Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO *INTERNET OF THINGS*

Linda Alejandra Estrada Cardona
Asesorado por el Ing. Byron Odilio Arrivillaga Méndez

Guatemala, febrero de 2020

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO *INTERNET OF THINGS*

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

LINDA ALEJANDRA ESTRADA CARDONA
ASESORADO POR EL ING. BYRON ODILIO ARRIVILLAGA MÉNDEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERA ELECTRÓNICA

GUATEMALA, FEBRERO DE 2020

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Christian Moisés de la Cruz Leal
VOCAL V	Br. Kevin Armando Cruz Lorente
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Byron Odilio Arrivillaga Méndez
EXAMINADOR	Ing. Armando Alonso Rivera Carrillo
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO *INTERNET OF THINGS*

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 21 de noviembre de 2018.

Linda Alejandra Estrada Cardona

Guatemala, 14 de octubre de 2019

Ingeniero
Julio Cesar Solares
Coordinador del Área de Electrónica
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero Solares:

Por este medio le informo que como asesor del Trabajo de Graduación de la estudiante de la carrera de ingeniería electrónica **Linda Alejandra Estrada Cardona**, quien se identifica con número de carné 201212549, procedí a revisar su trabajo de tesis, titulado: ***"DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO INTERNET OF THINGS",*** habiéndolo encontrado satisfactorio.

En tal virtud, **Lo doy por aprobado**, solicitando dar el trámite respectivo.

Sin otro particular, agradezco de antemano su atención.
Atentamente:

Byron Arrivillaga Méndez
Ingeniero Electrónico
Colegiado 5217

Byron Odilio Arrivillaga Méndez
Ingeniero Electrónico
Colegiado no. 5217
Asesor

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 23 de octubre de 2019

Señor Director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor Director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado **DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO INTERNET OF THINGS,** desarrollado por la estudiante **Linda Alejandra Estrada Cardona**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

FACULTAD DE INGENIERIA · UNIVERSIDAD DE SAN CARLOS DE GUATEMALA · DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA

Ing. Julio César Solares Peñate
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

REF. EIME 70. 2019.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto bueno del Coordinador de Área, al trabajo de Graduación del estudiante: LINDA ALEJANDRA ESTRADA CARDONA titulado: DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO INTERNET OF THINGS, procede a la autorización del mismo.

Ing. Armando Alonso Rivera Carrillo

FACULTAD DE INGENIERÍA · UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA

GUATEMALA, 31 DE OCTUBRE 2019.

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Decanato
Facultad de Ingeniería
24189102 - 24189103

DTG. 089.2020

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO DE UN SISTEMA DE ANÁLISIS, CONTROL Y SEGURIDAD DE USUARIOS APLICANDO *INTERNET OF THINGS*,** presentado por la estudiante universitaria: **Linda Alejandra Estrada Cardona,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Inga. Anabela Cordova Estrada
Decana

Guatemala, febrero de 2020

/gdech

ACTO QUE DEDICO A:

Dios	Por ser mi guía e iluminación todos los días.
Mis padres	Alma Cardona y Elmer Estrada, por su apoyo y amor incondicional.
Mi novio	Haroldo López de los Ríos, por todo su amor, paciencia y apoyo.
Mi hermanito	Johan Andrée Estrada, por todo su cariño y la felicidad que me brinda.
Mis abuelos	Margarita López, Norberto Estrada (q.e.p.d.), Teresa Arévalo (q.e.p.d.), por su cariño, apoyo y los momentos compartidos.
Mis amigos	Karla Kim, Ludwig Hernández, José David Axpuac y Byron Arrivillaga, por su amistad, apoyo y experiencias compartidas a lo largo de la carrera.
Mis tíos	Linda y Cony Estrada, Juan García y Siomara Rivera, por sus experiencias.
Mis primos	Elizabeth Giron, Meliza, Yaderline y Erick Polanco, por su apoyo.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser mi casa de estudios.
Facultad de Ingeniería	Por formarme como profesional.
Mis amigos de la Facultad	Karla Kim, Ludwig Hernández, José Axpuac, Alejandro Castillo y Jaganath León.
Alma Cardona	Por su amor y su amparo.
Elmer Estrada	Por sus instrucciones y conocimientos compartidos.
Haroldo José López de los Ríos	Por estar siempre que lo necesito y por todo su amor incondicional.
Byron Arrivillaga	Por compartir sus conocimientos y apoyarme.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperios
bps	Bits por segundo
°C	Centígrados
cm	Centímetro
I	Corriente
DC	Corriente directa
dBi	Decibelios de ganancia isotrópica
dBm	Decibelio-mili vatio
fps	Fotogramas por segundo
GHz	Giga Hertz
Gbps	Gigabits por segundo
Hz	Hertz
RH	Humedad Relativa
kB	Kilobytes
kg	Kilogramo
Mbps	Megabits por segundo
MP	Megapíxeles
m	Metro
mm	Milímetro
plg	Pulgada
s	Segundos

GLOSARIO

Arduino Placa de microcontrolador de código abierto basado en el microchip ATmega328P.

Baudrate Unidad de medida que representa el número de símbolos por segundo en un medio de transmisión digital.

Callback Función A que se usa como argumento de otra función B. Cuando se llama a B, esta ejecuta A.

GPRS Servicio general de paquetes vía radio.

GPS Sistema de posicionamiento global.

GSM Sistema global para las comunicaciones móviles.

Full-duplex Sistema que es capaz de mantener una comunicación bidireccional y simultáneo.

Half-duplex Envío de información que es bidireccional pero no simultáneo.

IoT Internet de las cosas (Internet of things).

Módem	Dispositivo que convierte las señales digitales en analógicas (modulación) y viceversa (desmodulación).
Ping	Utilidad de diagnóstico en redes de computadoras que comprueba el estado de la comunicación del anfitrión local con uno o varios equipos remotos de una red que ejecuten IP.
Router	Dispositivo de red que se encarga de llevar por la ruta adecuada el tráfico.
Web socket	Tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP.

RESUMEN

El sistema de internet de las cosas propuesto tiene cuatro módulos que conforman su hardware. Comprende los dispositivos o activos que pueden ser manipulables o capaces de ser monitoreados o ser controlados para realizar las acciones. La adquisición de datos en la cual se utilizaron dispositivos transductores, tales como sensores de iluminación, humedad, gases, identificación, etc. Son adaptados a la disposición de las habitaciones y al comportamiento que realiza el usuario dentro de ellas.

Obtener las condiciones del medio implica tener un procesamiento y conversión de los datos, los cuales son almacenados localmente e identificados por microcontroladores capaces de conectarse a los módulos de red. Son encargados de realizar la comunicación con la nube para el almacenamiento y manipulación estadística de los datos.

Para realizar el sistema se diseñaron dos clases de modelos. El primero para una red local conformada por los sensores, los módulos de acceso a internet en cada dispositivo, su conexión con el router y finalmente la llegada de la información a la nube, que permite la administración del encargado. Este modelo proporciona acceso y administración en el sitio. El segundo modelo implicó utilizar la información de los transductores mencionados en el modelo anterior, con la diferencia de que estos tienen conexión con módulos que proporcionan acceso al uso de la red GPRS / GSM / GPS. Este modelo hace posible utilizar la conexión móvil para manipular el comportamiento de los dispositivos y acceder a la información.

OBJETIVOS

General

Crear un sistema seguro de Internet de las cosas que permita controlar y analizar la información de los usuarios en ambientes comunes.

Específicos

1. Identificar los transductores adecuados dependiendo de las necesidades del ambiente establecidas en el modelo de negocio.

2. Diseñar una red de Internet de las cosas con dispositivos de bajo costo, que proporcionen al usuario datos y le permitan ahorrar energía.

3. Utilizar una red doméstica y una red celular para controlar los dispositivos, crean para 2 tipos de comunicaciones.

4. Facilitar el control de los dispositivos de Internet de las cosas mediante una interfaz que muestre la información, permita identificar las cosas e implementar acciones.

INTRODUCCIÓN

El aumento en el uso del internet y la modernización han permitido el avance y diseño de dispositivos comunes que ahora pueden comunicarse con la red. La implementación de estos sistemas permite controlar, analizar y estudiar los ambientes comunes; son accesibles para implementarse en los hogares, hoteles, restaurantes, entre otros.

En el capítulo uno se establecen los objetivos de esta propuesta y se delimita los alcances del sistema como un modelo de negocio, en el cual se muestra la entrega de una propuesta sólida y los requerimientos que esta cumplirá para volver del sistema un producto, mientras muestra sus características innovadoras.

El capítulo dos consta de la descripción sobre qué es el internet de las cosas; explica los cuatro módulos para la creación de un sistema de IoT, las tecnologías que se utilizan, protocolos de comunicación y las herramientas que permiten desarrollarlo y definen cada una.

Un sistema de Internet de las cosas permite tener un control sobre los ambientes para la obtención de la información; esta información crea estadísticas y predicciones, por lo que es necesario tener transductores que obtengan los datos del medio para transformarlos en datos digitales. Estos dispositivos son los sensores, los cuales son descritos con sus características en el capítulo tres.

El modelo propuesto es descrito en el capítulo 4. Este diseño muestra dos tipos de casos en los cuales se puede acceder al control de los ambientes y mantener siempre una conexión y actualización del estado. Se muestra cuáles son los problemas que se generan cuando se conecta varios dispositivos en una red de IoT. Se da a conocer la programación básica de los dispositivos utilizados. Se utilizó un servidor que funciona como un servicio MQTT y clientes que se suscriben a este para tener una comunicación a tiempo a real full-duplex.

.

1. DISEÑO DEL PROYECTO Y MODELO DE NEGOCIO

1.1. Diseño del proyecto

A continuación, se muestran las características para el diseño del proyecto:

1.1.1. Denominación del proyecto

En este proyecto se propone el diseño de un sistema de control de Internet de las cosas para el control de ambientes. Este mismo tiene sus objetivos basados en un tipo específico de cliente, como un modelo de negocio. Para encontrar sus características, definir sus necesidades y desarrollar el diseño más adecuado y funcional para sus instalaciones, se presenta más adelante los módulos que definen el negocio. Conviene expresar que el diseño propone innovación, actualización y factibilidad de uso.

1.1.2. Descripción del proyecto

El proyecto está basado en clientes encargados de la administración de hoteles con más de 30 habitaciones, por lo que el diseño del sistema de Internet de las cosas pretende darle al cliente la posibilidad de tener un control estadístico y remoto de las habitaciones y ambientes del hotel.

El sistema consta de varios sensores, encargados de la obtención de la información, los cuales serán distribuidos en los diferentes ambientes; además

posee servicios enlazados en la nube, para el almacenamiento y análisis de los datos estadísticos.

1.1.3. Fundamentación o justificación del proyecto

- Se crea este modelo de negocio como una propuesta para solventar ciertos problemas existentes y para dar un mayor control a los encargados de la administración, así como para permitirles mayor seguridad, tanto a los clientes como a los trabajadores del hotel.

- La estrategia para implementar y vender el proyecto consiste en aplicar los 9 módulos del lienzo canvas para conseguir los objetivos propuestos y determinar las necesidades en el diseño del cliente.

1.1.4. Objetivos del modelo de negocio

Se describen los objetivos generales y específicos del modelo de negocio.

1.1.4.1. Objetivo general

Diseñar un sistema de control de Internet de las cosas, eficiente, innovador, seguro, factible y que cumpla con los requerimientos necesarios.

1.1.4.2. Objetivos específicos

- Entregar al cliente una propuesta sólida del negocio
- Presentar siempre innovación y actualizaciones
- Desarrollar un sistema útil y seguro

1.2. Modelo de negocio

Un modelo de negocio describe las bases sobre las que una empresa se crea, proporciona y capta valor. La mejor manera de describir un modelo de negocio es dividirlo en los nueve módulos básicos que reflejan la lógica que se seguirá para el desarrollo de la propuesta y para cubrir las 4 áreas principales: clientes, oferta, infraestructura y viabilidad económica. A continuación, se presenta la estrategia que se aplicará en el desarrollo de este proyecto.

1.2.1. Segmentos de mercado

En este módulo se define los diferentes grupos de personas o entidades a los que se dirige una empresa. Un modelo de negocio puede definir uno o varios segmentos de mercado, ya sean grandes o pequeños. Por lo tanto, se debe seleccionar, con una decisión fundamentada, los segmentos a los que se van a dirigir y, al mismo tiempo, los que no se tendrán en cuenta.

Los grupos de clientes pertenecen a segmentos diferentes si:

- Sus necesidades requieren y justifican una oferta diferente
- Son necesarios diferentes canales de distribución para llegar a ellos
- Requieren un tipo de relación diferente
- Su índice de rentabilidad es muy diferente
- Están dispuestos a pagar por diferentes aspectos de la oferta

1.2.1.1. Selección del cliente

De acuerdo con lo anterior, se define al cliente como hoteles que cuenten con más de 30 habitaciones. Esto debido a que el proyecto requiere clientes

que posean un número considerable de ambientes para necesitar un control estadístico; además, la contratación de este servicio requiere una inversión monetaria considerable.

1.2.2. Propuestas de valor

En este módulo se describe el conjunto de productos y servicios que crean valor para un segmento de mercado específico. La propuesta de valor es el factor que hace que un cliente se decante por una u otra empresa; su finalidad es solucionar un problema o satisfacer una necesidad del cliente. Las propuestas de valor son un conjunto de productos o servicios que satisfacen los requisitos de un segmento de mercado determinado. En este sentido, la propuesta de valor constituye una serie de ventajas que una empresa ofrece a los clientes.

Algunas propuestas de valor pueden ser innovadoras y presentar una oferta nueva o rompedora, mientras que otras pueden ser parecidas a ofertas ya existentes e incluir alguna característica o atributo adicional.

1.2.2.1. Valores proporcionados al cliente

El proyecto tiene como objetivo ofrecer al cliente un sistema seguro, confiable, innovador, moderno, que además proporcione ahorro energético y un control estadístico para la obtención de información en tiempo real, verificable mediante internet, y con un diseño especial, de acuerdo con las condiciones y necesidades del lugar y los ambientes.

1.2.3. Canales

En el siguiente módulo se explica el modo en que una empresa se comunica con los diferentes segmentos de mercado para llegar a ellos y proporcionarles una propuesta de valor.

Los canales de comunicación, distribución y venta establecen el contacto entre la empresa y los clientes. Son puntos de contacto con el cliente, que desempeñan un papel primordial en su experiencia.

Los canales tienen, entre otras, las funciones siguientes:

- Dar a conocer a los clientes los productos y servicios de una empresa
- Ayudar a los clientes a evaluar la propuesta de valor de una empresa
- Permitir que los clientes compren productos y servicios específicos
- Proporcionar a los clientes una propuesta de valor
- Ofrecer a los clientes un servicio de atención posventa

1.2.3.1. Canal del proyecto

El canal para el desarrollo y de este proyecto es el internet, más específicamente una página web y los enlaces pertinentes.

1.2.4. Relaciones con los clientes

En este módulo se describen los diferentes tipos de relaciones que establece una empresa con determinados segmentos de mercado.

Las empresas deben definir el tipo de relación que desean establecer con cada segmento de mercado. La relación puede ser personal o automatizada. Las relaciones con los clientes pueden estar basadas en los fundamentos siguientes:

- Captación de clientes
- Fidelización de clientes
- Estimulación de las ventas

En sus inicios, las relaciones con clientes de los operadores de redes móviles se basaban en agresivas estrategias de captación. El tipo de relación que exige el modelo de negocio de una empresa repercute en gran medida en la experiencia global del cliente.

1.2.4.1. Tipo de relación con los clientes

- Asistencia personal en cuanto a la interacción con el cliente, ya sea para resolución de dudas o solventar problemas, para que el cliente pueda comunicarse con un representante del proyecto.

- Asistencia personal exclusiva; porque se le asigna al cliente una persona capaz de proyectar su proyecto y diseñar su sistema.

1.2.5. Fuentes de ingresos

Este módulo se refiere al flujo de caja que genera una empresa en los diferentes segmentos de mercado.

Si los clientes constituyen el centro de un modelo de negocio, las fuentes de ingresos son sus arterias. Cada fuente de ingresos puede tener un mecanismo de fijación de precios diferente: lista de precios fijos, negociaciones, subastas, según mercado, según volumen o gestión de la rentabilidad.

Un modelo de negocio puede implicar dos tipos diferentes de fuentes de ingresos:

- Ingresos por transacciones derivados de pagos puntuales de clientes.
- Ingresos recurrentes derivados de pagos periódicos realizados a cambio del suministro de una propuesta de valor o del servicio posventa de atención al cliente.

1.2.5.1. Valor por el cual están dispuestos a pagar los clientes

- Venta del servicio: es indispensable pago total del proyecto, debido a que este incluye un servicio completo que no puede ser llevado a cabo por partes.

- Cuota de suscripción por mantenimiento: incluye solventar cualquier necesidad, mejora o mantenimiento.

- Préstamo o alquiler del servicio.

1.2.6. Recursos clave

En este módulo se describen los activos más importantes para que un modelo de negocio funcione.

Todos los modelos de negocio requieren recursos clave que permiten a las empresas crear y ofrecer una propuesta de valor, llegar a los mercados, establecer relaciones con segmentos de mercado y percibir ingresos. Cada modelo de negocio requiere recursos clave diferentes. Los recursos clave pueden ser físicos, económicos, intelectuales o humanos. Además, la empresa puede tenerlos en propiedad, alquilarlos u obtenerlos de sus socios clave.

1.2.6.1. Recursos clave del proyecto

- Físicos: activos físicos, como instalaciones de fabricación, máquinas, sistemas, redes de distribución.
- Intelectuales: información privada, diseño, asociaciones y base de datos de los clientes.
- Humanos: recursos humanos en los ámbitos creativos, conocimientos.
- Económicos: garantías económicas como dinero en efectivo, líneas de crédito, entre otros.

1.2.7. Actividades clave

En el presente módulo se describe las acciones más importantes que debe emprender una empresa para que su modelo de negocio funcione.

Todos los modelos de negocio requieren una serie de actividades clave. Estas son las acciones más importantes que debe emprender una empresa para tener éxito y, al igual que los recursos clave, son necesarias para crear y ofrecer una propuesta de valor, llegar a los mercados, establecer relaciones con clientes y percibir ingresos. Además, las actividades también varían en función del modelo de negocio.

1.2.7.1. Actividades clave en el proyecto

- Producción: consta en la construcción y mantenimiento del proyecto.
- Resolución de problemas: para dar satisfacción y confiabilidad a los clientes es necesario, solventar cualquier tipo de duda o defecto en el proyecto.
- Atención inmediata: es aplicable ya que ofrecer un producto confiable, incluye atender las necesidades del cliente.
- Plataforma/red: mantener la web actualizada y todos los enlaces en orden.

1.2.8. Asociaciones clave

En este módulo se describe la red de proveedores y socios que contribuyen al funcionamiento de un modelo de negocio.

Las empresas se asocian por múltiples motivos y estas asociaciones son cada vez más importantes para muchos modelos de negocio. Las empresas crean alianzas para optimizar sus modelos de negocio, reducir riesgos o adquirir recursos. Se puede hablar de cuatro tipos de asociaciones:

- Alianzas estratégicas entre empresas no competidoras.
- Coopetición: asociaciones estratégicas entre empresas competidoras.
- *Joint Ventures* (empresas conjuntas) para crear nuevos negocios.
- Relaciones cliente-proveedor para garantizar la fiabilidad de los suministros.

1.2.8.1. Socios clave en el proyecto

Son socios indispensables: los distribuidores de productos electrónicos, los proveedores de internet y de servicios enlazados en la red.

1.2.9. Estructura de costes

En este último módulo se describen todos los costes que implica la puesta en marcha de un modelo de negocio.

Se describe los principales costes en los que se incurre al trabajar con un modelo de negocio determinado. Tanto la creación y la entrega de valor como el mantenimiento de las relaciones con los clientes o la generación de ingresos tienen un coste. Estos costes son relativamente fáciles de calcular una vez que se han definido los recursos clave, las actividades clave y las asociaciones clave. No obstante, algunos modelos de negocio implican más costes que otros.

1.2.9.1. Estructura de costes en el proyecto

Costes variables: variar en proporción a la extensión del diseño y el tipo de cliente. Características tales como el lugar donde se realiza la aplicación del diseño, extensión del hotel, cantidad de ambientes, variarán dependiendo al número de sensores y de enlaces que sea necesario colocar.

1.2.10. Canvas

Los nueve módulos del modelo de negocio forman la base de una herramienta útil: el lienzo del modelo de negocio. A continuación, se presenta el lienzo con el cual se definió el cliente y los objetivos del diseño.

Tabla I. **Lienzo**

<table>
<tr><th>Asociaciones clave</th><th>Actividades clave</th><th>Propuesta de valor</th><th>Relaciones con clientes</th><th>Segmentos de mercado</th></tr>
<tr><td rowspan="3">Distribuidora de productos electrónicos.</td><td>Producción
Resolución de problemas
Atención inmediata
Plataforma/ red.</td><td rowspan="3">Control estadístico de los usuarios.

Ahorro energéticok.

Seguridad.

Modernización
Accesibilidad.

Diseño.

Personalización.

Comodidad.

Reducción de riesgos.</td><td>Asistencia personal.

Servicios automáticos.

Asistencia personal exclusiva.</td><td rowspan="3">Hoteles con más de 30 habitaciones.</td></tr>
<tr><td>Recursos clave</td><td>Canales</td></tr>
<tr><td>Físicos
Intelectuales
Humanos
Económicos
Internet.</td><td>Ventas en internet.</td></tr>
<tr><td colspan="3">Estructura de costes</td><td colspan="2">Fuentes de ingresos</td></tr>
<tr><td colspan="3">Costes variables: desarrollo del software, tamaño del lugar, cantidad de ambientes, diseño.</td><td colspan="2">Venta del servicio.

Cuota de suscripción por mantenimiento.

Préstamo o alquiler.</td></tr>
</table>

Fuente: elaboración propia.

2. INTERNET DE LAS COSAS

2.1. ¿Qué *es* Internet de las cosas*?*

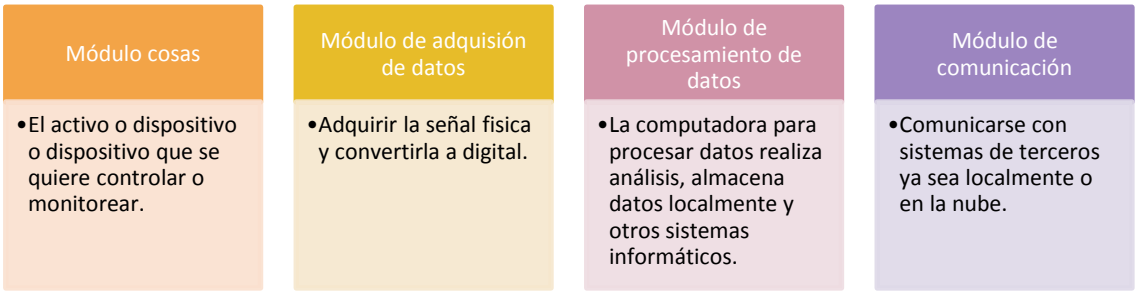
Internet de las cosas, también definido en inglés como *Internet of Things* (IoT) se refiere a la conexión de dispositivos físicos al internet; estos dispositivos y comparten datos. Los procesadores económicos y las redes inalámbricas han permitido convertir cualquier cosa en un dispositivo parte de Internet de las cosas, lo que agrega un nivel de inteligencia digital a los dispositivos comunes para que se comuniquen sin la participación de un ser humano.

Algunos ejemplos sencillos para la compresión de dispositivos IoT serían una bombilla, la cual se puede encender con una aplicación para teléfono inteligente o sensores de movimiento y termostatos inteligentes, que son colocados en oficinas o lugares donde se requiere mantener un control sobre el ambiente.

2.2. Hardware en un sistema de internet de las cosas

El hardware utilizado en los sistemas de internet de las cosas incluye dispositivos para un panel de control remoto, dispositivos para control, servidores, un dispositivo de enrutamiento o puente y sensores. Estos dispositivos administran tareas y funciones clave, como la activación del sistema, las especificaciones de acción, la seguridad, la comunicación y la detección para cumplir con objetivos y acciones específicas.

Figura 1. **Módulos de hardware de un sistema IoT**



Fuente: elaboración propia

2.2.1. Módulo cosas

En muchos productos de IoT, la "cosa" está totalmente integrada en el dispositivo inteligente.

Pero hay muchas otras aplicaciones en las que la "cosa" está sola, como un dispositivo "tonto", y se conecta un producto separado para convertirlo en un dispositivo inteligente.

2.2.2. Módulo adquisición de datos

Este es el componente de hardware que incluye todos los sensores que adquieren señales del mundo real, como temperatura, movimiento, luz, vibración, entre otro. El tipo y número de sensores que necesita dependen de su aplicación.

El módulo de adquisición de datos incluye más que sensores, sin embargo, también incluye el hardware necesario para convertir la señal del

sensor en información digital para que la computadora la use. Esto incluye el acondicionamiento de la señal, la conversión analógica a digital, el escalado y la interpretación.

Para el módulo de adquisición de datos, las consideraciones importantes son:

- Señales físicas por medir.
- Cantidad de sensores de cada tipo que son necesarios.
- Rapidez con la que se debe medir la señal.
- Precisión en la medición.

2.2.3. Módulo de procesamiento de datos

Las dos consideraciones más importantes para centrarse son:

- Potencia de procesamiento.
- Cantidad de almacenamiento de datos local (es decir, tamaño del disco duro).

Para determinar cuánta potencia de procesamiento necesita el sistema, se debe comenzar por comprender las diferentes tareas que debe realizar el dispositivo. Los elementos que afectarán su decisión incluyen:

- La cantidad de sensores que necesita leer.
- Control en tiempo real.
- Capacidad de procesamiento para actualizaciones, versiones de software.
- Crecimiento de los dispositivos.

2.2.4. Módulo de comunicación

El último bloque de construcción del hardware de un sistema IoT es el módulo de comunicaciones. Este es el circuito que permite las comunicaciones con su plataforma en la nube y con sistemas de terceros, localmente o en la nube.

Este módulo puede incluir puertos de comunicación como USB, serie (232/485), CAN o Modbus, por nombrar algunos. También puede incluir la tecnología de radio para comunicaciones inalámbricas como wifi, LoRA, ZigBee, entre otras.

El módulo de comunicaciones se puede incluir en el mismo dispositivo que sus otros módulos, o podría ser un dispositivo separado específicamente para comunicaciones. Este enfoque a menudo se conoce como una arquitectura de puerta de enlace.

Por ejemplo, si tiene tres sensores en una sala que necesitan enviar datos a la nube, puede tener esos sensores conectados a una única puerta de enlace en esa misma sala. Puerta de enlace consolida estos datos y los envía a la nube. De esa manera, solo se necesita un módulo de comunicaciones, no tres.

2.3. Tecnología para comunicaciones inalámbricas

Para desarrollar los proyectos en un entorno de Internet de las cosas son necesarias diversas tecnologías, que mejoran la eficacia y eficiencia de las empresas, hogares o ambientes. Dichas tecnologías contribuyen a cambiar los procesos y sistemas. Cada día hay más dispositivos que están a la espera de recibir instrucciones. Algo que se debe tener presente es que los dispositivos

inteligentes utilizan infraestructuras similares a las de las computadoras. Esta es una lista de los fundamentos primarios que están definidos en el mundo de IoT actualmente:

2.3.1. Topologías de redes

Una consideración clave al diseñar redes de IoT es la topología de las redes. Hay dos tipos principales de redes que se utilizan en el diseño de redes relacionadas con IoT: estrella y malla.

En una topología de red en estrella, todos los nodos están conectados a uno central, que suele ser la puerta de entrada a Internet. Una red wifi de uso común es donde el nodo central se denomina punto de acceso y los nodos conectados se denominan estaciones. Las redes en estrella se caracterizan por la transferencia de grandes bloques de datos, velocidades de interconexión rápidas y tiempos de respuesta rápidos.

En una red de malla, cada nodo está conectado entre sí. Fuera de los múltiples nodos dentro de dichas redes locales, algunos de ellos actúan como pasarelas de Internet y retransmiten datos dentro y fuera de la red de propiedad local a Internet. Dado que el proceso de comunicación es un gran número de pequeños saltos, la velocidad de comunicación dentro y fuera de una red de malla local es relativamente lenta. También son más complejos de diseñar que las redes de estrella.

2.3.2. Tipos de redes

Las redes de área personal generalmente tienen capacidad inalámbrica y cubren un rango de aproximadamente 10 metros. Un PAN inalámbrico común es un teléfono inteligente conectado a través de Bluetooth a varios accesorios. Los dispositivos PAN inalámbricos usualmente tienen baja potencia de transmisión de radio y funcionan con baterías pequeñas.

Las redes de área local son cableadas o inalámbricas o una combinación de ambas. Las LAN inalámbricas usualmente cubren un rango de hasta 100 metros. Un ejemplo es la red wifi doméstica que proporciona acceso a Internet a computadoras personales, teléfonos inteligentes, televisores y dispositivos domésticos de IoT.

Las redes de área de vecindarios generalmente tienen capacidad inalámbrica y pueden alcanzar aproximadamente 25 km. Utilizan altos niveles de potencia, pero transfieren bloques de datos relativamente bajos.

2.3.3. Estándares de interoperabilidad

Uno de los mayores desafíos en dispositivos, sensores, redes y aplicaciones de IoT es la capacidad de entender y comunicarse entre sí. Esto también se llama interoperabilidad. Varias instituciones, alianzas y foros han tomado la iniciativa de hacer avanzar las industrias relacionadas de manera cohesiva.

El Instituto de Ingenieros Eléctricos y Electrónicos ha contribuido a la familia de estándares IEEE 802.x. 802.3 con la especificación de Ethernet utilizada para redes de computadoras cableadas; 802.11 es la especificación

para LAN inalámbrica; 802.15.4 es la especificación para el estándar PAN utilizado en ZigBee, 6LoWPAN.

2.3.4. Protocolos inalámbricos

Sin embargo, algunos de los cambios más innovadores se producen con los protocolos de redes inalámbricas. Se pueden clasificar en función de las siguientes características operativas:

- Tamaño de bloques de transferencia de datos.
- Rango de conectividad.
- Requerimientos de energía.
- Topología de redes.

2.3.5. Bluetooth Low Energy, Bluetooth Smart

Bluetooth es una tecnología de comunicaciones de corto alcance, que se ha vuelto importante en productos de computación y consumo. Será la clave para los productos portátiles que se conectan a *Internet of Things* a través de teléfonos inteligentes en la mayoría de los casos. Sin embargo, para aplicaciones IoT es Bluetooth *Low-Energy* o *Bluetooth Smart*, que es más importante ya que su consumo de energía es menor que el de Bluetooth. A diferencia de Bluetooth, *Bluetooth Smart* no puede usarse para transferencias de archivos y su tamaño de paquete de datos es menor.

- Frecuencia: 2,4 GHz.
- Rango: menos de 150 m.
- Tasas de datos: 1 Mbps.

2.3.6. ZigBee

ZigBee y sus diversos perfiles industriales se basan en el protocolo IEEE802.15.4, que es una tecnología de red inalámbrica estándar en la industria. Está diseñado para aplicaciones que requieren transferencias de datos limitadas a bajas tasas de transferencia dentro de un rango de 100 m, generalmente en una casa o edificio. Tiene ventajas en sistemas complejos que requieren operación de bajo consumo de energía, altos niveles de seguridad, alta escalabilidad, altos recuentos de nodos y puede admitir redes de sensores y control inalámbrico en aplicaciones de IoT.

- Frecuencia: 2,4 GHz.
- Rango: menos de 100 m.
- Tasas de datos: 250 kbps.

2.3.7. Z-Wave

Es una tecnología de comunicación de baja potencia y baja velocidad de datos, diseñada para la automatización del hogar. Admite redes de malla completa y es escalable, lo que permite el control de hasta 232 dispositivos. Z-Wave utiliza un protocolo más simple que otros, lo que permite un desarrollo más rápido.

- Frecuencia: 900 MHz.
- Alcance: 30 m.
- Tasas de datos: menos de 100 kbps.

2.3.8. LoRaWAN

LoRaWAN se enfoca en aplicaciones de red de área amplia con bajos requisitos de energía, incluida la comunicación móvil en IoT, ciudad inteligente e industrial. Está específicamente optimizado para un bajo consumo de energía y admite redes con miles y millones de dispositivos. La tasa de transferencia de datos es muy baja, a menos de 50 kbps.

- Frecuencia: varios.
- Rango: 2-5 km urbano, 15 km suburbano.
- Tasas de datos: menos de 50 kbps.

2.3.9. Thread

Thread se basa en el protocolo de red IPv6 y está destinado a automatizar el entorno doméstico. Utiliza el silicio inalámbrico existente de los proveedores de chips y es compatible con una red de malla que utiliza IEEE802.15.4. Es capaz de manejar hasta 250 nodos con altos niveles de autenticación y cifrado. Se basa en 6LowPAN y está diseñado para complementar wifi. Reconoce que, si bien el wifi es bueno para muchos dispositivos de consumo, tiene limitaciones para su uso en una configuración de automatización del hogar. Una actualización de software permite a los usuarios ejecutar subprocesos en dispositivos existentes habilitados para IEEE802.15.4. Trabaja en 2,4 GHz de frecuencia.

2.3.10. Celular

Cualquier aplicación de IoT que requiera operaciones a largas distancias puede aprovechar las ventajas de Cellular GSM, 3G, 4G. Cellular es adecuado

para grandes volúmenes de datos, pero es probable que el costo y el consumo de energía para gestionar grandes volúmenes de transferencia de datos sean demasiado altos para la mayoría de las aplicaciones de IoT. Cellular es adecuado para proyectos de baja data impulsados por sensores, transferidos a través de Internet.

- Frecuencias: 900, 1 800, 1 900, 2 100 MHz.
- Rango: 35 km máx para GSM; 200 km máx para HSPA.
- Tasas de datos: menos de 170 kps GPRS, menos de 384 kbps EDGE, menos de 2 Mbps UMTS, menos de 10 Mbps HSP, 3-10 Mbps LTE.

2.4. Machine Learning

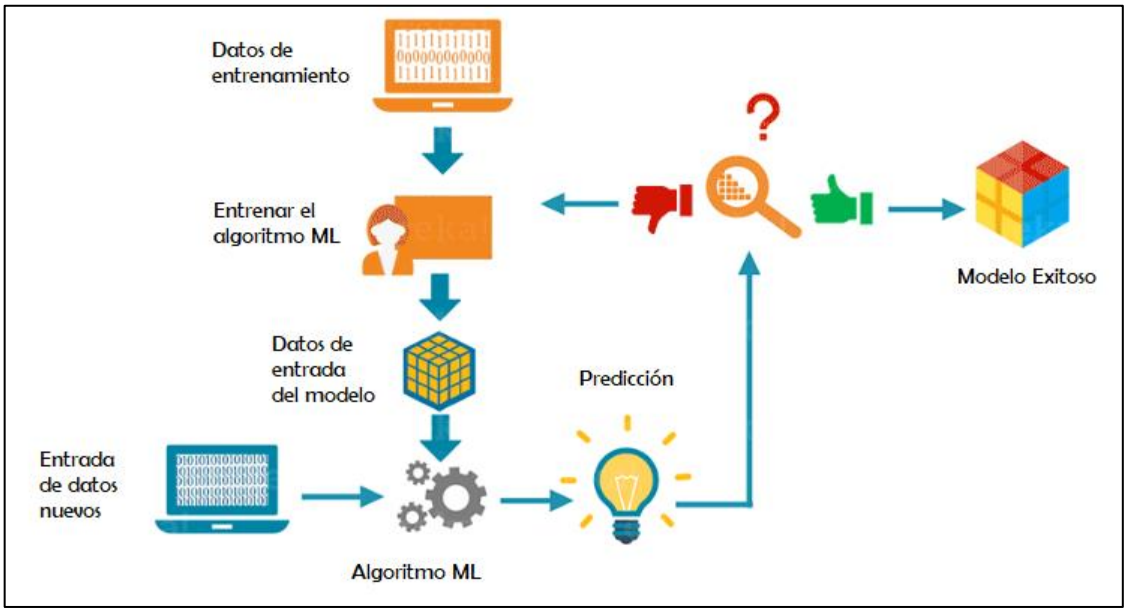
El aprendizaje automático es una aplicación de inteligencia artificial que proporciona a los sistemas la capacidad de aprender y mejorar automáticamente a partir de la experiencia, sin ser programado explícitamente. El aprendizaje automático se centra en el desarrollo de programas informáticos que pueden acceder a los datos y utilizarlos, es decir, aprender por sí mismos.

El proceso de aprendizaje comienza con observaciones o datos, como ejemplos, experiencia directa o instrucción; para buscar patrones en los datos y tomar mejores decisiones en el futuro en función de los ejemplos que brindamos. El objetivo principal es permitir que las computadoras aprendan automáticamente sin intervención o asistencia humana y ajustar las acciones en consecuencia.

2.4.1. Cómo trabaja Machine Learning

En el diagrama de la figura no.2 se muestra el funcionamiento de trabajo de Machine Learning.

Figura 2. **Diagrama de trabajo Machine Learning**



Fuente: elaboración propia.

El algoritmo de aprendizaje automático se entrena al utilizar un conjunto de datos de entrenamiento para crear un modelo. Cuando se introducen nuevos datos de entrada en el algoritmo ML, se hace una predicción sobre la base del modelo.

La predicción se evalúa con respecto a la precisión y, si la precisión es aceptable, se implementa el algoritmo de aprendizaje automático. Si la precisión

no es aceptable, el algoritmo de aprendizaje automático se entrena una y otra vez con un conjunto de datos de entrenamiento aumentado.

Este es solo un ejemplo de muy alto nivel, ya que hay muchos factores y otros pasos involucrados.

2.4.2. Tipos de Machine Learning

Se muestran los tipos de aprendizaje en Machine Learning:

2.4.2.1. Aprendizaje supervisado

El aprendizaje supervisado es aquel en el que puede considerar que este aprendizaje está guiado por un maestro. Tenemos un conjunto de datos que actúa como profesor y su función es entrenar el modelo o la máquina. Una vez que el modelo se entrena, puede comenzar a hacer una predicción o decisión cuando se le proporciona nuevos datos.

2.4.2.2. Aprendizaje sin supervisión

El modelo aprende a través de la observación y encuentra estructuras en los datos. Una vez que el modelo recibe un conjunto de datos, automáticamente encuentra patrones y relaciones en el conjunto de datos, y creando grupos en él. Lo que no puede hacer es agregar etiquetas al grupo, como no puede decir esto un grupo de manzanas o mangos, pero separará todas las manzanas de los mangos.

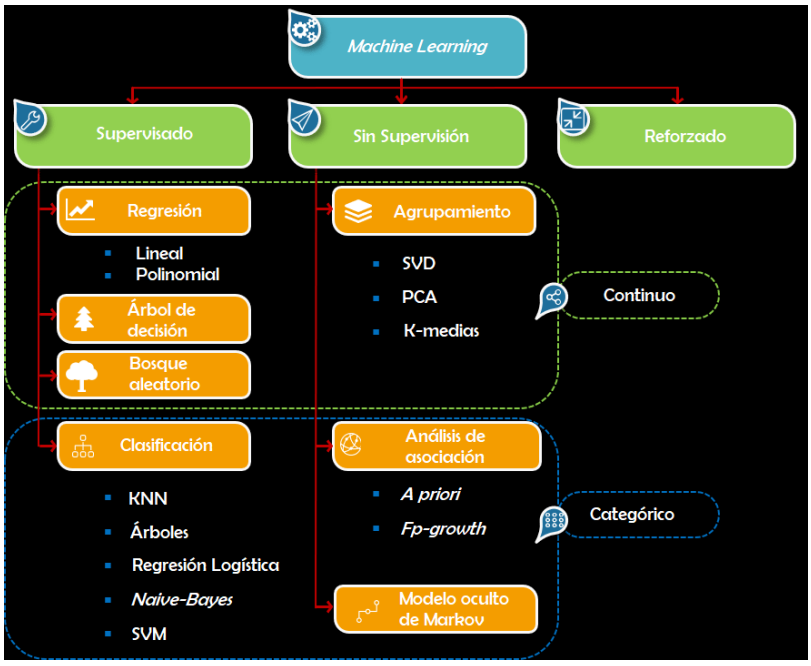
Supongamos que presentamos imágenes de manzanas, plátanos y mangos al modelo, de modo que lo que hace, en función de algunos patrones y

relaciones, en crear grupos y dividir el conjunto de datos en esos grupos. Ahora, si se alimenta un nuevo dato al modelo, se agrega a uno de los clústeres creados.

2.4.2.3. Aprendizaje reforzado

Es la capacidad de un agente para interactuar con el entorno y descubrir cuál es el mejor resultado. Sigue el concepto de hit y método de prueba. El agente es recompensado o penalizado con un punto por una respuesta correcta o incorrecta, y sobre la base de los puntos de recompensa positivos obtenidos, el modelo se entrena a sí mismo. Una vez más entrenado, se prepara para predecir los nuevos datos que se le presentan.

Figura 3. **Diagrama de tipos de Machine Learning**



Fuente: elaboración propia

2.4.3. Deep Learning

El aprendizaje profundo es una técnica de aprendizaje automático que enseña a las computadoras a hacer lo que es natural para los humanos: aprender con el ejemplo. Un ejemplo del funcionamiento sería afirmar que es la clave para el control de voz en dispositivos de consumo como teléfonos, tabletas, televisores y altavoces de manos libres.

En el aprendizaje profundo, un modelo de computadora aprende a realizar tareas de clasificación directamente desde imágenes, texto o sonido. Los modelos de aprendizaje profundo pueden lograr una precisión de vanguardia, que a veces supera el rendimiento a nivel humano. Los modelos se entrenan utilizando un gran conjunto de datos etiquetados y arquitecturas de redes neuronales que contienen muchas capas.

2.4.3.1. Cómo trabaja Deep Learning

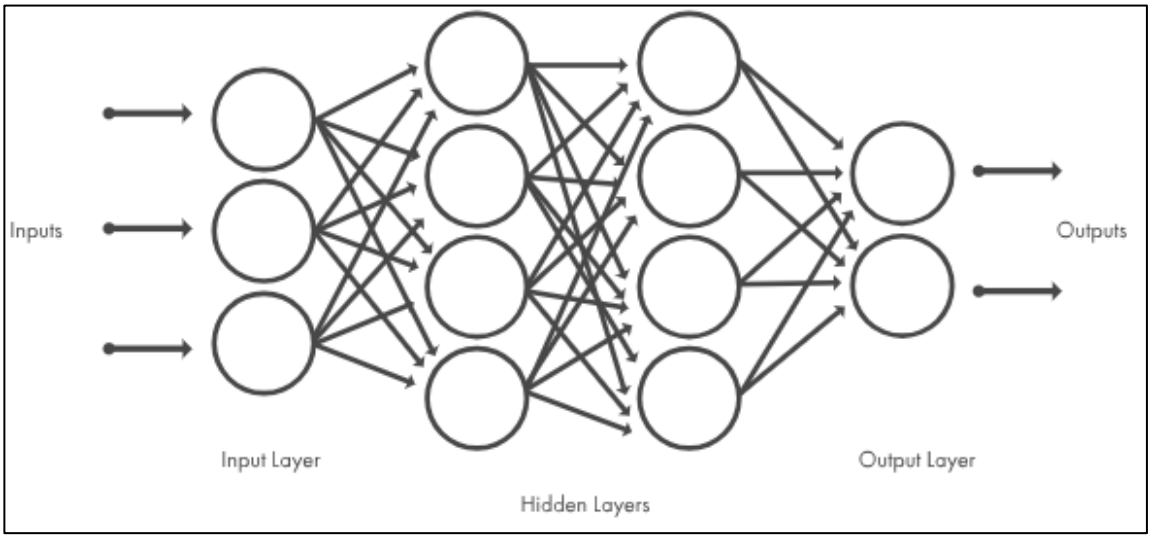
La mayoría de los métodos de aprendizaje profundo utilizan arquitecturas de redes neuronales, por lo que los modelos de aprendizaje profundo a menudo se conocen como redes neuronales profundas.

El término profundo generalmente se refiere al número de capas ocultas en la red neuronal. Las redes neuronales tradicionales solo contienen 2-3 capas ocultas, mientras que las redes profundas pueden tener hasta 150.

Los modelos de aprendizaje profundo se capacitan mediante el uso de grandes conjuntos de datos etiquetados y arquitecturas de redes neuronales que aprenden las características directamente de los datos sin la necesidad de

una extracción manual de las mismas. Las redes pueden tener decenas o cientos de capas ocultas.

Figura 4. **Redes neuronales que se organizan en capas formadas por un conjunto de nodos interconectados**



Fuente: Deep Learning. www.mathworks.com/discovery/deep-learning.html. Consulta: 8 de marzo de 2019.

2.4.3.2. Diferencia entre Machine Learning y Deep Learning

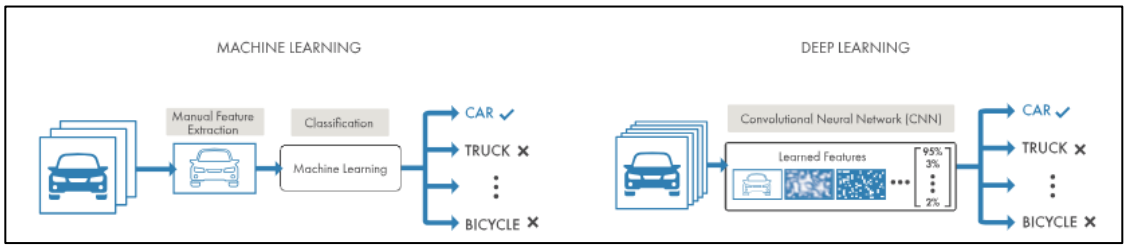
Deep Learning es una forma especializada de Machine Learning. Un flujo de trabajo de aprendizaje automático comienza con las funciones relevantes que se extraen manualmente de las imágenes. Las características se utilizan para crear un modelo que clasifica los objetos en la imagen. Con un flujo de trabajo de aprendizaje profundo, las características relevantes se extraen automáticamente de las imágenes. Además, el aprendizaje en profundidad realiza el aprendizaje de extremo a extremo, en el que a una red se le

proporcionan datos sin procesar y una tarea que realizar, como la clasificación, y aprende cómo hacerlo de forma automática.

Otra diferencia clave es la escala de los algoritmos de aprendizaje profundo con los datos, mientras que el aprendizaje superficial converge. El aprendizaje superficial se refiere a los métodos de aprendizaje automático que se estancan en un cierto nivel de rendimiento cuando agrega más ejemplos y datos de capacitación a la red.

Una ventaja clave de las redes de aprendizaje profundo es que a menudo continúan mejorando a medida que aumenta el tamaño de sus datos.

Figura 5. **Comparación entre un enfoque de aprendizaje automático para categorizar vehículos (izquierda) con aprendizaje profundo (derecha)**



Fuente: Deep Learning. www.mathworks.com/discovery/deep-learning.html. Consulta: 8 de marzo de 2019.

En la figura 5 se puede observar que la categorización de vehículos con Machine Learning involucra una extensión manual de funciones, mientras que con Deep Learning categorizar vehículos depende de las características aprendidas y almacenadas en la red neuronal.

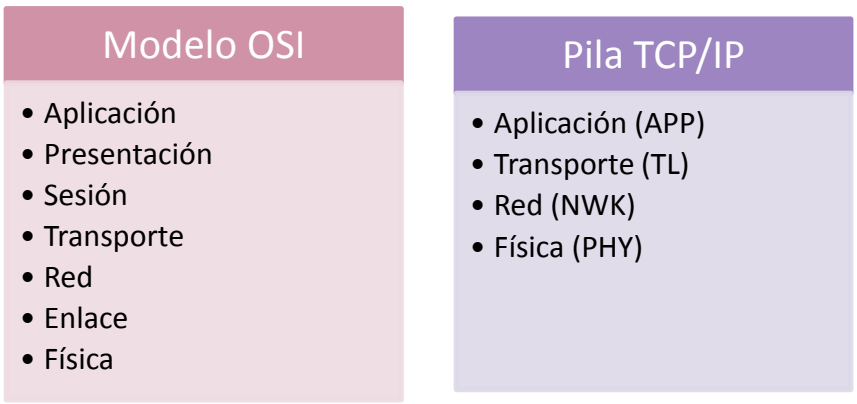
2.5. Protocolos de comunicación

A continuación, se explican los protocolos de comunicación para internet de las cosas.

2.5.1. Modelo TCP/IP aplicado a Internet de las cosas

Un modelo de siete capas fue estandarizado por ISO, TCP e IP, que eran los protocolos predominantemente utilizados, redujeron la arquitectura de siete capas a una de cuatro capas. Sin embargo, la funcionalidad y las características de las capas siguen siendo las mismas.

Figura 6. **Arquitectura OSI y TCP/IP para IoT**



Fuente: elaboración propia

2.5.2. Capa física

La capa física juega un papel vital en el establecimiento del canal de comunicación. La importancia de Internet de las cosas involucra los siguientes rasgos:

- Bajo consumo de energía
- Batería de larga duración
- Bajo consumo de ancho de banda
- Dispositivos más pequeños y livianos
- Capacidad para conectar y operar más dispositivos en un solo entorno

Las características anteriores se pueden lograr con estándares de capa física efectivos. IEEE 802.15.4 (ZigBee, 6LoWPAN, WirelessHART, Mi-Wi), IEEE 802.15.1 (Bluetooth Low Energy (BLE) - Bluetooth 4.0), Near Field Communication (NFC), entre otros. son los estándares establecidos por organismos específicos tales como IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) y proveedores propietarios (Z-Wave por SIGMA DESIGNS).

2.5.3. Capa de red

Una vez que se ha establecido la conectividad física, tiene que haber un método único para diferenciar los dispositivos que operan en su propio rango. La dirección de red desempeña un papel vital en la identificación de cada PC conectada al mismo enrutador, similar a los números únicos o ID que representan los buses que salen de la terminal al mismo tiempo.

En IoT, cada alianza tiene su propia dirección de red. Por ejemplo, ZigBee es una alianza con sus propias direcciones de red. Del mismo modo, BLE y Z-Wave tienen sus propias direcciones de red respectivas a sus entornos.

Los dispositivos wifi vienen con la pila de IP en su chip, lo que permite la conectividad basada en IP. La capa IP ayuda a los dispositivos respectivos a comunicarse de manera efectiva dentro de su rango operativo.

Los dispositivos 6LoWPAN (red de área personal inalámbrica de baja potencia IPv6) también funcionan en IEEE 802.15.4, pero tienen la pila de red con conectividad IP (IPv6). Como analogía, considera los controles remotos de TV que podrían operarse en Internet.

2.5.3.1. IEEE 802.15.4 MAC

La pila de software IEEE 802.15.4 MAC proporciona funciones básicas de transporte de datos, API, punto a punto y red en estrella (transmisión de salto único) para una variedad de plataformas inalámbricas 802.15.4.

2.5.3.2. Ethernet

Ethernet es la tecnología de red de área local (LAN) más utilizada. Proporciona una comunicación por cable para conectar los dispositivos a Internet. Es un protocolo físico y de capa de enlace en la pila TCP / IP que describe cómo los dispositivos en red comparten sus datos con la computadora u otros dispositivos de red a través del medio físico. Se basa en el estándar IEEE 802.3. Dentro de un sistema IoT, Ethernet puede usarse para conectar dispositivos IoT fijos. El cable Ethernet sirve como medio cableado para conectar la computadora y el enrutador. Este es un ejemplo simple de la tecnología Ethernet LAN.

La velocidad de los datos a través de Ethernet depende del tipo de cable y puede ser limitada por el administrador de la red. Puede haber cables de fibra óptica, coaxiales o de par trenzado utilizados para redes Ethernet. Ethernet tiene una latencia muy baja, lo que lo hace adecuado para aplicaciones IoT de misión crítica en las que los dispositivos pueden ubicarse en una red de área local.

2.5.3.3. Wifi

La conectividad wifi es una opción obvia para los desarrolladores, especialmente en entornos domésticos y LAN. Proporciona una transferencia de datos rápida y en grandes cantidades. Es probable que su consumo de energía sea demasiado alto para muchas aplicaciones de IoT.

- Frecuencias: bandas de 2,4 GHz y 5 GHz
- Alcance: aproximadamente 50 m.
- Velocidad de datos: 150-200 Mbps es típico, máximo 600 Mbps, la última 802,11-ac ofrece 500 Mbps a 1 Gbps.

2.5.4. 6LowPAN

6LowPAN significa IPv6 Red de área personal inalámbrica de baja potencia. Es un protocolo de red y puede usarse en bandas industriales, científicas y médicas de Ethernet, wifi, 802.15.4 y sub-1GHz. Un atributo clave es la pila de IPv6, que ha sido importante para habilitar IoT. IPv6 es el sucesor de IPv4 y permite que cualquier objeto en el mundo se conecte a Internet con su propia dirección IP única. Ha sido diseñado para la automatización de viviendas y edificios, y es un mecanismo de transporte que conecta sistemas de control complejos con dispositivos a través de una red inalámbrica de bajo consumo.

2.5.5. IPV4 e IPV6

A continuación, se explica la composición de los protocolos:

2.5.5.1. IPV4

Una dirección IPv4 es un número de 32 bits formado por cuatro octetos (números de 8 bits) en una notación decimal, separados por puntos. Un bit puede ser tanto un 1 como un 0 (2 posibilidades), por lo tanto, la notación decimal de un octeto tendría 2 elevado a la 8a potencia de distintas posibilidades (256 de ellas, para ser exactos). Ya que nosotros empezamos a contar desde el 0, los posibles valores de un octeto en una dirección IP van de 0 a 255. Para entender por qué el espacio de direcciones IPv4 es limitado a 4,3 mil millones de direcciones, podemos descomponer una dirección IPv4 al multiplicar los números de los 32 bits: 256*256*256*256 nos da un total 4 294 967 296 direcciones.

Algunos ejemplos de direcciones IPv4 son:

- 192.168.0.1
- 66.228.118.51
- 173.194.33.16

2.5.5.2. IPV6

Las direcciones IPv6 están basadas en 128 bits. Con la misma matemática anterior, tenemos 2 elevado a la 128a potencia para encontrar el total de direcciones IPv6, 340, 282, 366, 920, 938, 463, 463, 374, 607, 431, 768, 211, 456 direcciones. Ya que el espacio en IPv6 es mucho más extenso que el IPv4 sería muy difícil definir el espacio con notación decimal; se tendría 2 elevado a la 32a potencia en cada sección.

Para permitir el uso de esa gran cantidad de direcciones más fácilmente, IPv6 está compuesto por ocho secciones de 16 bits, separadas por dos puntos (:), ya que cada sección es de 16 bits, tenemos 2 elevado a la 16 de variaciones (las cuales son 65,536 distintas posibilidades). Usando números decimales de 0 a 65,535, tendríamos representada una dirección bastante larga. Para facilitarlo, las direcciones IPv6 están expresadas con notación hexadecimal (16 diferentes caracteres: 0-9 y a-f).

Algunos ejemplos de direcciones IPV6 son:

- 2607: f0d0: 4545: 3: 200: f8ff: fe21: 67cf
- 2001: db8: 85a3: 0:0 :8a2e: 370: 7334

2.5.6. Internet Protocol Security (Ipsec)

Perfecciona la arquitectura de los servicios de seguridad para el tráfico de redes IP. IPsec describe el marco para proporcionar seguridad en la capa IP, así como el conjunto de protocolos diseñados para proporcionar esa seguridad, a través de la autenticación y el cifrado de los paquetes de red IP. En IPsec también se incluye protocolos que definen los algoritmos criptográficos utilizados para cifrar, descifrar y autenticar paquetes, así como los protocolos necesarios para el intercambio y administración seguro de claves.

IPsec definió originalmente dos mecanismos para imponer seguridad en los paquetes IP:

- El protocolo Encapsulating Security Payload (ESP), que definía un método para cifrar datos en paquetes IP; y el protocolo de encabezado

de autenticación (AH), que definía un método para firmar digitalmente paquetes IP.

- El protocolo de intercambio de claves de Internet (IKE) se usa para administrar las claves criptográficas utilizadas por los hosts para IPsec.

2.5.6.1. Cómo trabaja Ipsec

- El primer paso ocurre cuando un host reconoce que un paquete debe ser transmitido usando IPsec.

- El segundo paso en el proceso, llamado fase 1 de IKE, permite que los dos hosts que utilizan IPsec negocien los conjuntos de políticas que utilizan para el circuito seguro, se autentiquen entre sí e inicien un canal seguro entre los dos hosts.

- El tercer paso para configurar un circuito IPsec es la fase 2 de IKE, que a su vez se realiza a través de la configuración del canal seguro en la fase 1. IKE.

- El cuarto paso de la conexión es el intercambio real de datos a través del túnel cifrado IPsec recién creado. Desde este punto, los paquetes se cifran y descifran mediante los dos puntos finales con la configuración de SA de IPsec en los tres pasos anteriores.

- El último paso es la terminación del túnel de IPsec, generalmente cuando se completa la comunicación entre los hosts, cuando se agota el tiempo de espera de la sesión o cuando se pasa un número de bytes previamente especificado a través del túnel de IPsec.

2.5.7. Capa de transporte

La capa de transporte desempeña el papel de cajas fuertes avanzadas y mecanismos de bloqueo en redes e intercambios de datos. Las redes IoT, al ser de baja potencia, son fácilmente propensas a los ataques. La capa de transporte debe estar integrada con características de seguridad eficientes, además de ser responsable del consumo efectivo de ancho de banda y el mantenimiento de la sesión.

Las aplicaciones desarrolladas en la parte superior de la capa de transporte tienen que seleccionar el protocolo de capa de transporte adecuado para la potencia efectiva y la preservación del ancho de banda.

2.5.7.1. User Datagram Protocol (UDP)

User Datagram Protocol es un protocolo de comunicaciones alternativo al Protocolo de Control de Transmisión (TCP) que se utiliza principalmente para establecer conexiones de baja latencia y de tolerancia a las pérdidas entre las aplicaciones de Internet. UDP se ejecuta sobre el Protocolo de Internet (IP) y, a veces, se denominan UDP / IP. Algunas características de UDP son:

2.5.7.1.1. Aplicaciones de UDP

- UDP es un protocolo ideal para aplicaciones de red en las que la latencia percibida es crítica, como en los juegos y las comunicaciones de voz y video, que pueden sufrir alguna pérdida de datos sin afectar negativamente la calidad percibida.

- UDP también se puede usar en aplicaciones que requieren transmisión de datos sin pérdida cuando la aplicación está configurada para administrar el proceso de retransmisión de paquetes perdidos y organizar correctamente los paquetes recibidos.

- UDP trabaja en conjunto con los protocolos de nivel superior para ayudar a administrar los servicios de transmisión de datos, incluyendo el Protocolo de transferencia de archivos trivial (TFTP), el Protocolo de transmisión en tiempo real (RTSP), el Protocolo de red simple (SNP) y las búsquedas del sistema de nombres de dominio (DNS).

2.5.7.2. Transmition Control Protocol (TCP)

TCP es un estándar que define cómo establecer y mantener una conversación de red a través de la cual los programas de aplicación pueden intercambiar datos. TCP funciona con el Protocolo de Internet (IP), que define cómo las computadoras se envían paquetes de datos entre sí.

TCP / IP especifica cómo se intercambian los datos a través de Internet al proporcionar comunicaciones de extremo a extremo que identifican cómo deben dividirse en paquetes, direccionarse, transmitirse, enrutarse y recibirse en el destino. TCP / IP requiere poca administración central, y está diseñado para que las redes sean confiables, con la capacidad de recuperarse automáticamente de la falla de cualquier dispositivo en la red.

TCP define cómo las aplicaciones pueden crear canales de comunicación a través de una red. También administra cómo un mensaje se ensambla en paquetes más pequeños antes de que luego se transmitan a través de Internet y se vuelvan a ensamblar en el orden correcto en la dirección de destino.

2.5.7.3. Datagram Transport Layer Security (DTLS)

DTLS es un protocolo basado en TLS que es capaz de asegurar el transporte de datagramas. DTLS es muy adecuado para proteger aplicaciones y servicios que son sensibles al retardo (y por lo tanto utilizan el transporte de datagramas), aplicaciones de túneles como VPN y aplicaciones que tienden a quedarse sin descriptores de archivos o buffers de socket.

2.5.8. Capa de aplicación

Se muestran los protocolos utilizados en la capa de aplicación:

2.5.8.1. Constrained Application Protocol (CoAP)

Es un protocolo que especifica cómo los dispositivos de baja potencia con limitaciones de cómputo pueden operar en el Internet de las cosas (IoT)

CoAP está diseñado para permitir que los dispositivos simples y restringidos se unan a la IoT incluso a través de redes restringidas con bajo ancho de banda y baja disponibilidad. El protocolo se utiliza generalmente para la comunicación máquina a máquina (M2M).

CoAP funciona como una especie de HTTP para dispositivos restringidos. Permite que equipos de nivel de componentes como sensores o actuadores se comuniquen en la IoT, y que sean controlados y transmitan sus datos como parte de un sistema. El protocolo está diseñado para ser confiable en bajo ancho de banda y alta congestión a través de su bajo consumo de energía y bajo costo de la red. Un ejemplo de ello sería que en una red con conectividad limitada o con mucha congestión, el CoAP puede continuar funcionando cuando

los protocolos basados en TCP, como MQTT, no logran completar un protocolo de enlace.

2.5.8.2. Message Queuing Telemetry Transport (MQTT)

MQTT significa MQ Telemetry Transport. Es un protocolo de mensajería de publicación y suscripción, extremadamente simple y liviano, diseñado para dispositivos restringidos y redes de bajo ancho de banda, alta latencia o poco confiables. Los principios de diseño son minimizar el ancho de banda de la red y los requisitos de recursos del dispositivo, al mismo tiempo que intentan garantizar la confiabilidad y cierto grado de seguridad de la entrega. Estos principios también hacen que el protocolo sea ideal para la emergente máquina a máquina (M2M) o Internet de las cosas del mundo de los dispositivos conectados, y para las aplicaciones móviles donde el ancho de banda y la energía de la batería son importantes.

2.5.8.3. Extensible Messaging and Presence Protocol (XMPP)

Es un protocolo basado *en* Extensible Markup Language (XML) y destinado a la mensajería instantánea (IM) y la detección de presencia en línea. Funciona entre servidores o entre ellos, y facilita el funcionamiento casi en tiempo real. El protocolo puede eventualmente permitir que los usuarios de Internet envíen mensajes instantáneos a cualquier otra persona en Internet, independientemente de las diferencias en los sistemas operativos y los navegadores.

Es un protocolo para transmitir elementos XML a través de una red con el fin de intercambiar mensajes e información de presencia casi en tiempo real. Este protocolo es utilizado principalmente por aplicaciones de mensajería instantánea como WhatsApp.

Estos son los requisitos básicos de cualquier Instant Messenger que cumple XMPP:

- Envía y recibe mensajes con otros usuarios.
- Verifica y comparte el estado de presencia.
- Gestiona las suscripciones desde otros usuarios.
- Gestiona lista de contactos.
- Bloquea las comunicaciones (recibir mensajes, compartir el estado de presencia, etc.) a usuarios específicos.

2.5.8.4. Advanced Message Queuing Protocol (AMQP)

Este protocolo es un estándar publicado de código abierto para mensajería asíncrona por cable. Permite la mensajería cifrada e interoperable entre las organizaciones y las aplicaciones. El protocolo se utiliza en la mensajería cliente hacia servidor y en la gestión de dispositivos de IoT.

AMPQ es un protocolo eficiente, portátil, multicanal y seguro. El protocolo binario ofrece autenticación y cifrado por medio de SASL o TLS, confiando en un protocolo de transporte como TCP. El protocolo de mensajería es rápido y ofrece una entrega garantizada con acuse de recibo de los mensajes recibidos. AMPQ funciona bien en entornos de múltiples clientes y proporciona un medio para delegar tareas y hacer que los servidores manejen las solicitudes

inmediatas más rápido. Debido a que AMPQ es un sistema de mensajería binaria con un comportamiento de mensajería muy estricto, la interoperabilidad de los clientes de diferentes proveedores está asegurada.

2.5.9. Capa de aplicación

La capa de aplicación es la capa de abstracción más alta del modelo TCP/IP que proporciona las interfaces y los protocolos que necesitan los usuarios. Combina las funcionalidades de la capa de sesión, la capa de presentación y la capa de aplicación del modelo OSI.

Las funciones de la capa de aplicación son:

- Facilitar al usuario utilizar los servicios de la red.
- Se utiliza para desarrollar aplicaciones basadas en red.
- Proporcionar servicios de usuario como inicio de sesión de usuario, dispositivos de red, nombres, mensajes de formato y correos electrónicos, transferencia de archivos, entre otros.
- Manejo de errores y la recuperación del mensaje en su totalidad.

A continuación, se hace mención a los protocolos utilizados en web que permiten la interacción del usuario con la capa de aplicación:

2.5.9.1. Hypertext Transfer Protocol (HTTP)

El protocolo de transferencia de hipertexto es el conjunto de reglas para transferir archivos, ya sea de texto, imágenes gráficas, sonido, video y otros archivos multimedia a la World Wide Web. Tan pronto como un usuario web abre su navegador web, hacen uso indirecto de HTTP.

HTTP es un protocolo de aplicación que se ejecuta sobre el conjunto de protocolos TCP / IP (los protocolos básicos para Internet).

Los conceptos HTTP incluyen la idea de que los archivos pueden contener referencias a otros archivos cuya selección provocará solicitudes de transferencia adicionales. Cualquier máquina de servidor web contiene, además de los archivos de página web que puede servir, un *daemon* HTTP, un programa que está diseñado para esperar solicitudes HTTP y manejarlas cuando llegan. Su navegador web es un cliente HTTP, que envía solicitudes a las máquinas servidor. Cuando el usuario del navegador ingresa las solicitudes de archivos, ya sea "abriendo" un archivo web o haciendo clic en un enlace de hipertexto, el navegador genera una solicitud HTTP y la envía a la dirección del Protocolo de Internet (dirección IP) indicado por la URL.

El *daemon* HTTP en la máquina del servidor de destino recibe la solicitud y devuelve el archivo solicitado o los archivos asociados con la solicitud.

2.5.9.2. Dynamic Host Configuration Protocol (DHCP)

El Protocolo de configuración dinámica de host es un protocolo de cliente y servidor que proporciona automáticamente un host de Protocolo de Internet (IP) con su dirección IP y otra información de configuración relacionada, como la máscara de subred y la puerta de enlace predeterminada.

Se utiliza un servidor DHCP para emitir direcciones IP únicas y configurar automáticamente otra información de red. En la mayoría de los hogares y pequeñas empresas, el router actúa como el servidor DHCP. En redes grandes, una sola computadora puede actuar como el servidor DHCP.

El proceso es el siguiente: un dispositivo “cliente” solicita una dirección IP de un enrutador host, después de lo cual el *host* asigna una dirección IP disponible para permitir que el cliente se comunique en la red.

Una vez que un dispositivo se enciende y se conecta a una red que tiene un servidor DHCP, enviará una solicitud al servidor, llamada solicitud *DHCPDISCOVER*. Cuando el paquete *DISCOVER* llega al servidor DHCP, el servidor intenta mantener una dirección IP que el dispositivo puede usar, y luego ofrece al cliente la dirección con un paquete DHCPOFFER.

Una vez que se ha realizado la oferta para la dirección IP elegida, el dispositivo responde al servidor DHCP con un paquete DHCPREQUEST para aceptarlo. Después el servidor envía un ACK que se utiliza para confirmar que el dispositivo tiene esa dirección IP específica y para definir la cantidad de tiempo que el dispositivo puede usar la dirección antes de obtener una nueva. Si el servidor decide que el dispositivo no puede tener la dirección IP, enviará un NACK.

2.5.9.3. Domain Name Systems (DNS)

El sistema de nombres de dominio es la forma en que los nombres de dominio de Internet se ubican y traducen a direcciones de protocolo de Internet (IP). El sistema de nombres de dominio asigna el nombre que la gente usa para ubicar un sitio web a la dirección IP que usa una computadora para ubicar un sitio web. Por ejemplo, si alguien escribe TechTarget.com en un navegador web, un servidor detrás de escena asignará ese nombre a la dirección IP 206.19.49.149.

La navegación web y la mayoría de las actividades de Internet dependen del DNS para proporcionar rápidamente la información necesaria para conectar a los usuarios a los hosts remotos. La asignación de DNS se distribuye a través de Internet en una jerarquía de autoridad. Los proveedores de acceso y las empresas, así como los gobiernos, universidades y otras organizaciones, suelen tener sus propios rangos de direcciones IP asignados y un nombre de dominio asignado; también suelen ejecutar servidores DNS para gestionar la asignación de esos nombres a esas direcciones. La mayoría de las direcciones URL se crean alrededor del nombre de dominio del servidor web que toma las solicitudes de los clientes.

2.5.9.4. TLS/SSL

A continuación, se muestra la definición de los protocolos de las capas de transportes:

2.5.9.4.1. Transport Layer Security (TLS)

Transport Layer Security, o TLS, es un protocolo de seguridad ampliamente adoptado diseñado para facilitar la privacidad y la seguridad de los datos para las comunicaciones a través de Internet. Un caso de uso principal de TLS es cifrar la comunicación entre las aplicaciones web y los servidores, como los navegadores web que cargan un sitio web. TLS también se puede utilizar para cifrar otras comunicaciones, como correo electrónico, mensajería y voz sobre IP (VOIP).

Una conexión TLS se inicia utilizando una secuencia conocida como el protocolo de enlace TLS *handshake*. Este establece una suite de cifrado para

cada sesión de comunicación. El conjunto de códigos cifrados es un conjunto de algoritmos que especifica detalles, como qué claves de cifrado compartidas o claves de sesión se utilizarán para esa sesión en particular. TLS puede establecer las claves de sesión coincidentes en un canal no cifrado gracias a una tecnología conocida como criptografía de clave pública.

El *handshake* también maneja la autenticación, que generalmente consiste en que el servidor demuestre su identidad al cliente. Esto se hace con claves públicas. Estas son claves de encriptación que usan encriptación unidireccional, lo que significa que cualquier persona puede descifrar los datos encriptados con la clave privada para asegurar su autenticidad, pero solo el remitente original puede cifrar los datos con la clave privada. Una vez que los datos se cifran y autentican, se firman con un código de autenticación de mensaje (MAC).

2.5.9.4.2. Secure Sockets Layer (SSL)

Secure Sockets Layer es un protocolo estándar utilizado para la transmisión segura de documentos a través de una red. Desarrollada por Netscape, la tecnología SSL crea un enlace seguro entre un servidor web y un navegador para garantizar la transmisión de datos privada e integral. SSL utiliza el Protocolo de control de transporte (TCP) para la comunicación.

En SSL, el socket se refiere al mecanismo de transferencia de datos entre un cliente y un servidor a través de una red. Cuando se utiliza SSL para transacciones seguras en Internet, un servidor web necesita un certificado SSL para establecer una conexión SSL segura. SSL encripta segmentos de conexión de red sobre la capa de transporte, que es un componente de conexión de red sobre la capa de programa.

SSL sigue un mecanismo criptográfico asimétrico, en el que un navegador web crea una clave pública y una clave privada (secreta). La primera se coloca en un archivo de datos conocido como una solicitud de firma de certificado (CSR). La segunda se emite solo al destinatario.

2.5.10. Formato de datos

A continuación, se describen los tipos de formatos:

2.5.10.1. Formato de datos en web

Se muestran los principales formatos utilizados para desarrollo web:

2.5.10.1.1. HyperText Markup Language (HTML)

HTML es un lenguaje informático diseñado para permitir la creación de sitios web. Estos pueden ser vistos por cualquier persona conectada a Internet.

El HTML consiste en una serie de códigos cortos escritos por el autor del sitio en un archivo de texto: estas son las etiquetas. Luego, el texto se guarda como un archivo html y se visualiza a través de un navegador, como Internet Explorer o Netscape Navigator. Este navegador lee el archivo y traduce el texto a una forma visible, con la esperanza de representar la página como el autor había querido. Escribir un HTML implica usar etiquetas correctamente para crear la visión. Se puede utilizar cualquier editor para redactarlo, desde un editor de texto rudimentario hasta un editor gráfico potente para crear páginas HTML.

2.5.10.1.2. Extensible Markup Language (XML)

Se utiliza para describir datos. El estándar XML es una forma flexible de crear formatos de información y compartir electrónicamente datos estructurados a través de la Internet pública, así como a través de redes corporativas.

El código XML, una recomendación formal del World Wide Web Consortium (W3C), es similar al lenguaje de marcado de hipertexto (HTML). Tanto XML como HTML contienen símbolos de marcado para describir el contenido de la página o el archivo. El código HTML describe el contenido de la página web (principalmente texto e imágenes gráficas) solo en términos de cómo se debe mostrar e interactuar.

El componente básico de un documento XML es un elemento definido por etiquetas, que tiene un comienzo y una etiqueta final. Todos los elementos en un documento XML están contenidos en un elemento más externo conocido como el elemento raíz. XML también puede soportar elementos anidados, o elementos dentro de elementos. Esta capacidad permite que XML soporte estructuras jerárquicas. Los nombres de los elementos describen el contenido del elemento y la estructura describe la relación entre los elementos.

2.5.10.1.3. JavaScript Object Notation (JSON)

Es un formato ligero de intercambio de datos. Es fácil de leer y escribir para los humanos; las máquinas, es fácil de analizar y generar. Se basa en un subconjunto del lenguaje de programación de JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza

convenciones que son familiares para los programadores de la familia C de lenguajes, incluida la C, C ++, C #, Java, JavaScript, Perl, Python y muchos otros. Estas propiedades hacen de JSON un lenguaje ideal para el intercambio de datos.

JSON está construido sobre dos estructuras:

- Una colección de pares de nombre / valor. En varios idiomas, esto se realiza como un objeto, registro, estructura, diccionario, tabla hash, lista con clave o matriz asociativa.

- Una lista ordenada de valores. En la mayoría de los idiomas, esto se realiza como una matriz, vector, lista o secuencia.

Son estructuras de datos universales. Prácticamente, todos los lenguajes de programación modernos los soportan de una forma u otra. Tiene sentido que un formato de datos que sea intercambiable con los lenguajes de programación también se base en estas estructuras.

2.5.11. Formatos en IoT

A continuación, se explican los formatos para el manejo de datos en internet de las cosas:

2.5.11.1. JSON para IoT

JSON se suele utilizar junto con los protocolos de IoT que no proporcionan soporte nativo para la serialización de la estructura de datos, como HTTP / Rest, WebSockets, MQTT y SMQ.

La mayoría de los protocolos de IoT utilizan TCP / IP como mecanismo de transporte. Es un protocolo basado en flujo que no incluye ninguna información de trama cuando se utiliza para enviar mensajes a través del cable. Los protocolos IoT agregan información de trama sobre TCP / IP cuando se transmiten datos, lo que facilita el envío de paquetes a través del cable. Por ejemplo, el protocolo WebSocket agrega un encabezado de tamaño a los datos, y una pila WebSocket proporciona una API basada en paquetes para el diseñador de la aplicación que usa la pila. Los protocolos de publicación / suscripción, como MQTT y SMQ, también proporcionan una API basada en paquetes.

Como los protocolos de IoT proporcionan una API basada en paquetes, cualquier codificador / decodificador JSON se puede usar para serializar y deserializar los datos estructurados enviados a través del cable. Sin embargo, en algunos casos, un protocolo IoT es una exageración que puede agregar memoria innecesaria y sobrecarga de procesamiento. Un diseñador de productos de IoT puede elegir utilizar directamente TCP / IP como la capa de transporte para enviar datos estructurados a través del cable. No se puede utilizar un codificador / decodificador JSON estándar cuando se utiliza una capa de transporte no basada en tramas.

2.5.11.2. Concise Binary Object Representation

La representación concisa de objetos binarios es un formato de serialización de datos binarios basado libremente en JSON. Al igual que JSON, permite la transmisión de objetos de datos que contienen pares nombre-valor, pero de una manera más concisa. Esto aumenta el procesamiento y las velocidades de transferencia a costa de la legibilidad humana.

Entre otros usos, es la capa de serialización de datos recomendada para el conjunto de protocolos CoAP Internet of Things y el formato de datos en el que se basan los mensajes COSE.

2.5.12. Herramientas utilizadas en IoT

Se describen las herramientas principales para el control de IoT.

2.5.12.1. Tensor Flow

TensorFlow es el sistema de segunda generación de Google Brain. La versión 1.0.0 se lanzó el 11 de febrero de 2017. Mientras que la implementación de referencia se ejecuta en dispositivos individuales, TensorFlow puede ejecutarse en múltiples CPU y GPU (con las extensiones opcionales CUDA y SYCL para computación de propósito general en unidades de procesamiento de gráficos). TensorFlow está disponible en Linux, macOS, Windows y plataformas de computación móviles de 64 bits, incluyendo Android e iOS.

Su arquitectura flexible permite el fácil despliegue de cómputo en una variedad de plataformas (CPU, GPU, TPU), y desde computadoras de escritorio hasta clústeres de servidores a dispositivos móviles y de vanguardia.

Los cálculos de TensorFlow se expresan como gráficos de flujo de datos con estado. El nombre TensorFlow deriva de las operaciones que realizan dichas redes neuronales en matrices de datos multidimensionales, que se denominan tensores. Durante la Conferencia de Google I / O en junio de 2016, Jeff Dean declaró que 1,500 repositorios en GitHub mencionaron TensorFlow, de los cuales solo 5 eran de Google.

En enero de 2018, Google anunció TensorFlow 2.0. En marzo de 2018, Google anunció la versión 1.0 de TensorFlow.js para aprendizaje automático en JavaScript y Gráficos TensorFlow para un aprendizaje profundo en gráficos de computadora.

Tensor flow ofrece varias herramientas como:

2.5.12.1.1. Modelo de construcción fácil

TensorFlow ofrece múltiples niveles de abstracción. Permite crear y entrenar modelos utilizando la API de alto nivel de Keras, que facilita el inicio de TensorFlow y el aprendizaje automático.

Si se necesita más flexibilidad, la ejecución ágil permite una iteración inmediata y una depuración intuitiva. Para tareas de capacitación de ML grandes, se usa la API de estrategia de distribución para la capacitación distribuida en diferentes configuraciones de hardware sin cambiar la definición del modelo.

2.5.12.1.2. Robusta producción de ML en cualquier lugar

Ya sea en servidores, dispositivos perimetrales o en la web, TensorFlow permite capacitar e implementar un modelo fácilmente, sin importar qué idioma o plataforma se use.

Se usa TensorFlow Extended (TFX) si se necesita un ducto ML de producción total. Para ejecutar inferencia en dispositivos móviles y de borde, se

usa TensorFlow Lite. Se puede entrenar y despliegan modelos en entornos de JavaScript utilizando TensorFlow.js.

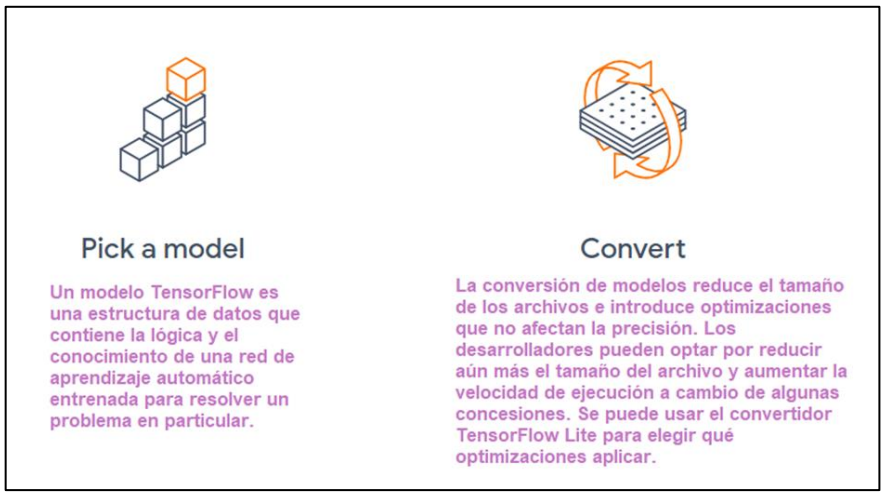
2.5.12.1.3. Potente experimentación para la investigación

Permite crear y entrenar modelos de última generación sin sacrificar la velocidad ni el rendimiento. Brinda la flexibilidad y el control con funciones como la API funcional de Keras y la API de subclasificación de modelos para la creación de topologías complejas. Para una creación de prototipos fácil y una depuración rápida, se usa la ejecución.

TensorFlow también es compatible con un ecosistema de potentes bibliotecas y modelos complementarios para experimentar, incluidos los Tensors Ragged, TensorFlow Probability, Tensor2Tensor y BERT.

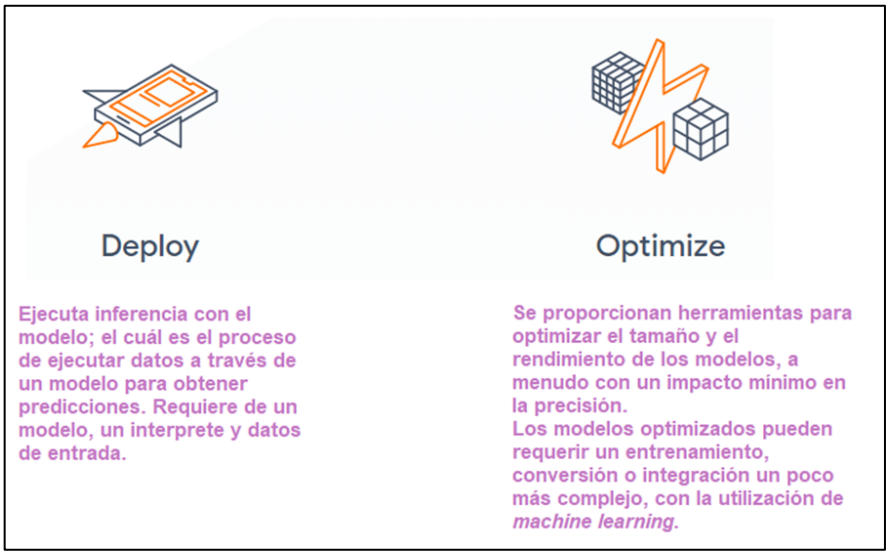
A continuación, se muestra un gráfico del proceso de trabajo de Tensorflow lite en la creación de un modelo aplicado para Internet de las cosas:

Figura 7. **Proceso de trabajo 1 y 2 de tensor Flow para IoT**



Fuente: elaboración propia.

Figura 8. **Proceso de trabajo 3 y 4 de tensor Flow para IoT**



Fuente: elaboración propia.

2.5.12.1.4. Aplicaciones de las herramientas de Tensor Flow Lite para IoT

- Clasificación de imágenes

Utiliza un modelo preentrenado y optimizado para identificar cientos de clases de objetos, incluidas personas, actividades, animales, plantas y lugares.

- Detección de objetos

Detecta múltiples objetos dentro de una imagen, con cuadros delimitadores. Reconoce 80 clases diferentes de objetos.

- Respuesta inteligente

El modelo de respuesta inteligente genera sugerencias de respuesta basadas en mensajes de chat. Las sugerencias pretenden ser respuestas contextuales relevantes, de un solo toque, que ayudan al usuario a responder fácilmente a un mensaje entrante.

2.5.12.2. Amazon IoT

Amazon para Internet de las cosas provee más soluciones para conectar, recabar, almacenar y analizar datos de los dispositivos.

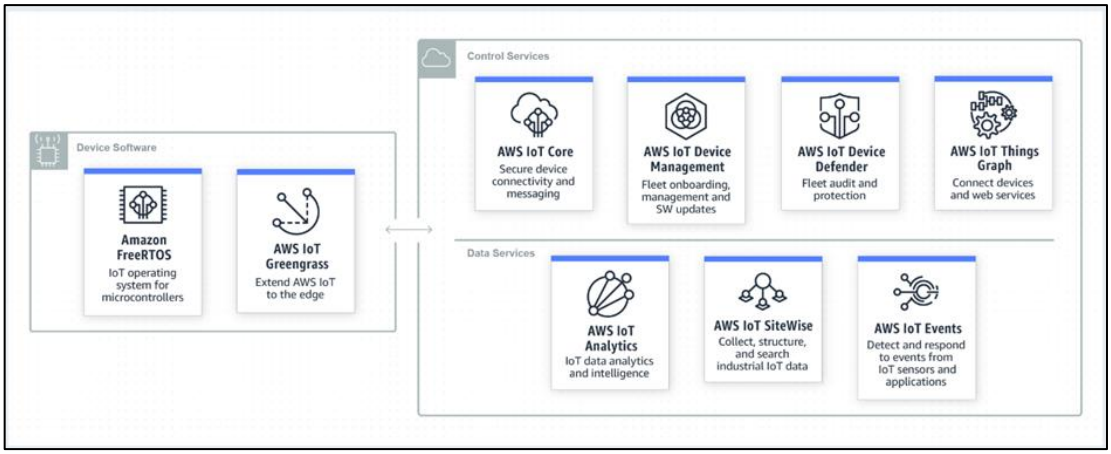
AWS IoT brinda funcionalidad amplia y profunda que amplía el borde hasta la nube, de tal manera que puede crear soluciones IoT para casi cualquier fin en una amplia variedad de dispositivos. Debido a que AWS IoT se integra

con servicios de inteligencia artificial, puede hacer que los dispositivos sean más inteligentes, incluso sin conexión a Internet. Incorporado en la nube de AWS, utilizado por millones de clientes en 190 países, AWS IoT puede escalarse fácilmente a medida que su flota de dispositivos aumenta y sus requisitos comerciales evolucionan. AWS IoT también ofrece las funciones de seguridad más completas para que pueda crear políticas preventivas de seguridad y responder inmediatamente a potenciales problemas de seguridad.

AWS IoT proporciona software de dispositivos, servicios de control y de datos. El software del dispositivo le permite conectar dispositivos, recabar datos y tomar acciones inteligentes a nivel local de manera segura, incluso sin conexión a Internet. Los servicios de control le permiten controlar, administrar y asegurar flotas de dispositivos distintas y de gran tamaño.

Los servicios de datos utilizados para extraer valor de los datos de IoT son los siguientes:

Figura 9. **Servicios de datos de Amazon IoT**



Fuente: AWS. https://aws.amazon.com/es/iot/. Consulta: 19 de julio del 2019.

2.5.12.2.1. Servicios de datos Amazon IoT

- Amazon FreeRTOS

Es un sistema operativo para microcontroladores que facilita la programación, implementación, protección, conexión y administración de los dispositivos de borde pequeños y de poca potencia.

- AWS IoT Greengrass

Es un software que le permite ejecutar capacidades de computación local, mensajería, almacenamiento de datos en caché, sincronización e inferencias de aprendizaje automático en dispositivos conectados de manera segura.

- AWS IoT Core

Permite que los dispositivos conectados interactúen de manera sencilla y segura con las aplicaciones en la nube y otros dispositivos.

- AWS IoT Device Management

Facilita la incorporación, la organización, la monitorización y la administración remota de dispositivos IoT a escala y de forma segura.

- AWS IoT Device Defender

Monitoriza y audita continuamente sus configuraciones de IoT para garantizar que no se aparten de las prácticas recomendadas de seguridad.

- AWS IoT Things Graph

Facilita la conexión de diferentes dispositivos y servicios en la nube para crear aplicaciones IoT.

- AWS IoT Analytics

Facilita la ejecución de análisis sofisticados en volúmenes masivos de datos de IoT.

- AWS IoT SiteWise

Facilita la recopilación, estructuración y búsqueda de datos de IoT que provienen de bases de datos de instalaciones industriales, que luego utiliza para analizar el rendimiento de los equipos y procesos.

- AWS IoT Events

Facilita las tareas de detección y respuesta a eventos que provienen de grandes cantidades de aplicaciones y sensores compatibles con IoT.

2.5.12.3. Azure IoT

Azure es un conjunto completo y en expansión constante de servicios de informática en la nube que ayuda a una organización a afrontar sus desafíos empresariales. Con Azure, la empresa u organización tiene la libertad de crear, administrar e implementar aplicaciones en una red mundial enorme usando las herramientas y los marcos que prefiera.

Azure IoT es un servicio totalmente administrado basado en Azure IoT Hub. Permite implementar las cargas de trabajo de la nube (inteligencia artificial, servicios de Azure y de terceros, o su propia lógica de negocios) para ejecutarlas en dispositivos IoT perimetrales mediante contenedores estándar. Al trasladar determinadas cargas de trabajo al perímetro de la red, los dispositivos pierden menos tiempo en comunicarse con la nube, reaccionan con más rapidez a cambios locales y funcionan de manera confiable incluso durante períodos prolongados sin conexión.

Los servicios que ofrece para este tipo de aplicaciones son los siguientes:

- Supervisión remota

Conecta y supervisa los dispositivos para mejorar los resultados empresariales mediante la automatización de procesos y el análisis de nuevos flujos de datos sobre el equipamiento.

- Mantenimiento predictivo

Anticiparse a las necesidades de mantenimiento y evitan tiempos de inactividad no programados al conectar y supervisar los dispositivos.

Existe una clasificación de las aplicaciones dependiendo el tipo de sistema IoT que se vaya a utilizar, a continuación, se explica algunas de ellas.

- Azure IoT Edge

Suministra la información de la nube en el entorno local, porque implementa y ejecuta inteligencia artificial, servicios de Azure y lógica

personalizada directamente en dispositivos IoT multiplataforma. Ejecuta la solución de IoT de forma segura y a escala, ya sea en la nube o fuera de ella.

- Azure Digital Twins

Se utiliza para la creación de modelos de las relaciones y las interacciones entre personas, lugares y dispositivos. Esta plataforma de inteligencia espacial incluye modelos de objetos gemelos predefinidos y ampliables que ayudan a crear representaciones virtuales del mundo físico y soluciones específicas del sector trabajado que reconozcan el contexto.

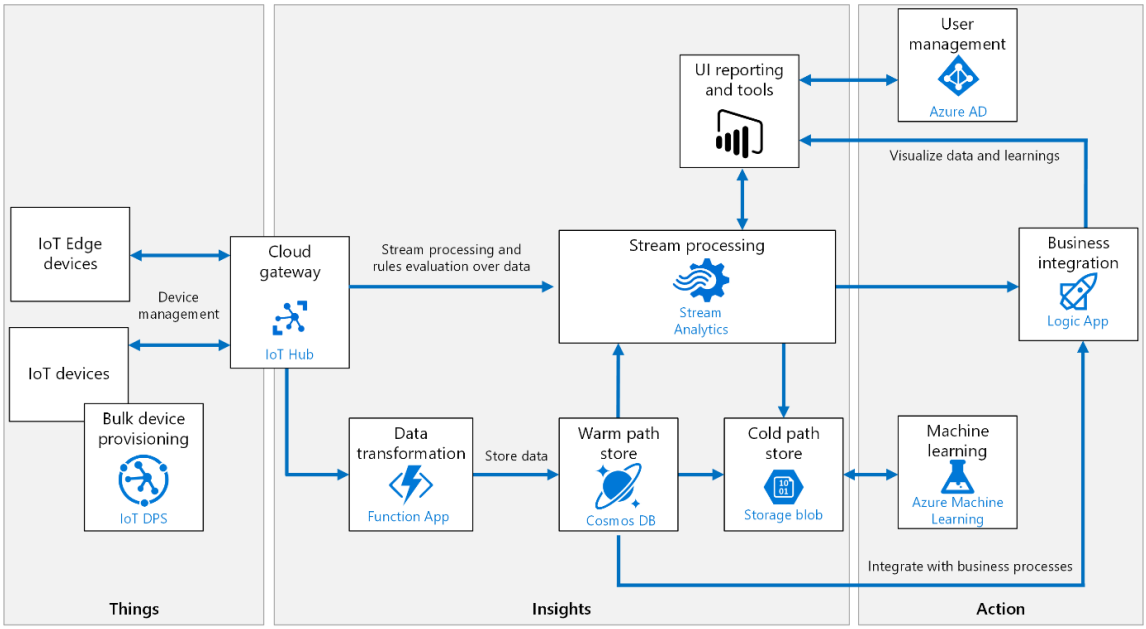
- Azure IoT Central

Esta aplicación provee una solución SaaS (software como servicio) de IoT global totalmente administrada que facilita la conexión, la supervisión y la administración de los recursos de IoT a escala, lo que permite comercializar los productos conectados en menos tiempo.

- Azure IoT Hub

Conecta y controla millones de dispositivos, recursos y sensores de forma segura.

Figura 10. **Modelo de la implementación de un servicio IoT utilizando Azure**



Fuente: Microsoft Azure. https://azure.microsoft.com/es-es/overview/iot/. Consulta: 19 de julio del 2019.

El modelo presenta la clasificación de tres áreas para el funcionamiento de un sistema con Azure, donde se puede ubicar el área de las cosas en la que se encuentran los dispositivos IoT y los encargados del manejo. La parte de percepción en la que Azure Stream Analytics se encarga de realizar el manejo, evaluación y proceso de la información y la parte de acción donde se efectúa el aprendizaje de máquina y la visualización e integración de los datos.

3. SENSORES Y DISPOSITIVOS

3.1. Sensores

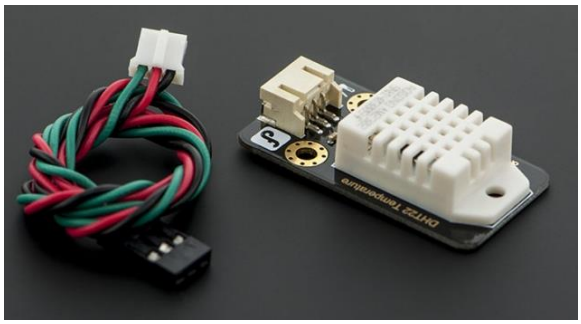
Los siguientes sensores son los transductores encargados de la obtención de los datos en tiempo real. A continuación, se muestran sus características y especificaciones técnicas y el uso que se le dará de forma detallada.

3.1.1. Sensor de temperatura y humedad DHT22

DHT22 es un sensor digital que utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos. Requiere una sincronización cuidadosa para tomar datos. El sensor se conecta con un microcontrolador de alto rendimiento de 8 bits. Cuando el sensor está detectado compensa la temperatura y una cámara precisa y el coeficiente es almacenado en un tipo de programa en la memoria OTP.

Estos sensores serán aplicados en el sistema para la determinación de las condiciones ambientales y climáticas de un sitio en específico, los cuales permitirán actualizar, optimizar y regular las condiciones de temperatura dependiendo el clima, así como mantener un ambiente libre de humedad para la prevención de daños, tanto a nivel estructural de las habitaciones y en la salud de las personas que interactúan con los ambientes.

Figura 11. **DHT22 Módulo de temperatura y humedad**

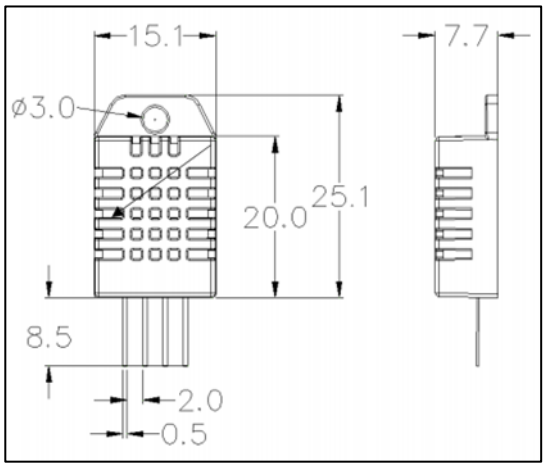


Fuente: DFROBOT. https://www.dfrobot.com/wiki/index.php/DHT22_ Temperature_and_humidity_module_SKU:SEN0137. Consulta: 6 de octubre del 2018.

3.1.1.1. Dimensiones

Todas las dimensiones son en milímetros.

Figura 12. **Dimensiones integrado DHT22**



Fuente: SparkfunElectronics. https://www.sparkfun.com/datasheets/Sensors/ Temperature/DHT22.pdf. Consulta: 6 de octubre del 2018

3.1.1.2. Especificaciones técnicas

A continuación, se muestran los detalles técnicos del sensor:

Tabla II. **Secuencia del número de pines: 1 2 3 4**

Pin	*Función*
1	VDD---- Alimentación
2	DATA---Señal
3	Nulo
4	GND----Tierra

Fuente: SparkfunElectronics. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf. Consulta: 6 de octubre del 2018

Tabla III. **Especificaciones técnicas**

Característica	Dato
Voltaje de alimentación	3.3-6V DC
Señal de salida	Señal digital en un único bus
Elemento sensible	Capacitor de polímeros
Rango de operación	Humedad 0-100%RH; temperatura -40-80 *Celsius*
Precisión	Humedad +-2%RH (Max +-5%RH); temperatura <+- 0.5 Celsius
Resolución o sensibilidad	Humedad 0.1%RH; temperatura 0.1 Celsius
Repetibilidad	Humedad +-1%RH; temperatura +-0.2 *Celsius*
Histéresis de humedad	+-0.3%RH
Estabilidad a largo plazo	+-0.5%RH/año
Periodo de censado	Promedio: 2s
Intercambiabilidad	Completamente intercambiable

Fuente: SparkfunElectronics. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf. Consulta: 6 de octubre del 2018.

3.1.1.3. Especificaciones de operación

A continuación, se describen las especificaciones de trabajo del sensor DHT22.

3.1.1.3.1. Voltaje y pines

El voltaje de alimentación debe ser 3.3-3.6V DC. Cuando el voltaje es aplicado al sensor dentro de un segundo para a un estado inestable. Un capacitor valuado en 100nF puede ser agregado entre VDD y GND para filtrar la onda.

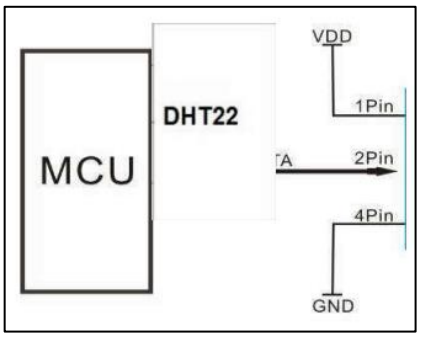
3.1.1.3.2. Comunicación y señal

Un único bus de datos es usado para la comunicación entre MCU y DHT, toma 5mS para iniciar el tiempo de comunicación.

3.1.1.4. Diagrama eléctrico de conexión

A continuación, se muestra el diagrama para conexión analógica de los pines.

Figura 13. **Diagrama de 3 pines**



Fuente: SparkfunElectronics. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf. Consulta: 6 de octubre del 2018.

3.1.2. Módulo embebido de reconocimiento óptico de huella GT-511C1R

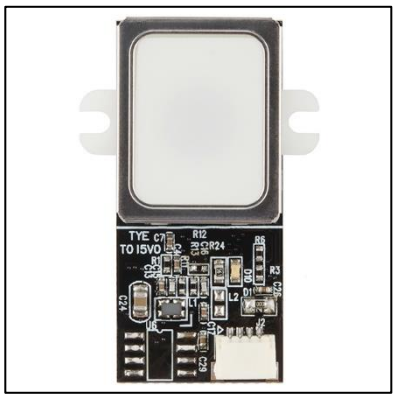
Los módulos serán utilizados como identificadores importantes en la obtención de datos sobre la participación e interacción de las personas. Estos permitirán las siguientes aplicaciones:

- Detección de persona para asistencia y obtención de estadísticas
- Accesos hacia lugares para personal restringido
- Cajas fuertes

Este dispositivo es un chip módulo con:

- Algoritmo de huella digital
- Sensor óptico

Figura 14. **Módulo GT-511C1R**



Fuente: TDROBOTICA. http://tdrobotica.co/lector-de-huella-ttl-5v/355.html. Consulta: 7 de octubre del 2018.

3.1.2.1. Funciones del módulo

- Alta precisión y velocidad en la identificación de huella digital.
- Sensor óptico ultradelgado.
- Verificación 1:1, Identificación 1:N.
- Descarga de la imagen de huella digital desde el dispositivo.
- Lectura y escritura de las plantillas de huella digital desde y hacia el dispositivo.
- Protocolo de comunicación UART & USB.

3.1.2.2. Especificaciones técnicas

La siguiente tabla muestra los datos de fabrica del módulo:

Tabla IV. **Datos técnicos**

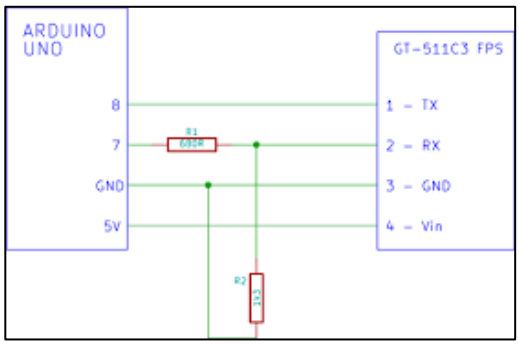
Característica	Dato
CPU	ARM Cortex M3 Core
Sensor	Sensor óptico
Área efectiva del sensor	14*12.5 (mm)
Tamaño de imagen	240*216 pixeles
Resolución	450 dpi
Número máximo de huellas digitales	20 huellas
Tamaño de la plantilla	504 Bytes (plantilla) + 2 bytes (suma de comprobación).
Interfaz de comunicación	UART, tasa de baudios por defecto = 9600 bps después de encendido.
Tasa de aceptación falsa (FAR)	< 0.001%
Tase de rechazo falso (FRR)	< 0.1%
Tiempo de inscripción	< 3 segundos (3 huellas)
Tiempo de identificación	< 1.5 segundos (20 huellas)
Voltaje de operación	DC 3.3 a 6 V
Corriente de operación	< 130mA
Temperatura ambiente de operación	-20 °C - +60°C

Fuente: SparkfunElectronics. https://cdn.sparkfun.com/datasheets/Sensors/Biometric/GT-511C1R_datasheet_V1%205_20140312.pdf. Consulta: 6 de octubre del 2018

3.1.2.3. Conexión con interface de microcontrolador

Cuando se usa un nivel de voltaje de 5V en sus pines, se debe utilizar un convertidor de voltaje que reduce los 5V de salida del microcontrolador al módulo FPS, dado que el módulo FPS puede únicamente utilizar 3.3V en sus pines UARTS. El divisor de voltaje consiste en dos resistencias que pueden ser usadas como convertidor en el módulo.

Figura 15. **Diagrama del convertidor de voltaje**

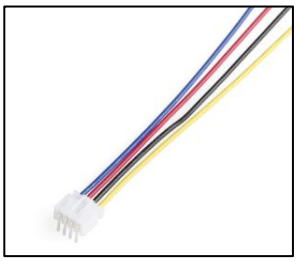


Fuente: STARTING ELECTRONICS. https://startingelectronics.org/articles/GT-511C3-fingerprint-scanner-hardware/. Consulta: 8 de octubre del 2018.

3.1.2.4. Especificaciones del conector

Es un conector de la serie JST-SH de 4 pines cuyo espaciamiento de pines es de 1 mm. Se necesita un conector hembra para conectar cuatro cables al FPS y conectar el módulo a un microcontrolador. Los cables listos pueden ser comprados, o un cable se puede hacer mediante la compra de una carcasa y cables de alambre trenzado.

Figura 16. **Conector JST-SH**

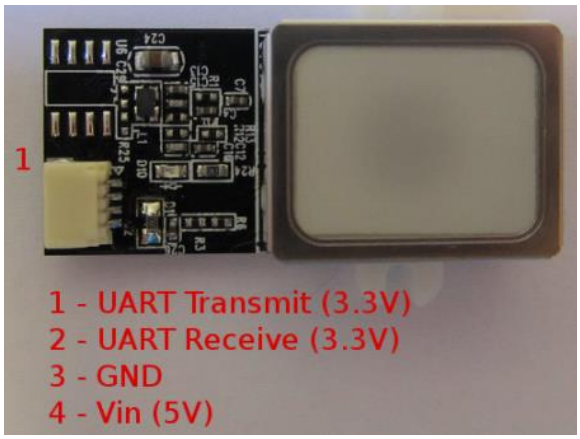


Fuente: SparkfunElectronics. https://www.sparkfun.com/products/9916Consulta: 8 de octubre del 2018.

3.1.2.5. Numeración de pines, funciones y especificaciones

A continuación, en la figura 17 se presenta la numeración de pines, funciones y especificaciones:

Figura 17. **Numeración de pines módulo GT-511C1R**



Fuente: STARTING ELECTRONICS. https://startingelectronics.org/articles/GT-511C3-fingerprint-scanner-hardware/. Consulta: 8 de octubre del 2018

- Pin 1: es el pin transmisor del UART en el FPS (UART Tx) y transmite un estado lógico alto de un máximo de 3.3 V.

- Pin 2: es el pin receptor del UART en el FPS (UART Rx) y puede recibir un estado lógico alto de hasta 3.3V. El nivel de voltaje enviado hacia este pin desde el microcontrolador necesita ser reducido cuando trabaja con microcontroladores de 5 voltios.

- Pin 3: es el pin GND común de 0 voltios en el módulo FPS.

- Pin 4: es la entrada de 5 voltios en el módulo FPS usado para alimentarlo. Este valor puede variar entre 4.5 V y 6 V.

3.1.3. Sensor infrarrojo pasivo HC-SR501

Es un dispositivo para detección de movimiento. Se basa en la medición de la radiación infrarroja. Todos los cuerpos (vivos o no) emiten una cierta cantidad de energía infrarroja, mayor cuanto mayor es su temperatura. Los dispositivos PIR disponen de un sensor piso eléctrico capaz de captar esta radiación y convertirla en una señal eléctrica.

Los sensores PIR serán utilizados para la detección de movimiento y el control del uso de los espacios habitacionales y sociales, para datos estadísticos de aprendizaje y para la activación de la iluminación (el aire acondicionado, entre otras cosas) que permitan realizar un ahorro energético en los sitios donde se puede prescindir de ciertas funciones cuando no se realiza ninguna interacción en la habitación o exteriores.

Figura 18. **Sensor PIR HC-SR501**

Fuente: PROMETEC. https://www.prometec.net/sensor-pir/. Consulta: 8 de octubre del 2018.

En realidad, cada sensor está dividido en dos campos y se dispone de un circuito eléctrico que compensa ambas mediciones. Si ambos campos reciben la misma cantidad de infrarrojos, la señal eléctrica resultante es nula. Por el contrario, si los dos campos realizan una medición diferente, se genera una señal eléctrica.

3.1.3.1. Óptica del sensor

Básicamente es una cúpula de plástico formada por lentes de fresnel que divide el espacio en zonas y enfoca la radiación infrarroja a cada uno de los campos del PIR. De esta manera, cada uno de los sensores capta un promedio de la radiación infrarroja del entorno. Cuando un objeto entra en el rango del sensor, alguna de las zonas marcadas por la óptica recibirá una cantidad distinta de radiación, que será captado por uno de los campos del sensor PIR y disparará la alarma.

Figura 19. **Cúpulas sensor infrarrojo**

Fuente: PROMETEC. https://www.prometec.net/sensor-pir/.Consulta: 8 de octubre del 2018.

3.1.3.2. Especificaciones técnicas

La tabla muestra los parámetros eléctricos:

Tabla V. **Parámetros eléctricos**

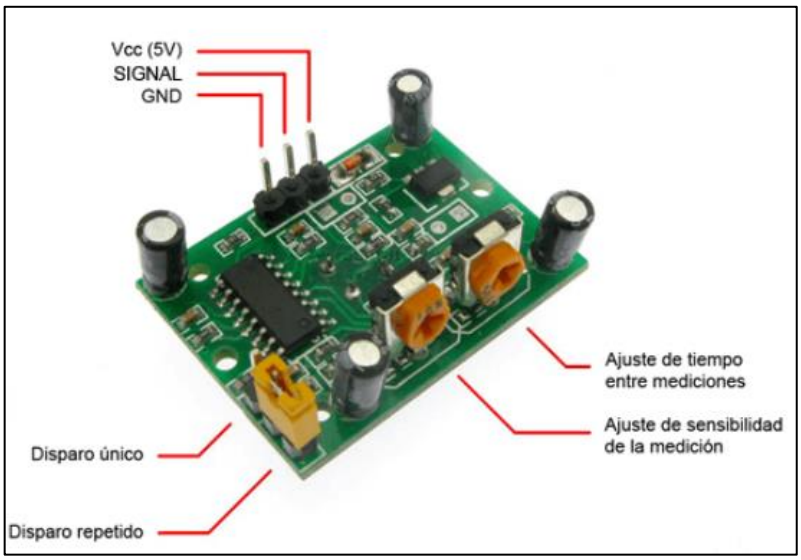
Características	Datos
Rango de voltaje de operación	DC 4,5-20 V
Corriente inactiva	<50uA
Nivel de salida	Alto 3,3 V / Bajo 0 V
Disparo	L no se puede repetir disparo/ H se puede repetir.
Tiempo de retardo	5-200s(ajustable) el rango es (0.xx segundo a decenas de segundo)
Tiempo de bloqueo	2,5 s (por defecto) Puede tener un rango de (0.xx a decenas de segundo)
Dimensiones de la tarjeta	32 mm*24 mm
Ángulo del sensor	<100° ángulo de cono
Temperatura de operación	-15 -+70 grados
Tamaño del lente sensor	Diámetro: 23 mm (por defecto)

Fuente: PROMETEC. https://www.prometec.net/sensor-pir/.Consulta: 8 de octubre del 2018.

3.1.3.3. Esquema eléctrico

El esquema muestra los pines de conexión del sensor PIR.

Figura 20. **Distribución de pines HC-SR501**

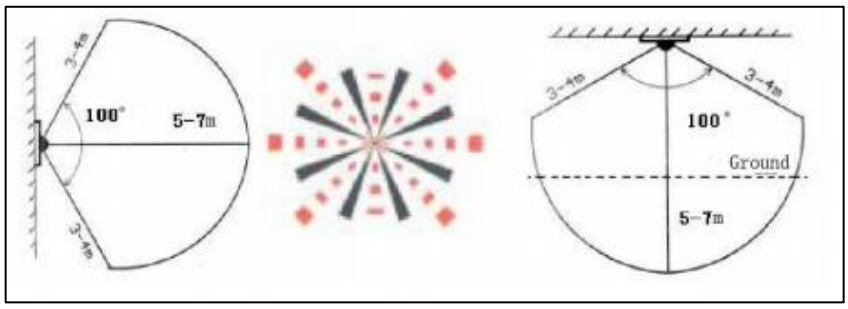


Fuente: PROMETEC. https://www.prometec.net/sensor-pir/.Consulta: 8 de octubre del 2018.

3.1.3.4. Rango de inducción

La figura 21 muestra el ángulo de cobertura del sensor.

Figura 21. **Ángulo del sensor**



Fuente: ELECTRONIC LAB. https://electronilab.co/wp-content/uploads/2013/12/HC-SR501.pdf.Consulta: 8 de octubre del 2018.

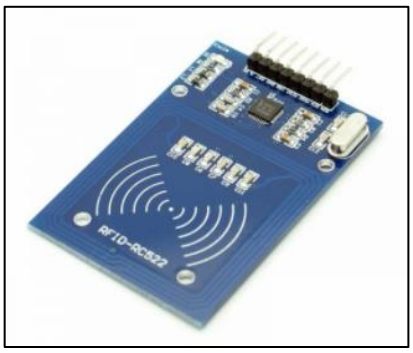
3.1.4. Sensor RFID-RC522 RF

El módulo lector RFID-RC522 RF utiliza 3.3V como voltaje de alimentación y se controla a través del protocolo SPI, así como el protocolo UART, por lo que es compatible con casi cualquier microcontrolador, Arduino o tarjeta de desarrollo.

El RC522 también utiliza un sistema avanzado de modulación y demodulación para todo tipo de dispositivos pasivos de 13.56Mhz. Puesto que se hará una lectura y escritura de la tarjeta, es necesario conocer las características de los bloques de memoria una tarjeta: de la misma tiene un módulo RFID cuenta con 64 bloques de memoria (0-63) donde se hace lectura o escritura. Cada bloque tiene la capacidad de almacenar hasta 16 bytes. El número de serie consiste en 5 valores hexadecimales; se podría utilizar esto para hacer una operación dependiendo del número de serie.

Estos dispositivos permitirán la gestión automatizada del estacionamiento, la identificación personal y el control de acceso de los huéspedes en las habitaciones del hotel, el uso de habitaciones sociales y ambientes privilegiados.

Figura 22. **Sensor RFID RC522**



Fuente: PROMETEC. https://www.prometec.net/arduino-rfid/.Consulta: 8 de octubre del 2017.

3.1.4.1. Características módulo RFID RC5222 RF

La siguiente tabla muestra las especificaciones técnicas del módulo.

Tabla VI. **Datos técnicos**

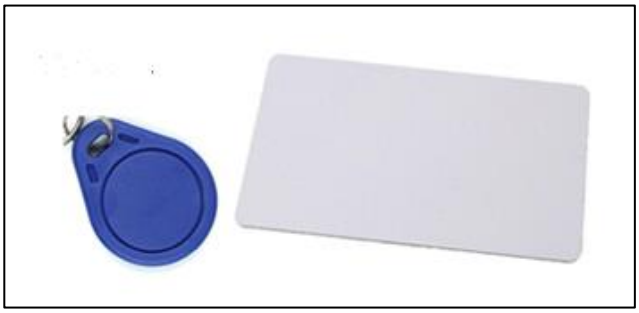
Modelo	MF522-ED
Corriente de operación	13-26mA a 3.3 V
Isb de *stand by*	10-13Ma A 3.3 V
Ism de *sleep-mode*	<80Ua
Im máxima	30Ma
Frecuencia de operación	13.56Mhz
Distancia de lectura	0 a 60mm
Protocolo de comunicación	SPI
Velocidad de datos máxima	10Mbit/s
Dimensiones	40 x 60 mm
Temperatura de operación	-20 a 80°C
Humedad de operación	5% - 95%
Máxima velocidad de SPI	10Mbit/s

Fuente: HETPRO. https://hetpro-store.com/TUTORIALES/modulo-lector-rfid-rc522-rf-con-arduino/.Consulta: 8 de octubre del 2017.

3.1.4.2. Tarjetas RFID para identificación personal

La figura 23 muestra las tarjetas utilizadas para identificar al usuario.

Figura 23. **Tags y tarjetas RFID**



Fuente: PROMETEC. https://www.prometec.net/arduino-rfid/.Consulta: 8 de octubre del 2018.

3.1.5. Sensor de lluvia FC-37

Este tipo de sensores detectan la presencia de lluvia por la variación de conductividad del sensor al entrar en contacto con el agua. Constructivamente, son sensores sencillos. Se dispone de dos contactos, unidos a unas pistas conductoras entrelazadas entre sí a una pequeña distancia, sin existir contacto entre ambas. Al depositarse agua sobre la superficie, se pone en contacto eléctrico ambos conductores, lo que puede ser detectado por un sensor.

Se envía con una placa de medición estándar con el comparador LM393, que permite obtener la lectura tanto como un valor analógico como de forma digital cuando se supera un cierto umbral, que se regula a través de un potenciómetro ubicado en la propia placa. Los valores analógicos medidos

varían desde 0 para una placa totalmente empapada, a 1 023 para una placa totalmente seca.

Estos sensores serán utilizados para activar un mecanismo y extender un toldo y registrar la cantidad de tiempo (días u horas), en las que se producen las precipitaciones. También se utilizará para la detección de inundaciones en áreas específicas y para activar una alerta.

Figura 24. **Sensor FC-37**

Fuente: CDMELECTRONICA. https://www.cdmxelectronica.com/producto/sensor-de-lluvia-para-arduino/. Consulta 7 de octubre de 2019.

Tabla VII. **Especificaciones técnicas**

Característica	Dato
Voltaje de operación	3.3-5 Voltios
Tamaño de PCB	3.2 x 1.4 cm
Tamaño de celda	5 x 4 cm
Chip comparador	LM393

Fuente: CDMELECTRONICA. https://www.cdmxelectronica.com/producto/sensor-de-lluvia-para-arduino/. Consulta 7 de octubre de 2019.

3.1.6. Módulo sensor de luz BH1750

Es un sensor de luz que entrega valores de medición en Lux (lumen /m²), que es una unidad de medida estándar para el nivel de iluminación (iluminancia). Tiene alta precisión y un rango ente 1 – 65535 lx, el cual es configurable.

El módulo tiene un regulador interno de 3.3V y puede alimentar con 5V sin problemas. La interfaz de comunicación es I2C, se puede implementar en la mayoría de microcontroladores. El módulo aparte de los pines de alimentación y pines I2C tiene un pin para establecer la dirección.

Estos sensores se utilizarán para obtener un ahorro energético en ambientes sociales que se iluminan cuando el nivel de lúmenes sea bajo; este sensor se aplicará también en habitaciones cuando no se encuentra ningún usuario utilizando la habitación.

Figura 25. **Sensor BH1750**

Fuente: NAYLAMPMECHATRONICS. https://naylampmechatronics.com/blog/44_Tutorial-m%C3%B3dulo-sensor-de-luz-BH1750.html. Consulta 7 de octubre de 2019.

3.1.6.1. Configuración de la resolución del sensor

La siguiente tabla muestra los modos de resolución en los que trabaja el sensor.

Tabla VIII. **Resolución del sensor**

Modo	Resolución	Tiempo de medición
*High resolution mode*2	0.5 lx	120ms
High resolution mode	1 lx	120ms
Low resolution mode	4 lux	16ms

Fuente: NAYLAMPMECHATRONICS. https://naylampmechatronics.com/blog/44_Tutorial-m%C3%B3dulo-sensor-de-luz-BH1750.html. Consulta 7 de octubre de 2019.

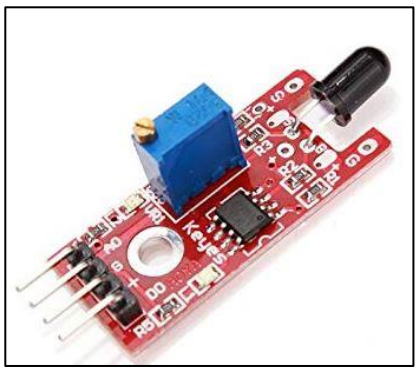
3.1.7. Módulo sensor de incendios KY-026

El sensor KY 026 es capaz de detectar llamas y posee una estructura led detectora de fuego. Cuando el circuito capta las ondas emitidas por la llama este enciende una advertencia, de tal modo que enciende una red. Este singular sensor está conectado a 3 interfaces digitales del led.

Las longitudes de ondas de llama pueden ser detectadas entre 760 nm y los 1 100 nm, cuando utiliza métodos infrarrojos es más sensible 60 grados. Este sensor posee salida AO: salida en tiempo real, la cual está estructurada para enviar señales de voltaje análogas. Su función consiste en enviar señales de advertencia mediante un led indicador; si el sensor no detecta ningún tipo de señal, sencillamente el led no encenderá.

Este sensor será colocado en todas las habitaciones como detector de incendios para enviar alertas que harán sonar alarmas y para activar rociadores para apagar incendios.

Figura 26. **Sensor KY-026**



Fuente: ELECTRONICASTORE. https://electronicastore.net/producto/sensor-ky-026-detector-de-fuego-compatible-con-arduino/. Consulta 7 de octubre de 2019.

3.1.7.1. Características generales

- Voltaje funcional: 0 – 15V DC
- Tamaño: 36 mm x 16 mm
- Ángulo de detección: 60 °C
- Ajustes por potenciómetro
- Salida analógica de cantidad
- Led indicador de alimentación

3.2. Dispositivos

A continuación, se describen los dispositivos utilizados para el diseño del sistema.

3.2.1. Cámara IP Xpy1210

Es una cámara IP inalámbrica de alta definición, permite programación mediante código QR.

Figura 27. **Cámara IP Xpy1210**

Fuente: Intelaf. http://www.intelaf.com/precios_stock_detallado.aspx?codigo=CAM-NXT-BXP1210#esp_tec. Consulta: 12 de marzo del 2019.

3.2.1.1. Características de la cámara

- Conexión inalámbrica IEEE 802.11n
- Compresión de video H.264
- Clasificación IP66 resistente a la intemperie para uso en exterior
- Luces led infrarrojas para visión nocturna
- Comunicación bidireccional
- Admite conexión entre pares P2P
- Configuración de áreas de privacidad
- Notificación por detección de movimiento vía correo electrónico
- DDNS integrado
- Incluye software para administración de video
- Incluye puerto para señal de entrada y salida de alarma
- Programación fácil mediante código QR
- Aplicación para Android y iOS

3.2.1.2. Especificaciones técnicas

La siguiente tabla muestra los datos técnicos de la cámara.

Tabla IX. **Especificaciones de cámara XPY1210**

Cámara	
Sensor de imagen	CMOS de alta definición, de ¼ plg.
Lente	Fijo de 2,8mm
Luces infrarrojas	30
Iluminación mínima	0 lux con luz infrarroja
Alcance de visión nocturna	20m
Filtro de corte infrarrojo	Filtro de corte infrarrojo con encendido y apagado automático
Terminaciones	RJ45
Dimensiones	165*67*67 mm
Peso	0,4 kg

Fuente: Intelaf. http://www.intelaf.com/precios_stock_detallado.aspx?codigo=CAM-NXT-BXP1210#esp_tec. Consulta: 12 de marzo del 2019.

Tabla X. **Especificaciones de imagen**

Imagen	
Resolución	720P/1MP
Compresión	H.264
Número efectivo de pixeles	1 280 (H)*720 (V)
Ángulo de visión	75°
Velocidad de grabación	Hasta 25fps
Imagen especular	Incluida
Funciones	QuickSync, detección de movimiento, detección de sonido, Onvif.

Fuente: Intelaf. http://www.intelaf.com/precios_stock_detallado.aspx?codigo=CAM-NXT-BXP1210#esp_tec. Consulta: 12 de marzo del 2019.

Tabla XI. **Especificaciones de red**

Red	
Interfaz	1 puerto RJ45 de 10/100 Mbps
Protocolos compatibles	Onvif,P2P,DDNS,IP,TCP,UDP,HTTP,HTTPS,SMTP,FTP,DHCP,UPnP,RTSP
DNS dinámico	Incluido
Cortafuegos para seguridad	Admite filtro de IP
Redireccionamiento de puertos	P2P,UPNP
Tipo de flujo	Doble flujo
DHCP	Automático

Fuente: Intelaf. http://www.intelaf.com/precios_stock_detallado.aspx?codigo=CAM-NXT-BXP1210#esp_tec. Consulta: 12 de marzo del 2019.

Tabla XII. **Especificaciones inalámbricas**

Características inalámbricas	
Normas	IEEE 802.11n: 150Mbps, IEEE 802.11g: 54Mbps, IEEE 802.11b: 11Mbps
Frecuencia	2,4 GHz
Antena(s)	Una, omnidireccional
Ganancia de antena	2dBi
Seguridad criptográfica	WEP, WPA, WPA2

Fuente: Intelaf. http://www.intelaf.com/precios_stock_detallado.aspx?codigo=CAM-NXT-BXP1210#esp_tec. Consulta: 12 de marzo del 2019.

Tabla XIII. **Información adicional**

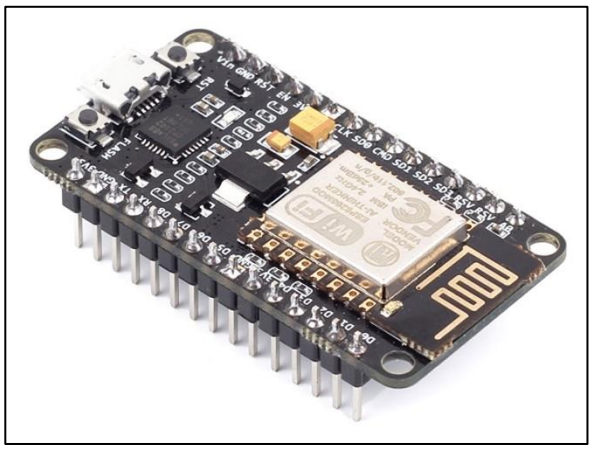
Información adicional	
Certificaciones	FCC (Comisión Federal de Comunicaciones)
Navegadores compatibles	Microsoft IE8, Mozilla Firefox, Google Chrome, Apple Safari
Requisitos mínimos del sistema	Windows 7, Mac OS, iOS, Android
Alimentación	12V/1A

Fuente: Intelaf. http://www.intelaf.com/precios_stock_detallado.aspx?codigo=CAM-NXT-BXP1210#esp_tec. Consulta: 12 de marzo del 2019.

3.2.2. Módulo ESP8266

Es un chip wifi de bajo coste con pila TCP/IP completa y capacidad de MCU (Micro Controller Unit) producida por el fabricante chino Espressif Systems. El ESP8266 tiene potentes capacidades a bordo de procesamiento y almacenamiento que le permiten integrarse con sensores y dispositivos específicos de aplicación a través de sus puertos de propósito general (GPIOs) con un desarrollo mínimo y carga mínima durante el tiempo de ejecución. Su alto grado de integración en el chip permite una circuitería externa mínima. La totalidad de la solución, incluyendo el módulo, está diseñado para ocupar el área mínima en un PCB.

Figura 28. **Módulo ESP8266**



Fuente: ELECTRONILAB. https://electronilab.co/tienda/nodemcu-board-de-desarrollo-con-esp8266-wifi-y-lua/. Consulta: 13 de marzo del 2019.

3.2.2.1. Características

- Código abierto
- Interactivo
- Programable
- Bajo costo
- Sencillo
- Inteligente
- Wifi
- Compatible con Arduino
- USB-TTL included, plug&play
- 10 GPIO, every GPIO can be PWM, I2C, 1-wire
- FCC CERTIFIED WIFI module
- PCB antena

3.2.2.2. Especificaciones técnicas

A continuación, se muestran los datos técnicos del módulo.

Tabla XIV. **Especificaciones**

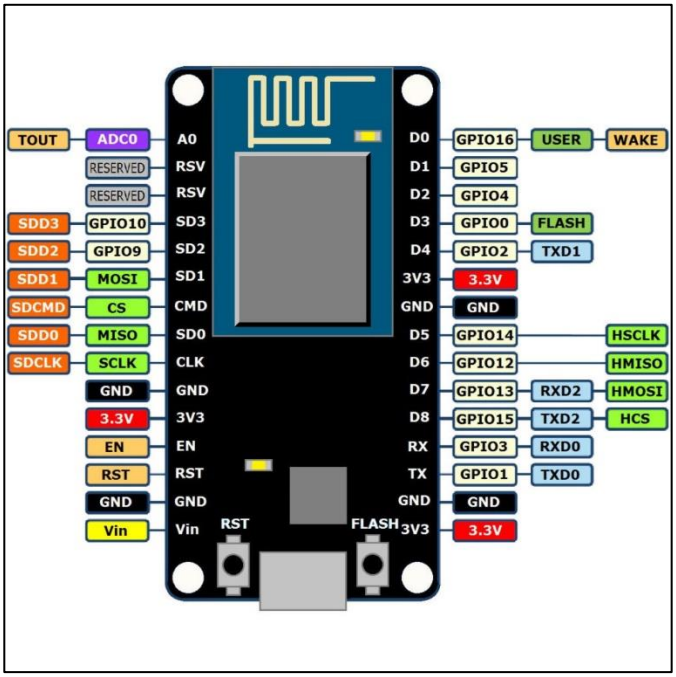
Característica	Dato
Procesador	Tensilica LX106 32 bit a 80 MHz (hasta 160 MHz).
Memoria RAM	80 kB (40 kB disponibles)
Memoria Flash	Hasta 4 MB
ROM	No
Alimentación	3.0 a 3.6 V
Rango de temperaturas	-40°C a 125°C
Consumo de corriente	80 mA (promedio). 225 mA máximo
Consumo en modo sueño profundo	20ua (RTC + memoria RTC)
Coprocesador de bajo consumo	No
WiFi	802.11 b/g/n (hasta +20dBm) WEP, WPA
Soft-AP	Sí

Fuente: ELECTRONILAB. https://electronilab.co/tienda/nodemcu-board-de-desarrollo-con-esp8266-wifi-y-lua/. Consulta: 13 de marzo del 2019.

3.2.2.3. Pines de salida

La figura siguiente muestra la distribución de pines para conexión analógica.

Figura 29. **Distribución de pines de salida**

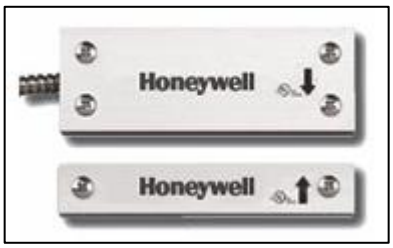


Fuente: ELECTRONILAB. https://electronilab.co/tienda/nodemcu-board-de-desarrollo-con-esp8266-wifi-y-lua/. Consulta: 13 de marzo del 2019.

3.2.3. Chapa magnética

Es un producto diseñado para sistemas de intrusión y atraco, grado 3.

Figura 30. **Contacto magnético 968XTP**



Fuente: Honeywell. https://www.security.honeywell.com/es/productos/intrusion/se/cm/ca/82031.html. Consulta: 7 de octubre del 2018.

3.2.3.1. Descripción

- Certificado EN50131-2-6:2008 Grado 3 clase ambiental II
- Contacto magnético montaje en superficie
- Polarizado para aplicaciones de máxima seguridad
- Resistencia contra sabotaje
- Incluye protección anti-sabotaje y tamper. NC, con cable armado.
- Apertura 11 mm

3.2.3.2. Características del componente

- Interruptor magnético en carcasa enlechada de aluminio con cable isocromático de cinco hilos.
- Dos imanes redondos de neodimio en una carcasa enlechada de aluminio.
- Placa de montaje con un imán recto de alnico integrado.
- Tornillo de montaje.

3.2.4. Thinker A7

A continuación, en la tabla XV se muestran los datos técnicos del módulo.

Figura 31. **Módulo Thinker A7**

Fuente: Labotec.https://www.labotec.pe/en_US/shop/product/frente-a7-gprs-modulo-con-antena-gsm-sms-voz-294?category=74. Consulta: 7 de octubre del 2019.

Tabla XV. **Especificaciones técnicas**

Característica	Dato
Frecuencia de trabajo	cuatribanda red, 850/900/1800 / 1900MHz
Voltaje de funcionamiento	5VDC- corriente de trabajo: máximo de 2A
Corriente en espera	5 mA
Comunicación puerto serial TTL	Interfaz
Velocidad de transmisión	115200bps, ajustable mediante comandos AT
GPS velocidad de transmisión del puerto serial	9600 bps, GPS_TXD
Voltaje lógico de interfaz	3.3 V

Fuente: MACTRONICA. https://www.mactronica.com.co/modulo-gsm-gprs-gps-a7-118418695xJM. Consulta: 9 de octubre del 2019.

3.2.4.1. Características

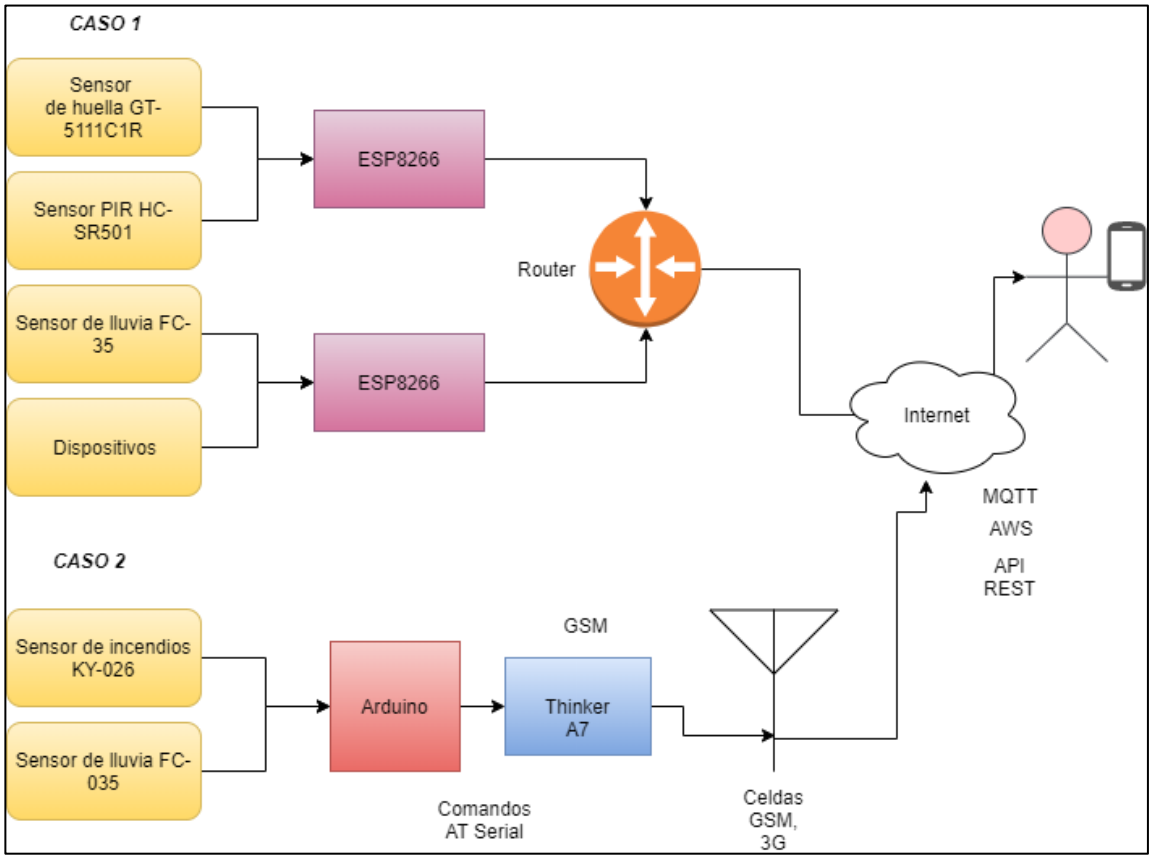
- Soporte de la tarjeta SIM micro a bordo, puede instalar micro de la tarjeta SIM.
- Fuente de alimentación externa USB micro.
- Hace y contesta llamadas telefónicas usando auricular y micrófono electret.
- Envía y recibe mensajes SMS.
- Envía y recibe datos GPRS (TCP / IP, http, entre otros).
- Es utilizado para probar el módulo GPRS / GPS A7 Thinker.
- Interfaz de antena SMA.

4. CONEXIONES Y CIRCUITOS

4.1. Modelo propuesto

El siguiente diagrama muestra el modelo para el diseño del sistema:

Figura 32. **Diagrama del modelo propuesto**



Fuente: elaboración propia.

El modelo propuesto de la red de dispositivos IoT se compone de una serie de conexiones y configuraciones para cubrir varios criterios al momento de que el usuario quiere controlar dispositivos o recibir actualizaciones de los sensores que están conectados en la red. Se propone este modelo tomando en cuenta las necesidades del usuario y sus limitaciones. A continuación, se describe los criterios tomados en cuenta para la elaboración del modelo.

4.1.1. Conexión del usuario desde una interfaz

Para la buena experiencia del usuario es indispensable contar con una interfaz donde pueda interactuar con los dispositivos o sensores que tiene configurados en su red de IoT. En este modelo se propone utilizar el sistema operativo Android para visualizar las interfaces en dispositivos móviles. Android es un sistema operativo desarrollado por Google basado en el kernel de Linux; está disponible para dispositivos celulares, *tablets,* relojes inteligentes, televisores inteligentes, automóviles, entre otros. Debido a la gran popularidad de Android, este sistema operativo cubre la mayor parte de usuarios que interactúan con estos dispositivos. Para la elaboración de la interfaz se recomienda utilizar el lenguaje de programación Java con el programa Android Studio versión 3.5 donde las partes visuales son diseñadas mediante el meta-lenguaje xml.

4.1.1.1. Librerías recomendadas para el envío de información en aplicaciones Android IoT

- Retrofit: es un cliente HTTP seguro que permite definir una API REST como una interfaz. Permite manipular el cuerpo, los encabezados, los parámetros de consulta y mucho más de las solicitudes del API, por

medio de anotaciones para que todo sea más legible y limpio. También permite la ejecución de llamadas al API síncronas y asíncronas.

- OkHttp: esta es una alternativa más simple para realizar un cliente HTTP pero no tiene tantas ventajas en configuración como las que tiene Retrofit.

- Volley: es otra alternativa para la utilización del protocolo HTTP que hace que la creación de redes en aplicaciones de Android sea más rápida. Su implementación es más difícil que las anteriores, pero se recomienda cuando se necesita enviar una gran cantidad de datos.

Figura 33. **Código de ejemplo para enviar información a un API Rest**

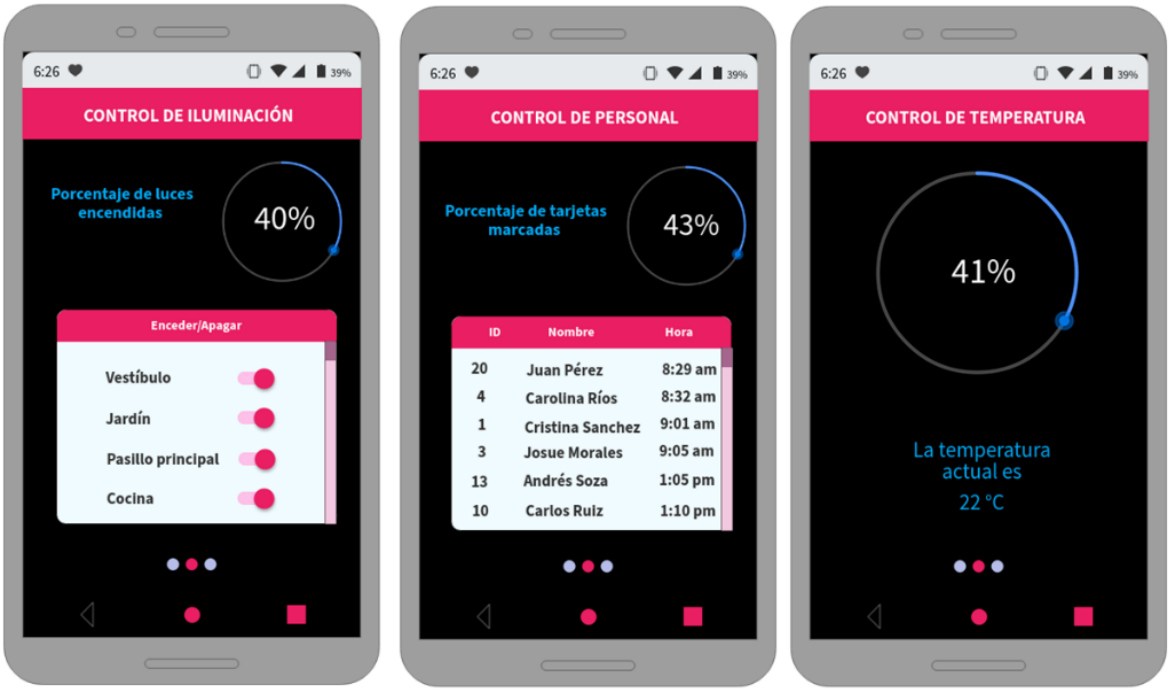
```
HttpUrl.Builder urlBuilder = HttpUrl.parse("https://ejemplo.com").newBuilder();
urlBuilder.addQueryParameter("sensor", "ultrasonico");
urlBuilder.addQueryParameter("valor", "3.45");
String url = urlBuilder.build().toString();

Request request = new Request.Builder()
                    .url(url)
                    .build();
```

Fuente: elaboración propia.

4.1.1.2. Elaboración de interfaces de usuario para el uso de IoT

Se muestra a continuación diseños para interfaces de usuario para mostrar y controlar los sensores o dispositivos finales en una red IoT.

Figura 34. **Interfaces de control de usuario**



Fuente: elaboración propia

4.1.2. Conexión de sensores o dispositivos terminales mediante una topología de red

Un posible escenario para la comunicación de los dispositivos de IoT es la conexión por medio de una infraestructura de red, en donde se requiere que cada dispositivo tenga conexión a internet para poderse comunicar al servicio REST publicado. En la solución propuesta se utiliza un router *Thomson* dwg849 para generar la red y crear la conexión a internet. El medio de conexión que se ha utilizado es wifi para que el control de los dispositivos IoT sea más cómodo para el usuario. Debido a estos requerimientos se utiliza el módulo ESP8266;

este es un dispositivo que utiliza un puerto TCP y funciona como servidor o como cliente, dependiendo del estado de funcionamiento del dispositivo.

4.1.2.1. Problemas generados en las conexiones de dispositivos MQTT

Al momento de tener múltiples dispositivos de IoT conectados en una red se generan diferentes problemas, tomando en cuenta que la red de IoT puede crecer en función de la cantidad de datos que se debe procesar y de cuántos dispositivos se quiera controlar.

4.1.2.1.1. Gran consumo de ancho de banda

Debido a que los dispositivos de IoT recolectan datos o esperan recibir datos, generan muchas conexiones que consumen un ancho de banda. Cuando se conectan muchos dispositivos, la red crece y estos consumen un gran ancho de banda que afectará a la conexión. Debido a esto, se necesita ocupar la menor cantidad de ancho de banda.

Si se utilizara el protocolo de comunicación WebSocket y se analizara su estructura, mostrada en la figura, se puede observar que se compone por las siguientes partes:

- El byte de control: contiene información relativa a los atributos de websocket.
    - Fin Bit: si no está configurado, le indica al servidor que continúe escuchando los segmentos del WebSocket.

- Extensión Bit: son los siguientes 3 bits y se utiliza para especificar el uso de extensiones; estas deben ser soportadas por todos los que consuman el servicio.
- OPCODE: son cuatro bits que incluyen la información utilizada en la comunicación.

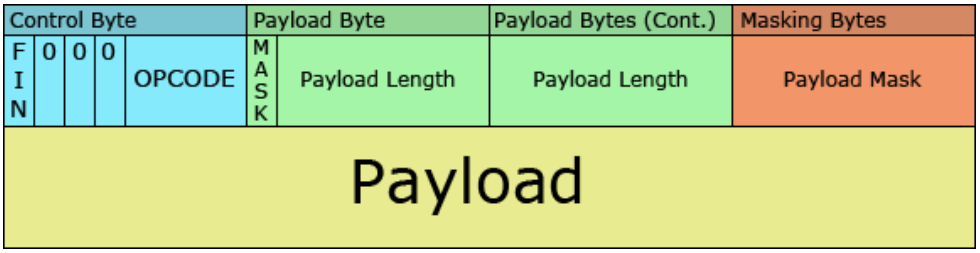
    - Marco de continuación (0x0): es una continuación del mensaje anterior y su contenido debe concatenarse al búfer general de mensaje.
    - Texto (0x1): establece si el contenido es texto con codificación UTF-8.
    - Binario (0x2): establece si el contenido es binario sin formato.
    - Cerrar (0x8): es una notificación de que el Websocket se está cerrando.
    - Ping (0x9): es un ping a un punto final.
    - Pong (0xA): es un pong devuelto al servidor.

- Los bytes de payload: establecen la longitud del payload para poder decodificar el mensaje.
- Los bytes de máscara: contienen la información de la máscara para poder decodificar la información contenida en el mensaje de la comunicación de WebSocket. Este utiliza los bytes de payload para hacer la decodificación a un XOR del bit codificado en la posición i-ésima del mensaje con la máscara en la posición del módulo cuatro del índice i-ésimo, en otras palabras

    - `DECODED[i] = ENCODED[i] ^ MASK[i % 4]`

- Payload: contiene el mensaje con una longitud establecida.

Figura 35. **Estructura en la transmisión de un websocket**



Fuente: GITHUB. https://github.com/OpenSuede/Suede/wiki/Websocket-Structure. Consulta: 7 de octubre del 2019.

Como se observa, la cantidad de bytes necesarios para establecer una conexión de WebSocket es muy extensa que cada mensaje que se envíe o se reciba consuma una mayor cantidad de ancho de banda. Este tipo de comunicación es full-duplex o bidireccional; es decir, que tanto el cliente como el servidor pueden enviar y recibir mensajes.

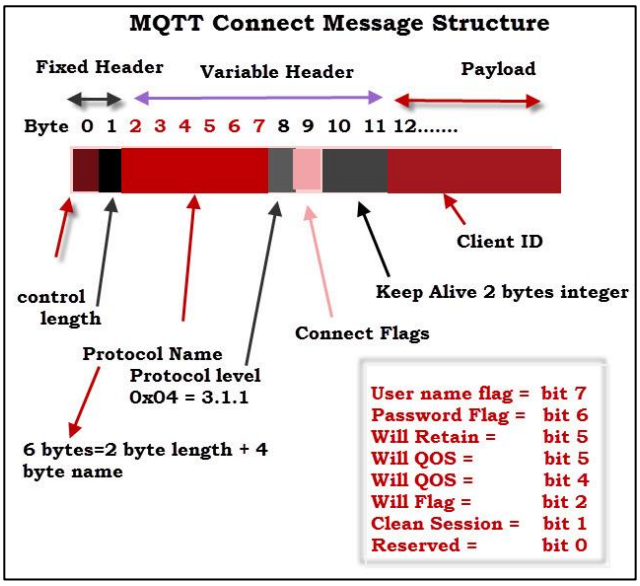
Una opción más eficiente es la utilización del protocolo MQTT, ya que, si se analiza su estructura interna mostrada en la figura, se compone de las siguientes partes:

- Control Header: siempre está presente en cada paquete de MQTT. Contiene bits de control para la comunicación del protocolo MQTT.

    - Message Type: es el tipo de mensaje y representa un tipo de solicitud de conexión. Cada valor de tipo de mensaje representa una solicitud de conexión diferente. En la tabla se describe los tipos de mensajes que pueden existir.

- Packet Length: es la longitud del paquete.
- Variable length header: no siempre está presente, y se utiliza para enviar información extra en un paquete de MQTT. Es similar para varios paquetes de mensajes MQTT. Por ejemplo, un paquete de conexión se muestra en la figura siguiente.

Figura 36. **Estructura de una conexión de mensaje MQTT**

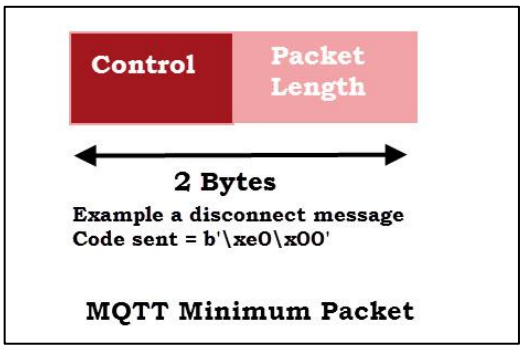


Fuente: BYTESOFGIGABYTES http://www.bytesofgigabytes.com/MQTT/MQTT-protocol-packet-structure/. Consulta: 9 de octubre del 2019.

- Payload: puede o no estar presente y contiene los datos que se desea enviar.

En el protocolo MQTT el mensaje más corto es de tan solo dos bytes, por ejemplo, el de desconexión, como se muestra en la figura

Figura 37. **Estructura de un mensaje de desconexión MQTT**



Fuente: BYTESOFGIGABYTES http://www.bytesofgigabytes.com/MQTT/MQTT-protocol-packet-structure/. Consulta: 9 de octubre del 2019.

Como se puede confirmar en el protocolo MQTT se tiene una menor cantidad de bytes de transmisión; por lo tanto, esto hace que el ancho de banda se optimice.

Tabla XVI. **Tipos de mensajes existentes en MQTT**

Tipo de mensaje	Valor	Descripción
Reserved	0	Reservado
CONNECT	1	El cliente se conecta al servidor o bróker
CONNACK	2	Petición de reconocimiento
PUBLISH	3	Publicar mensaje
PUBACK	4	Publicar reconocimiento
PUBREC	5	Publicar recibimiento
PUBREL	6	Publicar lanzamiento
PUBCOMP	7	Publicar completado
SUBSCRIBE	8	Cliente envía petición para suscribirse
SUBACK	9	Suscribir petición de reconocimiento
UN SUBACK	10	Petición de dar de baja
PINGREQ	11	Petición de PING
PINGRESP	12	Respuesta de PING
DISCONNECT	13	Cliente se desconecta
Reserved	14	Reservado

Fuente: BYTESOFGIGABYTES. http://www.bytesofgigabytes.com/MQTT/MQTT-protocol-packet-structure/. Consulta: 9 de octubre del 2019.

4.1.2.2. Consumo de energía

Otro problema que se presenta es el consumo de energía que necesitan los dispositivos de IoT. Si bien esto también depende de los elementos utilizados, se debe tratar de que la conexión utilice el menor consumo de energía. Como se describió en la sección anterior, la cantidad de bytes requeridos en MQTT es relativamente pequeña; por lo tanto, el tiempo de conexión será mucho más corto y se puede así aprovechar de mejor manera la energía.

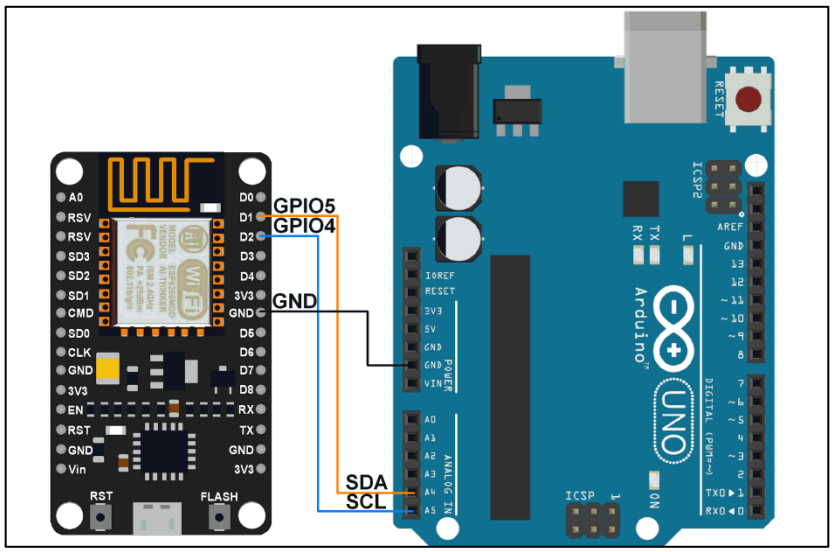
4.1.2.3. Identificación de los dispositivos de IoT

Se debe considerar que se van a conectar muchos dispositivos a la red de IoT, por lo tanto, se debe identificar individualmente cada uno. Se propone colocar un identificador único para cada uno de los dispositivos y revisar en una base de datos su existencia.

4.1.3. Conexión de dispositivos de IoT

Ya que las conexiones de los dispositivos IoT deben ser por medio de la red wifi se utilizará el módulo ESP8266. Este puede programarse con el lenguaje C de Arduino. La conexión que se utiliza entre el Arduino y el módulo ESP8266 se muestra en la figura 38.

Figura 38. **Conexión del módulo ESP8266 con Arduino**



Fuente: ARROW. https://www.arrow.com/es-mx/research-and-events/articles/protocols-for-the-internet-of-things. Consulta: 9 de octubre del 2019.

La conexión de los dispositivos de IoT se muestra en dos casos, los cuales se presentan a continuación:

4.1.3.1. Conexión a la red

Para las conexiones a la red, se proponen dos estados:

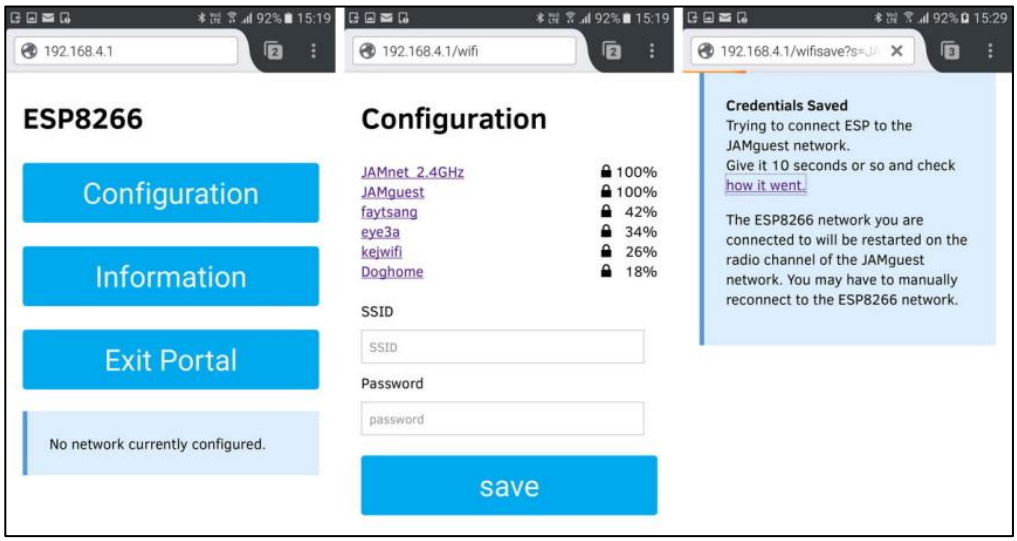
El primer estado: la primera vez que se conecta el dispositivo. Como paso inicial, el dispositivo de IoT debe conectarse a la red, la cual puede tener salida a internet. El dispositivo ESP8266 puede configurarse como servidor o como cliente. La primera vez debe encontrar a qué red wifi debe conectarse; por lo tanto, se crea un portal cautivo para que el usuario pueda configurar el

dispositivo al colocar el SSID y la contraseña de su red wifi. En la programación del módulo ESP8266 se utiliza las siguientes librerías:

- WifiManager
- DNSServer
- ESP8266WebServer

Para generar el portal cautivo se utiliza la librería WifiManager, la cual generará una red para que el usuario pueda conectarse al dispositivo de IoT. Por defecto, la dirección que se utiliza es la 192.168.4.1; cuando se entra a esta dirección se muestra una interfaz web donde se puede colocar los datos de la red. Una vez conectado se enviará un mensaje al servidor para que se genere un id, el cual servirá para que el dispositivo se identifique.

Figura 39. **Portal cautivo creado por la librería WifiManager**

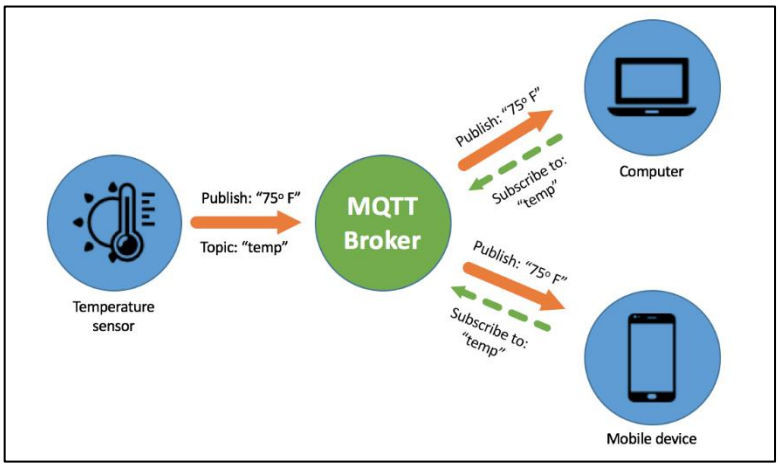


Fuente: elaboración propia.

Segundo estado: el dispositivo ya ha sido conectado. Al momento de que las credenciales sean correctas el módulo será capaz de autoconectarse a la red configurada. Manda un mensaje de connnect al servidor de MQTT y modifica su campo de estado de offline a online. Si el dispositivo es un sensor, una vez conectado se empieza a enviar la información del sensor al tiempo requerido por medio de la comunicación de MQTT. En la página del servicio en la nube se verán las actualizaciones en tiempo real, ya que esta comunicación es full-duplex.

Como se ha descrito, lo más eficiente es realizar una conexión con la comunicación MQTT. Para la configuración de este protocolo en el cliente Arduino se debe tener un servidor o Broker, como se muestra en la siguiente imagen.

Figura 40. **Estructura básica de conexión de un Broker con sus clientes**



Fuente: ARROW. https://www.arrow.com/es-mx/research-and-events/articles/protocols-for-the-internet-of-things. Consulta: 9 de octubre del 2019.

La programación de un cliente MQTT consiste en la configuración de la comunicación serial para la visualización de lo que se está enviado y recibiendo en la consola serial; también se debe realizar la configuración del wifi.

Como se describió, debe realizarse la validación de que el dispositivo ya haya sido conectado anteriormente; esto se hace mediante la función autoConnect de la librería WifiManager. Cuando ya ha sido conectado el dispositivo, se proporciona una dirección Ip que está dentro de la red a la cual se conectó. Para configurar el cliente de MQTT se utiliza la librería PubSubClient con la función setServer, y se envía como parámetro de la función el broker o servidor de MQTT y el puerto donde se escucha el servicio. Por último, se especifica una función de donde se recibirán los mensajes enviados por el broker de MQTT; en esta implementación se llamará callback. Toda esta programación en Arduino se muestra en la figura 40.

Figura 41. **Programación de la configuración de la conexión wifi de un módulo ESP8266**

```
void setup() {
    Serial.begin(115200);
    WiFiManager wifiManager;
    wifiManager.setBreakAfterConfig(true);
    //wifiManager.resetSettings();
    if(!wifiManager.autoConnect("RedIoT_inicial")){
        delay(3000);
        ESP.reset();
        delay(5000);
        Serial.println("fallo la conexión de modo configuración");
    }
    while(WiFi.status() != WL_CONNECTED){
        delay(500);
    }
    Serial.print("Ip address: ");
    Serial.println(WiFi.localIP());
    Serial.printf("MAC address = %s\n",WiFi.softAPmacAddress().c_str());
    Serial.println("Estamos conectados");
    server.begin();
    server.on("/ruta",[](){
        server.send(200,"text/plain","Esta conectado");
    });
    client.setServer(mqtt_server,1883);
    client.setCallback(callback);
}
```

```
void callback(char* topic, byte* pyload, unsigned int length){
    Serial.print("Llamada del server");
    Serial.print(topic);
}
```

Fuente: elaboración propia.

En el método loop (ver figura 41), se verifica si el cliente de wifi está conectado o si busca volver a conectar. En este método se realiza el censado de temperatura y se envía al servidor de MQTT por medio de la función publish; en esta implementación, este proceso se hará cada seis segundos.

Figura 42. **Programación de la configuración de la conexión Wi-Fi de un módulo ESP8266 en el método loop**

```
void loop() {
    if(!client.connected()){
        reconnect();
    }
    long now = millis();
    if(now - lastMsg > 6000){
        lastMsg = now;
        String msg = "Temperatura: ";
        msg = msg + "dato";
        char message[58];
        msg.toCharArray(message,58);
        Serial.println(message);
        client.publish("sensor", message);
    }
}
void reconnect(){
    while(!client.connected()){
        Serial.print("Esperando MQtt conexion..");

        String clientId = "ESP8266Client-";
        clientId += String(random(0xffff),HEX);

        if(client.connect(clientId.c_str())){
            Serial.println("Conectado a mqtt");
            client.subscribe("Cliente");
        }else{
            Serial.print("Fallo estado=");
            Serial.print(client.state());
            Serial.println("Espere 5 segundos");
            delay(5000);
        }
    }
}
```

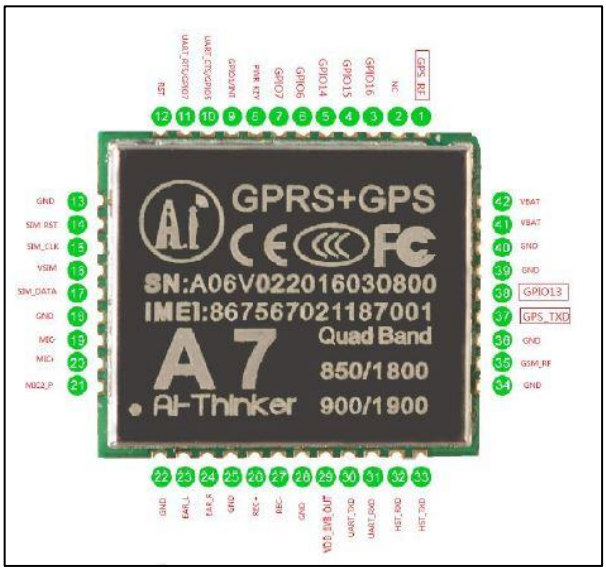
Fuente: elaboración propia

4.1.4. Conexión de sensores o dispositivos terminales mediante una conexión de red móvil celular

Cuando se necesita colocar sensores en lugares que no pueden tener acceso a una conexión de red fija, pero sí se tiene acceso mediante una conexión de red celular, se debe pensar cómo llegar a tener una conexión a internet utilizando esta red. Se propone utilizar el módulo Thinker A7, el cual realiza una conexión por medio la red GSM GPRS. El módulo Thinker A7 se comunica mediante comandos AT, los cuales utilizan la comunicación serial.

Para crear la comunicación serial con el módulo se utiliza un Arduino UNO por su facilidad de configuración.

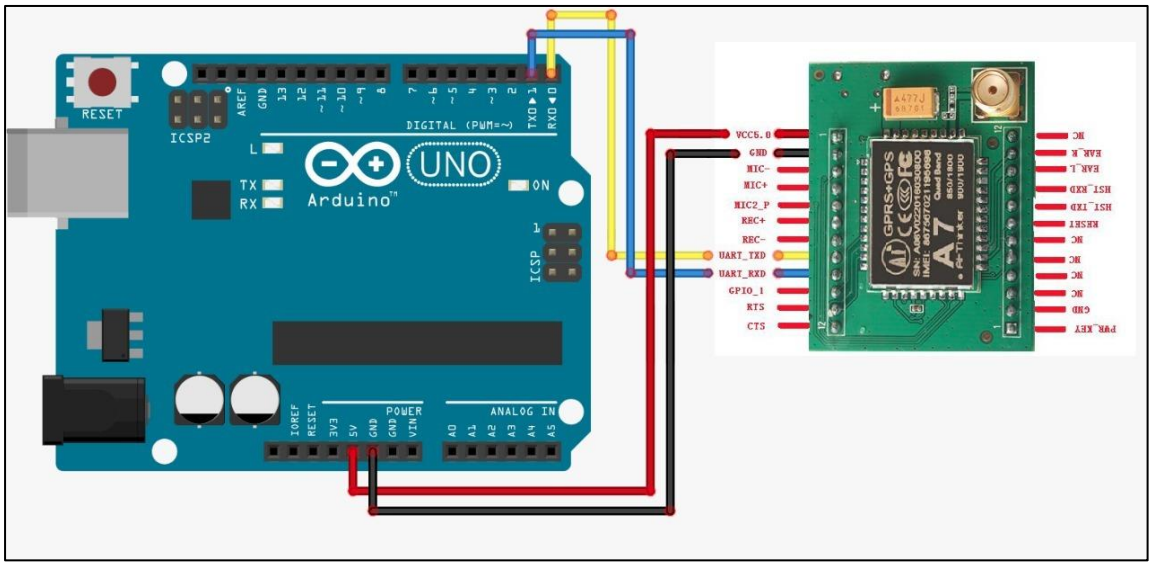
Figura 43. **Módulo Thinker A7**



Fuente: ACOPTEX. http://acoptex.com/project/286/basics-project-064a-ai-thinker-a7-gsm-gprs-gps-module-at-acoptexcom/. Consulta: 9 de octubre del 2019.

Para comunicar los sensores se utilizan los puertos del Arduino y este se comunica con el módulo Thinker A7 mediante una conexión serial. La primera configuración que se debe realizar es esta conexión. Se debe conectar los pines correspondientes del Arduino al módulo Thinker A7, como se muestra en la figura 43.

Figura 44. **Conexión del módulo Thinker A7 con Arduino**



Fuente: ACOPTEX. http://acoptex.com/project/286/basics-project-064a-ai-thinker-a7-gsm-gprs-gps-module-at-acoptexcom/. Consulta: 9 de octubre del 2019.

Se debe tener muy presente la velocidad de transmisión; esta puede ser 9600 bps o 115 200 bps. Si no se especifica la velocidad de transmisión tanto en la configuración de Arduino como en el módulo Thinker A7, generará una mala comunicación y mostrará errores no especificados.

Los comandos AT están explicados en la hoja de especificaciones técnicas del fabricante. Para tener la comunicación, algunos de los comandos son los siguientes:

- AT+CCID: Obtiene el id de la tarjeta SIM.
  - Respuesta: +SCID: SIM Card ID: 898602#4221620070426

- AT+CREG?: proporciona información sobre el estado de registro y la tecnología de acceso de la celda en servicio. Una respuesta posible puede ser “+ CREG: 1,1”, donde los números representan el estado de registro y los posibles valores de la tecnología de acceso. Los valores posibles del estado de registro son:

  - 0: No registrado
  - 1: Registrado, red doméstica
  - 2: No registrada
  - 3: Registros denegados
  - 4: Desconocidos
  - 5: Registrado, roaming
  - 6: Registrado para "solo SMS", red doméstica
  - 7: Registrado para "solo SMS", roaming
  - 8: Adjunto solo para servicios portadores de emergencia
  - 9: Registrado para "CSFB no preferido", red doméstica
  - 10: Registrado para "CSFB no preferido", roaming

  Los valores posibles para la tecnología de acceso son:

  - 0: GSM
  - 1: GSM Compact

- 2: UTRAN
- 3: GSM con EGPRS
- 4: UTRAN con HSDPA
- 5: UTRAN con HSUPA
- 6: UTRAN con HSDPA y HSUPA
- 7: E-UTRAN

- AT+CGATT=1: se usa para conectar o desconectar el dispositivo al paquete de servicio.

- AT+CGACT=1,1: se usa para activar o desactivar la conexión GPRS.

- AT + CGDCONT=1,IP,CMNET: establecen los parámetros de contexto PDP, como el tipo PDP (IP, IPV6, PPP, X.25, etc.), APN, compresión de datos, compresión de encabezado, entre otros.

- AT+CIPSTART=TCP,116.228.221.51,80: inician una conexión TCP o UDP. Se especifica el servidor a donde se quiere conectar y el puerto del servicio.

- AT+CIPSEND=80: se usa para enviar los datos prueba a través de la conexión TCP o UDP y configurar el puerto de envío.

- GET http://116.228.221.51?datos=13.45 HTTP/1.0 HOST:116.228.221.5: este comando se utiliza para enviar la información al servidor especificando el método de envío, como GET o POST y los parámetros enviados.

- AT+CIPCLOSE: se utiliza para cerrar la conexión.

4.1.5. Configuración de los servicios en la nube para la conexión total entre el usuario, dispositivos terminales o sensores

Todos los dispositivos, independientemente de cómo sea su conexión, se deben enlazar a un servidor en la nube. En esta implementación se utiliza como proveedor de servicios Amazon EC2, donde se aloja el servidor de MQTT. Se utiliza como lenguaje de servidor NodeJS por su fácil implementación. Para crear el Broker o servidor de MQTT se utilizará la librería llamada mosca.

En la configuración del servidor se debe primero importar la librería y configurar el puerto de acceso de los clientes. Además, se debe utilizar el evento ready para que se configure de forma correcta el servicio. La declaración y configuración se describe en la figura 44.

Figura 45. **Configuración del Broker MQTT en NodeJS**

```
var mosca = require('mosca')

var moscaSettings = {
port: 1883,
};

var server = new mosca.Server(moscaSettings);
server.on('ready', () => {
console.log('Servidor MQTT corriendo!')
});
```

Fuente: elaboración propia

Para poder enviar un mensaje desde el servidor a los clientes se utiliza el método publish con los parámetros del mensaje. Se puede colocar los eventos clientConnected y clientDisconnected para conectar o desconectar, respectivamente. Por último, esta librería permite la recepción de mensaje de los clientes conectados por medio del evento published. La programación de esta parte del servicio se describe en la figura 45.

Figura 46. **Programación de los eventos en un bróker MQTT**

```
var message = {
topic: '/tema',
payload: 'abcde',
qos: 0,
retain: false
};

server.publish(message, function() {
console.log('done!');
});

server.on('published', function(packet, client) {
    console.log('Published', packet);
    console.log('Client', client);
});

server.on('clientConnected', function(client) {
    console.log('Cliente conectado:', client.id);
});

server.on('clientDisconnected', function(client) {
    console.log('Client desconectado:', client.id);
});
```

Fuente: elaboración propia

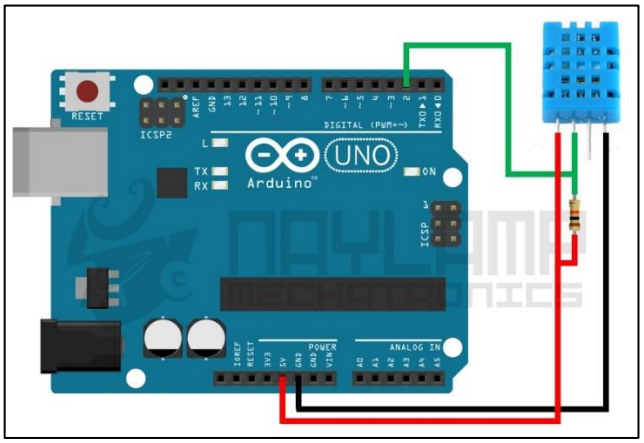
4.1.6. Conexión de los sensores

Se muestra la conexión de los pines en cada uno de los sensores con Arduino.

4.1.6.1. Sensores y Arduino

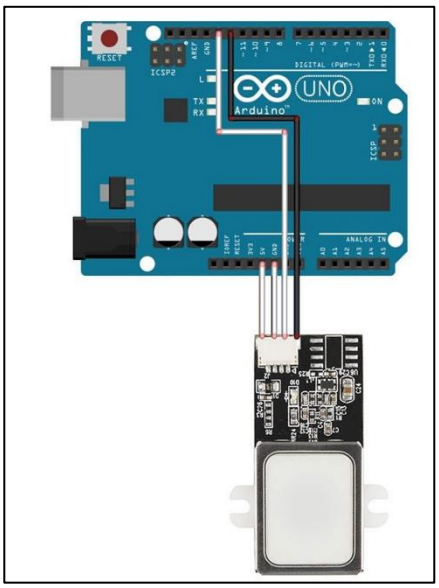
A continuación, se muestra el módulo y sus conexiones con Arduino.

Figura 47. **Módulo DHT22 conectado con Arduino**



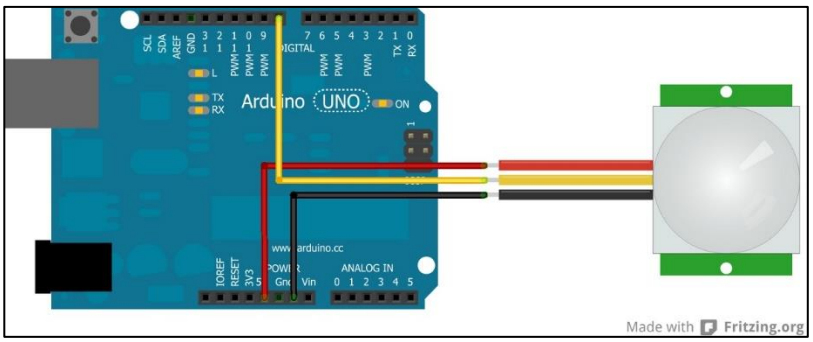
Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

Figura 48. **Módulo GT-511C1R conectado con Arduino**



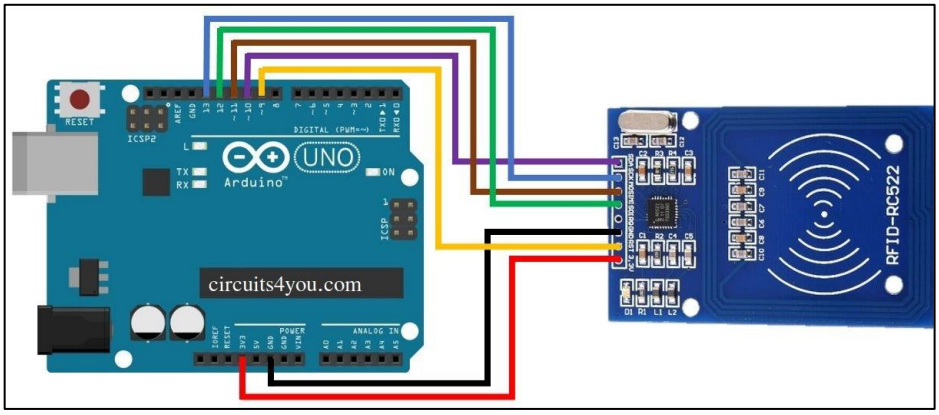
Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

Figura 49. **Módulo HC-SR501 conectado con Arduino**



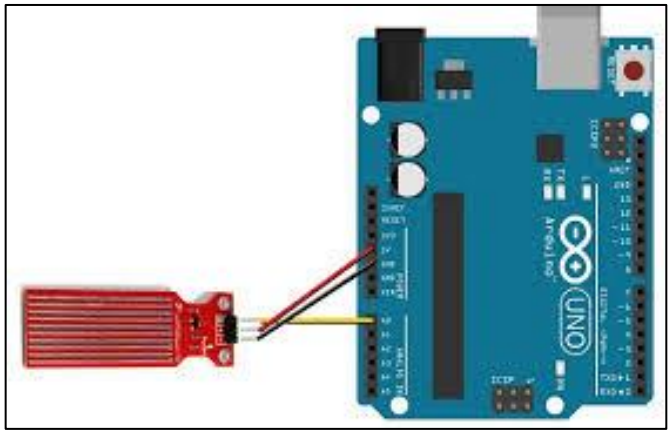
Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

Figura 50. **Módulo RC 522RF conectado con Arduino**



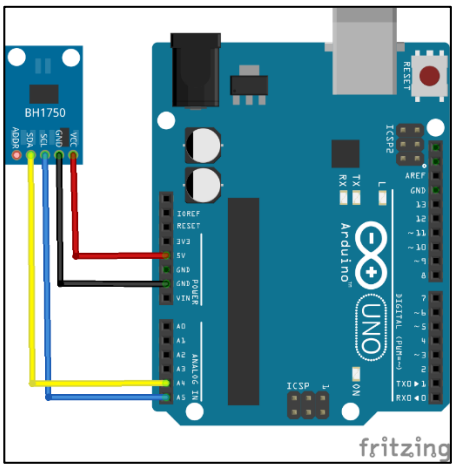
Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

Figura 51. **Módulo FC-37 conectado con Arduino**



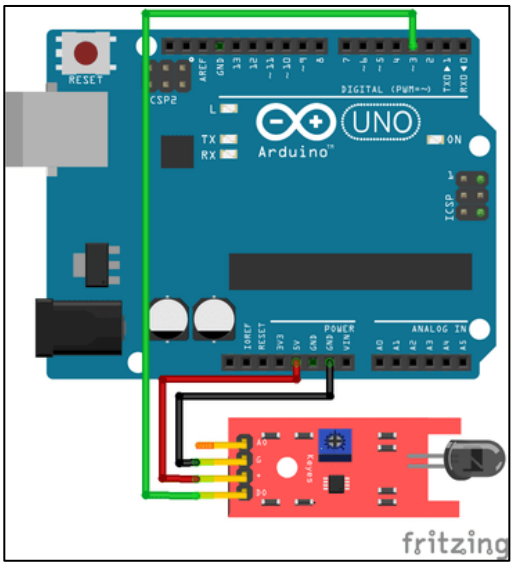
Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

Figura 52. **Sensor BH1750 conectado con Arduino**



Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

Figura 53. **Sensor KY-026 conectado con Arduino**



Fuente: IOTGUIDER. https://iot-guider.com/category/arduino/page/3/.Consulta: 9 de octubre del 2019.

CONCLUSIONES

1. Los sensores establecidos son aptos para cumplir los requerimientos y ejecutar las acciones que proporcionan ahorro energético y seguridad al hotel.

2. Se debe utilizar el protocolo MQTT para la creación de los modelos por su bajo consumo de ancho de banda y energía.

3. Los modelos de red utilizados permiten al usuario tener control administrativo del lugar, manejando este de forma local y también móvil.

4. Para controlar la información de los ambientes se utiliza una interfaz Android en los dispositivos móviles.

RECOMENDACIONES

1. Establecer un modelo de negocios es importante ya que permite ordenar y establecer las necesidades y los límites del diseño al momento de comercializar productos de Internet de las cosas.

2. Tomar en cuenta la cantidad de sensores que se van a utilizar al momento de determinar el tamaño del disco de almacenamiento y la velocidad de procesamiento de los datos, y si existirá un crecimiento en los dispositivos.

3. Utilizar protocolos de comunicación donde la cantidad de bytes en la estructura interna sea menor, como el protocolo MQTT, ya que esto dará eficiencia en la red de IoT.

4. Tomar en cuenta el tipo de cliente y ambientes para elegir que herramienta de aprendizaje de IoT es más conveniente.

BIBLIOGRAFÍA

1. Arrow. *Protocolos para Internet de las cosas.* [en línea]. <https://www.arrow.com/es-mx/research-and-events/articles/protocols-for-the-internet-of-things>. [Consulta: 9 de octubre de 2019].

2. BANAFA, Ahmed. *Secure and Smart Internet of Things*. Dinamarca: River Publishers, 2018. 164 p.

3. Bytes of gigabytes. *MQTT Packet Structurte.* [en línea]. <http://www.bytesofgigabytes.com/MQTT/MQTT-protocol-packet-structure/>. [Consulta: 8 de octubre de 2019].

4. CASSIMALLY, Hakim. *Designing Internet of Things*. Inglaterra: John Wiley and Sons, Ltd., 2014. 338 p.

5. EDUREKA. *Machine Learning.* [en línea]. <https://www.edureka.co/blog/what-is-machine-learning/>. [Consulta: 10 de junio de 2019].

6. GITHUB. *WebSocket Structurte.* [en línea]. <https://github.com/OpenSuede/Suede/wiki/Websocket-Structure>. [Consulta: 8 de octubre de 2019].

7. OSTERWALDER, Alexander. *Business Model Generation*. 2a ed. Hoboken, New Jersey: John Wyley & Sons, Inc., 2010. 288 p.

8. Steves internet guide. *MQTT Protocol.* [en línea]. <http://www.steves-internet-guide.com/MQTT-protocol-messages-overview/>. [Consulta: 9 de octubre de 2019].

9. VELTRI, F.G., CIRANI, P.M. *Internet of Things: Achitectures, Procols and Standads*. 1a ed. Estd: John Wiley and Sons, Ltd., 2019. 338 p.

APÉNDICE

Apéndice 1. **Código utilizado para los sensores**

- Sensor DHT22

sketch_oct10a §

```
#include "DHT.h"

#define DHTPIN 2     // Pin donde está conectado el sensor
#define DHTTYPE DHT22   // Sensor DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("Iniciando...");
  dht.begin();
  pinMode(13, OUTPUT);
}
long tiempoUltimaLectura=0;; //Para guardar el tiempo de la última lectura
void loop() {
  //---------Lectura del Sensor--------------------------
  if(millis()-tiempoUltimaLectura>2000)
  {
      float h = dht.readHumidity(); //Leemos la Humedad
      float t = dht.readTemperature(); //Leemos la temperatura en grados Celsius
      float f = dht.readTemperature(true); //Leemos la temperatura en grados Fahrenheit
      //--------Enviamos las lecturas por el puerto serial-------------
      Serial.print("Humedad ");
      Serial.print(h);
      Serial.print(" %t");
      Serial.print("Temperatura: ");
      Serial.print(t);
      Serial.print(" *C ");
      Serial.print(f);
      Serial.println(" *F");
      tiempoUltimaLectura=millis(); //actualizamos el tiempo de la última lectura
  }
  //----Fin de la lectura---------------------------

  //---------Resto del código  del proyecto--------
  //...
  //...
  //...
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
  //-------------------------------

}
```

Continuación del apéndice 1.

- Módulo GT-511C1R

FPS_Enroll

```
*****************************************************************/

#include "FPS_GT511C3.h"
#include "SoftwareSerial.h"

// set up software serial pins for Arduino's w/ Atmega328P's
// FPS (TX) is connected to pin 4 (Arduino's Software RX)
// FPS (RX) is connected through a converter to pin 5 (Arduino's Software TX)
FPS_GT511C3 fps(4, 5); // (Arduino SS_RX = pin 4, Arduino SS_TX = pin 5)

/*If using another Arduino microcontroller, try commenting out line 60 and
uncommenting line 69 due to the limitations listed in the
library's note => https://www.arduino.cc/en/Reference/softwareSerial . Do
not forget to rewire the connection to the Arduino*/

// FPS (TX) is connected to pin 10 (Arduino's Software RX)
// FPS (RX) is connected through a converter to pin 11 (Arduino's Software TX)
//FPS_GT511C3 fps(10, 11); // (Arduino SS_RX = pin 10, Arduino SS_TX = pin 11)

void setup()
{
  Serial.begin(9600); //set up Arduino's hardware serial UART
  delay(100);
  fps.Open();         //send serial command to initialize fps
  fps.SetLED(true);   //turn on LED so fps can see fingerprint

  Enroll();          //begin enrolling fingerprint
}
void Enroll()
{
  // Enroll test

  // find open enroll id
  int enrollid = 0;
  bool usedid = true;
  while (usedid == true)
  {
    usedid = fps.CheckEnrolled(enrollid);
    if (usedid==true) enrollid++;
  }
  fps.EnrollStart(enrollid);

  // enroll
  Serial.print("Press finger to Enroll #");
  Serial.println(enrollid);
  while(fps.IsPressFinger() == false) delay(100);
  bool bret = fps.CaptureFinger(true);
  int iret = 0;
  if (bret != false)
  {
    Serial.println("Remove finger");
    fps.Enroll1();
    while(fps.IsPressFinger() == true) delay(100);
    Serial.println("Press same finger again");
    while(fps.IsPressFinger() == false) delay(100);
```

Continuación del apéndice 1.

```
    bret = fps.CaptureFinger(true);
    if (bret != false)
    {
      Serial.println("Remove finger");
      fps.Enroll2();
      while(fps.IsPressFinger() == true) delay(100);
      Serial.println("Press same finger yet again");
      while(fps.IsPressFinger() == false) delay(100);
      bret = fps.CaptureFinger(true);
      if (bret != false)
      {
        Serial.println("Remove finger");
        iret = fps.Enroll3();
        if (iret == 0)
        {
          Serial.println("Enrolling Successful");
        }
        else
        {
          Serial.print("Enrolling Failed with error code:");
          Serial.println(iret);
        }
      }
      else Serial.println("Failed to capture third finger");
    }
    else Serial.println("Failed to capture second finger");
  }
  else Serial.println("Failed to capture first finger");
  }


  void loop()
  {
    delay(100000);
  }
```

Continuación del apéndice 1.

- Sensor HC-SR501

```
const int LEDPin = 13;          // pin para el LED
const int PIRPin = 2;           // pin de entrada (for PIR sensor)

int pirState = LOW;             // de inicio no hay movimiento
int val = 0;                    // estado del pin

void setup()
{
   pinMode(LEDPin, OUTPUT);
   pinMode(PIRPin, INPUT);
   Serial.begin(9600);
}

void loop()
{
   val = digitalRead(PIRPin);
   if (val == HIGH)   //si está activado
   {
      digitalWrite(LEDPin, HIGH);  //LED ON
      if (pirState == LOW)  //si previamente estaba apagado
      {
        Serial.println("Sensor activado");
        pirState = HIGH;
      }
   }
   else   //si esta desactivado
   {
      digitalWrite(LEDPin, LOW); // LED OFF
      if (pirState == HIGH)  //si previamente estaba encendido
    {
      Serial.println("Sensor parado");
      pirState = LOW;
    }
 }
```

Continuación del apéndice 1.

- Sensor RC 522RF

sketch_oct10b §

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN  9    //Pin 9 para el reset del RC522
#define SS_PIN  10   //Pin 10 para el SS (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN); ///Creamos el objeto para el RC522

void setup() {
  Serial.begin(9600); //Iniciamos La comunicacion serial
  SPI.begin();        //Iniciamos el Bus SPI
  mfrc522.PCD_Init(); // Iniciamos el MFRC522
  Serial.println("Control de acceso:");
}

byte ActualUID[4]; //almacenará el código del Tag leído
byte Usuario1[4]= {0x4D, 0x5C, 0x6A, 0x45} ; //código del usuario 1
byte Usuario2[4]= {0xC1, 0x2F, 0xD6, 0x0E} ; //código del usuario 2
void loop() {
  // Revisamos si hay nuevas tarjetas  presentes
  if ( mfrc522.PICC_IsNewCardPresent())
        {
      //Seleccionamos una tarjeta
            if ( mfrc522.PICC_ReadCardSerial())
            {
                  // Enviamos serialemente su UID
                  Serial.print(F("Card UID:"));
                  for (byte i = 0; i < mfrc522.uid.size; i++) {
                          Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                          Serial.print(mfrc522.uid.uidByte[i], HEX);
                            ActualUID[i]=mfrc522.uid.uidByte[i];
                  }
                  Serial.print("     ");
                  //comparamos los UID para determinar si es uno de nuestros usuarios
                  if(compareArray(ActualUID,Usuario1))
                    Serial.println("Acceso concedido...");
                  else if(compareArray(ActualUID,Usuario2))
                    Serial.println("Acceso concedido...");
                  else
                    Serial.println("Acceso denegado...");

                  // Terminamos la lectura de la tarjeta tarjeta  actual
                  mfrc522.PICC_HaltA();
```

Continuación del apéndice 1.

- Sensor FC-37

sketch_oct10b §

```
#define PIN_ANALOG_RAIN_SENSOR A1  // Entrada analógica para la señal del sensor lluvia
#define PIN_DIGITAL_RAIN_SENSOR 5  // Entrada digital para la señal del sensor de lluvia

void setup(){
   Serial.begin(9600);
}
void loop(){
   int sensorValue = analogRead(PIN_ANALOG_RAIN_SENSOR); // Leer datos del puerto analógico
   Serial.print("Analog value: ");
   Serial.println(sensorValue); // Salida de valor analógico al monitor de puerto

   sensorValue = digitalRead(PIN_DIGITAL_RAIN_SENSOR); // Leer datos del puerto digital
   Serial.print("Digital value: ");
   Serial.println(sensorValue); // Salida del valor digital al monitor del puerto

   delay(1000); // Retardo entre mediciones
}
```

- Sensor BH1750

sketch_oct10b §

```
#include <Wire.h>
#include <BH1750.h>

BH1750 Luxometro;


void setup(){
  Serial.begin(9600);
  Serial.println("Inicializando sensor...");
  Luxometro.begin(BH1750_CONTINUOUS_HIGH_RES_MODE); //inicializamos el sensor
}


void loop() {
  uint16_t lux = Luxometro.readLightLevel();//Realizamos una lectura del sensor
  Serial.print("Luz(iluminancia):  ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(500);
}
```

Continuación del apéndice 1.

- Sensor KY-026

sketch_oct10a §

```
const int sensorMin = 0; // Rango mínimo del sensor
const int sensorMax = 1024; //Rango máximo

void setup() {
Serial.begin(9600);

}

void loop() {
int sensorReading = analogRead;
int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

//Valor del rango:
switch (range) {
case 0:
Serial.println("** Peligro, Fuego cercano **");
break;
case 1:
Serial.println("** Fuego distante **");
break;
case 2:
Serial.println("Fuego no detectado");
break;
}
delay(1200); // Cantidad de tiempo en que Arduino tomara la lectura del sensor
}
```

Fuente: elaboración propia.