UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

SISTEMA DE MONITOREO Y CONTROL DE LOS PARÁMETROS OPERATIVOS DE LOS TRANSMISORES DE RADIODIFUSIÓN FM APLICANDO UNA ESTRUCTURA IOT Y REDES SDN

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Electrónica y Comunicaciones.

ÁREA: Electrónica y Comunicaciones
LÍNEA DE INVESTIGACIÓN: Programación y redes.
AUTOR: Jonathan Alexander Ronquillo Gómez.
TUTOR: Ing. Víctor Santiago Manzano Villafuerte, Mg.

AMBATO - ECUADOR
septiembre - 2022

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: SISTEMA DE MONITOREO Y CONTROL DE LOS PARÁMETROS OPERATIVOS DE LOS TRANSMISORES DE RADIODIFUSIÓN FM APLICANDO UNA ESTRUCTURA IOT Y REDES SDN, desarrollado bajo la modalidad Proyecto de Investigación por el señor Jonathan Alexander Ronquillo Gómez, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, septiembre 2022.

Firmado electrónicamente por:
VICTOR SANTIAGO MANZANO VILLAFUERTE

Ing. Víctor Santiago Manzano Villafuerte, Mg.

TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: SISTEMA DE MONITOREO Y CONTROL DE LOS PARÁMETROS OPERATIVOS DE LOS TRANSMISORES DE RADIODIFUSIÓN FM APLICANDO UNA ESTRUCTURA IOT Y REDES SDN es absolutamente original, autentico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, septiembre 2022.

Jonathan Alexander Ronquillo Gómez

C.C. 1805149653

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Jonathan Alexander Ronquillo Gómez, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE MONITOREO Y CONTROL DE LOS PARÁMETROS OPERATIVOS DE LOS TRANSMISORES DE RADIODIFUSIÓN FM APLICANDO UNA ESTRUCTURA IOT Y REDES SDN, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, septiembre 2022.

Firmado electrónicamente por:
CARLOS HUMBERTO SANCHEZ ROSERO

Ing. Carlos Sánchez, Mg.
PRESIDENTE SUBROGANTE
DEL TRIBUNAL

Firmado electrónicamente por:
JUAN PABLO PALLO NORONA

Ing. Juan Pablo Pallo, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
JULIO ENRIQUE CUJI RODRIGUEZ

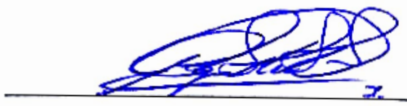
Ing. Julio Enrique Cuji, Mg.
PROFESOR CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, septiembre 2022



Jonathan Alexander Ronquillo Gómez
C.C. 1805149653
AUTOR

DEDICATORIA

El presente trabajo investigativo se lo dedico a mis padres, las personas más importantes en mi vida, quienes me han apoyado incondicionalmente en todo el transcurso de mi vida y cuyos buenos valores me han enseñado a trabajar duro para conseguir que mis sueños se cumplan en metas alcanzadas.

Un sentimiento especial de gratitud a cada integrante de mi familia, amigos y a todas las personas quienes me han brindado su apoyo durante todo este proceso.

AGRADECIMIENTOS

Este proyecto no habría sido posible sin el apoyo incondicional, inequívoco y cariñoso de mis padres, los cuales siempre creyeron en mí y supieron darme ánimos en los momentos difíciles.

Asimismo, quiero expresar mi gratitud a los docentes de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial quienes me han brindado de sabiduría en toda mi formación académica.

A mi tutor de tesis Ing. Santiago Manzano por la orientación, seguimiento, y supervisión continua en el desarrollo de este proyecto.

Y por último y no menos importante, quiero expresar mi gratitud al Ing. William Barriga jefe técnico del departamento de Ecuatronix en Ambato, quien me permitió llevar a cabo la presente investigación.

ÍNDICE GENERAL

Índice de Tablas

Índice de Figuras

RESUMEN EJECUTIVO

En el presente proyecto de investigación se implementa un sistema de monitoreo y control de los parámetros operativos de los transmisores de radiodifusión FM en el cerro Pilishurco, aplicando una estructura IoT y redes SDN, permitiendo realizar un breve diagnóstico del equipo ante distintas fallas que se produzcan en el equipo, identificar la eficiencia energética del mismo, configurar la potencia transmitida del equipo por medio de una interfaz web a la que se puede acceder fuera de la red local y un sistema de alertas que notificaran al técnico y demás personal autorizado las distintas reglas que se han cumplido, a través de la red social de Whatsapp.

El desarrollo del proyecto está compuesto por tres etapas, la primera es la configuración de la red SDN de ZeroTier Inc. en el servidor, enrutador y clientes, esto permite que los distintos servicios web del sistema como el bróker MQTT, base de datos y la aplicación web sean accesibles fuera de la red local de manera segura y confiable por parte de los clientes y el dispositivo IoT. La segunda parte consiste en el desarrollo de una aplicación web bajo el lenguaje de Vue.js y el motor de ejecución de código JavaScript denominado node.js, que actúa como interfaz de comunicación web entre el usuario y el dispositivo IoT adaptado al transmisor RF, y permite crear diversos usuarios con distintos dispositivos asociados a ellos, la creación de distintas plantillas y mesas de trabajo para la visualización y control de diversas variables, y la configuración de reglas aplicadas a las mismas.

La tercera parte es el desarrollo del dispositivo IoT, que se basa en un microcontrolador con capacidad de conexión WiFi, que cumple las funciones de adquisición y acondicionamiento de señales provenientes por los sensores y módulos instalados en el equipo para el monitoreo de parámetros eléctricos, parámetros de propagación de señal RF e imágenes del estado del transmisor, para posteriormente ser enviadas por medio del protocolo MQTT al servidor para luego distribuir esta información a los distintos clientes. El dispositivo IoT igualmente procesa los mensajes provenientes por el cliente para accionar los distintos actuadores adaptados al equipo.

Palabras clave: Redes SDN, desarrollo web, sistemas IoT, telemetría, transmisor RF.

ABSTRACT

This research project implements a monitoring and control system of the operating parameters of the FM broadcasting transmitters in the Pilishurco hill, applying an IoT structure and SDN networks, allowing to perform a brief diagnosis of the equipment in case of different failures that may occur in the equipment, identify the energy efficiency of the equipment, configure the transmitted power of the equipment through a web interface that can be accessed outside the local network and a system of alerts that will notify the technician and other authorized personnel of the different rules that have been met, through the Whatsapp social network.

The development of the project is composed of three stages, the first one is the configuration of the ZeroTier Inc. SDN network in the server, router and clients, this allows the different web services of the system such as the MQTT broker, database and web application to be accessible outside the local network in a secure and reliable way by the clients and the IoT device. The second part consists of the development of a web application under the Vue.js language and the JavaScript code execution engine called node.js, which acts as a web communication interface between the user and the IoT device adapted to the RF transmitter, and allows the creation of different users with different devices associated with them, the creation of different templates and work tables for the visualization and control of various variables, and the configuration of rules applied to them.

The third part is the development of the IoT device, which is based on a microcontroller with WiFi connection capability, which performs the functions of acquisition and conditioning of signals from the sensors and modules installed in the equipment for monitoring electrical parameters, RF signal propagation parameters and images of the transmitter status, which are then sent via the MQTT protocol to the server to then distribute this information to the various clients. The IoT device also processes the messages coming from the client to trigger the different actuators adapted to the equipment.

Keywords: SDN networks, web development, IoT systems, telemetry, RF transmitter.

CAPÍTULO I

MARCO TEÓRICO

1.1.Antecedentes Investigativos

En los últimos años se han desarrollado varios trabajos de investigación con relación a la estructura del internet de las cosas IoT basados en redes definidas por software SDN, lo que representa una herramienta fundamental en el diseño de sistemas de telemetría, industria 4.0, entre otras distintas aplicaciones.

En el año 2021, Katayoun Bakhshi, Amir Masoud y Amir Sabbagh publicaron el artículo de investigación denominado "A fault-tolerant architecture for internet-of-things based on software-defined networks" dentro de la revista Telecommunication Systems, propone una arquitectura integral basado en redes definidas por software SDN que implementa un esquema basado en un modelo matemático denominado Grupo de Enlaces de Riesgo Compartido(SRLG) que calcula las rutas redundantes así como las rutas principales y de respaldo no superpuestas entre los equipo de la red, como resultado de esta investigación se ha mejorado los parámetros de servicios en la red y una rápida recuperación ante fallos de la misma.[1]

En el trabajo de Anichur Rahman, Umme Sara y Dipanjali Kundu titulado "DistB-SDoIndustry: Enhancing Security in Industry 4.0 Services based on Distributed Blockchain through Software Defined Networking-IoT Enabled Architecture", publicado en el año 2020 en la revista International Journal of Advanced Computer Science and Applications (IJACSA) implementaron un sistema basado en Blockchain distribuida en un entorno de redes definidas por software esta investigación permitió que los distintos servicios de la industria 4.0 posean mayor robustez, privacidad y fiabilidad frente a los sistemas tradicionales. [2]

El artículo de investigación denominado “Secure and Reliable IoT Networks Using Fog Computing with Software-Defined Networking and Blockchain” publicado en el año 2019 por Ammar Muthanna, Abdelhamied Ateya y Abdukidir Khakimov dentro de la revista Journal of Sensor and Actuator Networks, implementan un firmware que emplea una capa denominada Fog Layer los mismo que son gestionados bajo los paradigmas de las redes definidas por software y blockchain, lo cual permitió obtener una mayor eficiencia en términos de latencia y utilización de recursos; venciendo así la gran mayoría de los retos que implica el Internet de las Cosas IoT como la seguridad, el tráfico masivo, alta disponibilidad y fiabilidad. [3]

En el año 2017 se publicó el artículo de investigación denominado “Software-Defined Fog Network Architecture for IoT” por Slavica Tomovic, Kenji Yoshigoe, Ivo Maljevic e Igor Radusinovic dentro de la revista Wireless Personal Communications; el cual implementan una arquitectura del Internet de las Cosas IoT no tradicional, haciendo uso de las redes definidas por software y la computación tipo Fog, como resultado de esta investigación la arquitectura IoT soporta un alto nivel de escalabilidad y la entrega de datos con mínima latencia, esta arquitectura es capaz de resolver problemas relacionados con el tráfico datos y la gestión de recursos. [4]

El artículo de investigación denominado “IoT-based Real-Time Telemetry System Design: An Approach (2017) Conference” publicado en 5th International Conference on Future Internet of Things and Cloud (FiCloud) por Ahmet Albayrak, se implementa un sistema de telemetría mediante la arquitectura del Internet de las cosas IoT aplicado a un entorno automovilístico mediante la tecnología de comunicaciones Zigbee, lo que permitió obtener así, un sistema de baja potencia y bajo coste para el seguimiento remoto de los vehículos.[5]

1.2.Contextualización del Problema

En el Ecuador según registros por el INEC, al 2020 el país contaba con una población de 17.510.643 habitantes de los cuales 11.031.705 residen en zonas urbanas, mientras que 6.478.937 residen en zonas rurales. Para mantener informado a la población con distintos acontecimientos a nivel nacional e internacional de manera rápida y oportuna, aparece la radiodifusión sonora como el principal medio de comunicaciones de fácil acceso pues se estima que el 75% de las viviendas posee un receptor de radio[6], y que ofrece una gran cobertura a nivel nacional debido a que existen más de 1266 estaciones de radio con transmisores FM o AM en puntos geográficamente estratégicos de difícil acceso. [7]

La provincia de Tungurahua posee una topografía montañosa y cuenta con una población de 504.583 personas, de los cuales el 59.26% habita en zonas rurales, mientras que el 40.73% en zonas urbanas según el último censo nacional (2010). Para mantener informada a la población tungurahuense las estaciones de radio buscan ubicar sus transmisores en zonas altas y alejados de las ciudades para una mejor propagación de su señal, como es el caso del cerro Pilishurco que se encuentra a más de 4000 metros de altura sobre el nivel del mar y a 45 minutos de la ciudad de Ambato. Siendo esta lejanía lo que dificulta el rápido actuar por parte del departamento técnico a cargo de los equipos de transmisión FM ante posibles fallos.[8]

En la actualidad el cerro Pilishurco acoge a más de 49 empresas dedicadas a prestar servicios de televisión abierta y radiodifusión sonora quienes instalan sus equipos de transmisión en este lugar [9]. Los transmisores cuentan con una vida útil de 5 años y la renovación de los mismos conlleva a realizar exorbitantes inversiones de dinero que van desde los 15.000 USD hasta los 30.000 USD, por lo que las estaciones de radio utilizan estos equipos más allá del tiempo de vida útil de los transmisores, los cuales sin un adecuado mantenimientos preventivo, su rendimiento puede verse comprometido, reduciendo así la potencia de propagación y daños a futuro en los componentes electrónicos del equipo [10].

El consumo energético en la provincia de Tungurahua fue de alrededor del 554.700 GWh en el año 2019, de los cuales el 4.92% fue energía no útil y el restante fue utilizado en distintos sectores como alumbrado público, zonas residenciales, comerciales e industriales[11]. Esta última zona es considerada para la tarifación del consumo eléctrico de los transmisores FM, que con una eficiencia del 95% y una potencia de transmisión de 3KW, mensualmente tienen un consumo de $250 USD, esta cantidad puede aumentar debido al excesivo consumo de potencia reactiva que muchos transmisores tienden a consumir una vez sobrepasado el tiempo de vida útil, reduciendo su eficiencia energética entre el 90% y el 70%, lo que conlleva a multas de hasta el doble del consumo eléctrico del equipo y perdidas de energía.[12]

El presente trabajo tiene como finalidad, implementar un sistema de monitoreo y control de parámetros operativos de los transmisores de radiodifusión FM bajo el internet de las cosas IoT en un entorno de redes SDN, lo que permite conocer distintos parámetros del transmisor como la potencia transmitida, la potencia reflejada, el consumo de corriente AC y voltaje AC, el factor de potencia, energía consumida; así como la capacidad de modificar la potencia de transmisión, y el apagado y encendido del equipo de manera remota e instantánea, obteniendo una gestión eficiente del equipo, además incorpora un sistema de alertas que permite el rápido actuar del personal técnico a cargo ante posibles fallos en el equipo, como pérdida de potencia o sobrecargas energéticas.

1.3.Fundamentación Teórica

1.3.1. Sistema de Comunicaciones Analógicas

Un sistema de comunicaciones es la transmisión de información de manera rápida, segura y eficiente desde un punto denominado fuente hacia otro punto denominado destino, existen dos tipos dependiendo la fuente de información que se utilice, como son los analógicos y los digitales, cada uno con sus distintos métodos de transmisión. Se denominan analógicos debido a que la señal en su fuente varia de forma continua en el tiempo, por lo general es utilizado para la transmisión de audio y video como la radio FM o AM, la televisión analógica y telefonía entre otros.[13]

Un sistema de comunicaciones se lo puede representar de manera general bajo el siguiente diagrama:

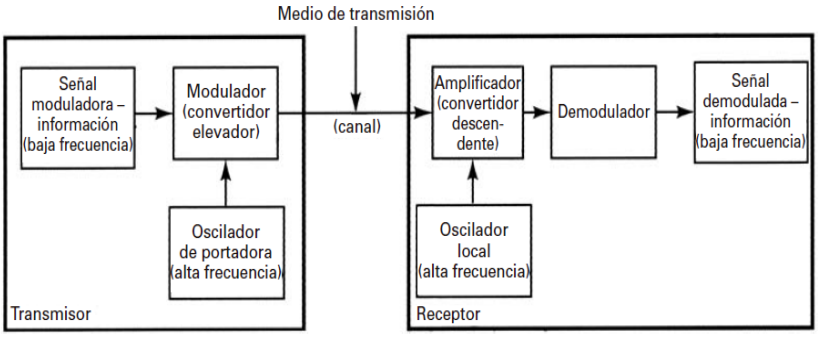


Figura 1: Diagrama de bloques de un sistema de comunicaciones[13]

En todo sistema de comunicaciones la modulación es un proceso importante ya que permite que dichos sistemas puedan irradiar la señales de información en este caso información analógica, haciendo que el sistema se adapte, es decir que se pueda utilizar los medios no guiados como el aire como su medio de transmisión; y además permite que no exista interferencia entre señales con la misma banda de frecuencias como la del audio que está en la banda de los 300 Hz a 15 KHz, es decir que permite multiplexar distintas señales en un mismo medio como es el aire.

En los sistemas de comunicación analógica existen dos tipos la modulación por amplitud y la modulación por frecuencia o modulación angular. [13, 14]

A. Transmisión por modulación angular

Este tipo de modulación es ampliamente utilizada ya que presenta muchos beneficios como mayor ancho de banda y mayor resistencia ante el ruido e interferencias; la diferencia con la modulación AM es que en la modulación angular la amplitud es constante, además es la frecuencia FM o fase PM de la señal portadora que cambia en función del mensaje o información a transmitir; cabe recalcar que son más complejos que la modulación anterior ya que requiere de muchos procesos de control en caso de errores, puede haber de dos tipos moduladores de fase PM y moduladores de frecuencia FM.[15]

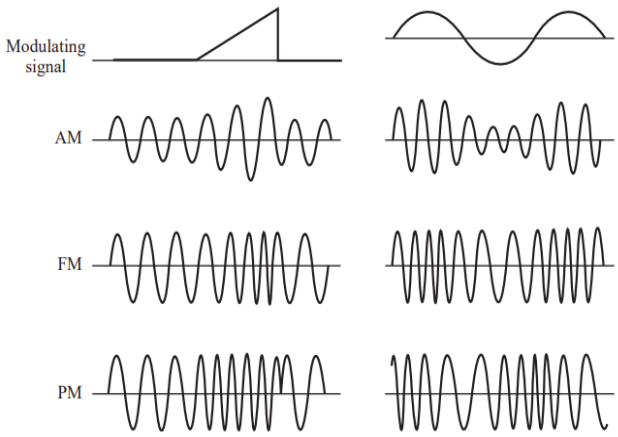


Figura 2: Formas de ondas de los distintos tipos de modulación[16]

De este tipo de modulación se puede apreciar los siguientes puntos:

Tabla 1: Principales puntos fuertes y débiles de la modulación angular.[15]

Ventajas	Desventajas
Presenta mayor inmunidad al ruido o interferencias.	Su circuitos son más complejos y por ende más costosos
Trabaja en la banda de frecuencias de VHF (30MHz – 300MHz).	No poseen mayor alcance respecto a la modulación AM
Permite una mayor calidad en el audio.	Utilizan un gran ancho de banda.

Modulador de fase PM

Es el encargado de cambiar la fase de la señal portadora en función de la señal de la información, una de sus ventajas es que al ocupar osciladores independientes para la frecuencia de la señal portadora permite una frecuencia de transmisión más estable ya que la frecuencia y la amplitud de la portadora permanecen constantes, se lo puede implementar mediante diodos tipo varactor o transistores. [16]

Modulador de frecuencia FM

Su función es la de cambiar la frecuencia de la portadora en relación con la señal de la información o denominada moduladora, mediante un análisis matemático se puede decir que la modulación de frecuencia es una derivada de la función de fase, por lo cual están estrechamente relacionados. A diferencia que la modulación AM que la información está contenida en la envolvente de la frecuencia portadora, la señal FM no posee envolvente debido a que la amplitud de la portadora es constante y por ende

es más tolerante a ruidos o interferencias, lo que conlleva a utilizar mayor ancho de banda, además la potencia total es igual a la de la potencia de la portadora no modulada. En la actualidad es ampliamente utilizada en la radiodifusión. [17]

B. Transmisores FM

En la actualidad existen muchos transmisores FM comerciales, los cuales siguen el esquema general de la figura 3, el cual detalla las etapas principales de un transmisor FM. La señal del audio pasa a un preamplificador con la finalidad de que la señal aumente su amplitud y sea apta para el proceso de modulación, la misma que se encarga de combinar la señal de audio pre amplificada y una señal de alta frecuencia obteniendo así una señal FM, para que esta señal resultante se propague y tenga mayor cobertura se utiliza un amplificador el cual eleva su potencia y por último, por medio de un sistema radiante poder propagarla.

En los transmisores FM, la principal característica es poder transmitir dos canales de audio, como son: el canal principal, el cual los receptores monofónicos lo utilizan y el canal secundario, en caso de que el receptor sea estereofónico, en este caso el receptor hace una resta entre el canal primario y secundario a fin de obtener el canal izquierdo L y el canal derecho R. [13, 15]

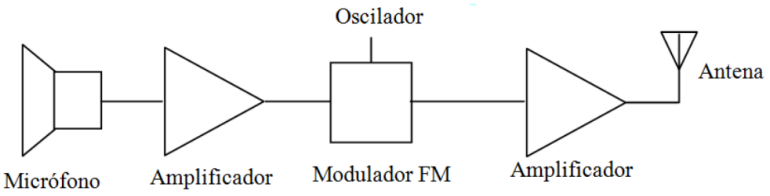


Figura 3: Diagrama de bloques general de un transmisor FM. [18]

Los parámetros técnicos generales que se debe revisar en un transmisor comercial son:[19]

- Rango de frecuencias [MHz].
- Potencia nominal de salida [KW].
- Temperatura de trabajo [℃].
- Humedad [%].

- Altitud de trabajo [m].
- Relación señal ruido dB.
- Tensión de alimentación de entrada AC [V_{AC}].
- Consumo de potencia aparente de AC [VA].
- Consumo de potencia activa de AC [W].
- Factor de potencia.

1.3.2. Sistemas IoT

Hoy en día el internet de las cosas se ha convertido en un gran aliado para los procesos de automatización y control en la industria y en muchos otros campos, estos sistemas están orientadas en tres aspectos como: orientadas al internet, es decir lograr una conexión eficiente entre los dispositivos por medio de la infraestructura de red basada en IP; es orientado a las cosas, lo que implica integrar los objetos físicos al mundo virtual o cibernético por medios capaces de identificarlos y posterior integrarlos; y por último es orientado a la semántica que permite que los datos de los dispositivos se puedan almacenar, gestionar y representar. De manera general los requerimientos de un sistema IoT consisten en: [20]

- Poseer una red de sensores para la recolección de datos.
- Incorporar un sistema de alertas que se generan cuando se cumplan determinadas reglas.
- Un sistema de análisis sobre los datos recolectados por los sensores los cuales deben ser continuos.
- Deben poder reaccionar ante el análisis de los datos para que activen los actuadores.
- Deben poseer un sistema de control para que con un algoritmo de control en función de los datos de entrada generen una determinada salida sobre los actuadores.
- La latencia debe ser mínima.
- De ser fiable y de alta disponibilidad.

Por lo que los sistemas IoT deben detectar, actuar y comunicar entre los distintos dispositivos para así obtener una percepción del entorno que los rodea además de poder reaccionar de manera autónoma ante los eventos que se presenten en el entorno, lo que conlleva que el sistema sea controlado con o sin intervención humana.[21]

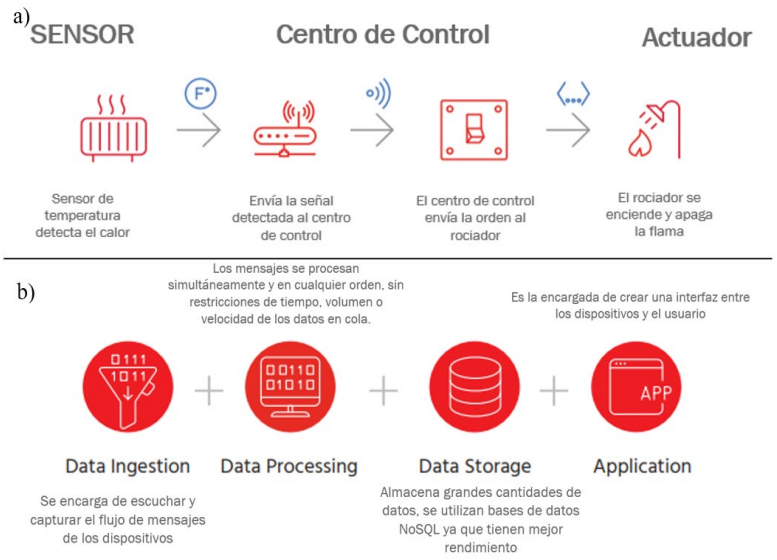


Figura 4: Sistemas IoT a) Estructura de hardware; b)Estructura de software. [22]

A. Arquitecturas y protocolos IoT

La arquitectura IoT abarca un sinfín de propuestas debido a las distintas necesidades a cubrir, por ende su arquitectura no está del todo estandarizada; la Figura 5 se detalla de forma general las capas de la arquitectura IoT, como son: la capa de dispositivos y actuadores que son gestionados por un software integrado; la capa de conectividad la cual se encargada de establecer una comunicación entre los dispositivos y la capa de aplicación esta última encargada de procesar los datos enviados por los dispositivos. [21]

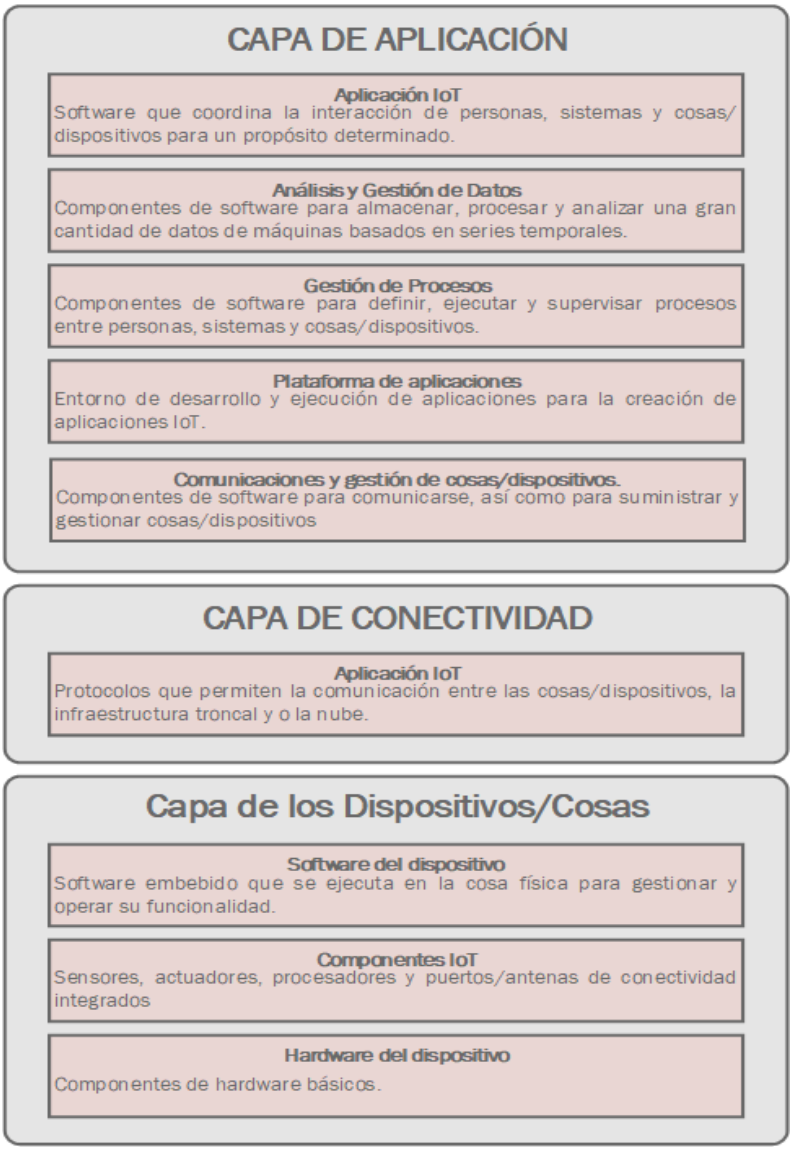


Figura 5: Capas generales de la arquitectura de un sistema IoT [21]

La arquitectura IoT se puede comparar con las capas del modelo TCP/IP, teniendo en cuenta que no abarca todas las tecnologías de este modelo, como se muestra en la Figura 6 en donde se exponen los principales protocolos de comunicación para que el sistema IoT funcione. Dichas tecnologías y protocolos han sido adaptadas a los requisitos del sistema, ya que los dispositivos no cuentan con mucha memoria de almacenamiento, además deben ser eficientes energéticamente, ofrecer comunicaciones fiables, ser altamente escalables entre otros.[21]

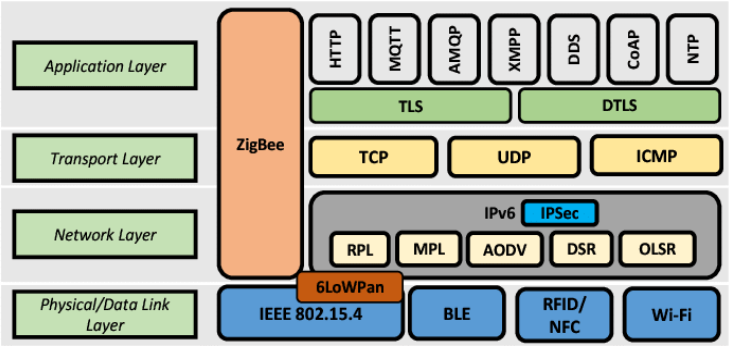


Figura 6: Arquitectura IoT en base al modelo TCP/IP[23]

En la actualidad existen muchas tecnologías alámbricas e inalámbricas, para lo cual se debe seleccionar la que más se adapte a las capacidades del nodo en función de la potencia y cobertura, en la tabla 2 se muestran las tecnologías más empleadas en las arquitecturas IoT.[21]

Tabla 2: Características de las tecnologías de comunicación en los sistemas IoT. [24]

Tecnología	Frecuencia de operación	Cobertura	Velocidad de transmisión	Latencia	Consumo Energético	Costo
PLC (Power line Communications)	120KHz	≈ 9 Km	10 Mbps	Varía	Medio	Medio
802.15.1 Bluetooth	2.4 GHz	≈ 8 – 10 m	1 – 24 Mbps	Bajo	Bajo	Bajo
802.11 WiFi	2.4/5/60GHz	≈ 20 m – 100 m	1 – 600 Mbps	Varia	Medio	Bajo
802.16 Wimax	2.3/3.5GHz	≈ 70 Km	100 Mbps – 1 Gbps	Varia	Alto	Alto
ZigBee 802.15.4 LR-WPAN	868 /915 MHz – 2GHz	≈ 8 – 10 m	40 – 250 Kbps	Bajo	Bajo	Medio
SigFox	Bandas regionales de sub-GHz	≈ 15 Km	100 bps de subida 600 bps de bajada	Alta	Bajo	Alto
LoRa	Bandas regionales de sub-GHz	≈ 13 Km	0.3 – 50 Kbps	Baja	Bajo	Medio
2G/3G/4G/5G	900/800 /1900 MHz	Varia	9.6 Kbps (2G) – 100 Mbps (4G)	Varia	Alto	Alto

En la actualidad se adoptan aplicaciones de la tecnologías de la información los cuales resuelven algunas limitaciones de comunicación de los sistemas IoT como: la tecnología 5G que permite un mayor número de usuarios con una mejor cobertura y velocidad de datos; LiFi fidelidad a la luz que mediante sus sistema de LED's de bajo consumo energético hace que el sistema sea de bajo costo aunque es muy susceptible al ruido e interferencias y por último las redes definidas por software SDN que proporciona a la arquitectura IoT que su red sea dinámica, gestionable, adaptable y rentable. [21]

Otro parámetro que define la arquitectura de los sistemas IoT son los protocolos que se utilizan en la capa de sesión de la Figura 6, en este aspecto se tiene los principales protocolos que son:

- **Protocolo CoAP**

Protocolo de aplicación restringida (CoAP) es un protocolo de mensajería ligera que emergió en el año 2010 el cual permite la comunicación por medio de dos modelos como Solicitud/Respuesta y Publicación/Suscripción; además está diseñado para ser interoperable con el protocolo HTTP por medio de la transferencia de estado representacional RESTful; entre la ventajas y desventajas de este protocolo se tiene:

Tabla 3: Aspectos positivos y negativos del protocolo CoAP. [23]

Ventajas	Desventajas
Se ejecuta sobre UDP (Protocolo de datagramas de usuario) que ofrece menos carga a la red en comparación con TCP	Se requiere enviar mensajes repetitivos para que el sistema sea fiable.
Implementa seguridad de la capa de transporte (DTLS) para brindar comunicaciones UDP encriptadas.	Es un protocolo en desarrollo y por ende es menos estable que MQTT; además podría a futuro tener problemas de interoperabilidad.

Comunicaciones Asíncronas ya que utiliza el modelo request/report para intercambiar mensajes.	Implementa un enfoque de mensajes no confírmales y los confírmales, los cuales confirman que el mensaje ha llegado pero no, en el caso de se haya decodificado correctamente.
Ofrece similitudes con HTTP y por ello para los desarrolladores web es más fácil de implementar este protocolo	Al utilizar UDP hace que exista mayor dependencia sobre la capa de aplicación ya que esta realizara las tareas que realiza TCP.

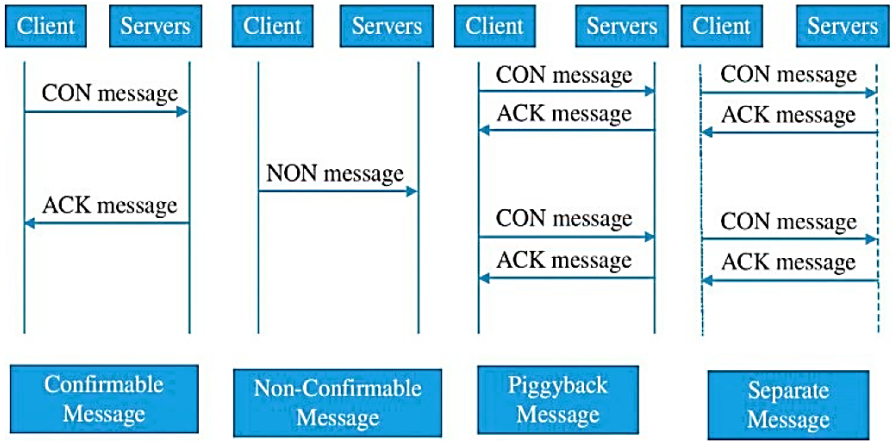


Figura 7: Estructura del protocolo CoAP. [25]

- **Protocolo XMPP**

Protocolo extensible de mensajería y presencia XMPP, es un protocolo de comunicaciones que se lo presento originalmente para aplicaciones de chat e intercambio de mensajes, posee baja latencia y se ejecuta sobre el protocolo TCP, al igual que el protocolo CoAP puede funcionar en ambos modelos Solicitud/Respuesta y Publicación/Suscripción, cabe recalcar que no ofrece garantía de calidad de servicio volviéndola no práctica para la comunicación máquina a máquina M2M.[25]

Tabla 4: Aspectos positivos y negativos del protocolo XMPP[26]

Ventajas	Desventajas
Protocolo de código abierto estandarizado por el Grupo de Trabajo de Ingeniería de Internet (IETF).	No cuenta con el mecanismo de calidad de servicio QoS.
Implementa seguridades como TLS y SSL para el cifrado punto a punto.	Al utilizar XML sobrecarga la red por la transmisión de texto mas no en binario.
No requiere un servidor central para manejar los demás nodos.	Es muy poco utilizado en los sistemas IoT, por lo cual puede llegar a tener problemas con la incompatibilidad.

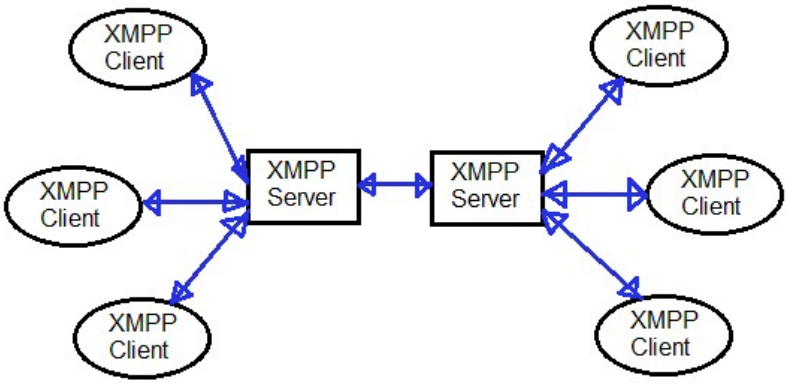


Figura 8: Estructura del protocolo XMPP[26]

- **Protocolo MQTT**

Transporte de telemetría de cola de mensajes (MQTT) es el protocolo de comunicaciones máquina a máquina M2M más ligero y fiable que apareció en el año 1999 por IBM, utiliza el modelo Publicación/Suscripción bajo el esquema TCP, en la Figura 9 se puede visualizar la estructura centralizada de comunicaciones de MQTT, como se muestra el Bróker es el que gestiona el ingreso de los mensajes publicados por los clientes y los reenvía a sus respectivos suscriptores; mientras que los clientes pueden realizar varias opciones como publicar y suscribirse a un "Tópico". La característica principal de este protocolo es su desacoplamiento tridimensional, es decir de espacio ya que el publicador y el suscriptor no tienen por qué conocerse; de tiempo ya que la información del tópico no necesariamente debe llegar en tiempo real y de sincronización ya que no necesita sincronización.[21]

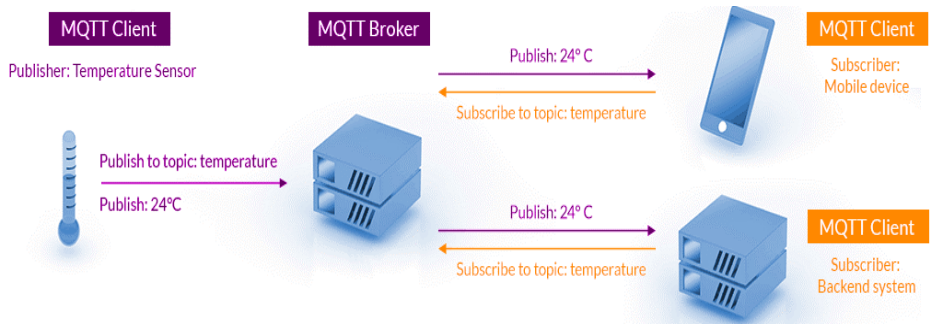


Figura 9: Estructura del protocolo MQTT [22]

Este protocolo ofrece tres tipos de calidad de servicio como son:

Tabla 5: Tipos de calidad de servicio en MQTT.[21]

Tipo de QoS	Funcionamiento
QoS 0	Se deja que el mensaje viaje sobre el esquema de la red TCP y su redundancia; el mensaje va a llegar a su destino como máximo una vez y es el que menos carga genera al BROKER y menos tráfico genera en la red.
QoS 1	se tiene la garantía de que al menos una vez se entregara el mensaje a los suscriptores, generando congestión en la comunicación ya que se generara protocolos de redundancia y de confirmación, también pueden existir mensajes duplicados.
QoS 2	MQTT se va a asegurar que el suscriptor reciba el mensaje solo una vez, este nivel es ideal para cuando mensajes duplicados pueden existir en el mensaje, se utiliza en redes malas.

MQTT ofrece múltiples parámetros de configuración como por ejemplo:

- **Sesiones Limpias**.- parámetro que pertenece a la conexión, cuando un cliente se conecta al bróker puede establecer una sesión limpia, esto quiere decir que si el cliente se desconecta o se da de baja cuando vuelva en línea los mensajes no estarán disponibles para él, perdiéndose los mensajes. En caso de no establecerse una sesión limpia el bróker guarda los mensajes y luego cuando se conecte el suscriptor le llegaran todos los mensajes guardados por el bróker.[27]

- **Ultimo Testamento**.- parámetro que se configura sobre la conexión, si se lo aplica el cliente envía al bróker un último mensaje, el mismo que lo guarda para que en el caso de que el cliente emisor se cae o se desconecta, el mensaje guardado se distribuya para todos los demás clientes, permitiendo así detectar cuando el emisor de un mensaje que periódicamente está enviando datos, se cae y se desconecta, suele utilizarse para definir banderas en caso de desconexión, o registros en base de datos.[28]

- **Retención de mensajes**.- este parámetro actúa sobre la publicación, esto quiere decir que mensaje por mensaje, se va a poder determinar si es un mensaje retenido o no, es decir si un mensaje sale con este parámetro, el mensaje viaja al bróker, el mismo que distribuye ese mensaje a todos los clientes que estén suscriptos, pero guarda una copia del mensaje, para utilizar esa copia en caso de que un cliente nuevo se suscriba al Tópico, el BROKER automáticamente envía el mensaje retenido al nuevo suscriptor.[27]

- **Sesión persistente**.- asegura que todos los mensajes lleguen a un cliente, aun cuando este haya perdido la conexión, el bróker guardará los mensajes y se los volverá a reenviar al cliente una vez que este vuela a reconectarse.[27]

Tabla 6: Aspectos positivos y negativos del protocolo MQTT[26]

Ventajas	Desventajas
Los dispositivos pueden publicar y suscribirse al tópico sin la necesidad de conocer se entre sí.	Al ser centralizado al caerse el bróker todo el sistema de comunicaciones cae, haciendo que los clientes puedan publicar o suscribirse a tópicos.
Un dispositivo puede publicar información sin la necesidad de que su destinatario este activado y recibirá la información una vez que esté conectado.	MQTT al ejecutarse sobre TCP puede llegar a ser una sobrecarga adicional, aunque se lo puede solventar utilizando la otra versión más actual (MQTT-SN).
Se puede cifrar las comunicaciones entre el cliente y el bróker mediante la seguridad TLS/SSL	La centralización limita la escalabilidad del sistema, ya que cada dispositivo cliente ocupa algo de sobrecarga.

Tabla 7: Comparación entre los principales protocolos de datos de los sistemas IoT[21]

	MQTT	**XMPP**	**CoAP**
Protocolo Base	TCP	TCP	UDP
Modelo	Asíncrono	Cerca a tiempo real	Síncrono
Esquema de Comunicación	Publicación/Suscripción	Solicitud/Respuesta Publicación/Suscripción	Solicitud/Respuesta publicación/Suscripción
Tamaño de Cabecera	2 Bytes	Varía	4 Bytes
Tamaño de Mensaje	Hasta los 256MB	20 a 50 KB	Hasta 1MB
Políticas de QoS	3	0	4
Codificación	UTF-8(Binary)	ASCII text	RESTful(Binay)
Seguridad	SSL/TLS	SSL/TLS	DTLS
Multidifusión	Si	Si	No
RESTful	No	No	Si
Estándar	ISO/IEC,OASIS	IETF(RFC3921)	IETF(RFC7252)
Licencia	Open Source	Open Source	Open Source

A. Aplicaciones del IoT en la industria

Debido a que la arquitectura IoT cuenta con distintos protocolos y tecnologías de comunicación que se adaptan a las necesidades de la industria como lo es la telemetría de procesos el cual consiste en medir magnitudes físicas y químicas desde la cadena de producción para posterior a ello enviarlos al operador del proceso.[22]

Dentro de las distintas aplicaciones IoT se tiene: en la Industria 4.0 los cuales utilizan sensores para monitorear y gestionar los procesos industriales, además mediante la recolección de datos de los sensores se pueden predecir fallos en el proceso. En edificios inteligentes que por medio de sensores son capaces de identificar la ubicación de las personas mediante el procesamiento de dichos datos se puede utilizar para diseñar de mejor manera los sistemas de calefacción, aire acondicionado, señalización inteligente para salidas de emergencias, entre otros; implementando sensores más especializados se puede monitorear y controlar el estado estructural de los edificios.[29]

Además las ciudades inteligentes requieren de los sistemas IoT para monitorear mediante sensores el tráfico generado por vehículos y personas, permitiendo que se

gestione de mejor manera el tiempo que en este caso los semáforos ocuparían. Y en los sistemas médicos el sistema IoT se encarga de monitorear en tiempo real los distintos sensores que los pacientes pueden ocupar en sus hogares con el fin de no saturar el sistema de salud, además de distintas aplicaciones más como botones de emergencia, diagnósticos médicos, entre otros.[20]

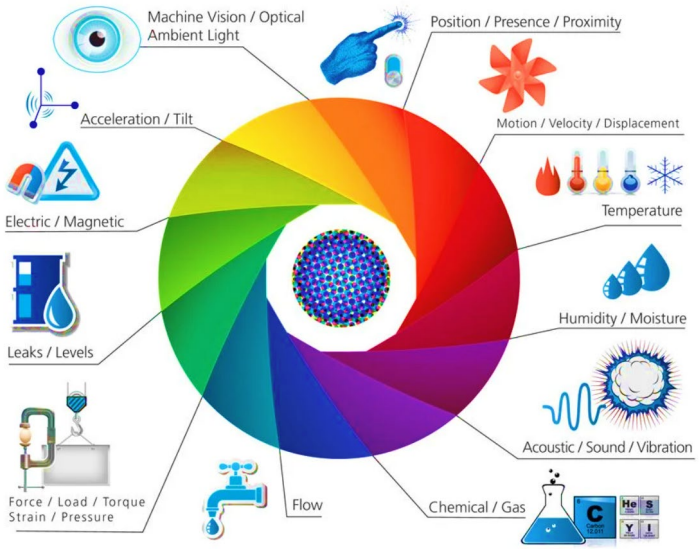


Figura 10: Aplicaciones de los sistemas IoT en distintas áreas.[30]

1.3.3. Placas de Desarrollo

Las placas de desarrollo de la empresa Espressif Systems, utilizan un alto nivel de integración y se caracterizan por lograr un consumo bajo de energía, las nuevas tarjetas de desarrollo tienen como base el integrado de la serie ESP32 que es un CPU de 32 bits con dos núcleos de hasta 240Mhz de velocidad, un convertidor USB a serial, regulador de voltaje e indicadores led. La tarjeta es capaz de utilizar diversas tecnologías de comunicación inalámbrica como: WiFi, Bluetooth y BLE en la banda de 2.4GHz, además cuenta con diversos periféricos para la conexión con sensores táctiles capacitivos, interfaz para la tarjeta SD, Ethernet, SPI, I2C. [31]

Estas tarjetas son programables bajo múltiples lenguajes de programación como C++, pyhton, RTOS, Micropython y en distintos entornos de desarrollo integrado como Platformio IDE y Arduino IDE, en la Tabla 8 se muestran las principales tarjetas de desarrollo de la empresa, con sus respectivas características.

Tabla 8: Principales tarjetas de desarrollo comerciales con conectividad WiFi. [31]

	ESPCAM	**ESP-EYE**	**ESP32-DevKit**	**ESP8266-DevKitC**
Placa				
Modulo	ESP32-S	ESP32	ESP32-WROOM-32	ESP8266MOD
Voltaje de Alimentación	5V	5V	5V	5V
Núcleos	2	2	2	1
Cámara	Si	Si	No	No
Tarjeta SD	Si	No	No	No
Voltaje lógico	3.3V	3.3V	3.3V	3.3V
Puertos UART	1	1	2	2
Puertos SPI	1	1	3	1
Puertos I2C	1	0	2	1
Pines Digitales	9	0	34	9
Pines PWM	10 (16 bits)	0	16 (16-bits)	9 (10-bits)
Pines ADC	7	0	10 (12 bits)	1
Pines DAC	2	0	2 (8 bits)	0
Memoria Flash	4 MB	4 MB	4 MB	4 MB
SRAM	520KB	8 MB	520KB	520KB
Frecuencia de operación	240 MHz	240 MHz	240 MHz	80 MHz
Timers	3 (16 bits)	3 (16 bits)	3 (16 bits)	2 (16 bits)
Protocolo WiFi	b/g/n/e/i			
Rango de frecuencias RF	2412~ 2484 MHz	2412~ 2484 MHz	2.4 ~ 2.5 GHz	2.4 ~ 2.5 GHz
Modos de WiFi	Station, SoftAP	Station, SoftAP	Station, SoftAP, SoftAP+Station, P2P	Station, SoftAP
Protocolos de red soportado	IPv4, IPv6, SSL, TCP, UDP, HTTP, FTP, MQTT			
Bluetooth	v4.2 BLE	NA	v4.2 BLE	-
Precio USD	$15,00	$22,50	$12,00	$7,00

Elaborado por: El investigador

1.3.4. Módulos de adquisición de parámetros eléctricos

Los módulos de adquisición de parámetros eléctricos AC, se basan en dos principios para la obtención del voltaje AC y la corriente AC, para luego utilizar un microcontrolador que junto a un modelo matemático obtener las demás variables eléctricas. El voltaje AC se obtiene reduciendo el voltaje por medio de resistencias o transformadores para tener valores de voltaje pico que el microcontrolador interpreta por medio de un convertidor análogo digital, mientras que la corriente AC utiliza un transformador toroidal, que al atravesar una corriente eléctrica alterna por medio del devanado primario del transformador, se produce un flujo magnético alterno, que induce una corriente alterna en el devanado secundario, corriente que es interpretado por el microcontrolador por medio de un ADC, para la obtención de la corriente AC, este principio es utilizado por muchos módulos comerciales como se detalla en la Tabla 9. [32]

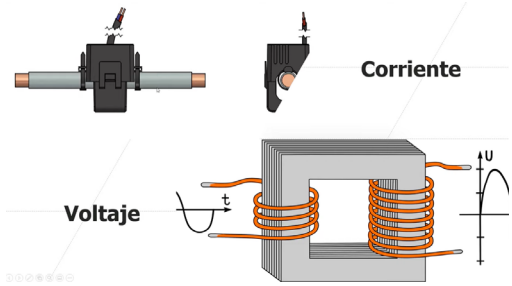


Figura 11: Conexión del transformador de corriente y de voltaje. [32]

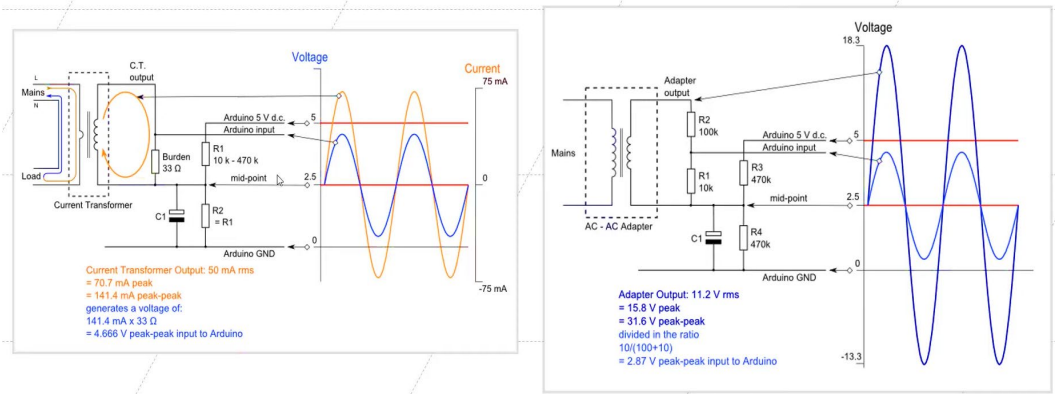


Figura 12: Diagrama eléctrico para la obtención de la señal de voltaje y corriente AC. [32]

Tabla 9: Sensores comerciales de corriente alterna. [32]

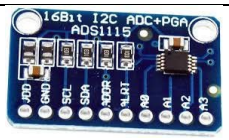
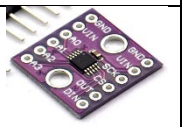
	PZEM-004T	**PZEM-020**	**ZMAi-90**
			
Versión	v3	v1	v1
Interfaz de Comunicación	TTL 9600, 8 bits, 1 stop bit, sin paridad	No	WiFi, UART
Rango de medición Vac	80 ~ 260 V, Resolución: 0.1V, Precisión: 0.5%	80 ~ 260 V, Resolución: 0.5V, Precisión: 1%	90 ~ 240 V, Resolución: 0.1V, Precisión: 1%
Rango de medición Iac	0 ~ 100 A, Resolución: 0.001A Precisión: 0.5%	0 ~ 10 A, Resolución: 0.005A Precisión: 0.8%	5 ~ 60 A, Resolución: 0.00A Precisión: 0.8%
Rango de medición potencia activa Pac	0 ~ 23 KW, Resolución: 0.1W Precisión: 0.5%	0 ~ 2,2 KW, Resolución: 0.5W Precisión: 1%	0 ~ 13,2 KW, Resolución: 0.1W Precisión: 1%
Rango de medición FP	0.00 ~ 1 Resolución: 0.01 Precisión: 1%	0.00 ~ 1 Resolución: 0.01 Precisión: 1%	0.00 ~ 1 Resolución: 0.01 Precisión: 1.5%
Rango de medición de frecuencia	45Hz ~ 65Hz Resolución: 0.1Hz Precisión: 0.5%		
Rango de medición de Energía Activa	0 ~ 9999.99 kWh Resolución: 1 Kh Precisión: 0.5%	0 ~ 999 kWh Resolución: 1 Kh Precisión: 0.5%	0 ~ 1200 kWh Resolución: 1 Kh Precisión: 1%
Precio USD	$ 29,00	$18,00	$ 55, 00

Elaborado por: El investigador

1.3.5. Convertidor análogo a digital

Los convertidores análogos a digital ADC tienen como finalidad transformar las señales analógicas como temperatura, voltaje, corriente, presión en una representación digital de esta señal, para su posterior procesamiento, manipulación o almacenamiento. El principio de funcionamiento del ADC, es muestrear una forma de onda analógica a intervalos de tiempos uniformes, para luego asignar un valor digital a cada muestra, a estos procesos de los denomina muestreo y cuantización. Muchos de los microcontroladores cuentan con un ADC interno, el cual tienen resolución de 8 a 12 bits, a mayor resolución, mejor será la precisión de medida, por lo que en la actualidad se ofrecen convertidores ADC comerciales como se muestra en la Tabla 10. [33]

Tabla 10: Convertidores ADC comerciales. [33]

	ADS 1115	**ADS 1015**	**ADS 1118**
Dispositivo			
Voltaje de operación	2.0V a 5.5V	2.0V a 5.5V	2.0V a 5.5V
Canales ADC	4	4	4
Comparador Integrado	Si	Si	No
Resolución	16 bits	12 bits	16 bits
Muestreo por segundo	8 a 860	128 a 3300	8 a 860
Consumo de corriente	150 uA		
Interfaz de Comunicación	I2C	I2C	SPI
Precio USD	$11,00	$11,00	$8,00

Elaborado por: El investigador

1.3.6. Redes Virtuales

El constante desarrollo de la tecnología de virtualización ha permitido que se puedan crear servidores corriendo en máquinas virtuales, administrarlos de manera sencilla, para los cuales se emplean muchos softwares de virtualización como VMware, Hyper-V, KVM; entre otros; además en los últimos años con la finalidad de crear servidores dedicados más eficientes, fáciles de gestionar y que utilicen la menor cantidad de recursos físicos se ha desarrollado la tecnología de los contenedores, Docker Inc. Es uno de los pioneros en desarrollar este tipo de tecnología. [34]

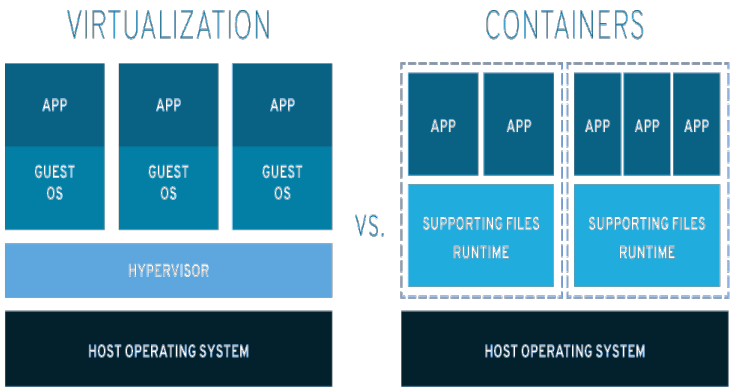


Figura 13: Estructura para la creación de aplicaciones mediante la tecnología de virtualización y contenedores.[34]

Para comunicar todas la máquinas virtuales es necesario hacer uso de las redes virtuales, la cual asigna una gran parte o en su totalidad los recursos de la red a una capa de protocolo específica, que permite asignar a cada aplicación una o varias redes lógicas para intercomunicarse entre cada aplicación. En donde si las máquinas virtuales o contenedores se encuentran en el mismo servidor físico, entonces se crea un conmutador de software para comunicar la máquinas virtuales entre sí; en el caso de que se encuentren en diferentes servidores físicos es necesario la creación de un túnel en la capa 2 del modelo OSI lo que permitirá crear una red virtual local VLAN como se muestra en la Figura 14, donde se puede apreciar la red física y lógica de las máquinas virtuales a comunicar.[35]

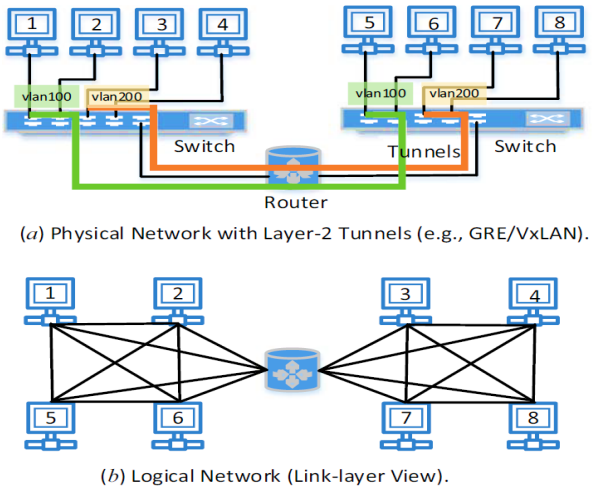


Figura 14: Comunicación entre máquinas virtuales en distintos servidores físicos.[35]

Las redes virtuales lógicas se desvinculan del hardware de la red y pueden aplicar los servicios de capa 2 o capa 3 del modelo OSI como los son la conmutación (switches, bridges, hubs) y el enrutamiento (routers) permitiendo a la organizaciones beneficiarse de la eficiencia y agilidad de los recursos informáticos, aunque para una mayor virtualización es necesario separar los planos de control y de datos como solo las redes definidas por software lo pueden hacer. Las redes virtuales se pueden clasificar en dos grupos como son las redes virtuales definidas en protocolos un ejemplo de ellas son las redes virtuales de área local (VLANs), redes virtuales privadas (VPNs), Servicios de LAN privadas (VPLS); mientras que otro tipo de redes virtuales son las que se basan en dispositivos virtuales como las redes que conectan a las máquinas virtuales, se debe considerar que ambas pueden trabajar en conjunto.[35]

Elementos de las redes virtuales basados en código abierto:[36]

- Quagga software de capa 3 que se encarga del enrutamiento de paquetes.
- Open vSwitch (OVS) está diseñado para actuar como un conmutador de software multicapa.

1.3.7. Redes definidas por software

Las redes definidas por software (SDN) hace hincapié de separar la arquitectura de la red en el plano de control y el plano de datos con la finalidad de obtener mayor control sobre la red, para ello se basa en un software dedicado el cual debe ser capaz de proporcionar un conjunto de interfaces de plataforma de aplicación (API) para abstraer sus recursos físicos y simplificar su gestión y asignación de recursos; tomar acciones y controlar la red basándose en reglas aplicadas a la red y de manera autónoma. Las redes SDN es una arquitectura de red que se caracteriza por ser dinámica, gestionable, rentable y adaptable. Esta arquitectura separa las funciones de control y reenvío de la red, haciendo que el control de la red sea directamente programable y que su infraestructura se disocie para las distintas aplicaciones y los servicios de red. Para ello se hace uso de protocolos y OpenFlow es uno de ellos, permitiendo así crear este tipo de redes. [35]

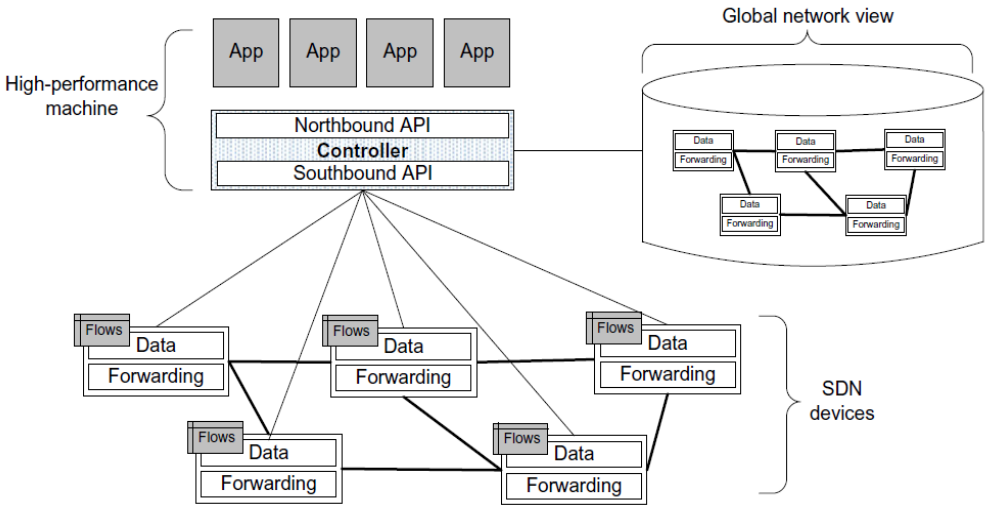


Figura 15: Estructura de una red definida por software[37]

A. Plano de control

El plano de control mantiene actualizada la tabla de reenvío con la finalidad de que el plano de datos pueda manejar la mayor cantidad de tráfico, este plano se encarga de procesar distintos protocolos de control que gestionan la topología de la red.[37]

Se debe tener en cuenta que este plano no genera ni reciben datos, sino más bien actúan como conductos para transportar los datos y que son uno o varios controladores que deben ser capaces de ejercer control sobre los conmutadores virtuales por medio de API's, los controladores suelen utilizar tres interfaces de comunicación. [36]

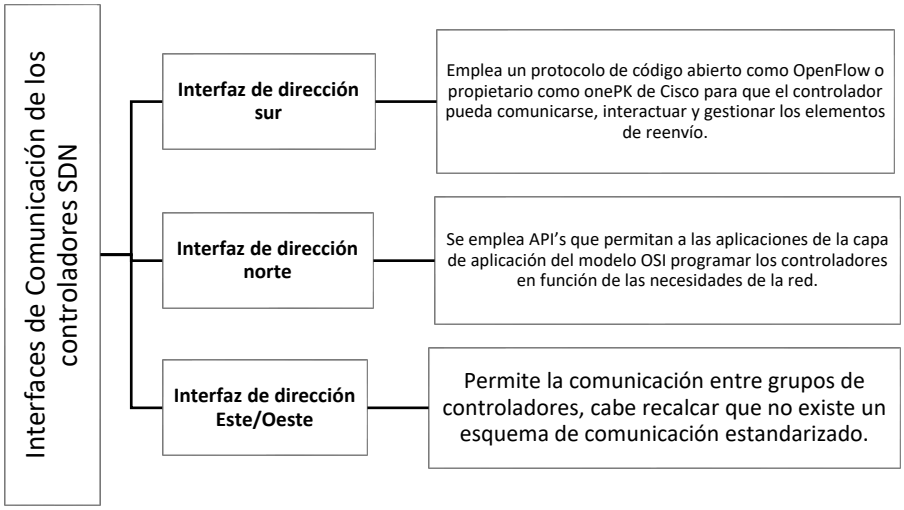


Figura 16: Tipos de interfaces de comunicación de los controladores SDN. [36]

B. Plano de datos

El plano de datos lo conforman los distintos puertos utilizados para la recepción y transmisión de paquetes; y una tabla asociada con el reenvió de los mismos, este plano se responsabiliza de almacenar los paquetes en el búfer del equipo así como de su programación; además de modificar las cabeceras y el reenvío de los paquetes. Sin embargo es necesario que la información de la cabecera de un paquete de datos este registrado en la tabla de reenvío, en caso contrario debe ser tratado por el plano de control. [35]

Una característica especial del plano de datos es que son agnósticos al protocolo que los puntos finales utilizan para sus comunicaciones, los dispositivos en este plano deben ser capaces de comunicarse con los controladores por medios de API's; estos dispositivos pueden ser de dos tipos como los basados en software como Open vSwitch que se mencionó previamente o basados en hardware como los switches Cisco o HP, los cuales tienen habilitados para manejar el protocolo OpenFlow. [37]

C. Protocolo OpenFlow

Es un protocolo encargada de comunicar entre los planos de control y los planos de datos en una arquitectura SDN. La estructura OpenFlow básica está formada por hosts finales, un dispositivo controlador y un dispositivo conmutador como vSwitch que soporten este tipo de protocolo, los conmutadores al soportar este tipo de protocolo ya no se limitan a ser un dispositivo de capa 2 y que se comunica con el controlador por medio de API's. Este protocolo procesa los paquetes entrantes de la siguiente manera:[35, 37]

1. Se rellena al conmutador con entrada de las tabla de flujos, esto lo realiza el controlador.
2. Se inicia una búsqueda de la cabecera del paquete en la tabla de flujos local a manera que coincidan y poderlo redirigir ; en caso contrario se lo envía al controlador para que este lo procese.
3. Los contadores de Bytes y paquetes son actualizados dentro de la tabla de flujos.
4. Se añaden al conjunto de acciones dentro de la tabla de flujos la o las acciones correspondientes a las reglas de flujo.
5. Realizado todo el procesamiento en las tablas de flujo

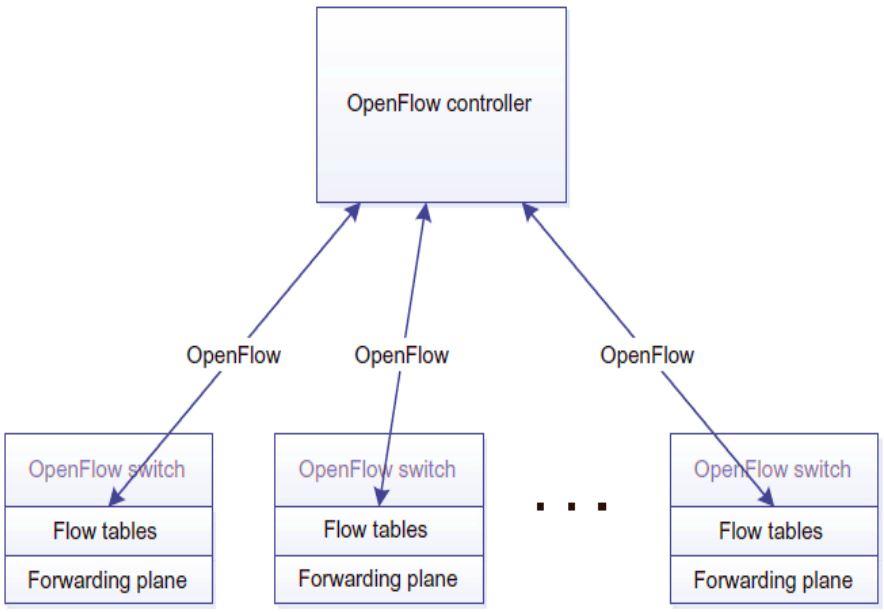


Figura 17: Estructura del protocolo OpenFlow[37]

D. Controladores SDN

El controlador SDN es el encargado de ser el cerebro de la red, el cual se encuentra ubicado en los dispositivos del plano de datos y las aplicaciones de alto nivel. Tiene la responsabilidad de crear entradas de flujo en los dispositivos de conmutación a manera de establecer cada flujo en la red, dichas entradas tienen dos modos de operación dentro de los dispositivos del plano de datos como el modo proactivo en donde las reglas de flujo son enviadas a los dispositivos del plano de datos como los vSwitch siempre y cuando tenga conocimiento de ello; y el modo reactivo donde se toma acciones en base a la comprobación del flujo de control con las políticas de la capa de aplicación, siempre y cuando un dispositivo envié las solicitudes. [36]

En el modo reactivo el controlador recibe la configuración de flujos enviados por los dispositivos del plano de datos, además el controlador es el encargado de crear una ruta para que los paquetes la atraviesen siempre y cuando cumpla con las reglas de la capa de aplicación. [35]

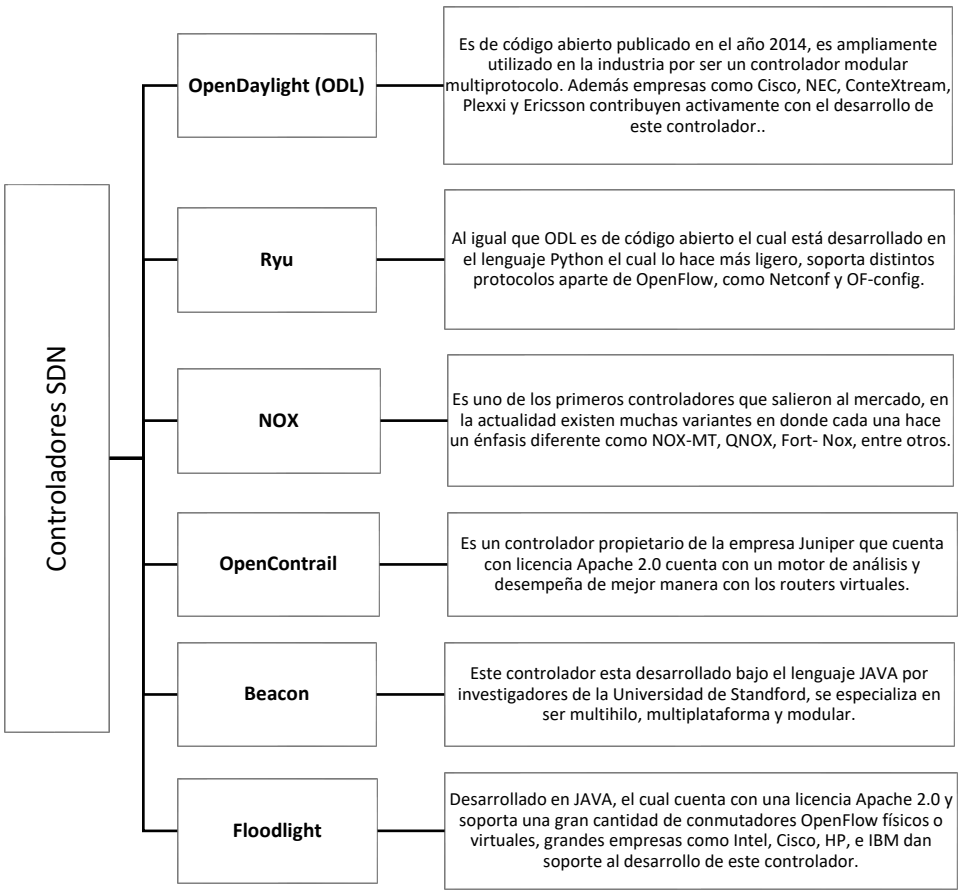


Figura 18: Controladores SDN que soportan el protocolo OpenFlow[35]

Tabla 11: Comparación entre los principales controladores SDN

	ZeroTier	**OpenDaylight**	**Ryu**	**FloodLight**
Interfaz grafica	Si	Si	Si	Web UI (Usando REST)
API REST	Si	Si	Si	Si
Código Abierto	Si	Si	Si	Si
Licencia	GNU General Public License V3 (GPLv3)	EPL v1.0	Apache 2.0	Apache 2.0
Controladores Distribuidos	Si	Si	No	No
Plataformas Soportadas	Windows, MacOS, Android, iOS, Linux, FreeBSD y NAS	Linux	Linux	Linux, mac y Windows
Lenguaje de programación	C++	Java	Python	Java
TLS Support	Si	Si	Si	Si
Dificultad de instalación y configuración	Baja	Media	Media	Media
Dificultad de escalabilidad de red	Baja	Alta	Alta	Alta

Elaborado por: El investigador

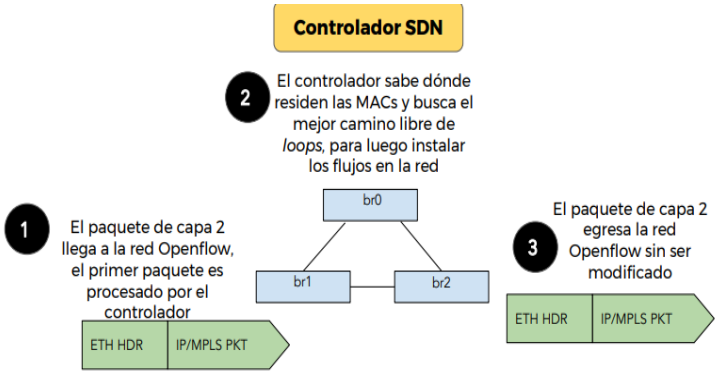


Figura 19: Esquema general de una red SDN con su respectivo controlador[36]

1.3.8. Controlador ZeroTier

El controlador diseñado por ZeroTier Inc. es un hipervisor de red el cual permite crear redes virtuales escalables, seguras y gestionables, además de que ofrece todas las funcionabilidades de las redes SDN. Este hipervisor utiliza una capa de virtualización de Ethernet similar a VXLAN e IPSec sobre una red encriptada punto a punto. El protocolo utilizado para establecer la conexión es de código abierto desarrollado por la misma empresa la cual consta de dos capas virtuales denominadas VL1 y VL2 cuyas funcionalidades se asemejan a las capas de transporte y de red del modelo OSI respectivamente.[38]

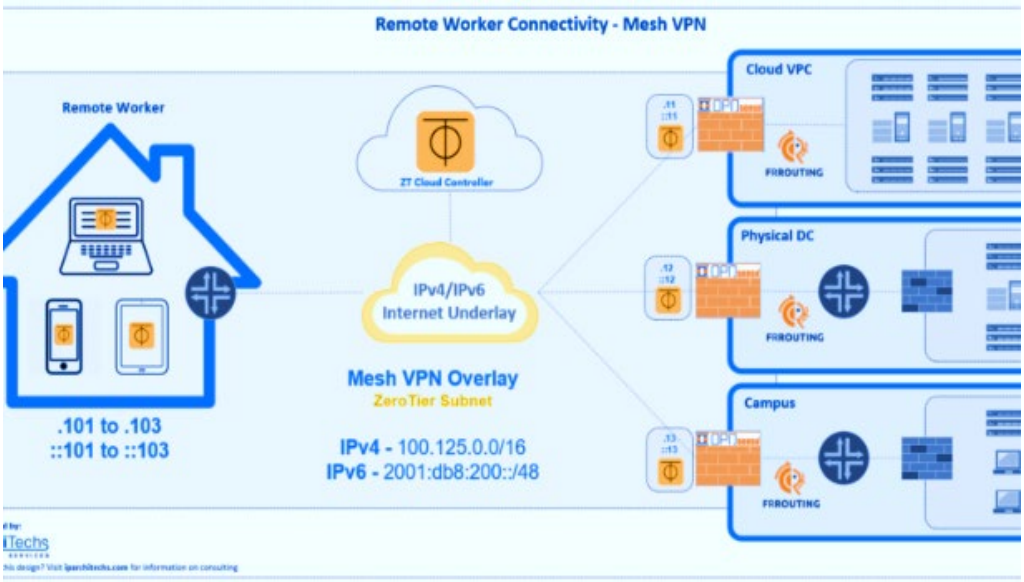


Figura 20: Aplicación del controlador ZeroTier[39]

A. Capa Virtual VL1: La red punto a punto ZeroTier

Es la encargada de establecer un "cable virtual" que posee cifrado, autenticación y demás parámetros de red para crear de manera dinámica conexiones punto a punto entre clientes ZeroTier, ya que no requiere configuración alguna, lo que conlleva que esta capa se estructure como el servicio de red DNS, que implica que en la base de la red exista una colección de servidores denominados "roots" los cuales ejecutan el mismo software de los puntos finales normales y se ubican demográficamente alrededor del mundo para garantizar una conexión estable y rápida de alrededor de 100ms de latencia.[40]

Los servidores root están organizados en 2 clústeres de 6 miembros cada uno los cuales están ubicados alrededor del mundo y son operados por la empresa, la cual lo distribuye como un servicio gratuito a sus clientes. Este grupo de servidores root se denomina Planeta, y puede tener una o más lunas, este último término es referido al servidor root instalado de manera local por el usuario final, esto con el propósito de tener un mejor rendimiento en la red, por ejemplo al instalar muchas lunas dentro de un edificio se puede mejorar la latencia y en caso de perder la conexión a internet, la red ZeroTier pueda seguir funcionando.[41]

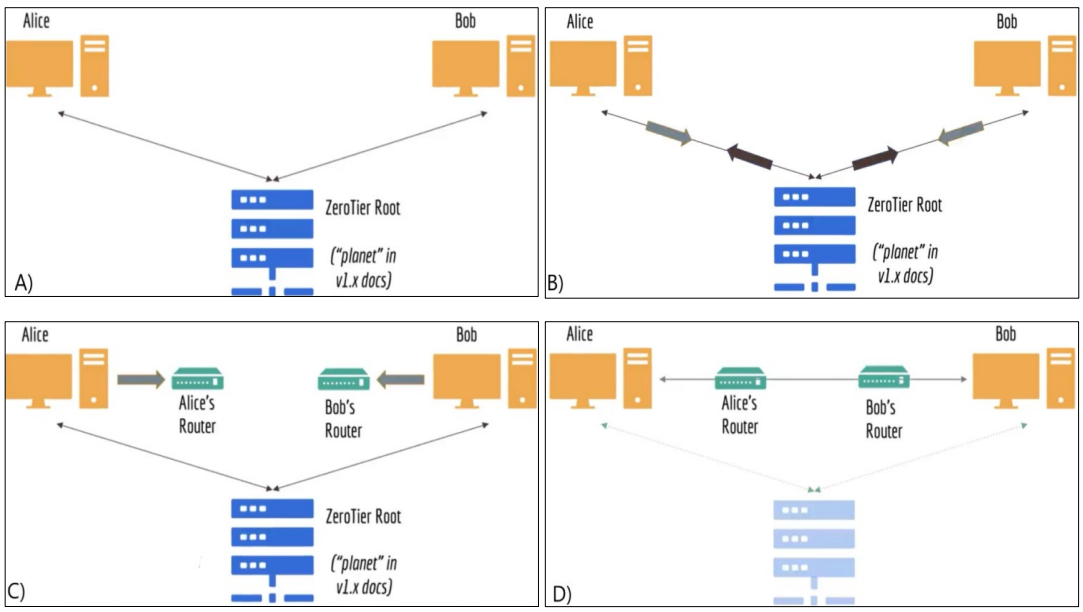


Figura 21: Proceso de conexión punto a punto en la capa VL1[42]

A) Alice y Bob no tienen comunicación entre ellos, más sin embargo tienen comunicación con el servidor root o planeta de ZeroTier.
B) El servidor root se percata que Alice y Bob se quieren comunicar, por lo que el servidor envía a estos dos clientes lo que sabe acerca de la ubicación de red (flecha oscura) de cada uno. Mientras que los clientes intercambian información sobre su ubicación de red, el tipo de NAT(Network Address Translation), etc con el servidor root (flecha gris).
C) Los clientes empiezan a utilizar distintas estrategias como UDP hole punching(utiliza puertos UDP disponibles en el router para crear un tunelamiento) para intentar la comunicación punto a punto entre ambos.
D) Una vez que el enlace directo entre los clientes se a establecido, estos se pueden comunicar y no requerirían al servidor root a no ser que se pierda el enlace, en ese caso el proceso se repite.[40]

La capa VL1 siempre priorizara la conexión directa entre clientes, para ello utilizara distintos procedimientos como es el descubrimiento de pares de LAN, la predicción de puertos para el cruce de NAT IPv4 simétricas y el mapeo explícito de puertos usando uPnP y/o NAT-PMP. En caso de no poder establecer la comunicación entre los clientes, ZeroTier utilizara el servidor root para la retransmisión de paquetes.[41]

Esta capa implementa un cifrado de clave pública asimétrica que se denomina Curve25519/Ed25519, una variante de curva elíptica de 256 bits, permitiendo asi que cada paquete tenga un cifrado punto a punto utilizando el método Salsa20 de 256 bits y autenticándose mediante el algoritmo Poly1305.[40]

B. Capa Virtual VL2: La capa de virtualización de Ethernet

Esta capa utiliza un protocolo similar a VXLAN pero se añadió funciones de administración de SDN, entre las funcionalidades de esta capa se tiene la implementación de limites VLAN seguros, multidifusión, reglas, seguridad basada en capacidades y control de acceso basado en certificados. Al estar construido sobre la capa VL1 y ser transportado por esta capa hereda el cifrado y la autentificación antes mencionado, además la capa VL1 permite que la capa VL2 pueda ser implementada sin problema alguno debido a que no es afectada por la topología de red física subyacente.[40]

Se debe tomar en cuenta que esta capa no tiene ninguna relación con la capa VL1, por lo que dos clientes conectados pueden tener múltiples redes, pero estas tendrán una misma ruta que es proporcionada por VL1 la cual asigna una dirección de 40 bits (10 dígitos hexadecimales) a cada cliente para determinar la ruta, mientras que la capa VL2 identifica a la red mediante un ID de 64 bits (16 dígitos hexadecimales), la cual contiene la dirección ZeroTier del controlador y el número de red en el controlador.[42]

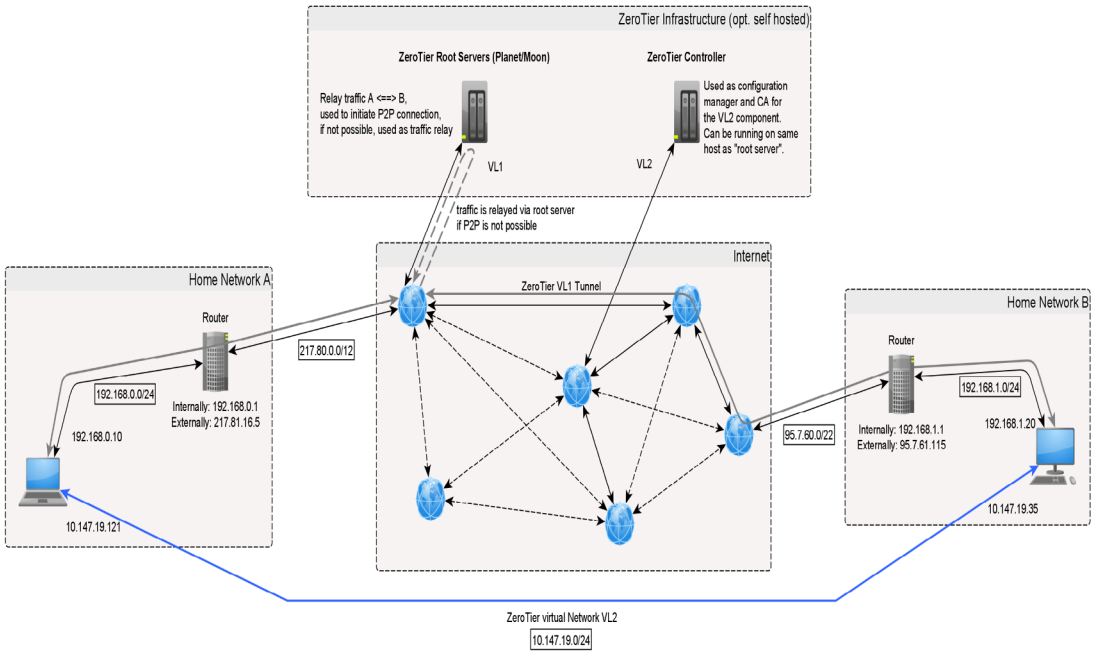


Figura 22: Arquitectura general de ZeroTier[42]

Cuando un cliente se une a la red de ZeroTier o hace una petición de actualización a esta red, el cliente envía una petición de configuración de red al controlador de red por medio de la capa VL1, para luego el controlador pueda utilizar la dirección asignada por VL1 al cliente para enviar le los certificados, credenciales y la información de configuración de red respectiva. Lo que se resume que las redes virtuales VL2 usan las direcciones de VL1 como puertos de este gran conmutador virtual a nivel global.[40]

1.3.9. OpenWrt

Es un sistema operativo basado en Linux y de código abierto lanzado en Enero del 2004, se instala principalmente en sistemas embebidos como los enrutadores de red que cuentan con un firmware propietario inestable o carece de características avanzadas, este firmware se compone de una imagen liviana que se puede ejecutar en dispositivos con 8 MB de memoria principal y almacenamiento no volátil de 2 MB, y además de paquetes personalizados que son utilizados para:[43]

- configuración de túneles SSH
- Configuraciones de red VPN
- Capturar y analizar el tráfico de la red
- Abrir o cerrar puertos
- Configuración del tráfico y calidad de servicio sobre los paquetes de la red
- Crear una red de invitados
- Habilitar servidores FTP o IRC
- Monitoreo en tiempo real de la red

Entre las principales alternativas a OpenWrt están DD-WRT y Tomato como se detalla en la Tabla 12, sin embargo no soportan la misma cantidad de dispositivos que OpenWrt, siendo este la principal desventaja, puesto que OpenWrt soporta más de 1000 tipos de enrutadores y cuenta con una gran comunidad para la producción y mantenimiento de paquetes instalables[44].

Tabla 12: Comparación entre firmwares de enrutadores de código abierto. [44]

	OpenWrt	DD-WRT	Tomato
Logo			
Licencia	GPLv2	GPLv2	GPLv2
Página Oficial	https://openwrt.org/	https://dd-wrt.com/	https://freshtomato.org/
Versión Actual	v21.02	V24	v2022.3
Interfaz Grafica	Si	Si	Si
Documentación	Alta	Alta	Baja
Comunidad de desarrolladores	Alta	Media	Baja
Memoria RAM requerido	Min: 64MB Recomendado: >128MB	Min: 4MB Recomendado: >4MB	Min: 16MB Recomendado: >64MB
Memoria Flash requerido	Min: 8MB Recomendado: >16MB	Min:4MB Recomendado: >8MB	Min: 4MB Recomendado: >8
Dificultad de instalar	Baja	Media	Alta

Elaborado por: El investigador

La instalación de este firmware requiere de ciertos parámetros técnicos como son:

- Memoria ROM
- Memoria RAM
- Velocidad de CPU

En la Tabla 13 se detallan los principales las características de los modelos de enrutadores que se pueden encontrar en el mercado:

Tabla 13: Principales modelos de routers que soportan OpenWrt disponibles en el mercado[45].

	My Net N750	**Dir-825**	**TL-WR940N**
Aspecto Físico			
Marca	Western Digital	D-Link	TP-Link
Versión del hardware	-	B1	v6
Versión OpenWrt actual soportado	v21.02.3	v21.02.3	V18.06.9
Tarjeta	ar71xx-ath79	ar71xx-ath79	ar71xx-ath79
CPU	Atheros AR9344	Atheros AR7161	Qualcomm Atheros TP9343
Núcleos	1	1	1
CPU MHz	560	680	750
Flash MB	16	8	4
RAM MB	128	64	32
Puertos ethernet 100M	0	0	5
Puertos ethernet Gbit	5	5	0
Switch	Atheros AR8327N	Realtek RTL8366SR	Qualcomm Atheros TP9343
WLAN 2.4GHz	b/g/n	b/g/n	b/g/n
WLAN 5.0GHz	a/n	a/n	-
Alimentación	12 Vdc, 2.0 A	12 Vdc, 2.0 A	9 Vdc, 0.6 A
Precio USD	$ 18.00	$ 47.00	$ 26.50

Elaborado por: El investigador

1.3.10. Docker

Proporciona el servicio de contenedores más utilizado, debido a que es una solución robusta, segura, rentable y fácil de implementar en cualquier tipo de ordenador y cualquier sistema operativo que este tenga. Este servicio es de código abierto el cual es mantenido por un gran número de empresas reconocidas a nivel mundial DigitalOcean, Microsoft entre otros.[46] Los componentes esenciales de Docker son:

Tabla 14: Elementos que componen la estructura de Docker[47]

Componentes	Definición
Engine	Aplicación instalada en la máquina final o "host" para construir, ejecutar u gestionar los contenedores, al ser el núcleo del sistema, este reúne todos los componentes de la plataforma en un solo lugar.
Daemon	Escucha y procesa la peticiones API para gestionar los distintos componentes de Docker como las imágenes, contendores, redes, volúmenes. Además puede comunicarse con otro demonios para levantar correctamente los servicios de Docker.
Client	Cuenta con una interfaz de comandos CLI, para comunicarse con Docker Daemon y así poder gestionar los elementos de Docker.
Image	Es un archivo de solo lectura que posee bibliotecas, dependencias, códigos de aplicación necesarios para crear los contenedores
Container	Es una instancia creada a partir de la imagen, que ejecuta un microservicio o una pila de aplicaciones completa.
Registry	Se basa en un sistema de catalogación para alojar, extraer y lanzar imágenes, ya sea de manera local o en la nube.

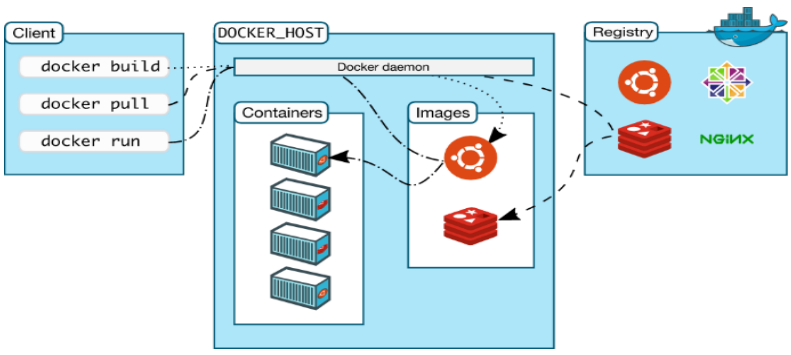


Figura 23: Arquitectura de Docker[46]

La arquitectura de Docker se basa en la arquitectura cliente-servidor, en donde el cliente habla con el servicio de "Docker Daemon" el cual hace todo el trabajo de construir y levantar los contenedores, cabe recalcar que el cliente y Docker Daemon pueden instalarse en un mismo servidor o el cliente se puede comunicar de manera remota con Docker Daemon haciendo uso de una interfaz de programación de aplicaciones de transferencia de estados representativos (API REST)[46].

A. Portainer IO

Es un contenedor que proporciona una interfaz gráfica UI, la cual permite gestionar de una mejor manera todos los componentes de Docker, sustituyendo así la interfaz CLI de Docker. Es de código abierto y está disponible para todos los sistemas operativos, de cualquier arquitectura, este contenedor se levanta por medio de comandos Docker y una vez ejecutado se puede acceder a la interfaz gráfica por medio de un navegador web y accediendo a la dirección IP del servidor seguido del puerto que por defecto es 9000.[48]

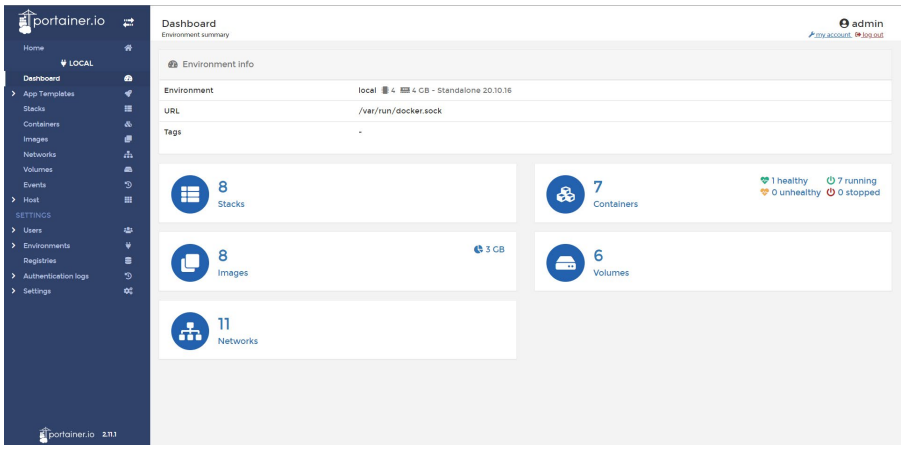


Figura 24: Interfaz web de Portainer.io[49]

Portainer esconde la complejidad de la gestión de contenedores detrás de una interfaz de usuario fácil de usar. Al eliminar la necesidad de utilizar la CLI, escribir YAML o entender los manifiestos, Portainer hace que el despliegue de aplicaciones y la resolución de problemas sea tan fácil que cualquiera puede hacerlo[49].

Actualmente brinda dos versiones de su programa, la primera es Portainer Community Edition la cual es una aplicación ligera que permite gestionar todos los recursos de Docker como contenedores, imágenes, volúmenes, redes, entre otros; por medio de una interfaz gráfica y/o una amplia API. La segunda es una versión pagada la cual se denomina Portainer Business Edition la cual incluye una serie de características y un conjunto de funciones avanzadas, así como soporte y son ideales para satisfaces necesidades empresariales[48].

B. MongoDB

Es una base de datos que permite el almacenamiento y acceso ordenado a los datos de manera no estructurada (NoSQL) por lo que además puede almacenar documentos como de distintos procesadores de texto como Word, Excel, PDF, almacenar emails, SMS, coordenadas geográficas, audio, video, entre otros. MongoDB utiliza documentos tipo JSON (JavaScript Object Notation) con esquemas. Esta base de datos puede ser instalado como un contenedor más en Docker, facilitando su uso.[50]

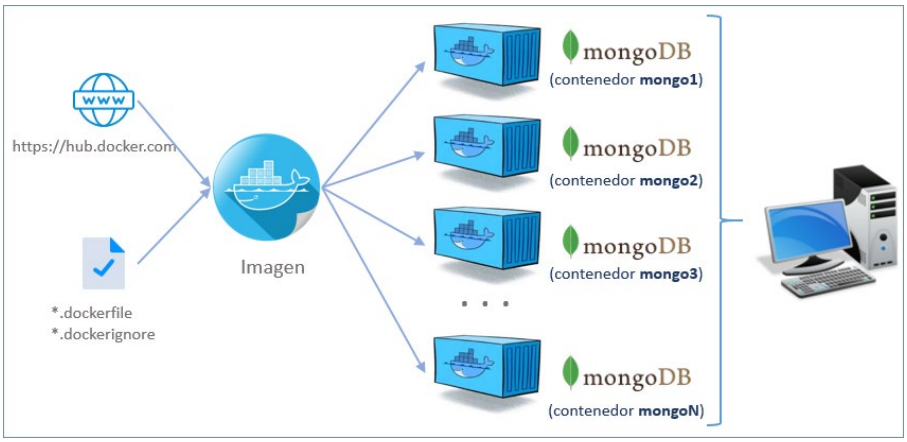


Figura 25: Creación de un clúster de base de datos con MongoDB y Docker[51]

Tabla 15: Componentes clave de la arquitectura de MongoDB[51]

_id	Es la identificación de un documento y por ende se crea de manera y se asigna de manera automática con un identificador único de 24 dígitos a cada documento dentro de un colección.
Collection	Grupo de documentos de MongoDB, dicha colección se encuentra dentro de una única base de datos
Cursor	Es un puntero por el cual los clientes pueden iterar para recuperar los resultados.
Database	La base de datos contiene las colecciones, un servidor MongoDB puede almacenar múltiples bases de datos.
Document	Es el registro dentro de una colección, el cual tendrá de un nombre de campo, con sus respectivos valores.
Field	Es un par que consiste en un nombre y un valor dentro de un documento.
JSON	Formato de texto plano, legible para el ser humano, el cual expresa datos estructurados.

En la Tabla 16 se muestra las especificaciones técnicas que mongo presenta frente a principales bases de datos utilizados en el IoT.

Tabla 16: Comparación entre distintas bases de datos utilizados en el IoT. [52]

	MongoDB	**Influxdb**	**Firebase**	**MySQL**
Licencia	GNU AGPLv3.0 & Comercial	Open Source	Comercial	GPLv2 & Comercial
Tipo de Base de datos	NoSQL, Document-oriented	Time series database	NoSQL, Real-time database	SQL
Escalabilidad	Horizontal	Vertical	Horizontal	Vertical, complejo

Plataformas Soportadas	Windows, Solaris, Linux, OS X	Linux, macOS	-	Linux, macOS, Windows, Solaris
Desarrollador	MongoDB Inc.	InfluxData Inc.	Google	MySQL AB
Lenguajes Soportados	Java, JavaScript, PHP, NodeJS, C, C#, Perl, Python, etc.	JavaScript, Python, Go, PHP, C#, C, Kotlin, etc.	Java, PHP, NodeJS, JavaScript, Switf, C++, etc.	PERL, C, C++, JAVA, PHP
API y otros métodos de acceso	Protocolo propio que utiliza JSON	InfluxDB 2.0 API	Android, iOS, JavaScript API, RESTful HTTP API	REST API
Seguridad	Alta	Media	Media	Baja
Documentación	Alta	Media	Alta	Media

Elaborado por: El investigador

C. EMQX

Es un bróker escalable MQTT que cuenta con un motor de alto rendimiento basado en Erlang para el procesamiento de mensajes en tiempo real con una baja latencia, este bróker está diseñado para el acceso masivo de clientes para lo cual realiza un enrutamiento de mensajes rápido entre los dispositivos físicos de red. [53]

Un solo bróker puede soportar 2 millones de conexiones de clientes MQTT, mediante miles de rutas, admite complementos personalizados para la autentificación de clientes, conexión con base de datos, API REST, motor de reglas SQL entre otros. Ademas ofrece soporte a los distintos protocolos IoT, como MQTT, MQTT-SN, CoAP, LwM2M, y otros protocolos propietarios basados en TCP/UDP. De igual manera se lo puede instalar en un contenedor haciendo fácil su uso. [53]

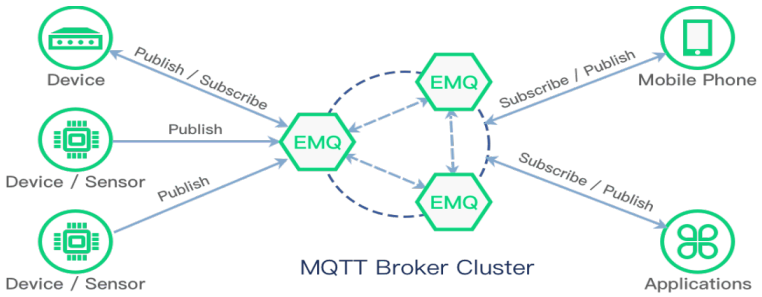


Figura 26: Arquitectura de comunicaciones de EMQX[53]

Tabla 17: Principales brókeres MQTT[54]

	EMQX	**Mosquitto**	**AWS IoT Core**
Licencia	Open Source Apache 2.0 Y Comercial	Open Source EPL/EDL	Comercial
Interfaz Grafica	Si	No	Si
Lenguaje en el que está programado	Erlang/OTP	C	C++
API REST	Si	No	Si
Modo Bridge	Si	Si	No
Clustering	Si	No	No
Motor de reglas	Si	No	Si
Suscripciones compartidas	Si	No	No
Protocolos Soportados	MQTT, MQTT-SN, CoAP, LwM2M, entre otros protocolos basados en TCP/UDP	MQTT	MQTT LoRaWAN HTTPS
Plataformas Soportadas	Linux	Linux, Mac, Windows	-
Documentación	Alta	Media	Media
Tipo de Broker	Self-hosted	Self-Hosted	Managed

Elaborado por: El investigador

D. Pi-hole

Es un servicio que se interpone entre el proveedor DNS del proveedor de internet con el dispositivo final, funge como sumidero DNS, de manera que intercepta todas las peticiones enviadas por el dispositivo final y es la aplicación de Pi-hole quien responde, mediante este funcionamiento se puede agregar listas de dominios permitidos o bloqueadas, por lo que se comporta como un filtro DNS para evitar anuncios, sitios web maliciosos, sitios web para adultos, entre otros. Igualmente Pi-hole funciona como un servidor DNS, el cual traduce un dominio y lo dirige a cualquier servidor que puede estar dentro de la red, o inclusive al servidor que está ejecutando esta aplicación, la cual puede ser ejecutada sobre cualquier ordenador de cualquier arquitectura, o en su defecto correr este servicio en un contenedor de Docker.[55]

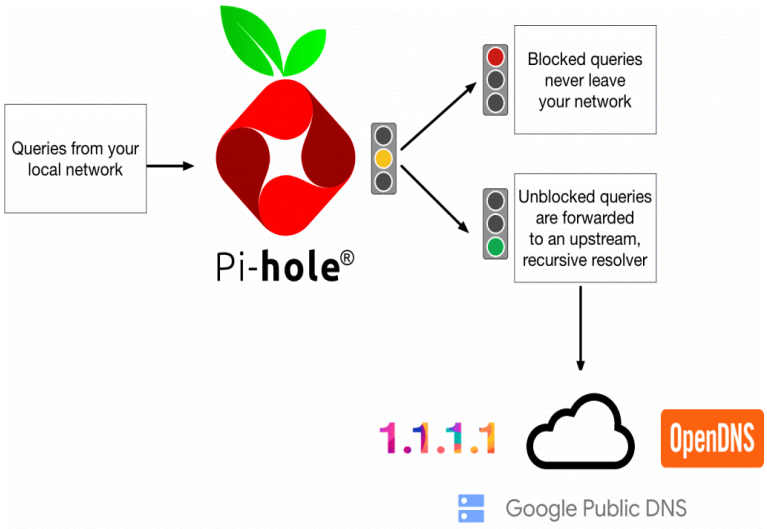


Figura 27: Esquema de funcionamiento de Pi-hole[55]

Tabla 18: Especificaciones de los servidores DNS disponibles para Linux. [56]

	Pi-hole	Adguard	DNSMasq
Licencia	European Union Public License (EUPL)	Comercial y de código abierto	GPLv3
Interfaz Gráfica	Si	Si	No
Dificultad en la configuración	Baja	Baja	alta
Filtro DNS	Si	Si	No
Comunidad de Soporte	Alta	Media	Baja
Documentación	Alta	Alta	Baja
Seguridad	Alta	Media	Alta

Elaborado por: El investigador

E. Node.js

Esta es un plataforma de software de código abierto la cual permite crear múltiples aplicaciones web tanto para la creación del frontend y el backend con gran escalabilidad, la cuales pueden ejecutarse del lado del servidor o del lado del cliente, para ello utilizan el lenguaje de programación de JavaScript, por lo que brinda un mejor rendimiento y eficiencia de las aplicaciones de tiempo real ya que node.js trabaja de manera asíncrona. Además cuenta con un motor JavaScript de Google V8 para la ejecución del código, el cual convierte el código JavaScript en código máquina. Gracias a su soporte HTTP y sockets esta plataforma puede actuar como un servidor web, lo que reduce recursos de terceros como es Apache. Esta plataforma se puede ejecutar en Mac OS X, Windows y Linux.[57]

Node.js trabaja bajo dos conceptos que son la asincronía es decir que trabaja en base a eventos, los procesa cada uno de forma independiente, este concepto se une al no bloqueo de entradas y salidas I/O, lo que implica trabajar con múltiples peticiones sin bloquearlas mientras esperan una respuesta.[58]

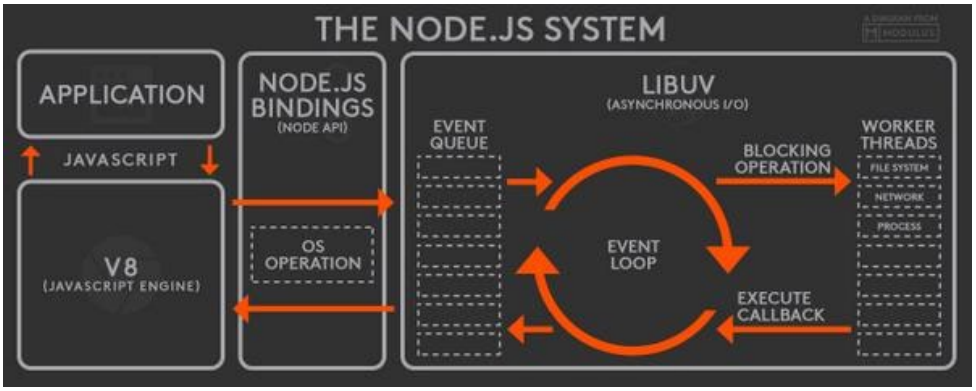


Figura 28: Diagrama de bloques [57]

1.3.11. Entorno de desarrollo web

A. Framework para el desarrollo del frontend

El framework de desarrollo del frontend es una herramienta que permite crear páginas web de manera eficiente y optimizando el tiempo; consta de un numeroso conjunto de módulos que permiten la incorporación de librerías en las aplicaciones web, entre las ventajas de usar frameworks para el desarrollo web se tiene:[59]

- Ofrece funcionalidades integradas.
- Se puede reutilizar el código.
- El desarrollo web se realiza de manera ordenada.
- Reducción en la cantidad de errores.

Actualmente existen diferentes tipos frameworks en función a las necesidades a cubrir de la aplicación web, en donde se destaca a Vue.js como la encargada de crear interfaces de usuario y aplicaciones de una sola página, en la Tabla 19 se detalla las principales características de Vue.js y su comparación con otros frameworks de desarrollo. [60]

Tabla 19: Comparación entre distintos frameworks para el desarrollo del frontend. [60]

	Vue.js	**React**	**Angular**
Página Oficial	https://vuejs.org/	https://react.org	https://angular.io
Fundadores	Desarrollador independiente	Mantenido por Facebook	Con ayuda de Google
Escrito en	JavaScript	JavaScript	TypeScript
Estabilidad	Buena	Buena	Buena
Documentación	Buena	Normal	Normal
Velocidad de codificación	Rápida	Normal	Baja
Ideal para	Adecuado para el desarrollo de webs ligeras y aplicaciones de una sola página	Perfecto para el desarrollo de webs moderno y aplicaciones nativas para iOS y Android	Aplicaciones a gran escala y con muchas funciones
Reutilización del código	Si, CSS y HTML	SI	No, solo CSS
Dificultad de uso	Baja	Alta	Alta

Elaborado por: El investigador

A. Lenguaje de desarrollo del backend

El backend se encarga de gestionar todos los datos solicitados por el frontend a través de diversas herramientas, siendo las interfaces de programación de aplicaciones (API) las más utilizadas. [61] El desarrollo del backend se puede realizar en múltiples lenguajes, sin embargo el lenguaje JavaScript se ha convertido popular entre los desarrolladores web gracias a sus características expuestas en la Tabla 20.

Tabla 20: Comparación entre los principales lenguajes de programación para el desarrollo del backend [62]

	JavaScipt	**Python**	**PHP**
Rendimiento y velocidad	Alto	Alto	Moderado
Funciona dentro de un navegador	Si	No	No
Comunidad y Soporte	Extensa	Extensa	Grande
Depuración	Rápida a moderada	Rápida	Baja
Seguridad	Alta	Alta	Moderada
Popularidad	Alta	Alta	Moderada

Elaborado por: El investigador

1.4.Objetivos

1.4.1. Objetivos General

Implementar un sistema de monitoreo y control de los parámetros operativos de los transmisores de radiodifusión FM aplicando una estructura IoT y redes SDN

1.4.2. Objetivos Específicos

- Identificar sensores y actuadores que mejor se adapten para el monitoreo de los parámetros operativos de los transmisores FM.
- Determinar el protocolo IoT y el controlador SDN para la intercomunicación de los dispositivos del sistema con los usuarios finales.
- Diseñar una interfaz web para el monitoreo y control de los transmisores FM de manera remota.

CAPÍTULO II

METODOLOGÍA

2.1.Materiales

La implementación del sistema de monitoreo y control de los parámetros operativos de los transmisores FM requirió de distintos herramientas y materiales tanto de hardware como de software, en la parte de hardware se desarrolló una placa electrónica para la fijación de elementos como ESP32, modulo Pzem-004t 3.0 Ttl Modbus 100a, amplificadores operacionales, resistencias, capacitores y demás elementos; para el desarrollo de software se utilizó Visual Studio como editor de texto, Docker para la virtualización de los servicios de EMQX, Mongodb y Node.js, ZeroTier para la creación de redes SDN, y Raspbian como sistema operativo para el servidor. Además manuales de los equipos transmisores FM analógicos.

2.2.Métodos

2.2.1. Modalidad de Investigación

En el presente trabajo de investigación se aplicó la investigación bibliográfica fundamentando los conceptos teóricos necesarios sobre la arquitectura IoT, redes SDN y tecnologías inalámbricas para el diseño del sistema de monitoreo y control de los parámetros operativos de los radiotransmisores FM basado, en fuentes como artículos científicos, libros o revistas académicas.

Se realizó una investigación aplicada empleando todos los conocimientos adquiridos en el transcurso de la investigación bibliográfica, con la finalidad de desarrollar un sistema de telemetría para los transmisores de radiofrecuencia RF, el cual se basó en software libre, empezando desde su arquitectura de red hasta sus dispositivos finales.

Una vez realizado el diseño del sistema se aplicó la investigación experimental realizándose en el cerro Pilishurco lugar donde se encuentran ubicados los transmisores de radiodifusión FM a cargo de la empresa Ecuatronix Cía. Ltda. permitiendo recolectar información relacionados a las magnitudes de los parámetros técnicos a manera de proponer un proyecto de investigación que cumpla con los objetivos propuestos.

2.2.2. Recolección de Información

Para la recolección de información se tomó como fuentes de información revistas indexadas, tesis artículos y libros publicados en los últimos años con relación a la arquitectura del internet de las cosas IoT y redes definidas por software SDN, obteniendo así una mayor nivel de confianza en la investigación.

2.2.3. Procesamiento y análisis de datos

Para el procesamiento de datos se tomó en cuenta la información obtenida en diferentes medios documentados, llegando a realizar un análisis crítico en los siguientes puntos:

- Estudio de los parámetros técnicos de los transmisores de radiofrecuencia FM comerciales.
- Protocolo IoT que mejor se adapte a las necesidades del sistema de telemetría.
- Obtención de un esquema de red definida por software.
- Interpretación de datos en base a la solvencia del problema.
- Presentación de resultados con base al fundamento teórico.

2.2.4. Desarrollo del proyecto

Para desarrollo del proyecto se realizaron las siguientes actividades, las mismas que permitieron la correcta implementación del sistema de telemetría aplicado a los transmisores FM:

- Identificación de los parámetros operativos de los equipos de radio transmisión RF analógicos.
- Selección de los componentes como sensores, actuadores, microcontroladores del sistema de telemetría.
- Análisis de los protocolos IoT y controladores SDN que mejor se adapten a los requerimientos del sistema.
- Implementación de una arquitectura IoT que gestione los parámetros operativos del sistema sobre una red SDN.
- Selección del lenguaje de programación y las librerías a ocupar en la plataforma web.
- Creación del frontend y backend de la plataforma web.
- Realización de pruebas de funcionamiento del sistema.
- Corrección de errores del sistema.
- Elaboración de informe final del proyecto.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1.Análisis y discusión de los resultados

La implementación de un sistema de monitoreo y control de los parámetros operativos de los transmisores de radiodifusión FM aplicando una estructura IoT y redes SDN permite una gestión remota sobre el transmisor sin tener la necesidad de acudir al sitio ahorrando tiempo y dinero ante fallas en el quipo. Se ha implementado distintos sensores como medidor de voltaje y corriente AC, medidores de voltaje DC y una cámara al dispositivo IoT, de igual manera se implementó actuadores como interruptores AC mecánicos y un motor a pasos, estos se comunican con un microcontrolador, que por medio de la tecnología de comunicación WiFi 802.11n, envía los datos bajo el protocolo MQTT hacia el enrutador configurado con una dirección IP de la red SDN para retransmitir esos datos hacia el Broker, que se encuentra en la misma red SDN pero geográficamente ubicado en un distinto sitio, esto permite que el sistema de comunicaciones diseñado sea de bajo costo, debido a que no se requirió el uso de un VPS, o cualquier servicio de la nube, además es una red segura debido a que solo usuarios autorizados pueden unirse a la red SDN y las comunicaciones entre estas son encriptadas; esta red resulto ser muy escalable y flexible puesto que el controlador SDN se pudo instalar en diferentes plataformas como Android, Linux, OpenWrt, iOS y Windows.

3.2.Desarrollo de la propuesta

En este proyecto se desarrolló un sistema de comunicación entre el personal técnico que supervisa los transmisores FM y los sensores y actuadores de estos, mediante una aplicación web, que posee diferentes vistas o páginas para los distintos propósitos del sistema, como es la visualización de los sensores y/o control de los actuadores, creación de reglas, dispositivos, plantillas de trabajo. El sistema cuenta con un chatbot de whatsapp que permite notificar las alarmas generadas por el transmisor a distintos usuarios.

En la Figura 29 se muestra el sistema de comunicaciones implementado, en donde los parámetros operativos de los transmisores consisten en:

- Tensión de alimentación AC [V_{AC}].
- Corriente de alimentacion AC [I_{AC}].
- Potencia activa eléctrica consumida [W].
- Factor de potencia del equipo [%]
- Potencia transmitida [W].
- Potencia reflejada [W].

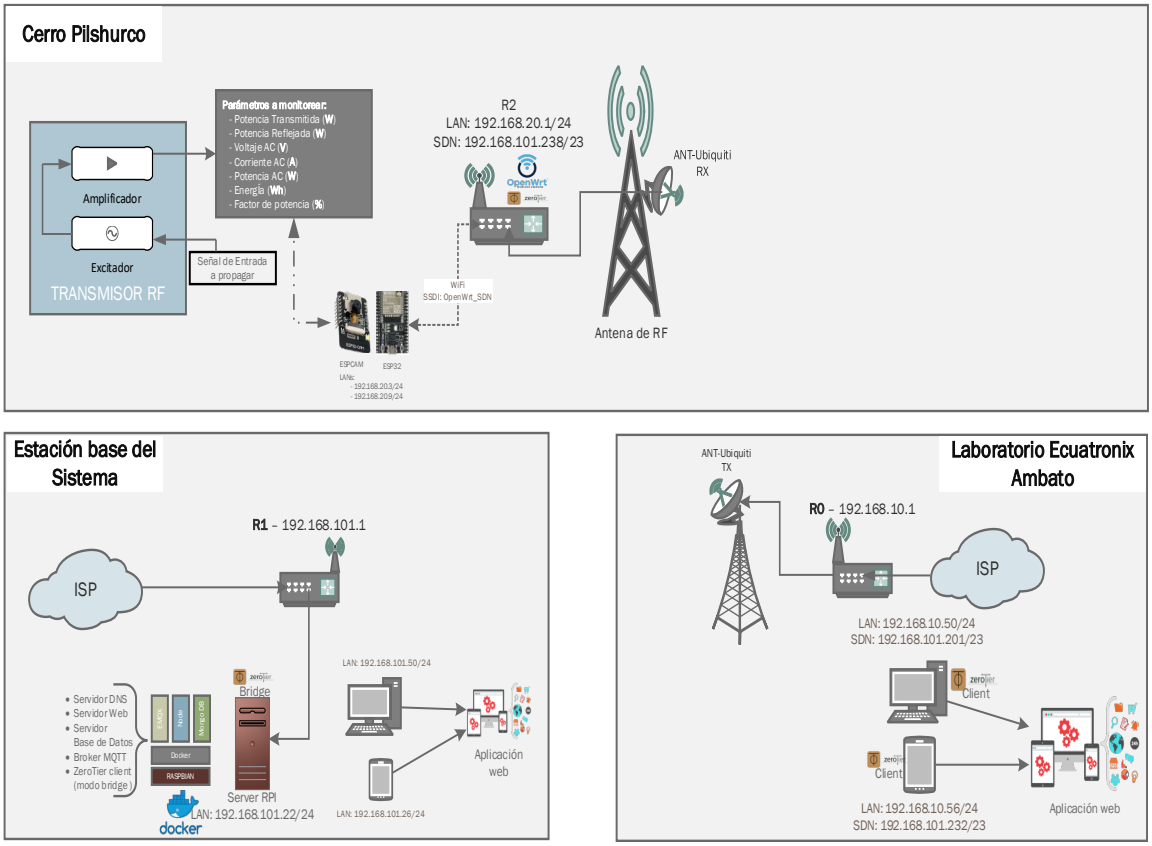


Figura 29: Sistema de telemetría implementado

Elaborado por: El investigador

3.2.1. Controladores, sensores y actuadores del sistema

En la selección del controlador para el dispositivo, se debe tener en cuenta la capacidad de almacenamiento para los sketches, que cuente con tecnologías de conectividad inalámbrica como WiFi y que sea eficiente en el manejo de energía, es por ello que para el manejo de los sensores se optó por utilizar la tarjeta de desarrollo ESP32-DevKit, por otro lado para la adquisición de imágenes se eligió la ESPCAM debido a sus características técnicas como se muestra en la Tabla 8.

Para la obtención de los parámetros operativos de los transmisores RF como son voltaje AC, corriente AC, potencia activa AC, su factor potencia FP y el consumo energético se utilizó el módulo PZEM-004T gracias a sus características expuestas en la Tabla 9, por otro lado dado a que el transmisor FM modelo TEX100 posee sensores internos los cuales se pueden acceder por medio de los terminales de telemetría, se requiere un convertidor ADC para interpretar las señales analógicas y por medio de un modelo matemático adecuar dicha información, es por ello que en base a la Tabla 10 se utilizó el sensor ADS1115, ya que ofrece 4 canales ADC, y la comunicación es por I2C, permitiendo que el microcontrolador no se sobrecargue y liberando así recursos del mismo.

En lo que respecta actuadores se necesita de relés de 5V capaces de interrumpir cargas de hasta 10 Amperios, en caso de sobrepasar la carga se recomienda utilizar contactores, por otro lado el transmisor regula su potencia por medio de un potenciómetro de alta precisión, por lo que se debe controlarlo haciendo uso de un motor a pasos, ya que este tiene la capacidad de realizar giros de manera precisa.

3.2.2. Selección del software y hardware para las redes SDN

Controlador SDN

El controlador para la red SDN para la comunicación para este proyecto es ZeroTier, gracias a su alta escalabilidad, seguridad, y demás otras funcionalidades, lo hace idóneas para la creación de redes de datos, en la Tabla 11 se compara a ZeroTier con los principales controladores SDN, además este controlador cuenta con múltiples usos como:

- Alternativa a la VPN
- LAN
- SD-WAN
- Gestión de MSP
- Infraestructura como código y DevOps
- Túnel de software
- IDS/Supervisión de redes
- SDK (IoT, redes integradas)

Router Inalámbrico

El router cumple la función de vincular los dispositivos IoT dentro de la red SDN, es por ello debe tener compatibilidad con ZeroTier, en la actualidad solo los productos de red de la empresa MikroTik ofrece en su firmware propietario la posibilidad de instalar este software, como un paquete más, otros firmwares propietarios como TP-Link, Netgear, Cisco, entre otros no disponen esta opción, es por ello y en base a lo expuesto en la Tabla 12 que se recurre a cambiar el firmware de estos dispositivos y utilizar el firmware "OpenWrt".

Un vez seleccionado el firmware idóneo para el proyecto, se procede a seleccionar el hardware o el router físico, para ello es necesario recurrir a la tabla de dispositivos aceptados que ofrece OpenWrt en su página oficial, el cual contempla distintas marcas y modelos de routers comerciales, cabe recalcar que este firmware no solo se limita a estos dispositivos, ya que se los puede instalar en extensores Wifi, micro ordenadores como la Raspberry pi o en máquinas virtuales.

En base a la Tabla 13, para la elaboración de este proyecto se ha utilizado el router My Net N750 de la marca Western Digital, debido a las características técnicas que posee, a su precio y por ser compatible con la última versión de OpenWrt.

3.2.3. Selección de los servicios web

Para la elaboración de la aplicación web requiere de distintos servicios como una base de datos en donde se guardará toda la información referente a los usuarios, los dispositivos, las plantillas, las reglas para las alarmas, imágenes tomadas, entre otros tipos de información; asimismo se necesitó un bróker MQTT el cual retransmitió todos los valores de los sensores hacia la aplicación web, además de crear reglas para las distintas alarmas y por último un servidor de dominio para que la aplicación web sea accesible de una mejor manera, por medio de un dominio, en lugar de la IP del servidor.

En base a lo expuesto en la Tabla 17 y la Tabla 18 se ha elegido al bróker EMQX ya que no solo soporta el protocolo MQTT sino que además soporta otros protocolos como CoAP por lo que es altamente escalable, por otro lado se utilizó el servicio de Pi-hole como un servidor DNS que apuntara a la aplicación web, mediante el dominio http://telemetriaex.com hacia la IP del servidor que en este caso es 192.168.101.22.

3.2.4. Configuración de la red SDN

Se necesitó crear una red SDN, para ello se creó una cuenta en la página oficial ZeroTier https://my.zerotier.com/ mediante un correo electrónico, luego se procedió a crear la red SDN con un solo click en el recuadro tomate, después de la creación de la red y acceder a esta, se tiene una interfaz gráfica para la configuración de todos los parámetros de la red como el id de la red de ZeroTier, el nombre, el tipo de acceso(pública o privada), la dirección IP, el enrutamiento, una tabla de usuarios conectados para poder gestionar su autorización a la red, y por ultimo un apartado para la configuración de reglas y filtros.

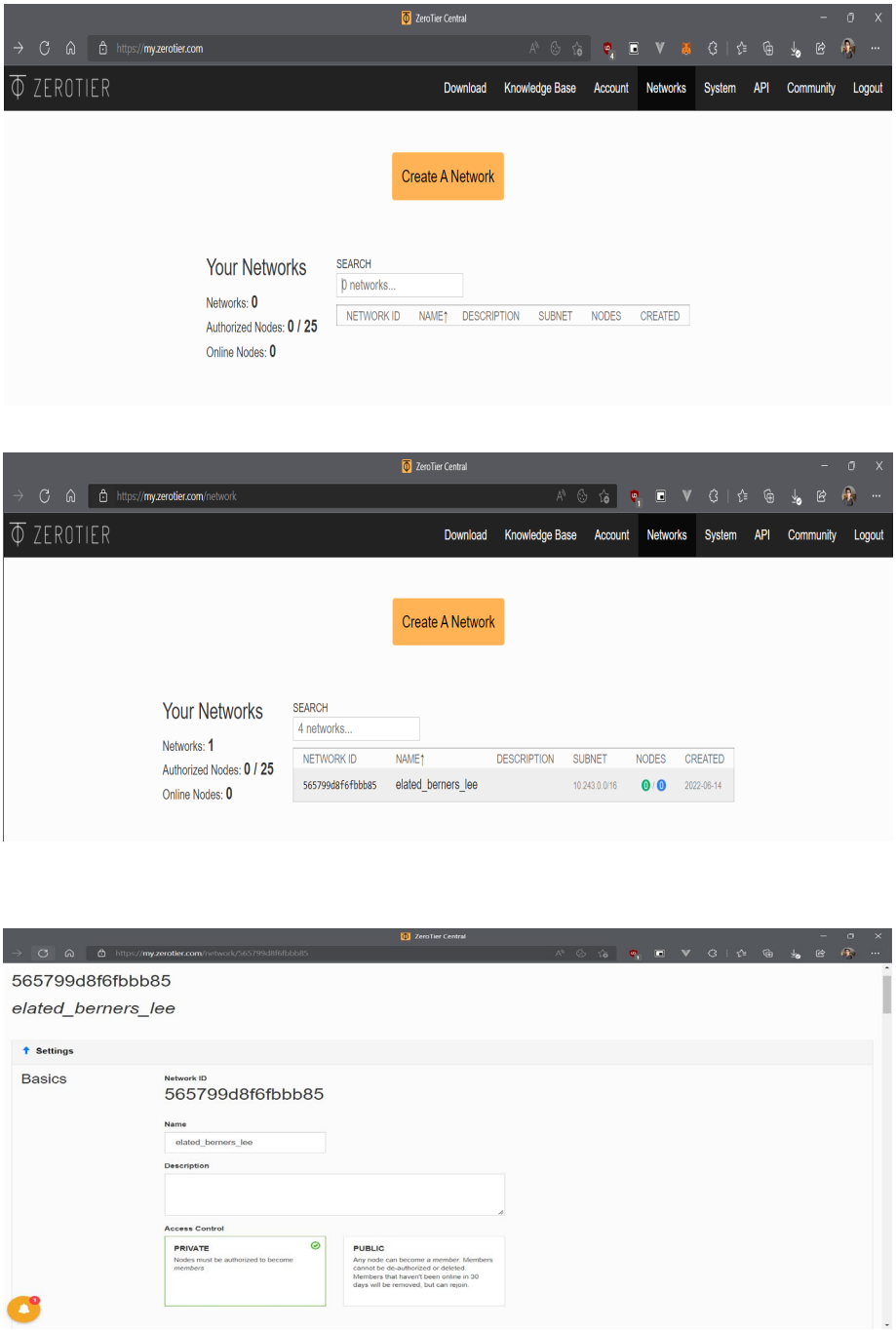


Figura 30: Creación de una red SDN en ZeroTier

Elaborado por: El investigador

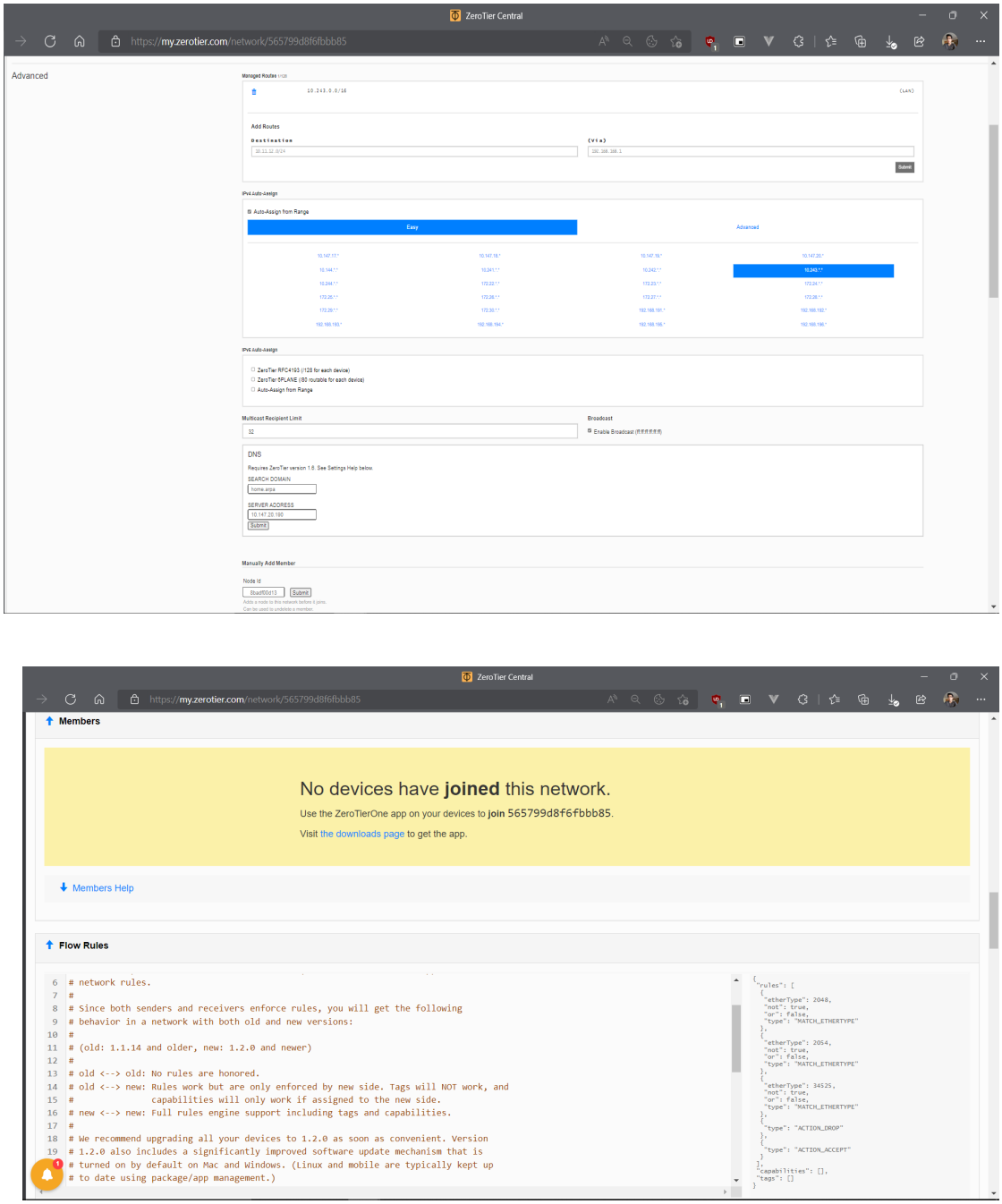


Figura 31: Interfaz web de Zerotier para la administración de la red SDN

Elaborado por: El investigador

Una vez creada la red, se procedió a configurar los dispositivos finales, ZeroTier ofrece una amplia gama de plataformas soportadas para poder instalar el controlador en distintos dispositivos móviles como celulares o tabletas, y en ordenadores como computadoras, laptops e inclusive en mini ordenadores como es el caso de la Raspberry Pi. Al ser este último, el servidor que aloja la página web tiene como sistema operativo Raspbian x64

LITE, la misma que es una versión de Linux, por ello en la terminal se deberá ejecutar el siguiente comando para instalar el controlador:

```
curl -s https://install.zerotier.com | sudo bash
```

Y para unirse a una red SDN se debe hacerlo mediante el siguiente comando:

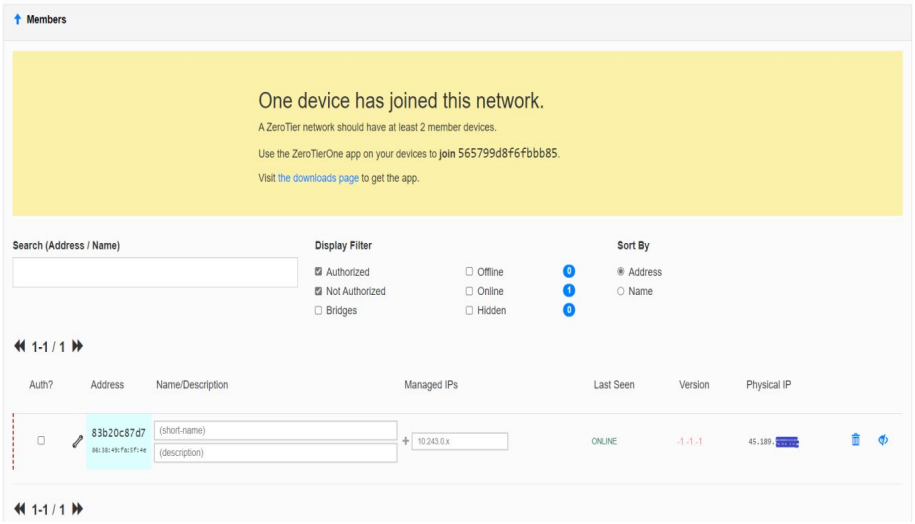
```
sudo zerotier-cli join $NETWORK_ID
```

```
pi@raspberrypi:~ $ sudo zerotier-cli join 565799d8f6fbbb85
200 join OK
```

Figura 32: Respuesta a la solicitud de unirse a la red

Elaborado por: El investigador

Finalmente se accedió a la interfaz web para autorizar al dispositivo, para lo cual se marcó la casilla de autorización para cada uno de los dispositivos, una vez autorizado en la Raspberry se creó una interfaz de red virtual, en donde se muestra la dirección IP de la red que se unió, mascara, entre otros parámetros de red.




```
pi@raspberrypi:~ $ ip add | grep ztr2qwey3t
75: ztr2qwey3t: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2800 qdisc pfifo_fast state
 UNKNOWN group default qlen 1000
    inet 10.243.158.2/16 brd 10.243.255.255 scope global ztr2qwey3t
```

Figura 33: Autorización del nuevo dispositivo que se incorpora a la red y la verificación de esta red por parte del cliente

Elaborado por: El investigador

Para ordenares cuyo sistema operativo es Windows se descargó el instalador "ZeroTier One.msi" desde la página oficial, cuyo asistente de instalación es intuitivo y solo requirió en dar click en siguiente y finalizar, una vez instalado se creó un ícono con el logo de la empresa.

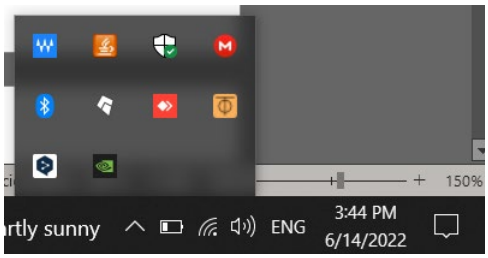


Figura 34: ZeroTier como cliente instalado en Windows

Elaborado por: El investigador

Para unirse a la red se necesitó hacer click derecho en el logo de la empresa, para luego seleccionar unirse a la red, luego de ello aparecerá una ventana emergente para ingresar el ID de la red ZeroTier creada, luego se necesitó autorizar a este nuevo dispositivo desde la interfaz web de ZeroTier, para poder verificar si el ordenador se ha unido a la red correctamente se debió ir al ícono del logo de la empresa de nuevo, y dar click derecho, en donde se visualizó el ID de la red en esta apartado junto al nombre de la red, además al ubicar el cursor en este campo se despliega un apartado para las configuraciones de la red y en el apartado de "Gestión de Direcciones" se puede observar la dirección IP del ordenador que ZeroTier le proporcionó mediante DHCP, tal y como se muestra en la Figura 36. Al trabajar con un servidor DNS como Pi-hole, se debió marcar la casilla "Permitir la configuración DNS".

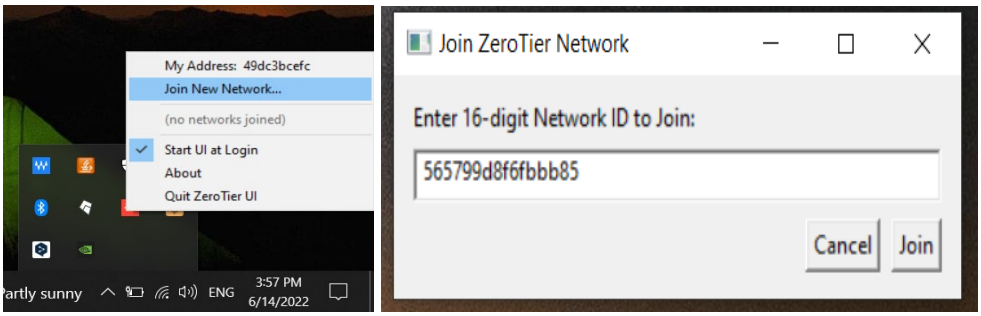


Figura 35: Proceso para unirse a la red de ZeroTier en Windows 10

Elaborado por: El investigador

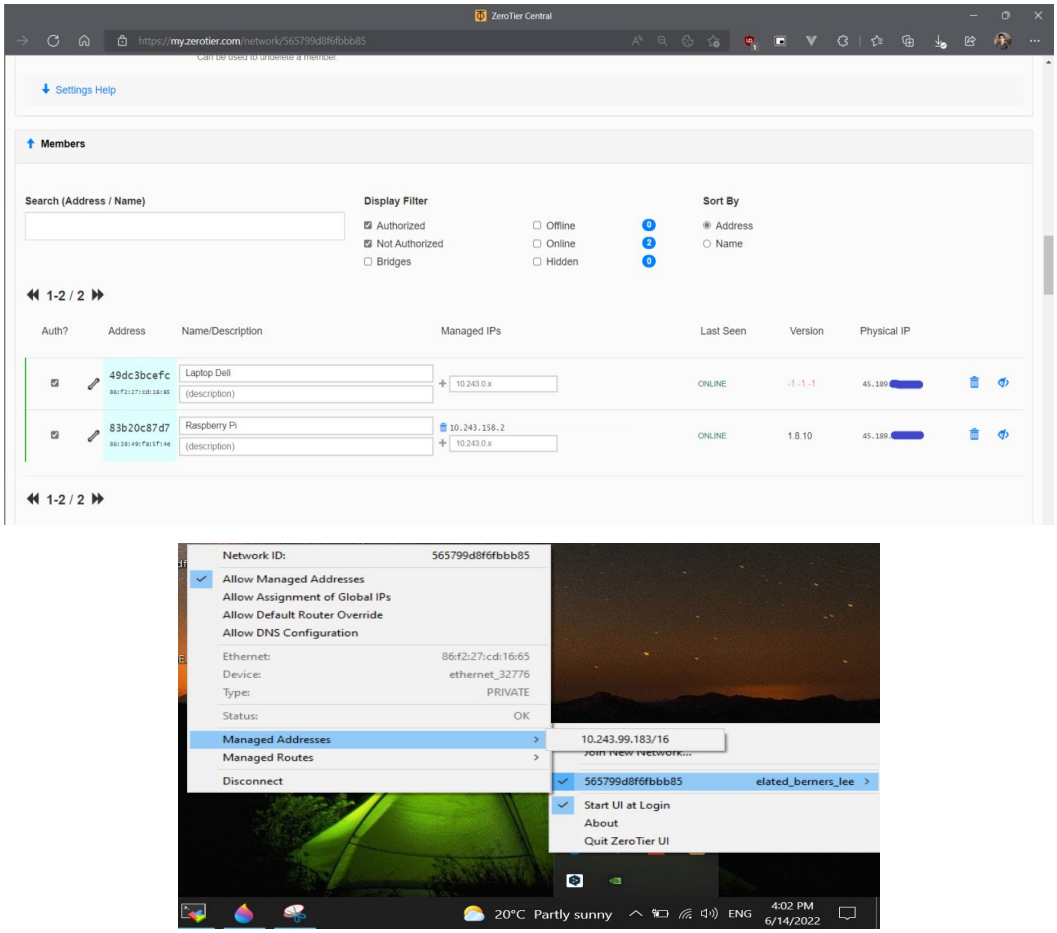


Figura 36: Autorización del cliente en el panel de control de ZeroTier y su comprobación.

Elaborado por: El investigador

Por otro lado para la configuración del dispositivo móvil, se debió descargar la aplicación de ZeroTier One, la misma que se encuentra en la Play Store para dispositivos Android o en la App Store para dispositivos iOS, una vez instalada la aplicación se creará un ícono con el logo de la empresa, al ingresar a la aplicación se puede observar dos botones en la parte superior con forma de un signo “mas” la cual permite añadir redes ZeroTier y otra con una llave es la configuración de la aplicación, para otorgarle permisos para que funcione con datos móviles, o desactivar el protocolo IPv6, en este caso se debe marcar las casillas.

Para unirse a la red se debe pulsar en el símbolo “más” la cual despliega otra interfaz en donde se puede agregar el ID de la red y en este caso se seleccionó la opción “Network DNS”, la cual permite que Pi-hole funcione a través de esta red, al unirse se debe deslizar el interruptor para activarla, como es por primera vez, el dispositivo pedirá una confirmación sobre el uso de la aplicación, tal y como se muestra en la Figura 37.

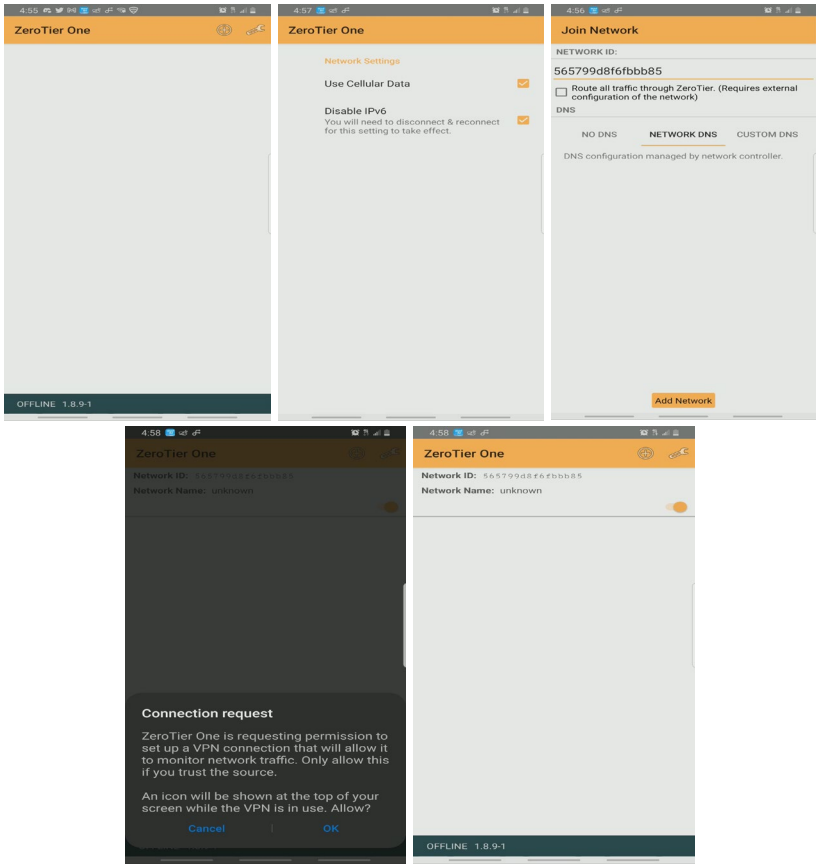


Figura 37: Proceso para unirse a la red de ZeroTier en dispositivos móviles
Elaborado por: El investigador

Luego de haber agregado el dispositivo móvil a la red, se debe acceder a la interfaz web de ZeroTier para autorizar el dispositivo marcando la casilla que se encuentra a lado del dispositivo agregado, en la Figura 38 se muestra que una vez autorizado en el panel de control de la empresa, la aplicación móvil ya cuenta con una dirección IP de la red creada, nótese que el dispositivo móvil está funcionando con la red celular y es por ello que en el panel de control la columna de IP física empieza con 200, al realizar las pruebas de conectividad desde el servidor una Raspberry Pi 4 hacia el celular, se puede comprobar que entre ambos puntos existe una conectividad, como se muestra en la Figura 39.

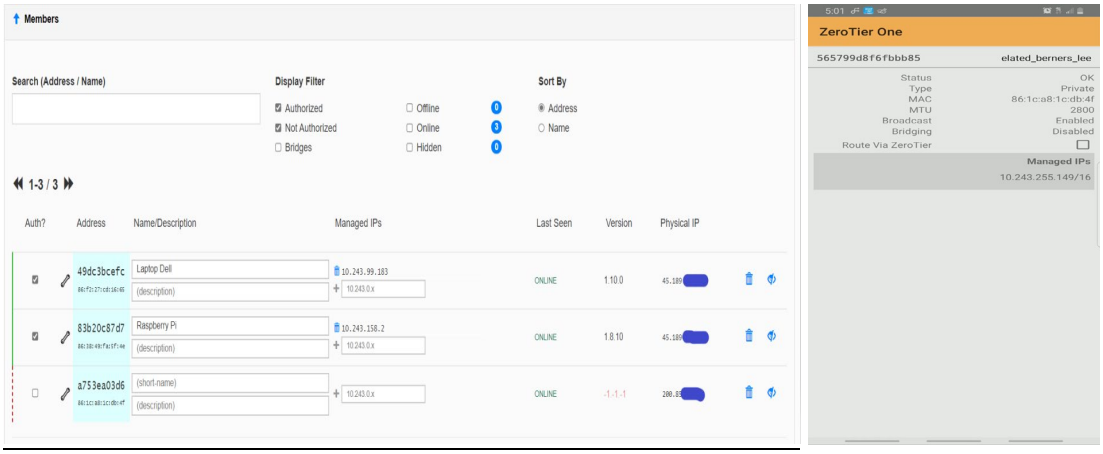


Figura 38: Autorización en el panel de control de ZeroTier del dispositivo móvil y la verificación de asignación de dirección IP

Elaborado por: El investigador

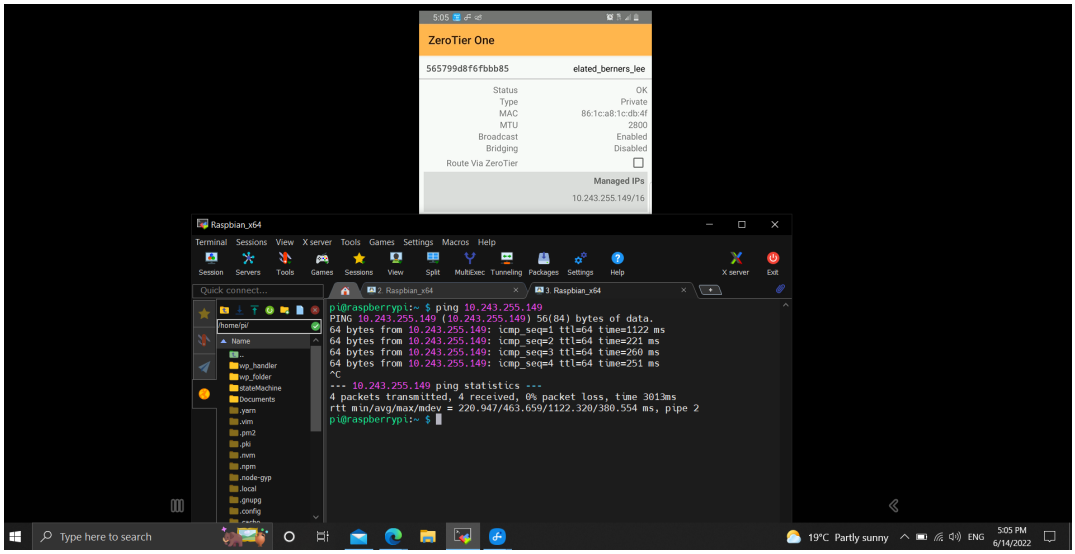


Figura 39: Comprobación de conectividad entre el servidor que se encuentra de manera local y el dispositivo móvil que hace uso de la red celular.

Elaborado por: El investigador

Por último, se debe realizar y configurar la red SDN dentro de un enrutador que en base a la Tabla 13 se eligió el enrutador de la marca Western Digital cuyo modelo es My Net N750 el cual se procedió a cambiar del firmware propietario de la empresa al firmware libre "OpenWrt", la misma que puede ser descargada en la página oficial y digitando el modelo del enrutador como se muestra en la Figura 40, se procedió a descargar 2 archivos de extensión ".bin", el primero es para la instalación de este firmware, y el segundo es

para la actualización de este. Cabe recalcar que debido a inconvenientes con la versión actual, se optó por utilizar la versión v19.07.8.

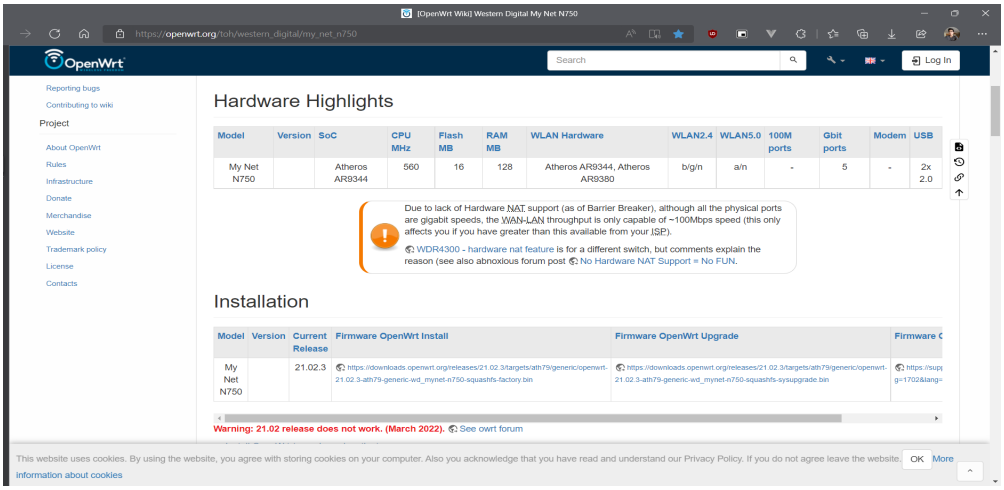


Figura 40: Página web para la descarga del firmware de OpenWrt v21.02.3

Elaborado por: El investigador

Una vez con los archivos se procedió a conectar el enrutador como se muestra en el esquema de la Figura 41, y por medio del ordenador acceder a la dirección 192.168.1.1 en donde se muestra una interfaz web de configuración. En esta interfaz se accedió a las configuraciones avanzadas para la actualización de firmware, se carga el primer archivo y luego al dar click en cargar se presenta una interfaz en donde se debe esperar un tiempo de alrededor de 5 minutos para que el nuevo firmware sea cargado correctamente en el enrutador como se muestra en la Figura 42.

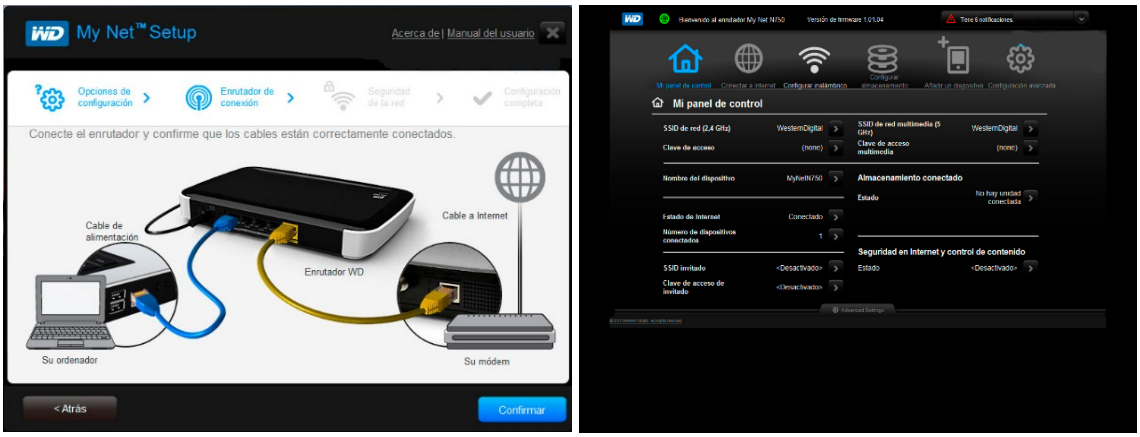


Figura 41: Configuración del enrutador en la red, y la interfaz web

Elaborado por: El investigador

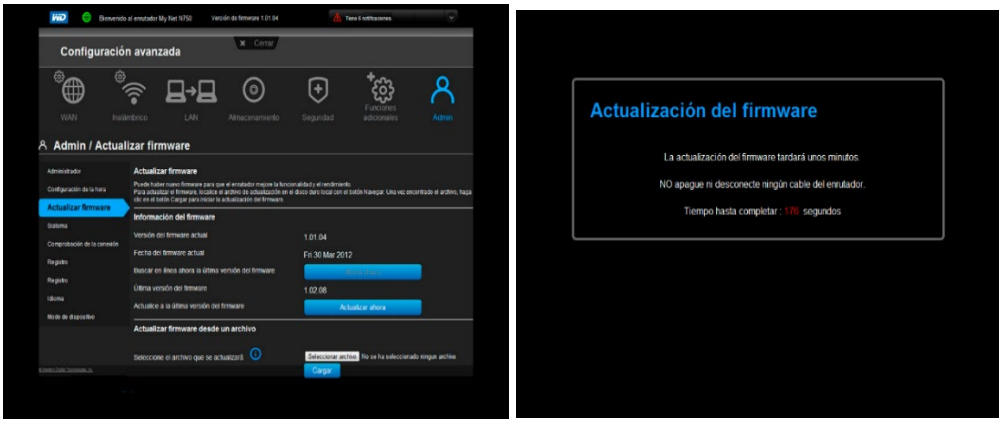


Figura 42: Pasos para instalar el firmware de OpenWrt

Elaborado por: El investigador

Luego de la instalación, se accede a la dirección 192.168.1.1, en donde se muestra la nueva interfaz web del enrutador, la misma que es proporcionado por OpenWrt, en donde se debe configurar una contraseña y por defecto el nombre de usuario es root. Al ingresar se procede a actualizar el firmware que se lo realiza en el apartado de "Software" y en la opción de "Back up/Flash Firmware" al deslizar hasta la opción "Flash new firmware image" se tiene que seleccionar el archivo de actualización de OpenWrt.

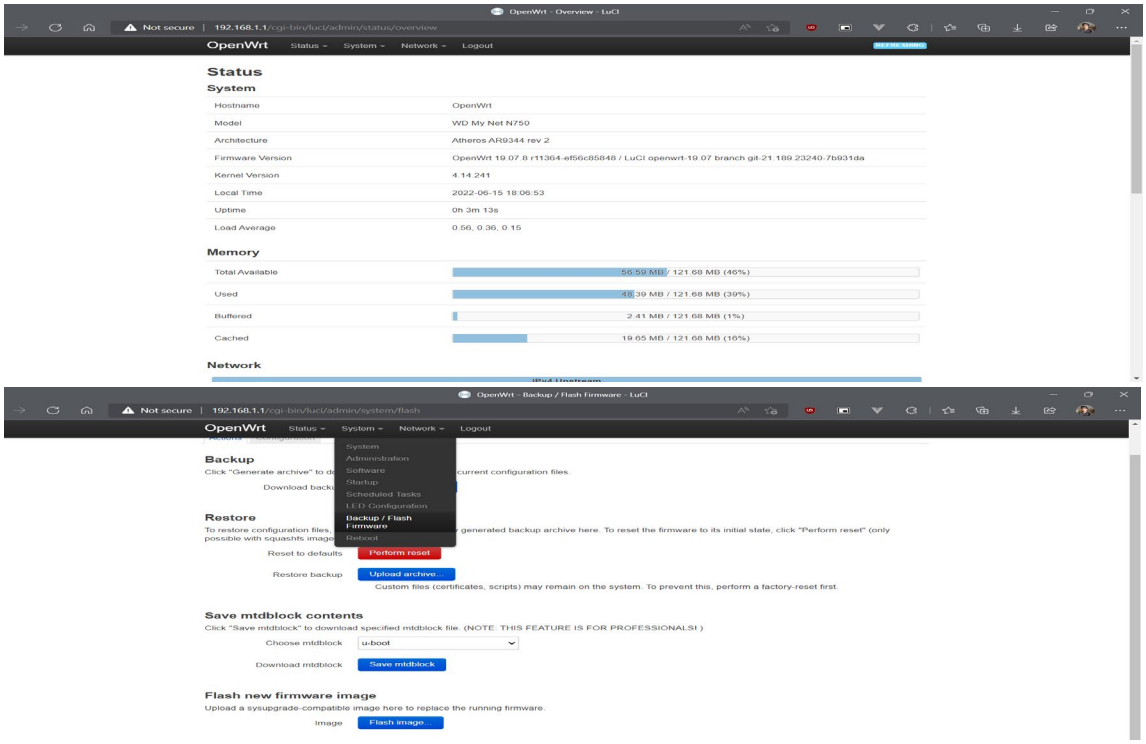


Figura 43: Pasos para la actualización del firmware OpenWrt

Elaborado por: El investigador

Por último, la instalación de ZeroTier se lo realiza en el apartado de “System>Software” en donde se debe actualizar las librerías y por último se busca el paquete de “ZeroTier”, para instalarlo se despliega una ventana emergente en donde se muestra todas las dependencias que necesita como se muestra en la Figura 44, para comprobar la instalación se puede buscar la aplicación de ZeroTier en aplicaciones instaladas.

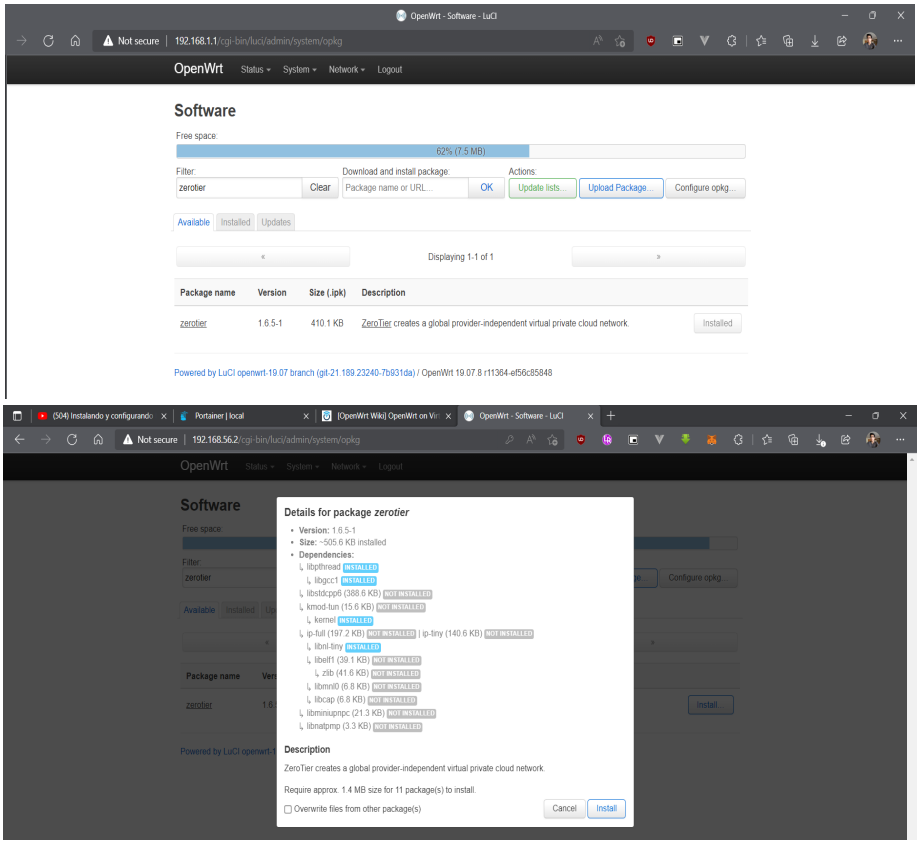


Figura 44: Instalación de Zerotier en OpenWrt
Elaborado por: El investigador

Por el momento el paquete instalado de Zerotier carece de una interfaz gráfica para la configuración de la red, por tanto se necesitó acceder al enrutador por medio del protocolo SSH como se muestra en la Figura 45. Para luego ingresar el comando

```
zerotier-cli join $NETWORK_ID
```

De igual manera, se requiere autorizar al enrutador para que se pueda conectar a la red, una vez autorizado se puede verificar que se ha creado una interfaz de red el mismo que contiene una dirección IP de la red de Zerotier.

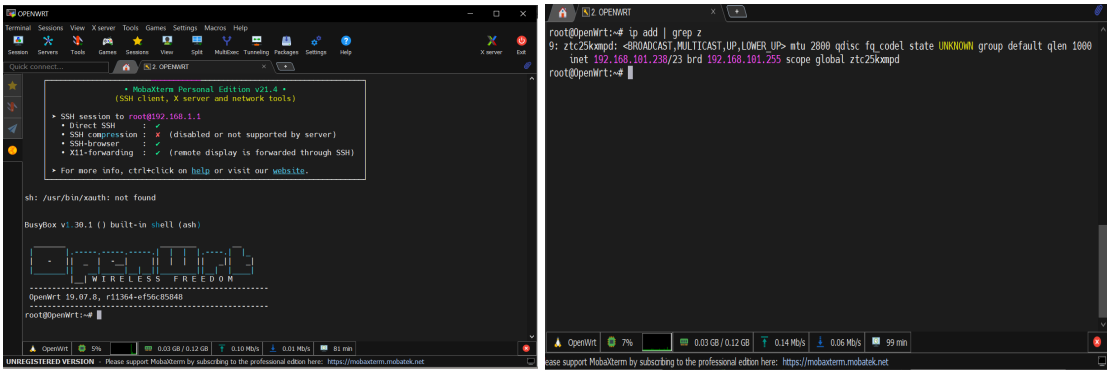


Figura 45: Ingreso por SSH al enrutador y verificación de la red de Zerotier

Elaborado por: El investigador

Para configurar al enrutador como punto de acceso a la red, fue necesario crear una interfaz virtual que conecte con la interfaz de red de Zerotier, para luego crear las respectivas reglas del firewall mediante los siguientes comandos:

```
uci set network.ZeroTier=interface
uci set network.ZeroTier.ifname=$(ifconfig | grep zt | awk '{print $1}')
uci set network.ZeroTier.proto='none'
```

```
uci add firewall zone
uci set firewall.@zone[-1].name='vpn'
uci set firewall.@zone[-1].network='ZeroTier'
uci set firewall.@zone[-1].input='ACCEPT'
uci set firewall.@zone[-1].forward='ACCEPT'
uci set firewall.@zone[-1].masq='1'
uci set firewall.@zone[-1].output='ACCEPT'
uci add firewall forwarding
uci set firewall.@forwarding[-1].dest='lan'
uci set firewall.@forwarding[-1].src='vpn'
uci add firewall forwarding
uci set firewall.@forwarding[-1].dest='vpn'
uci set firewall.@forwarding[-1].src='lan'
```

Finalmente, el enrutador ha sido configurado correctamente como se puede observar en la Figura 46, lo que permite que los usuarios conectados al enrutador puedan acceder a los servicios web del servidor, al ser este un servidor DNS, el enrutador debe ser capaz de asignar las direcciones DNS a sus clientes, el cual corresponde a la dirección “192.168.101.22”, para ello se accede a las configuraciones de la interfaz LAN, en el apartado de “DHCP Server, Advanced Settings” se asigna la dirección IP en la opción de “DHCP-Options” en el formato mostrado en la Figura 46.

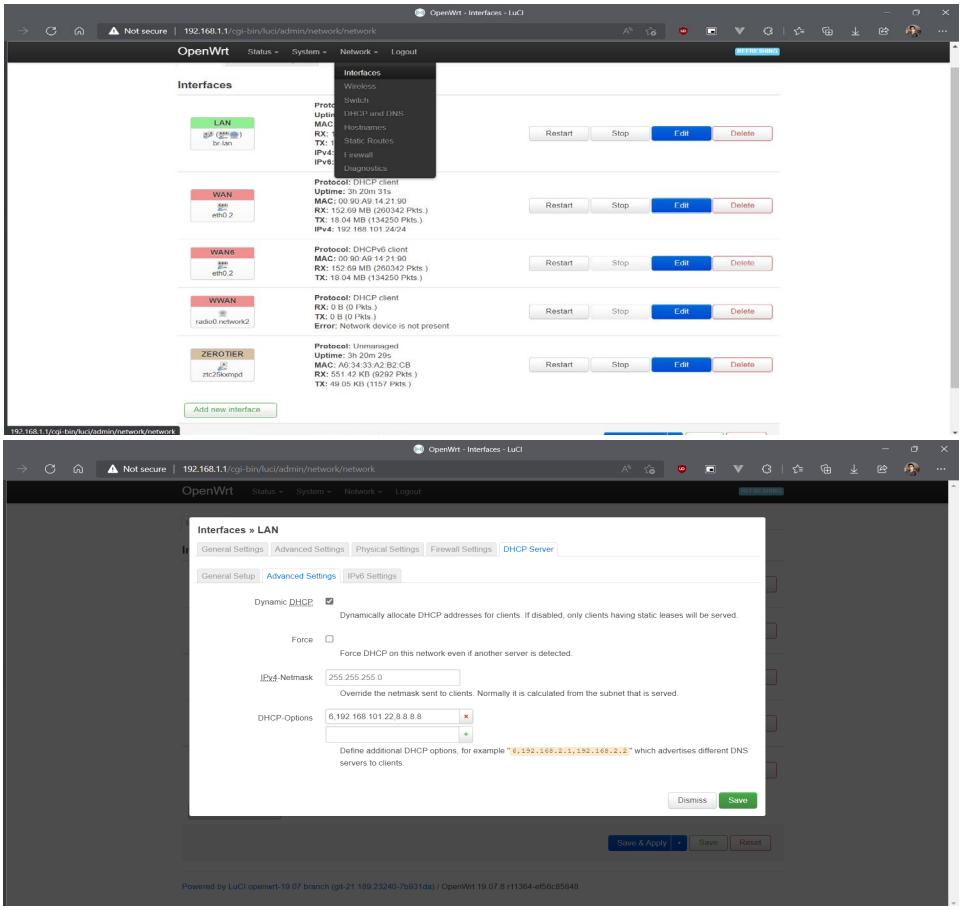


Figura 46: Creación de la interfaz para la red Zerotier y configuración del servidor DHCP

Elaborado por: El investigador

3.2.5. Selección del lenguaje de programación para la aplicación web

La aplicación web es la interfaz de gestión entre el usuario final y los dispositivos, al ser esta una página de tipo CSR(Client-side rendering) es decir se renderiza en el lado del cliente, se libera recursos del servidor puesto que este solo va a manejar las peticiones API REST para cargar con información a la página renderizada por el cliente, en base a la tabla 19 se ha elegido el framework de Vue.js para el desarrollo del frontend y además se utilizó JavaScript como el lenguaje para el desarrollo del backend gracias a sus características como se muestra en la tabla 20.

Respecto a las librerías utilizadas por la aplicación web, en la tabla 21 se contemplan las principales librerías utilizadas, cabe recalcar que se utilizó el gestor de paquetes de node (npm) para instalarlas.

Tabla 21: Principales librerías utilizadas para el desarrollo de la aplicación web

Librería	Versión	Función
node-sass	^4.12.0	Proporciona un enlace entre Node.js a LibSass, un popular preprocesador de hojas de estilo
sass-loader	^7.3.1	Carga archivos Sass y/o SCSS y los compila en formato CSS
Nuxt	^2.15.3	Framework JS que se construye sobre Vue.js, permite renderizar contenido en el cliente y en el servidor.
Mongoose	^5.10.15	Herramienta asíncrona para el modelado de objetos en MongoDB
Bootstrap	4.3.1	Front-end Framework para el desarrollo de la página
Bcrypt	^5.0.0	Permite crear hash para la encriptación de datos
jsonwebtoken	^8.5.1	Permite la creación de token para el acceso a datos
Axios	^5.13.1	Cliente HTTP basado en promesas
Element-ui	^2.14.1	Contiene componentes para Vue.js
Express	^4.17.1	Permite la creación de API's
jqwidgets-scripts	^12.2.1	Librería para importar elementos para el dashboard, como Gauge, sliders, charts, etc
Mqtt	^4.2.6	Cliente mqtt para la conexión
Whatsapp-web.js	^1.16.6	Librería que proporciona un API para la conexión con Whatsapp
Morgan	^1.10.0	Middleware de registro de peticiones HTTP para node.js

Elaborado por: El investigador

3.2.6. Programación de la página web

Para el desarrollo de la aplicación web fue necesario crear los distintos servicios como EMQX que funciona como el bróker MQTT, MongoDB es la base de datos del sistema, Pi-hole es el servidor DNS y por último Node.js servicio que copila la aplicación web; la instalación de dichos servicios se lo realiza por medio de contenedores separados, por medio de Docker, este último se debe instalar en el servidor bajo los siguientes y se comprueba su instalación mediante el comando "docker --version" (Figura 47).

- sudo apt update && sudo apt upgrade -y
- curl -sSL https://get.docker.com | sh
- sudo usermod -aG docker pi
- sudo reboot

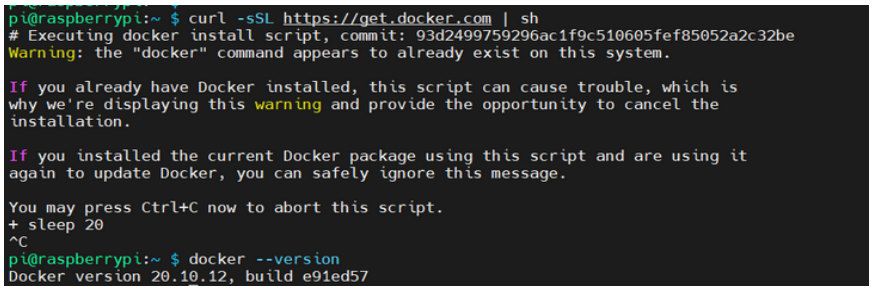


Figura 47: Instalación de Docker en el servidor
Elaborado por: El investigador

Para una mayor gestión sobre los contenedores se optó por instalar Portainer.io, en donde es necesario crear un volumen en Docker, descargar la imagen, abrir puertos y por último configurarla, para ello se ingresó los siguientes comandos :

- docker volume create portainer_data
- docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce

Al acceder a la dirección del servidor que en este caso es “192.168.101.22” y con el puerto 9000 se puede ingresar a la interfaz de Portainer.io como se muestra en la Figura 48, en donde se podrá crear contenedores por medio de comandos o por medio de una archivo denominado “Docker-compose” siendo este último el formato más utilizado para crear contenedores.

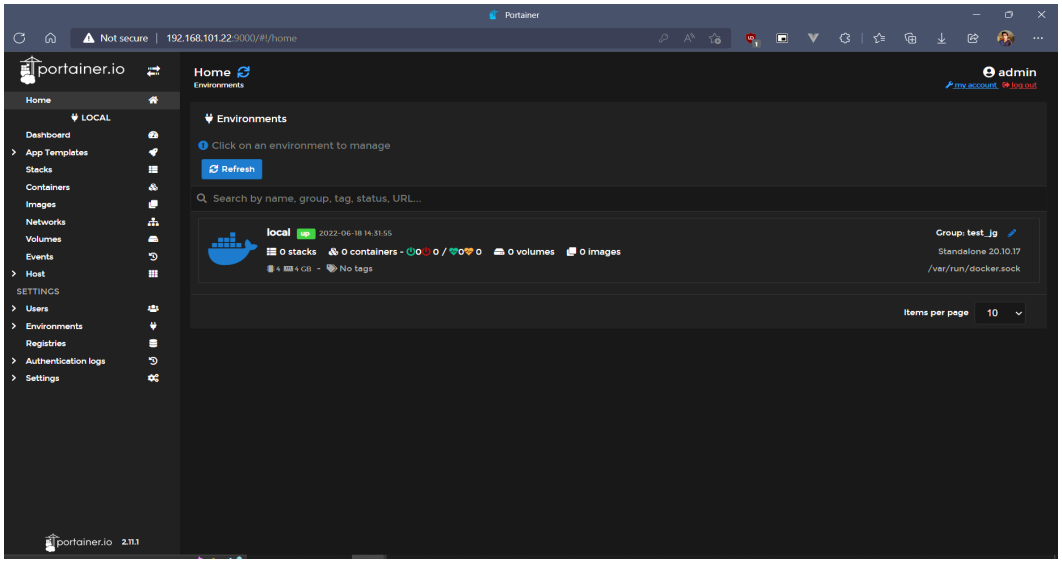


Figura 48: Interfaz gráfica de Portainer.io
Elaborado por: El investigador

Una vez instalado el gestor de contenedores se procede a instalar Pi-hole, este proceso se lo realiza en la sección de “stack>add stack” en donde se despliega una interfaz para ingresar el nombre del stack y el Docker-compose del mismo (Figura 49), en este archivo se define el nombre del contenedor, la imagen, los puertos a utilizar, variables de entorno como la contraseña y la zona horaria, los volúmenes necesarios para funcionar, y por último la condición de cuando reiniciar el contenedor como se muestra en el siguiente ejemplo:

```
version: "3"

services:
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    # For DHCP it is recommended to remove these ports and instead add: network_mode: "host"
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "67:67/udp" # Only required if you are using Pi-hole as your DHCP server
      - "85:80/tcp"
    environment:
      TZ: 'America/Guayaquil'
      WEBPASSWORD: '*********'
    # Volumes store your data between container upgrades
    volumes:
      - './etc-pihole:/etc/pihole'
      - './etc-dnsmasq.d:/etc/dnsmasq.d'
    #   https://github.com/pi-hole/docker-pi-hole#note-on-capabilities
    cap_add:
      - NET_ADMIN # Recommended but not required (DHCP needs NET_ADMIN)
    restart: unless-stopped
```

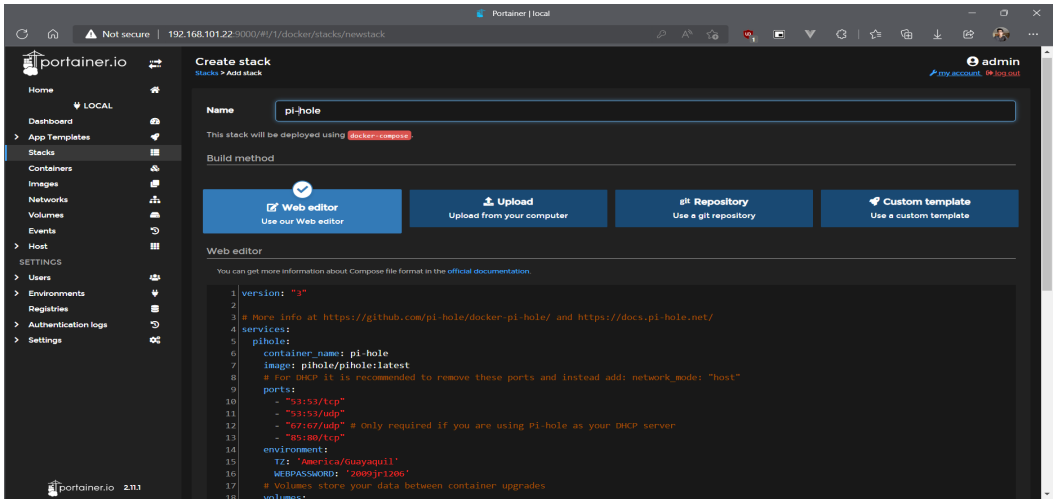


Figura 49: Creación del contenedor de Pi-hole

Elaborado por: El investigador

Para acceder a Pi-hole, en el navegador se ingresa a la dirección de servidor, el puerto del contenedor y la palabra admin, que en este caso fue la dirección "192.168.101.22:85/admin" dentro del panel de control se realizó la configuración para asignar un nombre de dominio a una IP como se muestra en la Figura 50, con la finalidad de que los distintos clientes puedan acceder por medio de este nombre a todas los servicios web del servidor. Para la configuración fue necesario irse al apartado de "Local DNS > DNS Records" y se agregó los campos solicitados, para luego seleccionar "add" el cual agrega en la lista "List of local DNS domains" el dominio agregado.

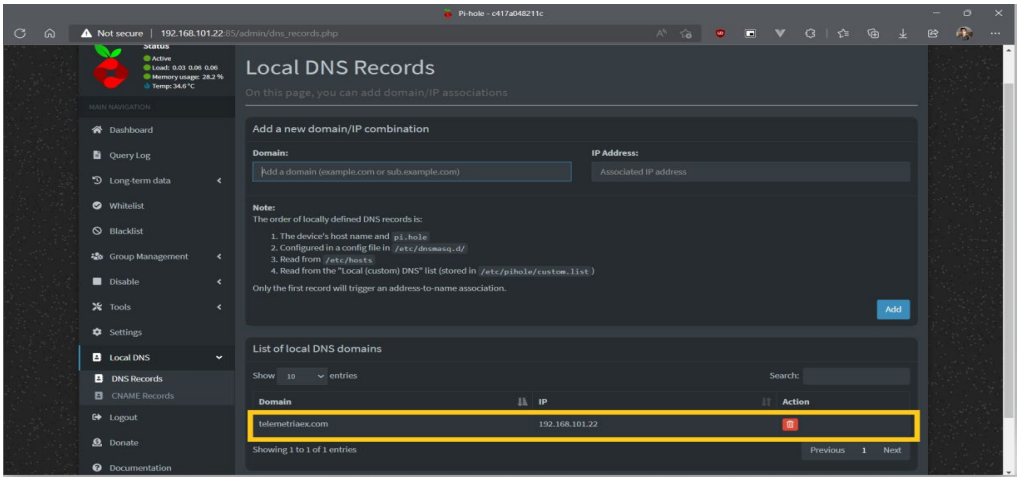


Figura 50: Asignación de un dominio al servidor en Pi-hole

Elaborado por: El investigador

Con la finalidad de que los clientes de la red de Zerotier tengan como dirección de DNS la dirección del servidor en donde se está ejecutando Pihole, se debe ir al panel de control de Zerotier y asignar la dirección IP del servidor DNS.

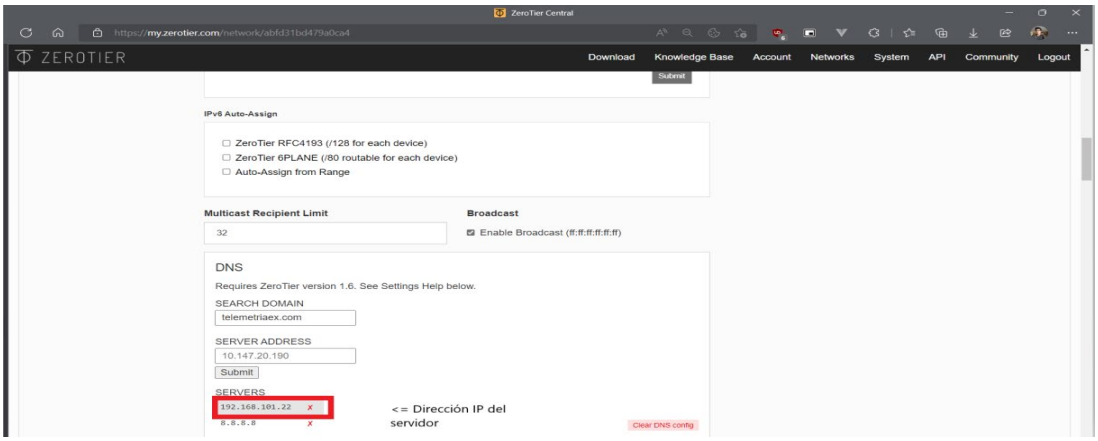


Figura 51: Asignación de un servidor DNS en Zerotier

Elaborado por: El investigador

Para comprobar el funcionamiento del dominio, los clientes de Zerotier podrán sustituir la IP del servidor, que en este caso es "192.168.101.22" por el dominio "telemetriaex.com", como por ejemplo se muestra en la Figura 52.

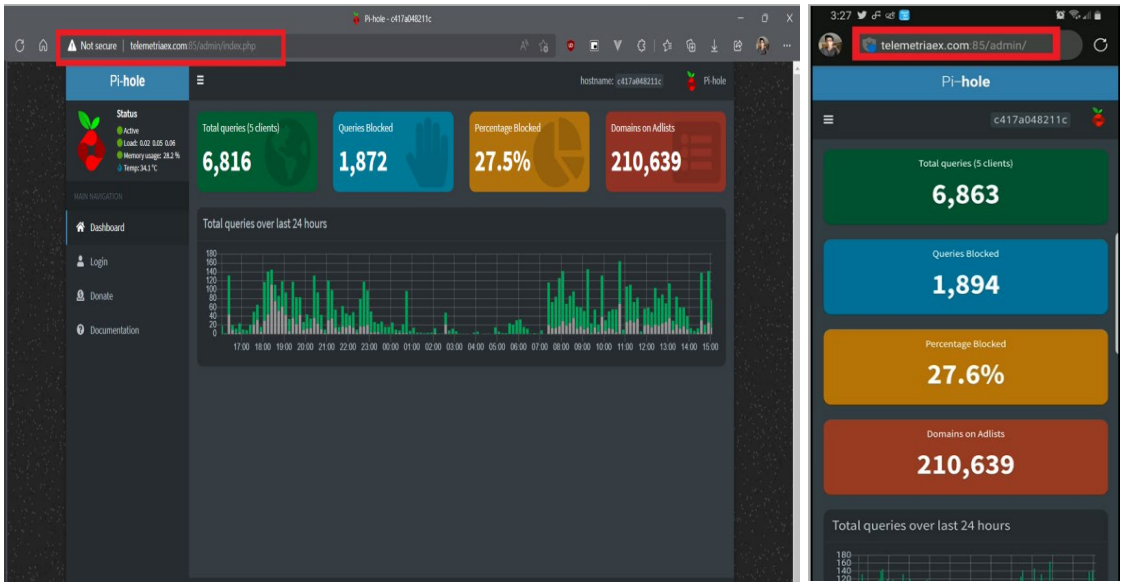


Figura 52: Dominio "telemetriaex.com" funcionando correctamente en clientes de la red SDN Zerotier

Elaborado por: El investigador

Luego se procedió a crear los servicios de EMQX y Mongo, que al igual que Pi-hole se lo crea en portainer.io, mediante un nuevo stack que contiene el siguiente código:

```
version: "3.8"
volumes:
  vol-emqx-data:
    name: foo-emqx-data
#  vol-emqx-etc:
#    name: foo-emqx-etc
  vol-emqx-log:
    name: foo-emqx-log

services:
  mongo:
    container_name: mongo
    image: mongo:4.4
    restart: always
    environment:
      TZ: "$[63]"
      MONGO_INITDB_ROOT_USERNAME: "${mongo_username}"
      MONGO_INITDB_ROOT_PASSWORD: "${mongo_password}"
    volumes:
      - ./mongodata:/data/db
    ports:
      - ${mongo_port_ext}:27017

  emqx:
    container_name: emqx
    #image: emqx/emqx:4.2.11
    image: emqx/emqx:4.2.11
    restart: always
    ports:
      - 18083:18083
      - 1883:1883
      - 8883:8883
      - 8083:8083
      - 8085:8081

    volumes:
      - /etc/localtime:/etc/localtime:ro
      - vol-emqx-data:/opt/emqx/data
    #  - vol-emqx-etc:/opt/emqx/etc
      - vol-emqx-log:/opt/emqx/log

    links:
      - mongo

    extra_hosts:
      - "localhost:192.168.101.22"

    environment:
      EMQX_NAME: jr2009
      EMQX_HOST: 127.0.0.1

      TZ: "$[63]"
      EMQX_DASHBOARD__DEFAULT_USER__PASSWORD: "${EMQX_DEFAULT_USER_PASSWORD}"
      EMQX_MANAGEMENT__DEFAULT_APPLICATION__SECRET: "${EMQX_DEFAULT_APPLICATION_SECCRET}"
      EMQX_ALLOW_ANONYMOUS: "false"
      EMQX_NOMATCH: "deny"

      # MONGO CONNECTION
      EMQX_AUTH__MONGO__TYPE: single
      EMQX_AUTH__MONGO__TOPOLOGY__POOL_SIZE: 1
      EMQX_AUTH__MONGO__TOPOLOGY__MAX_OVERFLOW: 0

      EMQX_AUTH__MONGO__SERVER: "mongo:${mongo_port_ext}"
      EMQX_AUTH__MONGO__POOL: 8
```

```
      EMQX_AUTH__MONGO__LOGIN: "${mongo_username}"
      EMQX_AUTH__MONGO__PASSWORD: "${mongo_password}"
      EMQX_AUTH__MONGO__AUTH_SOURCE: admin

      EMQX_AUTH__MONGO__DATABASE: "IoT_GL"
      EMQX_AUTH__MONGO__AUTH_QUERY__COLLECTION: "emqxauthrules"

      EMQX_AUTH__MONGO__SUPER_QUERY__COLLECTION: "emqxauthrules"
      EMQX_AUTH__MONGO__SUPER_QUERY__SUPER_FIELD: "is_superuser"
      EMQX_AUTH__MONGO__SUPER_QUERY__SELECTOR: "username=%u"
      EMQX_AUTH__MONGO__SUPER_QUERY: "off"

      EMQX_AUTH__MONGO__AUTH_QUERY__PASSWORD_HASH: plain
      EMQX_AUTH__MONGO__AUTH_QUERY__PASSWORD_FIELD: "password"
      EMQX_AUTH__MONGO__AUTH_QUERY__SELECTOR: "username=%u"
      EMQX_AUTH__MONGO__ACL_QUERY: "on"
      EMQX_AUTH__MONGO__ACL_QUERY__COLLECTION: "emqxauthrules"
      EMQX_AUTH__MONGO__ACL_QUERY__SELECTOR: "username=%u"
      EMQX_LOADED_PLUGINS: "emqx_recon,emqx_retainer,emqx_management,emqx_dashboard,emqx_auth_mongo"
      EMQX_LISTENER__TCP__EXTERNAL__MAX_CONNECTIONS: 10000
      EMQX_NODE__DIST_BUFFER_SIZE: "200MB"
      EMQX_SYSMON__LARGE_HEAP: "1024MB"
      EMQX_ZONE__EXTERNAL__FORCE_SHUTDOWN_POLICY: "100000|64MB"
      EMQX_ZONE__EXTERNAL__MQUEUE_STORE_QOS0: "false"
      EMQX_MQTT__MAX_PACKET_SIZE: "1MB"
```

En este código contiene la ejecución de MongoDB que establece el nombre del contenedor, la versión de la imagen, configuración de variables de entorno como la zona horaria, el nombre de usuario root, la contraseña de usuario root, además se configura el volumen a utilizar para almacenar los datos y por último el puerto por donde va a trabajar. De igual manera la ejecución de EMQX se estableció los mismos parámetros con la diferencia que este requiere de más puertos para trabajar como son:

- 1883 (Conexión MQTT)
- 8883 (Conexión MQTT de manera segura)
- 8093 (Conexión por medio de Web Sockets)
- 8081 (API)
- 18083 (Interfaz gráfica de EMQX)

Y demás variables de entorno para su óptimo funcionamiento, como son: el método de autenticación, tamaño del buffer, cargar extensiones, tamaño máximo de paquetes MQTT, entre otros.

Se pudo verificar que EMQX se ha instalado correctamente, ingresando a la dirección “telemetriaex.com:18083”, mientras que para MongoDB, se requirió de un programa adicional que permita tener gestión sobre la base de datos el cual es “MongoDB

Compass” para ingresar se deberá crear una nueva conexión con la siguiente dirección “mongodb://$username:$password@telemetriaex.com:27017/?authSource=admin” como se muestra en la siguiente figura:

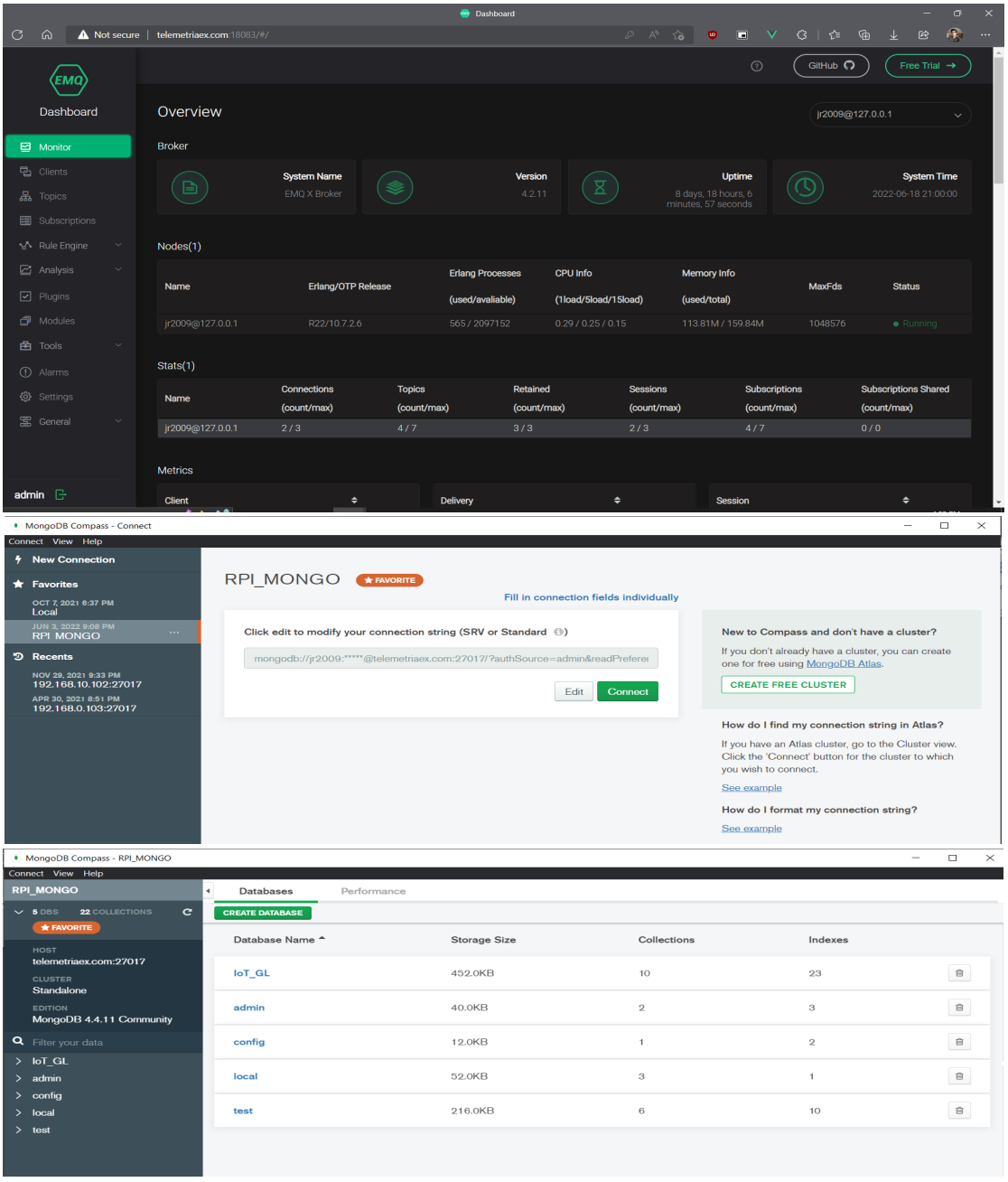


Figura 53: Servicios de EMQX y MongoDB creados en Portainer.io
Elaborado por: El investigador

Por último el contenedor de Node, encargado de copilar la página final, requiere una versión inferior a node 14, al trabajar con la librería “whatsapp-web.js” en la aplicación

web, el contenedor requiere de un navegador de Chromium, para lo cual se crea una imagen que contenga node y chromium integrado, mediante el archivo .dockerfile, para luego por medio de Portainer.io copilar la imagen:

```
FROM node:14-alpine3.15
# Installs latest Chromium (100) package.
RUN apk add --no-cache \
    chromium \
    nss \
    freetype \
    harfbuzz \
    ca-certificates \
    ttf-freefont \
    g++ \
    make \
    python2
# Tell Puppeteer to skip installing Chrome. We'll be using the installed package.
ENV PUPPETEER_SKIP_CHROMIUM_DOWNLOAD=true \
    PUPPETEER_EXECUTABLE_PATH=/usr/bin/chromium-browser
# Puppeteer v13.5.0 works with Chromium 100.
RUN npm install -g puppeteer@13.5.0
# Add user so we don't need --no-sandbox.
RUN addgroup -S pptruser && adduser -S -G pptruser pptruser \
    && mkdir -p /home/pptruser/Downloads /app \
    && chown -R pptruser:pptruser /home/pptruser \
    && chown -R pptruser:pptruser /app
# Run everything after as non-privileged user.
USER pptruser
```

Para construir la imagen se necesitó ir a “images>Build image”, en donde se llenan los campos como el nombre a la imagen y el código del archivo Docker file, como se muestra en la siguiente figura:

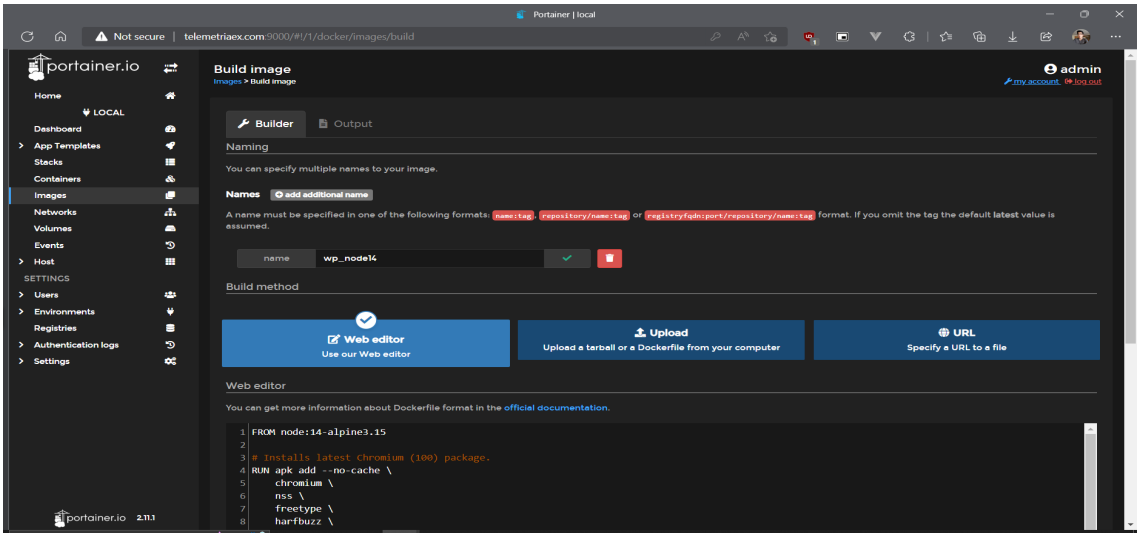


Figura 54: Creación de la imagen de Node v14 junto a chromium

Elaborado por: El investigador

Creado la imagen se procedió a crear el respectivo “stack” para levantar el contenedor, este contenedor se lo ejecuta cuando la aplicación web entre en modo de producción, es decir que el código ha pasado todas las pruebas de funcionamiento y se ha corregido los respectivos errores, el archivo Docker Compose para levantar la aplicación web, requiere del nombre del contenedor, la imagen, la zona horaria, un volumen donde estará el proyecto final, y los puertos 3001 para peticiones API y el 80 para la proyección de la página web, cabe recalcar que al inicializar el proyecto carecerá de librerías, y por ende se las debe instalar con “npm install”, luego de este paso se requiere copilar el proyecto por medio de “npm run build && npm run start” como se muestra en el siguiente código:

```
77réate77: “3.8”
services:
  node:
    container_name: node_thesis
    image: “puppeteer-chrome-linux:v1”
    restart: unless-stopped
    environment:
      TZ: “America/Guayaquil”
    working_dir: /home/node/app
    volumes:
      - /home/pi/Documents/final_app:/home/node/app
    ports:
      - “3001:3001”
      - “80:80”
    #command: sh -c “npm install”
    command: sh -c “npm run build && npm run start
```

Levantado los servicios web, se procede con el diseño del Frontend, el cual es una adaptación del proyecto de código abierto de Creative Tim “Nuxt Black Dashboard” y está basado en Bootstrap 4, viene en dos versiones Oscura y clara, además ofrece una variedad de componentes para el diseño de un sistema completo de gestión y visualización de datos, para instalarlo se requirió clonar el proyecto desde su página oficial hasta el servidor, el proyecto descargado consta de 7 ejemplos de vistas, 3 plugins y 16 elementos para configurar. Para instalar el proyecto fue necesario ejecutar el comando “npm install” y luego de ello para copilar la aplicación web se ejecutó el comando “npm run dev” como se muestra en la Figura 55.

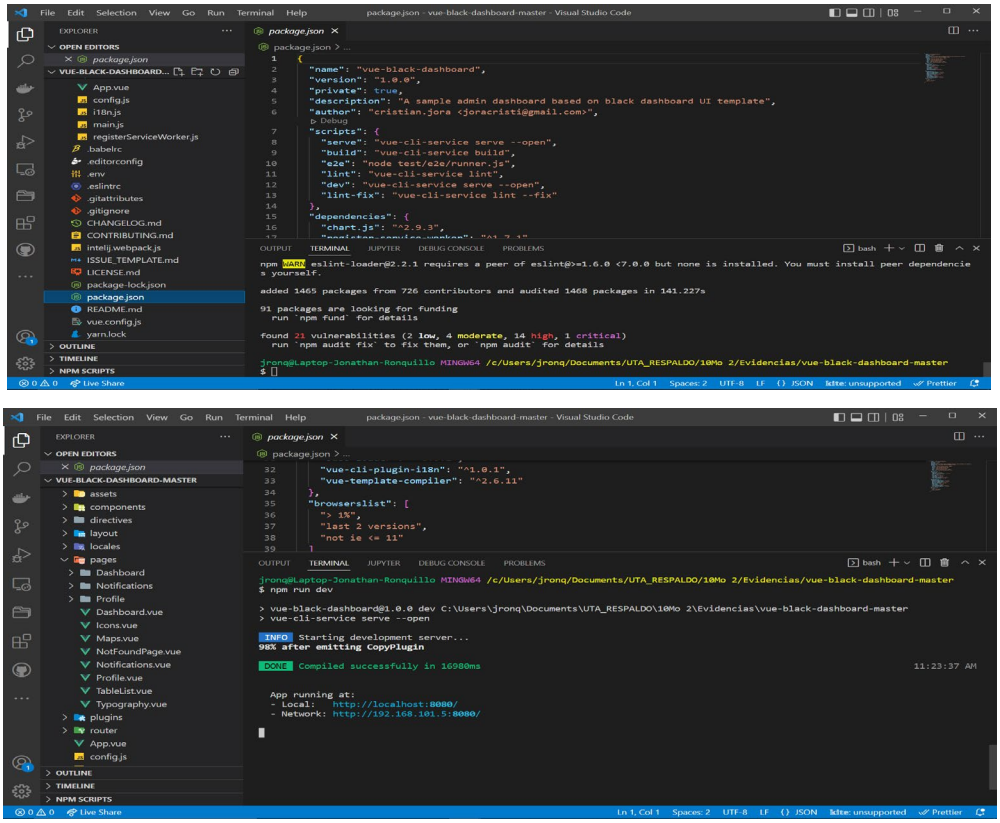


Figura 55: Instalación de paquetes y compilación del proyecto “Vue Black Dashboard”

Elaborado por: El investigador

Al acceder a la aplicación web por medio del puerto 8080, se mostró la siguiente interfaz (Figura 56), esta interfaz maneja muchas librerías y cuenta con demasiado código innecesario para el proyecto, por lo que se optó por realiza un proyecto en blanco con nuxt mediante el código “npm init nuxt-app $project_name”, el mismo que despliega un asistente para la creación del proyecto como se muestra en la Figura 57.

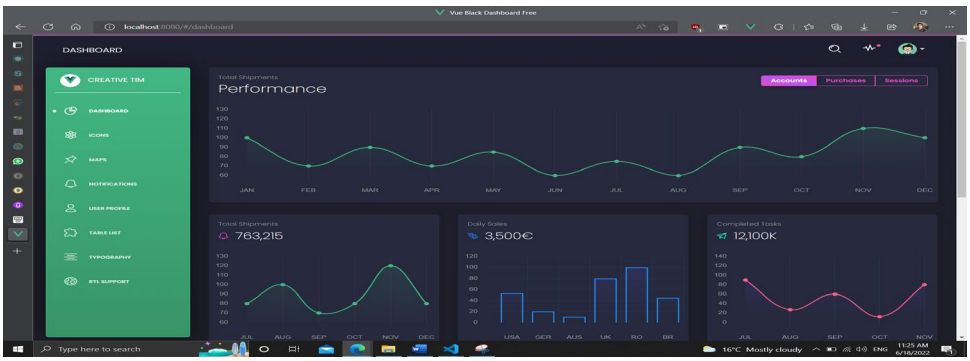


Figura 56: Interfaz web de “Vue Black Dashboard”

Elaborado por: El investigador

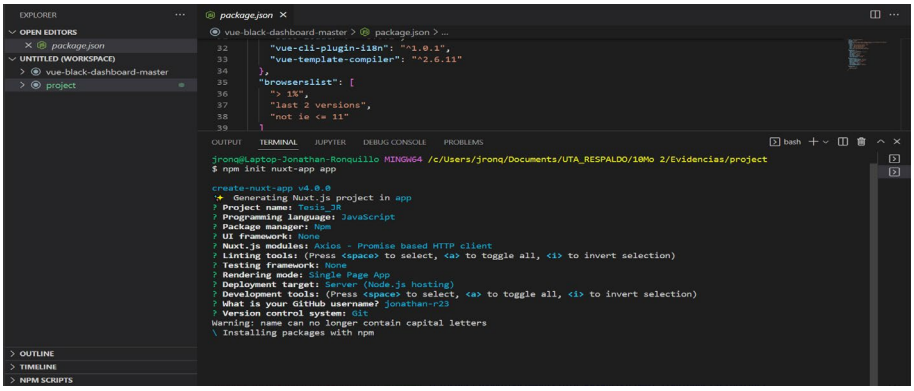


Figura 57: Creación de un proyecto en Nuxt.js

Elaborado por: El investigador

Ya creado el proyecto se accede a la carpeta “app” por consola y se ingresa el comando “npm run dev” cabe recalcar que el desarrollo de la aplicación web se lo realizó de manera local, para posteriormente pasarlo al servidor por medio de Github ya una vez corregido errores y probando que todo funcione de mejor manera, es por ello que se accedió a “localhost:3000” en lugar “telemetriaex.com” como se muestra en la siguiente figura:

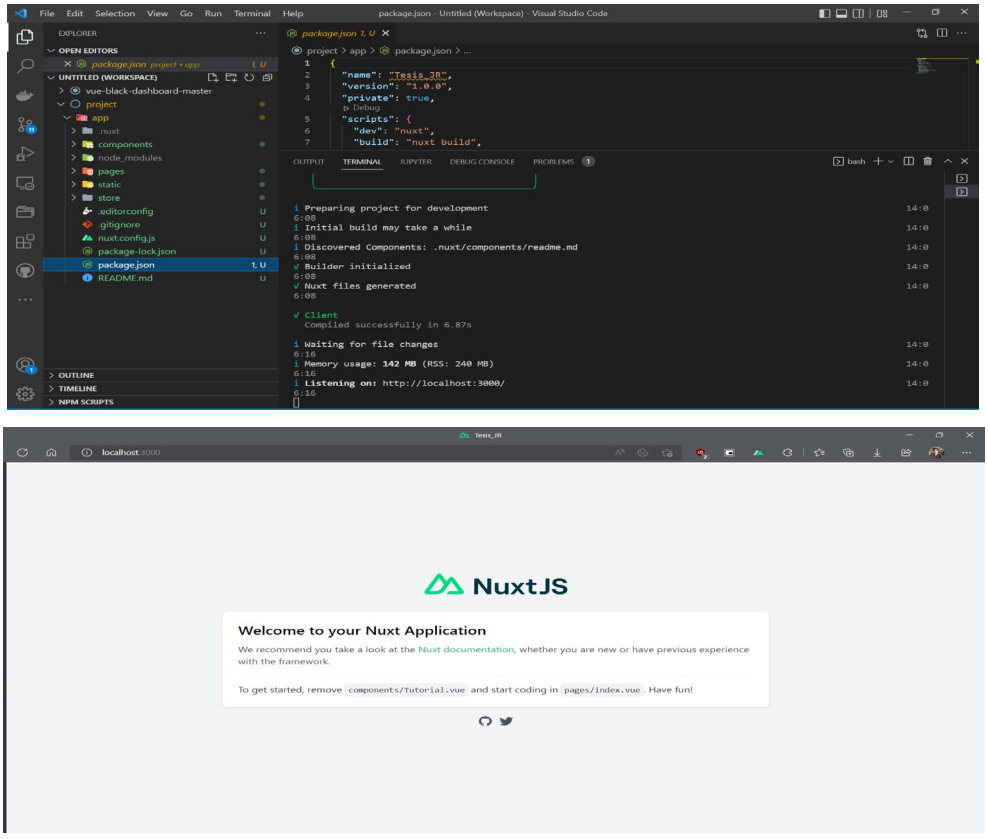


Figura 58: Interfaz web de Nuxtjs

Elaborado por: El investigador

Dentro de la carpeta “app” se copiaron los archivos del proyecto “Vue Black Dashboard” como las carpetas components, layouts, pages, plugins, statics, útil, y los archivos config.js y nuxt.config,js. Dentro de la carpeta del proyecto se agregó en package.json los paquetes tratados en la tabla 21, y además en scripts se agregó el comando npm para correr el backend, obteniendo asi el siguiente archivo package.json:

```
{
  “name”: “app”,
  “80réate80”: “1.0.0”,
  “private”: true,
  “scripts”: {
    “dev”: “nuxt”,
    “devn”: “nodemon api/index.js –exec babel-node –ignore wp-session.json”,
    “build”: “nuxt build”,
    “start”: “nuxt start”,
    “generate”: “nuxt generate”
  },
  “80réate80eña80s”: {
    “@nuxtjs/axios”: “^5.13.1”,
    “@nuxtjs/pwa”: “^3.2.2”,
    “@patrikstas/finite-state-machine”: “^3.0.0-rc.3”,
    “bcrypt”: “^5.0.0”,
    “80réate80eñ”: “4.3.1”,
    “chart.js”: “^2.7.1”,
    “colors”: “^1.4.0”,
    “cookieparser”: “^0.1.0”,
    “core-js”: “^3.9.1”,
    “cors”: “^2.8.5”,
    “d3”: “^5.7.0”,
    “dotenv”: “^10.0.0”,
    “element-ui”: “^2.14.1”,
    “es6-promise”: “^4.1.1”,
    “express”: “^4.17.1”,
    “fuse.js”: “^3.2.0”,
    “jqwidgets-scripts”: “^12.2.1”,
    “js-cookie”: “^2.2.1”,
    “jsonwebtoken”: “^8.5.1”,
    “mongodb”: “^4.4.1”,
    “mongoose”: “^5.10.15”,
    “mongoose-unique-validator”: “^2.0.3”,
    “80réate”: “^1.10.0”,
    “mqtt”: “^4.2.6”,
    “nuxt”: “^2.15.3”,
    “nuxt-highcharts”: “^1.0.3”,
    “path”: “^0.12.7”,
    “perfect-scrollbar”: “^1.3.0”,
    “qrcode-terminal”: “^0.12.0”,
    “register-service-worker”: “^1.5.2”,
    “tween.js”: “^16.6.0”,
    “vue-chartjs”: “^3.4.0”,
    “vue2-transitions”: “^0.2.3”,
    “whatsapp-web.js”: “^1.16.5”,
    “ws”: “^8.3.0”
  },
  “devDependencies”: {
    “@babel/cli”: “^7.12.1”,
    “@babel/core”: “^7.12.3”,
    “@babel/node”: “^7.12.6”,
    “@babel/preset-env”: “^7.12.1”,
    “babel-plugin-component”: “^1.1.1”,
    “node-sass”: “^4.12.0”,
    “sass-loader”: “^7.3.1”
  }
}
```

Luego de ello y dentro de la carpeta app se instalaron los paquetes con el comando “npm install”, luego se editó el archivo nuxt.config.js. para crear el nombre del proyecto, la dirección base de la API, variables de entorno para el proyecto, el puerto por donde se desea que la aplicación sea presentada y la dirección de los clientes que pueden acceder a esta, quedando de la siguiente manera el archivo:

```
export default {
  ssr: false,
  /*
  ** Headers of the page
  */
  head: {
    title: 'Monitoreo EX',
    meta: [
      { charset: 'utf-8' },
      { name: 'viewport', content: 'width=device-width, initial-scale=1' },
      { hid: 'description', name: 'description', content: process.env.npm_package_description || ''}
    ],
    link: [
      { rel: 'icon', type: 'image/x-icon', href: '/logo_ex1.png' },
      { rel: 'stylesheet', href: 'https://fonts.googleapis.com/css?family=Poppins:200,300,400,600,700,800' },
      { rel: 'stylesheet', href: 'https://cdnjs.cloudflare.com/81réa/libs/81réa-awesome/5.10.2/css/all.min.css'
},
      { rel: 'stylesheet', type: 'text/css', href: '/styles/jqx.base.css' }
    ],
    bodyAttrs: {
      class: '' // Add `81réat-content` class here to enable "81réat" mode.
    }
  },
  router: {
    linkExactActiveClass: 'active'
  },
  /*
  ** Customize the progress-bar color
  */
  loading: { color: '#fff' },

  css: [
    'assets/css/demo.css',
    'assets/css/81réate-icons.css',
    'assets/sass/black-dashboard.scss'
  ],
  /*
  ** Plugins to load before mounting the App
  */
  plugins: [
    `~/plugins/dashboard-plugin.js`,
  ],
  // Auto import components: https://go.nuxtjs.dev/config-components
  components: [
    {
      path: '~/components',
      pathPrefix: false,
      extensions: ['vue', 'jsx'],
    }
  ],
  buildModules: [],

  modules: [
    '@nuxtjs/pwa',
    '@nuxtjs/axios',
    'nuxt-highcharts',
  ],
  // Axios module configuration: https://go.nuxtjs.dev/config-axios
```

```
  axios: {
    baseURL: process.env.AXIOS_BASE_URL
  },

  env: {
    mqtt_prefix: process.env.MQTT_SSL_PREFIX,
    mqtt_host: process.env.EMQX_HOST,
    mqtt_ws_port: process.env.MQTT_WS_PORT

  },

  build: {
    transpile: [/^element-ui/],
    extend(config, ctx) {
    },
    babel: {
      plugins: [
        [
          'component',
          {
            'libraryName': 'element-ui',
            'styleLibraryName': 'theme-chalk'
          }
        ]
      ]
    }
  },
  server: {
    port: process.env.APP_PORT,
    host: '0.0.0.0',
  }
}
```

Copiados estos archivos y carpetas, y configurando los archivos mencionados, se procedió a diseñar el Frontend, el cual tiene vistas o páginas web como: el registro, inicio de sesión, creación de plantillas, creación de dispositivos, creación de reglas y el tablero de control.

Cabe recalcar que todos los archivos de vistas y componentes tendrán extensión .vue y cada uno de ellos consta de dos partes o hasta tres, la primera es la encarga de crear entradas, botones, texto, entre otros, por medio del lenguaje Vue, la segunda parte es la encargada de dar funciones a la vista como cargar la con datos, llenar tablas, realizar peticiones API y demás, haciendo uso del lenguaje JavaScript, mientras que la tercera parte es la encargada de configurar aspectos de la vista como color, tamaño, etc. Se pueden diferenciar entre ellos mediante las etiquetas que utilizan como: <template></template> para el código vue <script> </script> para el código JavaScript y <style> </style > para configurar la apariencia de la página como se muestra en el siguiente código, el cual pertenece a la vista de registro y contempla el ingreso de 4 datos como son: Nombre, Email, celular y una contraseña, datos que al dar click en el botón

registrar envía por medio de método post hacia la API para que realice la creación del respectivo usuario, obteniendo el siguiente código

```
<template>
  <div class="container login-page">
    <div class="col-lg-4 col-md-6 ml-auto mr-auto">
      <card class="card-login card-white">
        <template slot="header">
          <img src="img//card-primary.png" alt="" />
          <h1 class="card-title">IoT GL</h1>
        </template>

        <form  @submit.prevent>
         <base-input name="name" v-model="user.name" placeholder="Name" addon-left-icon="tim-icons icon-badge">
          </base-input>

          <base-input type="email" name="email" v-model="user.email" placeholder="Email" addon-left-icon="tim-
icons icon-email-85">
          </base-input>

          <base-input type="number" name="83réa" v-model="user.phone" placeholder="Phone" addon-left-icon="tim-
icons icon-mobile">
          </base-input>

         <base-input type="password" name="password" v-model="user.password" placeholder="Password" addon-left-
icon="tim-icons icon-lock-circle">
          </base-input>

          <base-button native-type="submit" type="primary" class="mb-3" size="lg" @click="register" block>
            Register
          </base-button>
        </form>

        <div slot="footer">

          <div class="pull-left">
            <h6>
              <nuxt-link class="link footer-link" to="/login">
                login
              </nuxt-link>
            </h6>
          </div>

          <div class="pull-right">
            <h6><a href="#help!!!" class="link footer-link">Need Help?</a></h6>
          </div>
        </div>
      </card>
    </div>
  </div>
</template>

<script>
export default {
  layout: "auth",
  data() {
    return {
      user: {
        name: "",
        email: "",
        password: "",
        83réa: ""
      }
    };
  },
  methods: {
    checkForm: functionlibrería {
```

```
      if (this.user.name && this.user.password && this.user.email && this.user.phone) {

      }
    },
    register() {
      this.$axios.post("/register", this.user)
      .then((res) => {
          If (res.data.status == "success") {
            console.log(res.data);
            this.$notify({
              type: "success",
              icon: "tim-icons icon-check-2",
              message: "Success! Now you can login..."
            });
            this.user.name = "";
            this.user.password = "";
            this.user.email = "";
            this.user.phone = "";
            return;
          }
      })
      .catch(librería=>{
        console.log(e.response.data);
          if (e.response.data.error.errors.email.kind == "unique") {
            this.$notify({
              type: "danger",
              icon: "tim-icons icon-alert-circle-exc",
              message: "User already exists ☹"});
            return;
          } else {
            this.$notify({
              type: "danger",
              icon: "tim-icons icon-alert-circle-exc",
              message: "Error creating user..."
            });
            return;
          }});}}};
</script>
<style>
.navbar-nav .nav-item p {
  line-height: inherit;
  margin-left: 5px;
}
</style>
```

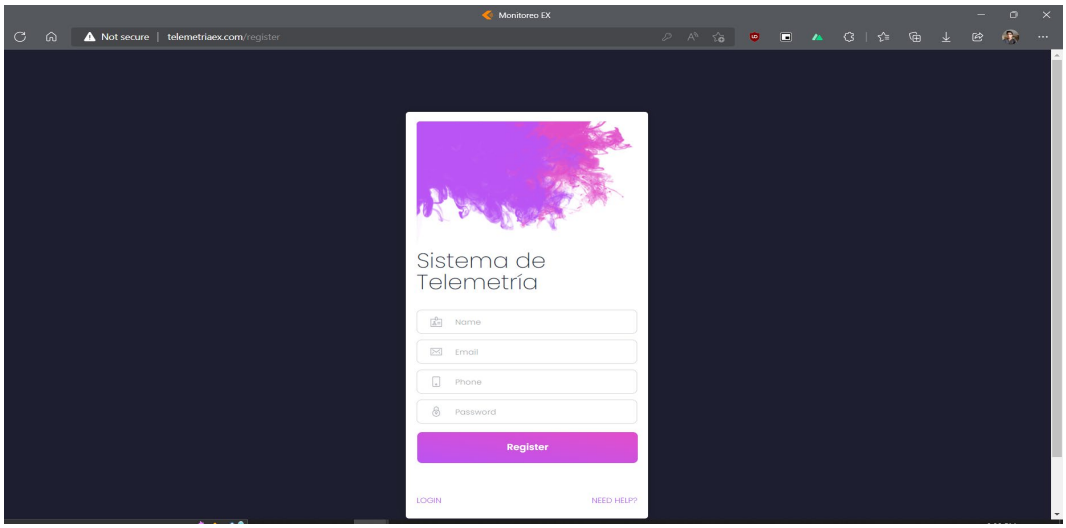


Figura 59: Vista para el registro de nuevos usuarios
Elaborado por: El investigador

La creación de la API para el registro se debe realizar en el backend para lo cual se crea una carpeta dentro de la carpeta “app” llamada “api”, la misma que contiene todo el backend de la aplicación web, en donde el frontend hace peticiones API para el manejo de la información y su posterior proyección. Dentro de esta carpeta se crea el archivo “index.js” que se conecta a MongoDB y busca dentro de la carpeta “routes” todas las API de la aplicación WEB como se muestra en el siguiente código y la carpetas “models” contiene los esquemas para guardar en la base de datos y la carpeta “routers” contiene los archivos para crear las API’s necesarias para la aplicación web, en este caso la creación e ingreso al usuario están en el archivo “api/routes/users.js”.

```
//Importa 85réate85eñ
const express = require('express');
const mongoose = require("mongoose"); //Para trabajar con la db de mongo
const 85réate = require("85réate"); //librería que funciona como middleware
const cors = require("cors"); //Setea las 85réate85eñ de acceso
const colors = require("colors");  //Permite imprimir outputs con colores

require('dotenv').config();

const app = express();

app.use(85réate("tiny")); //muestra una salida por cada endpoint
app.use(express.json()); //permite trabajar con json
app.use(
    express.urlencoded({
        extended: true
    })
);
app.use(cors());

//express routes
app.use("/api1", require("./routes/devices.js"));
app.use("/api1", require("./routes/users.js"));
app.use("/api1", require("./routes/templates.js"));
app.use("/api1", require("./routes/webhooks.js"));
app.use("/api1", require("./routes/emqxapi.js"));
app.use("/api1", require("./routes/alarms.js"));
app.use("/api1", require("./routes/dataprovider.js"));
app.use("/api1", require("./routes/wp.js"));
module.exports = app; //ordena todas las rutas o endpoint en archivos separados

//Listener
app.listen(process.env.API_PORT, ()=>{
    console.log("Servidor API escuchando por el puerto " + process.env.API_PORT);
});

const mongoUserName = process.env.mongo_username;
const mongoPassword = process.env.mongo_password;
const mongoHost = process.env.mongo_host;
const mongoPort = process.env.mongo_port_ext;
const mongoDatabase = process.env.mongo_database;

var uri = "mongodb://" + mongoUserName + ":" + mongoPassword + "@" + mongoHost + ":" + mongoPort + "/" +
mongoDatabase;

const options = {
    useNewUrlParser: true,
    useCreateIndex: true,
    useUnifiedTopology: true,
```

```
    useNewUrlParser: true,
    authSource: "admin"
};
try {
    mongoose.connect(uri, options).then(
        () => {
            console.log("✓ Mongo Successfully Connected!".green);
            global.check_mqtt_superuser();
        },
        (err) => {
            console.log("    Mongo Connection Failed    ".red);
            console.log(err);
        }
    );
} catch (error) {
    console.log("ERROR CONNECTING MONGO ");
    console.log(error);
}
```

Mientras que la API para crear usuarios viene dado bajo el siguiente código, que está en el archivo users.js, el cual toma los datos y los guarda en la base de datos IoT_GL en la colección "users"

```
const express = require("express");
const router = express.Router();
const jwt = require("jsonwebtoken");
const bcrypt = require("bcrypt");
const { checkAuth } = require('../middlewares/authentication.js');
import User from "../models/user.js";
import EmqxAuthRule from "../models/emqx_auth.js";
router.post("/register", async (req, res) => {
    try{
        const name = req.body.name;
        const email = req.body.email;
        const 86réa = req.body.phone;
        const password = req.body.password;
        const encryptedPassword = bcrypt.hashSync(password, 10);
        const newUser = {
            name: name,
            email: email,
            86réa: 86réa,
            password: encryptedPassword
        };
        var user = await User.create(newUser);
        const toSend = {
            status: "success",
        };
        res.status(200).json(toSend);
    }catch(error){
        console.log("ERROR - REGISTER ENDPOINT")
        console.log(error)
        const toSend = {
            status: "error",
            error: error
        };
        res.status(500).json(toSend);
    }
});
```

En la Figura 60 se detalla el proceso general a seguir para la creación de un usuario:

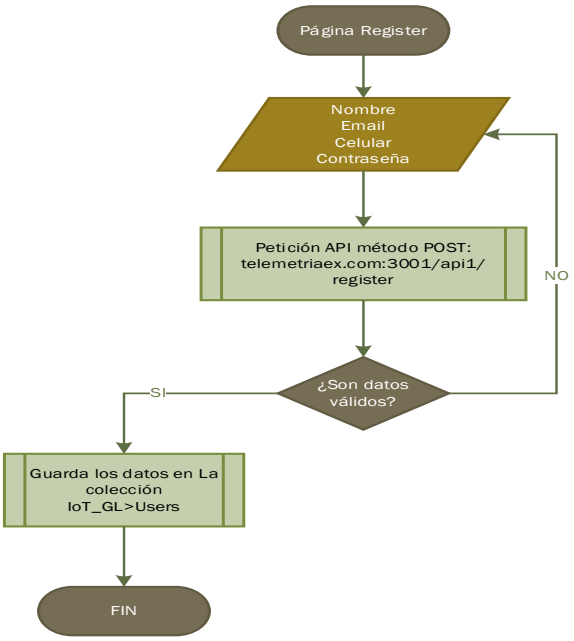


Figura 60: Diagrama de flujo de para el registro de usuarios
Elaborado por: El investigador

De igual manera para el ingreso del usuario al sistema, se requiere de la vista "login" el cual consta de dos entradas de texto que son: el correo y la contraseña, el botón "acceder" el cual llama a una función que envía una petición API tipo POST al enlace "telemetriaex.com:3001/api1/login" con los datos ingresados por el usuario en donde el backend los verifica.

```
<template>
  <div class="container login-page">
    <div class="col-lg-4 col-md-6 ml-auto mr-auto">
      <card class="card-login card-white">
        <template slot="header">
          <img src="img//card-primary.png" alt="" />
          <h1 class="card-title">IoT GL </h1>
        </template>

        <div>
          <base-input name="email" v-model="user.email" placeholder="Email" addon-left-icon="tim-icons icon-
email-85">
          </base-input>

          <base-input name="password" v-model="user.password" type="password" placeholder="Password" addon-
left-icon="tim-icons icon-lock-circle">
          </base-input>
        </div>

        <div slot="footer">
          <base-button native-type="submit" type="primary" class="mb-3" size="lg" @click="login" block>
            Login
          </base-button>

          <div class="pull-left">
            <h6>
              <nuxt-link class="link footer-link" to="/register">
```

```
                Create Account
              </nuxt-link>
            </h6>
          </div>
        </div>
      </card>
    </div>
  </div>
</template>

<script>
const Cookie = process.client ¿ require(“js-cookie”) : undefined;
export default {
  middleware: ‘notAuthenticated’,
  name: “login-page”,
  layout: “auth”,
  data() {
    return {
      user: {
        email: “”,
        password: “”
      }
    };
  },

  mounted() {

  },

  methods:{

    login() {
      this.$axios.post(“/login”, this.user)
      .then((res) => {
        //success! – Usuario creado.
          If (res.data.status == “success”) {
            this.$notify({
              type: “success”,
              icon: “tim-icons icon-check-2”,
              message: “Ha ingresado con éxito “ + res.data.userData.name
            });

            console.log(res.data);

            const auth = {
              token: res.data.token,
              userData: res.data.userData
            }
            this.$store.commit(‘setAuth’,auth);
            localStorage.setItem(‘auth’, JSON.stringify(auth));
            $nuxt.$router.push(‘/dashboard’);

            return;
          }
      })

} } };
</script>

<style>
.navbar-nav .nav-item p {
  line-height: inherit;
  margin-left: 5px;
}
</style>
```

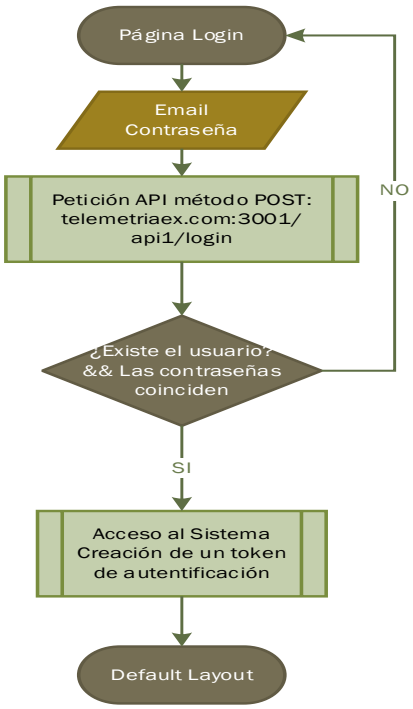


Figura 61: Diagrama de flujo de para el ingreso del usuario al sistema
Elaborado por: El investigador

La API de tipo POST para la validación de la información ingresada por el usuario, viene dado por el siguiente código:

```
//Login
router.post("/login", async (req, res) => {
    const email = req.body.email;
    const password = req.body.password;
    var user = await User.findOne({ email: email });
    if (¡user) {
        const toSend = {
            status: "error",
            error: "Credenciales Invalidas"
        }
        return res.status(401).json(toSend);
    }
    if (bcrypt.compareSync(password, user.password)) {
        user.set('password', undefined, { strict: false }); //borra el campo de password de user
        const token = jwt.sign({ userData: user }, 'ambato2009jr099632', { expiresIn: 60 * 60 * 24 * 30 });

        const toSend = {
            status: "success",
            token: token,
            userData: user
        }
        return res.json(toSend);
    } else {
        const toSend = {
            status: "error",
            error: "Invalid Credentials"
        }
        return res.status(401).json(toSend);
    }
});
```

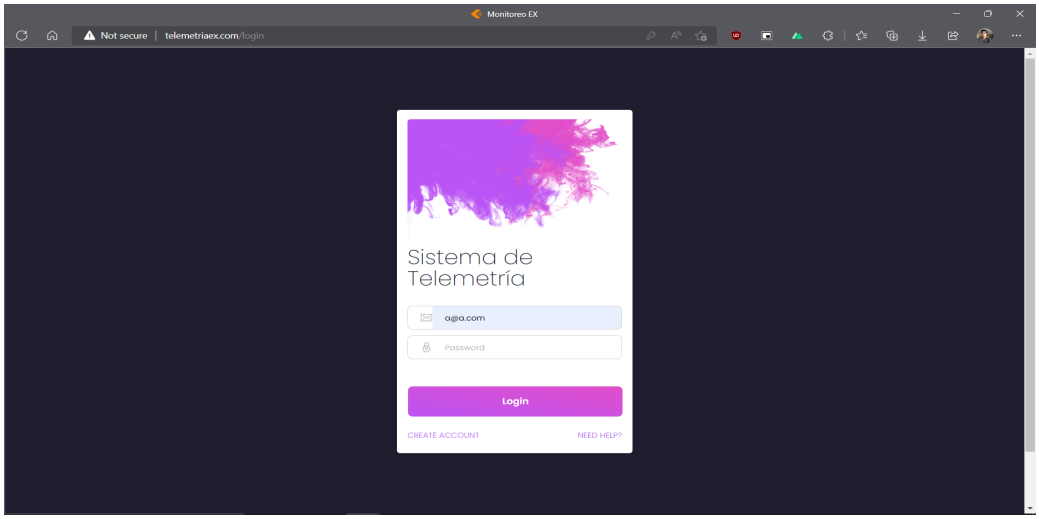


Figura 62: Vista para el ingreso al sistema

Elaborado por: El investigador

Al ingresar al sistema por defecto presenta la vista “default.vue” que se encuentra en la carpeta de “layouts” esta vista contiene un menú de opciones en la parte lateral que redirige al usuario a distintas vista como, el dashboard, dispositivos, alarmas y plantillas, como se muestra en el siguiente código:

```
<side-bar :background-color=”sidebarBackground” short-title=”JR” title=”Monitoreo Ex” >
      <template slot-scope=”props” slot=”links”>
        <sidebar-item
          :link=”{
            name: ‘Dashboard’,
            icon: ‘tim-icons icon-chart-pie-36’,
            path: ‘/dashboard’
          }”
        >
        </sidebar-item>
        <sidebar-item
          :link=”{
            name: ‘Dispositivos’,
            icon: ‘tim-icons icon-mobile’,
            path: ‘/devices’
          }”
        >
        </sidebar-item>
        <sidebar-item
          :link=”{
            name: ‘Alarmas’,
            icon: ‘tim-icons icon-bell-55’,
            path: ‘/alarms’
          }”
        >
        </sidebar-item>
        <sidebar-item
          :link=”{
            name: ‘Plantillas’,
            icon: ‘tim-icons icon-components’,
            path: ‘/templates’
          }”
        >
```

```
        </sidebar-item>
        <li class="active-pro">
          <a href="https://www.ecuatronix.com.ec/" target="_blank">
            <i class="tim-icons icon-world"></i>
            <p>Acerca de Ex </p>
          </a>
        </li>
      </template>
    </side-bar>
```

Este archivo también es capaz de conectarse al bróker MQTT como un cliente más, y tiene funciones como obtener las credenciales MQTT para la conexión, en caso de caducarse la sesión pedir nuevamente las credenciales para la reconexión, además de inicializar el cliente, suscribirlo a los siguientes tópicos: en caso de los mensajes enviados por dispositivos "userid/+/+/sdata", en caso de notificaciones generadas "userid/+/+/notif" y en caso de las imágenes enviadas por la ESPCAM "userid/+/+/espcam" al escuchar estos eventos esta página enviar por NUXT la información a los distintos componentes o widgets creados. Este archivo crea un oyente NUXT bajo el tópico 'mqtt-sender' en donde los widgets encargados de manejar actuadores envián su información a los distintos dispositivos, mediante el tópico y el mensaje .

```
 import { SlideYDownTransition, ZoomCenterTransition } from 'vue2-transitions';
 import mqtt from 'mqtt';
 //default code
export default {
    components: {
      DashboardNavbar,
      ContentFooter,
      DashboardContent,
      SlideYDownTransition,
      ZoomCenterTransition,
      SidebarShare
    },
    data() {
      return {
        sidebarBackground: 'blue', //vue|blue|91réate|green|red|primary
        client: null,
        options: {
          host: process.env.mqtt_host,
          port: process.env.mqtt_ws_port,
          endpoint: "/mqtt",
          clean: true,
          connectTimeout: 5000,
          reconnectPeriod: 5000,
          clientId: "web_" + this.$store.state.auth.userData.name + "_" + Math.floor(Math.random() * 1000000 +
1),
          username: "",
          password: ""
        }
      };
    },
methods: {
      async getMqttCredentials() {
        try {
          const axiosHeaders = {
```

```
            headers: {
              token: this.$store.state.auth.token
            }
          };
          const credentials = await this.$axios.post("/getmqttcredentials", null ,axiosHeaders);
          console.log(credentials.data)
          if(credentials.data.status=="success"){
            this.options.username = credentials.data.username;
            this.options.password = credentials.data.password;
          }
        } catch (error) {
          console.log(error);
          if (error.response.status == 401) {
          console.log("NO VALID TOKEN");
          localStorage.clear();

          const auth = {};
          this.$store.commit("setAuth", auth);

          window.location.href = "/login";
          }
        }
      },
      async getMqttCredentialsForReconnection(){
        try {
          const axiosHeaders = {
            headers: {
              token: this.$store.state.auth.token,
            }
          };
          const credentials = await this.$axios.post("/getmqttcredentialsforreconnection", null
,axiosHeaders);
          console.log(credentials.data);
          if(credentials.data.status=="success"){
            this.client.options.username = credentials.data.username;
            this.client.options.password = credentials.data.password;
          }
        } catch (error) {
          console.log(error);
          if (error.response.status == 401) {
          console.log("NO VALID TOKEN");
          localStorage.clear();
          const auth = {};
          this.$store.commit("setAuth", auth);

          window.location.href = "/login";
        }}
      },
      async startMqttClient() {
        await this.getMqttCredentials();
        const deviceSubscribeTopic = this.$store.state.auth.userData._id + "/+/+/sdata";
        const notifSubscribeTopic = this.$store.state.auth.userData._id + "/+/+/notif";
        const espcamSubscribeTopic = "$queue/"+ this.$store.state.auth.userData._id + "/+/+/espcam";
        const connectUrl = process.env.mqtt_prefix + this.options.host + ":" + this.options.port +
this.options.endpoint;
        try {
          this.client = mqtt.connect(connectUrl, this.options);
        } catch (error) {
          console.log(error);
        }
        this.client.on('connect', () => {
          this.client.subscribe(espcamSubscribeTopic, {qos:0}, (err)=>{
            if (err){
              console.log("Error in DeviceSubscription");
              return;
            }
            console.log("ESPCAM subscription Success");
          })
          //SDATA SUBSCRIBE
          this.client.subscribe(deviceSubscribeTopic, {qos:0}, (err) => {
            if (err){
```

```
            console.log("Error in DeviceSubscription");
            return;
          }
          console.log("Device subscription Success");
        });
        this.client.subscribe(notifSubscribeTopic, {qos:0}, (err) => {
          if (err){
            console.log("Error in NotifSubscription");
            return;
          }
          console.log("Notif subscription Success");
        });
      });
      this.client.on('error', error => {
          console.log('Connection failed', error)
      })
      this.client.on("reconnect", (error) => {
          console.log("reconnecting:", error);
          this.getMqttCredentialsForReconnection();
      });
      this.client.on('message', (topic, message) => {
        try {
          const splittedTopic = topic.split("/");
          const msgType = splittedTopic[3];
          if(msgType === "notif"){
            this.$notify({ type: 'danger', icon: 'tim-icons icon-alert-circle-exc', message:
message.toString()});
            this.$store.dispatch("getNotifications");

          }else if (msgType === "sdata"){

            $nuxt.$emit(topic, JSON.parse(message.toString()));
            return;

          }else if (msgType === "espcam"){
             $nuxt.$emit(topic, message);
             return;
           }
        } catch (error) {
          console.log(error);
        }
      });
      $nuxt.$on('mqtt-sender', (toSend) => {
        this.client.publish(toSend.topic, JSON.stringify(toSend.msg))
      });

    },
}
```

De igual manera, la vista default se comunica con el frontend por medio de las API's de método POST como son: "telemetriaex.com:3001/api1/getmqttcredentials" y "telemetriaex.com:3001/api1/getmqttcredentialsforreconnection" las mismas que harán llamadas a distintas funciones para crear credenciales, actualizar las y crear una identificación aleatoria, como se muestra en la siguiente código:

```
router.post("/getmqttcredentialsforreconnection", checkAuth, async (req, res) =>{
    try {
        const userId = req.userData._id;
        const credentials = await getWebUserMqttCredentialsForReconnection(userId);
        const toSend = {
            status: "success",
            username: credentials.username,
            password: credentials.password
```

```
        }
        res.json(toSend);
        setTimeout(() => {
            getWebUserMqttCredentials(userId);
        }, 15000);
    } catch (error) {
        console.log(error);
    }
})
async function getWebUserMqttCredentials(userId) {
    try {
        var rule = await EmqxAuthRule.find({ type: "user", userId: userId });
        if (rule.length == 0) {
            const newRule = {
                userId: userId,
                username: makeid(10),
                password: makeid(10),
                publish: [userId + "/#"],
                subscribe: [userId + "/#"],
                type: "user",
                time: Date.now(),
                updatedTime: Date.now()
            }
            const result = await EmqxAuthRule.create(newRule);
            const toReturn = {
                username: result.username,
                password: result.password
            }
            return toReturn;
        }
        const newUserName = makeid(10);
        const newPassword = makeid(10);
        const result = await EmqxAuthRule.updateOne({ type: "user", userId: userId }, { $set: { username:
newUserName, password: newPassword, updatedTime: Date.now() } });

        if (result.n == 1 && result.ok == 1) {
            return {
                username: newUserName,
                password: newPassword
            }
        } else {
            return false;
        }

    } catch (error) {
        console.log(error);
        return false;
    }}
async function getWebUserMqttCredentialsForReconnection(userId){
    try {
        const rule = await EmqxAuthRule.find({type: "user", userId: userId});
        if (rule.length == 1){
            const toReturn = {
                username: rule[0].username,
                password: rule[0].password
            }
            return toReturn;
        }
    } catch (error) {
        console.log(error);
        return false;
    }}
function makeid(length) {
    var result = '';
    var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    var charactersLength = characters.length;
    for (var i = 0; i < length; i++) {
        result += characters.charAt(Math.floor(Math.random() * charactersLength));
    }
    return result;
}
```

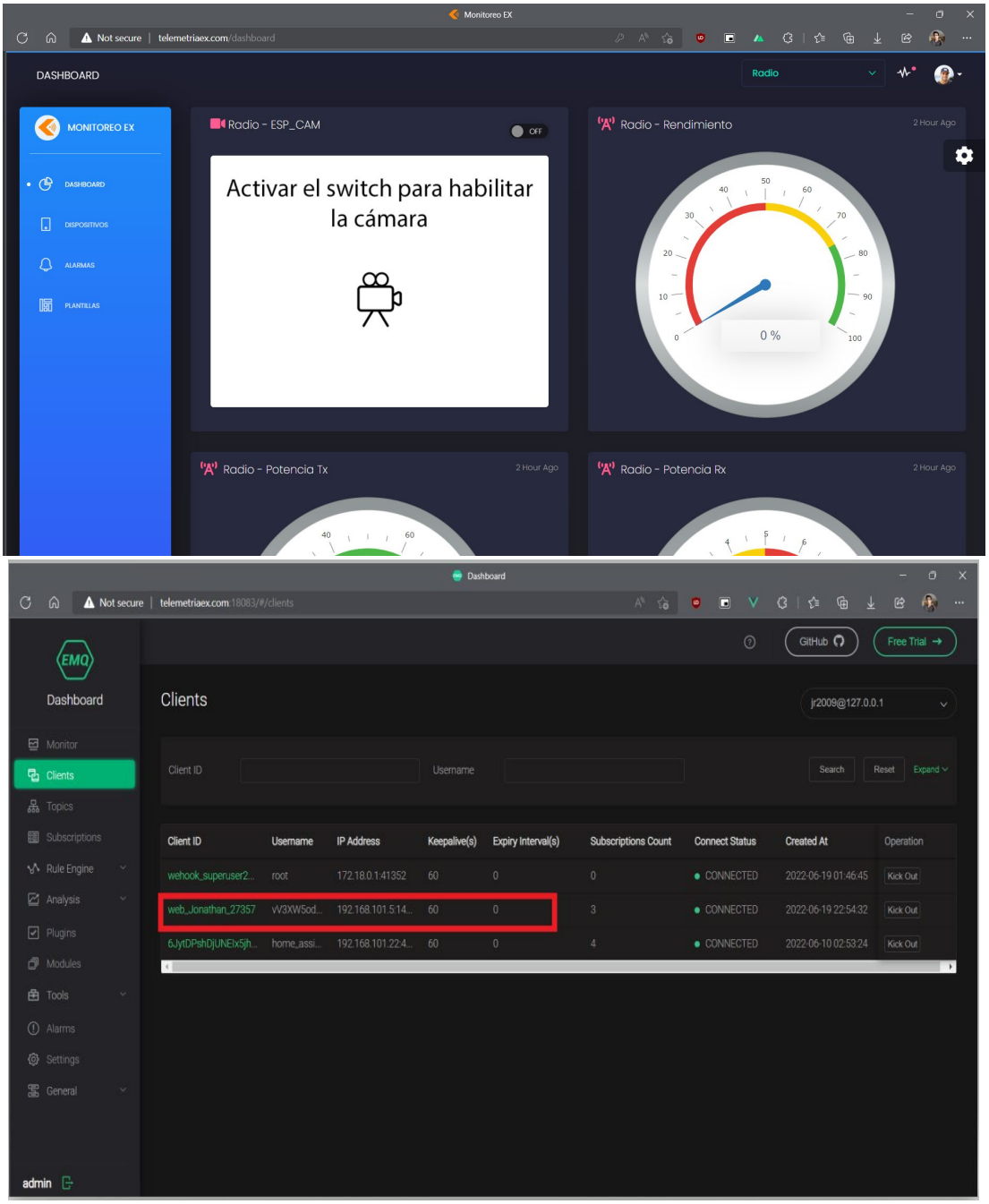


Figura 63: Diseño por defecto de la aplicación web y la verificación de que se ha creado un cliente MQTT correctamente.

Elaborado por: El investigador

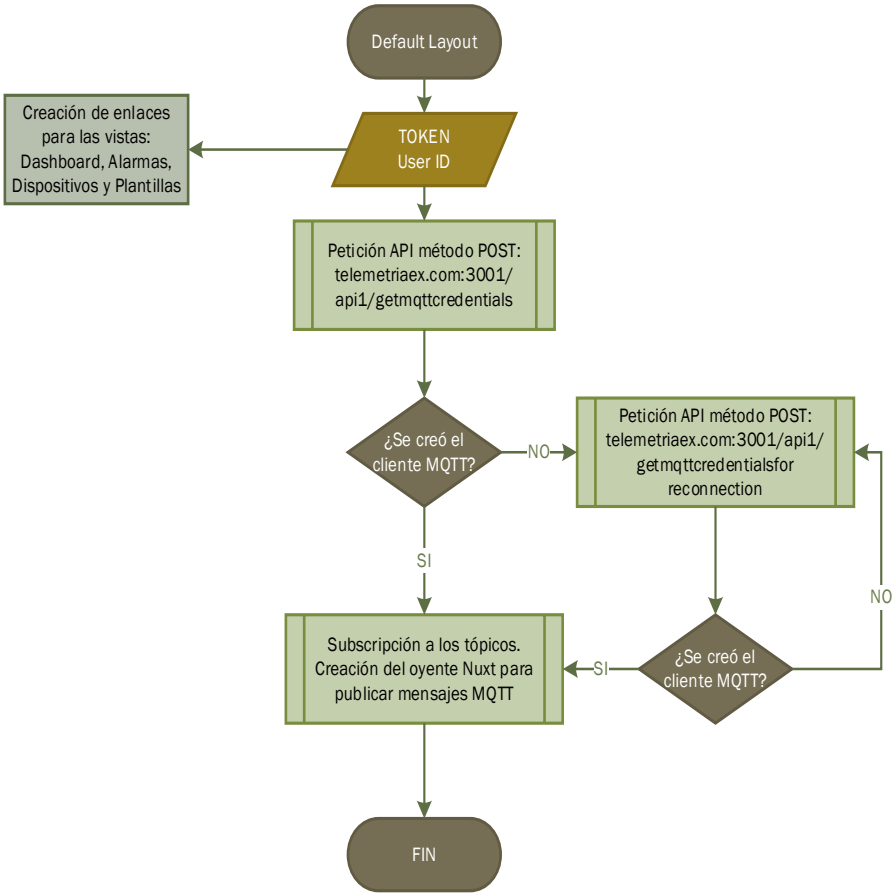


Figura 64: Diagrama de flujos de la vista default layout

Elaborado por: El investigador

La vista de dispositivos contiene un formulario en donde el usuario debe ingresar el nombre del dispositivo, un identificador y seleccionar una plantilla, la misma que está cargada de widgets o componentes como tablas, botones, indicadores booleanos, deslizadores, cajas para la presentación de video, entre otros. Además esta página muestra una lista de todos los dispositivos agregados, y debe crear funciones para guardar los datos de los dispositivos y eliminar los dispositivos, como se muestra en la siguiente figura y el diagrama de flujos correspondientes a la página dispositivos.

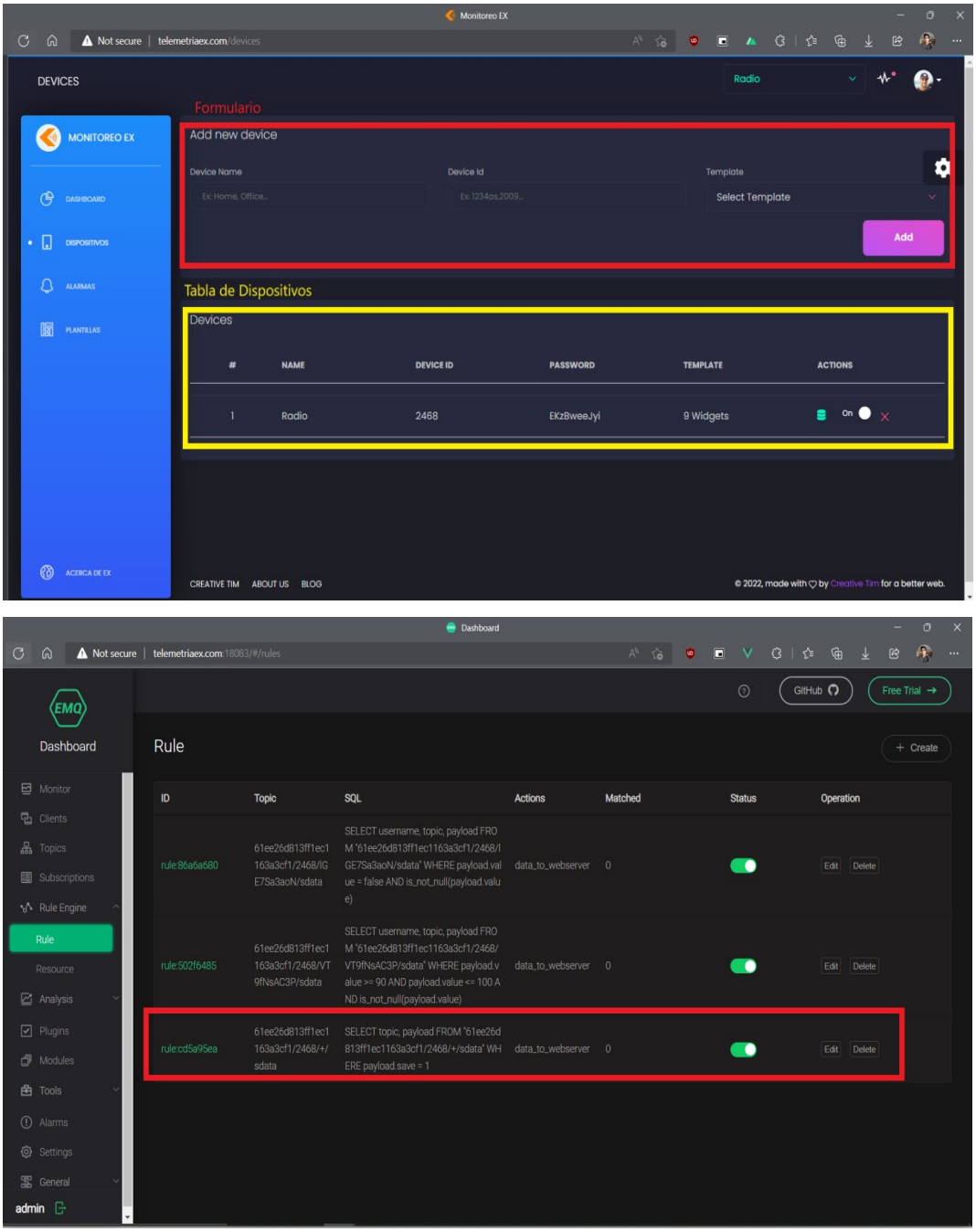


Figura 65: Vista de dispositivos y creación de una regla para guardar los datos de los dispositivos

Elaborado por: El investigador

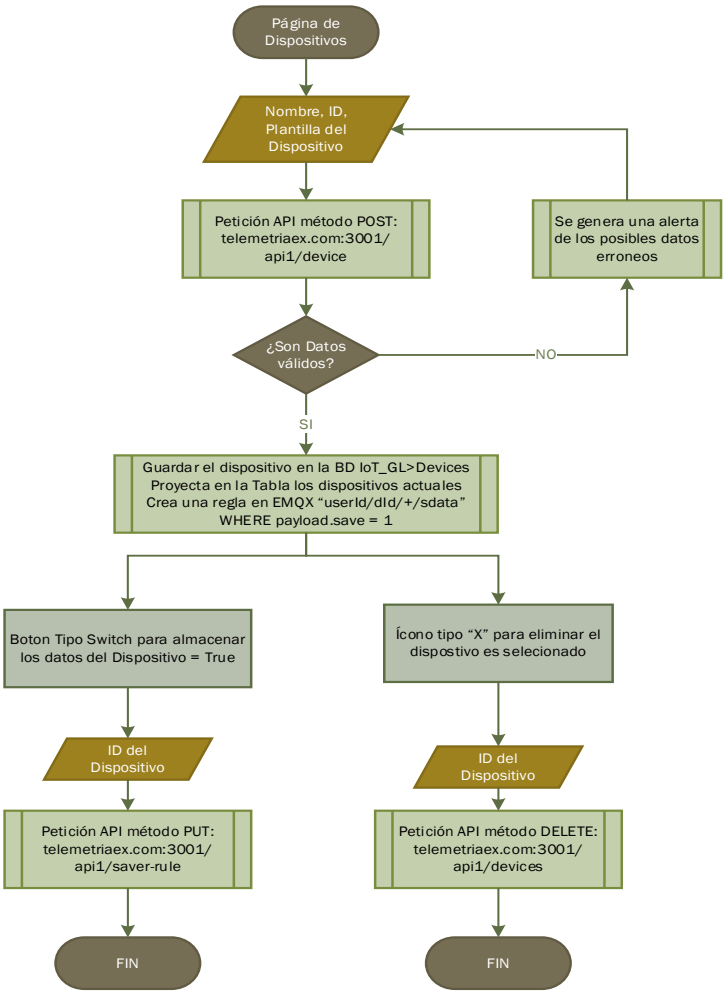


Figura 66: Vista de dispositivos

Elaborado por: El investigador

El código del frontend y el backend de la vista dispositivos se puede encontrar más detalladamente en el Anexo 1.

Por otro lado, para crear las plantillas del panel de control, se lo realiza en una nueva carpeta llamada "Widgets" del proyecto, y cada widget creado tiene su propio archivo de extensión ".vue", el sistema actualmente consta de 6 widgets creados y totalmente configurables por parte del usuario, estos widgets son:

- **Number Chart**.- tabla que muestra datos guardados en una base de datos y muestra el valor actual que pueda llegar por medio de NUXT, es utilizada para la visualización de sensores analógicos.

- **Boolean Indicator**.- indicador booleano, representa verdadero o falso, es utilizado para la representación de datos booleanos.
- **Camara**.- muestra la imagen enviada por la ESPCAM en tiempo real y cuenta con dos interruptores para activar y desactivar la cámara; y encender o apagar el led.
- **Switch Button** .- interruptor booleano, que permanece en un solo estado.
- Button.- botón booleano que permite enviar un mensaje de cualquier tipo a los dispositivos
- **Gauge Component**.- permite la visualización de datos numéricos de mejor manera, por medio de colores, y un puntero que indica su valor.

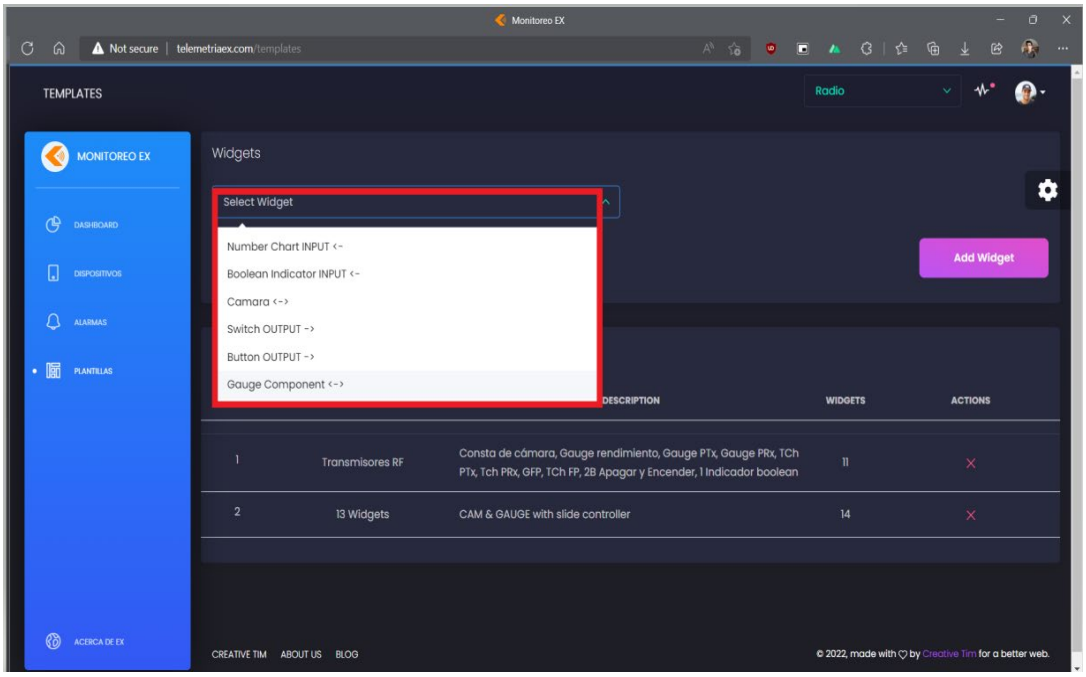


Figura 67: Vista de plantillas, en donde se pueden crear y configurar los 6 widgets del sistema

Elaborado por: El investigador

La creación del widget “Number Chart”, los datos ingresados por el usuario para la respectiva configuración del widget son:

Tabla 22: Parámetros de configuración del componente tipo tabla

Parámetros	**Tipo**	**Función**
Nombre de la Variable	Texto	Crea un nombre para la identificación de la variable
Unidades	Texto	Crea la unidad de medida de la variable a medir
Número de decimales	Números	Se define el número de decimales a mostrarse en la tabla.
Ícono	Texto	Configura el ícono identificador de la tabla, pueden ser cualquiera de Font Awesome 4 gratuitos.
Frecuencia de envió	Números	Se configura en segundos y es la frecuencia en la que los datos son enviados por MQTT
Datos en la tables desde	Números	Se configura en minutos y muestra en la tabla valores históricos a partir del tiempo configurado.
Tipo de estilo	Opciones	Define el color de la tabla: verde, purpura, naranja o rojo.
Tamaño	Opciones	Configura el tamaño de la tabla que puede ser de 3, 6, y hasta 12 (ancho completo).

Elaborado por: El investigador

Este widget se encuentra en la carpeta “components>Widgets” bajo el nombre de Rtnumberchart.vue, el cual consta de elementos como títulos de la variable, ícono, y una tabla, mientras que en la parte de JavaScript el algoritmo está diseñado para configurar la tabla con valores predeterminados en caso de no tenerlos, además se activa el oyente NUXT para escuchar cualquier mensaje MQTT que provenga de los dispositivos para su posterior procesamiento, realiza una petición API de tipo GET para obtener los valores de la tabla y posee un contador el cual mide el tiempo que se deja de recibir un mensaje, como se muestra en la siguiente figura:

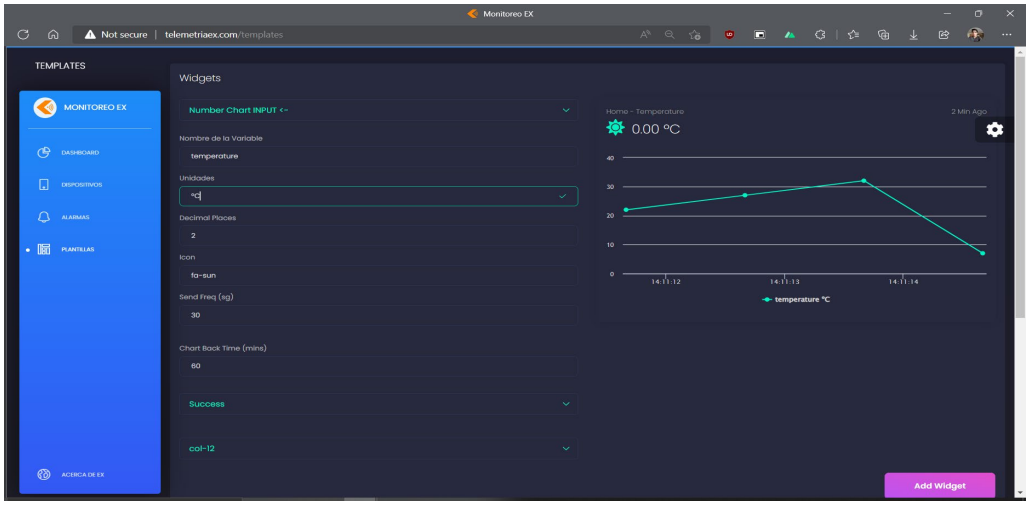


Figura 68: Formulario de configuración del componente tipo tabla

Elaborado por: El investigador

Este widget, puede ser representado mediante el siguiente flujograma, y su código puede encontrase en el Anexo 2.

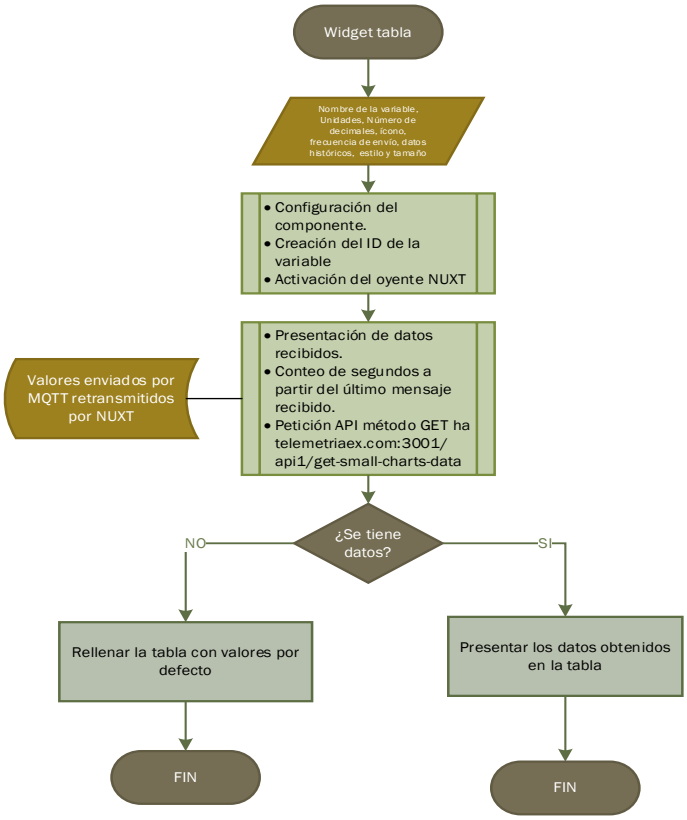


Figura 69: Diagrama de flujo del componente tabla

Elaborado por: El investigador

El componente denominado “Bolean Indicator”, permite el ingreso de los siguientes datos, para su respectiva configuración:

Tabla 23: Parámetros de configuración del componente tipo indicador booleano

Parámetros	**Tipo**	**Función**
Nombre de la Variable	Texto	Crea un nombre para la identificación de la variable
Ícono	Texto	Configura el ícono identificador de la tabla, pueden ser cualquiera de Font Awesome 4 gratuitos.
Frecuencia de envió	Números	Se configura en segundos y es la frecuencia en la que los datos son enviados por MQTT
Tipo de estilo	Opciones	Define el color de la tabla: verde, purpura, naranja o rojo.
Tamaño	Opciones	Configura el tamaño de la tabla que puede ser de 3, 6, y hasta 12 (ancho completo).

Elaborado por: El investigador

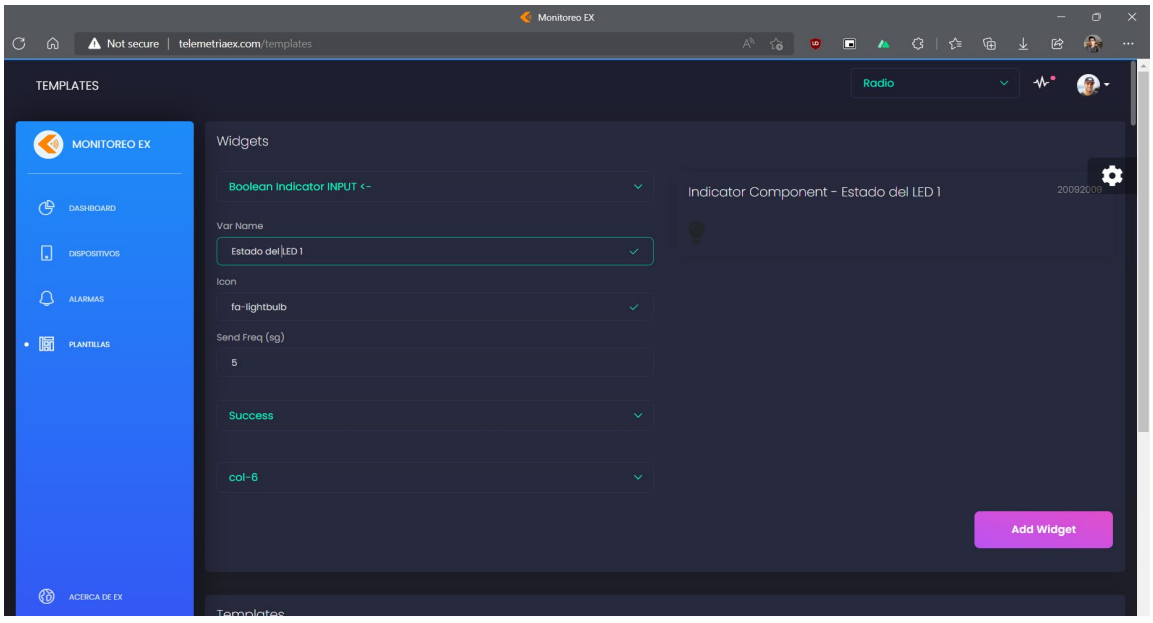


Figura 70: Formulario de configuración del componente tipo indicador booleano

Elaborado por: El investigador

Este tipo de widget no requiere de una petición API, su función es la de escuchar mensajes “True” o “False” enviados desde el dispositivo IoT, para ello es necesario crear un oyente

de NUXT que pueda recibir los mensajes, además se posee una función para cambiar de tono cuando el mensaje recibido sea un “true” y lo apaga en el caso contrario, se puede representar este widget de la siguiente manera, mientras que el código puede encontrarse en el Anexo 3.

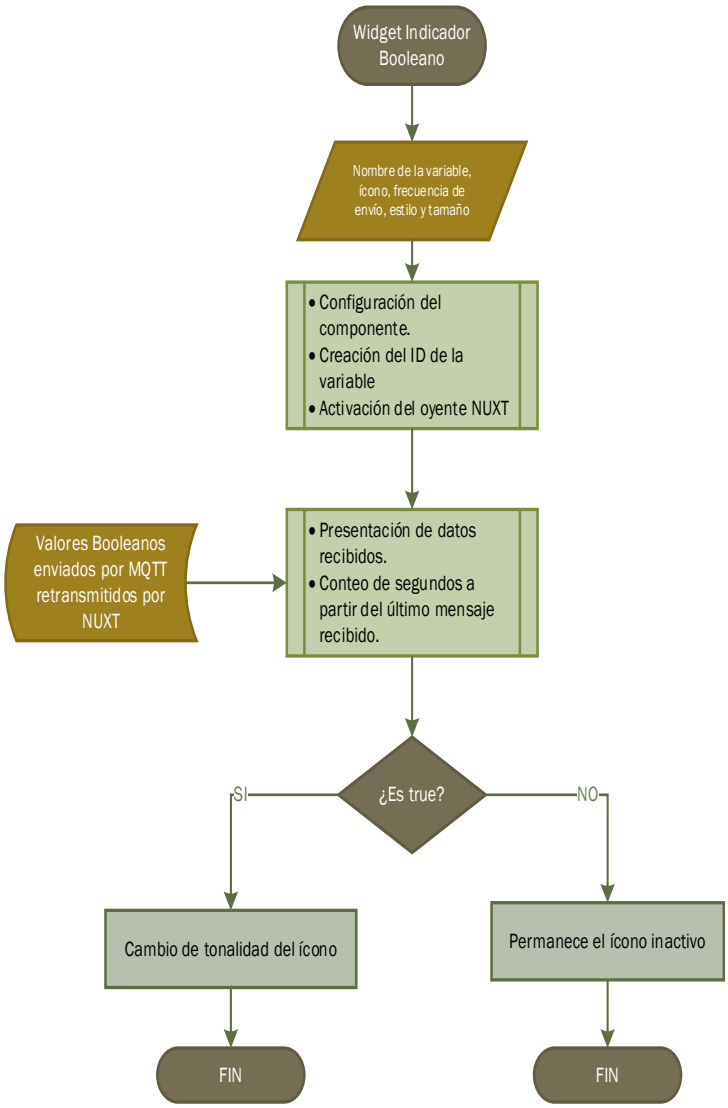


Figura 71: Diagrama de flujo del componente indicador booleano

Elaborado por: El investigador

El widget tipo cámara es un componente bidireccional, es decir que es capaz de recibir la información y de enviarla, consta de un cuadro en donde se proyectara las imágenes captadas por el dispositivos y dos interruptores para la activación y desactivación de la cámara y la del estado del led flash.

Tabla 24: Parámetros de configuración del componente tipo cámara

Parámetros	**Tipo**	**Función**
Nombre de la Variable	Texto	Crea un nombre para la identificación de la variable
Ícono	Texto	Configura el ícono identificador de la tabla, pueden ser cualquiera de Font Awesome 4 gratuitos.
Tipo de estilo	Opciones	Define el color de la tabla: verde, purpura, naranja o rojo.
Tamaño	Opciones	Configura el tamaño de la tabla que puede ser de 6 y 12 (ancho completo).

Elaborado por: El investigador

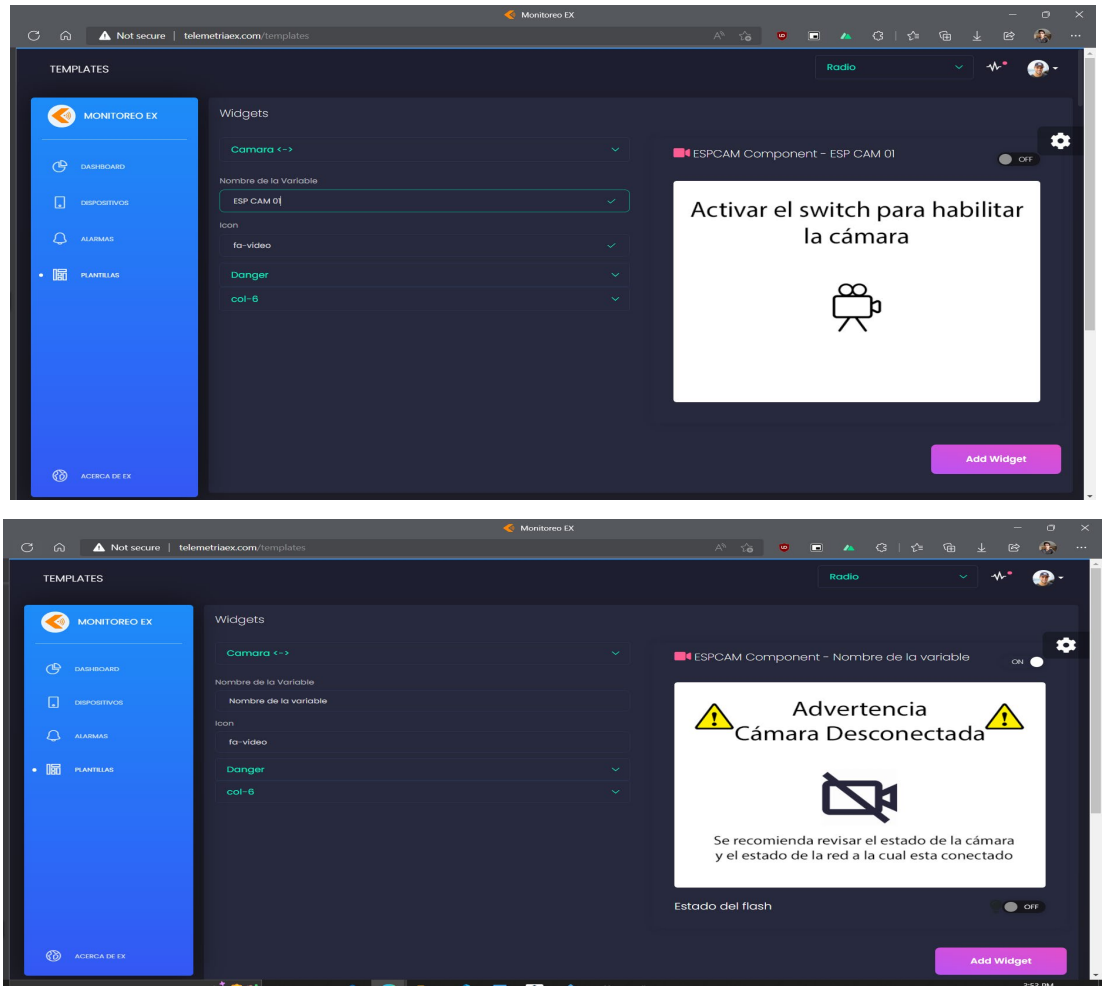


Figura 72: Formulario de configuración del componente tipo cámara y sus respectivos fotos de estado

Elaborado por: El investigador

Este componente se encarga de transformar los datos binarios enviados por ESPCAM y transformarlos a formato jpg, si el componente esta activado y este no percibe datos de entrada, el componente muestra una imagen de advertencia y sugerencias del posible fallo. Cuando la cámara este activada se habilitará un interruptor para encender y apagar el led flash, como se demuestra en el siguiente diagrama de flujos, y además su código puede encontrarse en el Anexo 4.

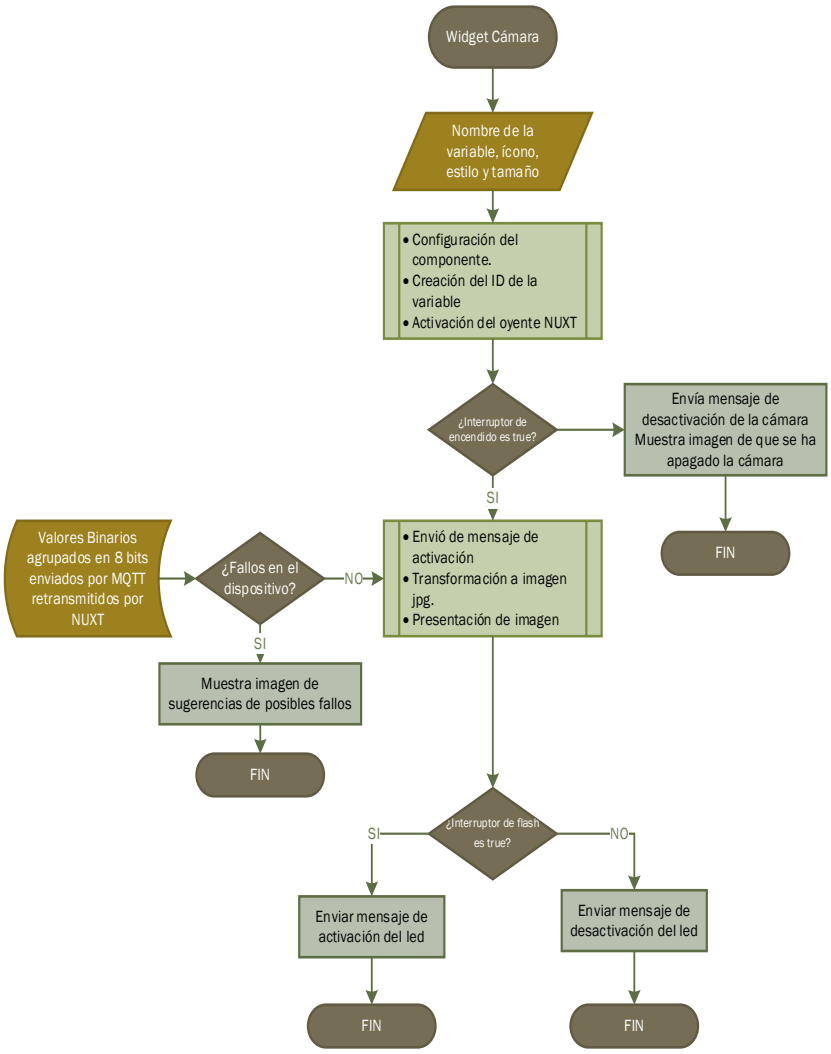


Figura 73: Diagrama de flujo del componente cámara

Elaborado por: El investigador

Por otro lado, el componente "switch" es un componente unidireccional, el cual envía un valor booleano hacia los dispositivos IoT, para su configuración el usuario debera ingresar los siguientes datos:

Tabla 25: Parámetros de configuración del componente tipo interruptor

Parámetros	**Tipo**	**Función**
Nombre de la Variable	Texto	Crea un nombre para la identificación de la variable
Ícono	Texto	Configura el ícono identificador de la tabla, pueden ser cualquiera de Font Awesome 4 gratuitos.
Tipo de estilo	Opciones	Define el color de la tabla: verde, purpura, naranja o rojo.
Tamaño	Opciones	Configura el tamaño de la tabla que puede ser de 3, 6, y hasta 12 (ancho completo).

Elaborado por: El investigador

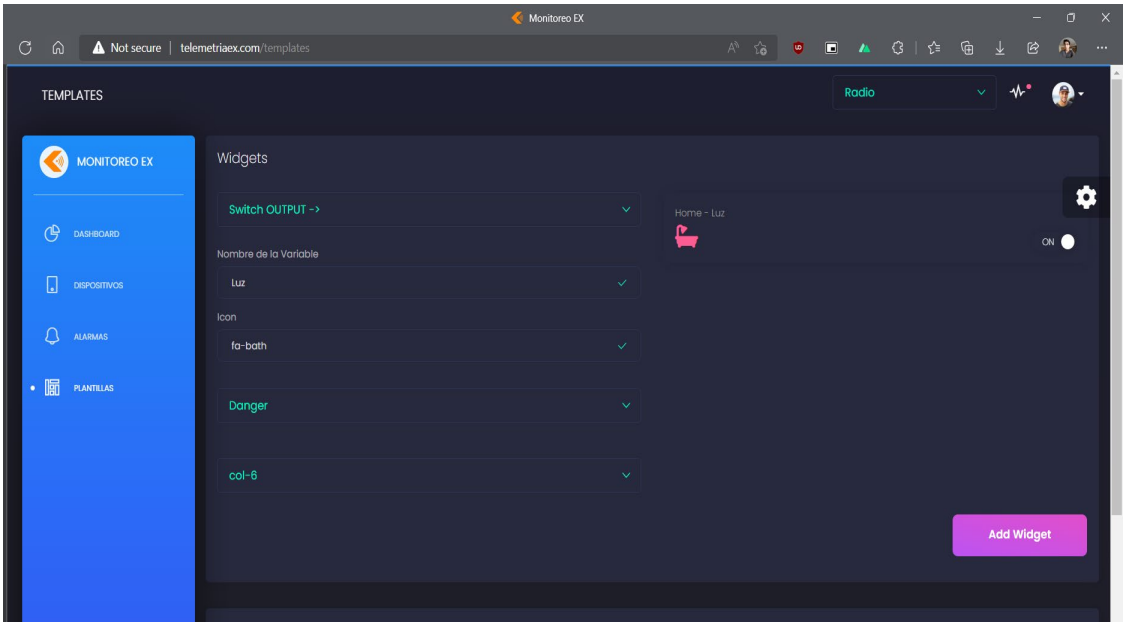


Figura 74: Formulario para la configuración del componente tipo interruptor

Elaborado por: El investigador

Mientras que el código (Anexo 5) se basa en enviar mensajes booleanos por NUXT para luego ser retransmitidos por medio de MQTT hacia los dispositivos y cambiar el color del ícono en función del estado del interruptor, como se explica en el siguiente esquema:

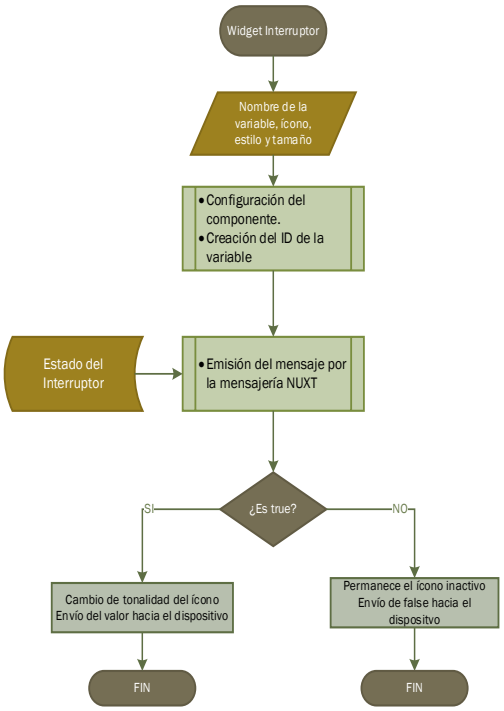


Figura 75: Diagrama de flujo del componente interruptor
Elaborado por: El investigador

El componente tipo "Button", permite enviar mensajes hacia el dispositivos mensajes tipo texto, en donde el dispositivo se encarga de interpretarlos, la configuración de este dispositivo esta dado por las siguientes variables:

Tabla 26: Parámetros de configuración del componente tipo botón

Parámetros	**Tipo**	**Función**
Nombre de la Variable	Texto	Crea un nombre para la identificación de la variable
Mensaje para enviar	Texto	Se define un mensaje tipo texto a enviar, puede ser además
Título	Texto	Nombre del botón, ejemplo: Encender, Apagar
Ícono	Texto	Configura el ícono identificador de la tabla, pueden ser cualquiera de Font Awesome 4 gratuitos.
Tipo de estilo	Opción	Define el color de la tabla: verde, purpura, naranja o rojo.
Tamaño	Opción	Configura el tamaño de la tabla que puede ser de 3, 6, y hasta 12 (ancho completo).

Elaborado por: El investigador

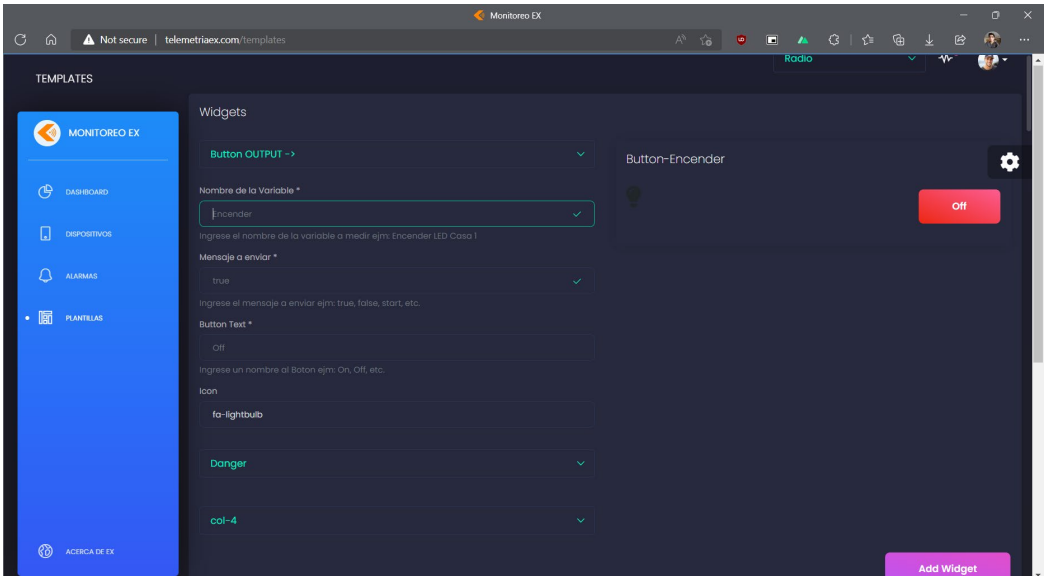


Figura 76: Formulario para la configuración del componente tipo botón

Elaborado por: El investigador

En lo que respecta funciones (Anexo 6) usadas por este componente, es capaz de cambiar la tonalidad del ícono, cuando este sea pulsado y enviar por medio de NUXT el mensaje configurado para su respectiva retransmisión, como se muestra en el siguiente flujograma:

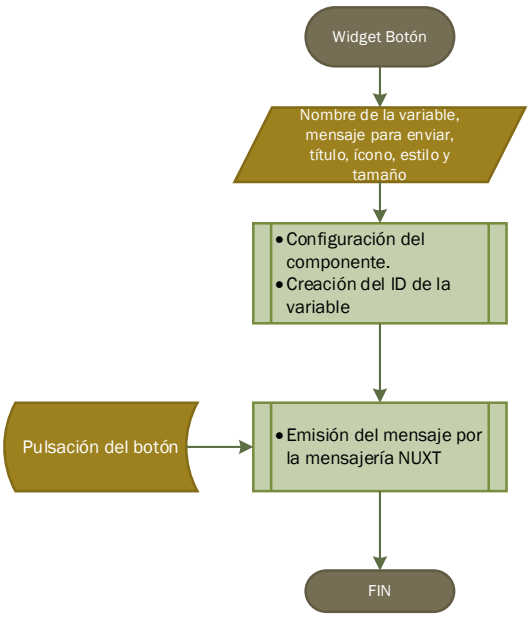


Figura 77: Diagrama de flujo del componente botón

Elaborado por: El investigador

Por último el widget “gauge indicator” es un componente de comunicación bidireccional que permite visualizar datos numéricos y por medio de un deslizador enviar de igual manera valores numéricos para que el microcontrolador pueda interpretarlos, para su correcto funcionamiento el usuario debe ingresar los siguientes parámetros:

Tabla 27: Parámetros de configuración del componente tipo indicador gauge

Parámetros	**Tipo**	**Función**
Nombre de la Variable	Texto	Crea un nombre para la identificación de la variable
Unidades	Texto	Crea la unidad de medida de la variable a medir
Valor máximo que medir	Número	Configura el límite máximo a medir del componente
Limites franja verde	Número	Requiere de un valor mínimo y máximo para configurar una segmento de color verde
Limites franja amarillo	Número	Requiere de un valor mínimo y máximo para configurar una segmento de color amarillo.
Limites franja rojo	Número	Requiere de un valor mínimo y máximo para configurar una segmento de color rojo.
Número de decimales	Número	Se define el número de decimales a mostrarse en la tabla.
Ícono	Texto	Configura el ícono identificador de la tabla, pueden ser cualquiera de Font Awesome 4 gratuitos.
Etiquetas numéricas	Número	Define el espacio que debe haber entre cada etiqueta numérica.
Líneas pequeñas	Número	Establece la separación entre líneas pequeñas en el componente.
Slider	Opción	Marcar para que aparezca el deslizador.
Valor máximo por enviar	Número	Configuración para el deslizador y determina el máximo valor entero a enviar, así como el mínimo.

Pasos de los botones	Número	Configuración para los botones del deslizador, el cual determina el valor entero a enviar.
Frecuencia de envió	Número	Se configura en segundos y es la frecuencia en la que los datos son enviados por MQTT
Tipo de estilo	Opción	Define el color de la tabla: verde, purpura, naranja o rojo.

Elaborado por: El investigador

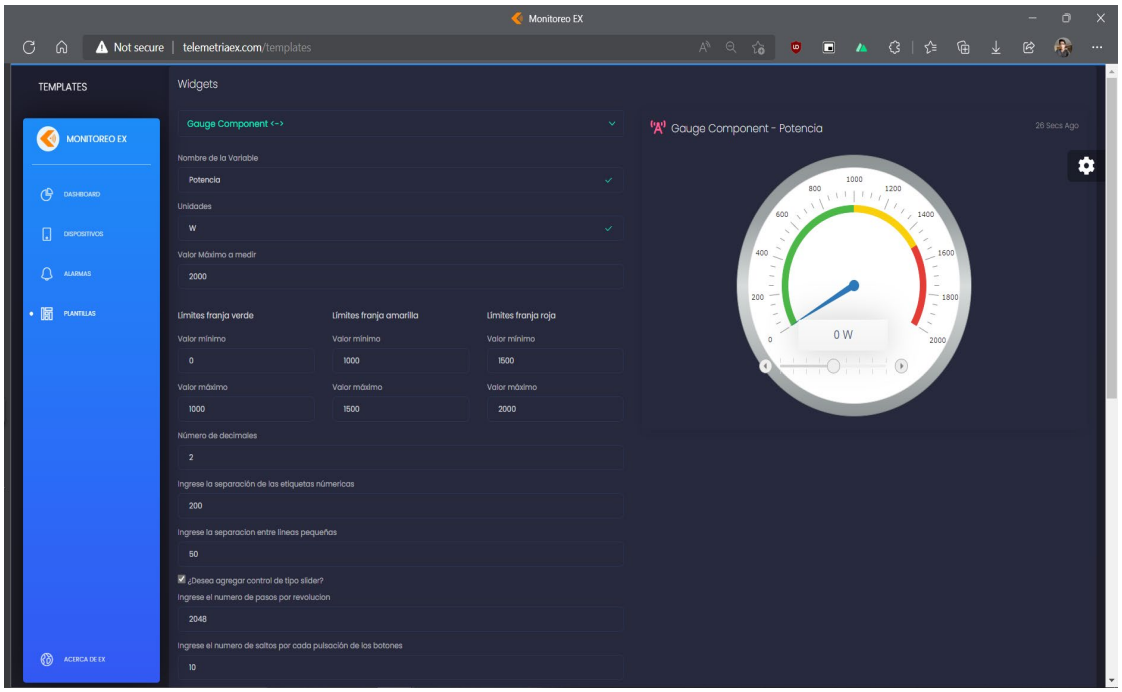


Figura 78: Formulario para la configuración del componente tipo indicador gauge

Elaborado por: El investigador

La funciones necesarias para que este widget pueda trabajar (Anexo 7) consta en crear un oyente NUXT y emitir mensajes con valores numéricos enteros a en función de la posición del slider, el proceso de funcionamiento de este componente se puede representar mediante el siguiente diagrama de flujo:

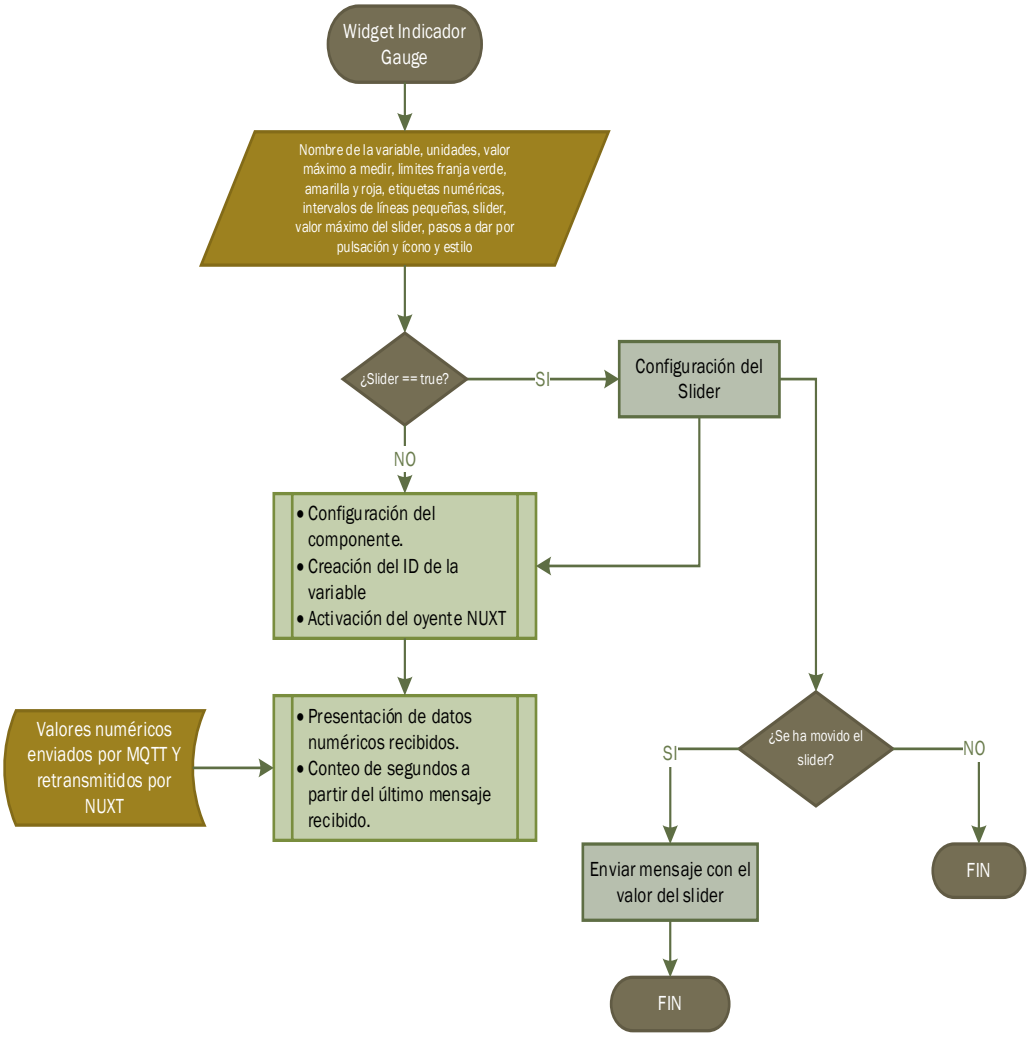


Figura 79: Diagrama de flujo del componente indicador gauge

Elaborado por: El investigador

Los widgets mencionados deben ser llamados por la vista plantillas, el cual consta de un formulario en función del componente seleccionado, para que el usuario configure los widgets acorde a sus necesidades, además una vez configurados y agregados, se puede observar una previsualización de los componentes agregados, y para guardarlos se requiere de un nombre y una descripción; mediante una tabla se puede observar la plantilla junto a las demás plantillas creadas previamente, esta tabla cuenta además con una opción para poder eliminarlos de la base de datos, como se muestra en la siguiente figura:

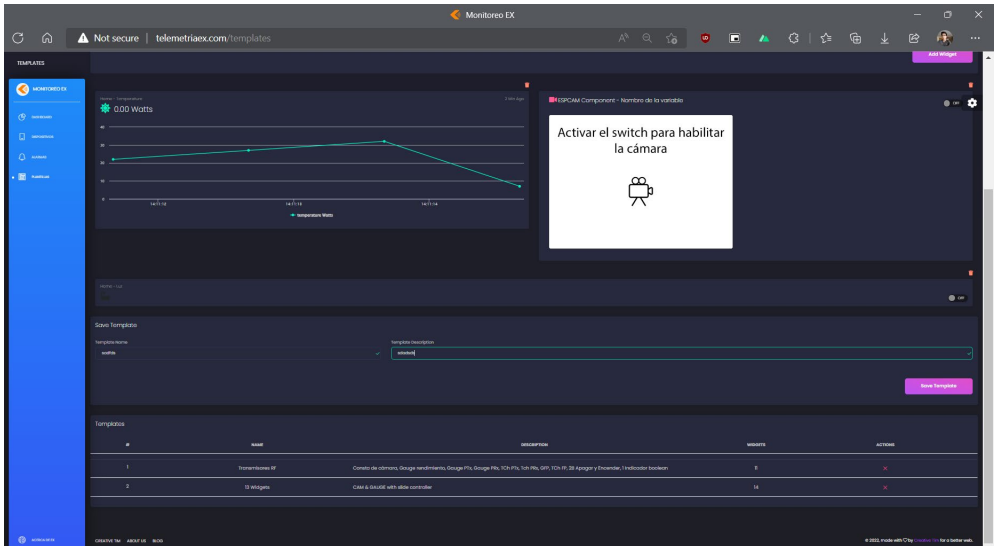


Figura 80: Vista plantillas para gestionar los distintos componentes creados.

Elaborado por: El investigador

Por último, al crear las plantillas y asociarlas a los dispositivos, la página de panel de control es una representación de la plantilla creada, en donde se encarga de renderizar cada componente acorde a los parámetros definidos por el usuario.

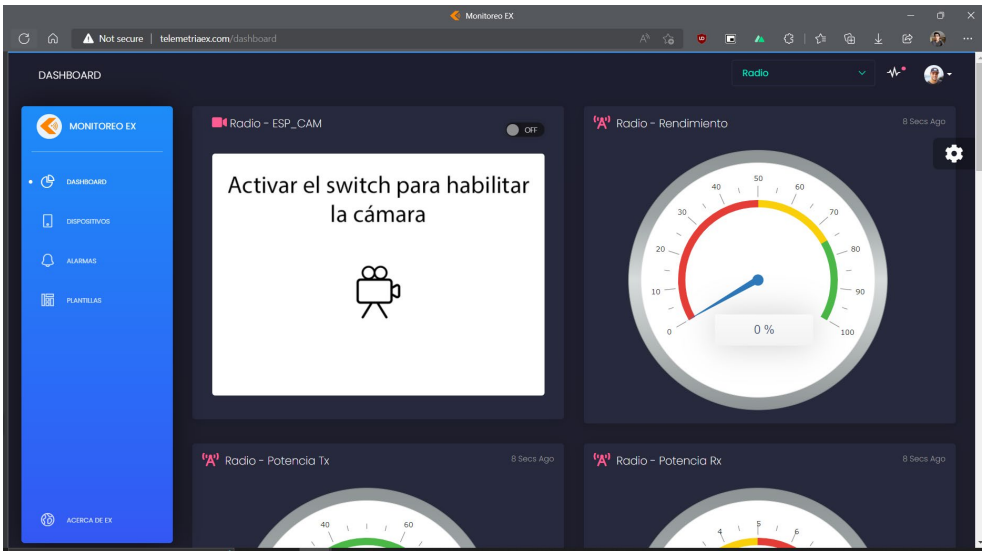


Figura 81: Página de panel de control sobre los dispositivos.

Elaborado por: El investigador

La siguiente vista a realizar es la vista de alarmas, la cual permite crear reglas a variables las cuales son activadas en función de una condición numérica o booleana, la vista está

compuesta de un formulario en donde se selecciona la variable, la condición numérica o booleana, el valor y el intervalo de tiempo a enviar las notificaciones, las mismas que son enviadas al Frontend y al número de Whatsapp registrado por el usuario. Además se tiene una tabla en donde se muestran las reglas creadas junto a sus parámetros de configuración y la posibilidad de desactivarlos o activarlos e inclusive eliminarlos, como se muestra en la siguiente figura.

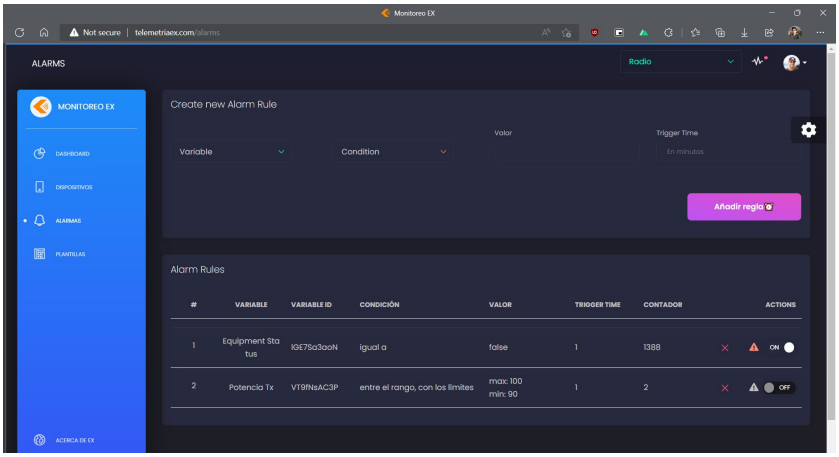


Figura 82: Vista alarmas para crear y gestionar las distintas reglas sobre las variables.

Elaborado por: El investigador

La vista consta de distintas funciones para comunicase con la API "telemetriaex.com:3001/api1/alarm-rule" haciendo uso de diferentes métodos, como: el método "delete" para la eliminación de las reglas, el método POST para crear nuevas alarmas, el método PUT para actualizar el estado de las alarmas, esa API de igual manera realizara una petición API hacia el servicio de EQMX "http://telemetiraex.com:8085/api/v4/rules" de igual manera con los mismos métodos agregando como parámetro la regla ID creada.

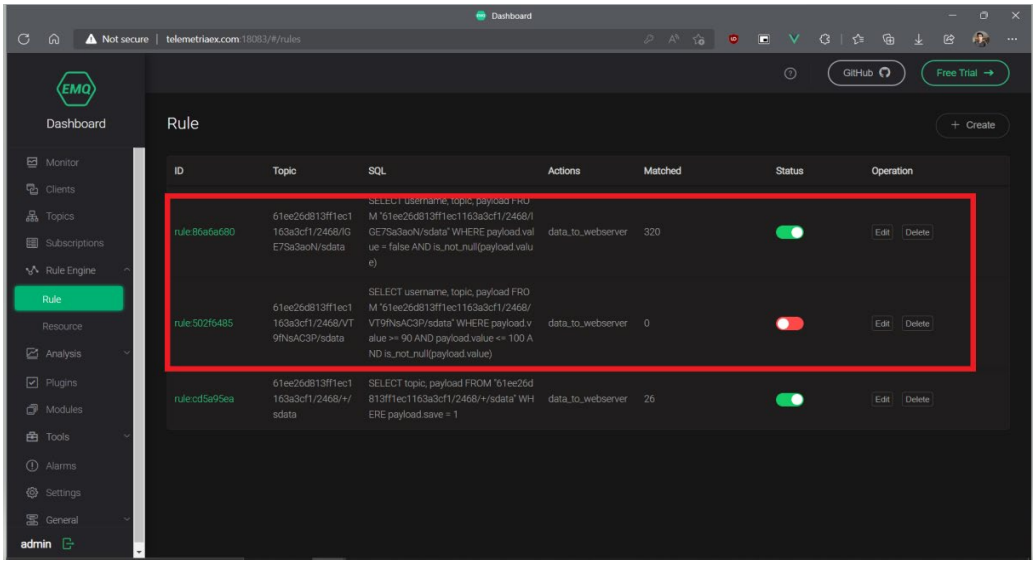


Figura 83: Verificación de las reglas creadas en la página web en el servicio de EMQX

Elaborado por: El investigador

Este proceso se puede representar mediante el siguiente flujograma:

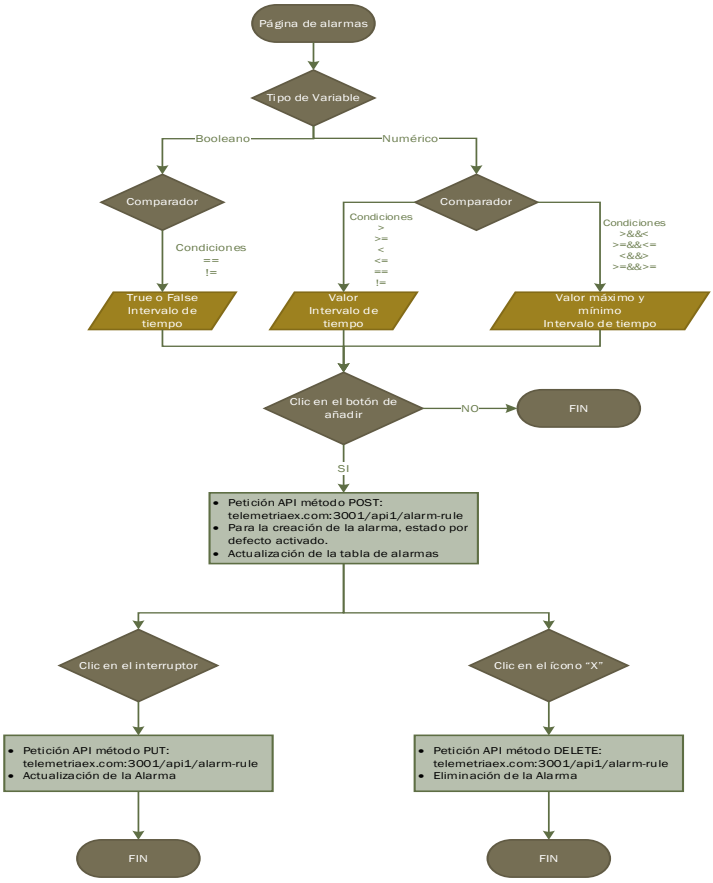


Figura 84: Diagrama de flujo de la vista alarmas

Elaborado por: El investigador

Por último y no menos importante, se requirió crear una API (Anexo 8) para que las alertas generadas por el servicio lleguen a notificarle al usuario final por medio de Whatsapp, para ello, la API debe ser capaz de procesar la información enviada por el servicio EMQX y retransmitirlo de manera legible hacia el o los números asociados al dueño de la cuenta, como se muestra en el siguiente flujograma:

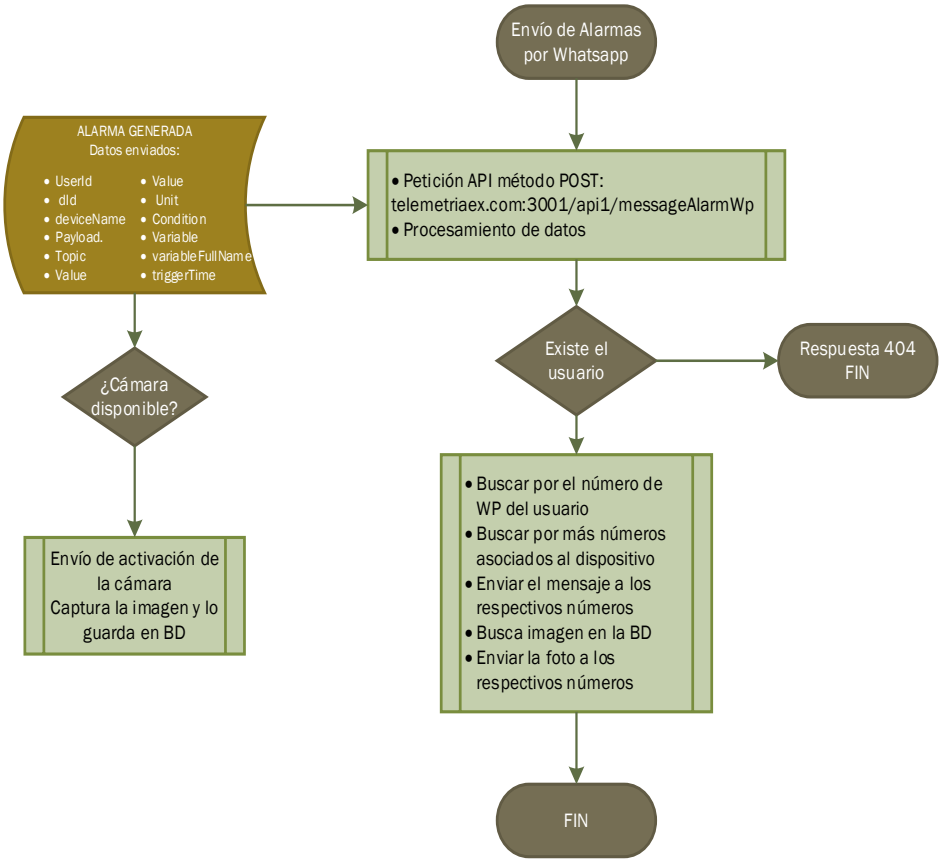


Figura 85: Diagrama de flujo de alertas enviadas por Whatsapp

Elaborado por: El investigador

El funcionamiento de las alarmas se demuestra en la Figura 86, en donde se muestra que las alarmas generadas por el dispositivo le notifican al usuario de la cuenta, mientras que la Figura 87, por medio de un chatbot detallado en la Figura 85, el usuario de la cuenta es capaz de agregar más números y por ende alertarlos de la misma manera.

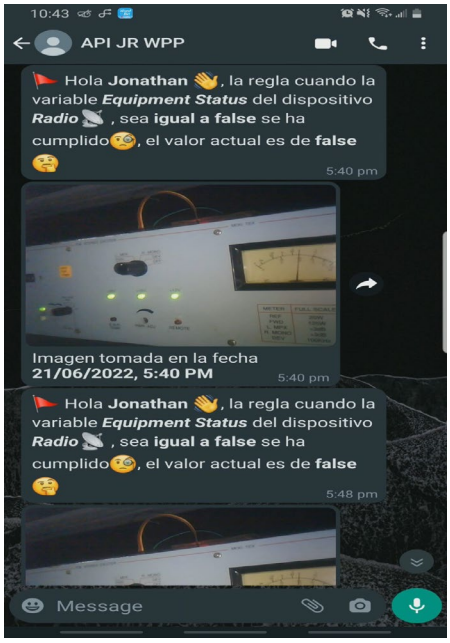


Figura 86: Mensaje generado y enviado por Whatsapp cuando una regla se haya cumplido

Elaborado por: El investigador

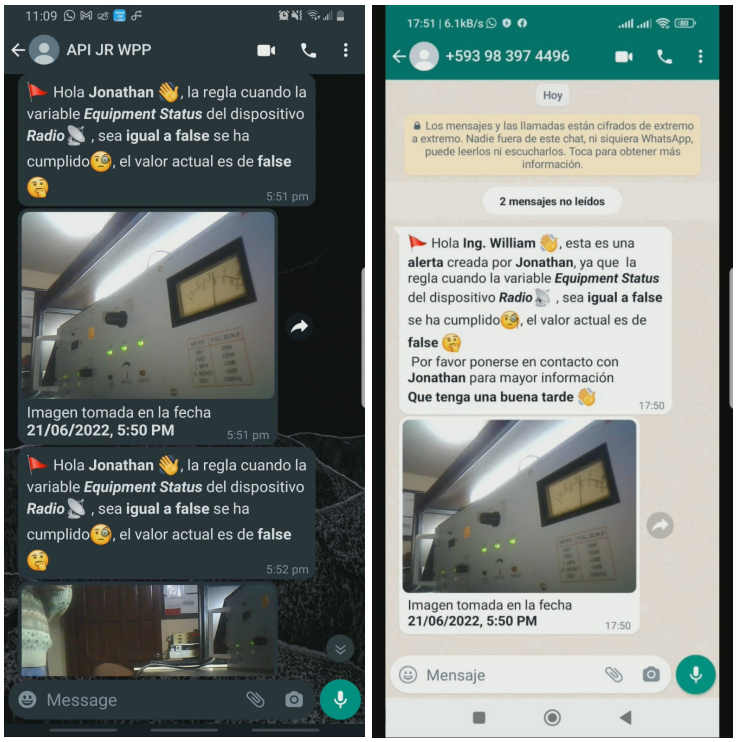


Figura 87: Envió de alarmas generadas por el equipo al usuario del sistema y a los contactos asociados por Whatsapp

Elaborado por: El investigador

Finalmente, la creación del chatbot, es capaz de preguntar al usuario propietario de la cuenta por más números, los mismos que pueden ser asociados a distintos dispositivos que el administrador gestione, para ello es necesario crear un archivo en "api/routes/wp.js" en donde el código puede examinarse en el (Anexo 8), y su funcionamiento puede ser expresado de la siguiente manera:

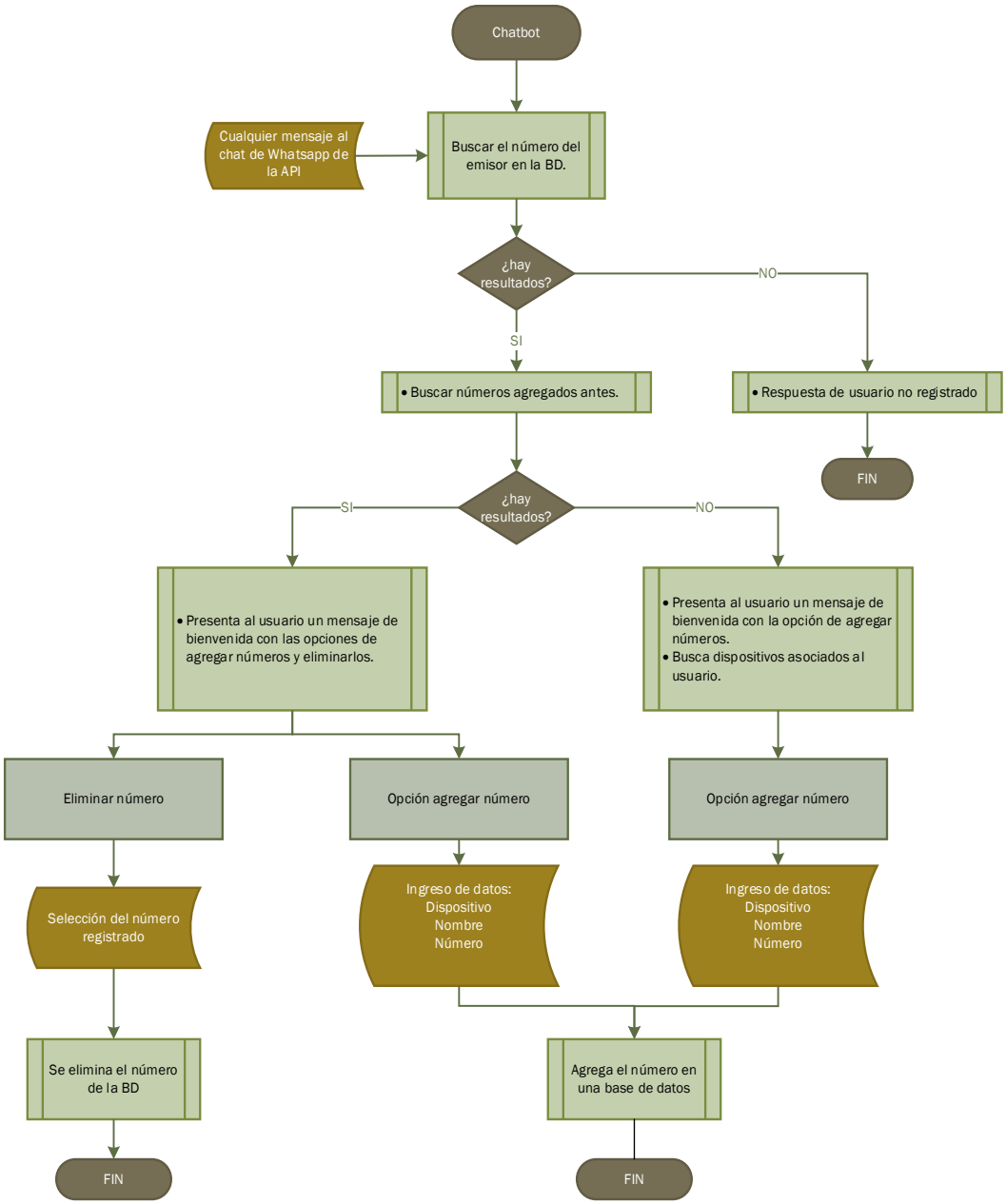


Figura 88: Diagrama de flujo del chatbot de Whatsapp

Elaborado por: El investigador

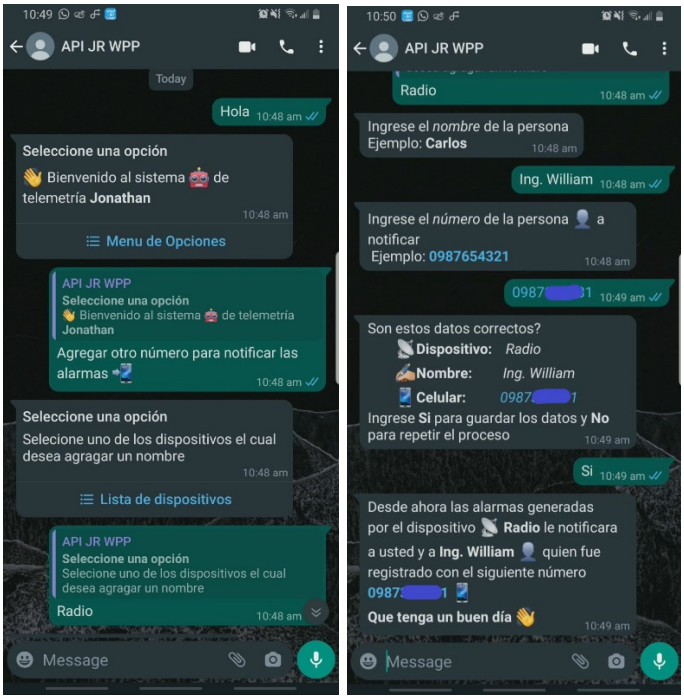


Figura 89: Chatbot para el registro de otras personas a ser notificadas por las alarmas generadas por el dispositivo.

Elaborado por: El investigador

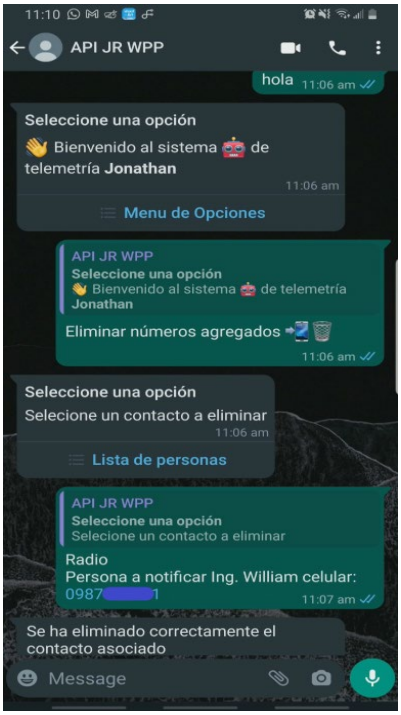


Figura 90: Eliminación del contacto asociado

Elaborado por: El investigador

Figura 91: Mensaje de respuesta ante un usuario no registrado

Elaborado por: El investigador

3.2.7. Programación del dispositivo IoT.

El dispositivo IoT debe ser capaz de acondicionar los parámetros operativos del transmisor RF como son:

- Tensión de alimentación AC [V_{AC}].
- Corriente de alimentacion AC [I_{AC}].
- Potencia activa eléctrica consumida [W].
- Factor de potencia del equipo [%]
- Potencia transmitida [W].
- Potencia reflejada [W].

La mayoría de estos parámetros se pueden encontrar en la parte posterior del equipo, que en este caso es un transmisor de la marca "RVR TEX100" (Anexo 9), ya que cuenta con un terminal de telemetría, este puerto maneja muchos parámetros como la potencia de transmisión, potencia reflejada, SWR, temperatura, voltaje y corriente en los amplificadores, audio, entre otros. Dichos valores lo representan en función del voltaje DC con valores máximos de +12V y mínimos de +1V, por lo que se requiere acondicionar cada una de las señales, como se puede observar en el Anexo 10, obteniendo así los siguientes resultados:

Tabla 28: Relación entre la potencia transmitida y el voltaje DC

Ptx[W]	Vdc 12V	Vdc 3V
2.00	1.13	0.257
10.00	2.99	0.665
20.00	4.29	0.978
30.00	5.37	1.230
40.00	6.40	1.456
50.00	7.26	1.663
60.00	8.01	1.842
70.00	8.70	1.992
80.00	9.32	2.142
88.00	9.76	2.236

Elaborado por: El investigador

Tabla 29: Relación entre la potencia reflejada y el voltaje DC

Prx[W]	Vdc 12V	Vdc 3V
1.50	1.29	0.28
3.50	4.47	0.96
6.00	5.61	1.21
7.00	6.70	1.44
8.00	6.80	1.46
9.00	7.34	1.58
10.00	7.90	1.70

Elaborado por: El investigador

Como se mencionó previamente, se optó por utilizar el módulo ADS1115 que es un convertidor análogo digital de 15 bits y dado que el controlador ESP32 maneja lógica de 3.3 bits, es por ello por lo que se procede, a realizar un divisor de voltaje, que reduzca la muestra de voltaje al menos el 22%, en este caso se utilizó un divisor de resistencias de $2.2K\Omega$ y $7.9K\Omega$, por lo que se procedió a realizar el cálculo para normalizar este voltaje a valores inferiores a 3.3V.

Por último se requirió linealizar los datos, cabe recalcar que estos datos no son proporcionales y por ende se realizó una regresión polinomial para encontrar la ecuación de cada una de las potencias, obteniendo así las siguientes formulas.

Al representar los datos se pudo observar que corresponde a una ecuación de segundo grado, por lo que se procedió a aplicar una suma de cuadrados para obtener los coeficientes de la ecuación cuadrática.

$$a_0 n + a_1 \sum x_i + a_2 \sum x_i^2 = \sum y_i$$
$$a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 = \sum x_i \cdot y_i$$
$$a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 = \sum x_i^2 \cdot y_i$$

Ecuación 1: Sistema de ecuaciones para la obtención de los coeficientes de la ecuación cuadrática.

Obteniendo así el siguiente procedimiento para obtener la ecuación cuadrática de la potencia transmitida.

Tabla 30: Obtención de los coeficientes del sistema de ecuaciones para el cálculo de la potencia transmitida

X_i	Y_i	x_i^2	x_i^3	x_i^4	$X_i \cdot Y_i$	$X_i^2 \cdot Y_i$
0.257	2.00	0.07	0.02	0.00	0.51	0.13
0.665	10.00	0.44	0.29	0.20	6.65	4.42
0.978	20.00	0.96	0.94	0.91	19.56	19.13
1.230	30.00	1.51	1.86	2.29	36.90	45.39
1.456	40.00	2.12	3.09	4.49	58.24	84.80
1.663	50.00	2.77	4.60	7.65	83.15	138.28
1.842	60.00	3.39	6.25	11.51	110.52	203.58
1.992	70.00	3.97	7.90	15.75	139.44	277.76
2.142	80.00	4.59	9.83	21.05	171.36	367.05
2.236	88.00	5.00	11.18	25.00	196.77	439.97

$\sum X_i$	$\sum Y_i$	$\sum x_i^2$	$\sum x_i^3$	$\sum x_i^4$	$\sum X_i \cdot Y_i$	$\sum X_i^2 \cdot Y_i$
14.46	450.00	24.81	45.95	88.85	823.10	1580.52

Elaborado por: El investigador

Reemplazando valores se tiene el siguiente sistema de ecuaciones, que al resolverla se obtiene la siguiente ecuación:

$$\begin{cases} 10a_0 + 14.46a_1 + 24.81a_2 = 450 \\ 14.46a_0 + 24.81a_1 + 45.95a_2 = 823.10 \\ 24.81a_0 + 45.95a_1 + 88.85a_2 = 1580.52 \end{cases}$$

$$\begin{cases} a_0 = -0.43 \\ a_1 = 6.12 \\ a_2 = 14.73 \end{cases} \quad \therefore \boldsymbol{f(x) = -0.43 + 6.12x + 14.73x^2}$$

Ecuación 2: Ecuación cuadrática para la estimación de la potencia transmitida

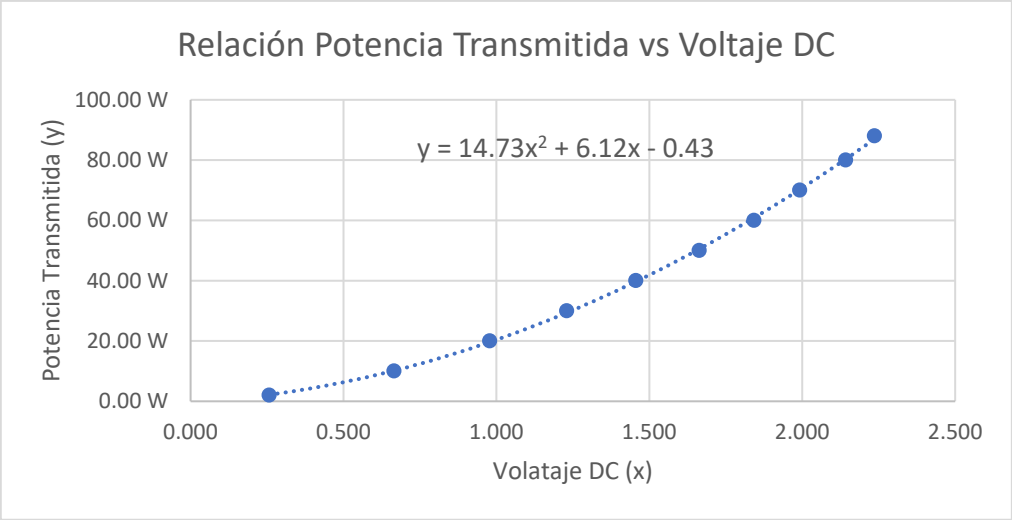


Figura 92: Representación de los datos de la Tabla 28 con su respectiva función cuadrática de regresión

Elaborado por: El investigador

Por otro lado, se procede a realizar el mismo procedimiento para la obtención de la función cuadrática que permita obtener los valores de la potencia reflejada.

Tabla 31: Obtención de los coeficientes del sistema de ecuaciones para el cálculo de la potencia transmitida

X_i	Y_i	x_i^2	x_i^3	x_i^4	$X_i \cdot Y_i$	$X_i^2 \cdot Y_i$
0.277	1.50	0.08	0.02	0.01	0.42	0.12
0.961	3.50	0.92	0.89	0.85	3.36	3.23
1.206	6.00	1.45	1.75	2.12	7.24	8.73
1.441	7.00	2.08	2.99	4.31	10.08	14.53
1.462	8.00	2.14	3.12	4.57	11.70	17.10
1.578	9.00	2.49	3.93	6.20	14.20	22.41
1.699	10.00	2.88	4.90	8.32	16.99	28.85

$\sum X_i$	$\sum Y_i$	$\sum x_i^2$	$\sum x_i^3$	$\sum x_i^4$	$\sum X_i \cdot Y_i$	$\sum X_i^2 \cdot Y_i$
8.62	45.00	12.04	17.61	26.37	63.98	94.96

Elaborado por: El investigador

Reemplazando valores en la Ecuación 1, se obtiene el siguiente sistema de ecuaciones, que resolviendo se obtiene la función cuadrática para representar la potencia reflejada:

$$\begin{cases} 7a_0 + 8.62a_1 + 12.04a_2 = 45 \\ 8.62a_0 + 12.04a_1 + 17.61a_2 = 63.98 \\ 12.04a_0 + 17.61a_1 + 26.37a_2 = 94.96 \end{cases}$$

$$\begin{cases} a_0 = 1.56 \\ a_1 = -1.21 \\ a_2 = 3.70 \end{cases} \therefore \boldsymbol{f(x) = 1.56 - 1.21 + 3.70x^2}$$

Ecuación 3: Ecuación cuadrática para la estimación de la potencia reflejada

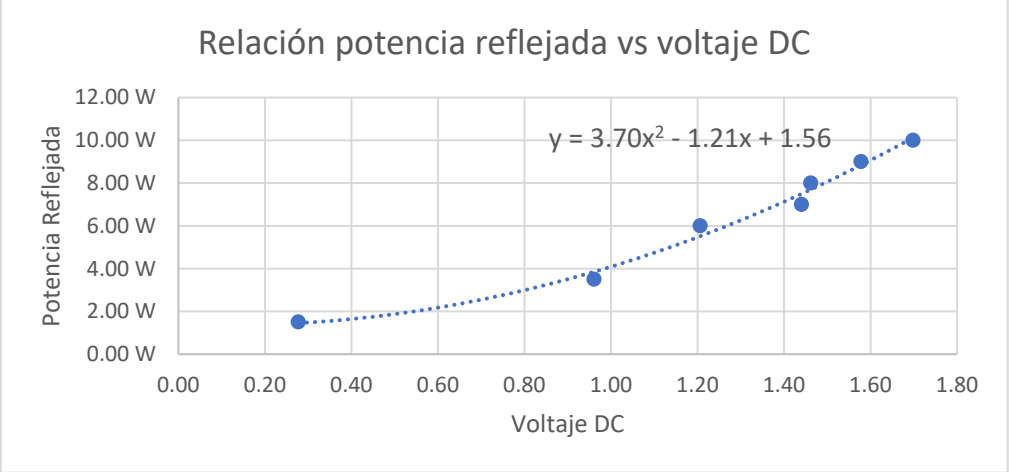


Figura 93: Representación de los datos de la Tabla 29 con su respectiva función cuadrática de regresión

Elaborado por: El investigador

Una vez halladas las ecuaciones se procede a programar el microcontrolador, puesto que los demás sensores no requieren de ninguna configuración adicional como es el caso del módulo PZEM-004T, el código consta de dos partes, uno para el dispositivo que gestiona los sensores y actuadores; y el otro para el caso de la monitorización del equipo a través de imágenes haciendo uso de la ESPCAM.

El algoritmo para el dispositivo de gestión de sensores se muestra en el Anexo 11, en este se definen la librerías a ocupar y demás variables, el código se basa en la conexión Wifi, que en este caso se conecta al enrutador instalado OpenWrt previamente, crea un cliente MQTT para enviar los datos de los sensores a los diferentes widgets, un cliente HTTP para la petición de credenciales MQTT, procesa los datos de los sensores para enviarlos acorde a los tiempos programados por el usuario y escucha los mensajes provenientes

desde el sistema para procesar los distintos actuadores, como se muestra en el siguiente diagrama de flujos:

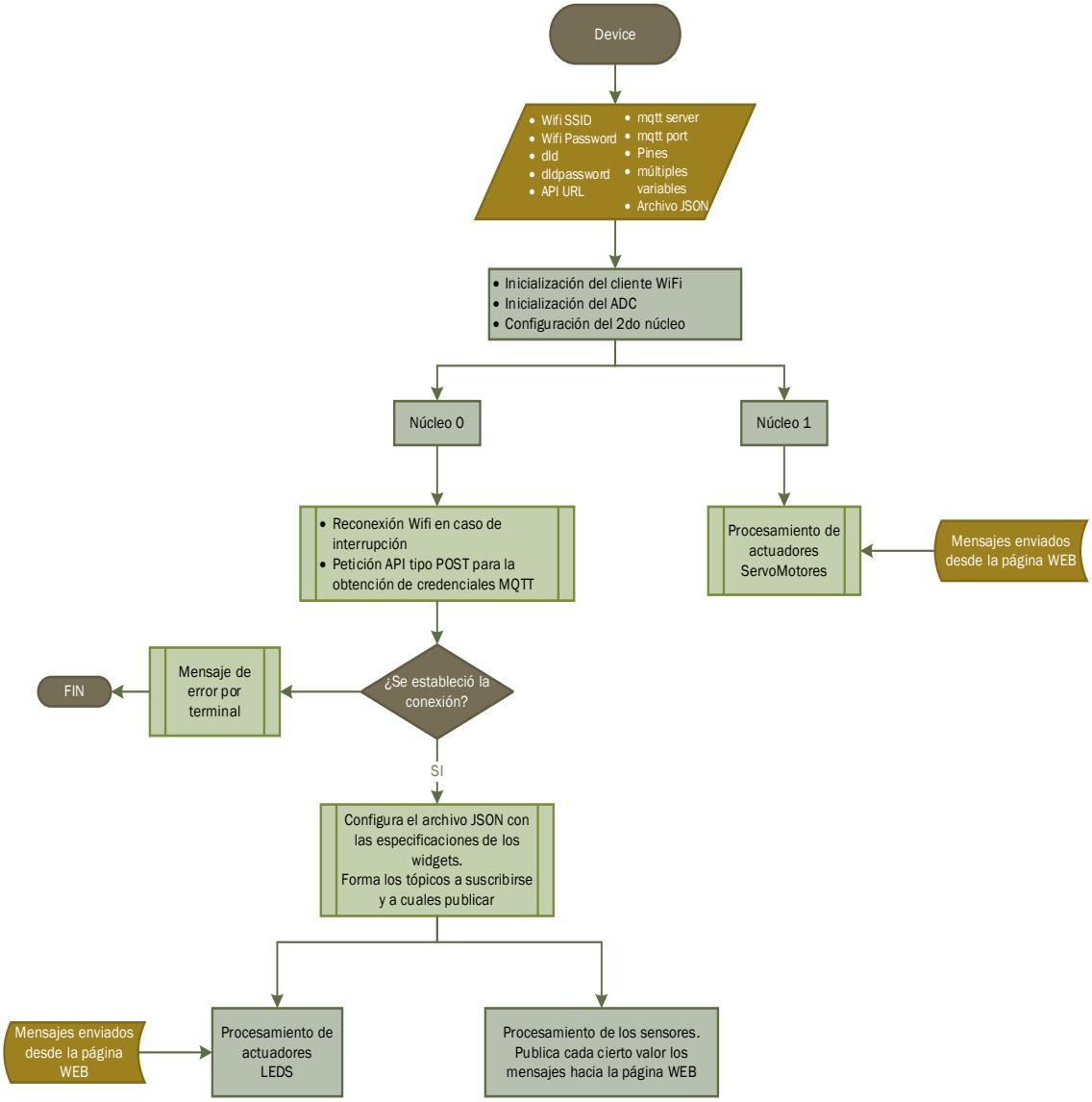


Figura 94: Diagrama de flujo general de los dispositivos IoT
Elaborado por: El investigador

Por otro lado, el dispositivo encargado de obtener las imágenes y enviarlas por medio del protocolo MQTT a la aplicación WEB, en el Anexo 12 se detalla todo el código, en donde el dispositivo al igual que el anterior, requiere de un cliente WiFi, para luego crear un cliente HTTP para la obtención de las credenciales MQTT y por último crear un cliente MQTT que emita las imágenes en arreglos de 8 bits y que además es capaz de escuchar mensajes provenientes desde el usuario, como se muestra en la Figura 94.

3.2.8. Hardware del sistema

En base a lo expuesto en el apartado anterior, se diseñó una tarjeta de control, el cual consta de tres etapas como son: los sensores, el microcontrolador y los actuadores. Por lo que se procedió a realizar un diagrama electrónico con la conexiones necesarias, para luego desarrollar la placa.

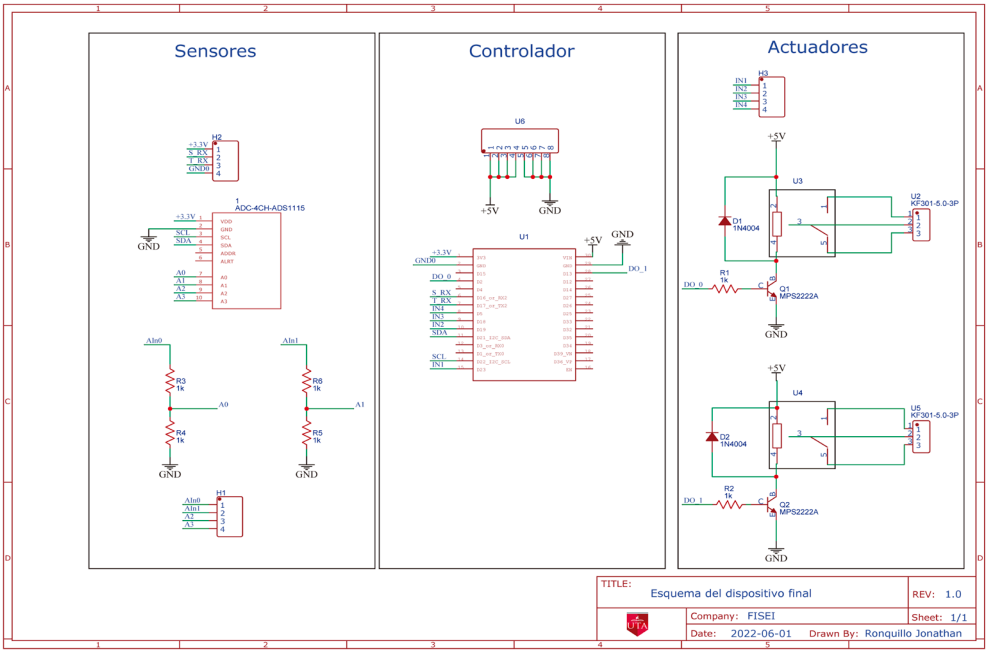


Figura 95: Diagrama electrónico de la tarjeta controladora.
Elaborado por: El investigador

La etapa de acondicionamiento de sensores en la tarjeta de comunicaciones se normalizó el voltaje de 15V proporcionado por los pines remotos del transmisor a 3.3V, lo que significa reducir el voltaje en un 78%, para ello se empleó un divisor de voltaje con la siguiente fórmula:

$$R_1 + R_2 = 100K\Omega$$
$$R_1 = 78K\Omega \ \& \ R_2 = 22K\Omega$$
$$\therefore V_{out} = V_{R_2}$$
$$V_{out} = V_{in} * \left(\frac{R_2}{R_1 + R_2}\right)$$
$$V_{out} = V_{in} * \left(\frac{22K\Omega}{100K\Omega}\right)$$
$$\boldsymbol{V_{out} = 0.22V_{in}}$$

Ecuación 4: Cálculo de resistencias para la normalización de voltaje de 15V a 3.3V.

Este voltaje normalizado pasa a un módulo convertidor de análogo digital (ADC) de 16 bits, para el respectivo muestreo, cuantización y codificación de la señal, obteniendo así una resolución de 0.05mV, como se demuestra en la siguiente fórmula:

$$V_{res} = \frac{V_{ref}}{\#\ de\ niveles} = \frac{3.3V}{2^{16}} = \frac{3.3V}{65536} = \mathbf{0.05\ mV}$$

Ecuación 5: Resolución de voltaje de la tarjeta de comunicaciones

Por otro lado, el sensor de potencia AC se comunica con la tarjeta diseñada por el protocolo receptor-transmisor asíncrono universal (UART) o serial, mediante el estándar RS232 permite la comunicación a distancias de hasta 15 metros.

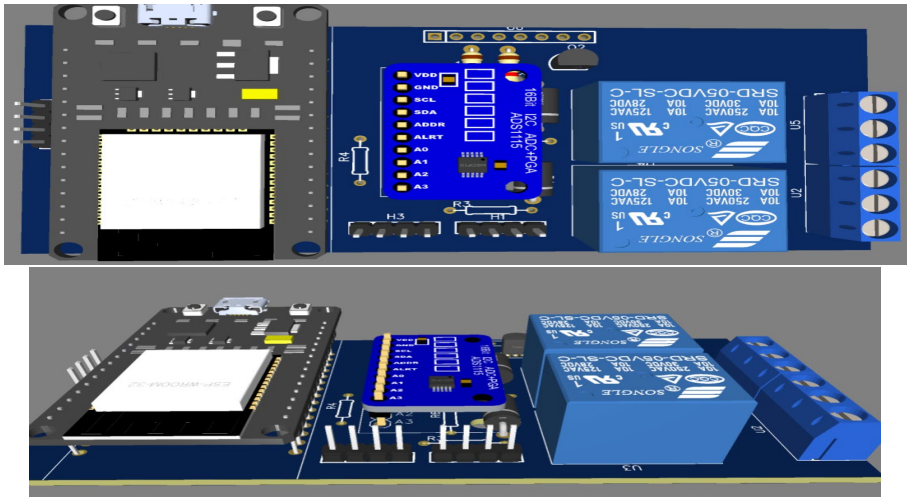


Figura 96: Vista superior y frontal del circuito en 3D.
Elaborado por: El investigador

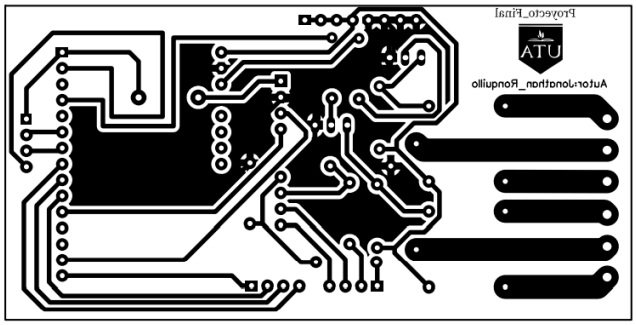


Figura 97: Circuito impreso junto a la máscara de soldadura.
Elaborado por: El investigador

La etapa de control fue diseñada para que la ESP32 se comunique con el router por medio de la tecnología inalámbrica WiFi utilizando el estándar 802.11, para ello el módulo cuenta con una antena integrada en la placa, la misma que presenta las siguientes ganancias como se muestra en la siguiente figura:

Model	Test Item	Test State	Frequency (MHz)	Efficiency (%)	Gain (dB)	Note
ESP-ANT B	Gain	Free Space	2412	73.79	2.39	Vertical 30°
			2417	77.04	2.97	
			2422	79.83	2.80	
			2427	81.19	2.89	
			2432	80.54	3.04	
			2437	76.86	2.86	
			2442	76.17	2.99	
			2447	73.99	2.96	
			2452	72.00	2.80	
			2457	70.71	2.72	
			2462	71.31	2.94	
			2467	71.32	3.12	
			2472	72.03	3.28	
			2477	72.71	3.24	
			2482	75.42	3.40	

Figura 98: Ganancia de la antena PCB de la ESP32 [31]

Obteniendo 3dB de ganancia promedio de la antena PCB. Tomando de referencia el modelo de propagación para interiores y realizando un presupuesto de enlace para determinar la potencia de recepción se tienen los siguientes cálculos:

$$FSPL = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}\left(\frac{4\pi}{c}\right)$$

Ecuación 6: Cálculo de las perdidas en el espacio libre

Aplicando la Ecuación 6, en donde:

- d corresponde a la distancia entre el router WiFi y la tarjeta de comunicaciones en metros.
- f la frecuencia de operación, que en este caso es de 2.4GHz
- c velocidad de la luz

Se tiene:

$$FSPL = 20\log_{10}(3) + 20\log_{10}(2.4 \times 10^9) - 147.55$$

$$\therefore \boldsymbol{FSPL = 49.59}$$

Aplicando el presupuesto de enlace para obtener la potencia de recepción se tiene:

$$P_{Rx} = P_{Tx} + G_{Tx} - L_{Tx} - L_{FS} - L_M - L_{Rx} + G_{Rx}$$

Ecuación 7: Cálculo de potencia de recepción

En donde:

- P_{Rx} potencia de recepción (dBm)
- P_{Tx} potencia de transmisión (dBm)
- G_{Tx} Ganancia de la antena de transmisión (dBi)
- L_{Tx} Perdidas en el transmisor (cables, conectores, etc.) (dB)
- L_{FS} Perdidas en el aire libre (FSPL)(dB)
- L_M Otras perdidas como paredes, espejos, personas, entre otros. (dB)
- L_{Rx} Perdidas en el receptor (cables, conectores, etc.) (dB)
- G_{Rx} Ganancia de la antena del receptor (dBi)

En este caso el transmisor es el router My Net N750, en la Tabla 12 se detallan los datos de propagación de la señal WiFi y junto a los datos obtenidos en el apartado se tiene:

$$P_{Rx} = 22dBm + 10dB - 0 - 49.59 - 0 - 20 + 3dB$$
$$\therefore \boldsymbol{P_{Rx} = -34.59dBm}$$

Por lo tanto, se tuvo una buena cobertura entre el router y dispositivo IoT, mediante este análisis se puede determinar la cobertura máxima del sistema, la misma que sobrepasa los 50 metros en escenarios sin ningún tipo de obstáculos.

En la etapa de potencia de la tarjeta, se calculó el ancho de la pista necesaria para soportar un amperaje de no mayor a los 10 amperios, para ello se empleó la siguiente fórmula:

$$A = \left(\frac{I}{K \times T_{rise}^{b}}\right)^{1/C}$$

Ecuación 8: Cálculo del área de la pista de la PCB

En donde:

- I corriente máxima (A)
- T_{rise} Aumento de temperatura (ºC)
- K, b, c son constantes que resultan del ajuste de la curva a las curvas de la IPC-2221, los cuales son $k = 0.048, b = 0.44, c = 0.725$ para capas externas

La corriente máxima fue dada en la necesidad de cumplir con los parámetros eléctricos del transmisor y la disponibilidad de los relés de 5V para soportar cargas de hasta 10 amperios, con esta premisa se reemplazan los datos:

$$A = \left(\frac{10}{0.048 \times 1^{0.44}}\right)^{\frac{1}{0.725}} = 1578.65\ mil^2$$

$$W = \frac{A}{t \times 1.378} = \frac{1578.65}{11.81 \times 1.378} = 97mil \approx \mathbf{2.47\ mm}$$

Este resultado se interpreta como el ancho mínimo requerido para que la placa pueda manejar cargas de 10 amperios.

Por último, el consumo de energía del dispositivo IoT está dado por los siguientes cargas:

Tabla 32: Consumo energético en corriente continua del dispositivo IoT

Componente	Corriente [mA]	Voltaje [V]	Potencia [W]
Tarjeta de comunicaciones	280	5	1.4
Cámara	240	5	1.2
Motor a pasos	300	5	1.5
Total	**820**	**5**	**4.1**

Elaborado por: El investigador

En base a la tabla expuesta se necesitó una fuente de alimentación de más de 820mA, y se requirió aislar la fuente de poder del motor a pasos del resto de componentes debido a que es una carga inductiva e influye en el correcto funcionamiento del dispositivo.

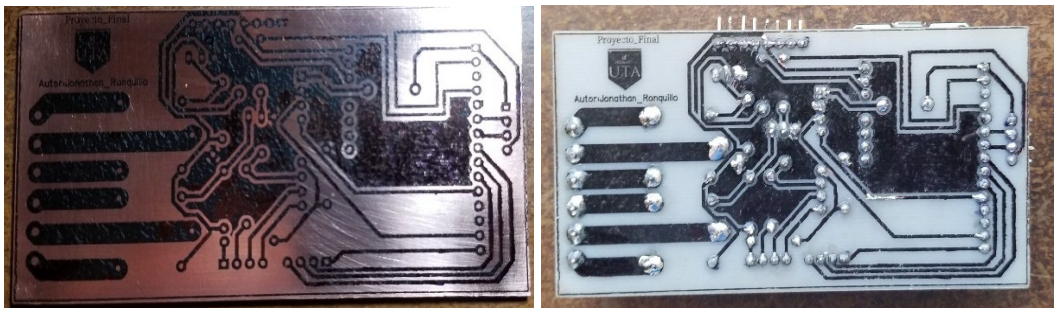


Figura 99: Elaboración de la PCB por el método del planchado.
Elaborado por: El investigador

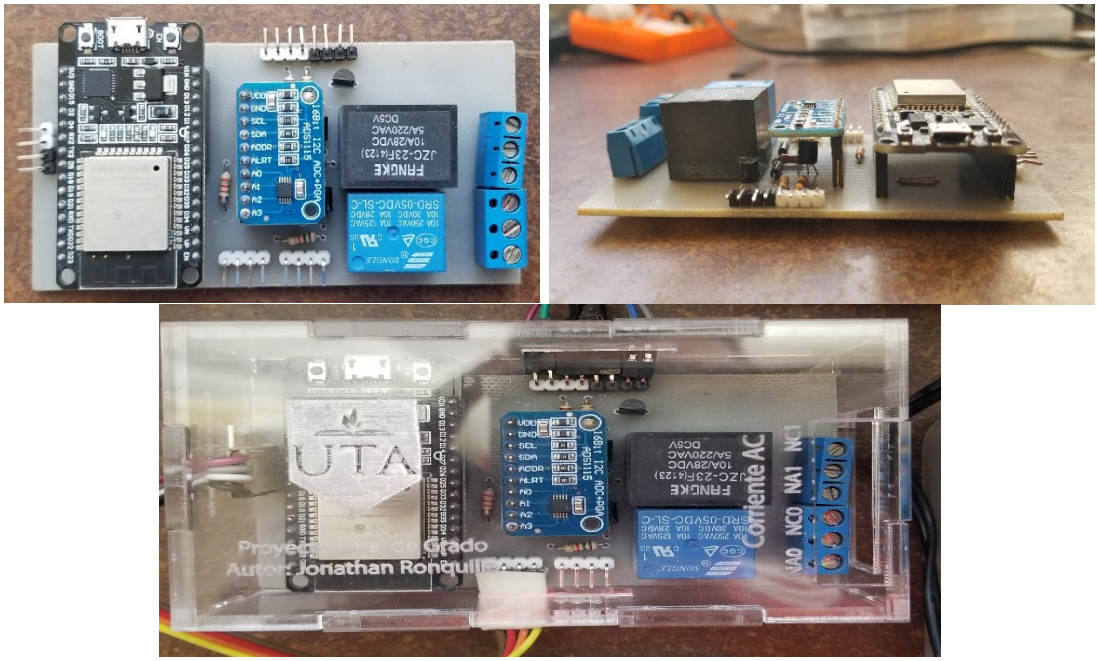


Figura 100: Visto frontal y lateral de la tarjeta de comunicaciones.
Elaborado por: El investigador

3.2.9. Pruebas del sistema implementado

La implementación del dispositivo IoT está conformado por dos partes, la primera fue la implementación del dispositivo en el transmisor RVR tex 100 en los laboratorios de Ecuatronix, para las respectivas calibraciones de sensores como se muestra en la Figura 101, y la segunda parte fue la implementación de este transmisor junto al dispositivo IoT en una caseta del cerro Pilishurco supervisada por la empresa como se detalla en la Figura 102, en donde se aprovechó la infraestructura de red implementada (Anexo 13) previamente para distintos propósitos y se utilizó para la implementación de este proyecto.

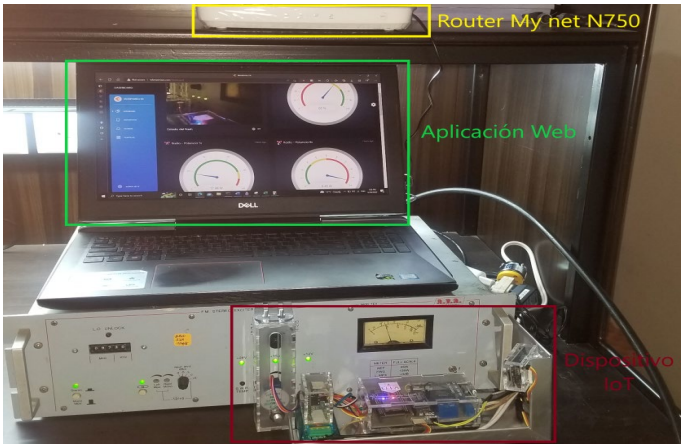


Figura 101: Sistema implementado en el equipo transmisor de RF en los laboratorios de Ecuatronix

Elaborado por: El investigador

Figura 102: Sistema implementado en el equipo transmisor de RF en la caseta del cerro Pilishurco

Elaborado por: El investigador

En la Figura 103 se detalla los componentes del dispositivo IoT los cuales están formados por un motor a pasos con su respectivo controlador, que regula la potencia de transmisión de manera lenta y segura, puesto que a cambios bruscos de este potenciómetro de regulación el transmisor corre el riesgo de quemarse, por otro lado se tiene el sensor de potencia AC que mide la corriente, voltaje, potencia activa, factor de potencia, consumo energético y frecuencia en corriente alterna, así mismo, se tiene la tarjeta de comunicaciones que posee un convertidor análogo a digital que procesa las muestras tomadas desde los terminales de telemetría del equipo y poder obtener así la potencia transmitida y reflejada del equipo. Por otro lado se tiene actuadores como los relés mecánicos para la interrupción de corriente del equipo y poder así apagarlo. Finalmente una cámara que permitirá la visualización del estado del equipo por medio de indicadores led.

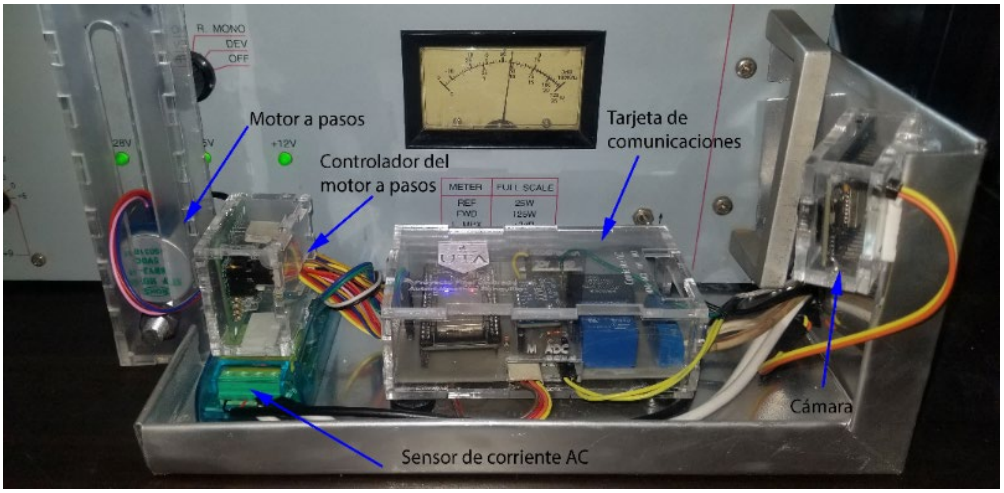


Figura 103: Dispositivo IoT y sus componentes

Elaborado por: El investigador

La tarjeta controladora y la cámara se conectan al bróker de EMQX como se detalla en la siguiente figura:

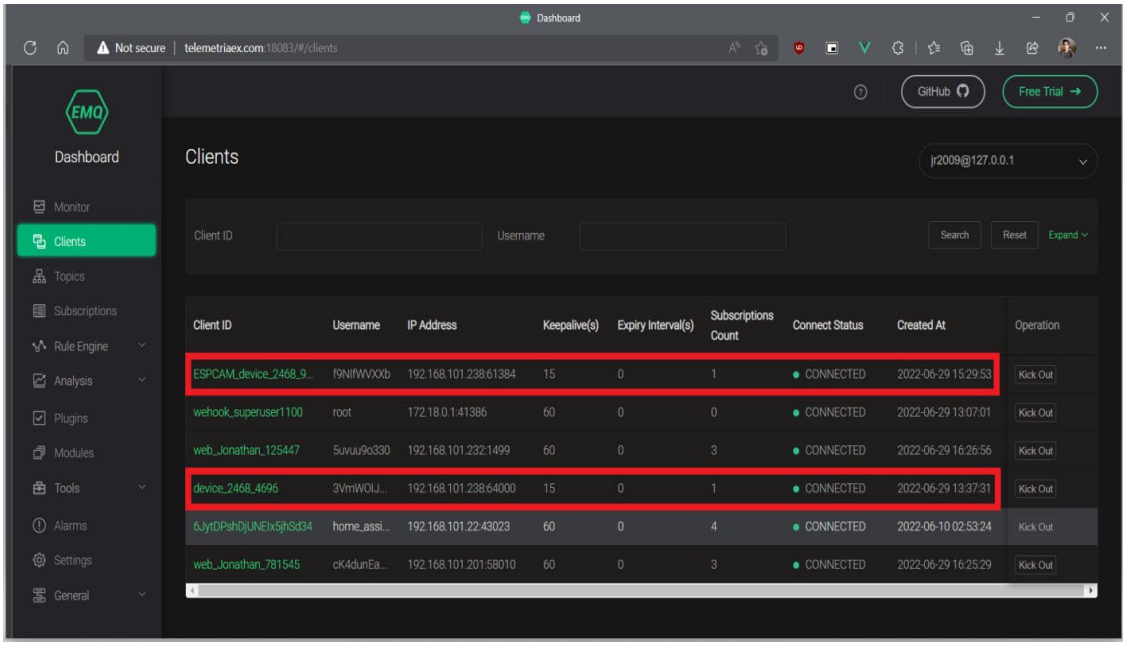


Figura 104: Clientes MQTT conectados al bróker
Elaborado por: El investigador

En la Figura 105, se muestra la latencia de la red en distintos escenarios como son:

- **Escenario 1**.- ping desde un dispositivo conectado a la red Wifi de la caseta del cerro Pilishurco hacia el servidor.
- **Escenario 2**.- ping desde un dispositivo conectado a la red de OpenWrt ubicado en los laboratorios de Ecuatronix hacia el servidor.
- **Escenario 3**.- ping desde un dispositivo conectado a la red de Wifi ubicado en los laboratorios de Ecuatronix hacia el servidor.
- **Escenario 4**.- ping desde un dispositivo móvil conectado a la red de CNT en el sector de la Cdla. España, hacia el servidor.
- **Escenario 5**.- ping desde un dispositivo móvil conectado a la red de Movistar en el sector de la Cdla. España, hacia el servidor.
- **Escenario 6**.- ping desde un dispositivo móvil conectado a la red de Claro en el sector de la Cdla. España, hacia el servidor.

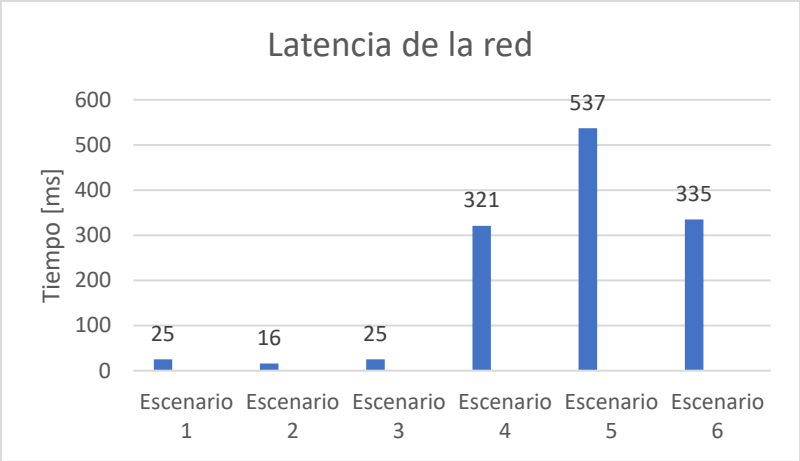


Figura 105: Latencia de la red SDN en distintos escenarios
Elaborado por: El investigador

En este gráfico, se muestra que la red SDN tiene menor latencia en redes fijas, conectadas a un solo proveedor de internet; mientras que la latencia aumenta en redes móviles, siendo la red CNT con menor latencia, ver Anexo 14.

En lo que respecta al ancho de banda que consume el dispositivo IoT, la cámara encendida consume un mayor ancho de banda de alrededor de 1.5Mbps, mientras que solo la transmisión de datos, es decir sin la cámara activada consume alrededor de 200Kbps como se muestra en las gráficas de la Figura 106. Por otro lado el servidor está limitado a una ancho de banda de 50Mbps, que al retransmitir la información proporcionada por el dispositivo IoT a cada cliente conectado, se traduce en la siguiente tabla:

Tabla 33: Número máximo de clientes soportados del sistema en función del tipo de datos retransmitidos

	Número de Clientes soportados.
Retransmisión de datos de la cámara	33
Retransmisión de los datos de los sensores	250
Ambos tipos de datos	29

Elaborado por: El investigador

Figura 106: Ancho de banda consumida por el dispositivo IoT con cámara funcionado lado izquierdo y sin la cámara lado derecho

Elaborado por: El investigador

Analizando los recursos consumidos por el enrutador My net N750, se evidencia mayor consumo con recursos con la cámara encendida llegando a un consumo de CPU del 23.5% y del 6.5% limitando al sistema al envió solo de datos de los sensores recolectados por la tarjeta controladora. Así mismo, los recursos consumidos por el servidor están en función de los clientes conectados, sin embargo el servidor consume alrededor de 1GB de RAM y 5% de CPU tan solo por ejecutar la aplicación web y distintos servicios web mencionados en el apartado 3.2.3. Mientras que al conectar 3 clientes del sistema y junto a la retransmisión del servidor hacia estos, el servidor consume alrededor de 1.03GB de RAM, 10% de CPU y un ancho de banda de 6.5Mbps de subida y de bajada 2.21Mbps, como se muestra en la siguiente figura:

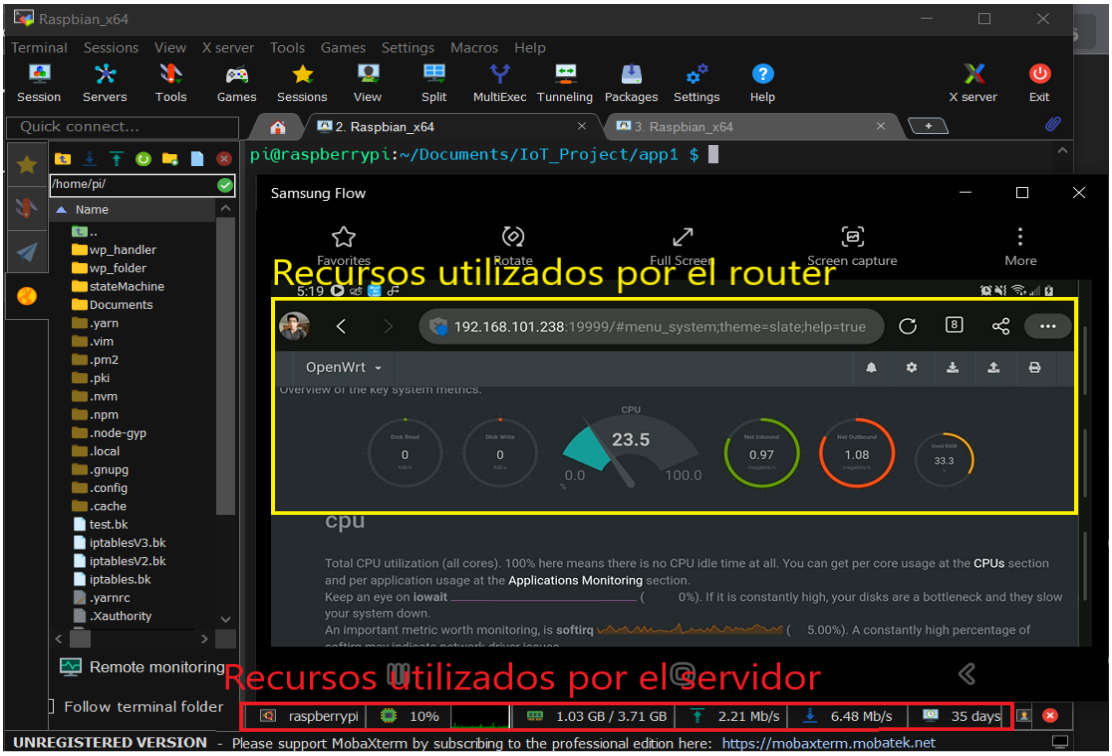


Figura 107: Recursos consumidos por el router y el servidor

Elaborado por: El investigador

La seguridad de la red está garantizada por Zerotier, el cual encripta toda la información que fluye al salir desde el servidor hacia el cliente final, como se demuestra en la Figura 108, la primera imagen muestra la codificación de los datos que salen desde la dirección IP publica donde está conectado el servidor, hacia el cliente dentro de la red LAN de la empresa, cabe recalcar que el sistema no utiliza una encriptación de datos SSL y por ende los usuarios autorizados que estén conectados a la red SDN, sí pueden ver los datos enviados por los dispositivos, como se muestra en la Figura 108 segunda imagen.

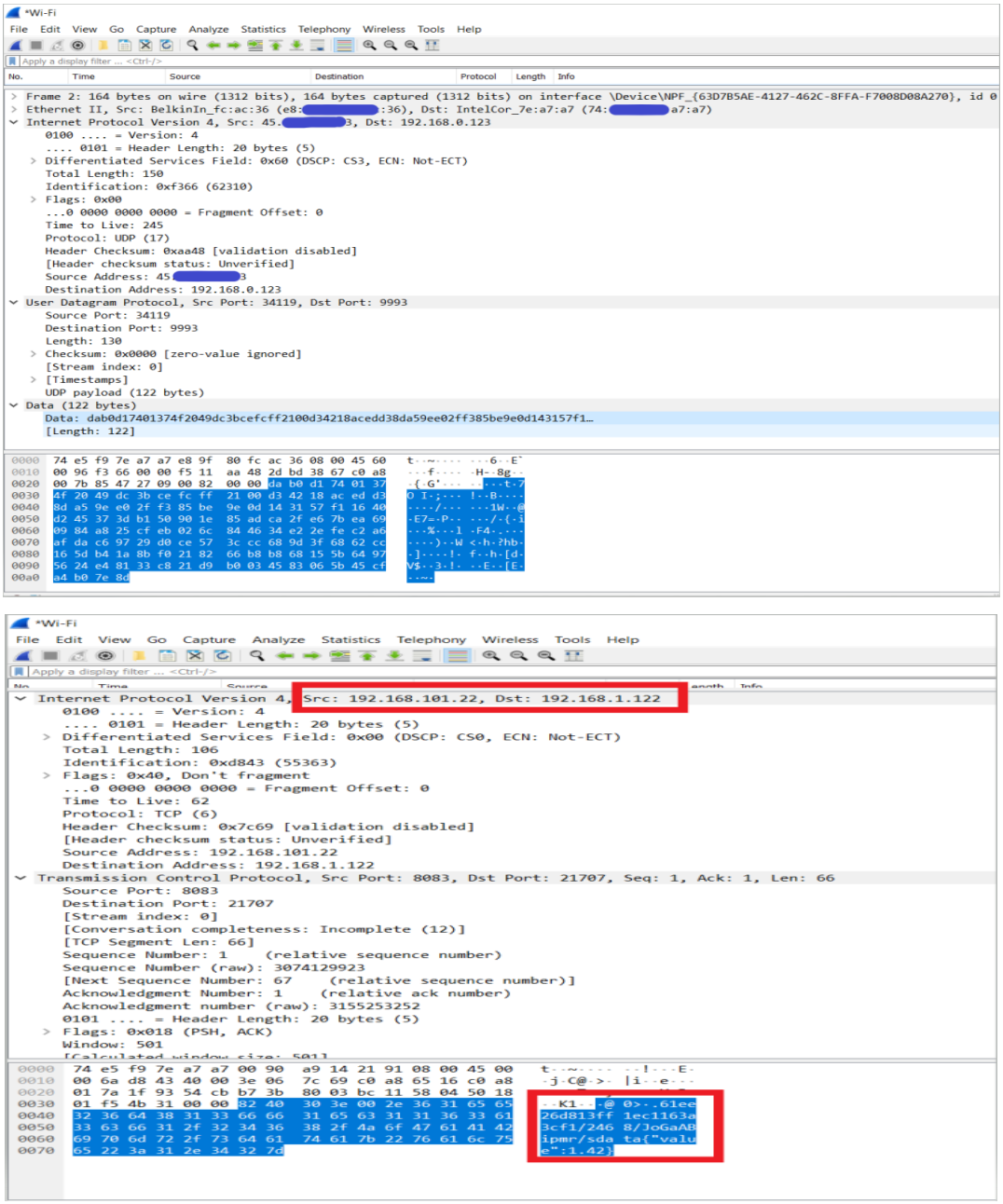


Figura 108: Encriptación de los datos enviados por la red de Zerotier

Elaborado por: El investigador

Por último, en la Tabla 34 se muestra los resultados obtenidos tanto por el sistema como por los instrumentos de medición, ver Anexo 10; en donde se configuró el equipo para propagar señal RF a una potencia de 50W, obteniendo márgenes de errores de alrededor del 1% en parámetros eléctricos del equipo y del 1% al 7% en parámetros de potencia RF del equipo, como se muestra en la siguiente tabla:

Tabla 34: Porcentaje de error del sistema

Magnitud	Medición del sistema	Medición con instrumentos	Error Relativo	Porcentaje de Error
Potencia de Transmisión	49.13W	50.00W	± 0.87	1.74%
Potencia Reflejada	1.86W	2.00W	± 0.14	7.00%
Voltaje AC	114.30V	112.00V	± 2.30	2.05%
Corriente AC	1.39A	1.40A	± 0.01	0.93%
Potencia AC	158.53A	156.80A	± 1.73	1.11%

Elaborado por: El investigador

3.2.10. Presupuesto

El presente proyecto está compuesto por dos partes, como son el dispositivo IoT y la infraestructura de red, como se detallan en la Tabla 35 y Tabla 36 respectivamente.

Dispositivo IoT

El dispositivo consta de diversos módulos y controladores, así como de carcasas y materiales para la producción del dispositivo tal y como se muestran en la Tabla 35.

Tabla 35: Presupuesto del dispositivo IoT.

No.	Concepto	Cantidad	Precio Unitario	Precio Total
1	ESP-32	1	$12,00	$12,00
2	ESPCAM	1	$15,00	$15,00
3	ADS1115	1	$11,00	$11,00
4	Motor a Pasos	1	$5,00	$5,00
5	Relé 5V	2	$0,50	$1,00
6	Borneras	2	$0,25	$0,50
7	Espadines	2	$0,50	$1,00
8	Baquelita	1	$5,00	$5,00
9	Resistencias	6	$0,05	$0,30
10	Transistores	2	$0,15	$0,30
11	Cables de Conexión.	1	$3,00	$3,00
12	Carcasa de acrílico	3	$8,00	$24,00
13	Bandeja Metálica	1	$20,00	$20,00
14	Fuente de Alimentación	2	$8,00	$16,00
15	Materiales varios	1	$10,00	$10,00
Subtotal				**$124.10**

Elaborado por: El investigador

Infraestructura de red

Por otro lado, en la infraestructura de red consta del servidor y del router respectivamente, en la Tabla 36; cabe recalcar que no se contempla los costos de instalación de la red punto a punto entre los laboratorios de Ecuatronix hacia el cerro Pilishurco, ya que esta infraestructura fue montada previamente.

Tabla 36: Presupuesto de la infraestructura de red SDN

No.	Concepto	Cantidad	Precio Unitario	Precio Total
1	Router	1	$18,00	$18,00
2	Raspberry Pi4	1	$220,00	$220,00
Subtotal				**$238,00**

Elaborado por: El investigador

Presupuesto final

Contemplando los costos anteriores y gastos imprevistos, el costo final para la implementación del proyecto es de $478.36, cabe recalcar que el usuario final solo tendrá que cubrir el costo del dispositivo, router, transporte e instalación que suman un total de $192,10.

Tabla 37: Presupuesto Final del proyecto

Concepto	Precio Total
Dispositivo IoT	$124,10
Infraestructura de red	$238,00
Costos de movilización	$30,00
Subtotal	$392.10
Imprevistos(10%)	$39.21
IVA(12%)	$47.05
Total	**$478.36**

Elaborado por: El investigador

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1.Conclusiones

- Una vez realizado el análisis a diferentes transmisores de RF, se identificó que la mayoría de estos equipos ofrece puertos de telemetría para su gestión, relacionando los parámetros de potencia RF emitida y reflejada con voltajes DC de manera no lineal, requiriendo convertidores analógicos a digitales para la adquisición de estos datos para su posterior procesamiento; mientras que el monitoreo de variables eléctricas referentes al equipo se utilizaron módulos de alta precisión que miden la corriente AC, voltaje AC, potencia activa, factor de potencia y energía consumida.

- El protocolo MQTT ofrece baja latencia y bajo consumo de energía, se puede instalar clientes MQTT en dispositivos pequeños, y asegura que el mensaje llegue a su destino mediante la calidad de servicio; características que permitió el correcto funcionamiento del dispositivo IoT en condiciones de alta interferencia electromagnética y bajo ancho de banda en la red, como fue el caso de la red en el cerro Pilishurco. Mientras que ZeroTier fue el mejor controlador SDN para el sistema, puesto que es escalable, seguro y soporta cualquier tipo de dispositivo final, permitiendo crear una red SDN accesible desde cualquier lugar, siempre y cuando el dispositivo sea autorizado previamente.

- Se diseñó una aplicación web intuitiva que permite el monitoreo y control del dispositivo IoT adaptado al transmisor FM, aplicación que se ejecuta del lado del cliente liberando recursos del servidor y que está disponible para navegadores web móviles y de escritorio, el frontend fue desarrollado en el lenguaje Vue.js y el backend en JavaScript lenguajes que hoy en día son populares y de alta documentación.

4.2.Recomendaciones

Con el desarrollo de este proyecto se recomienda lo siguiente:

- Los recursos de hardware de los microcontroladores en los dispositivos IoT, se pueden reducir utilizando módulos o sensores externos, permitiendo así que el microcontrolador solo se dedique a comunicar los datos de los sensores y en caso de fallas en los sensores no se bloquee.

- Utilizar módulos convertidores análogo a digital, los cuales a diferencia de los ADC de los microcontroladores, los módulos poseen mayor resolución y tolerancia al ruido, lo que permite una mejor toma de muestras del voltaje DC.

- El bróker MQTT de EMQX posee múltiples parámetros de configuración, entre las que destacan son la memoria máxima del búfer la cual permite que se envíen mensajes pesados por este medio, esto es indispensable para la cámara por lo que se recomienda colocar valores grades como 200MB caso contrario el bróker se reiniciara.

- Si bien la comunicación entre usuarios de la red SDN de Zerotier es encriptada, se recomienda utilizar una encriptación adicional sobre los servicios que se estén ejecutando, en este caso la aplicación web y el bróker MQTT, para mayor seguridad.

- Para el óptimo desempeño de la red Zerotier, es necesario utilizar versiones estables de los controladores, de igual manera en el firmware OpenWrt ya que las versiones actuales provocan errores en la red como desconexiones imprevistas de la red, o el bloqueo del router en el caso de OpenWrt.

REFERENCIAS BIBLIOGRÁFICAS

[1] K. B. Kiadehi, A. M. Rahmani, and A. S. Molahosseini, "A fault-tolerant architecture for internet-of-things based on software-defined networks " *Telecommunication Systems,* paper vol. 77, pp. 155-169, 06 January 2021, doi: 10.1007/s11235-020-00750-1.

[2] Anichur Rahman *et al.*, "DistB-SDoIndustry: Enhancing Security in Industry 4.0 Services based on Distributed Blockchain through Software Defined Networking-IoT Enabled Architecture," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 11, pp. 674-681, 18 December 2020, doi: 10.14569/IJACSA.2020.0110980.

[3] Ammar Muthanna *et al.*, "Secure and Reliable IoT Networks Using Fog Computing with Software-Defined Networking and Blockchain," *Journal of Sensor and Actuator Networks,* vol. 15, pp. 1-17, 18 February 2019, doi: 10.3390/jsan8010015.

[4] Slavica Tomovic, Kenji Yoshigoe, Ivo Maljevic, and I. Radusinovic, "Software-Defined Fog Network Architecture for IoT," *Wireless Personal Communications* vol. 92, pp. 181–196, 24 October 2017, doi: 10.1007/s11277-016-3845-0.

[5] A. Albayrak, "IoT-based Real-Time Telemetry System Design: An Approach," presented at the 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, 20 November, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8114469.

[6] Secretaría Nacional de Comunicación, "PROYECTO REHABILITACIÓN DE RADIO CIUDADANA ". [Online]. Available: https://www.presidencia.gob.ec/wp-content/uploads/2019/03/k_proy_radio_feb_2019.pdf

[7] ARCOTEL. Listado completo de estaciones de radiodifusión sonora y televisión abierta a nivel nacional [Online] Available: https://www.arcotel.gob.ec/wp-content/uploads/2021/06/8.1.3.-Listado_RTV-Mayo-2021.xls

[8] Alonso Llanos, "Gestión del espectro radioeléctrico en Ecuador." [Online]. Available: https://repositorio.uasb.edu.ec/bitstream/10644/3833/1/SM138-Llanos-Gestion.pdf

[9] ARCOTEL, "Listado Completo de frecuencias auxiliares de radiodifusión sonora y televisión abierta nivel nacional." [Online]. Available: https://www.arcotel.gob.ec/wp-content/uploads/downloads/2015/06/757_Listado_completo_de_frecuencias_auxiliares_de_radiodifusion_sonora_y_television_abierta_a_nivel_nacional.xlsx

[10] ARCOTEL, "NORMA TÉCNICA PARA EL SERVICIO DE RADIODIFUSIÓN SONORA EN FRECUENCIA MODULADA ANALÓGICA." [Online]. Available: https://www.arcotel.gob.ec/wp-content/uploads/downloads/2015/04/NORMA-TECNICA.pdf

[11] EEASA, "Estadistica Anual 2019," p. 55. [Online]. Available: https://www.eeasa.com.ec/content/uploads/2020/08/Estadistica-Anual-2019.pdf

[12] Asamblea Nacional de Ecuador, *PLIEGO TARIFARIO PARA LAS EMPRESAS ELÉCTRICAS DE DISTRIBUCIÓN*, Quito-Ecuador: ARCONEL, 2019, p. 19. [Online]. Available: https://www.cnelep.gob.ec/wp-

content/uploads/2020/01/pliego_tarifario_del_spee_2020_resolucion_nro_035_19.pdf.
[13] W. Tomasi, "Introducción a las comunicaciones electrónicas," in *Sistemas de Comunicaciones Electrónicas*, Pearson Ed., 4ª ed., PEARSON, Ed. México, 2003, ch. 1, pp. 1-11.
[14] Francesc Rey Micolau and Francesc Tarrés Ruiz, *Comunicaciones analógicas : modulaciones AM y FM*, Cataluña-España: Universidad Abierta de Cataluña, 2013, pp. 19-23. [Online]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/69406/5/Sistemas%20de%20comunicaci%C3%B3n%20I_M%C3%B3dulo%202_Comunicaciones%20analog%C3%B3gicas%3B%20modulaciones%20AM%20y%20FM.pdf.
[15] S. Haykin, *Communication Systems*, 4 ed. NY-USA: John Wiley & Son, INC, 2001, pp. 88-121. [Online]. Available: https://ict.iitk.ac.in/wp-content/uploads/EE320A-Principles-Of-Communication-CommunicationSystems-4ed-Haykin.pdf.
[16] A. B. Carlson and P. B. Crilly, *Communication Systems An Introduction to Signal and Noise in Electrical Communication*, 5 ed. NY-USA: McGraw-Hill, 2010, pp. 163-179. [Online]. Available: https://research.iaun.ac.ir/pd/naghsh/pdfs/UploadFile_9495.pdf.
[17] C. Pablo. "Transmisor / Interceptor de AM." http://www.pablin.com.ar/electron/circuito/radio/interfam/index.htm (accessed 15 de Junio de 2019, 2010).
[18] F. R. M. F. T. Ruiz, *Comunicaciones analógicas: modulaciones AM y FM*. 2013.
[19] *TEX2003TFT USER MANUAL*, Italy, 2021, pp. 1-3. [Online]. Available: https://www.rvr.it/backstage/pages/page13/205/docu_eng/M1TEX2003TFT11EN.pdf.
[20] D. S. M. Wolf, *Internet-of-Things (IoT) Systems*, 4 ed. Atlanta, USA: Springer, 2018, p. 102. [Online]. Available: http://alvarestech.com/temp/smar/Smar/Book2021/Industry4.0/2019/Dimitrios%20Serpanos,Marilyn%20Wolf%20(auth.)%20-%20%20Internet-of-Things%20(IoT)%20Systems_%20Architectures,%20Algorithms,%20Methodologies-Springer%20International%20Publishing%20(2018).pdf.
[21] E. Hossain, *INTERNET OF THINGS A TO Z*, 1 ed. New Jersey, USA: John Wiley & Sons, Inc., 2018, p. 705. [Online]. Available: https://www.wiley.com/en-us/Internet+of+Things+A+to+Z%3A+Technologies+and+Applications-p-9781119456742.
[22] G. Dunko, J. Misra, J. Robertson, and T. Snyder, *A Reference Guide to the Internet of Things*, 1 ed. North Carolina , USA: Bridgera LLC, 2017, p. 82. [Online]. Available: https://bridgera.com/wp-content/uploads/2018/10/IoTeBook3.pdf.
[23] R. P. Colón, Sergio Navajas, and E. Terry, "IoT IN LAC 2019: ," pp. 0-40. [Online]. Available: https://publications.iadb.org/publications/english/document/IoT_IN_LAC_2019_Taking_the_Pulse_of_the_Internet_of_Things_in_Latin_America_and_the_Caribbean_en.pdf

[24] g. labs. "Table Comparing Wireless Protocols for IoT Devices." http://glowlabs.co/wireless-protocols/ (accessed 25 de Octubre de 2021, 2021).
[25] H. GENG, *INTERNET OF THINGS AND DATA ANALYTICS*, WILEY, ed., 1 ed. California, USA: WILEY, 2017, p. 816. [Online]. Available: https://www.wiley.com/en-us/Internet+of+Things+and+Data+Analytics+Handbook-p-9781119173649.
[26] R. W. World. "Advantages of XMPP protocol | disadvantages of XMPP protocol." https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-XMPP-protocol.html (accessed 25 de Octubre de 2021, 2021).
[27] InterSystems. "Settings for the MQTT Adapter." https://docs.intersystems.com/healthconnectlatest/csp/docbook/DocBook.UI.Page.cls?KEY=EMQTT_SETTINGS (accessed 26 de Octubre de 2021, 2021).
[28] OASIS Message Queuing Telemetry Transport (MQTT). "MQTT Specifications." https://mqtt.org/mqtt-specification/ (accessed 26 de Octubre de 2021, 2021).
[29] W. E. Day, "Smart Cities – Adoption of Future Technologies," pp. 1-24. [Online]. Available: https://worldengineeringday.net/wp-content/uploads/2020/03/Smart-City-IOT-WFEO-Version-1.pdf
[30] T. Harwood. "What Is The "Internet of Things"?" https://www.postscapes.com/what-exactly-is-the-internet-of-things-infographic/ (accessed 27 de Octubre de 2021, 2021).
[31] Julpin Inc. "Placa De Desarrollo ESP32." https://www.julpin.com.co/inicio/microprocesadores/1294-placa-de-desarrollo-esp32-wifi-bluetooth.html (accessed 7 Jul. , 2022).
[32] D. Das. "AC Current Measurement using Current Transformer and Arduino." https://circuitdigest.com/electronic-circuits/ac-current-measurement-circuit-using-current-transformer-and-arduino (accessed 9 Jul 2022).
[33] Sparkfun. "Analog to Digital Conversion." https://learn.sparkfun.com/tutorials/analog-to-digital-conversion/all (accessed 10 Jul 2022).
[34] RedHat. "Diferencias entre los contenedores y las máquinas virtuales." https://www.redhat.com/es/topics/containers/containers-vs-vms (accessed 27 de Octubre de 2021, 2021).
[35] D. Huang, A. Chowdhary, and S. Pisharody, *Software-Defined Networking and Security*, 1 ed. Arizona, USA: CRC Press, 2018, p. 357. [Online]. Available: https://www.routledge.com/Software-Defined-Networking-and-Security-From-Theory-to-Practice/Huang-Chowdhary-Pisharody/p/book/9780815381143.
[36] T. D. Nadeau and K. Gray, *SDN: Software Defined Networks*, 1 ed. Massachusetts, USA.: O'Reilly Media, Inc., 2013, p. 384. [Online]. Available: https://www.oreilly.com/library/view/sdn-software-defined/9781449342425/.
[37] P. Göransson, C. Black, and T. Culver, *Software Defined Networks*, 2 ed. Massachusetts, USA.: Elsevier Inc., 2018, p. 357. [Online]. Available: https://www.elsevier.com/books/software-defined-networks/goransson/978-0-12-804555-8.
[38] S. Saha. "Building a Secure Network from Cloud to Edge with ZeroTier." https://gestaltit.com/events/sulagna/building-a-secure-network-from-cloud-to-edge-with-zerotier/ (accessed 9 Jun. , 2022).

[39] K. Myers. "Remote workers – rapid and cost-effective VPN scale with ZeroTier, OPNSense and FRRouting." https://stubarea51.net/2020/03/10/remote-workers-rapid-and-cost-effective-vpn-scale-with-zerotier-opnsense-and-frrouting/ (accessed 8 Jun., 2022).
[40] ZeroTier Inc. "Protocol Design Whitepaper." https://docs.zerotier.com/zerotier/manual (accessed 4 Jun. , 2022).
[41] M. Stubbig. "ZeroTier: The Internet as a Global Switch." https://www.opensourceforu.com/2022/03/zerotier-the-internet-as-a-global-switch/ (accessed 8 Jun. , 2022).
[42] J. Rico. "ZeroTier - A smart p2p SDN solution." https://blog.rico-j.de/zerotier-one/ (accessed 7 Jun. , 2022).
[43] Devopedia. "OpenWrt." https://devopedia.org/openwrt (accessed 10 Jul 2022).
[44] Devopedia. "DD-WRT vs. Tomato vs. OpenWRT: Which Router Firmware Is the Best?" https://www.maketecheasier.com/dd-wrt-vs-tomato-vs-openwrt-router-firmware/ (accessed 10 Jul 2022).
[45] W. Digital. "WD My Net N750 HD Dual Band Router Wireless N Router WiFi acelerar HD." https://www.amazon.com/-/es/N750-HD-Dual-Router-Wireless-acelerar/dp/B007KZQM9Q (accessed 10 Jul 2022).
[46] Docker Inc. "Docker overview." https://docs.docker.com/get-started/overview/#:~:text=Docker%20uses%20a%20client%2Dserver,to%20a%20remote%20Docker%20daemon. (accessed 7 Jun. , 2022).
[47] E. Kisller. "THE BASICS: A Beginner's Guide to Docker." https://jfrog.com/knowledge-base/the-basics-a-beginners-guide-to-docker/ (accessed 7 Jun. , 2022).
[48] Portainer.io. "Portainer Documentation." https://github.com/portainer/portainer (accessed 8 Jun. , 2022).
[49] Docksal. "DOCKER WEB UI: PORTAINER." https://docs.docksal.io/use-cases/portainer/ (accessed 9 Jun. , 2022).
[50] MongoDB Inc. "What is MongoDB?" https://www.mongodb.com/es/what-is-mongodb (accessed 6 Jun. , 2022).
[51] D. Taylor. "What is MongoDB? Introduction, Architecture, Features & Example." https://www.guru99.com/what-is-mongodb.html (accessed 4 Jun. , 2022).
[52] Stackshare. "InfluxDB vs MongoDB: What are the differences?" https://stackshare.io/stackups/influxdb-vs-mongodb (accessed 10 Jul 2022).
[53] EMQX Inc. "Introduction to EMQX." https://www.emqx.io/docs/en/v4.4/#features-list (accessed 9 Jun. , 2022).
[54] Steve I. "MQTT Brokers and Cloud Hosting Guide." http://www.steves-internet-guide.com/mqtt-hosting-brokers-and-servers/ (accessed 11 Jul 2022).
[55] Pi-hole Inc. "Overview to Pi-hole." https://docs.pi-hole.net/ (accessed 9 Jun. , 2022).
[56] Kritopher I. "Pi-Hole vs AdGuard Home for Ad Blocking – 12 Key Differences." https://www.smarthomebeginner.com/pi-hole-vs-adguard-home/ (accessed 11 Jul 2022).
[57] Node.js Inc. "What is Node.js?" https://hub.docker.com/_/node (accessed 8 Jun. , 2022).
[58] V. Suthar. "Explain the working of Node.js." https://www.geeksforgeeks.org/explain-the-working-of-node-

js/#:~:text=to%20multiple%20clients.-,Node.,blocking%20it%20for%20one%20request.&text=Non%2Dblocking%20I%2Fo%3A,thread%20for%20a%20single%20request. (accessed 8 Jun. , 2022).
[59] J. J. Gutiérrez. "Conceptos de un framework web " http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf (accessed 11 Jul 2022).
[60] A. Barragán. "Qué es Vue JS y qué lo diferencia de otros frameworks." https://openwebinars.net/blog/que-es-vue-js-y-que-lo-diferencia-de-otros-frameworks/#:~:text=es%20Vue%20JS-,Vue.,You%2C%20en%20el%20a%C3%B1o%202014. (accessed 11 Jul 2022).
[61] M. Martin. "What is Backend Developer? Skills Need for Web Development." https://www.guru99.com/what-is-backend-developer.html (accessed 11 Jul 2022).
[62] M. Graf. "Which Backend Language Should You Choose?" https://medium.com/swlh/which-backend-language-should-you-choose-eb924902a9b3 (accessed 11 Jul 2022).
[63] C. S. Skorton DJ , García E , Geiser EA , Hillard W , Koppes W , Linker D , Schwartz G "Digital signal and image processing in echocardiography," vol. 110, no. 6. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0002870385900249?via%3Dihub

ANEXOS

Anexo 1

Creación de la vista de dispositivos

La creación de la vista devices se requiere de un archivo “devices.vue” en la carpeta “pages” luego se crea el formulario y una tabla para presentar todos los dispositivos, ademas de las funciones necesarias para guardar en la base de datos el dispositivo y crear la regla en EQMX

```
<template>

  <div>
    <div class="row">
      <card>
        <div slot="header">
          <h4 class="card-title">Add new
device</h4>
        </div>

        <div class="row">
          <div class="col-4">
            <base-input label="Device Name"
type="text" placeholder="Ex: Home, Office..." v-
model="newDevice.name"></base-input>
          </div>
          <div class="col-4">
            <base-input label="Device Id"
type="text" placeholder="Ex: 1234as,2009..." v-
model="newDevice.dId"></base-input>
          </div>
          <div class="col-4">
            <slot name="label">
              <label>Template</label>
            </slot>

            <el-select v-
model="selectedIndexTemplate" placeholder="Select
Template" class="select-primary" style="width:
100%">
              <el-option v-for="(template, index)
in templates" :key="template._id" class="text-dark"
:value="index" :label="template.name"></el-
option>
            </el-select>
          </div>
        </div>

        <div class="row pull-right">
          <div class="col-12">
            <base-button @click="createNewDevice()"
type="primary" class="mb-3" size="lg">Add</base-
button>
          </div>
        </div>
      </card>
    </div>

    <!-- Tabla de dispositivos -->
    <div class="row">
      <card>
        <div slot="header">
          <h4 class="card-title">Devices</h4>
        </div>

        <el-table :data="$store.state.devices">

          <el-table-column label="#" min-
width="50" align="center">
              <div slot-scope="{row, $index}">
                {{$index + 1}}
              </div>
          </el-table-column>

          <el-table-column prop="name"
label="Name"></el-table-column>

          <el-table-column prop="dId"
label="Device Id"></el-table-column>

          <el-table-column prop="dIdpassword"
label="Password"></el-table-column>

          <el-table-column prop="templateName"
label="Template"></el-table-column>

          <el-table-column label="Actions">
              <div slot-scope="{row, $index}">
                <el-tooltip content="Saver
Status Indicator" style="margin-right:10px">
                    <i class="fas fa-database "
:class="{'text-success' : row.saverRule.status,
'text-dark' : !row.saverRule.status}"></i>
                </el-tooltip>

                <el-tooltip content="Almacenar
Datos">
                    <base-switch
@click="updateServerRuleStatus(row.saverRule)"
:value="row.saverRule.status" type="primary" on-
text="On" off-text="Off"></base-switch>

                </el-tooltip>

                <el-tooltip content="Borrar"
effect="light" :open-delay="300" placement="top">
                    <base-button type="danger"
icon size="sm" class="btn-link"
@click="deleteDevice(row)">
                    <i class="tim-icons icon-
simple-remove"></i>
                    </base-button>
```

```
                    </el-tooltip>

                </div>


            </el-table-column>

        </el-table>

      </card>
    </div>

  </div>
</template>

<script>
import { Table, TableColum, TableColumn } from
'element-ui';
import { Select, Option } from 'element-ui';
import BaseButton from
"../components/BaseButton.vue";
export default {
  middleware: 'authenticated',
  components: {
    [Table.name]: Table,
    [TableColumn.name]: TableColumn,
    [Option.name]: Option,
    [Select.name]: Select,
  },
  data(){
      return{
        templates: [],
        selectedIndexTemplate: null,
        newDevice: {
          name: "",
          dId: "",
          templateId: "",
          templateName: ""
        },
      };
  },
  mounted(){
    this.getTemplates();
  },

  methods:{
    async getTemplates() {
      const axiosHeaders = {
        headers: {
          token: this.$store.state.auth.token
        }
      };
      try {
        const res = await
this.$axios.get("/template", axiosHeaders);
        if (res.data.status == "success") {
          this.templates = res.data.data;
        }
      } catch (error) {
        this.$notify({
          type: "danger",
          icon: "tim-icons icon-alert-circle-exc",
          message: "Error getting templates..."
        });
        console.log(error);
        return;
      }
    },

    createNewDevice(){
      if (this.newDevice.name == "") {
        this.$notify({
          type: "warning",
          icon: "tim-icons icon-alert-circle-exc",
          message: " Device Name is Empty :("
        });
        return;
        }
        if (this.newDevice.dId == "") {
          this.$notify({
            type: "warning",
            icon: "tim-icons icon-alert-circle-exc",
            message: " Device ID is Empty :("
          });
          return;
        }
        if (this.selectedIndexTemplate == null) {
          this.$notify({
            type: "warning",
            icon: "tim-icons icon-alert-circle-exc",
            message: " Tempalte must be selected"
          });
          return;
        }
        const axiosHeaders = {
          headers: {
            token: this.$store.state.auth.token
          }
        };
        this.newDevice.templateId =
this.templates[this.selectedIndexTemplate]._id;
        this.newDevice.templateName =
this.templates[this.selectedIndexTemplate].name;

        const toSend = {
          newDevice: this.newDevice
        };
        this.$axios
          .post("/device", toSend, axiosHeaders)
          .then(res => {
            if (res.data.status == "success") {
              this.$store.dispatch("getDevices");
              this.newDevice.name = "";
              this.newDevice.dId = "";
              this.selectedIndexTemplate = null;
              this.$notify({
                type: "success",
                icon: "tim-icons icon-check-2",
                message: "Success! Device was added"
              });
              return;
            }
          })
          .catch(e => {
            if (
              e.response.data.status == "error" &&
              e.response.data.error.errors.dId.kind
== "unique"
            ) {
              this.$notify({
                type: "warning",
                icon: "tim-icons icon-alert-circle-
exc",
                message:
                  "The device is already registered
in the system. Try another device"
              });
              return;
            } else {
              this.showNotify("danger", "Error");
              return;
            }
```

```
        });
    },
    async getTemplates() {
      const axiosHeaders = {
        headers: {
          token: this.$store.state.auth.token
        }
      };
      try {
        const res = await
this.$axios.get("/template", axiosHeaders);
        if (res.data.status == "success") {
          this.templates = res.data.data;
        }
      } catch (error) {
        this.$notify({
          type: "danger",
          icon: "tim-icons icon-alert-circle-exc",
          message: "Error getting templates..."
        });
        console.log(error);
        return;
      }
      },
      deleteDevice(device){

        const axiosHeader = {
          headers: {
            token: this.$store.state.auth.token
          },
          params: {
            dId: device.dId
          }
        };

        this.$axios
          .delete("/device", axiosHeader)
          .then(res => {
            if (res.data.status == "success") {
              this.$notify({
                type: "success",
                icon: "tim-icons icon-check-2",
                message: device.name + " deleted!"
              });
            this.$store.dispatch("getDevices");
          }
          })
```

```
          .catch(e => {
            console.log(e);
            this.$notify({
              type: "danger",
              icon: "tim-icons icon-alert-circle-
exc",
              message: " Error deleting " +
device.name
            });
          });
      },
      updateServerRuleStatus(rule){
        var ruleCopy =
JSON.parse(JSON.stringify(rule));
        ruleCopy.status = !ruleCopy.status;
        const toSend = { rule: ruleCopy};
        const axiosHeaders = {
          headers: {
            token: this.$store.state.auth.token
          }
        };
         this.$axios
        .put("/saver-rule", toSend, axiosHeaders)
        .then(res => {
          if (res.data.status == "success") {
            this.$store.dispatch("getDevices");
            this.$notify({
              type: "success",
              icon: "tim-icons icon-check-2",
              message: " Device Saver Status
Updated"
            });
          }
          return;
        })
      }
  }

};
</script>
```

Mientras que el backend se encuentra en la carpeta “api/routes” bajo el nombre de “devices.js”

```
const express = require("express");
const router = express.Router();
const { checkAuth } =
require('../middlewares/authentication.js');
const axios = require("axios");

const auth = {
    auth: {
        username: 'admin',
        password:
process.env.EMQX_DEFAULT_APPLICATION_SECCRET
    }
};
//Importamos el modelo
import Device from '../models/device.js';
```

```
import SaverRule from
'../models/emqx_saver_rule.js';
import Template from '../models/template.js';
import AlarmRule from
'../models/emqx_alarm_rule.js';
import EmqxAuthRule from "../models/emqx_auth.js";

//******************
//**** A P I *******
//******************

//Obtenemos la lista de los dispositivos
router.get("/device", checkAuth , async (req, res)
=> {
    try {
```

```
        const userId = req.userData._id;
        //get devices
        var devices = await Device.find({ userId:
userId });
        devices =
JSON.parse(JSON.stringify(devices));
        //get saver rules
        const saverRules = await
getSaverRules(userId);
        //get templates
        const templates = await
getTemplates(userId);
        //get alarm rules
        const alarmRules = await
getAlarmRules(userId);
        devices.forEach((device, index) => {
            devices[index].saverRule =
saverRules.filter(saverRule => saverRule.dId ==
device.dId)[0];
            devices[index].template =
templates.filter(template => template._id ==
device.templateId)[0];
            devices[index].alarmRules =
alarmRules.filter(alarmRule => alarmRule.dId ==
device.dId);
        });
        const response = {
            status: "success",
            data: devices
        };

        res.json(response);

    } catch (error) {
        console.log("ERROR GETTING DEVICES");
        console.log(error);
        const response = {
            status: "error",
            error: error,
        }
        return res.status(500).json(response);

    }
});
//Crear nuevo dispositivo
router.post("/device", checkAuth , async (req, res)
=> {

    try {
        const userId = req.userData._id;
        var newDevice = req.body.newDevice;
        newDevice.userId = userId;
        newDevice.createdTime = Date.now();
        newDevice.dIdpassword = makeid(10);
        //console.log(newDevice);

        await createSaverRule(userId,newDevice.dId,
false);

        const device = await
Device.create(newDevice);

        await selectDevice(userId, newDevice.dId);

        const response = {
            status: "success"

        }
        return res.json(response);

    } catch (error) {
        console.log("ERROR CREATING NEW DEVICE");
        console.log(error);
        const response = {
            status: "error",
            error: error,
        }
        return res.status(500).json(response);

    }

});
//Eliminar el Dispositivo
router.delete("/device",checkAuth, async (req, res)
=> {
    try {
        const userId = req.userData._id;
        const dId = req.query.dId;

        await deleteSaverRule(dId);

        await deleteAllAlarmRules(userId, dId);

        await deleteMqttDeviceCredentials(dId);

        const result = await Device.deleteOne({
userId: userId, dId: dId });

        const devices = await Device.find({userId:
userId});

        if(devices.length >= 1){
            var found = false;
            devices.forEach(devices => {
                if(devices.selected == true){
                    found = true;
                }
            });

            if (!found){
                await Device.updateMany({ userId:
userId }, { selected: false });
                await Device.updateOne(
                    { userId: userId, dId:
devices[0].dId },
                    { selected: true }
                );
            }
        }

        const response = {
            status: "success",
            data: result
        };

        return res.json(response);

    } catch (error) {
        console.log("ERROR DELITING DEVICE");
        console.log(error);
        const response = {
            status: "error",
            error: error,
        }
        return res.status(500).json(response);
    }
});
//Actualizar el dispositivo
router.put("/device", checkAuth, async (req, res)
=> {

    try {
```

```
        const dId = req.body.dId;
        const userId = req.userData._id;

        if (await selectDevice(userId, dId)) {
            const response = {
                status: "success"
            };

            return res.json(response);
        } else {
            const response = {
                status: "error"
            };

            return res.json(response);
        }
    } catch (error) {
        console.log(error);
    }


});

//SAVER-RULE STATUS UPDATER
router.put('/saver-rule', checkAuth, async (req,
res) => {

    try {
        const rule = req.body.rule;

        //console.log(rule)

        await
updateSaverRuleStatus(rule.emqxRuleId, rule.status)

        const response = {
            status: "success"
        };

        res.json(response);
    } catch (error) {
        console.log(error);
    }
});

//**********************
//**** FUNCTIONS *******
//**********************

async function selectDevice(userId, dId) {
    try {
        const result = await Device.updateMany(
            { userId: userId },
            { selected: false }
        );

        const result2 = await Device.updateOne(
            { dId: dId, userId: userId },
            { selected: true }
        );

        return true;

    } catch (error) {
        console.log("ERROR IN 'selectDevice'
FUNCTION ");
        console.log(error);
        return false;
    }
}
```

```
/*
    SAVER RULES FUNCTIONS
*/

// get templates
async function getTemplates(userId) {

    try {
        const templates = await Template.find({
userId: userId });
        return templates;
    } catch (error) {
        return false;
    }

}

//get alarm rules funtion
async function getAlarmRules(userId){
    try {
        const rules = await AlarmRule.find({
userId: userId});
        return rules;
    } catch (error) {
        return "error";
    }
}

//get saver rules

async function getSaverRules(userId){

    try {
        const rules = await SaverRule.find({userId:
userId});
        return rules;
    } catch (error) {
        return false;
    }

}

//create saver rule
async function createSaverRule(userId, dId, status)
{

    try {
        const url = "http://" +
process.env.EMQX_HOST + ":8085/api/v4/rules";

        const topic = userId + "/" + dId +
"/+/sdata";

        const rawsql = "SELECT topic, payload FROM
\"" + topic + "\" WHERE payload.save = 1";

        var newRule = {
            rawsql: rawsql,
            actions: [
                {
                    name: "data_to_webserver",
                    params: {
                        $resource:
global.saverResource.id,
                        payload_tmpl: '{"userId":"'
+ userId +
'","payload":${payload},"topic":"${topic}"}'
                    }
                }
            ],
            description: "SAVER-RULE",
```

```
            enabled: status
        };
        //save rule in emqx - grabamos la regla en
emqx
        const res = await axios.post(url, newRule,
auth);

        if (res.status === 200 && res.data.data) {
            //console.log(res.data.data);
            await SaverRule.create({
                userId: userId,
                dId: dId,
                emqxRuleId: res.data.data.id,
                status: status
            });
            return true;
        } else {
            return false;
        }
    } catch (error) {
        console.log("Error creating saver rule");
        console.log(error);
        return false;
    }
}
//update saver rule
async function updateSaverRuleStatus(emqxRuleId,
status) {
    const url = "http://" + process.env.EMQX_HOST +
":8085/api/v4/rules/" + emqxRuleId;

    const newRule = {
        enabled: status
    };

    const res = await axios.put(url, newRule,
auth);

    if (res.status === 200 && res.data.data) {
        await SaverRule.updateOne({ emqxRuleId:
emqxRuleId }, { status: status });
        console.log("Saver Rule Status
Updated...".green);
        return {
            status: "success",
            action: "updated"
        };
    }
}
//delete saver rule
async function deleteSaverRule(dId) {
    try {
        const mongoRule = await SaverRule.findOne({
dId: dId });

        const url = "http://" +
process.env.EMQX_HOST + ":8085/api/v4/rules/" +
mongoRule.emqxRuleId;
        const emqxRule = await axios.delete(url,
auth);
        const deleted = await SaverRule.deleteOne({
dId: dId });

        return true;
    } catch (error) {
        console.log("Error deleting saver rule");
        console.log(error);
        return false;
    }
}
```

```
function makeid(length) {
    try {
        var result = "";
        var characters =
            "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl
mnopqrstuvwxyz0123456789";
        var charactersLength = characters.length;
        for (var i = 0; i < length; i++) {
            result +=
characters.charAt(Math.floor(Math.random() *
charactersLength));
        }
        return result;
    } catch (error) {
        console.log(error);
    }
}
//delete ALL alarm Rules...
async function deleteAllAlarmRules(userId, dId) {
    try {
        const rules = await AlarmRule.find({
userId: userId, dId: dId });

        if (rules.length > 0) {
            asyncForEach(rules, async rule => {
                const url = "http://" +
process.env.EMQX_HOST + ":8085/api/v4/rules/" +
rule.emqxRuleId;
                const res = await axios.delete(url,
auth);
            });

            await AlarmRule.deleteMany({ userId:
userId, dId: dId });
        }

        return true;
    } catch (error) {
        console.log(error);
        return "error";
    }
}

async function asyncForEach(array, callback) {
    for (let index = 0; index < array.length;
index++) {
        await callback(array[index], index, array);
    }
}

//delete ALL emqx device  auth rules
async function deleteMqttDeviceCredentials(dId) {
    try {
        await EmqxAuthRule.deleteMany({ dId: dId,
type: "device" });

        return true;
    } catch (error) {
        console.log(error);
        return false;
    }
}


module.exports = router;
```

Anexo 2

Creación del componente tabla

El algoritmo para la creación del widget “Number Chart”, y su backend es de la siguiente manera:

```
<template>

    <card type="chart">

        <template slot="header">

            <h5 class="card-category pull-
right">{{getTimeAgo((nowTime - time) / 1000)}} ago
</h5>



            <h5 class="card-category">{{
config.selectedDevice.name }} - {{
config.variableFullName }}</h5>

            <h3 class="card-title">
                <i class="fa "
:class="[config.icon, getIconColorClass()]" aria-
hidden="true"
                    style="font-size: 30px;"></i>
                <span>{{value.toFixed(config.decima
lPlaces)}} {{config.unit}}</span>
            </h3>
        </template>

        <div class="chart-area" style="height:
300px">
            <highchart style="height: 100%" v-
if="isMounted" :options="chartOptions"/>
        </div>

    </card>
</template>

<script>
    export default {
        name: 'rtnumberchart',
        props: ['config'],
        data() {
            return {
                receivedTime: 0,
                value: 0,
                time: Date.now(),
                nowTime: Date.now(),
                isMounted: false,
                topic:"",
                chartOptions: {
                    credits: {
                        enabled: false
                    },
                    chart: {
                        renderTo: 'container',
                        defaultSeriesType: 'line',
                        backgroundColor:
'rgba(0,0,0,0)',
                    },
                    title: {
                        text: ''
                    },
                    xAxis: {
                        type: 'datetime',
                        labels: {
                            style: {
                                color: '#d4d2d2'
                            }
                        }
                    },
                    yAxis: {
                        title: {
                            text: ''
                        },
                        labels: {
                            style: {
                                color: '#d4d2d2',
                                font: '11px
Trebuchet MS, Verdana, sans-serif'
                            }
                        }
                    },
                    plotOptions: {
                        series: {
                            label: {
                                connectorAllowed:
false,
                            },
                            pointStart: 2010
                        }
                    },
                    series: [{
                        name: '',
                        data: [],
                        color: "#e14eca"
                    },],
                    legend: {
                        itemStyle: {
                            color: '#d4d2d2'
                        }
                    },
                    responsive: {
                        rules: [{
                            condition: {
                                maxWidth: 500
                            },
                            chartOptions: {
                                legend: {
                                    layout:
'horizontal',
                                    align:
'center',
                                    verticalAlign:
'bottom'
                                }
                            }
                        }]
                    }
                },
```

```
            };
        },
        watch: {
            config: {
                immediate: true,
                deep: true,
                handler() {
                    setTimeout(() => {
                        this.value = 0;
                        this.chartOptions.series[0]
.data = [];
                        this.getChartData();
                        this.$nuxt.$off(this.topic
+ "/sdata", this.procesReceivedData);
                        this.topic =
this.config.userId + '/' +
this.config.selectedDevice.dId + '/' +
this.config.variable;
                        //console.log(this.topic);
                        this.$nuxt.$on(this.topic +
"/sdata", this.procesReceivedData);
                        this.chartOptions.series[0]
.name = this.config.variableFullName + " " +
this.config.unit;
                        this.updateColorClass();
                        window.dispatchEvent(new
Event('resize'));
                    }, 300);
                }}},
        mounted() {
            this.getNow();
            this.updateColorClass();
        },
        beforeDestroy() {
            this.$nuxt.$off(this.topic + "/sdata",
this.procesReceivedData);
        },
        methods: {
            updateColorClass() {

                var c = this.config.class;
                if (c == "success") {
                    this.chartOptions.series[0].col
or = "#00f2c3";
                }
                if (c == "primary") {
                    this.chartOptions.series[0].col
or = "#e14eca";
                }
                if (c == "warning") {
                    this.chartOptions.series[0].col
or = "#ff8d72";
                }
                if (c == "danger") {
                    this.chartOptions.series[0].col
or = "#fd5d93";
                }
                this.chartOptions.series[0].name =
this.config.variableFullName + " " +
this.config.unit;
            },
            getChartData() {
                if (this.config.demo) {
                    this.chartOptions.series[0].dat
a = [[1606659071668, 22], [1606659072668, 27],
[1606659073668, 32], [1606659074668, 7]];
                    this.isMounted = true;
                    return;
                }

                const axiosHeaders = {
                    headers: {
                        token:
this.$store.state.auth.token,
                    },

                    params: { dId:
this.config.selectedDevice.dId, variable:
this.config.variable, chartTimeAgo:
this.config.chartTimeAgo }
                }
                this.$axios.get("/get-small-charts-
data", axiosHeaders)
                    .then(res => {
                        this.chartOptions.series[0]
.data = [];
                        const data = res.data.data;
                        //Aqui formamos los datos
para el eje X e Y
                        data.forEach(element => {
                            var aux = []
                            aux.push(element.time +
(new Date().getTimezoneOffset() * 60 * 1000 * -1));
                            aux.push(element.value)
;
                            this.chartOptions.serie
s[0].data.push(aux);
                        });

                        this.isMounted = true;
                        return;
                    })
                    .catch(e => {
                        console.log(e)
                        return;
                    });
            },
            getIconColorClass() {
                if (this.config.class == "success")
{
                    return "text-success";
                }
                if (this.config.class == "primary")
{
                    return "text-primary";
                }
                if (this.config.class == "warning")
{
                    return "text-warning";
                }
                if (this.config.class == "danger")
{
                    return "text-danger";
                }
            },
            procesReceivedData(data) {
                this.time = Date.now();
                this.value = data.value;
                setTimeout(() => {
                    if(data.save==1){
                        this.getChartData();
                    }
                }, 1000);
            },
            getNow() {
                this.nowTime = Date.now();
                setTimeout(() => {
                    this.getNow();
                }, 1000);
            },
            getTimeAgo(seconds) {
                if (seconds < 0) {
```

```
                    seconds = 0;
                }
                if (seconds < 59) {
                    return seconds.toFixed() + "
secs";
                }
                if (seconds > 59 && seconds <=
3600) {
                    seconds = seconds / 60;
                    return seconds.toFixed() + "
min";
                }
```

```
if (seconds > 3600 && seconds <= 86400) {
                    seconds = seconds / 3600;
                    return seconds.toFixed() + "
hour";
                }
                if (seconds > 86400) {
                    seconds = seconds / 86400;
                    return seconds.toFixed() + "
day";
                }},}};
</script>
```

Mientras que la API que se deberá crear debe ser capaz de pedir datos en la BD

```
const express = require('express');
const router = express.Router();
const jwt = require("jsonwebtoken");
const { checkAuth } =
require('../middlewares/authentication.js');

//models import
import Data from '../models/data.js';

router.get('/get-small-charts-data', checkAuth,
async (req, res) => {
    try {
        const userId = req.userData._id;
        const chartTimeAgo =
req.query.chartTimeAgo;
        const dId = req.query.dId;
        const variable = req.query.variable;

        const timeAgoMs = Date.now() -
(chartTimeAgo * 60 * 1000);

        const data = await Data.find({ userId:
userId, dId: dId, variable: variable, "time": {
$gt: timeAgoMs } }).sort({ "time": 1 });
```

```
const response = {
            status: "success",
            data: data
        }

        return res.json(response)

    } catch (error) {

        console.log(error)

        const response = {
            status: "error",
            error: error
        }

        return res.json(response);

    }

});

module.exports = router
```

Anexo 3

Creación del componente indicador bolenao

El algoritmo para la creación del widget “Boolean Indicator”, es de la siguiente manera:

```
<template>
  <card>
    <div slot="header">
      <h5 class="card-category pull-
right">{{config.selectedDevice.dId}}</h5>
      <h4 class="card-title">
        {{ config.selectedDevice.name }} - {{
config.variableFullName }}
      </h4>
    </div>
    <i
      class="fa "
      :class="[config.icon, getIconColorClass()]"
      style="font-size: 30px"
    ></i>
  </card>
</template>

<script>
export default {
  props: ['config'],
  data() {
    return {
      value: false,
      topic: "",
      props: ['config']
    };
  },
  watch:  {
            config: {
                immediate: true,
                deep: true,
                handler() {
                    setTimeout(() => {
                        this.value = false;

                        this.$nuxt.$off(this.topic)
;
                        const topic =
this.config.userId + "/" +
this.config.selectedDevice.dId + "/" +
this.config.variable + "/sdata";
                        this.$nuxt.$on(topic,
this.processReceivedData);

                    }, 300);}
            }
        },
  mounted(){
    const topic = this.config.userId + "/" +
this.config.selectedDevice.dId + "/" +
this.config.variable + "/sdata";
    console.log(topic);
    this.$nuxt.$on(topic,
this.processReceivedData);
  },
  beforeDestroy(){
    this.$nuxt.$off(this.topic);
  },
  methods: {

    processReceivedData(data){
      try {
        this.value = data.value;
      } catch (error) {
        console.log(error);
      }
    },

    getIconColorClass() {
      if (!this.value) {
        return "text-dark";
      }

      if (this.config.class == "success") {
        return "text-success";
      }
      if (this.config.class == "primary") {
        return "text-primary";
      }
      if (this.config.class == "warning") {
        return "text-warning";
      }
      if (this.config.class == "danger") {
        return "text-danger";
      }
    }

  }
};
</script>
```

Anexo 4

Creación del componente cámara

El algoritmo para la creación del widget “Camera”, es de la siguiente manera:

```
<template>
  <card class="col-12">
      <template slot="header">
            <h4 class="card-title">
                <i class="fa "
:class="[config.icon, getIconColorClass()]" aria-
hidden="true" style="font-size: 20px;"></i>
                {{ config.selectedDevice.name }} -
{{ config.variableFullName }}
                <base-switch @click="value =
!value; sendValue()" :value="value" type="primary"
on-text="ON" off-text="OFF" style="margin-top:
10px;" class="pull-right"></base-switch>
            </h4>
        </template>
    <img
    :srcset="`${src} 320w,
             ${src} 480w,
             ${src} 800w`"

    sizes="(max-width: 320px) 280px,
            (max-width: 480px) 440px,
            800px"
    id="img" :src="`${src}`" />
    <br/>
    <br/>

    <h4 v-if="value" class="card-title">Estado del
flash

        <base-switch @click="flash_value =
!flash_value; sendFlashStatus()"
:value="flash_value" type="primary" on-text="ON"
off-text="OFF" style="margin-top: 2px;"
class="pull-right"></base-switch>
        <i class="pull-right fa "
:class="[config.flash_icon, getFlashColorClass()]"
aria-hidden="true" style="font-size: 25px;"></i>
    </h4>
  </card>
</template>

<script>
    export default{
        props: ['config'],

        data(){
            return{
                src: "img/ESPCAM_TEST.png",
                base64String: "",
                lastValue: "",
                value: false,
                flash_value: false,
                flash_status: false,
                isMounted: false,
                topic:"",
                polling: null,
                ReceivedDataStatus:false,
                checkIncomingData: null
            }
        },

        watch:{
            config:{
                immediate: true,
                deep: true,
                handler(){
                    setTimeout(() => {
                        this.src =
"img/ESPCAM_TEST.png"
                        this.value = false;
                        this.flash_value = false;
                        this.flash_status = false
                        this.$nuxt.$off(this.topic
+ "/espcam", this.procesReceivedData);
                        this.$nuxt.$off(this.topic
+ "/sdata", this.incomingFlashStatus);
                        this.topic =
this.config.userId + '/' +
this.config.selectedDevice.dId + '/' +
this.config.variable;
                        console.log(this.topic +
"/espcam")
                        this.$nuxt.$on(this.topic +
"/espcam", this.procesReceivedData);
                        this.$nuxt.$on(this.topic +
"/sdata", this.incomingFlashStatus);

                        window.dispatchEvent(new
Event('resize'));
                    }, 300);
                }
            }
        },

        mounted(){
         this.test();
         this.checkDataReceived();
        },

        beforeDestroy() {
            this.$nuxt.$off(this.topic + "/espcam",
this.procesReceivedData);
            this.$nuxt.$off(this.topic + "/sdata",
this.incomingFlashStatus);
            clearInterval(this.polling);
            clearInterval(this.checkIncomingData);

        },

        methods:{
            sendValue(){

                const toSend = {
                    topic: this.config.userId + '/'
+ this.config.selectedDevice.dId + '/' +
this.config.variable + '/actdata',
                    msg: {
                        value: this.value
```

```
                    }
                };
                $nuxt.$emit('mqtt-sender', toSend);
                if (!this.value) {
                    this.ReceivedDataStatus = false
                    const toSend2 = {
                        topic: this.config.userId +
'/' + this.config.selectedDevice.dId + '/' +
this.config.variable + '/actdata',
                        msg:{
                            flash_value:
this.value,
                        }
                    }
                    this.flash_value = false;
                    $nuxt.$emit('mqtt-sender',
toSend2);

                }
            },
            sendFlashStatus(){
                const toSend = {
                    topic: this.config.userId + '/'
+ this.config.selectedDevice.dId + '/' +
this.config.variable + '/actdata',
                    msg: {
                        flash_value:
this.flash_value
                    }
                }
                $nuxt.$emit('mqtt-sender', toSend);

            },
            procesReceivedData(data){
                 this.base64String = btoa(
                    String.fromCharCode.apply(null,
new Uint8Array(data))
                );
                if(!this.value){
                    this.src =
"img/ESPCAM_TEST.png"
                }
                if(this.value){
                    this.src =
"" + this.base64String;
                    this.ReceivedDataStatus = true
                }
                return;

            },
            incomingFlashStatus(data){
                this.flash_status =
data.flash_status;
                this.flash_value =
data.flash_status;
                this.value = data.camera_satus;
                if(!this.value){
                    this.src =
"img/ESPCAM_TEST.png"
                }
            },
            getIconColorClass() {
                if (!this.value){
                    return "text-danger";
                }
                if (this.config.class == "success")
{
                    return "text-success";
                }
                if (this.config.class == "primary")
{
                    return "text-primary";
                }
                if (this.config.class == "warning")
{
                    return "text-warning";
                }
                if (this.config.class == "danger")
{
                    return "text-danger";
                }
            },
            getFlashColorClass(){
                if (!this.flash_status){
                    return "text-dark";
                }
                if (this.config.class == "success")
{
                    return "text-success";
                }
                if (this.config.class == "primary")
{
                    return "text-primary";
                }
                if (this.config.class == "warning")
{
                    return "text-warning";
                }
                if (this.config.class == "danger")
{
                    return "text-danger";
                }
            },
            test(){

                this.polling = setInterval(() => {
                    if (!this.ReceivedDataStatus &&
this.value) {
                        this.src =
"img/warningCameraOff.png";
                        const toSend = {
                            topic:
this.config.userId + '/' +
this.config.selectedDevice.dId + '/' +
this.config.variable + '/actdata',
                            msg: {
                                value: true
                            }
                        };

                        $nuxt.$emit('mqtt-sender',
toSend);

                    }

                    if (!this.ReceivedDataStatus &&
!this.value) {
                        this.src =
"img/ESPCAM_TEST.png"
                    }
                }, 800);
            },
            checkDataReceived(){
                this.checkIncomingData =
setInterval(() => {
                    this.ReceivedDataStatus = false
                }, 3000);
            }
        }
    }
</script>
```

Anexo 5

Código del componente interruptor

El algoritmo consta de los siguientes componentes y funciones:

```
<template>
    <card>
        <template slot="header">
            <h5 class="card-category"> {{
config.selectedDevice.name }} - {{
config.variableFullName }}</h5>
            <h3 class="card-title">
                <i class="fa "
:class="[config.icon, getIconColorClass()]" aria-
hidden="true"
                    style="font-size: 30px;"></i>
                <base-switch @click="value =
!value; sendValue()" :value="value" type="primary"
on-text="ON" off-text="OFF" style="margin-top:
10px;" class="pull-right">
                </base-switch>

            </h3>

        </template>

    </card>

</template>

<script>
    export default {
        name: 'iotswitch',
        props: ['config'],

        data() {
            return {
                value: true,
                topic: "",
            };
        },
        watch: {
            config: {
                immediate: true,
                deep: true,
                handler() {
                    setTimeout(() => {
                        this.value = false;

                        this.$nuxt.$off(this.topic
+ "/sdata", this.procesReceivedData);

                        this.topic =
this.config.userId + '/' +
this.config.selectedDevice.dId + '/' +
this.config.variable;
                        console.log(this.topic +
"/sdata");
                        this.$nuxt.$on(this.topic +
"/sdata", this.procesReceivedData);
                        window.dispatchEvent(new
Event('resize'));
                    }, 300);
                }}
        },
        mounted() {

        },
        beforeDestroy() {
            this.$nuxt.$off(this.topic + "/sdata",
this.procesReceivedData);
        },
        methods: {

            procesReceivedData(data){
                console.log(data);
                this.value = data.value;
                return;
            },

           sendValue(){
                const toSend = {
                    topic: this.config.userId + '/'
+ this.config.selectedDevice.dId + '/' +
this.config.variable + '/actdata',
                    msg: {
                        value: this.value
                    }
                };
                console.log(toSend.topic);
                $nuxt.$emit('mqtt-sender', toSend);
            },
            getIconColorClass() {
                if (!this.value){
                    return "text-dark";
                }
                if (this.config.class == "success")
{
                    return "text-success";
                }
                if (this.config.class == "primary")
{
                    return "text-primary";
                }
                if (this.config.class == "warning")
{
                    return "text-warning";
                }
                if (this.config.class == "danger")
{
                    return "text-danger";
                }
            },
        }
    };
</script>
```

Anexo 6

Código del componente botón

El algoritmo para la creación del widget “button” se lo realiza de la siguiente forma:

```
<template>
    <card>
        <div slot="header">
            <h4 class="card-title" v-
if="config.variableFullName ==
''">{{config.selectedDevice.name}}-Encender</h4>
            <h4 class="card-title" v-
if="config.variableFullName !=
''">{{config.selectedDevice.name}}-
{{config.variableFullName}}</h4>
        </div>
        <i class="fa " :class="[config.icon,
getIconColorClass()]" style="font-size: 30px"></i>
        <base-button v-if="config.text == ''"
@click="sendValue()" :type="config.class"
class="mb-3 pull-right" size="lg">Off</base-button>
        <base-button v-if="config.text != ''"
@click="sendValue()" :type="config.class"
class="mb-3 pull-right"
size="lg">{{config.text}}</base-button>

    </card>
</template>

<script>
export default {
    props: ['config'],
    data(){

        return{
            sending: false,
        }
    },
    mounted() {

    },
    methods:{
        sendValue(){
           this.sending = true;
           setTimeout(() => {
               this.sending = false;
           }, 500);

           const toSend={
               topic: this.config.userId + "/" +
this.config.selectedDevice.dId + "/" +
this.config.variable + "/actdata",
               msg:{
                   value: this.config.message
               }
            };
            console.log(toSend);
            this.$nuxt.$emit('mqtt-sender', toSend)

        },
        getIconColorClass(){
            if(!this.sending){
                return "text-dark"
            }

            if(this.config.class == "success"){
                return "text-success"
            }
            if(this.config.class == "primary"){
                return "text-primary"
            }
            if(this.config.class == "warning"){
                return "text-warning"
            }
            if(this.config.class == "danger"){
                return "text-danger"
            }
        }
    }
}
</script>
```

Anexo 7

Código del componente indicador gauge

El algoritmo para la creación del widget “indicador gauge” se lo realiza de la siguiente forma:

```
<template>
    <card type="chart">
        <template slot="header">

            <h5 class="card-category pull-
right">{{getTimeAgo((nowTime - time) / 1000)}} ago
</h5>



            <h4 class="card-title">
                <i class="fa "
:class="[config.icon, getIconColorClass()]" aria-
hidden="true" style="font-size: 20px;"></i>
                {{ config.selectedDevice.name }} -
{{ config.variableFullName }}

            </h4>

        </template>

        <div style="position: relative; height:
100px">

            <JqxGauge ref="myGauge"
class="container"
                      :animationDuration="1100"
                      :ranges="ranges" :cap="cap"
:border="border"
                      :ticksMinor="ticksMinor"
:ticksMajor="ticksMajor"
                      :labels="labels"
:pointer="pointer"
                      :max="config.max_value"
                      :radius="200" :width="400"
:height="400"
                      >
            </JqxGauge>

            <JqxSlider v-if="config.control ==
true" ref="mySlider" style="position: absolute;
top: 305%; left: 26%"
                      @slideStart="onSlideStart($even
t)" @slideEnd="onSlideEnd($event)"
@change="onValueChanging($event)"
                      :width="250" :value="0" :min="-
(config.slider_mx_value)"
:max="config.slider_mx_value"
:step="config.sliderStep"
                      :ticksFrequency="(config.slider
_mx_value/4)" :mode="'fixed'" :showButtons="true"
:tooltip="true"
                      :tooltipFormatFunction="tooltip
FormatFunction">
            </JqxSlider>

            <div ref="gaugeValue"
class="gaugeValue"></div>
        </div>
    </card>
</template>

<script>
    import JqxGauge from "jqwidgets-
scripts/jqwidgets-vue/vue_jqxgauge.vue";
    import JqxSlider from "jqwidgets-
scripts/jqwidgets-vue/vue_jqxslider.vue";

    var slideEndStatus = false;
    var slideStatus = false;
    export default {
        props: ['config'],
        components: {
            JqxGauge,
            JqxSlider
        },
        data() {

            var endvalue0 =
this.config.green_min_value;
            var endvalue1 =
this.config.green_value;
            var endvalue2 =
this.config.yellow_min_value;
            var endvalue3 =
this.config.yellow_value;
            var endvalue4 =
this.config.red_min_value;
            var endvalue5 = this.config.red_value;
            var Minorticks =
this.config.Minor_interval;
            var Majorticks =
this.config.Major_interval;
            return {

                isMounted: false,
                changeValue: false,
                value:0,
                topic:"",
                time: Date.now(),
                nowTime: Date.now(),
                ticksMinor: { interval: Minorticks,
size: '5%' },
                ticksMajor: { interval: Majorticks,
size: '10%' },
                labels: { position: 'outside',
interval: Majorticks },
                cap: { size: '5%', style: { fill:
'#2e79bb', stroke: '#2e79bb' } },
                border: { style: { fill: '#8e9495',
stroke: '#7b8384', 'stroke-width': 1 } },
                pointer: { style: { fill: '#2e79bb'
}, width: 5 },
                ranges: [
                    { startValue: endvalue0,
endValue: endvalue1, style: { fill: '#4cb848',
```

```
stroke: '#4cb848' }, startDistance: 0, endDistance:
0 },
                    { startValue: endvalue2,
endValue: endvalue3, style: { fill: '#fad00b',
stroke: '#fad00b' }, startDistance: 0, endDistance:
0 },
                    { startValue: endvalue4,
endValue: endvalue5, style: { fill: '#e53d37',
stroke: '#e53d37' }, startDistance: 0, endDistance:
0 }
                ]
            }
        },
        watch:{
            config:{
                immediate: true,
                deep: true,
                handler() {
                    setTimeout(() => {
                        try {
                            this.value = 0;
                            this.$nuxt.$off(this.to
pic + "/sdata", this.procesReceivedData);
                            this.topic =
this.config.userId + '/' +
this.config.selectedDevice.dId + '/' +
this.config.variable;
                            console.log(this.topic+
"/sdata");
                            this.$nuxt.$on(this.top
ic + "/sdata", this.procesReceivedData);
                            this.$refs.gaugeValue.i
nnerHTML = this.value + " " + this.config.unit;
                            this.$refs.myGauge.valu
e = this.value;
                            if(this.config.control)
{
                                this.$refs.mySlider
.setValue(0);
                                this.config.variabl
eType = "bidi"
                            }
                            if(!this.config.control
){
                                this.config.variabl
eType = "input"
                            }


                            window.dispatchEvent(ne
w Event('resize'));
                        } catch (error) {

                        }


                    }, 300);
                }
            }
        },

        mounted: function () {

            this.getNow();

        },

        beforeDestroy(){
            this.$nuxt.$off(this.topic + "/sdata",
this.procesReceivedData);
        },
```

```
        methods: {

            tooltipFormatFunction: function (value)
{
                var newValue = Math.round(value);
                return newValue;
            },

            onSlideStart(event) {
                slideStatus=true;
            },

            onValueChanging: function (event)
{

                if (this.config.demo){
                    this.$refs.gaugeValue.innerHTML
= Math.round(event.args.value) + "
"+this.config.unit;
                }
                var valueC =
event.args.value;
                if((event.args.type == 'mouse' ||
event.args.type == 'keyboard')&&
slideStatus==false){
                    if(valueC != -0.1){
                        if(valueC > 0){
                            const toSend={
                                topic:
this.config.userId + "/" +
this.config.selectedDevice.dId + "/" +
this.config.variable + "/actdata",
                                msg:{
                                    value:
this.config.sliderStep
                                }
                            };
                            this.$nuxt.$emit('mqtt-
sender', toSend)
                        }
                        if(valueC < 0){
                            const toSend={
                                topic:
this.config.userId + "/" +
this.config.selectedDevice.dId + "/" +
this.config.variable + "/actdata",
                                msg:{
                                    value: -
(this.config.sliderStep)
                                }
                            };
                            this.$nuxt.$emit('mqtt-
sender', toSend)
                        }
                    }
                    setTimeout(() => {
                        this.$refs.mySlider.val(-
0.1);
                    }, 6000);
                }
            },
            onSlideEnd: function (event) {
                slideStatus = false;
                slideEndStatus = !slideEndStatus;
               let val = this.$refs.mySlider.value;
                const toSend={
                    topic: this.config.userId + "/"
+ this.config.selectedDevice.dId + "/" +
this.config.variable + "/actdata",
                    msg:{
```

```
                        value: val
                    }
                };
                this.$nuxt.$emit('mqtt-sender',
toSend)

                setTimeout(() => {
                    if (slideEndStatus == false) {
                        this.$refs.mySlider.val(0);

                    }
                    if (slideEndStatus == true){
                        this.$refs.mySlider.val(0.1
);
                    }
                }, 500);
            },
            procesReceivedData(data){
                try {
                  this.time =
Date.now();
                    this.value = data.value;
                    if(this.value != null ||
undefined){

                        this.$refs.myGauge.value =
this.value;
                        this.$refs.gaugeValue.inner
HTML = this.value + " " + this.config.unit;
                    }

                    return;
                }

            },
            getNow() {
                this.nowTime = Date.now();
                setTimeout(() => {
                    this.getNow();
                }, 1000);
            },
            getTimeAgo(seconds) {
                if (seconds < 0) {
                    seconds = 0;
                }
                if (seconds < 59) {
                    return seconds.toFixed() + "
secs";
                }
                if (seconds > 59 && seconds <=
3600) {
                    seconds = seconds / 60;
                    return seconds.toFixed() + "
min";
                }
                if (seconds > 3600 && seconds <=
86400) {
                    seconds = seconds / 3600;
                    return seconds.toFixed() + "
hour";
                }
                if (seconds > 86400) {
                    seconds = seconds / 86400;
                    return seconds.toFixed() + "
day";
                }
            },

            getIconColorClass() {
                if (this.config.class == "success")
{
                    return "text-success";
                }
                if (this.config.class == "primary")
{
                    return "text-primary";
                }
                if (this.config.class == "warning")
{
                    return "text-warning";
                }
                if (this.config.class == "danger")
{
                    return "text-danger";
                }
            },
        }
    }
</script>
<style>
    body, html {
        height: 100%;
    }
    .gaugeValue {
        background-image: -webkit-gradient(linear,
50% 0%, 50% 100%, color-stop(0%, #fafafa), color-
stop(100%, #f3f3f3));
        background-image: -webkit-linear-
gradient(#fafafa, #f3f3f3);
        background-image: -moz-linear-
gradient(#fafafa, #f3f3f3);
        background-image: -o-linear-
gradient(#fafafa, #f3f3f3);
        background-image: linear-gradient(#fafafa,
#f3f3f3);
        -webkit-border-radius: 3px;
        -moz-border-radius: 3px;
        border-radius: 3px;
        -webkit-box-shadow: 0 0 50px rgba(0, 0, 0,
0.2);
        -moz-box-shadow: 0 0 50px rgba(0, 0, 0,
0.2);
        box-shadow: 0 0 50px rgba(0, 0, 0, 0.2);
        padding: 10px;
        position: absolute;
        top: 250%;
        left: 35%;
        font-family: Sans-Serif;
        text-align: center;
        font-size: 18px;
        width: 150px;
    }
</style>
```

Anexo 8

Código del chatbot

El algoritmo para la creación del chatbot consta de las siguientes API's y funciones para el procesamiento de los mensajes:

```
const express = require('express');
const router = express.Router();
const { checkAuth } =
require("../middlewares/authentication.js");
const fs = require('fs');
const qrcode = require('qrcode-terminal');
const { Client, List, Buttons, MessageMedia,
LocalAuth } = require('whatsapp-web.js');

const { createFsmManager } =
require("@patrikstas/finite-state-machine");
const { chatStates } =
require('../stateMachine/statesDifinend');
const { createMongoStorage, mongoClientConnection }
= require('../stateMachine/mongoConnection');
const { mySetInterval, myClearInterval } =
require("../stateMachine/timersFunctions");



import User from "../models/user.js";
import Device from "../models/device.js";
import deviceAlert from "../models/deviceAlerts"
import AlarmRule from
"../models/emqx_alarm_rule.js";

// Use the saved values
const client = new Client({
    restartOnAuthFail: true,
    takeoverOnConflict: true,
    puppeteer: {
        headless: true,
        args: [
            '--no-sandbox',
            '--disable-setuid-sandbox',
            '--disable-dev-shm-usage',
            '--disable-accelerated-2d-canvas',
            '--no-first-run',
            '--no-zygote',
            '--single-process',
            '--disable-gpu'
        ],
    },
    authStrategy: new LocalAuth()
});


client.initialize();

client.on('qr', qr => {
    qrcode.generate(qr, { small: true });
});



client.on('ready', () => {
    console.log('Client is ready!');
    let info = client.info;
    setTimeout(() => {
        client.sendMessage(info.me.user + '@c.us',
`Hola ⚾ *_${info.pushname}_* tu servidor API💻 se
ha iniciado correctamente✔\n▪Recuerda seguir las
instrucciones de uso📄, para recordarlas envía a
este chat la palabra *_instrucciones_*\n▪Dudas e
inquietudes❓ envía a este chat la palabra
*_ayuda_*`);
    }, 1500);

});



client.on('message', async msg => {

    let chat = await msg.getChat();
    let phonenum = number(msg.from);
    var user = await User.findOne({ phone: phonenum
});

    chat.sendStateTyping();

    if (!user) {
        setTimeout(() => {
            client.sendMessage(msg.from, '❌
*ERROR* ❌\nUsuario *NO REGISTRADO* para más
información póngase en contacto📞 con el técnico a
cargo de sus equipos📡');
            console.log(msg.from);
            chat.clearState();
        }, 1500);
        return;
    }


    try {
        let rowsOptions = [
            { id: 'lst3', title: 'Agregar otro
número para notificar las alarmas 📄' }
        ]


        let hasNumberAdded = await
deviceAlert.find({createdBy: phonenum});

        if (hasNumberAdded.length >= 1) {
            rowsOptions.push({ id: 'lst4', title:
'Eliminar números agregados 📄🗑️'});
        }

        let sections = [{ rows: rowsOptions}];
        let list = new List('⚾ Bienvenido al
sistema 👹 de telemetría *' + user.name + '* ',
'Menu de Opciones', sections, 'Seleccione una
opción', 'footer');
        var currentState = await
wpChatStates(msg.from);
```

```
        console.log(currentState);

        if (msg.type != "list_response" &&
currentState == "off"){
            setTimeout(() => {

                client.sendMessage(msg.from, list);
                chat.clearState();
            }, 1500);
        }
        else if (currentState == "option_3a"){
            setTimeout(async () => {
                var response = await
wpChatStates(msg.from, "next_op3", msg.body);
                client.sendMessage(msg.from,
response);
                chat.clearState();
            }, 1500);

        }
        else if (currentState == "option_3b"){
            setTimeout(async () => {
                var response = await
wpChatStates(msg.from, "next_op3", msg.body);
                client.sendMessage(msg.from,
response);
                console.log(response, 'Opcion 3B');
                chat.clearState();
            }, 1500);

        }
        else if (currentState == "option_3c"){
            if (msg.body == "No" || msg.body ==
"no") {
                var oldMessages = await
chat.fetchMessages({ limit: 30 });
                var myMessages =
oldMessages.find(element => element.body ===
'Hola');
                setTimeout(async () => {
                    client.sendMessage(msg.from,
'text', { quotedMessageId:
myMessages.id._serialized })

                    chat.clearState();
                }, 1500);
                return

            }
            var response = await
wpChatStates(msg.from, "next_op3", msg.body);

            setTimeout(async () => {
                client.sendMessage(msg.from,
response);
                chat.clearState();
            }, 1500);
        }
        else if (msg.type === "list_response"){

            switch (msg.selectedRowId) {


                case "lst3":

                    var response = await
wpChatStates(msg.from, "next_op3");

                    let createdRows = [];
                    var devices = await
Device.find({ userId: user._id });
                    devices =
JSON.parse(JSON.stringify(devices));

                    devices.forEach((device, index)
=> {
                        createdRows.push({ id:
`${devices[index].dId}`, title:
`${devices[index].name}`});
                    });
                    let sections = [{rows:
createdRows}];

                    let list = new List('Selecione
uno de los dispositivos el cual desea agragar un
nombre', 'Lista de dispositivos', sections,
'Seleccione una opción', 'footer');

                    setTimeout(() => {
                        client.sendMessage(msg.from
, list );
                        chat.clearState();
                    }, 1000);
                    return

                case "lst4":
                    var response = await
wpChatStates(msg.from, "next_op4");
                    let hasNumberAdded1 = await
deviceAlert.find({ createdBy: phonenum });
                    var numbersAdded =
JSON.parse(JSON.stringify(hasNumberAdded1));
                    console.log(numbersAdded);
                    let createdRows1 = [];
                    numbersAdded.forEach((device,
index) => {
                        createdRows1.push({ id:
`${numbersAdded[index].alarmNumber}`,
title:`${numbersAdded[index].deviceName}`,
description: `Persona a notificar
${numbersAdded[index].userName} celular:
0${numbersAdded[index].number}`});
                    });
                    createdRows1.push({ id: 'c1',
title: 'Cancelar ❌'})
                    let sections1 = [{ rows:
createdRows1 }];
                    let list1 = new List('Selecione
un contacto a eliminar', 'Lista de personas',
sections1, 'Seleccione una opción', 'footer');
                    setTimeout(() => {
                        client.sendMessage(msg.from
, list1);
                        chat.clearState();
                    }, 1000);
                    return
            }
            var device = "";
            var hasNumberAdded1="";
            if (msg.selectedRowId !== "c1") {
                device = await Device.findOne({
dId: msg.selectedRowId })
                hasNumberAdded1 = await
deviceAlert.findOne({ alarmNumber:
msg.selectedRowId });
            }
            if (!hasNumberAdded1 && currentState ==
'option_4' && msg.selectedRowId !== "c1") {
                console.log('no se ha encontrado
datos al respecto');
```

```
                return
            }

            if (!device && currentState !==
'option_4'){
                console.log('no se ha encontrado un
device');
                return
            }

            if (currentState !== 'option_4'){
                switch (msg.selectedRowId) {
                    case device.dId:
                        //Save the device ID in the
chat_form collection device.dId
                        var response = await
wpChatStates(msg.from, "next_op3", device);
                        setTimeout(() => {
                            client.sendMessage(msg.
from, response);
                            chat.clearState();
                        }, 1000);
                        return


                }
            }

            if (currentState === 'option_4'){

                switch (msg.selectedRowId) {

                    case "c1":
                        var response = await
wpChatStates(msg.from, "disable");
                        console.log(response);

                        setTimeout(() => {
                            var response =
"Solicitud Cancelada" + waveMessage()
                            client.sendMessage(msg.
from, response);
                            chat.clearState();
                        }, 1000);
                        return


                    case
hasNumberAdded1.alarmNumber.toString():

                        var response = await
wpChatStates(msg.from, "next_op4",
hasNumberAdded1.alarmNumber);
                        setTimeout(() => {
                            client.sendMessage(msg.
from, response);
                            chat.clearState();
                        }, 1000);
                        return
                }
            }

            chat.clearState();
        }

    } catch (error) {
        console.log(error)
    }

});
```

```
client.on('message_create', async msg => {
    if (msg.fromMe) {
    }
});

client.on('message_ack', (msg, ack) =>{
    if (ack == 3) {

    }
});

router.post("/messageAlarmWp", async (req, res) =>
{
    try {
        var user = await User.findOne({ _id:
req.body.userId });

        if (!user) {
            res.sendStatus(500);
        }
        res.sendStatus(200);

        var number = user.phone.toString();
        var wpNumber =
phoneNumberFormatter(number);

        var msgToSend = msgToSend = `🏳 Hola
*${user.name}* 👌, la regla cuando la variable
*_${req.body.variableFullName}_* del dispositivo
*_${req.body.deviceName}_*, sea
*${req.body.condition}* *${req.body.value}
${req.body.unit}* se ha cumplido😉, el valor actual
es de *${req.body.payload.value}${req.body.unit}*
🤔`;

        if (req.body.valueMn != null) {
            msgToSend = `🏳 Hola *${user.name}*
👌, la regla cuando la variable
*_${req.body.variableFullName}_* del dispositivo
*_${req.body.deviceName}_*, este
*${req.body.condition}* de *min:
${req.body.valueMn} ${req.body.unit}* a *max:
${req.body.value} ${req.body.unit}* se ha
cumplido😉, el valor actual es de
*${req.body.payload.value}${req.body.unit}* 🤔`;
        }
        if (req.body.unit == undefined ||
req.body.unit == 'undefined'){
            msgToSend = msgToSend = `🏳 Hola
*${user.name}* 👌, la regla cuando la variable
*_${req.body.variableFullName}_* del dispositivo
*_${req.body.deviceName}_*🔔 , sea
*${req.body.condition}* *${req.body.value}* se ha
cumplido😉, el valor actual es de
*${req.body.payload.value}* 🤔`;
        }
        setTimeout(async () => {
            var photo = await AlarmRule.findOne({
emqxRuleId: req.body.emqxRuleId });
            var media1 = new
MessageMedia("image/jpeg",photo.picture_data)
            client.sendMessage(wpNumber, `Imagen
tomada en la fecha *${photo.picture_date_zone}*`, {
media: media1});
        }, 750);

        client.sendMessage(wpNumber, msgToSend);
```

```
        var moreNumbers = await
findMoreNumbers(req.body.dId.toString());

        console.log(moreNumbers);

        moreNumbers[0].forEach((t,index) => {

            setTimeout(async function () {

                var newMessage = `🏳 Hola
*${moreNumbers[1][index]}* 👌, esta es una *alerta*
creada por *${user.name}*, ya que`
                newMessage += msgToSend.replace(`🏳
Hola *${user.name}* 👌,`, " ");
                newMessage += `\n Por favor ponerse
en contacto con *${user.name}* para mayor
información` + waveMessage();
                client.sendMessage(moreNumbers[0][i
ndex], newMessage);
                setTimeout(async () => {
                    var photo = await
AlarmRule.findOne({ emqxRuleId: req.body.emqxRuleId
});
                    var media1 = new
MessageMedia("image/jpeg",photo.picture_data)
                    client.sendMessage(moreNumbers[
0][index], `Imagen tomada en la fecha
*${photo.picture_date_zone}*` ,{media: media1});
                }, (index * 1000) + 750);

            }, index * 1000);
        });
        retur
    } catch (error) {
        console.log(error);
    }});

// FUNCTIONS

async function findMoreNumbers(dId) {

    var moreNumbers = await deviceAlert.find({
deviceDid: dId });
    var sendNumbers = [];
    var sendNames = [];

    moreNumbers.forEach( (t,index) => {
        var wpFormat =
phoneNumberFormatter(moreNumbers[index].number.toSt
ring());
        sendNumbers.push(wpFormat);
        sendNames.push(moreNumbers[index].userName)
;
    });

    return [sendNumbers, sendNames]
}

function number(number) {

    let formatted = number.replace(/\D/g, '');

    if (formatted.startsWith('593')) {
        formatted = '0' + formatted.substr(3);
    }

    if (formatted.endsWith('@c.us')) {
        formatted -= '@c.us';
    }
    return formatted;
}

function phoneNumberFormatter(number) {

    let formatted = number.replace(/\D/g, '');

    if (formatted.startsWith('9')) {
        formatted = '593' + formatted;
    }

    if (!formatted.endsWith('@c.us')) {
        formatted += '@c.us';
    }

    return formatted;
}

//Create a number ID
function makeid(length) {
    try {
        var result = "";
        var characters =
            "0123456789";
        var charactersLength = characters.length;
        for (var i = 0; i < length; i++) {
            result +=
characters.charAt(Math.floor(Math.random() *
charactersLength));
        }
        return result;
    } catch (error) {
        console.log(error);
    }}

function phoneNumberVerification(inputtxt) {
    var phoneno = /^\d{10}$/;
    if (inputtxt.match(phoneno)){

        return true;
    }
    else {

        return false;
    }
}

function waveMessage() {
    var today = new Date()
    var curHr = today.getHours()
    var response = ''
    if (curHr < 12) {
        response = "\n*Que tenga un buen día 👌*"
        return response;
    } else if (curHr < 18) {
        response = "\n*Que tenga una buena tarde
👌*"
        return response;
    } else {
        response = "\n*Que tenga una buena noche
👌*"
        return response;
    }}

async function wpChatStates(id, transition,
message){
    const strategy = await createMongoStorage();
    const fsmManager = createFsmManager(strategy,
chatStates);
```

```
    const fsmMongoCollection = await
mongoClientConnection();
    var user='';
    try {
        user = await fsmManager.fsmLoad(id);
    } catch (error) {
        user = await fsmManager.fsmCreate(id);
    }
    var state = await user.getState();

    if (typeof transition === 'undefined') {

        return state
    }

    if (typeof transition === "disable") {
        await user.doTransition("disable");
        return
    }

    await user.doTransition(transition);

    var stateAfterTransition = await
user.getState();

    if (stateAfterTransition === "option_3"){

    } else if (stateAfterTransition ===
"option_3a"){

        var response = await
fsmMongoCollection.updateOne({ fsmId: id }, { $set:
{ "fsmData.deviceNameResponse": message.name,
"fsmData.deviceIdResponse": message.dId } });
        return 'Ingrese el _nombre_ de la persona
\nEjemplo: *Carlos*'
    }else if (stateAfterTransition ===
"option_3b"){
        var response = await
fsmMongoCollection.updateOne({ fsmId: id }, { $set:
{ "fsmData.nameResponse": message } });
        return 'Ingrese el _número_ de la persona
🕵 a notificar\n Ejemplo: *0987654321*'

    }else if (stateAfterTransition ===
"option_3c"){
        var numberAdded =
phoneNumberVerification(message);
        if (!numberAdded) {
            var response = await
fsmMongoCollection.updateOne({ fsmId: id }, { $set:
{ "fsmData.state": "option_3b" } });
            return `Por favor ingrese un número
válido`
        }
        var response = await
fsmMongoCollection.updateOne({ fsmId: id }, { $set:
{ "fsmData.numberResponse": message } });
        const queryFsmCollection = await
fsmMongoCollection.find({ fsmId: id }).toArray();
        let device =
queryFsmCollection[0].fsmData.deviceNameResponse;
        let name =
queryFsmCollection[0].fsmData.nameResponse;
        let phone =
queryFsmCollection[0].fsmData.numberResponse;

        return `Son estos datos correctos?
        *📢Dispositivo:*    _${device}_
        *📲Nombre:*         _${name}_
        *📱Celular:*          _${phone}_
Ingrese *Si* para guardar los datos y *No* para
repetir el proceso`

    } else if (stateAfterTransition === "check") {

        if (message === "Si" || "si" || "SI" ||
"sI") {
            const queryFsmCollection = await
fsmMongoCollection.find({ fsmId: id }).toArray();
            let device =
queryFsmCollection[0].fsmData.deviceNameResponse;
            let deviceId =
queryFsmCollection[0].fsmData.deviceIdResponse;
            let name =
queryFsmCollection[0].fsmData.nameResponse;
            let phone =
queryFsmCollection[0].fsmData.numberResponse;
            //save data
            var save = await deviceAlert.create({
                alarmNumber: makeid(5),
                deviceName: device,
                deviceDid: deviceId,
                userName: name,
                number: phone,
                createdBy: number(id)
            })

            await user.doTransition("next_op3");
            var wave = waveMessage();
            return `Desde ahora las alarmas
generadas por el dispositivo 📢 *${device}* le
notificara a usted y a *${name}* 🕵 quien fue
registrado con el siguiente número *${phone}* 📱
${wave}`
        }
    } else if (stateAfterTransition ===
"option_4a"){
        console.log(message);
        const query = await deviceAlert.deleteOne({
alarmNumber: message });
        await user.doTransition("next_op4");
        var wave = waveMessage();
        return `Se ha eliminado correctamente el
contacto asociado ${wave}`
    }}

module.exports = router
```

Anexo 9

Descripción del transmisor FM RVR tex100

Especificaciones técnicas, indicadores y puertos del transmisor.

6. External Description

This chapter describes the elements of the front and rear panels of the TEX100.

6.1 Front Panel (TEX100/S stereo version)

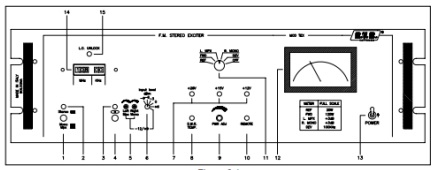


Figure 6.1

[1]	STEREO/MONO	MPX Selects STEREO or MONO/MPX operation
[2]	STEREO LED	When lit, indicates the operation of the stereo encoder
[3]	19KHz LED	When lit, indicates the 19KHz pilot tone presence
[4]	MODE SELECTOR	Selects the enabled/disabled of the pilot tone inside the MPX signal
[5]	L/MPX & R/MONO LEVEL	L/MPX and R/MONO input level adjustable from -12 to +9dBm; this is possible if the input level [6] switch is completly turned clockwise
[6]	INPUT LEVEL	Input signal attenuator adjustable in 5 steps from -9 to +6 dBm, plus one position for continuous regulation
[7]	SUPPLY LED	When lit, indicates the presence of internal operating voltages
[8]	SWR + TEMP	Indicates that the reflected power exceeds 10W
[9]	PWR ADJ	Multi-turn trimmer to regulate the power output of the exciter. AGC maintains constant the output level set by this control
[10]	REMOTE	When lit, indicates that the exciter has been shutdown by remote control
[11]	MEAS. SELECTOR	The measurement made by the meter corresponds to the position of this selector
[12]	METER	Analog meter used to display the following operating parameters of the exciter:
		Direct power f.s. 125W
		Reflected power f.s. 25W
		Deviation f.s. 100KHz
		Right channel input level f.s. +3dB
		Left channel input level f.s. +3dB
[13]	POWER	Switch ON/OFF selector
[14]	MHz/KHz	Rotatory frequency selector
[15]	L.O. UNLOCK	When lit, indicates that the VCO is not locked to the reference frequency. The output power will drop zero in this condition

6.3 Rear Panel

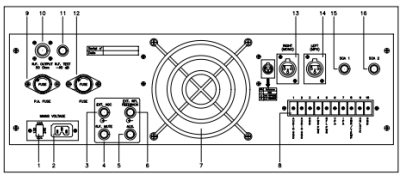
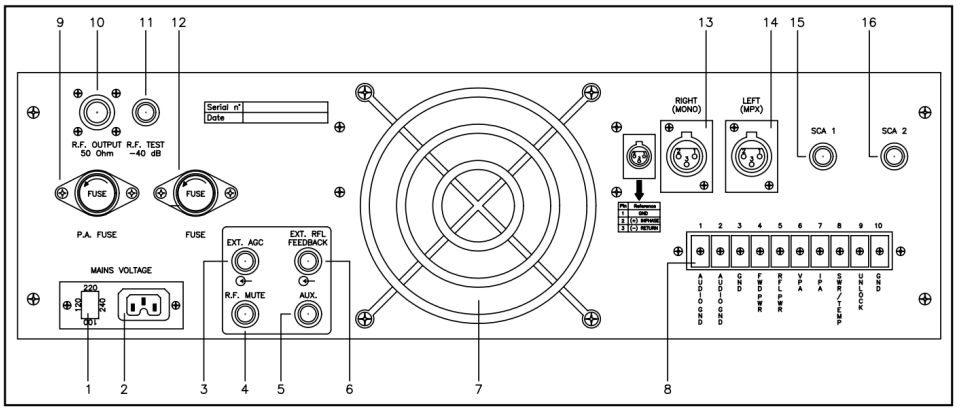


Figure 6.3

[1]	FUSE BLOCK	Fuse block and line voltage selector. Use a small screwdriver to change the fuse or the line voltage. Rotate the block and position it so that the arrow corresponds to the working voltage wished
[2]	PLUG	Line power connector
[3]	REMOTE 1	BNC connector, extenal AGC input
[4]	REMOTE 2	BNC connector, extenal reflected feedback input
[5]	EXT REF 1KHz	External 1KHz reference (optional)
[6]	REMOTE 3	BNC connector. Connecting the central conductor to ground will cause the Rf output power level to drop to zero and to stay there until the short is removed. When used with an R.V.R. amplifier, this connector should be connected to the REMOTE output of the power amplifier
[7]	FAN	Fan assisted cooling for the power stage and the power supply
[8]	TELEMETRY TERMINALS	Telemetry terminals board
[9]	P.A.. FUSE	Power amplifier protection fuse 8A
[10]	R.F. OUTPUT	N type connector, 50Ohm
[11]	R.F. TEST POINT	-40dB output referred to the output power level
[12]	FUSE	Main protection fuse 6A
[13]	RIGHT (MONO)	BNC connector for FCC unbalanced version; cannon XLR for CCIR version with balanced input
[14]	LEFT (MPX)	BNC connector for FCC version; cannon XLR for CCIR version with balanced input
[15]	SCA 1	BNC connector, unbalanced SCA1 input
[16]	SCA 2	BNC connector, unbalanced SCA2 input or output (internally selectable) for pilot tone (i.e. for R.D.S. encoder)



7. Technical Specifications

7.1 Mechanical Specifications

Panel size	483 mm (19") x 132.50 mm (5.20") (3 HE)
Depth	345 mm (13.7")
Weight	15 Kg
Temperature range	-10 °C , +50 °C

7.2 Electrical Specifications

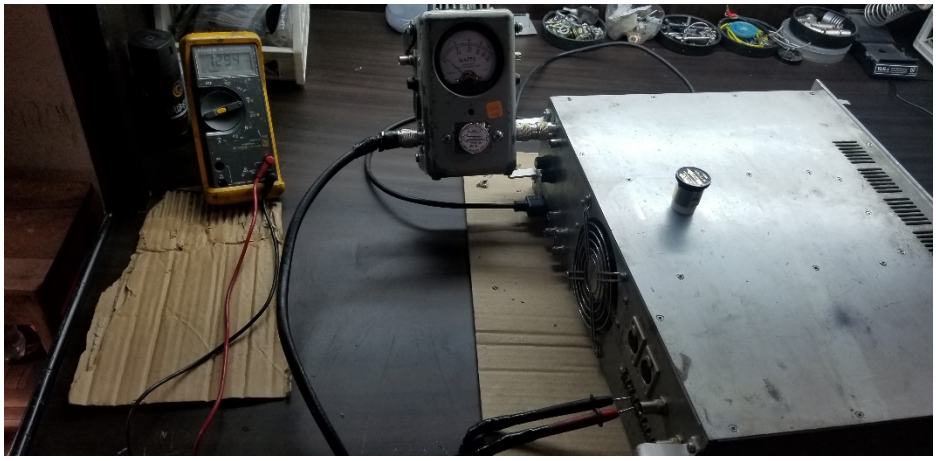
General

A.C. Supply	117-230 V ±10%, 50-60 Hz single phase
Power consuption	approx.230W
Cooling	Forced ventilation
Frequency range	from 87.5 to 108MHz in steps of 10KHz
Output power Adjustable	Adjusable from 5 to 100W
Automatic output level control	Stabilizes the set RF output level, septate with internal or external reference
S.W.R. protection	The maximum output power comes diminished in proportional way in case the reflected power increase; this control acts with internal or external reference
Output Impedance	50Ohm
Output connector	Standard "N"-type
Temperature control	Intervene in case of temperature excess of the final stage reducing the output power
Harmonic suppression	> -70dB
Spurious signal suppression	> -80dB
Intermodulation distorion	0.05% or less, measured at 1KHz and 1.3KHz, ratio 1:1 at 100% modulation
Frequency stability	± 500Hz (typically ± 300Hz) from 0° to 50° C
Modulation type	Direct frequency modulation of the RF oscillator at fundamental frequency
Frequency deviation	± 75KHz nominal
Harmonic distortion	< 0.05% (tipically 0.01%)
FM signal/noise ratio	> 75dB mono, > 70dB stereo measured with 75KHz deviation in the 30Hz to 15KHz band RMS.
Residual AM (asyncronous)	approx. 0.05% = 65dB RMS
Residual AM (syncronous)	0.1% = 60dB
Pre-emphasis	50µs ±2% or 75µs ±2% internally selectable
Audio input impedance	10KOhm balanced or 50KOhm unbalanced (600Ohm on request)
Audio input level	Selectable from -9 to +6dBm in 5 steps, continuosly from -12 to +9 dBm
Audio frequency range	30-15000Hz, MONO input 30-10000Hz, MPX input
Audio input filter	> 45dB at 19KHz (mono) > 40dB from 20KHz to 100KHz
Mono frequency response	±0.3dB from 30Hz to 15KHz
MPX frequency response	±0.5dB from 30Hz to 75KHz
Stereo Separation	> 45dB (typically 50dB)
Pilot tone frequency	19KHz ± 1Hz

Anexo 10

Toma de medidas al transmisor FM por el puerto de telemetría

Para acondicionar la señal se requiere comparar las magnitudes eléctricas con las variables respectivas, para ello se requiere de un instrumento de medición, como es el caso de un vatímetro de RF, para poder medir la potencia emitida y la reflejada.



Medición del amperaje consumido por el equipo y el dispositivo IoT

Anexo 11

Algoritmo de programación para los dispositivos de gestión de sensores

El algoritmo está diseñado para el control de 13 widgets, del panel de control, para ello se debe configurar la red wifi, la contraseña, el id del dispositivo y la contraseña del mismo, en el siguiente código:

```
#include <Arduino.h>
#include "Colors.h"
#include "IoTicosSplitter.h"
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>
#include <Stepper.h>
#include "DHTesp.h"
#include <PZEM004Tv30.h>
#include <Adafruit_ADS1X15.h>
#include <math.h>

#include "soc/timer_group_struct.h"
#include "soc/timer_group_reg.h"

// WiFi
const char *wifi_ssid = "OpenWrt_SDN";
const char *wifi_password = "2009jr1206";
long lastReconnectAttemp = 0;

String dId = "2468";
String dIdpassword = "EKzBweeJyi";
String webhook_endpoint =
"http://192.168.101.22:3001/api1/getdevicecredentia
ls";
const char *mqtt_server = "192.168.101.22";
const int mqtt_port = 1883;
const int stepsPerRevolution = 2048;
float humidity, temperature;

// Defined PINS
#define led 2
#define led_1 34

// ULN2003 Motor Driver Pins
#define IN1 23
#define IN2 19
#define IN3 18
#define IN4 5
//DHTPIN
#define DHTPIN 27

//ADC
int16_t adc0, adc1;
float volts0, volts1, pfwd, prx;

//Potenciometers pins
const int potPin2 = 35;
const int potPin = 34;

float prev_pfwd = 0;
float prev_prx = 0;
float prev_consumption = 0;
int steps = 0;
long varsLastSend[20];
```

```
long lastStats = 0;
long lastUpdated_0 = 0;
long lastUpdated_1 = 0;
long lastUpdated_2 = 0;
String last_received_msg = "";
String last_received_topic = "";
IoTicosSplitter splitter;
DHTesp dht;
TaskHandle_t Task1;
PZEM004Tv30 pzem(Serial2, 16, 17);
// FUNTIONS CALLS
void clear();
void setup_wifi();
bool get_mqtt_credentials();
void check_mqtt_connection();
bool reconnect();
void process_sensors();
void process_actuators();
void send_data_to_broker();
void callback(char *topic, byte *payload, unsigned
int length);
void process_incoming_msg(String topic, String
incoming);
void print_stats();
void feedTheDog();

// initialize the stepper library
Stepper myStepper(stepsPerRevolution, IN1, IN3,
IN2, IN4);
DynamicJsonDocument mqtt_data_doc(8192);
WiFiClient espClient;
PubSubClient client(espClient);
Adafruit_ADS1115 ads; /* Use this for the 16-bit
version */

/*****************************
****   TAREA OTRO NUCLEO   ****
*****************************/
void codeForTask1(void *parameter)
{

  for (;;)
  {
    if
(mqtt_data_doc["variables"][2]["last_incomming"]["v
alue"] != 0 )
    {
      steps =
mqtt_data_doc["variables"][2]["last_incomming"]["va
lue"];
      myStepper.step(steps);
      mqtt_data_doc["variables"][2]["last_incomming
"]["value"] = 0;

    }
    digitalWrite(IN1, LOW);
```

```
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);

    vTaskDelay(10);
  }
}

void setup()
{
  dht.setup(DHTPIN, DHTesp::DHT11);
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  myStepper.setSpeed(5);
  clear();
  setup_wifi();

  if (!ads.begin())
  {
    Serial.println("Failed to initialize ADS.");
    while (1)
      ;
  }

  xTaskCreatePinnedToCore(
      codeForTask1, /* Task function. */
      "Task_1",     /* name of task. */
      10000,        /* Stack size of task */
      NULL,         /* parameter of the task */
      1,            /* priority of the task */
      &Task1,       /* Task handle to keep track of
created task */
      0);           /* Core */
}

void loop()
{
  check_mqtt_connection();
  feedTheDog();
}
//----------Plantilla de Conexiones-------------

bool reconnect()
{
  if (!get_mqtt_credentials())
  {
    Serial.println(boldRed + "\n\n      Error
getting mqtt credentials :( \n\n RESTARTING IN 10
SECONDS");
    Serial.println(fontReset);
    delay(10000);
    ESP.restart();
  }

  // Setting up Mqtt Server
  client.setServer(mqtt_server, mqtt_port);

  Serial.print(underlinePurple + "\n\n\nTrying MQTT
Connection" + fontReset + Purple + "  ↴");

  String str_client_id = "device_" + dId + "_" +
random(1, 9999);
  const char *username = mqtt_data_doc["username"];
  const char *password = mqtt_data_doc["password"];
  String str_topic = mqtt_data_doc["topic"];

  if (client.connect(str_client_id.c_str(),
username, password))
  {
    Serial.print(boldGreen + "\n\n         Mqtt
Client Connected :) " + fontReset);
    delay(2000);

    client.subscribe((str_topic +
"+/actdata").c_str());
  }
  else
  {
    Serial.print(boldRed + "\n\n         Mqtt
Client Connection Failed :( " + fontReset);
  }
}

void check_mqtt_connection()
{
  if (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(Red + "\n\n       WiFi Connection
-> Failed :(   ");
    Serial.println(" -> Restarting..." +
fontReset);
    delay(15000);
    ESP.restart();
  }

  if (!client.connected())
  {
    long now = millis();
    if (now - lastReconnectAttemp > 5000)
    {
      lastReconnectAttemp = millis();
      if (reconnect())
      {
        lastReconnectAttemp = 0;
      }
    }
  }
  else
  {
    client.loop();
    process_sensors();
    print_stats();
    send_data_to_broker();

  }
}

void clear()
{
  Serial.write(27);    // ESC command
  Serial.print("[2J"); // clear screan command
  Serial.write(27);
  Serial.print("[H"); // cursor to home command
}

void setup_wifi()
{
  Serial.print(underlinePurple + "\nWiFi Connection
in Progress " + fontReset + Purple);
  WiFi.begin(wifi_ssid, wifi_password);
  int counter = 0;
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
    counter++;
    if (counter > 10)
    {
      Serial.print("  ↴" + fontReset);
      Serial.print(Red + "\n\n       WiFi
Connection -> Failed :(   ");
```

```
      Serial.println(" -> Restarting..." +
fontReset);
      delay(2000);
      ESP.restart();
    }
  }
  Serial.print("  ↝" + fontReset);
  Serial.println(boldGreen +
"\n\n              WiFi Connection -> SUCCESS :)");
  Serial.print(fontReset + "\n              Local
IP -> " + boldBlue);
  Serial.print(WiFi.localIP());
  Serial.print(fontReset +
"\n              Mask     -> " + boldBlue);
  Serial.print(WiFi.subnetMask());
  Serial.print(fontReset +
"\n              Gateway  -> " + boldBlue);
  Serial.print(WiFi.gatewayIP());
  Serial.print(fontReset + "\n              DNS
1    -> " + boldBlue);
  Serial.print(WiFi.dnsIP(0));
  Serial.print(fontReset + "\n              DNS
2    -> " + boldBlue);
  Serial.print(WiFi.dnsIP(1));
  Serial.println(fontReset);
  client.setCallback(callback);
}
bool get_mqtt_credentials()
{
  Serial.print(underlinePurple + "\n\n\nGetting
MQTT Credentials from WebHook" + fontReset + Purple
+ "  ↝");
  delay(1000);

  String toSend = "dId=" + dId + "&dIdpassword=" +
dIdpassword;

  HTTPClient http;
  http.begin(webhook_endpoint);
  http.addHeader("Content-Type", "application/x-
www-form-urlencoded"); // Siempre por defecto

  int response_code = http.POST(toSend);

  if (response_code < 0)
  {
    Serial.print(boldRed + "\n\n         Error
Sending Post Request :( " + fontReset);
    http.end();
    return false;
  }

  if (response_code != 200)
  {
    Serial.print(boldRed + "\n\n         Error in
response :(   e-> " + fontReset + " " +
response_code);
    http.end();
    return false;
  }

  if (response_code == 200)
  {
    String responseBody = http.getString();

    Serial.print(boldGreen + "\n\n         Mqtt
Credentials Obtained Successfully :) " +
fontReset);
    deserializeJson(mqtt_data_doc, responseBody);
    for (int i = 0; i <
mqtt_data_doc["variables"].size(); i++)
    {
      if
(mqtt_data_doc["variables"][i]["variableType"] ==
"bidi")
      {
        mqtt_data_doc["variables"][i]["values_incom
ing"][0]["value"] = 0;
        mqtt_data_doc["variables"][i]["values_outgo
ing"]["current_value"] = 0;
      }
    }
    http.end();
    delay(1000);
  }
  return true;
}

//------------------------------
void process_sensors()
{
  // get LED status
  mqtt_data_doc["variables"][8]["last"]["value"] =
(HIGH == digitalRead(led));
  mqtt_data_doc["variables"][1]["last"]["value"] =
pzem.pf() * 100;
  mqtt_data_doc["variables"][9]["last"]["value"] =
pzem.voltage();
  mqtt_data_doc["variables"][10]["last"]["value"] =
pzem.power();
  mqtt_data_doc["variables"][11]["last"]["value"] =
pzem.current();
  mqtt_data_doc["variables"][12]["last"]["value"] =
pzem.frequency();
  mqtt_data_doc["variables"][13]["last"]["value"] =
pzem.energy();

  // save temp Conditions
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);

  volts0 = ads.computeVolts(adc0);
  volts1 = ads.computeVolts(adc1);


  long now = millis();
    if (now - lastUpdated_0 > 200){
      lastUpdated_2 = millis();
      float dif1 = pzem.energy() -
prev_consumption;
      if (dif1 < 0)
      {
        dif1 *= -1;
      }

      if (dif1 >= 0.5)
      {
        mqtt_data_doc["variables"][13]["last"]["sav
e"] = 1;
      }
      else
      {
        mqtt_data_doc["variables"][13]["last"]["sav
e"] = 0;
      }
      prev_consumption = pzem.energy();
    }
    if (now - lastUpdated_0 > 500)
    {
      lastUpdated_0 = millis();
```

```
      pfwd = 14.81 * (pow(volts0, 2)) + (5.9382 *
volts0) - 0.3351;
      float pfwdR = roundf(pfwd * 100) / 100;
      mqtt_data_doc["variables"][2]["last"]["value"
] = pfwdR;
      mqtt_data_doc["variables"][4]["last"]["value"
] = pfwdR;
      int dif = pfwdR - prev_pfwd;
      if (dif < 0)
      {
        dif *= -1;
      }

      if (dif >= 3)
      {
        mqtt_data_doc["variables"][4]["last"]["save
"] = 1;
      }
      else
      {
        mqtt_data_doc["variables"][4]["last"]["save
"] = 0;
      }
      prev_pfwd = pfwdR;
    }

  if (now - lastUpdated_1 > 250){
    lastUpdated_1 = millis();
    prx = 3.5141 * (pow(volts1, 2)) - (0.84 *
volts1) + 1.42;
    float prxR = roundf(prx * 100) / 100;

    mqtt_data_doc["variables"][3]["last"]["value"]
= prxR;
    mqtt_data_doc["variables"][5]["last"]["value"]
= prxR;
    int dif = prxR - prev_prx;
    if (dif < 0)
    {
      dif *= -1;
    }

    if (dif >= 1)
    {
      mqtt_data_doc["variables"][5]["last"]["save"]
= 1;
    }
    else
    {
      mqtt_data_doc["variables"][5]["last"]["save"]
= 0;
    }
    prev_prx = prxR;
  }


}
void process_actuators()
{
  if
(mqtt_data_doc["variables"][7]["last"]["value"] ==
"true")
  {
    digitalWrite(led, HIGH);
    mqtt_data_doc["variables"][7]["last"]["value"]
= "";
    varsLastSend[4] = 0;
  }
  else if
(mqtt_data_doc["variables"][6]["last"]["value"] ==
"false")
  {
    digitalWrite(led, LOW);
    mqtt_data_doc["variables"][6]["last"]["value"]
= "";
    varsLastSend[4] = 0;
  }

}

void send_data_to_broker()
{

  long now = millis();

  for (int i = 0; i <
mqtt_data_doc["variables"].size(); i++)
  {

    if
(mqtt_data_doc["variables"][i]["variableType"] ==
"output")
    {
      continue;
    }
    if
(mqtt_data_doc["variables"][i]["variableType"] ==
"esp_cam")
    {
      continue;
    }

    int freq =
mqtt_data_doc["variables"][i]["variableSendFreq"];

    if (now - varsLastSend[i] > freq * 1000)
    {

      varsLastSend[i] = millis();

      String str_root_topic =
mqtt_data_doc["topic"];
      String str_variable =
mqtt_data_doc["variables"][i]["variable"];
      String topic = str_root_topic + str_variable
+ "/sdata";

      String toSend = "";

      serializeJson(mqtt_data_doc["variables"][i]["
last"], toSend);

      client.publish(topic.c_str(),
toSend.c_str());

      // STATS
      long counter =
mqtt_data_doc["variables"][i]["counter"];
      counter++;
      mqtt_data_doc["variables"][i]["counter"] =
counter;
    }
  }
}

void callback(char *topic, byte *payload, unsigned
int length)
{

  String incoming = "";
  //Serial.println(Purple + topic + fontReset);
  for (int i = 0; i < length; i++)
```

```
  {
    incoming += (char)payload[i];
  }

  incoming.trim();

  process_incoming_msg(String(topic), incoming);
}

void process_incoming_msg(String topic, String
incoming)
{
  last_received_topic = topic;
  last_received_msg = incoming;

  String variable = splitter.split(topic, '/', 2);

  for (int i = 0; i <
mqtt_data_doc["variables"].size(); i++)
  {

    if (mqtt_data_doc["variables"][i]["variable"]
== variable)
    {

      DynamicJsonDocument doc(512);
      deserializeJson(doc, incoming);
      if
(mqtt_data_doc["variables"][i]["variableType"] ==
"bidi")
      {
        if (doc.containsKey("value"))
        {
          mqtt_data_doc["variables"][i]["last_incom
ming"] = doc;

        }
      }
      else if
(mqtt_data_doc["variables"][i]["variableType"] ==
"output")
      {
        // Serial.println(backgroundCyan + "OUTPUT"
+ fontReset);
        if (doc.containsKey("value"))
        {
          //
mqtt_data_doc["variables"][3]["last"]["value"] ==
"true"
          mqtt_data_doc["variables"][i]["last"] =
doc;
        }
      }

        // STATS
        long counter =
mqtt_data_doc["variables"][i]["counter"];
        counter++;
        mqtt_data_doc["variables"][i]["counter"] =
counter;
    }
  }

  process_actuators();
}
void print_stats()
{
  long now = millis();

  if (now - lastStats > 2000)
  {
    lastStats = millis();
    clear();

    Serial.print("\n");
    Serial.print(Purple +
"\n╔═══════════════════════╗" + fontReset);
    Serial.print(Purple + "\n║       SYSTEM
STATS       ║" + fontReset);
    Serial.print(Purple +
"\n╚═══════════════════════╝" + fontReset);
    Serial.print("\n\n");
    Serial.print("\n\n");

    Serial.print(boldCyan + "#" + " \t Name" + "
\t\t Var" + " \t\t Type" + " \t\t Count" + " \t\t
Last V" + fontReset + "\n\n");

    for (int i = 0; i <
mqtt_data_doc["variables"].size(); i++)
    {

      String variableFullName =
mqtt_data_doc["variables"][i]["variableFullName"];
      String variable =
mqtt_data_doc["variables"][i]["variable"];
      String variableType =
mqtt_data_doc["variables"][i]["variableType"];
      String lastMsg =
mqtt_data_doc["variables"][i]["last"];
      long counter =
mqtt_data_doc["variables"][i]["counter"];

      Serial.println(String(i) + " \t " +
variableFullName.substring(0, 5) + " \t\t " +
variable.substring(0, 10) + " \t " +
variableType.substring(0, 5) + " \t\t " +
String(counter).substring(0, 10) + " \t\t " +
lastMsg);
    }

    Serial.print(boldGreen + "\n\n Free RAM -> " +
fontReset + ESP.getFreeHeap() + " Bytes");

    Serial.print(boldGreen + "\n\n Last Incomming
Topic -> " + fontReset + last_received_topic);
    Serial.print(boldGreen + "\n\n Last Incomming
Msg -> " + fontReset + last_received_msg);
  }
}

void feedTheDog()
{
  // feed dog 0
  TIMERG0.wdt_wprotect = TIMG_WDT_WKEY_VALUE; //
write enable
  TIMERG0.wdt_feed = 1;                       //
feed dog
  TIMERG0.wdt_wprotect = 0;                   //
write protect
  // feed dog 1
  TIMERG1.wdt_wprotect = TIMG_WDT_WKEY_VALUE; //
write enable
  TIMERG1.wdt_feed = 1;                       //
feed dog
  TIMERG1.wdt_wprotect = 0;                   //
write protect
}
```

Anexo 12

Algoritmo de programación para el procesamiento de imágenes

El algoritmo está diseñado para el control del widgets cámara, para lo cual se requiere configurar la red wifi, la contraseña, el id del dispositivo y la contraseña de este, en el siguiente código:

```
#include <WiFi.h>
#include <HTTPClient.h>
#include "IoTicosSplitter.h"
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include "esp_camera.h"
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"
#define LED_BUILTIN 4
const char *wifi_ssid = "OpenWrt_SDN";
const char *wifi_password = "2009jr1206";

// WEBHOOK
String dId = "2468";
String dIdpassword = "EKzBweeJyi";
String webhook_endpoint =
"http://192.168.101.22:3001/api1/getdevicecredentia
ls";

// MQTT variables
const char *mqtt_server = "192.168.101.22";

const int MAX_PAYLOAD = 60000;
long lastReconnectAttemp = 0;
long varsLastSend[20];

String last_received_msg = "";
String last_received_topic = "";
IoTicosSplitter splitter;
WiFiClient espClient;
PubSubClient client(espClient);
DynamicJsonDocument mqtt_data_doc(8192);

// FUNTIONS CALLS
void setup_wifi();
void cameraInit();
```

```
void callback(String topic, byte *payload, unsigned
int length);
void process_incoming_msg(String topic, String
incoming);
void take_picture();
void process_actuators();
void sendMQTT(const uint8_t *buf, uint32_t len);
bool reconnect();
bool get_mqtt_credentials();
void check_mqtt_connection();

void setup()
{

  // Define Flash as an output
  pinMode(LED_BUILTIN, OUTPUT);

  // Initialise the Serial Communication
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
  cameraInit();
  setup_wifi();

  Serial.print("Camera adress:");
  Serial.print(WiFi.localIP());

  // Set MQTT Connection Parameters
  client.setServer(mqtt_server, 1883);
  client.setBufferSize(MAX_PAYLOAD); // This is the
maximum payload length
  client.setCallback(callback);
}

void loop()
{
  check_mqtt_connection();
```

```
}
void cameraInit()
{
  // Config Camera Settings
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  // init with high specs to pre-allocate larger
buffers
  if (psramFound())
  {
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  }
  else
  {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }


  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
  {
    Serial.printf("Camera init failed with error
0x%x", err);
    return;
  }

  sensor_t *s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and
colors are a bit saturated
  if (s->id.PID == OV3660_PID)
  {
    s->set_vflip(s, 1);       // flip it back
    s->set_brightness(s, 1);  // up the blightness
just a bit
    s->set_saturation(s, -2); // lower the
saturation
  }
  s->set_framesize(s, FRAMESIZE_SVGA);


}
void setup_wifi()
{
  Serial.print("\nWiFi Connection in Progress ");
  WiFi.begin(wifi_ssid, wifi_password);
  int counter = 0;
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
    counter++;
    if (counter > 10)
    {
      Serial.print("  ↴");
      Serial.print("\n\n       WiFi Connection ->
Failed :(   ");
      Serial.println(" -> Restarting...");
      delay(2000);
      ESP.restart();
    }
  }
  Serial.print("  ↴");
  Serial.println("\n\n             WiFi Connection
-> SUCCESS :)");
  Serial.print("\n             Local IP -> ");
  Serial.print(WiFi.localIP());
  Serial.print(+"\n              Mask     -> ");
  Serial.print(WiFi.subnetMask());
  Serial.print(+"\n              Gateway  -> ");
  Serial.print(WiFi.gatewayIP());
  Serial.print(+"\n              DNS 1    -> ");
  Serial.print(WiFi.dnsIP(0));
```

```
  Serial.print(+"\n              DNS 2    -> ");
  Serial.print(WiFi.dnsIP(1));
  Serial.println();
}
bool reconnect()
{
  while(!client.connected()){
    if (!get_mqtt_credentials())
    {
      Serial.println("\n\n      Error getting mqtt credentials :( \n\n RESTARTING IN 5 SECONDS");
      delay(5000);
      ESP.restart();
    }

    Serial.print("\n\n\nAttempting MQTT connection...                ↴");

    String str_client_id = "ESPCAM_device_" + dId + "_" + random(1, 9999);
    const char *username = mqtt_data_doc["username"];
    const char *password = mqtt_data_doc["password"];
    String str_topic = mqtt_data_doc["topic"];


    if (client.connect(str_client_id.c_str(), username, password))
    {
      Serial.print("\n\n       MQTT Client connected (Y)");
      client.subscribe((str_topic + "+/actdata").c_str());
      return true;
    }
    else
    {
      Serial.print("\n\n       MQTT Client Connection Failed :c ");
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 3 seconds");
      delay(3000);
      return false;
    }
  }
```

```
}
bool get_mqtt_credentials(){
  Serial.print("\n\n\nGetting MQTT Credentials from WebHook        ↴");
  delay(500);
  String toSend = "dId=" + dId + "&dIdpassword=" + dIdpassword;
  HTTPClient http;
  http.begin(webhook_endpoint);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded"); // Siempre por defecto
  int response_code = http.POST(toSend);
  if(response_code < 0){
    Serial.print("\n\n         Error Sending Post Request :( ");
    http.end();
    return false;
  }
  if (response_code != 200){
    Serial.print("\n\n         Error in response :(   e->   " + response_code);
    http.end();
    return false;
  }
  if (response_code == 200){
    String responseBody = http.getString();
    Serial.print("\n\n         Mqtt Credentials Obtained Successfully :) ");
    deserializeJson(mqtt_data_doc, responseBody);
    for (int i = 0; i < mqtt_data_doc["variables"].size(); i++)
    {
      if (mqtt_data_doc["variables"][i]["variableType"] == "esp_cam")
      {
        mqtt_data_doc["variables"][i]["values_incoming"][0]["value"] = false;
        mqtt_data_doc["variables"][i]["values_incoming"][0]["flash_value"] = false;
        mqtt_data_doc["variables"][i]["values_outgoing"]["camera_satus"] = false;
        mqtt_data_doc["variables"][i]["values_outgoing"]["flash_status"] = false;
      }
    }
    http.end();
```

```
    delay(1000);
  }
  return true;
}
void check_mqtt_connection(){
  if (WiFi.status() != WL_CONNECTED)
  {
    Serial.print("\n\n       WiFi Connection ->
Failed :(    ");
    Serial.println(" -> Restarting...");
    delay(15000);
    ESP.restart();
  }
  if(!client.connected()){
    long now = millis();
    if(now - lastReconnectAttemp > 5000){
      lastReconnectAttemp = millis();
      if(reconnect()){
        lastReconnectAttemp = 0;
      }
    }
  }else{
    client.loop();
    take_picture();
  }
}
void take_picture()
{
  camera_fb_t *fb = NULL;
  fb = esp_camera_fb_get();
  if (!fb)
  {
    Serial.println("Camera capture failed");
    delay(2000);
    ESP.restart();
  }
  sendMQTT(fb->buf, fb->len);
  esp_camera_fb_return(fb);
}
void sendMQTT(const uint8_t *buf, uint32_t len)
{
  long now = millis();
  String str_root_topic = mqtt_data_doc["topic"];
  String str_variable =
mqtt_data_doc["variables"][0]["variable"];
  String CameraTopic = str_root_topic +
str_variable + "/espcam";
  String FlashTopic = str_root_topic + str_variable
+ "/sdata";
  if (len > MAX_PAYLOAD)
  {
    Serial.print("Picture too large, increase the
MAX_PAYLOAD value");
  }
  if
(mqtt_data_doc["variables"][0]["values_incoming"][0
]["value"] == true)
  {
    mqtt_data_doc["variables"][0]["values_outgoing"
]["camera_satus"] = true;
    client.publish(CameraTopic.c_str(), buf, len,
false);
  }
  if
(mqtt_data_doc["variables"][0]["values_incoming"][0
]["value"] == false)
  {
    mqtt_data_doc["variables"][0]["values_outgoing"
]["camera_satus"] = false;
  }
  int freq =
mqtt_data_doc["variables"][0]["variableSendFreq"];
  if(now - varsLastSend[0] > freq * 1000){
    varsLastSend[0] = millis();
    String toSend = "";
    serializeJson(mqtt_data_doc["variables"][0]["va
lues_outgoing"], toSend);
    client.publish(FlashTopic.c_str(),
toSend.c_str());
  }
}
void process_actuators()
{
  if
(mqtt_data_doc["variables"][0]["values_incoming"][0
]["flash_value"] == true)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    mqtt_data_doc["variables"][0]["values_outgoing"
]["flash_status"] = true;
    varsLastSend[0] = 0;
  }
```

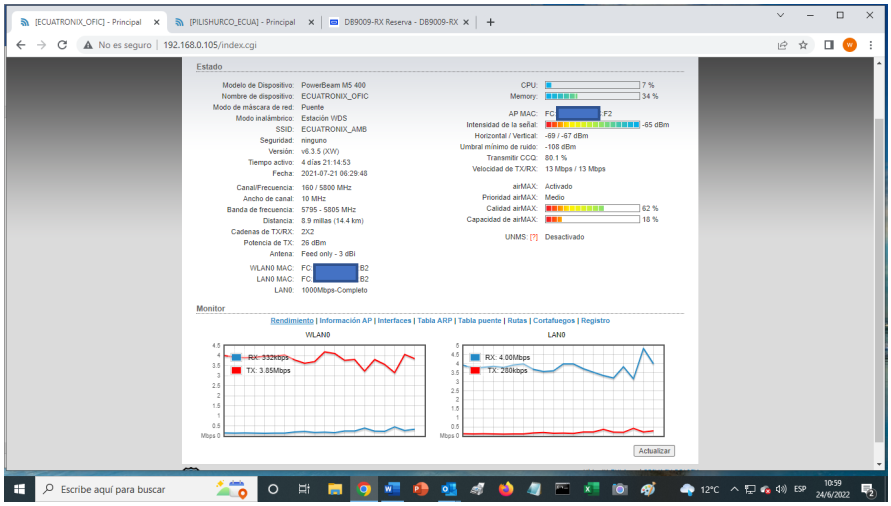
```
  if
(mqtt_data_doc["variables"][0]["values_incoming"][0
]["flash_value"] == false)
  {
    digitalWrite(LED_BUILTIN, LOW);
    mqtt_data_doc["variables"][0]["values_outgoing"
]["flash_status"] = false;
    varsLastSend[0] = 0;
  }
}
void callback(String topic, byte *payload, unsigned
int length)
{
  String incoming;
  for (int i = 0; i < length; i++)
  {
    incoming += (char)payload[i];
  }
  incoming.trim();
  process_incoming_msg(String(topic), incoming);
}
void process_incoming_msg(String topic, String
incoming){
  last_received_topic = topic;
  last_received_msg = incoming;
  String variable = splitter.split(topic, '/', 2);
  for (int  i = 0; i <
mqtt_data_doc["variables"].size(); i++)
  {
    if (mqtt_data_doc["variables"][0]["variable"]
== variable){
      DynamicJsonDocument doc(1024);
      deserializeJson(doc, incoming);
      if
(mqtt_data_doc["variables"][0]["variableType"] ==
"esp_cam")
      {
        if (doc.containsKey("value"))
        {
          mqtt_data_doc["variables"][0]["values_inc
oming"][0]["value"] = doc["value"];
        }
        if (doc.containsKey("flash_value"))
        {
          mqtt_data_doc["variables"][0]["values_inc
oming"][0]["flash_value"] = doc["flash_value"];
        }
      }else{
        continue;
      }
    }
  }
  process_actuators();
}
```

Anexo 13

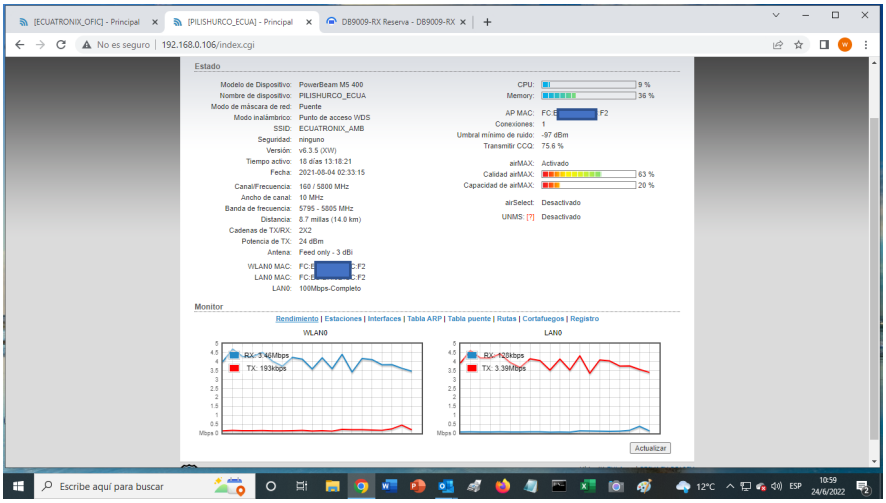
Enlace de internet punto a punto

Actualmente la empresa tiene una infraestructura de red punto a punto desde los laboratorios de Ecuatronix Ambato hasta el cerro Pilishurco por medio de los equipos Ubiquiti modelo Power Beam M5 400, los mismo que tienen un alcance de 25 km y trabajan en la banda licenciada de los 5GHz, y dado que la distancia de estos dos puntos es de 14.4Km es idóneo para este tipo de aplicaciones, en las siguientes imágenes se muestran el estado de los enlaces entre estos dos puntos.

Estado del enlace en el laboratorio de Ecuatronix ubicado en la ciudadela Alborada



Estado del enlace en el cerro Pilishurco



Anexo 14

Pruebas de latencia en distintas escenarios

Se realizo distintas pruebas de respuesta de latencia en la red por medio de la herramienta ping de Windows, en donde se contemplan distintos escenarios:

<table>
<tr><td>Ping desde el router OpenWrt situado en el cerro Pilishurco</td><td>Ping desde el router OpenWrt situado en los laboratorios</td></tr>
<tr><td>

```
Command Prompt

C:\Users\jronq>ping 192.168.101.22

Pinging 192.168.101.22 with 32 bytes of data:
Reply from 192.168.101.22: bytes=32 time=14ms TTL=63
Reply from 192.168.101.22: bytes=32 time=14ms TTL=63
Reply from 192.168.101.22: bytes=32 time=59ms TTL=63
Reply from 192.168.101.22: bytes=32 time=16ms TTL=63

Ping statistics for 192.168.101.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 59ms, Average = 25ms

C:\Users\jronq>
```

</td><td>

```
Command Prompt

C:\Users\jronq>ping 192.168.101.22

Pinging 192.168.101.22 with 32 bytes of data:
Reply from 192.168.101.22: bytes=32 time=17ms TTL=64
Reply from 192.168.101.22: bytes=32 time=19ms TTL=64
Reply from 192.168.101.22: bytes=32 time=15ms TTL=64
Reply from 192.168.101.22: bytes=32 time=15ms TTL=64

Ping statistics for 192.168.101.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 15ms, Maximum = 19ms, Average = 16ms
```

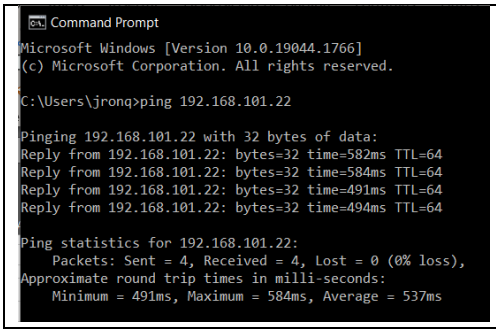
</td></tr>
<tr><td>Ping desde la red WiFi de los laboratorios de Ecuatronix</td><td>Ping desde la red móvil de CNT en el sector de la Cdla. España Ambato-Ecuador</td></tr>
<tr><td>

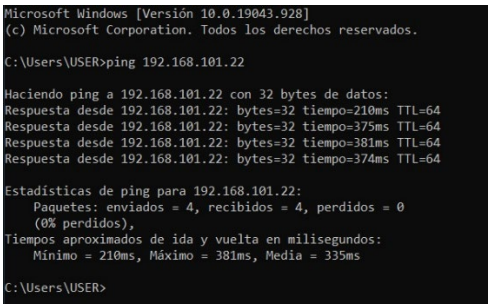
```
Command Prompt

C:\Users\jronq>ping 192.168.101.22

Pinging 192.168.101.22 with 32 bytes of data:
Reply from 192.168.101.22: bytes=32 time=17ms TTL=64
Reply from 192.168.101.22: bytes=32 time=16ms TTL=64
Reply from 192.168.101.22: bytes=32 time=45ms TTL=64
Reply from 192.168.101.22: bytes=32 time=23ms TTL=64

Ping statistics for 192.168.101.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 16ms, Maximum = 45ms, Average = 25ms
```

</td><td>

```
Command Prompt

C:\Users\jronq>ping 192.168.101.22

Pinging 192.168.101.22 with 32 bytes of data:
Reply from 192.168.101.22: bytes=32 time=346ms TTL=64
Reply from 192.168.101.22: bytes=32 time=309ms TTL=64
Reply from 192.168.101.22: bytes=32 time=336ms TTL=64
Reply from 192.168.101.22: bytes=32 time=296ms TTL=64

Ping statistics for 192.168.101.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 296ms, Maximum = 346ms, Average = 321ms
```

</td></tr>
<tr><td>Ping desde la red móvil de Movistar en el sector de la Cdla. España Ambato-Ecuador</td><td>Ping desde la red móvil de Claro en el sector de la Cdla. España Ambato-Ecuador</td></tr>
</table>





:

Anexo 15

Instalación del dispositivo en el cerro Pilishurco

La instalación del proyecto se lo realizo en la caseta de Ecuatronix , como se muestra en las siguientes imágenes:


ECUATRONIX