se presenta un diagrama representando los componentes de Amazon AWS que para este trabajo de investigación se ha utilizado como parte del diseño de la arquitectura.

9.1 Descripción de Servicios

- **API Gateway:** es un servicio completamente administrado que facilita a los desarrolladores la creación, la publicación, el mantenimiento, el monitoreo y la protección de API a cualquier escala. Con tan solo unos clics en la consola de administración de AWS, puede crear API REST y API WebSocket que actúen como "puerta delantera" para que las aplicaciones obtengan acceso a datos, lógica de negocio o funcionalidades desde sus servicios backend.
- **Kinesis Firehose:** es un servicio que ofrece una manera sencilla de cargar datos de streaming de manera fiable en almacenes de datos y herramientas de análisis.

- **AWS S3:** es un servicio de almacenamiento de objetos creado para almacenar y recuperar cualquier volumen de datos desde cualquier ubicación: sitios web y aplicaciones móviles, aplicaciones corporativas y datos de sensores o dispositivos von IoT.
- **Kinesis Data Analytics:** ofrece una manera sencilla de procesar datos de streaming en tiempo real con SQL estándar sin tener que aprender a usar lenguajes de programación ni marcos de procesamiento nuevos.
- **Kinesis Data Stream:** es un servicio de streaming de datos en tiempo real con un alto nivel de escalabilidad y durabilidad. Puede registrar de manera continua gigabytes de datos por segundo de cientos de miles de orígenes, como transmisiones de clics de sitios web, transmisiones de eventos de bases de datos, transacciones financieras, fuentes de redes sociales, registros de TI y eventos de seguimiento de ubicaciones. Los datos recopilados se encuentran disponibles en milisegundos para posibilitar los casos de uso de análisis en tiempo real, como paneles en tiempo real, detección de anomalías en tiempo real y precios dinámicos, entre otros.
- **Dynamo DB:** es una base de datos de documentos y valores clave que ofrece un rendimiento en milisegundos de un solo dígito a cualquier escala. Se trata de una base de datos completamente administrada, en varias regiones y multimaestra con seguridad, copias de seguridad y restauración integradas y almacenamiento en memoria caché de aplicaciones a escala de Internet.
- **AWS Lambda:** permite ejecutar código sin aprovisionar ni administrar servidores. Se paga por el tiempo informático que consuma. Sin costo cuando el código no es ejecutado. (Amazon Web Services, 2018)

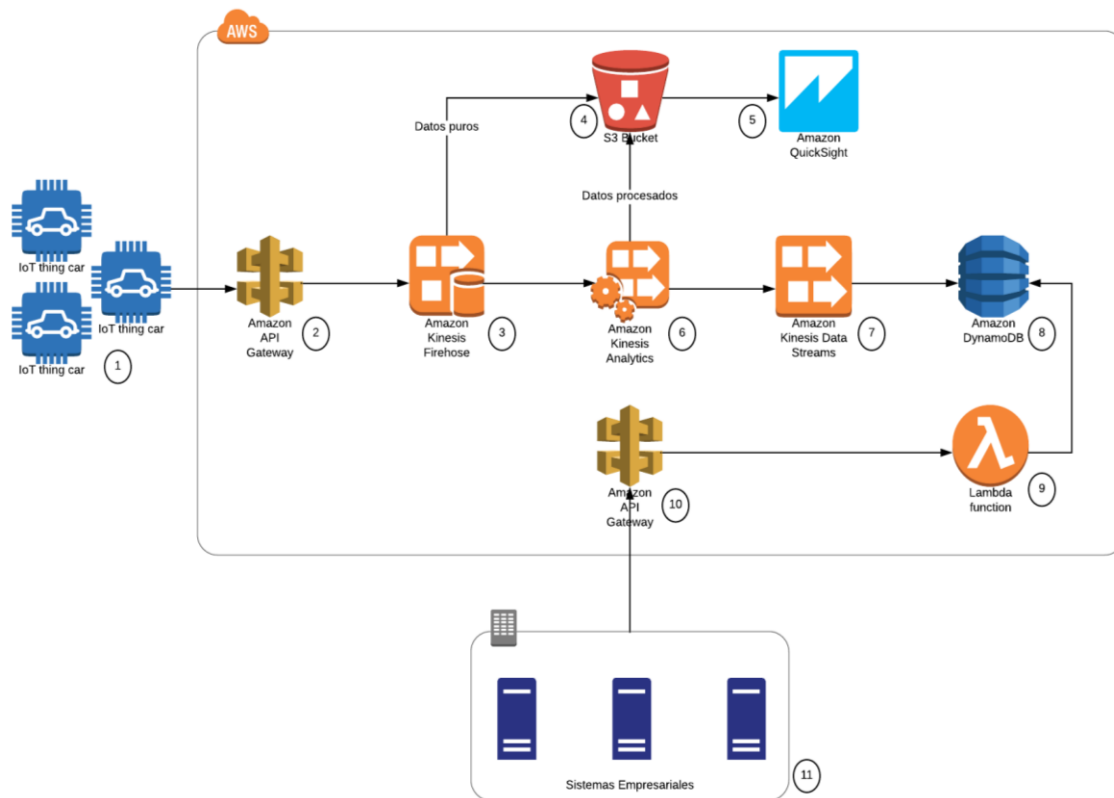


Ilustración 14 Diagrama de componentes de Cloud Computing
Elaboración propia

El flujo de datos a través de los diferentes servicios de AWS, tal como se muestra en la ilustración, comienza por la extracción de los datos del vehículo por medio del prototipo de hardware (1), siendo este el único elemento externo a la infraestructura en la nube. Los datos extraídos son enviados a la API REST (2), punto de entrada de los datos hacia el sistema. Una vez los datos ingresan a la API, son enviados al servicio de streaming de alta disponibilidad (3), donde son guardados en el almacén (4) y enviados al servicio de streaming de procesamiento de datos en tiempo real (6). Cuando los datos han sido procesados, son enviados al servicio de streaming (7) que los inserta en la base de datos (8).

El flujo previamente descrito permite tener disponibles los datos procesados y consultarlos en cualquier momento. Los datos pueden ser extraídos por medio de la API REST (10), la cual utiliza funciones de extracción (9) para obtener la información de la base de datos (8). La API REST (10) permite tener la información e integrarla a los sistemas empresariales existentes (11).

9.2 Diagrama de servicios expuesto para el envío de la información

En la arquitectura propuesta en la nube se definen servicios de consulta de información previamente registrada en el almacén de datos y otro conjunto de servicios que brindaran el registro de la información de los vehículos que fue extraída, es decir este último servicio será el canal de entrada de la información que el prototipo de hardware extraerá directamente del vehículo.

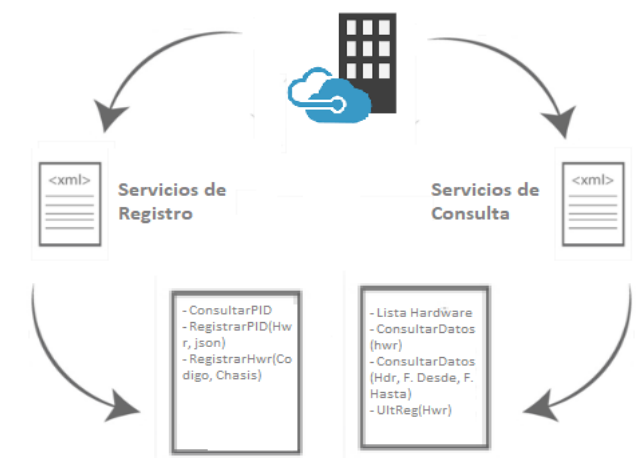


Ilustración 15 Diagrama de servicios de cloud computing
Elaboración propia

9.3 Comunicación con la API

La comunicación entre todos los componentes que conforman la arquitectura IoT será por medio de servicios web, teniendo como punto de partida la extracción de la información del vehículo por medio del prototipo de hardware que estará conectado directamente al puerto OBDII del vehículo y que este tiene la funcionalidad de transmitir información vía GPRS por medio de protocolo HTTP. En la capa de cloud computing se mantendrán activos servicios de registros de información que se definieron

en el diagrama de servicios visto previamente en la ilustración 12. Este servicio registrará cada petición realizada en el almacén de datos en la nube.

Con la información almacenada los sistemas empresariales de Gx u otros sistemas que Gx autorice podrán acceder a la información registrada por medio de los servicios web de consulta para el análisis o procesamiento de la información obtenida por la arquitectura IoT.

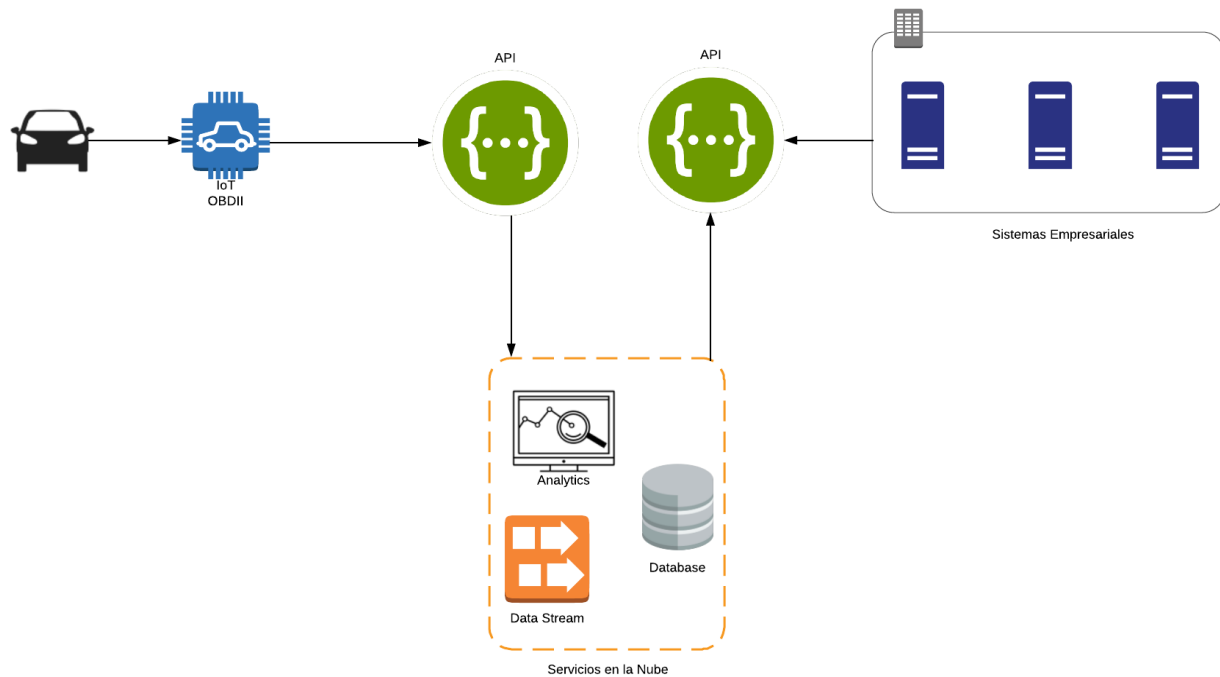


Ilustración 16 Diagrama de comunicación entre el prototipo de hardware y cloud computing
Elaboración propia

9.4 Diagrama de componentes

La arquitectura en Cloud Computing tiene componentes propios de su solución estos se comunican entre sí para poder brindar el servicio al prototipo de hardware.

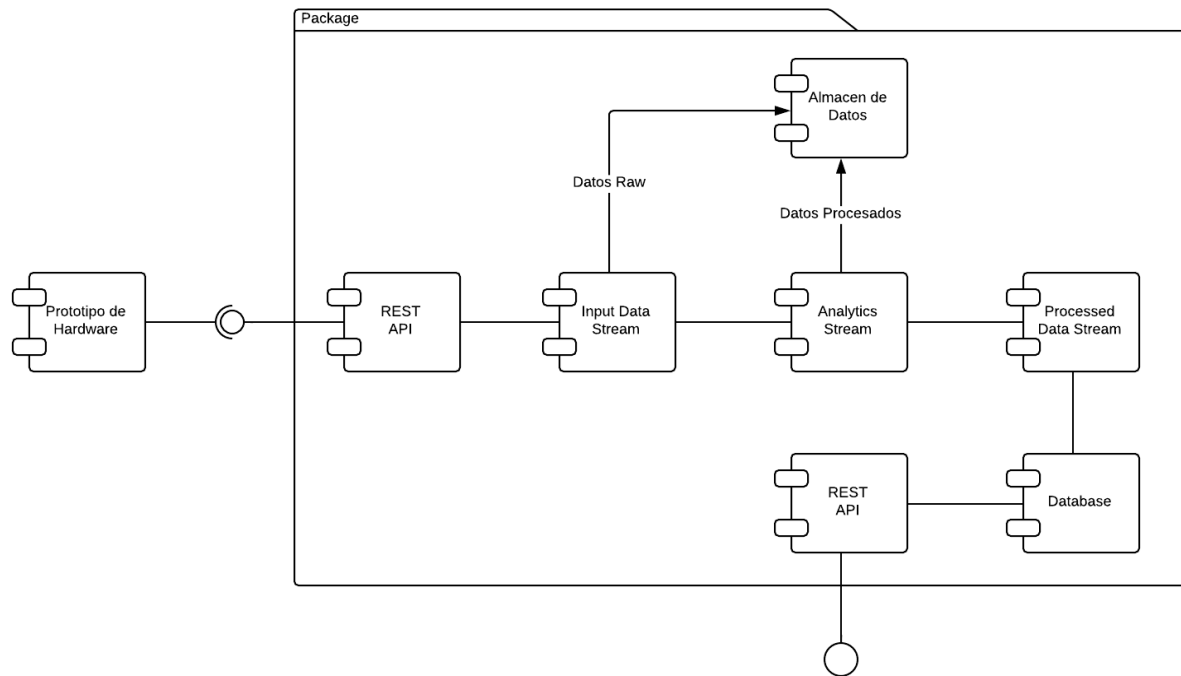


Ilustración 17 Diagrama de componentes en Cloud Computing
Elaboración propia

9.5 Diagrama de actividades

En el presente diagrama se muestra la secuencia de actividades a realizar para al momento de generar un registro de información previamente extraída del vehículo; hay que recordar que este es un servicio web por lo que existe un componente “listener” que tomara todas las peticiones realizadas por el prototipo de hardware a través de protocolos http; teniendo en cuenta lo anterior se procede a validar la información enviada y en caso la información no sea correcta se enviara a un proceso de registro de la petición con error, describiendo el error generado y terminando la actividad en caso contrario es decir que la información sea la correcta se registra la información en el almacén de datos y se termina la actividad.

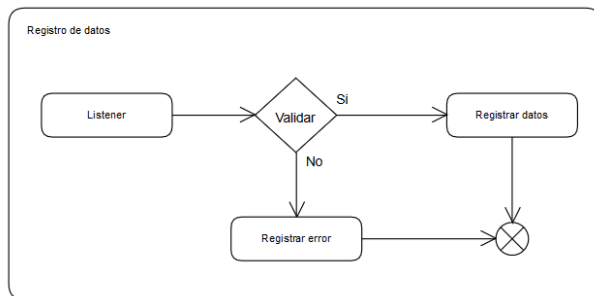


Ilustración 18 Diagrama de actividades para el registro de datos
Elaboración propia

El siguiente diagrama de actividades hace referencia a la secuencia que realiza cada actividad al momento de realizar consultas al servicio web que estará disponible para el análisis de la información con herramientas externas o sistemas de información que se integran a la arquitectura para la continuidad del negocio.

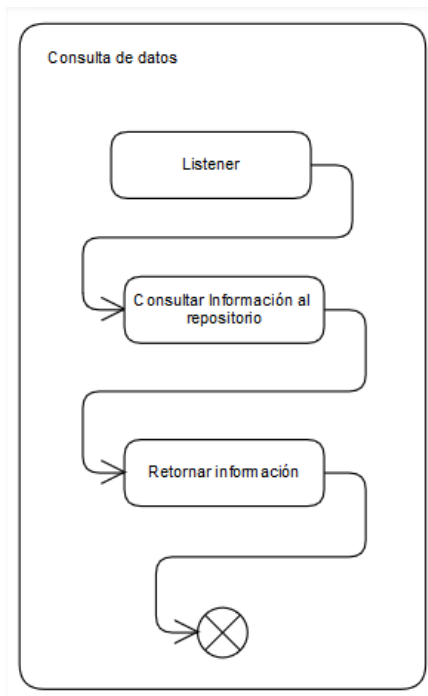


Ilustración 19 Diagrama de actividades para la consulta de información
Elaboración propia

El siguiente diagrama de actividades hace referencia a la secuencia de actividades que se realizan al momento de consultar los códigos de diagnóstico del vehículo.

10 Requisitos mínimos

- Prototipo de hardware
- Arduino Studio
- Librerías Freematics
- Driver de Freematics One
- Chip SIM con acceso a internet por medio de redes GPRS
- Algoritmo de extracción de la información
- Cuenta activa con proveedor de servicios en la nube
- Creación y activación de objetos en la nube
- Vehículo automotor con sistema OBDII

11 Resultados

Esta etapa se muestran los resultados de las pruebas realizadas por el equipo con el prototipo seleccionado y realizando extracciones directas de vehículos imprimiendo en consola los resultados como las primeras pruebas y luego evolucionando con pruebas piloto con toda la arquitectura propuesta funcionando.

11.1 Pruebas iniciales: Algoritmo de extracción

Luego de realizar los desarrollos correspondientes según el diseño planteado se obtuvieron resultados satisfactorios. En las siguientes imágenes se muestran los pasos que se realizaron para cargar el algoritmo de extracción en el prototipo de hardware por medio de “Arduino Studio” utilizando puerto USB.



Ilustración 20 Carga del algoritmo de extracción en el prototipo de hardware (Puerto USB).
Elaboración propia

Utilizando Arduino Studio se realiza la carga del sketch de forma satisfactoria

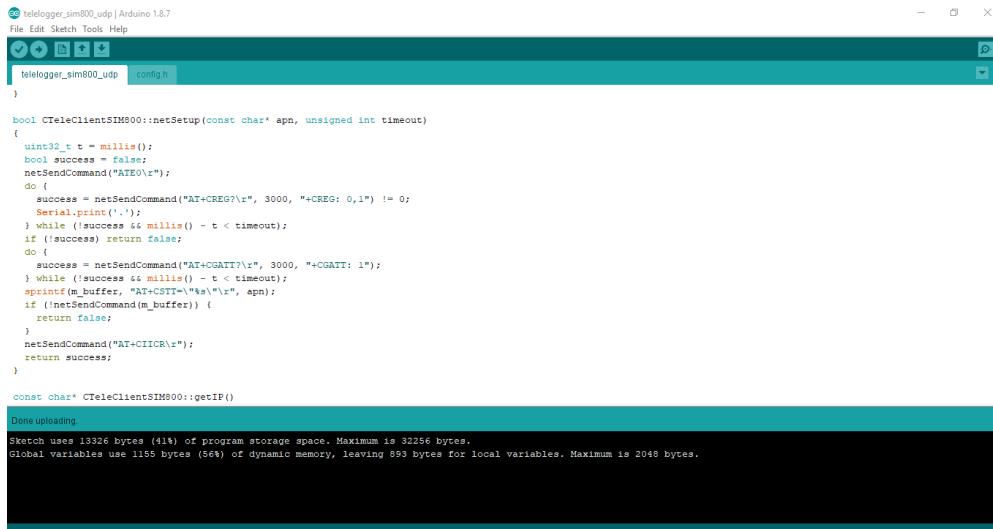


Ilustración 21 Carga del algoritmo de extracción en el prototipo de hardware (Sketch).
Elaboración propia

A continuación, se muestra la imagen donde se conecta el prototipo de hardware en el puerto OBD de un vehículo; este comienza a funcionar y enviar información la nube al instante de conectarse en dicho puerto. En la imagen se demuestra que sigue conectado por medio de un cable micro USB hacia una computadora esto con el propósito de monitorear los datos extraídos desde el vehículo por medio de monitor que Arduino Studio nos proporciona.



Ilustración 22 Carga del algoritmo de extracción en el prototipo de hardware.
Elaboración propia

Resultado de los datos que se extraen por medio del puerto OBD y se envía directamente a la nube utilizando las redes GPRS del chip previamente configurado en el prototipo de hardware.

```
COM3
Freematrix ONE
OBD...NO
Standby
.OBD...OK
GPS...NO
NEMS...OK
SIM900...OK
GPRS.....OK
IP:10.248.195.102
LOGIN...O
080=64739,VIN=00001_TRACKER,TS=64739,30=0,10D=0,10C=942,104=25,111=13,10F=43,20=-91;4:32,82=39,24=1450,
[0] 98B sent
080=65380,VIN=00001_TRACKER,TS=65380,30=0,10D=0,10C=946,104=24,111=13,20=-93;3:30,82=39,24=1450,4=1450*7
[1] 98B sent
080=65922,VIN=00001_TRACKER,TS=65922,30=0,10D=0,10C=939,104=24,111=13,20=-92;5:30,82=39,24=1450,84
[2] 98B sent
080=66462,VIN=00001_TRACKER,TS=66462,30=0,10D=0,10C=944,104=24,111=13,20=-93;4:31,82=39,24=1450,88
[3] 98B sent
080=67188,VIN=00001_TRACKER,TS=67188,30=0,10D=0,10C=945,104=24,111=13,20=-92;4:31,82=40,24=1450,88
[4] 98B sent
080=67732,VIN=00001_TRACKER,TS=67732,30=0,10D=0,10C=941,104=24,111=13,20=-91;3:30,82=39,24=1450,8C
[5] 98B sent
080=68275,VIN=00001_TRACKER,TS=68275,30=0,10D=0,10C=942,104=24,111=13,20=-91;4:31,82=40,24=1460,83
[6] 98B sent
080=68814,VIN=00001_TRACKER,TS=68814,30=0,10D=0,10C=943,104=23,111=13,20=-91;4:31,82=40,24=1450,85
[7] Unsent
080=74139,VIN=00001_TRACKER,TS=74139,30=0,10D=0,10C=923,104=24,111=13,20=-92;3:29,82=40,24=1460,82
[7] 98B sent
080=74684,VIN=00001_TRACKER,TS=74684,30=0,10D=0,10C=920,104=24,111=13,20=-92;3:32,82=39,24=1450,83
[8] 98B sent
080=75229,VIN=00001_TRACKER,TS=75229,30=0,10D=0,10C=917,104=24,111=13,20=-92;3:31,82=40,24=1450,8B
[9] 98B sent
080=75811,VIN=00001_TRACKER,TS=75811,30=0,10D=0,10C=919,104=24,111=13,20=-94;3:32,82=39,24=1460,80
[10] 98B sent
080=76409,VIN=00001_TRACKER,TS=76409,30=0,10D=0,10C=932,104=24,111=13,20=-93;5:32,82=40,24=1450,88
[11] 98B sent
```

Ilustración 23 Log consola de arduino studio
Elaboración propia

11.2 Pruebas Iniciales: Computación en la Nube

Para las pruebas iniciales en la nube se utilizó AWS como proveedor, una de las principales razones fue que gracias a su capa gratuita se pudo generar un prototipo funcional de la arquitectura sin costo. Por otra parte, se utilizó el paradigma de “Infraestructura como Código” por medio de “Serverless Framework”, el cual, como su nombre lo indica, permite utilizar código para definir la infraestructura a utilizar en la nube; esto permite la replicación casi inmediata de la infraestructura necesaria para este proyecto.

A continuación, se describen brevemente los pasos para desplegar el proyecto:

- Crear cuenta en AWS

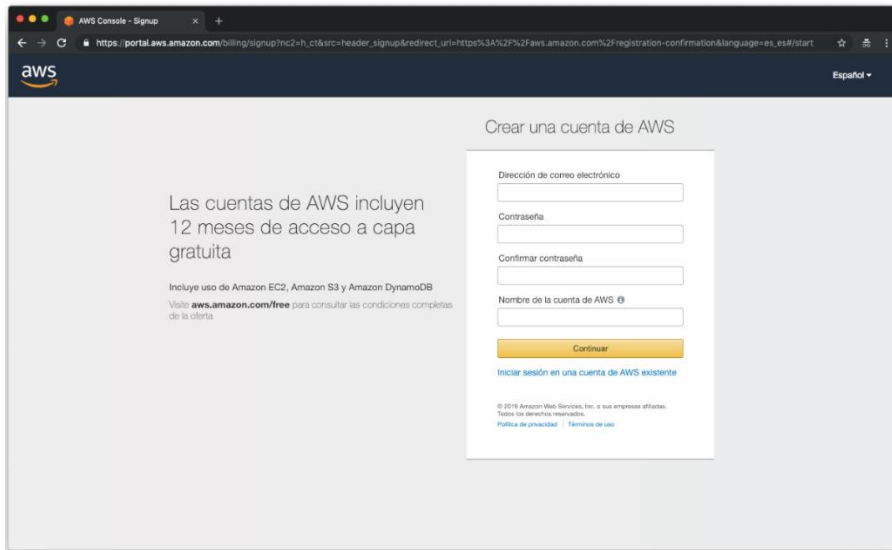


Ilustración 24 Formulario de cuenta AWS
Referencia AWS

- Descargar AWS CLI

```
FerDuran — -bash — 80x24
MacBook-Air-de-Fernando:~ FerDuran$ sudo pip install awscli
```

Ilustración 25 Comando de instalación de Consola AWS
Elaboración propia

- Ingresar las credenciales de usuario AWS en **AWS CLI**

```
FerDuran — -bash — 80x24
MacBook-Air-de-Fernando:~ FerDuran$ aws configure
AWS Access Key ID [*****X3VA]:
AWS Secret Access Key [*****0GYv]:
Default region name [None]:
Default output format [None]:
```

Ilustración 26 Comando de configuración de credenciales AWS
Elaboración propia

- Instalar **Serverless Framework**

```
FerDuran — -bash — 80x24
MacBook-Air-de-Fernando:~ FerDuran$ npm install serverless -g
```

Ilustración 27 Comando de instalación de Serverless Framework
Elaboración propia

- Abrir el proyecto *serverless-iot-obd*

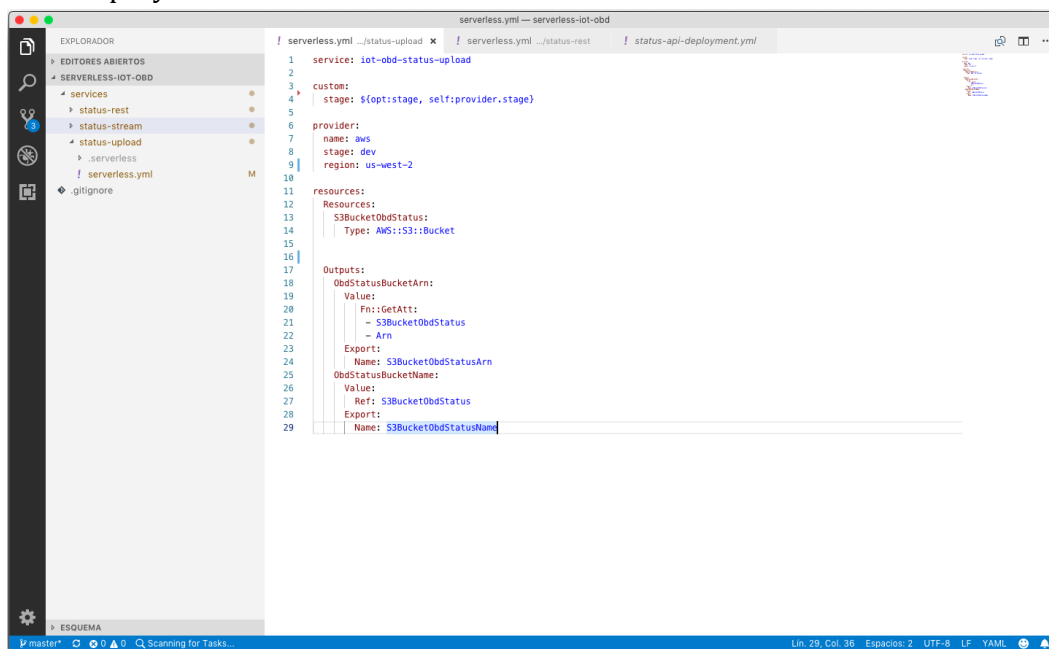


Ilustración 28 Proyecto Serverless
Elaboración propia

- Desplegar el proyecto por medio del Serverless Framework

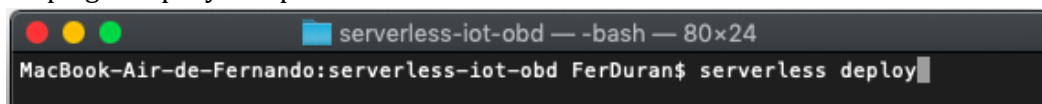


Ilustración 29 Comando de despliegue de proyecto Serverless en AWS
Elaboración propia

12 Conclusiones

En esta investigación se ha presentado una propuesta a partir de un caso de estudio que tiene la necesidad de automatizar los procesos de recolección de información de los vehículos, por lo tanto, se ha realizado el diseño de una arquitectura basada en tecnologías IoT y computación en la nube, que incluye la extracción de información de los vehículos por medio del sistema OBDII en tiempo cercano al real y que a su vez dicha información es registrada en la nube. Esta propuesta contribuye a superar cada objetivo descrito al inicio de la investigación.

Basado en el objetivo “selección de un prototipo de hardware para la extracción de información del vehículo, a partir de una evaluación de los prototipos existentes en el mercado”, se seleccionó un prototipo de hardware según la evaluación realizada a múltiples prototipos, tomando en cuenta las características requeridas a cumplir.

Basado en el objetivo “diseñar un algoritmo de extracción para el prototipo de hardware que permita obtener la información de diagnóstico de vehículos automotores por medio del puerto OBDII”, se diseñó el algoritmo para recolectar la información generada por los vehículos automotores para posteriormente enviar la información a servicios en la nube.

Basado en los objetivos “diseñar la arquitectura IoT de comunicación para el registro y consumo de la información de diagnóstico proveniente del prototipo de hardware en un almacén de datos” y “diseñar la arquitectura en la nube de los servicios de recolección, procesamiento y revisión de datos”, se diseñó la arquitectura en la nube basada en los servicios de Amazon AWS para la recolección de los datos, exponiendo servicios de recopilación, transformación, procesamiento y almacenamiento de la información proveniente de los prototipos de hardware instalados en los vehículos.

Además, se realizó la documentación del proceso de selección del prototipo de hardware y la documentación del diseño de la arquitectura propuesta basado en UML.

13 Recomendaciones

Para la mejora de lo propuesto en la investigación, se recomienda lo siguiente:

- Mejorar el prototipo para bajar los costos de fabricación, para esto se debe realizar una investigación de los elementos y técnicas de construcción del dispositivo.
- Fortalecer la comunicación entre los servicios utilizando un dinamismo más personalizado para la extracción de valores PID.
- Implementar protocolos de seguridad.

14 Bibliografía

Internet de las cosas - Breve reseña, octubre 2015, Accedido 09-01-2019: <https://www.internetsociety.org/wp-content/uploads/2017/09/report-InternetOfThings-20160817-es-1.pdf>

Amazon Web Services, 2019, AWS. Obtenido de: <https://aws.amazon.com/>

Google Cloud, 2019, Google. Obtenido de: <https://cloud.google.com/solutions/iot-overview>

Microsoft Azure, 2018, Microsoft. Obtenido de: <https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-introduction>

ImayaTech, 2018. Obtenido de: <http://www.imayatech.com/it-management/iot.html>

Forbes 2014, accedido 09-01-2019: <https://www.forbes.com/sites/gilpress/2014/06/18/a-very-short-history-of-the-internet-of-things/>

Amidata S.A., accedido 27-10-2017: <http://es.rs-online.com/web/generalDisplay.html?id=i/iot-internet-of-things>

IBM Think Academy, (2015), accedido 27-10-2017 <https://www.youtube.com/watch?v=QSIPNhOiMoE>

Jorge Alvarado, Red Hat México, El Impacto Real del IoT en las Empresas, accedido 27-10-2017: <https://sg.com.mx/revista/51/el-impacto-real-del-iot-las-empresas#.WfQJgmjWzDc>

ERP Software Market by Deployment (On-premise deployment and Cloud deployment) and Function (Finance, Human resource, Supply chain and others) - Global Opportunity Analysis and Industry Forecast, 2013 – 2020. Shrikant Chaudhari and Akshay Ghone, ALLIED MARKET RESEARCH, Marzo 2015

IDC's Worldwide Semiannual IT Spending Guide, International Corporation Data, 2016

IoT Enabled Enterprise Resource Planning (ERP): Market Outlook and Forecasts 2017 – 2022, Research and Market, Enero 2017

Arvon L. Mitcham, 2000, <https://nepis.epa.gov/Exe/ZyPdf.cgi?Dockkey=P1002KO4.pdf>

http://reedlatam.com/sadmoweb/files/modulos/ConferenciasTalleres/infosecurity/2017/workshops-salon-2/presentacion/presentacion_nyce_.pdf

Marco Arenas, 2016, https://www.owasp.org/images/3/36/IoT_CyberSecurity.pdf

Beka Kezherashvili, 2011, <https://www.us-cert.gov/sites/default/files/publications/CloudComputingHuthCebula.pdf>

Alexa Huth, James Cebula, 2011 Accedido el 2019-01-09 <https://www.us-cert.gov/sites/default/files/publications/CloudComputingHuthCebula.pdf>

ALBORS, J. (2014). Amenazas IoT. Obtenido de <https://www.welivesecurity.com/la-es/2014/09/12/amenazas-iot-dispositivos-no-imaginabas-conectados/>

AutoTap. (19 de 01 de 2018). AutoTap. Obtenido de obdii: <http://www.obdii.com/connector.html>

Bahga, A., & Madisett, V. (2014). Internet of Things (A Hands-on-Approach). VPT 1st Ed.

Basallo, Y. A., Díaz Estrada, A., & González Gómez, S. (2009). Una experiencia en integración de aplicaciones empresariales. Carretera a San Antonio de los Baños. km 2 ½, Torrens, Boyeros. Ciudad de La Habana. Cuba.: Universidad de las Ciencias Informáticas. Departamento de Seguridad Informática.

Blasco, V. (19 de 01 de 2018). SISTEMA DE DIAGNOSTICO OBD II. Obtenido de http://www.electronicar.net/IMG/Articulo_OBDII.pdf

Concepcion, M. (2011). OBD-II Repair Strategies: (Including State Inspections). CreateSpace Independent Publishing Platform.

Evans, D. (2011). Informe Técnico, Internet of Things: La próxima evolución de Internet lo está cambiando todo. Cisco.

Gemalto. (22 de 11 de 2017). www.gemalto.com. Obtenido de <http://www.gemalto.com/latam/iot/seguridad-en-iot>

Hougland, B. (2014). TEDx. Obtenido de https://www.youtube.com/watch?v=_AlcRoqS65E&t=627s

Ley de Arrendamiento Financiero. (20 de junio de 2002). Diario Oficial No. 126, Tomo 356 de fecha 9 de julio de 2002. El Salvador: Asamblea Legislativa de la República de El Salvador.

Mell, P., & Timothy, G. (2011). The NIST Definition of Cloud Computing. Gaithersburg: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

Oracle. (19 de 01 de 2018). Oracle Corp. Obtenido de [dosc.oracle.com: https://docs.oracle.com/javase/6/tutorial/doc/gijvh.html](https://docs.oracle.com/javase/6/tutorial/doc/gijvh.html)

Pérez, R. (19 de 01 de 2015). Trabajo de Grado: Desarrollo de prototipo de sensor IoT usando la red SigFox. Obtenido de Departamento de Ingeniería Telemática, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla: <http://bibing.us.es/proyectos/abreproy/90269/fichero/tfgRamonPerezHernandez.pdf>

Propuesta de un plan de mantenimiento automotriz para la flota vehicular. (19 de 01 de 2012). Obtenido de [dspace.ups.edu.ec: https://dspace.ups.edu.ec/bitstream/123456789/1936/12/UPS-CT002335.pdf](https://dspace.ups.edu.ec/bitstream/123456789/1936/12/UPS-CT002335.pdf)

Richards, M. (2015). Software Architectre Patterns. O'Reilly Media.

Rountree, D., & Castrillo, I. (2014). The Basics of Cloud Computing, Understanding the Fundamentals of Cloud Computing in Theory and Practice. Syngress.

Santos, J., & Costas. (2014). Seguridad informática. ProQuest Ebook Central: RA-MA Editorial.

SAP. (19 de 01 de 2018). SAP Company. Obtenido de <https://www.sap.com/products/enterprise-management-erp.html>

W3C. (19 de 01 de 2018). W3C. Obtenido de www.W3C.es: <https://www.w3c.es/estandares/>

Gustavo Galeano, (2009). Programacion de Sistemas Embebidos en C. Alfaomega; Alfaomega Grupo Editor (MX).

Milan Verle, (2009). PIC Microcontrollers - Programming in C. MikroElektronika; 1st edition

SAP, (2018) HandBook, Cloud for SAP, CLD100.

Celio gil aros, 2009, avances investigación en ingeniería, diseño, desarrollo e implementación de un web service para los establecimientos es del distrito capital

Krutchén, P. (2003). The Rational Unified Process – An Introduction (3rd edition). Reading, MA: Addison-Wesley.

15 Catálogo de Tablas

Tabla 1 Trama de consulta modo 1	15
Tabla 2 Trama de consulta modo 3	17
Tabla 3 Matriz de congruencia.....	24
Tabla 4 Tabla de puntajes	26
Tabla 5 Matriz de evaluación.....	27
Tabla 6 Calendario del proyecto.	30
Tabla 7 Lista de requisitos.....	32
Tabla 8 Matriz comparativa de prototipos en el mercado	34

16 Catálogo de Ilustraciones

Ilustración 1 Evolución de IoT	9
Ilustración 2 Arquitectura IoT	13
Ilustración 3 Trama estándar OBDII.....	15
Ilustración 4 Formato de DTC	16
Ilustración 5 Microcontrolador single chip	17
Ilustración 6 Arquitectura de microcontrolador.....	18
Ilustración 7 Modelos de servicios en la nube.....	19
Ilustración 8 Trama de consulta modo 3	21
Ilustración 9 Diagrama de clases del algoritmo de extracción.....	36

Ilustración 10 Diagrama de componentes de algoritmo de extracción	37
Ilustración 11 Diagrama de secuencia de algoritmo de extracción	38
Ilustración 12 Diagrama de estado de algoritmo de extracción	39
Ilustración 13 Diagrama de actividades de algoritmo de extracción.....	40
Ilustración 14 Diagrama de componentes de Cloud Computing	41
Ilustración 15 Diagrama de servicios de cloud computing.....	42
Ilustración 16 Diagrama de comunicación entre el prototipo de hardware y cloud computing.....	43
Ilustración 17 Diagrama de componentes en Cloud Computing	44
Ilustración 18 Diagrama de actividades para el registro de datos.....	44
Ilustración 19 Diagrama de actividades para la consulta de información.....	45
Ilustración 20 Carga del algoritmo de extracción en el prototipo de hardware (Puerto USB).....	46
Ilustración 21 Carga del algoritmo de extracción en el prototipo de hardware (Sketch).....	47
Ilustración 22 Carga del algoritmo de extracción en el prototipo de hardware.	47
Ilustración 23 Log consola de arduino studio	48
Ilustración 24 Formulario de cuenta AWS.....	49
Ilustración 25 Comando de instalación de Consola AWS.....	49
Ilustración 26 Comando de configuración de credenciales AWS	49
Ilustración 27 Comando de instalación de Serverless Framework.....	49
Ilustración 28 Proyecto Serverless	50
Ilustración 29 Comando de despliegue de proyecto Serverless en AWS	50

17 Anexos

17.1 Anexo 1 – Tabla de principales PID para OBD

PID	Bytes	Descripción	Min.	Max.	Uds.	Fórmula
00	4	PIDs soportados [01-20]	-	-	-	32 Bits: BitX=0 > PIDX Soportado BitX=1 > PIDX No soportado
01	4	Estado de MIL y número de DTCs almacenados.	-	-	-	Bits codificados
03	2	Estado del sistema de combustible	-	-	-	Bits codificados
04	1	Valor de carga del motor	0	100	%	$A \times 100 / 255$
05	1	Temperatura del refrigerante	-40	215	°C	$A - 40$
0A	1	Presión del combustible	0	765	kPa	$A \times 3$
0C	2	Revoluciones por minuto del motor	0	16.383	rpm	$((A \times 256) + B) / 4$
0D	1	Velocidad del vehículo	0	255	Km/h	A
0F	1	Temperatura del aire de entrada	-40	215	°C	$A - 40$
11	1	Posición del acelerador	0	100	%	$A \times 100 / 255$
13 ... 1B	2	Voltaje en los distintos sensores de oxígeno	0	1.275	V	$A / 200$ $(B - 128) \times 100 / 128$

1C	1	Estándar OBD conforme al vehículo	-	-	-	Bits Codificados
1F	2	Tiempo transcurrido desde el arranque	0	65.535	seg.	$(A \times 256) + B$
20	4	PIDs soportados [21-40]	-	-	-	32 Bits: BitX=0 > PIDX Soportado BitX=1 > PIDX No soportado
21	2	Distancia recorrida con el indicador MIL encendido	0	65.535	Km	$(A \times 256) + B$
24 ... 2B	4	Voltaje equivalente de la sonda lambda	0	8	V	$((A \times 256) + B) \times R$ $((C \times 256) + D) \times R$ Con $R = 2/65535$
2C	1	Recirculación de gases de escape o EGR	0	100	%	$A \times 100/255$
2E	1	Depuración por evaporación	0	100	%	$A \times 100/255$
2F	1	Nivel de combustible	0	100	%	$A \times 100/255$
31	2	Distancia recorrida desde el último limpiado de errores	0	65535	km	$(A \times 256) + B$
34 ... 3B	4	Corriente equivalente de la sonda lambda	-128	128	mA	$((A \times 256) + B) / 32.768$ $((C \times 256) + D) / 128$
3C ... 3F	2	Temperatura del catalizador	-40	6.513	°C	$((A \times 256) + B) / 10 - 40$
40	4	PIDs soportados [41-60]	-	-	-	32 Bits: BitX=0 > PIDX Soportado BitX=1 > PIDX No soportado
42	2	Módulo de control de voltaje	0	65.535	V	$(A \times 256) + B$
46	1	Temperatura ambiente	-40	215	°C	$A - 40$
4D	2	Tiempo de marcha mientras el indicador MIL esta encendido	0	65.535	min.	$(A \times 256) + B$
51	1	Tipo de combustible	-	-	-	Bits codificados
5B	1	Nivel de batería híbrida	0	100	%	$A \times 100/255$

17.2 Anexo 2 – Algoritmo de extracción de datos

Principales métodos del algoritmo de extracción: Referencia FreemanticsONE

```
bool CTeleClientSIM800::netBegin()
{
    if (m_stage == 0) {
        xbBegin(XBEE_BAUDRATE);
        m_stage = 1;
    }
    for (byte n = 0; n < 10; n++) {
        // try turning on module
        xbTogglePower();
        sleep(3000);
        // discard any stale data
        xbPurge();
        for (byte m = 0; m < 3; m++) {
            if (netSendCommand("AT\r")) {
                m_stage = 2;
                return true;
            }
        }
    }
    return false;
}

void CTeleClientSIM800::netEnd()
{
    if (m_stage == 2) {
        xbTogglePower();
        m_stage = 1;
    }
}

bool CTeleClientSIM800::netSetup(const char* apn, unsigned int timeout)
{
    uint32_t t = millis();
    bool success = false;
    netSendCommand("ATE0\r");
    do {
        success = netSendCommand("AT+CREG?\r", 3000, "+CREG: 0,1") != 0;
        Serial.print('.');
    } while (!success && millis() - t < timeout);
    if (!success) return false;
    do {
        success = netSendCommand("AT+CGATT?\r", 3000, "+CGATT: 1");
    } while (!success && millis() - t < timeout);
}
```

```

sprintf(m_buffer, "AT+CSTT=\"%s\"\\r", apn);
if (!netSendCommand(m_buffer)) {
    return false;
}
netSendCommand("AT+CIICR\\r");
return success;
}

const char* CTeleClientSIM800::getIP()
{
    for (uint32_t t = millis(); millis() - t < 60000; ) {
        if (netSendCommand("AT+CIFSR\\r", 3000, ".")) {
            char *p;
            for (p = m_buffer; *p && !isdigit(*p); p++);
            char *q = strchr(p, '\\r');
            if (q) *q = 0;
            return p;
        }
    }
    return "";
}

int CTeleClientSIM800::getSignal()
{
    if (netSendCommand("AT+CSQ\\r", 500)) {
        char *p = strchr(m_buffer, ':');
        if (p) {
            p += 2;
            int db = atoi(p) * 10;
            p = strchr(p, '.');
            if (p) db += *(p + 1) - '0';
            return db;
        }
    }
    return -1;
}

const char* CTeleClientSIM800::getOperatorName()
{
    // display operator name
    if (netSendCommand("AT+COPS?\\r") == 1) {
        char *p = strstr(m_buffer, "\\");
        if (p) {
            p += 2;
            char *s = strchr(p, '\\');
            if (s) *s = 0;
            return p;
        }
    }
    return "";
}

```

```

bool CTeleClientSIM800::netOpen(const char* host, uint16_t port)
{
    //netSendCommand("AT+CLPORT=\"TCP\",8181\r");
    netSendCommand("AT+CIPSRIP=1\r");
    //netSendCommand("AT+CIPUDPMODE=1\r");
    sprintf(m_buffer, "AT+CIPSTART=\"UDP\", \"%s\", \"%u\" \r", host, port);
    return netSendCommand(m_buffer, 3000);
}

void CTeleClientSIM800::netClose()
{
    netSendCommand("AT+CIPCLOSE\r");
}

bool CTeleClientSIM800::netSend(const char* data, unsigned int len)
{
    sprintf(m_buffer, "AT+CIPSEND=%u\r", len);
    if (netSendCommand(m_buffer, 200, ">")) {
        xbWrite(data);
        xbWrite("\r");
        if (netSendCommand(0, 5000, "\r\nSEND OK")) {
            return true;
        }
    }
    return false;
}

char* CTeleClientSIM800::netReceive(int* pbytes, unsigned int timeout)
{
    if (netSendCommand(0, timeout, "RCV FROM:")) {
        char *p = strstr(m_buffer, "RCV FROM:");
        if (p) p = strchr(p, '\r');
        if (!p) return 0;
        while (*(++p) == '\r' || *p == '\n');
        int len = strlen(p);
        if (len > 2) {
            if (pbytes) *pbytes = len;
            return p;
        } else {
            if (netSendCommand("AT\r", 1000, "\r\nOK", true)) {
                p = m_buffer;
                while (*p && (*p == '\r' || *p == '\n')) p++;
                if (pbytes) *pbytes = strlen(p);
                return p;
            }
        }
    }
    return 0;
}

```

```

bool CTeleClientSIM800::netSendCommand(const char* cmd, unsigned int timeout, const char*
expected, bool terminated)
{
    if (cmd) {
        xbWrite(cmd);
    }
    m_buffer[0] = 0;
    byte ret = xbReceive(m_buffer, sizeof(m_buffer), timeout, expected);
    if (ret) {
        if (terminated) {
            char *p = strstr(m_buffer, expected);
            if (p) *p = 0;
        }
        return true;
    } else {
        return false;
    }
}

bool CTeleClientSIM800::getLocation(NET_LOCATION* loc)
{
    if (netSendCommand("AT+CIPGSMLOC=1,1\r", 3000)) do {
        char *p;
        if (!(p = strchr(m_buffer, ':'))) break;
        if (!(p = strchr(p, ','))) break;
        loc->lng = atof(++p);
        if (!(p = strchr(p, ','))) break;
        loc->lat = atof(++p);
        if (!(p = strchr(p, ','))) break;
        loc->year = atoi(++p) - 2000;
        if (!(p = strchr(p, '/'))) break;
        loc->month = atoi(++p);
        if (!(p = strchr(p, '/'))) break;
        loc->day = atoi(++p);
        if (!(p = strchr(p, ','))) break;
        loc->hour = atoi(++p);
        if (!(p = strchr(p, ':'))) break;
        loc->minute = atoi(++p);
        if (!(p = strchr(p, ':'))) break;
        loc->second = atoi(++p);
        return true;
    } while(0);
    return false;
}

```

17.3 Anexo 3 - Serverless

Estructura del Proyecto:

serverless-iot-obd

- status-rest
 - resources
 - status-api-deployment.yml
 - status-api-post-method.yml
 - status-api-resource.yml
 - status-api-role.yml
 - status-api.yml
 - serverless.yml
- status-stream
 - resources
 - status-stream-firehose.yml
 - status-stream-role.yml
 - serverless.yml
- status-upload
 - serverless.yml

**status-rest/
resources/status-api-deployment.yml**

Resources:

ApiGatewayDeploymentObdStatus:

Type: [AWS::ApiGateway::Deployment](#)

DependsOn:

- [ApiGatewayPostMethodObdStatus](#)

Properties:

RestApiId: [!Ref ApiGatewayRestApiObdStatus](#)

StageName: [dev](#)

resources/status-api-post-method.yml

Resources:

ApiGatewayPostMethodObdStatus:

Type: [AWS::ApiGateway::Method](#)

Properties:

ApiKeyRequired: [true](#)

AuthorizationType: [NONE](#)

HttpMethod: [POST](#)

Integration:

Type: [AWS](#)

Credentials:

Fn::GetAtt: [[IamRoleApiGatewayObdStatus](#), [Arn](#)]

Uri:

Fn::Join:

- ""
- "arn:aws:apigateway:"
- Ref: [AWS::Region](#)
- ":firehose:action/PutRecord"

```

IntegrationHttpMethod: POST
RequestTemplates:
  application/json:
    Fn::Join:
      - ""
      - - "{\n"
        - "\"DeliveryStreamName\": \"\""
        - Fn::ImportValue: KinesisFirehoseObdStatusId
        - "\"\", \n"
        - "\"Record\": { \"Data\": \"${input.body}\" } \n}"
RequestParameters:
  integration.request.header.Content-Type: "application/x-amz-json-1.1"
IntegrationResponses:
  - StatusCode: 200
    ResponseTemplates:
      application/json: '{"status": "OK"}'
MethodResponses:
  - StatusCode: 200

ResourceId: !Ref ApiGatewayResourceObdStatus
RestApiId: !Ref ApiGatewayRestApiObdStatus

```

resources/status-api-resource.yml

```

Resources:
  ApiGatewayResourceObdStatus:
    Type: AWS::ApiGateway::Resource
    Properties:
      RestApiId:
        Ref: "ApiGatewayRestApiObdStatus"
      ParentId:
        Fn::GetAtt:
          - "ApiGatewayRestApiObdStatus"
          - "RootResourceId"
      PathPart: "status"

```

resources/status-api-role.yml

```

Resources:
  IamRoleApiGatewayObdStatus:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - apigateway.amazonaws.com
      Action:

```

```
- sts:AssumeRole
Path: "/"
Policies:
- PolicyName: lamRoleApiGatewayObdStatusPolicy
  PolicyDocument:
    Version: '2012-10-17'
    Statement:
    - Effect: Allow
      Action:
      - firehose:PutRecord
      Resource: "*"
  Resource: "*"
Resource: "*"

```

Outputs:

```
lamRoleApiGatewayObdStatusId:
  Value:
    Ref: lamRoleApiGatewayObdStatus
  Export:
    Name: lamRoleApiGatewayObdStatusId

```

resources/status-api.yml

Resources:

```
ApiGatewayRestApiObdStatus:
  Type: AWS::ApiGateway::RestApi
  Properties:
    Name:
      Fn::Join:
      - ""
      - - Ref: AWS::StackName
        - "-api"

```

Outputs:

```
ApiGatewayRestApiObdStatusId:
  Value:
    Ref: ApiGatewayRestApiObdStatus
  Export:
    Name: ApiGatewayRestApiObdStatusId

```

serverless.yml

service: iot-obd-status-rest

custom:

```
stage: ${opt:stage, self:provider.stage}
```

provider:

```
name: aws
stage: dev
region: us-west-2

```

```

resources:
  # IAM Role
  - ${file(resources/status-api-role.yml)}
  # API Gateway REST
  - ${file(resources/status-api.yml)}
  # API Gateway Resource
  - ${file(resources/status-api-resource.yml)}
  # API Gateway Method
  - ${file(resources/status-api-post-method.yml)}
  # API Gateway Deployment
  - ${file(resources/status-api-deployment.yml)}

```

status-stream/ resources/status-stream-firehose.yml

```

Resources:
  KinesisFirehoseObdStatus:
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN:
          Fn::ImportValue: S3BucketObdStatusArn
        BufferingHints:
          IntervallInSeconds: 60
          SizeInMBs: 5
        CompressionFormat: GZIP
        Prefix: status/
      RoleARN:
        Fn::GetAtt:
          - lamRoleKinesisFirehoseObdStatus
          - Arn
    Outputs:
      KinesisFirehoseObdStatusId:
        Value:
          Ref: KinesisFirehoseObdStatus
        Export:
          Name: KinesisFirehoseObdStatusId

```

resources/status-stream-role.yml

```

Resources:
  lamRoleKinesisFirehoseObdStatus:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:

```

```

    Service:
      - firehose.amazonaws.com
    Action:
      - sts:AssumeRole
    Path: "/"
    Policies:
      - PolicyName: PolicyKinesisFirehoseObdStatus
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - s3:AbortMultipartUpload
                - s3:GetBucketLocation
                - s3:GetObject
                - s3:ListBucket
                - s3:ListBucketMultipartUploads
                - s3:PutObject
              Resource:
                - Fn::ImportValue: S3BucketObdStatusArn
                - Fn::Join:
                    - ""
                    - - Fn::ImportValue: S3BucketObdStatusArn
                      - "/*"

```

```

Outputs:
  IamRoleKinesisFirehoseObdStatusId:
    Value:
      Ref: IamRoleKinesisFirehoseObdStatus
    Export:
      Name: IamRoleKinesisFirehoseObdStatusId

```

serverless.yml

```

service: iot-obd-status-stream

```

```

custom:
  stage: ${opt:stage, self:provider.stage}

```

```

provider:
  name: aws
  stage: dev
  region: us-west-2

```

```

resources:
  # IAM Role
  - ${file(resources/status-stream-role.yml)}
  # Kinesis Firehose
  - ${file(resources/status-stream-firehose.yml)}

```

status-upload/ serverless.yml

service: **iot-obd-status-upload**

custom:

stage: **\${opt:stage, self:provider.stage}**

provider:

name: **aws**

stage: **dev**

region: **us-west-2**

resources:

Resources:

S3BucketObdStatus:

Type: **AWS::S3::Bucket**

Outputs:

ObdStatusBucketArn:

Value:

Fn::GetAtt:

- **S3BucketObdStatus**

- **Arn**

Export:

Name: **S3BucketObdStatusArn**

ObdStatusBucketName:

Value:

Ref: **S3BucketObdStatus**

Export:

Name: **S3BucketObdStatusName**

17.4 Anexo 4 – Instrumentos de recolección de datos cuantitativos

1. Para elegir el dispositivo OBDII a implementar en las unidades, que tan importante es...	Indispensable (5)	Muy importante (4)	Medianamente importante (3)	Poco importante (2)	No se toma en cuenta (1)
a. Precio					
b. Tiempo en el mercado					

c. Soporte					
d. Disponibilidad					

2. A continuación se presentan los requisitos identificados, en base a las entrevistas con el personal interesado. Favor numerarlas según la prioridad que tenga cada una
(LISTADO DE REQUISITOS FUNCIONALES)

3. El sistema, además de los requisitos operacionales identificados, deberá contemplar los siguientes:	Indispensable (5)	Muy importante (4)	Medianamente importante (3)	Poco importante (2)	No se toma en cuenta (1)
(LISTADO DE REQUISITOS NO FUNCIONALES)					

17.5 Anexo 5 – Códigos de diagnostico

Revisar el documento anexo con nombre “obd2-codigos-fallos.pdf”