UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

"SISTEMA DE CONTROL DE TRÁFICO VEHICULAR APLICANDO LA ARQUITECTURA FOG COMPUTING"

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

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

AUTOR: Edisson Miguel Ibarra Nuñez

TUTOR: Ing. Marco Antonio Jurado Lozada Mg.

Ambato - Ecuador

Marzo 2021

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: "SISTEMA DE CONTROL DE TRÁFICO VEHICULAR APLICANDO LA ARQUITECTURA FOG COMPUTING", desarrollado bajo la modalidad Proyecto de Investigación por el señor Edisson Miguel Ibarra Nuñez, 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, marzo 2021.

Firmado electrónicamente por:
MARCO ANTONIO JURADO LOZADA

Ing. Marco Antonio Jurado Lozada Mg.
TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: "SISTEMA DE CONTROL DE TRÁFICO VEHICULAR APLICANDO LA ARQUITECTURA FOG COMPUTING" es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2021.

Edisson Miguel Ibarra Nuñez

C.C. 1804719373

AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Edisson Miguel Ibarra Nuñez, 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 CONTROL DE TRÁFICO VEHICULAR APLICANDO LA ARQUITECTURA FOG COMPUTING", 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 la señora Presidenta del Tribunal.

Ambato, marzo 2021.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
VICTOR SANTIAGO MANZANO VILLAFUERTE

Ing. Santiago Manzano, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
ANA PAMELA CASTRO

Ing. Pamela Castro, Mg.
PROFESORA CALIFICADORA

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, marzo 2021.

Edisson Miguel Ibarra Nuñez

C.C. 1804719373

AUTOR

DEDICATORIA

Dedido el presente proyecto de titulación a mi padre, pilar fundamental en mi vida y mi ejemplo ha seguir, quien me ha enseñado a esforzarme para alcanzar los objetivos, ser una mejor persona con valores y principios, y además por esa gran motivación y apoyo incondicional para culminar la carrera.

Edisson Miguel Ibarra Nuñez

AGRADECIMIENTO

Agradezo a Dios por darme fortaleza y sabiduría en los momentos difíciles.

A mi madre por cuidarme, bendecirme y estar pediente de mí en todo momento, a mis hermanos por darme su confianza y ánimos para seguir adelante.

A Mayra Noboa por su cariño y apoyo incondicional a lo largo de la carrera.

A Ricardo Fiallos por el soporte brindado al proyecto de investigación.

A los docentes de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial por todos los conocimientos impartidos, de manera especial para el Ing. Marco Jurado por guiarme para la culminación de este proyecto.

Edisson Miguel Ibarra Nuñez

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La arquitectura de comunicaciones fog computing permite implementar aplicaciones que requieren del procesamiento de datos en tiempo real como es el caso de la gestión de tránsito vehicular mediante semáforos inteligentes, por esta razón el presente proyecto se centra en el diseño de un sistema de control de tráfico vehicular para la Avenida Cevallos y Calle Eugenio Espejo de la ciudad de Ambato, el cual permite procesar los datos en tiempo real del volumen de tráfico y actuar en los intervalos de tiempo de semaforización de manera automática.

El diseño del sistema se basa en la instalación de un nodo fog o servidor en la niebla para la comunicación, almacenamiento y procesamiento de datos a través de tecnologías de comunicación inalámbrica Wifi, redes de telefonía móvil celular 4G LTE, protocolos de comunicación Mqtt y WebSocket para los dispositivos IoT y un algoritmo de programación el cual permite realizar el conteo de los vehículos utilizando geovallas o perímetros virtuales para determinar la fase de semaforización adecuada, visualizados mediante una interfaz web de monitoreo en tiempo real. El sistema además utiliza una aplicación móvil diseñada para enviar la ubicación GPS del vehículo, un servidor en la nube de AWS para obtener los registros históricos del sistema y la construcción de un prototipo de semáforo inteligente el cual permite verificar el funcionamiento de las fases de semaforización.

Por último se compara los parámetros técnicos que tiene el fog con respecto al cloud computing, obteniendo ventajas en el fog computing ya que presenta baja latencia y mayor capacidad de procesamiento.

Palabras claves:

Niebla, Nube, Semáforos, IoT, Gps, Tráfico, Vehicular.

ABSTRACT

The fog computing communications architecture allows the implementation of applications that require real time data processing, such as traffic management through intelligent traffic lights, for this reason this project focuses on the design of a traffic control system vehicle for Cevallos Avenue and Eugenio Espejo Street in the city of Ambato, which allows to process the data in real time of the traffic volume and act automatically in the traffic light time intervals.

The system design is based on the installation of a fog node or server in the fog for communication, storage and data processing through Wifi wireless communication technologies, 4G LTE cellular mobile phone networks, Mqtt and WebSocket communication protocols for IoT devices and a programming algorithm which allows to count vehicles using geofences or virtual perimeters to determine the appropriate traffic light phase, viewed through a real time monitoring web interface. The system also uses a mobile application designed to send the vehicle's GPS location, a server in the AWS cloud to obtain the historical records of the system and the construction of an intelligent traffic light prototype which allows to verify the operation of the traffic light phases.

Finally, the technical parameters that fog has with respect to cloud computing are compared, obtaining advantages in fog computing since it has low latency and greater processing capacity.

Keywords:

Fog, Cloud, Traffic lights, IoT, Gps, Traffic, Vehicular.

CAPÍTULO I

MARCO TEÓRICO

1.1 Tema de Investigación

Sistema de control de tráfico vehicular aplicando la arquitectura fog computing.

1.2 Antecedentes Investigativos

En los últimos años se han desarrollado varios trabajos de investigación acerca de la arquitectura fog computing el cual distribuye funciones de computación, almacenamiento y control más cerca a los usuarios mediante la aplicación de una capa de niebla el cual recibe, procesa los datos generados por los dispositivos IoT y los envía a la nube si es necesario.

La arquitectura fog computing es un elemento clave para la implementación de soluciones a aplicaciones en tiempo real, como es el caso del control de tráfico vehicular y a la vez un campo de gran interés para la comunidad científica.

En el proyecto de Investigación "Sistema de semaforización inteligente para el control de flujo vehicular mediante el Procesamiento Digital de Imágenes" realizada en la Universidad Técnica de Ambato por Carla Chávez en el año 2015 en el cual se realizó un prototipo para el control de los tiempos de las fases de los semáforos según el número de vehículos que se encuentren en la intersección formada por la Av. Cevallos y la calle J.L. Mera.

La implementación del proyecto de investigación se realiza mendiante el uso de cámaras y programación desarrollada en Matlab, se identifica el número de vehiculos para finalmente asignar los tiempos de fase de los semáforos utilizando una placa arduino.

Se concluyó que el prototipo se ve afectado directamente por la luminosidad existente ya que varía por el clima y en el transcurso de las horas, además se indica que el porcentaje más alto de tráfico no responde a las horas pico ya que varía dependiendo del día y lugar [1].

En el trabajo investigado titulado "OpenFog Reference Architecture for Fog Computing" del Consorcio OpenFog en el cual participan marcas comerciales como ARM, Cisco, Dell, Intel y Microsoft desarrollado en Estados Unidos en el año 2017, se presenta una arquitectura abierta de niebla genérica diseñada para acelerar implementaciones en aplicaciones inteligentes enfocadas en transporte, energía, atención médica y fabricación de objetos materiales.

El trabajo investigativo describe las características que tiene la cadena de suministros de arquitectura de niebla que corresponde a fabricantes de componentes, vendedores de sistemas, proveedores de software y desarrolladores de aplicaciones.

Como resultado define el uso del estándar internacional ISO / IEC / IEEE 42010: 2011 y el uso de bancos de pruebas de OpenFog, con el propósito de garantizar que esta arquitectura de referencia dé como resultado sistemas interoperables y seguros, respaldados por un proveedor de ecosistema abierto y dinámico [2].

En el año 2019 el artículo científico denominado "IoT enabled Smart Fog Computing for Vehicular Traffic Control" publicado por Alianza Europea para la Innovación (EAI), en la que se implementa dos métodos experimentales para Smart Cities.

El primer método experimental implica el uso estándar de la nube y el segundo emplea la computación de niebla utilizando nodos de sensor IoT en las áreas y rutas de la universidad.

Con la finalidad de comparar el rendimiento se obtiene como resultado los siguientes datos: el tiempo de procesamiento de extremo a extremo se redujo de 29.44 a 6.7 segundos, casi un 77% menos, el número de saltos atravesados se redujo de 56 a 4 saltos, casi un 92% menos, y el uso del ancho de banda se redujo de 247 a 8 kbps casi 96.7% menos.

Se concluyó que a partir de las configuraciones experimentales Fog e IoT procesan los datos de tráfico localmente en los dispositivos de borde, lo que reduce el tiempo de extremo a extremo [3].

En la Universidad Técnica de Ambato se desarrolló en el año 2019 el trabajo de investigación para maestría titulado "Análisis de tráfico vehicular mediante visión artificial" por Jovann Pérez el cual propone el desarrollo de un programa orientado al

análisis de tráfico para el diseño y/o construcción de obras de vialidad derivada de la necesidad de crecimiento sostenible de la urbe.

La investigación aplica principalmente visión artificial para el seguimiento de vehículos y el conteo de los mismos, además utiliza una red neuronal de flujo directo el cual tiene un rendimiento significativamente superior a los sistemas basados en técnicas de sustracción de fondo para detectar objetos en movimiento debido a su mayor porcentaje de asertividad y tolerancia a perturbaciones.

Como conclusión el programa de análisis de tráfico permite solventar los problemas asociados a los de tiempo real y generar reportes del flujo vehicular, estimaciones de distancia recorridas y velocidades promedio registradas [4].

En la tesis doctoral titulado "Fog Computing based Traffic Safety for Connected Vulnerable Road Users" elaborado por Esubalew Jalew en la Universidad de Bourgogne en el año 2019, realizada con el fin de salvaguardar la vida de usuarios vulnerables en la vía pública al predecir accidentes de tránsito mediante el uso de dispositivos móviles con la computación en la niebla.

El funcionamiento del sistema se basa en extraer la posición del usuario mediante el dispositivo móvil y envíar a un servidor de niebla, el servidor estima la colisión usando un algoritmo de predicción y envía un mensaje de alerta si se predice una colisión a punto de ocurrir.

En la tesis además se describe cada etapa del proyecto, para la precisión del sistema de posicionamiento global (GPS) se utiliza un algoritmo de correspondencia de mapas, el filtro de Kalman para suavizar las lecturas atípicas y se soluciona los problemas de privacidad utilizando un sistema de gestión de confianza que permite a los clientes evaluar el nivel de confianza de los servidores de niebla antes de otorgar el servicio.

Se concluye que el uso de dispositivos móviles combinados con el concepto fog computing tiene un gran potencial para mejorar la seguridad vial y específicamente para reducir el número excesivo de accidentes de tráfico ya que generan alertas en tiempo real y supera a otras arquitecturas de seguridad vial relacionadas en términos de confiabilidad, escalabilidad y latencia [5].

1.2.1 Contextualización del Problema

Durante la última década, la tendencia ha sido utilizar la computación en la nube ya que permite obtener servicios de informática, control y almacenamiento de datos a través de una red de internet, actualmente con el emergente Internet de las cosas (IoT) la nube se enfrenta a grandes desafíos para cumplir con los nuevos requerimientos necesarios para su aplicación [6]. En el estudio publicado por Cisco el Internet de las cosas actualmente conecta alrededor de 8.9 mil millones de dispositivos en todo el mundo con un crecimiento de 2,4 veces es decir 10.6 mil millones para el año 2021, Cisco pronostica que el tráfico del centro de datos en la nube a nivel mundial será 3.3 veces mayor para el 2021 el cual alcanzará 19.5 zettabytes que representa el 95 por ciento de la capacidad total de la nube, con respecto a 14,1 zettabytes actuales que representa el 92 por ciento de la capacidad total de la misma [7]. Además Cisco realizó una encuesta orientada a líderes ejecutivos de tecnología de la información y tecnología operativa para obtener datos sobre cómo las organizaciones están aprovechando el IoT, como resultado el 40 por ciento de los encuestados afirma que en los próximos años la mayoría de los datos generados por sus soluciones de IoT se procesarán, con un crecimiento anual del 3.4 por ciento del total de dispositivos IoT a nivel mundial es decir del 33 por ciento en 2018 al 50 por ciento para el año 2023, siendo las aplicaciones domésticas y el automóvil conectado los servicios de más rápido crecimiento [8]. El tráfico en la nube está creciendo rápidamente y su capacidad es insuficiente para adoptar servicios que generen nuevas demandas como en el caso del IoT que requiere grandes cantidades de ancho de banda para el almacenamiento, análisis o procesamiento de los datos.

Además del tráfico de datos, la computación en la nube se ve también afectada por la latencia, ya que el procesamiento centralizado de los datos implica al número de hops (saltos) que un paquete de datos recorre desde la fuente al destino el cual son numerosos servidores ubicados en todo el mundo generalmente en América del Norte, Europa y Asia, es decir que si el número de saltos o la distancia aumenta, mayor será el desfase entre el paquete de datos que genera el dispositivo IoT y el mismo paquete de datos en la nube [9].

Si bien varios países pueden alcanzar niveles aceptables de latencia, muchos no reúnen las condiciones necesarias para utilizar servicios en la nube de nivel intermedio y avanzado, los niveles de latencia especificados por Cisco para los requisitos del uso de la nube con calidad de servicio (QoS) son los siguientes, para el nivel básico 160 ms, para el nivel intermedio entre 159 a 100 ms, y para el nivel avanzado menor a 100 ms. Una evaluación realizada por la Naciones Unidas acerca de las condiciones para cumplir con los requisitos QoS propuestos por Cisco para utilizar servicios en la nube de nivel intermedio y avanzado, el cual utilizó datos relativos de 138 economías de diferentes países del mundo, de los cuales 34 economías no cumplen con los requisitos, África está en primera posición con 13 economías, seguida de Asia y Oceanía con 10 economías y América Latina y el Caribe con 11 economías, siendo la latencia el impedimento más importante para este grupo dificultando la ejecución de aplicaciones en tiempo real de servicios en la nube [10].

Según Luis Montero, director de Microsoft, la computación en la nube también carece de estandarización dificultando su seguridad, ya que existe una variedad de dispositivos que engloba decenas de sistemas de comunicación para su interoperabilidad [11]. El costo promedio de un registro perdido o robado continúa aumentando, de acuerdo con el estudio de Infracción de Costos de Datos de IBM Security es globalmente de $150 en 2019 en comparación con un promedio de $148 en 2018 [7].

En conclusión los modelos de nube actuales no están diseñados para volumen, variedad y velocidad de datos que genera el IoT.

La novedosa computación en la niebla (fog computing) propone soluciones para la arquitectura de nube tradicional, cumpliendo un papel de capa intermedia en la red en la que se decide qué datos se procesan localmente y cuáles se envían a la nube o a un centro de datos [9]. El IDC (International Data Corporation) estima que a nivel mundial la cantidad de datos analizados en dispositivos que están físicamente cerca del internet de las cosas es acercándose al 40 por ciento, entonces hay una buena razón para analizar los datos de donde se recopilan con el afán de minimizar la latencia,

gigabytes de tráfico, mantener los datos confidenciales dentro de la red y así abordar casos de uso emergentes del IoT [12].

Un campo de aplicación que requiere de una arquitectura fog computing es el control de tráfico vehicular mediante semáforos y dispositivos móviles inteligentes, ya que necesita la capacidad de procesamiento de datos en tiempo real del volumen de tráfico en las vías y poder gestionar el comportamiento de los semáforos con el objetivo de mejorar la movilidad y dar mayor fluidez, siendo el tráfico vehicular uno de los problemas que enfrenta actualmente la ciudad de Ambato. La Dirección Municipal de Tránsito, Transporte y Movilidad el cual monitorea el tráfico vehicular mediante 26 cámaras de video instaladas en sitios estratégicos del casco central de la ciudad, de los cuales, 11 cámaras registran una intensidad significativa en las Avdas. Cevallos, Los Andes, González Suárez, calles 13 de Abril y Espejo [13]. Además un estudio realizado por Mery Ruiz, Cesar Mayorga, Darwin Aldaz y Jonh Reyes de la Dirección de Investigación y Desarrollo (DIDE) de la Universidad Técnica de Ambato, la congestión vehicular provoca un coste económico social que corresponde al tiempo de viaje del transporte urbano, como resultado cada usuario que utilicé el servicio de transporte urbano debe asumir un valor de USD 27,20 anual, es decir, USD 2,27 mensuales por el mayor uso de horas hombre por viaje, o menor cantidad de viajes que se realizan por congestionamiento vehicular, normalmente este costo social lo asumen las empresas, por el contrario la tarifa sería mayor [14].

El desarrollo del proyecto brindará una importancia teórica práctica para futuras investigaciones destinadas a soluciones a la problemática que tiene la implantación total del IoT con la finalidad de mejorar la calidad de vida, para el caso del transporte si se reduce el tiempo de viaje las personas tardarán menos en llegar a sus centros de trabajo o a sus hogares aumentando su capacidad para ser productivos, además se disminuye la emisión de contaminantes a la atmósfera y optimiza el uso de combustible y así llegar a ser una ciudad sostenible [15].

1.2.2 Fundamentación teórica

1.2.2.1 Tráfico Vehicular

El tráfico vehicular o también llamado tránsito vehicular, es aquel que se produce al existir un flujo de vehículos en una vía, calle o autopista. Dicho flujo puede potencialmente provocar un congestionamiento vehicular, que puede ser corregido mediante la utilización de semáforos [16].

1.2.2.2 Congestión Vehicular

La congestión vehicular es aquella que se presenta cuando existe una saturación producida por el exceso de vehículos en las vías, lo que provoca un incremento en los tiempos de viaje e inconvenientes para circular [17].

1.2.2.3 Semáforos

Los semáforos son dispositivos electromagnéticos y electrónicos proyectados específicamente para facilitar el control de tránsito de vehículos y peatones, mediante indicaciones visuales de luces de colores universalmente aceptados, como lo son rojo, amarillo y verde [18].

En la Figura 1 se puede visualizar un semáforo de control vehicular básico y las partes que le componen:

Figura 1: Partes de un semáforo vehicular.

Elaborado por: El investigador basado en [18].

1.2.2.4 Clasificación de los Semáforos

Los semáforos se pueden clasificar según el mecanismo de operación de sus controles. Según esto, tenemos la siguiente división [19]:

Semáforo vehicular: El semáforo vehicular indica cuando el vehículo debe detenerse o seguir su marcha.

Semáforo peatonal: El semáforo peatonal indica cuando los peatones deben detenerse o cruzar la calle.

Semáforo direccional: El semáforo direccional indica cuando el vehículo debe girar a la derecha o izquierda.

1.2.2.5 Descripción de los colores de las luces del semáforo vehicular

Los colores de las luces del semáforo normalmente son tres rojo, amarillo y verde, a continuación se explica el uso que se da para cada luz en un semáforo vehicular [18]:

Semáforo vehicular en luz roja: El semáforo vehicular en luz roja indica que los conductores deberán detenerse antes de la línea de parada. Ningún peatón debe cruzar la vía, a menos que esté seguro de no interferir con algún vehículo o que un semáforo peatonal le indique su paso. Para determinar el tiempo en rojo se utiliza la Ecuación 1.

$$R = \frac{w+L}{v} \quad \textbf{(Ecuación 1)}$$

En donde:

$w =$ Es el ancho de la intersección en metros (m)

$L =$ Es la longitud del vehículo (valor típico 6,10 m)

$v =$ Es la velocidad de aproximación de los vehículos (m/s)

Semáforo vehicular en luz amarilla: El semáforo vehicular en luz amarilla advierte o anticipa a los conductores el cambio del semáforo a luz a roja, los vehículos deben disminuir la velocidad y detenerse antes de llegar a la línea de pare, mientras que a los

peatones indica que no dispone de tiempo suficiente para cruzar la vía, excepto cuando exista algún semáforo indicándoles que pueden realizar el cruce.
Para determinar el tiempo en ámbar se utiliza la Ecuación 2:

$$A = t + \frac{v}{2a}$$ **(Ecuación 2)**

En donde:
$t =$ Es el tiempo de percepción y reacción del conductor (usualmente 1,00 s)
$v =$ Es la velocidad de aproximación de los vehículos (m/s)
$a =$ Es la tasa de desaceleración $(valor\ usual\ 3{,}05\ m/s^2)$

Semáforo vehicular en luz verde: El semáforo vehicular en luz verde indica que los conductores podrán circular de frente o girar a la derecha o a la izquierda, a menos que alguna señal prohíba dicho giro mientras que los peatones que avancen hacia el semáforo y observen esta luz no podrán cruzar la vía, a menos que otro semáforo indique lo contrario.

1.2.2.6 Parámetros de control de los semáforos vehiculares

Los parámetros de control de los semáforos vehiculares permiten el ajuste periódico o dinámico del tiempo en luz roja, amarillo o verde según las variaciones del volumen de tránsito vehicular y son los siguientes [18] :

Indicación de señal: La indicación de señal del semáforo vehicular se refiere al encendido de una de las luces del semáforo o una combinación de varias luces al mismo tiempo.

Ciclo: El ciclo del semáforo vehicular es el tiempo total requerido para una secuencia completa de las indicaciones del semáforo.

Intervalo: El intervalo del semáforo vehicular es cualquiera de las diversas divisiones del ciclo, durante la cual no cambian las indicaciones de señal del semáforo.

Fase: La fase del semáforo vehicular es la parte del ciclo correspondiente a un movimiento individual, o la combinación de movimientos no conflictivos durante uno o más intervalos de tiempo por ejemplo un solo movimiento vehicular, un solo movimiento peatonal, o una combinación de movimientos vehiculares y peatonales.

1.2.2.7 Semáforos inteligentes

Los semáforos inteligentes son dispositivos de señales luminosas utilizados para gestionar el tráfico, capaces de tomar decisiones en función de las condiciones del tráfico dadas y una serie de parámetros de entrada (como el flujo de vehículos, la velocidad media, etc.) [20].

El funcionamiento de los semáforos inteligentes se basa en los cambios dinámicos de las luces en base a mediciones por inducción del flujo del tráfico en la carretera y según la carga se inclinan a favorecer una u otra dirección. El sistema que posee estos semáforos inteligentes se compone de tres mecanismos: semáforos, sensores o cámaras colocadas en las carreteras y la central de control. Los sensores indica la situación del tráfico a la central de control y así este último toma la decisión de mantener la calle libre de congestión vehicular. Además, el sistema también utiliza un historial de mediciones para determinar si existe algún tipo de ventaja en realizar el cambio de luz como, por ejemplo, bajar los niveles de contaminación y a su vez disminuir el consumo de combustible [21].

1.2.2.8 Fog computing

El fog computing o computación en la niebla es una arquitectura de red que distribuye funciones de computación, almacenamiento y control más cerca a los usuarios [2].

La computación en la niebla es una extensión de la computación tradicional basada en la nube y su uso complementario puede formar una interacción mutuamente beneficiosa al determinar que funciones ofrece mayor ventaja realizarse en la niebla o en la nube, además el fog computing proporciona formas efectivas para superar requerimientos técnicos de latencia, ancho de banda y capacidad de almacenamiento presentes en las arquitecturas informáticas existentes que dependen solo de la computación en la nube y en dispositivos de usuario final [6].

1.2.2.9 Características de la arquitectura fog computing

Las principales características de la arquitectura fog computing se puede resumir em el acrónimo CEAL compuesto por cognición, eficiencia, agilidad y latencia como se explica a continuación [6]:

Cognición: La cognición de la arquitectura fog computing determina adecuadamente dónde llevar a cabo las funciones de computación, almacenamiento y control al estar cerca de la fuente de información. La niebla al estar cerca de los usuarios finales conoce y refleja los requisitos del cliente.

Eficiencia: La eficiencia de la arquitectura fog computing permite integrarse más estrechamente con los sistemas del usuario final y mejorar el rendimiento general, aprovechando al máximo los recursos a lo largo de la comunicación entre la nube y el usuario. Esto es especialmente importante para el rendimiento de sistemas ciberfísicos críticos.

Agilidad: La agilidad de la arquitectura fog computing facilita adaptarse a cambios de manera rápida y económica ya que se experimenta con dispositivos cliente y periféricos en lugar de esperar a que los proveedores de grandes redes inicien o adopten una innovación, además facilita la creación de un mercado abierto para las personas y pequeños equipos a innovar, desarrollar, implementar y operar nuevos servicios.

Latencia: La latencia de la arquitectura fog computing permite el análisis de datos en el borde de la red y puede admitir funciones de control urgentes sensibles al tiempo en sistemas locales y sistemas ciberfísicos por ejemplo en las aplicaciones de inteligencia artificial integradas con tiempos de reacción de milisegundos.

1.2.2.10 Componentes de la arquitectura fog computing

La arquitectura fog computing está compuesta por tres capas como se muestra en la Figura 2, que en conjunto constituye la infraestructura de computación en la niebla que puede ser implementado mediante la virtualización en software o a través de hardware y software dedicado para aplicaciones que admitan computación en la niebla. La capa inferior de la arquitectura fog computing llamada “borde” son aquellos dispositivos

conectados a la red de comunicaciones, la capa intermedia denominada “niebla” corresponde a los servidores que están cerca a los usuarios y permite conectar a los dispositivos en el borde de la red con la capa superior llamada “nube” que son servidores o centros de datos ubicados en todo el mundo [9].

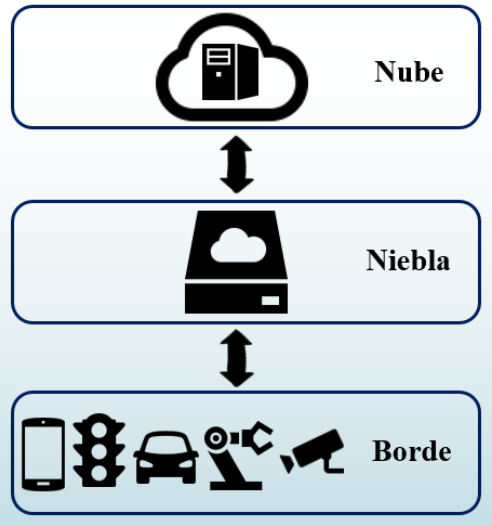


Figura 2: Componentes de la arquitectura fog computing.
Elaborado por: El investigador basado en [9].

Borde: La capa borde de la arquitectura fog computing procesa los datos que se genera en el mismo terminal o envía a un servidor (nodo fog) en la capa de niebla. Está compuesta por todos los dispositivos que se encargan de realizar funciones de adquisición de información en aplicaciones IoT relacionados a la sensorización y control de actuadores en campos que usan cámaras, semáforos inteligentes, dispositivos móviles, maquinaria, vehículos conectados al internet, etc.

Niebla: La capa niebla de la arquitectura fog computing recibe los datos de la capa de borde, los prepara y los envía a la nube si es menester. Está compuesta por servidores de gran rendimiento ubicados cerca de donde se generan los datos de las aplicaciones del internet de las cosas IoT, en el cual se realizan funciones de almacenamiento, calculo, control y comunicación de datos.

Nube: La capa nube de la arquitectura fog computing constituye el punto final en donde se almacena los datos para su análisis posterior. Está compuesta por servidores remotos ubicados en cualquier parte del mundo encargados de virtualizar recursos y

equipos utilizados para almacenamiento, administración de datos y entregar contenido o servicios para distintas aplicaciones a través de internet.

1.2.2.11 Funcionamiento de la arquitectura fog computing

En la arquitectura fog computing el nodo de niebla más cercana al borde de la red recibe los datos generados por los dispositivos IoT, luego la aplicación de niebla dirige la información al lugar óptimo para su procesamiento de datos tomando en cuenta las siguientes condiciones [12]:

- Los datos más sensibles al tiempo se procesan en el nodo de niebla más cercano a las cosas que generan los datos, de esta manera se puede ejecutar aplicaciones IoT para el control y análisis de datos en tiempo real con un tiempo de respuesta de milisegundos, proporcionar almacenamiento transitorio, a menudo de 1 a 2 horas o enviar resúmenes de datos periódicos a la nube.
- Los datos que son menos sensibles al tiempo se envían a la nube para su procesamiento y de esta manera obtener información empresarial, registros históricos, análisis de big data y almacenamiento a largo plazo.

1.2.2.12 Aplicaciones de la arquitectura fog computing

Las aplicaciones de la computación en la niebla son tan diversas como la propia Internet de las cosas, teniendo en común la supervisión o el análisis datos en tiempo real de cosas conectadas a la red y luego iniciar una acción. La acción puede involucrar comunicaciones de máquina a máquina (M2M) o interacción hombre a máquina (HMI). Los ejemplos incluyen bloquear una puerta, cambiar la configuración del equipo, aplicar los frenos en un tren, hacer zoom en una cámara de vídeo, abrir una válvula en respuesta a una lectura de presión, enviar una alerta a un técnico para que realice una acción preventiva, etc. Las aplicaciones que están proliferando rápidamente con el uso de la arquitectura fog computing son la manufactura, petróleo, gas, minería y transporte, [12].

1.2.2.13 Análisis de ventajas y desventajas de la arquitectura fog computing

La computación en la niebla propone diversas soluciones a diferentes problemas inherentes a las infraestructuras de tecnologías de la información basadas en la nube, pero al ser un complemento adicional de la nube tradicional conlleva a tener algunos inconvenientes, como se muestra en la Tabla 1.

Tabla 1: Ventajas y desventajas de la arquitectura fog computing [9].

Ventajas	Desventajas
Menor tráfico: La arquitectura reduce el tráfico entre los dispositivos IoT y la nube.	Costes superiores de hardware: La arquitectura añade una unidad adicional de procesamiento.
Ahorro de costes en el uso de redes externas: La arquitectura evita que se contrate servicios de velocidad de carga en la nube.	Protección limitada ante caídas o abusos: Esta arquitectura no puede verificar si las empresas están equipadas con controladores difíciles de proteger en el borde de la red de los dispositivos de IoT.
Disponibilidad offline: Los dispositivos de IoT en una arquitectura de fog también están disponibles offline.	Necesidad creciente de mantenimiento: El procesamiento de los datos en esta arquitectura tiene un mayor esfuerzo de mantenimiento ya que los controladores están distribuidos por toda la red.
Menor latencia: La computación en la niebla acorta las vías de comunicación, acelerando así los procesos de análisis y decisión.	Mayores requisitos a la seguridad de red: La computación en la niebla es vulnerable a ataques mar in the middle.
Seguridad de los datos: En esta arquitectura los datos sensibles permanecen en la empresa, además pueden cifrarse antes de subirlos a la nube.	

La computación en la niebla principalmente tiene la ventaja de acortar las vías de transferencia y limitar la carga en la nube a un mínimo y su uso tiene beneficios significativos como la agilidad empresarial, seguridad elevada, privacidad de información y menos gasto operativo.

Hay que tener en cuenta que pese a todo los beneficios que brinda la computación en la niebla, el procesamiento descentralizado en el borde de la red también conlleva desventajas que resultan sobre todo del trabajo que requiere mantener y administrar un sistema distribuido, ya que se añade un elemento adicional a la capa de red por medio de nodos fog o unidades de procesamiento.

1.2.2.14 Protocolos de comunicación de la arquitectura fog computing

Los protocolos de comunicación utilizados en la arquitectura fog computing son los siguientes: HTTP Rest, MQTT y WebSocket, los cuales son diseñados para la comunicación segura de nodo a dispositivo.

En la Tabla 2 se muestra las características que tienen estos protocolos.

Tabla 2: Características de los protocolos de comunicación de la arquitectura fog computing [22].

	HTTP Rest	MQTT	WebSocket
Protocolo de Transporte	TCP	TCP	TCP
Seguridad	SSL/TLS	SSL/TLS	WSS(TLS)
Con estado	NO	SI	SI
Tipo de mensajes	Solicitar Respuesta	Publicar, Suscribirse y Solicitar respuesta	Publicar, Suscribirse y Solicitar respuesta
Características Principales	- Estandarizado para servicios web - Protocolo cliente servidor sin estado - Alto consumo de ancho de banda	- Enfocado para el uso en telemetría. - Posee tres niveles de QoS - Bajo consumo de recursos	-Diseñada para servidores web -Conexión full dúplex entre el cliente y servidor - Alto consumo de ancho de banda

Los protocolos de comunicación nombrados anteriormente son alternativas para realizar la comunicación en una infraestructura fog ya que presentan características de conexión cliente servidor en un solo canal de comunicaciones TCP diseñados para mantener un bajo consumo de recursos y reducir la latencia y el tráfico innecesario de datos, además presenta mecanismos de seguridad para cumplir los requerimientos a soluciones de aplicaciones IoT que necesitan una comunicación permanente sin la necesidad de realizar cada vez una petición al servidor o mantener variables de sección activas.

1.2.2.15 Protocolo de comunicación MQTT

El protocolo de comunicación MQTT (Message Queue Telemetry Transport) es un protocolo basado en TCP desarrollado por IBM y luego liberado para su desarrollo en código abierto, está diseñado para conexiones con ubicaciones remotas donde se requiera el envío de datos a dispositivos con restricciones de memoria, potencia de procesamiento limitado y redes de bajo ancho de banda y alta latencia [23].

1.2.2.16 Funcionamiento del protocolo de comunicación MQTT

El funcionamiento del protocolo de comunicación MQTT se encuentra compuesto por dos elementos: los nodos publicadores/suscriptores y un nodo central de gestión o broker. La operación inicia en el broquer el cual recopila los datos que los nodos publicadores le transmiten y dependiendo de las políticas gestionadas por el broker sobre publicación y suscripción se enviarán hacia los nodos suscriptores que solicitan la información. Para lograr esta tarea, interviene el concepto de "topic" o "tema" ya que a través de estos temas se articula la comunicación entre nodos, puesto que los nodos publicadores o suscriptores deben formar parte de un tema común para poder establecer la comunicación [23].

Como se observa en la Figura 3, el nodo suscriptor envía un mensaje de suscripción (tema) para informar al broker de su interés en el tema indicado, mientras que el nodo publicador envía un mensaje de publicación (tema, datos) que contiene los datos que se van a publicar junto con el tema relacionado. Si existe una coincidencia entre los temas, el broker transfiere el mensaje de publicación (tema, datos) hacia el suscriptor [23].

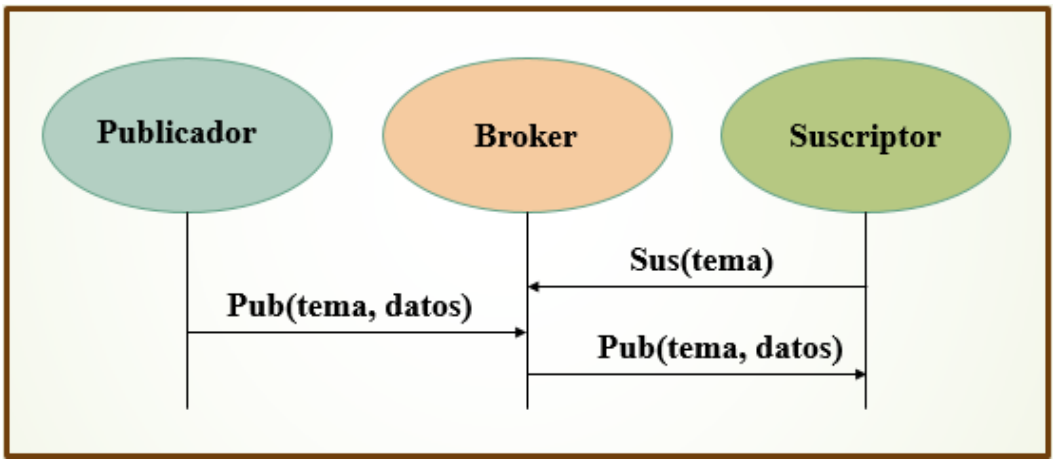


Figura 3: Modelo de funcionamiento del MQTT basado en temas [23].

1.2.2.17 Calidad de servicio para el protocolo de comunicación MQTT

La calidad de servicio (QoS) es el conjunto de requisitos que la red debe cumplir para asegurar un nivel de servicio adecuado en la transmisión de los datos [24]. El protocolo de comunicación MQTT tiene integrado en modo nativo para la calidad de servicio con la posibilidad de definir en los nodos publicadores el QoS mediante tres niveles ya establecidos [23]:

QoS nivel 0: El Qos de nivel 0 entrega el mensaje una sola vez. Esto significa que el mensaje se envía sin garantías de la recepción. El broker no informa al nodo emisor sobre si se ha recibido el mensaje.

QoS nivel 1: El Qos de nivel 1 entrega el mensaje al menos una vez. El nodo emisor se encarga de transmitir el mensaje varias veces si es necesario, hasta que el broker confirme que ha recibido el mensaje.

QoS nivel 2: El QoS de nivel 2 guarda el mensaje obligatoriamente en el nodo emisor, el cual lo transmitirá siempre que el nodo receptor no confirme su recepción. La principal diferencia radica en que el emisor utiliza una fase de reconocimiento más sofisticada con el broker para evitar la duplicación de los mensajes, siendo un proceso más lento, pero más seguro.

1.2.2.18 Protocolo de comunicación Websockets

El protocolo WebSocket permite una comunicación bidireccional entre un cliente que ejecuta un código no fiable en un entorno controlado a un host remoto que ha optado por las comunicaciones a partir de ese código. El modelo de seguridad utilizado para esto es el modelo de seguridad basado en el origen comúnmente utilizado por los navegadores web. El protocolo consiste en un handshake abierto seguido de un enmarcado básico de mensajes, en capas sobre TCP. El objetivo de esta tecnología es proporcionar un mecanismo para las aplicaciones basadas en navegadores que necesitan una comunicación bidireccional con servidores que no dependen de la apertura de múltiples conexiones HTTP [25].

Websockets puede ser usado como un conducto externo para el protocolo MQTT y así colocar el paquete MQTT en un paquete websockets y ser enviado al cliente, para que este último realice el proceso contrario de desempaqueta el paquete MQTT del paquete websockets y luego lo procesa como un paquete MQTT normal. Mqtt sobre Websockets permite recibir datos MQTT directamente en un navegador web para que aproveche todas las funciones de MQTT, generalmente se usa el puerto 9001 pero no es fijo [26].

1.2.2.19 Servidor Proxy

Un Servidor Proxy actúa como intermediario entre el navegador (cliente) y los servicios de Internet como sitios web o plataformas de software como servicio (SaaS). Un servidor proxy se coloca delante de un cliente o una red de clientes y gestiona en su nombre todo el tráfico, permitiendo que la conexión sea un poco más rápida, además proporciona mayor privacidad y seguridad. Esto último debido a que puede mejorar la seguridad actuando como rostro público exclusivo de su red. Vistos desde el exterior, todos los usuarios de la red son anónimos, pues quedan ocultos tras la dirección IP del proxy [27] .

1.2.2.20 Protocolo de transporte TCP

El protocolo TCP que significa (Protocolo de Control de Transmisión) permite la transmisión de datos entre dos máquinas y el control de la misma, además posibilita la administración de los datos que vienen del Internet Protocol (IP). Cuando se usa este protocolo, las aplicaciones pueden establecer una conexión de forma segura, la máquina emisora (cliente) es la que solicita la conexión, y la máquina receptora (servidor) es quien recibe la petición, por eso se dice que TCP es en un entorno Cliente-Servidor [28].

Entre las características principales del protocolo de transporte TCP se pueden mencionar [28]:

- El protocolo TCP permite el monitoreo del flujo de los datos y así evitar la saturación de la red.
- El protocolo TCP permite poner nuevamente los datagramas o paquete de datos en orden cuando vienen del protocolo IP.
- El protocolo TCP permite que los datos se formen en segmentos de longitud variada para entregarlos al protocolo IP.
- El protocolo TCP admite la multiplexación de los datos, es decir, que la información que proviene de diferentes fuentes pueda circular simultáneamente por la misma línea de transporte.

Para establecer una comunicación utilizando el protocolo TCP se requiere que los puertos de comunicación estén abiertos y se realiza mediante tres etapas que se explican a continuación [28]:

Primero se establece la conexión usando el procedimiento llamado negociación en tres pasos o 3-way handshake que permiten sincronizar al cliente y al servidor para empezar la comunicación, luego se realiza la transferencia de datos utilizando mecanismos para ordenar paquetes de datos recibidos, detectar errores ocurridos durante la transmisión y paquetes de datos duplicados, por último se finaliza la conexión mediante el procedimiento llamado negociación en cuatro pasos o four-way

handshake, el cual termina la comunicación independientemente desde cada lado del servidor o cliente.

1.2.2.21 Internet de las cosas (IoT)

El internet de las cosas es la interconexión de objetos cotidianos con internet, se conectan a la red para enviar información que obtienen por medio de sensores o incluso para permitir que distintos sistemas interactúen con el mundo a través de actuadores.
El término internet de las cosas se usa con una denotación de conexión avanzada de dispositivos, sistemas y servicios que va más allá de la aplicación tradicional de las comunicaciones M2M o comunicaciones máquina a máquina y cubre una amplia variedad de protocolos, dominios y aplicaciones.
Los casos de uso y las oportunidades que el IoT proporciona a las diferentes industrias son numerosas, lo que hace que se vayan definiendo un conjunto de retos y patrones a resolver. Algunos de estos retos tienen que ver con la interconexión de hardware, software, sistemas operativos o requerimientos de los gateways de la red [22].

1.2.2.22 Componentes de la arquitectura IoT

La arquitectura de un sistema IoT permite que varios objetos separados físicamente estén conectados entre sí mediante una red de comunicaciones, está compuesto por tres elementos básicos: dispositivos, puerta de enlace y nube, como se ve en la Figura 4.

Dispositivos: Los dispositivos de la arquitectura IoT corresponden a todos los objetos conectados a internet con su propio software y hardware, para ello deben enlazarse en red entre ellos y a servicios en la nube, estos dispositivos están dotados de sensores para recolectar datos y actuadores para realizar tareas que se envían desde la nube.

Puerta de enlace: La puerta de enlace (gateway) de la arquitectura IoT permite que los dispositivos que no están directamente conectados a internet puedan alcanzar los servicios en la nube, el gateway es el encargado de mover datos a la nube y viceversa,

de esta manera poder recibir tareas que se envían desde la nube para efectuarse en los dispositivos IoT.

Nube: La nube de la arquitectura IoT permite procesar y combinar los datos de los diferentes dispositivos, además de analizarlos, tomar decisiones y actuar sobre ellos mediante servidores remotos ubicados en cualquier parte del mundo. [22]

Figura 4: Componentes de la arquitectura IoT.
Elaborado por: El investigador basado en [22].

1.2.2.23 Telefonía móvil celular

La telefonía móvil celular es un sistema de comunicación inalámbrico formada por celdas, cada una con su propia estación base de pequeña o mediana potencia para dar servicio a un área limitada [29], como se ve en la Figura 5, con la finalidad de incrementar la capacidad de usuarios, reducir el uso de energía y ofrecer mayor cobertura [30].

Las características que tiene la telefonía móvil celular son [30]:

- Cada celda en la telefonía móvil celular puede utilizar una subbanda o subconjunto de frecuencias, dentro de la banda total que el operador tenga asignada.

- Una celda en la telefonía móvil celular sólo ofrece una parte de todos los radiocanales de los que el operador dispone.
- Los radiocanales de una celda en la telefonía móvil celular se comparten entre todos los móviles que están en la celda y se asignan de forma dinámica permitiendo reutilizar dichos radiocanales.
- El diseño del número de radiocanales en la telefonía móvil celular necesarios en una celda se calcula en función del tráfico esperado que responda a la probabilidad del bloqueo de llamada.

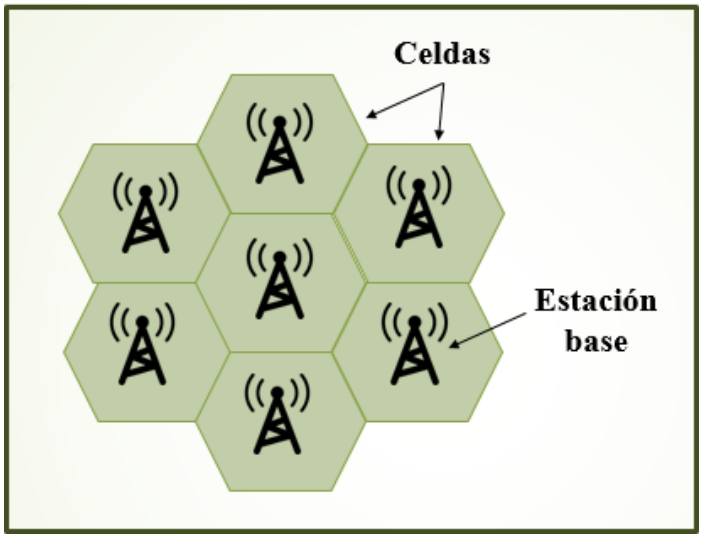


Figura 5: Sistema de telefonía celular.

Elaborado por: El investigador basado en [29] .

Los sistemas de telefonía móvil celular ha tenido un gran desarrollo desde sus inicios que fue la primera generación 1G caracterizada por ser análoga y estrictamente para voz la cual presentaba una calidad de los enlaces de voz muy baja, transferencia entre celdas muy imprecisa, baja capacidad de usuarios y la seguridad no existía.

Con el avance de la telefonía móvil apareció la segunda generación 2G ya digital que soportaba velocidades de información más altas para voz pero limitadas en comunicaciones de datos, ya ofrecía diferentes niveles de encriptación utilizados en servicio tales como datos, fax y SMS.

Seguida de la segunda generación apareció la tercera generación 3G en la que se encuentra el sistema universal de comunicaciones móviles o UMTS el cual tuvo como

objetivo aumentar la capacidad de transmisión de datos con la finalidad de convertir la red de telefonía móvil en una red de telecomunicaciones móvil.

Actualmente ya se utiliza la cuarta generación 4G o LTE con el estándar de comunicaciones inalámbricas de transmisión de datos de alta velocidad que es 50 veces mayor que la tercera generación y se habla de un proyecto a largo plazo para la quinta generación o 5G, que permiten resolver los retos que presenta el IoT [31].

1.2.2.24 Sistema universal de comunicaciones móviles (UMTS)

El sistema universal de comunicaciones móviles es un sistema de tercera generación estandarizado para la convergencia de voz y datos con acceso inalámbrico a Internet, aplicaciones multimedia y altas transmisiones de datos [31].

Los protocolos empleados en los sistemas 3G soportan altas velocidades de información enfocados para aplicaciones más allá de la voz tales como audio (MP3), video en movimiento, video conferencia y acceso rápido a Internet.

El impulso de los estándares de la 3G es apoyada por la ITU (International Telecomunications Union) y a este proyecto se le conoce como IMT-2000 (International Mobile Telephone), que busca incrementar y extender el potencial de las tecnologías móviles, inalámbricas y satelitales.

Las características del sistema universal de comunicaciones son las siguientes [30]:

- Tasa transferencia de datos de 144 hasta 512 Kbps para áreas de cobertura amplias y puede llegar hasta los 2 Mbps en áreas locales.
- Latencia aproximada de 200 ms.
- Está basado en la multiplicación por división de código o CDMA
- La tercera generación está estandarizada internacionalmente por el 3GPP.

1.2.2.25 Evolución a largo plazo (LTE)

La evolución a largo plazo o LTE es una tecnología de cuarta generación con velocidades de subida y bajada de datos superior, tiene menor latencia y mayor alcance, permitiendo un enlace de video con terminales móviles de alta calidad [32].

Los sistemas con tecnología LTE son totalmente tecnologías IP, que hacen uso de las características de calidad de servicio QoS permitiendo al usuario y al proveedor de servicio priorizar el tráfico de datos dependiente del tipo de aplicación que esté utilizando el ancho de banda.

El despliegue de las redes 4G, mediante la computación de paquetes, mantiene el servicio de voz y a la vez ayuda a mejorar la funcionalidad de aplicaciones como video conferencias, accesos a videos de alta calidad, juegos en tiempo real, televisión móvil, etc. Además las redes de cuarta generación ofrecen un ancho de banda mayor a los usuarios en movimiento a altas velocidades dentro del área de cobertura manteniendo el QoS [33].

Las características del sistema en evolución a largo plazo son [33] :

- Velocidades de transmisión de 100 Mbps de subida y 50 Mbps de bajada.
- Latencia en el usuario inferior a 10 ms.
- Está basado en el acceso múltiple por división de frecuencias ortogonales u OFDMA.
- La cuarta generación está estandarizada por el 3GPP.

1.2.2.26 La quinta generación (5G)

La quinta generación será diseñada para brindar el nivel de rendimiento necesario del IoT y satisfacer las necesidades de comunicación de miles de millones de dispositivos conectados al internet, con las compensaciones correctas entre velocidad, latencia y costo.

La generación 5G se orienta a soluciones de aplicaciones IoT e infraestructuras de comunicaciones críticas en distintos entornos, por ejemplo, la implementación de servicios como vehículos sin conductor requieren un tiempo de respuesta muy rápido a velocidades bajas y los servicios de base empresarial en la nube con análisis de datos masivos requieren velocidades altas en lugar de baja latencia.

La quinta generación puede llegar a tener velocidades altas ya que utiliza el rango de frecuencias más cortas en la que se encuentran las ondas milimétricas entre 30GHz y

300GHz, siendo la principal la razón por la cual el 5G puede ser más rápida ya que a menor frecuencia mayor es el ancho de banda.

Para la implementación de la tecnología 5G en el sistema de telefonía móvil celular se necesita que los operadores de redes móviles puedan funcionar en un nuevo espectro es decir en el rango de 6 a 300 GHz, lo que significa inversiones masivas en la infraestructura de la red, además para alcanzar latencias bajas de 1 ms, implica que la red de la estación base del sistema celular se conecte a través de fibra óptica [34].

Las características que tendrá el 5G son las que se describen a continuación [34]:

- Tasa de datos de hasta 10 Gbps es decir de 10 a 100 veces mejor que las redes 4G.
- Latencia de 1 ms.
- Cobertura del 100%.
- Disminución del 90% en el consumo de energía de la red.

1.2.2.27 Estándar 802.11 (WIFI)

WiFi es un protocolo de salto-único para redes ad-hoc que provee funcionalidades de ahorro de energía, generalmente para altas velocidades de transmisión, y los transceptores disponibles requieren una cantidad mayor de energía. Dado que los nodos pueden tener que recibir tramas en cualquier momento, necesitan monitorear los medios de forma permanente. Los estándares 802.11 se centran en los dos niveles inferiores del modelo ISO, la capa física y la capa de enlace. Una tecnología que compite con este estándar es HiperLAN2 que es una tecnología de LAN inalámbrica que funciona en la banda U-NII de 5 GHz (5,4 a 5,7 GHz) sin licencia [35] .

1.2.2.28 Sistema de Posicionamiento Global

El sistema de Posicionamiento Global o GPS es un sistema que tiene como objetivo determinar las coordenadas espaciales de puntos que están ubicados en cualquier lugar del planeta, estos puntos pueden estar estáticos o en movimiento y se pueden obtener en cualquier momento del día [36].

1.2.2.29 Funcionamiento del GPS

El funcionamiento del sistema GPS para obtención de coordenadas de un punto se basa en la determinación simultánea de las distancias d1, d2, d3 y d4 a cuatro satélites mínimo como se muestra en la Figura 6, estas distancias se obtienen a partir de las señales emitidas por los satélites las cuales son recibidas por receptores especialmente diseñados [36], un receptor GPS para uso civil puede alcanzar una precisión de localización que varía entre 4 y 16 metros y va a depender del número de satélites que se conecte el receptor [37].

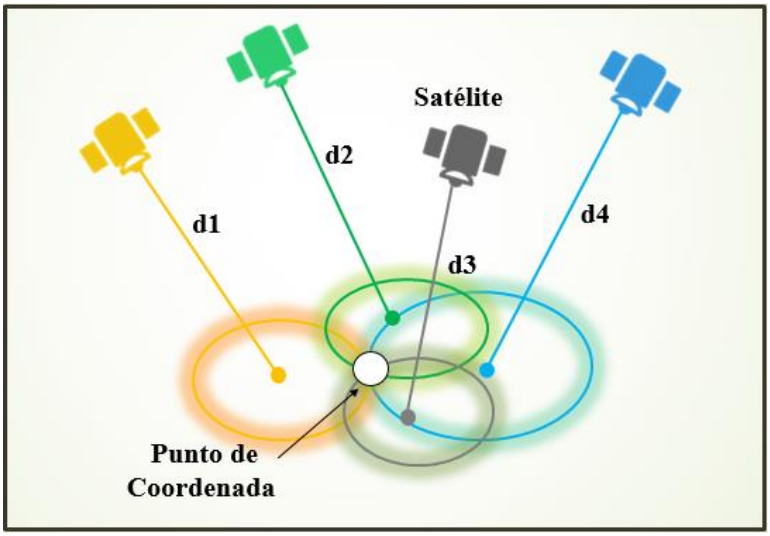


Figura 6: Funcionamiento del GPS**.**

Elaborado por: El investigador basado en [36].

1.2.2.30 Sistema de Coordenadas Geográficas

El sistema de coordenadas geográficas es el sistema que referencia cualquier punto de la superficie terrestre a través de dos coordenadas angulares, latitud y longitud**,** se indican en la Figura 7.

Latitud: La latitud proporciona la localización de un lugar en dirección Norte o Sur desde el Ecuador y se expresa en medidas angulares que varían desde los 0º del Ecuador hasta los 90ºN (+90º) del polo Norte o los 90ºS (-90º) del polo Sur.

Longitud: La longitud proporciona la localización de un lugar en dirección Este u Oeste desde el meridiano de referencia 0º o meridiano de Greenwich, expresándose en medidas angulares comprendidas desde los 0º hasta 180ºE (+180º) y 180ºW (-180º) [38].

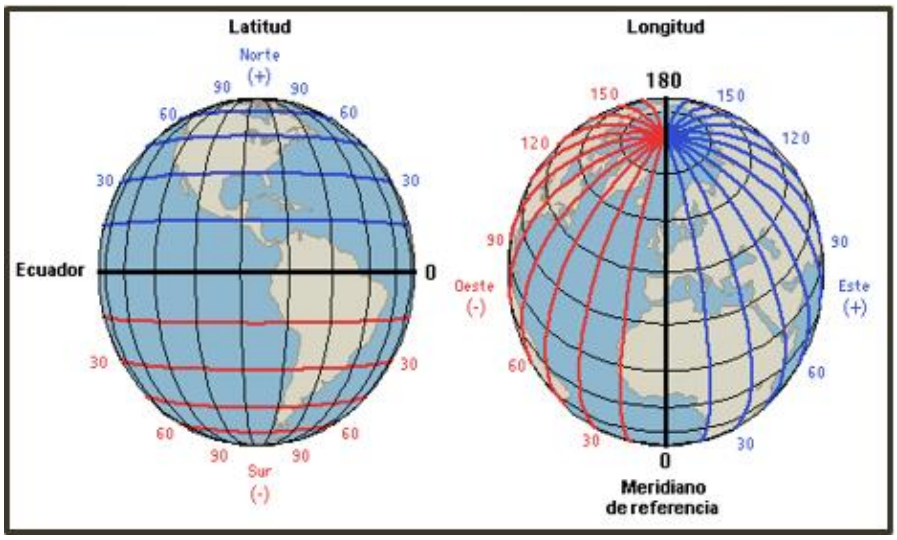


Figura 7: Latitud y longitud del globo terráqueo [39].

1.2.2.31 Servidor LAMP

Un servidor LAMP es un servidor creado con un software probado para ofrecer aplicaciones web de alto rendimiento, mediante una plataforma de desarrollo web de código abierto que utiliza Linux como sistema operativo flexible, Apache como el servidor web, MySQL como sistema de gestión de bases de datos y PHP como lenguaje de programación orientado a objetos muy seguro, a veces se utiliza Perl o Python en lugar de PHP.

- Linux es un sistema operativo, gratuito, libre y con diferentes versiones dependiendo de la necesidad del usuario.
- Apache es un servidor HTTP de código abierto para sistemas operativos modernos, diseñado para proporcionar un servidor seguro, eficiente y escalable que puede proporcionar servicios HTTP sincronizados con los estándares actuales.
- MySQL es una base de datos de código abierto en la que se almacena información de datos y los administra de manera eficiente y fácil.

- PHP es un lenguaje desarrollado a partir de PERL, utilizado para desarrollar scripts de manera más fácil, segura, orientados a objetos, eficiente e independientes de la arquitectura en la que se aplique [40].

1.2.2.32 Lenguaje de programación HTML

El lenguaje de programación HTML (HyperText Markup Language) o en español lenguaje de marcado de hipertexto, describe la estructura y el contenido de una página web, ya que en ella se representa textos con estructura y formato sobre la capa de presentación de la página. El navegador interpreta todo el lenguaje o código de principio a fin, interpreta desde la primera línea y así sucesivamente hasta llega a la última línea del código y una vez que termine de interpretarla se dice que la página se ha se ha cargado completamente [41].

1.2.2.33 Lenguaje de programación JavaScrip

El lenguaje JavaScript permite configurar la interactividad y el dinamismo de la página web. A través de los eventos que son acciones que pueden suceder una vez se ha cargado la página completamente, las acciones programadas en este lenguaje modifican el aspecto, estructura o visualización de la página. la ventaja de este lenguaje es que es interpretado por el navegador. El código Javascript viaja a través de la red desde el servidor tal cual se ha programado, el navegador realiza las llamadas a las funciones Javascript programadas, como si este estuviera embebido en la página web [41].

1.3 Objetivos

1.3.1 Objetivo general

El objetivo general del proyecto de investigación es implementar un sistema de control de tráfico vehicular aplicando la arquitectura fog computing el cual permite procesar el volumen de tráfico en la Avenida Cevallos y Calle Eugenio Espejo de la ciudad de Ambato y actuar en los intervalos de tiempo de semaforización en dicha intersección.

1.3.2 Objetivos específicos

Analizar la arquitectura fog computing y las tecnologías de comunicación en dispositivos IoT.

- Identificación de las tecnologías y protocolos de comunicación en la arquitectura fog computing y dispositivos IoT.
- Selección del hardware y software adecuado para la arquitectura fog computing y dispositivos IoT.

Desarrollar un servidor fog y un servidor cloud, para la comunicación, almacenamiento y procesamiento de la información del sistema.

- Instalación de un servidor de niebla o fog y un servidor en la nube o cloud.
- Determinación de reglas en el servidor fog y cloud para el control de tráfico de datos en la red.
- Diseño de la interfaz web en el servidor de niebla y en la nube para el monitoreo y registros históricos del sistema.

Diseñar un prototipo de sistema de control de tráfico vehicular utilizando la arquitectura fog computing en la Avenida Cevallos y Calle Eugenio Espejo de la ciudad de Ambato.

- Diseño de un algoritmo para el control de tráfico vehicular.
- Construcción del prototipo de semáforo inteligente.
- Realización de pruebas de funcionamiento del sistema.
- Comparación de los parámetros técnicos de la arquitectura fog con cloud computing.

CAPÍTULO II

METODOLOGÍA

2.1 Materiales

Para el diseño e implementación del presente proyecto de investigación se requiere información de la Dirección de Tránsito, Transporte Terrestre y Seguridad Vial de la ciudad de Ambato acerca del sistema de semaforización actual, congestión vehicular, además de artículos, revistas, tesis, libros y estudios realizados acerca de la arquitectura de comunicaciones fog computing.

2.2 Métodos

2.2.1 Modalidad de la investigación

Para el desarrollo del presente proyecto de investigación se utilizará investigación aplicada, debido a que se pondrá en práctica los conocimientos adquiridos para la implementación y programación de servidores y dispositivos IoT para el sistema de control de tráfico vehicular.

Para la recolección de información se empleará una investigación bibliográfica, debido a que se analizaran artículos científicos, libros, tesis relacionadas con el tema, e información web para conocer los requerimientos y especificaciones de la arquitectura fog computing en el campo del control de tráfico vehicular mediante la implementación de semáforos inteligentes.

La presente investigación será de carácter experimental, ya que a través de pruebas se analizarán las ventajas que tiene el uso de la arquitectura fog a comparación del cloud computing.

Además será investigación de campo debido a que se tendrá datos reales sobre el funcionamiento actual del sistema de semaforización instalado en la ciudad de

Ambato, lo que permitirá diseñar adecuadamente el sistema de control de tráfico vehicular.

2.2.2 Recolección de información

Para la recolección de la información se analizará artículos, libros, estudios y tesis desarrolladas en los últimos años, cada uno de estos relacionados a la arquitectura fog computing y su aplicación en el control de tráfico vehicular. Además, se recopilará información del Consorcio OpenFog el cual define normas y estándares para la implementación de esta arquitectura.

2.2.3 Procesamiento y análisis de datos

Para el procesamiento y análisis de datos se llevará a cabo los pasos descritos a continuación:

- Lectura comprensiva de la información recopilada.
- Interpretación y optimización de la información.
- Determinación de la mejor alternativa para la solución del problema.
- Planteamiento de la propuesta de solución.

2.2.4 Desarrollo del proyecto

Para cumplir con los objetivos planteados en el proyecto de investigación se llevará a cabo los siguientes pasos:

- Análisis del sistema actual de semaforización y congestión vehicular en la zona centro de Ambato
- Análisis de la arquitectura fog computing.
- Identificación de las tecnologías y protocolos de comunicación en dispositivos IoT.
- Selección del hardware y software para el desarrollo del sistema.

- Instalación de un servidor de niebla o fog y un servidor en la nube o cloud.
- Determinación de reglas en el servidor fog y cloud para el control de tráfico de datos en la red.
- Elaboración de una aplicación móvil para obtener la ubicación e información del vehículo.
- Diseño de un algoritmo para el control de tráfico vehicular para la intersección ubicada en la Avenida Cevallos y Calle Eugenio Espejo de la ciudad de Ambato.
- Diseño de la interfaz web en el servidor de niebla para monitorear el funcionamiento de los semáforos, estado del sistema y la posición de los vehículos en tiempo real.
- Diseño de la interfaz web para el servidor en la nube para obtener registros históricos del sistema de control de tráfico vehicular sobre el conteo de vehículos y fases de semaforización.
- Construcción del prototipo de semáforo inteligente.
- Programación de las funciones del semáforo inteligente.
- Realización de pruebas de funcionamiento del sistema.
- Comparación de los parámetros técnicos de la arquitectura fog con cloud computing.
- Elaboración del informe final.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1 Introducción

El sistema de control de tráfico vehicular implementado actualmente en la ciudad de Ambato es un campo de aplicación que requiere el uso de nuevas tecnologías que permita hacer de este un servicio que satisfaga los requerimientos de los usuarios y además brinde un crecimiento tecnológico en el contexto de ciudad inteligente.

Para ello el presente proyecto se basa en la implementación de la arquitectura fog computing para el control de tráfico vehicular, el cual permite la gestión eficiente del tránsito mediante el procesamiento de datos en tiempo real del volumen de tráfico en las vías y el cambio de los intervalos de tiempo de semaforización de manera automática.

3.2 Estudio de factibilidad

A continuación se realiza el análisis de factibilidad que tiene el proyecto para su adecuado desarrollo:

3.2.1 Factibilidad técnica

El proyecto es técnicamente factible ya que las tecnologías que se ha propuesto para el desarrollo del proyecto se han venido estudiando a lo largo de la carrera y los elementos o dispositivos electrónicos para su implementación se pueden adquirir en el país.

3.2.2 Factibilidad Económica

El desarrollo del proyecto de investigación presenta una factibilidad económica ya que supone un bajo coste para su implementación mediante hardware y software libre, la misma que será financiada por el investigador.

3.2.3 Factibilidad Bibliográfica

La investigación presenta factibilidad bibliográfica ya que se encuentra una variedad de libros, tesis, artículos científicos y estudios realizados por organizaciones a nivel mundial sobre la arquitectura de comunicaciones fog computing y acerca de la semaforización en la gestión de tránsito vehicular.

3.3 Análisis de la congestión vehicular en la zona centro de Ambato

La congestión vehicular en la zona centro de la ciudad de Ambato según la Dirección de Tránsito, Transporte Terrestre y Seguridad Vial se da por diferentes factores especialmente en horas pico en los días de feria como son, lunes, miércoles, viernes y sábados aproximadamente entre las 08h00 a 19h00 horas, en las diferentes Avenidas y calles del casco central de la ciudad, específicamente en los alrededores de los mercados, paradas de buses, sector bancario, comercio informal, así como también el irrespeto de las leyes de transito por parte de conductores y peatones [42].

3.4 Análisis del sistema actual de semaforización en la zona centro de Ambato

El sistema semafórico centralizado del casco central urbano son controlados y operados por los técnicos de monitoreo semafórico desde el “Centro de Gestión de Tránsito de Ambato” mediante el sistema Adimot implementado desde octubre de 2013.

La estructura del sistema centralizado de semaforización está compuesto por el centro de control y los elementos en calle dotado de una red de comunicaciones a través de fibra óptica. El centro de control está conformada por el cuarto de equipos donde se encuentra el banco de baterías y la rack de fibra óptica conectadas a la sala de monitoreo y gestión, los elementos en calle está compuesto por el regulador de tránsito, los semáforos y las cámaras para el conteo vehicular comunicadas inalámbricamente con el centro de control como se ve en la Figura 8:

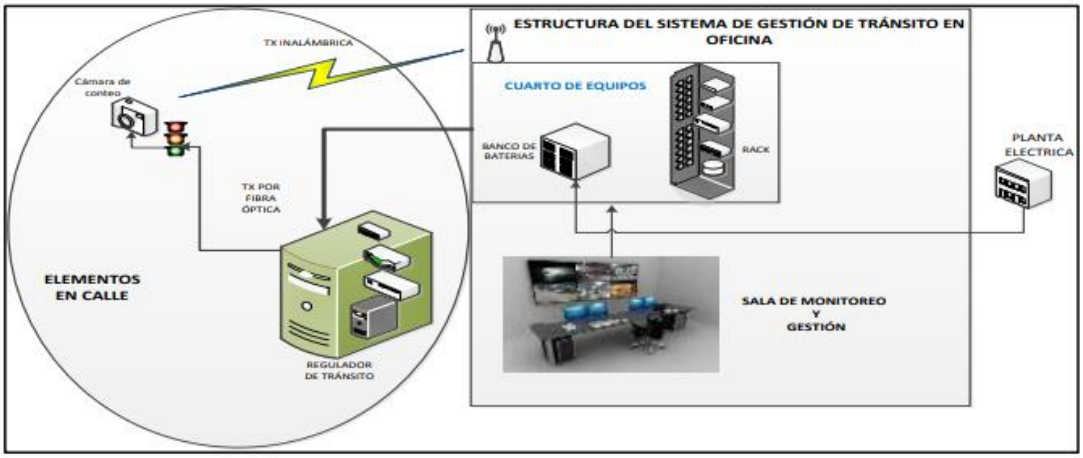


Figura 8: Estructura del sistema centralizado de semaforización [42].

Los semáforos del sistema centralizado tienen características de ser semi adaptativos o micro regulados es decir que las decisiones de su funcionamiento se toman por parte del técnico que se encuentra en el centro de control, el cual puede cambiar los tiempos de fases y crear planes de tráfico en horarios para brindar una mayor fluidez en el tránsito.

3.5 Requerimientos del sistema

Para el desarrollo del sistema de control de tráfico vehicular a través de la arquitectura de comunicaciones fog computing, el cual permite actuar en los intervalos de tiempo de los semáforos según el número de vehículos que se encuentran en la Avenida Cevallos y Calle Eugenio Espejo de la ciudad de Ambato, se especifican los siguientes requerimientos:

- Obtener la ubicación de los vehículos en tiempo real y comunicar con un servidor en la niebla.
- Comunicar, almacenar y procesar los datos en el servidor de niebla correspondientes a ubicaciones de los vehículos e intervalos de tiempo para los semáforos inteligentes.
- Comunicar y almacenar datos históricos del sistema con un servidor en la nube.
- Visualizar los datos en una interfaz web tanto en la niebla como en la nube.

3.6 Diagrama de bloques del sistema

El diagrama de bloques del sistema para el control de tráfico vehicular basado en la arquitectura fog computing está compuesta por tres etapas como se observa en la Figura 9. En la primera etapa se encuentran los dispositivos IoT que son los semáforos inteligentes y los dispositivos móviles. La adquisición de datos se realiza través de una aplicación instalada en el dispositivo móvil para obtener la ubicación en tiempo real mediante el sensor GPS del dispositivo en el vehículo. En la segunda etapa se encuentra el servidor de niebla el cual almacena los datos de la ubicación de los vehículos, procesa dicha información para actuar en el funcionamiento de los semáforos inteligentes y además se visualizan los datos en una interfaz de monitoreo. La tercera etapa está compuesta por el servidor en la nube el cual almacena los datos y presenta históricos del sistema de tráfico vehicular. La comunicación de datos entre el dispositivo móvil y la niebla se realiza mediante la red de telefonía móvil celular 4G LTE, mientras que la comunicación entre la niebla y los semáforos inteligentes se establece mediante la red de área local inalámbrica conocida como WLAN usando el estándar inalámbrico Wifi, y la comunicación entre la niebla y la nube de igual manera mediante comunicación inalámbrica.

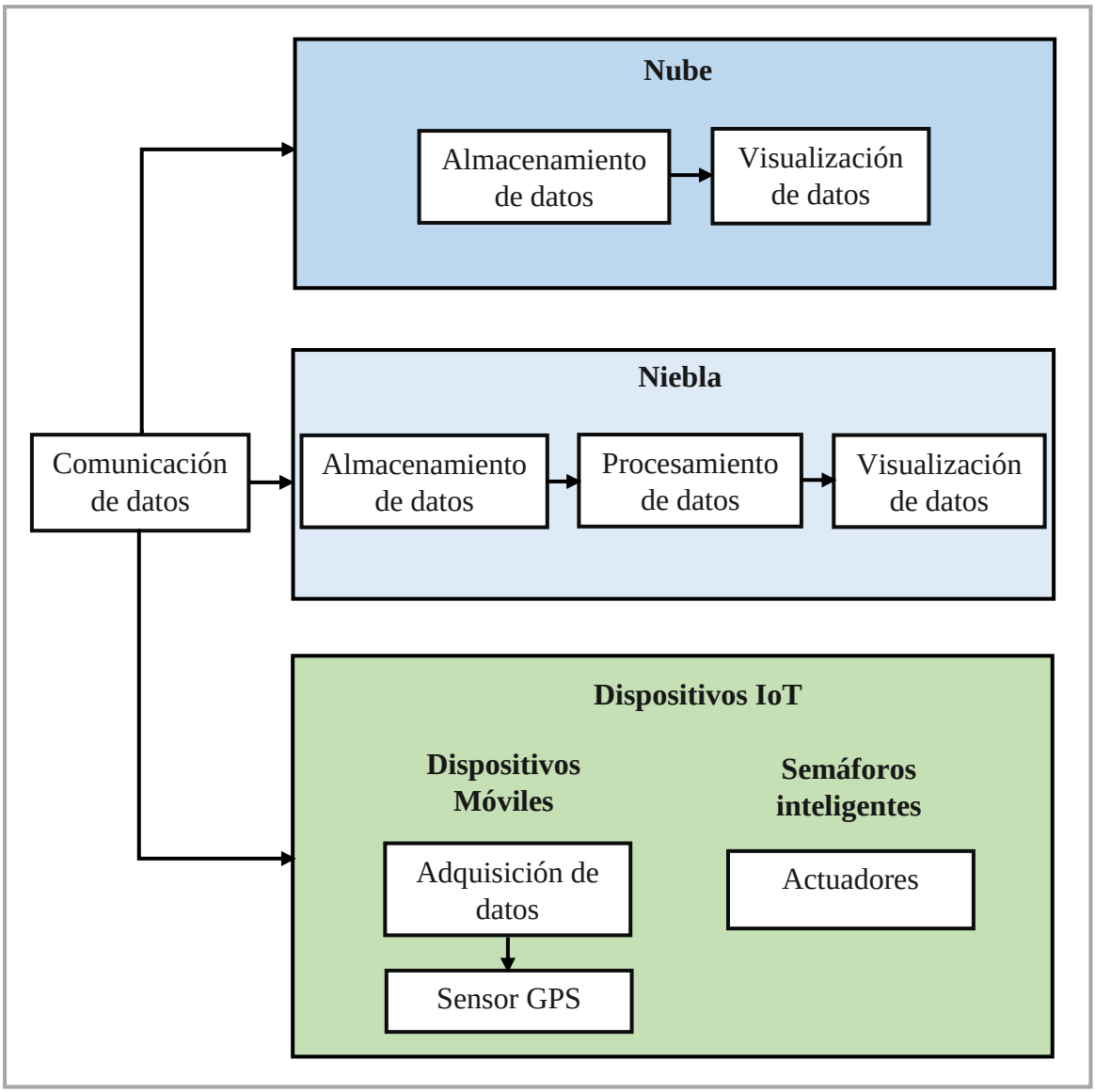


Figura 9: Diagrama de bloques del sistema.

Elaborado por: El investigador.

En la Figura 10, se representa gráficamente los componentes de la arquitectura fog computing para el control de tráfico vehicular en la intersección ubicada en la Avenida Cevallos y Calle Eugenio Espejo.

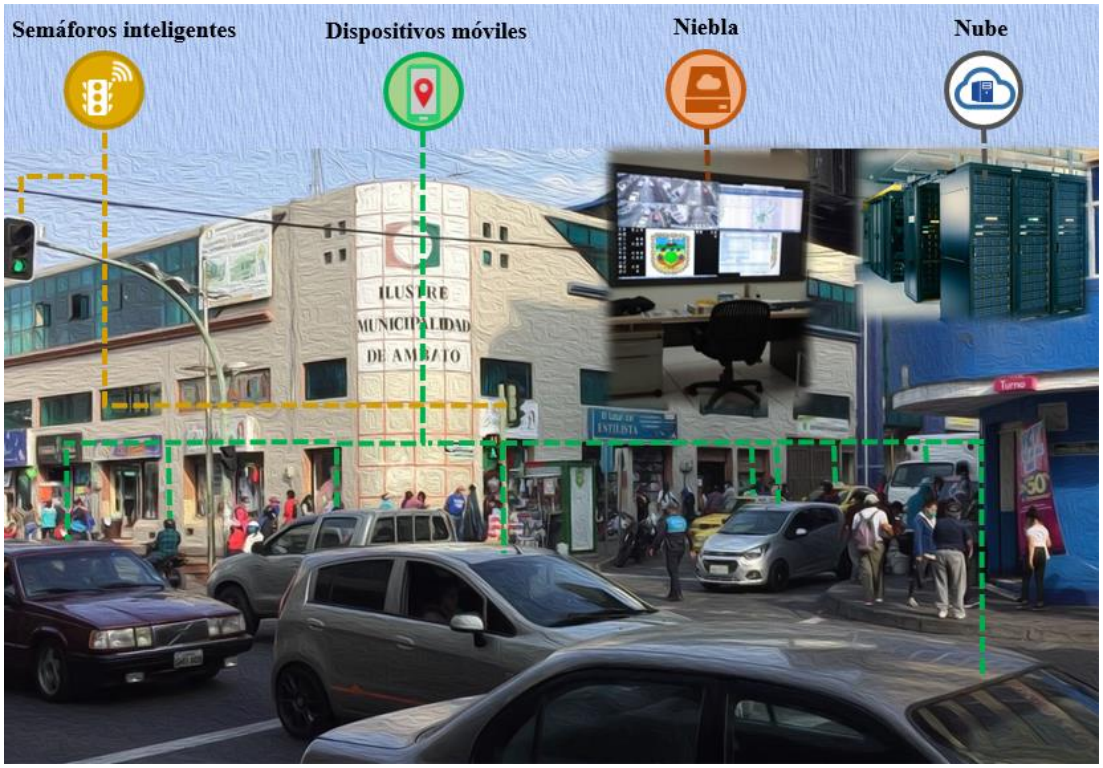


Figura 10: Representación gráfica de los componentes de la arquitectura fog computing en la intersección ubicada en la Av. Cevallos y Calle Eugenio Espejo.

Elaborado por: El investigador.

3.7 Selección del hardware y software para la implementación del sistema

3.7.1 Software para el desarrollo de la aplicación móvil

Para el desarrollo de la aplicación móvil la cual permite obtener la ubicación del vehículo en tiempo real y comunicarse con el servidor en la niebla, existen actualmente diferentes alternativas de software para la programación de la aplicación como son Android Studio, Xamarin y Eclipse. En la Tabla 3 se compara las características que tienen dichos entornos de programación.

Tabla 3: Comparación de los diferentes entornos de desarrollo de aplicaciones móviles [43], [44], [45].

Entornos de programación	Android Studio	Xamarin	Eclipse
Desarrollador	Software de código abierto creado por Google	Software de código abierto desarrollado por Microsoft	Software de código abierto desarrollado por la Fundación Eclipse
Lenguajes de Programación	Admite los lenguajes de programación como Java y Kotlin	Lenguaje de programación C#	Java, C, C++, JSP, perl, Python, Ruby y PHP
Plataforma	Android	iOS, Android y Windows	Android
Requerimientos del sistema	- Procesador de 1.2 GHz - Memória RAM recomendada de 4 GB - Sistema operativo Windows 7 ó posterior - Almacenamiento de 1.2 GB de espacio disponible	- Procesador de 1.6 GHz - Memória RAM de 8 GB como mínimo - Sistema operativo Windows 7 ó posterior - Almacenamiento de al menos 80GB de espacio disponible	- Procesador de 1.2 GHz - Memória RAM de 4 GB como mínimo - Sistema operativo Windows 7 ó posterior - Almacenamiento de 1 GB de espacio disponible

Según las características que tienen los diferentes entornos de programación utilizados actualmente por los desarrollados de aplicaciones móviles se ha elegido Android Studio principalmente por tener requisitos de sistema alcanzables para el correcto funcionamiento del software, además que admite el lenguaje de programación Java estudiado anteriormente en los módulos de la carrera y por ser una herramienta de código abierto para dispositivos Android, según datos de la consultora de tecnología IDC, los dispositivos Android representaron más del 86% de los dispositivos móviles distribuidos en el 2019, y los que tienen sistema operativo iOS de Apple casi el 14% restante [46], razón importante para elegir dicha herramienta para el desarrollo de la aplicación móvil del sistema.

3.7.2 Hardware para el servidor en la niebla

El fog computing requiere de una capa de gateway intermedia para realizar las funciones de computación en la niebla, es decir una inversión significativa en hardware para la implementación del sistema a través de un ordenador de placa reducida el cual se encarga de la conexión de los dispositivos IoT y la nube para el almacenamiento y procesamiento de datos. Existen varios modelos de ordenadores de placas reducidas en el mercado que tienen las características detalladas en la Tabla 4.

Tabla 4: Comparación de los modelos de ordenador de placa reducida [47], [48].

Ordenador de placa reducida	**Raspberry Pi 3 B** 	**Rock64** 	**Banana Pi M64** 
Procesador	Broadcom BCM2837	Rockchip RK3328	Allwinner A64
CPU	Quad core ARM Cortex-A53 1,2 GHz 64 bits	Quad core ARM Cortex-A53 1,2 GHz 64 bits	Quad core ARM Cortex-A53 1,2 GHz 64 bits
RAM	1 GB	1 GB	1 GB
Conectividad	Wireless LAN 802.11.b/g/n/ac Bluetooth 4.1 Ethernet 10/100 Mbit/s	Gigabit Ethernet 100 Mbit/s	Wireless LAN 802.11 b/g/n Bluetooth 4.0 Ethernet 10/100/1000 Mbit/s
Tipo de memoria SD	Micro SD	Micro SD	Micro SD
Alimentación	5V ,2A DC Micro USB/GPIO	5V, 3A DC Barrel Power	5V, 2A DC Micro USB
Costo	$80	$50	$70

Para la implementación del nodo fog o servidor en la niebla se eligió el ordenador de placa reducida Raspberry Pi modelo 3 B ya que cuenta con un procesador de cuatro núcleos a 1,2 GHz de 64 bits y 1 GB de memoria RAM, ideal para el procesamiento de datos en tiempo real, tiene una memoria interna micro SD expandible para el almacenamiento de datos, además permite la conectividad inalámbrica a través de Wifi óptima para comunicación del sistema. Utiliza sistema operativo de software libre y tiene un bajo costo, siendo una herramienta ideal para este tipo de proyectos.

3.7.3 Software para el servidor en la Nube

La nube consiste en el punto final para el almacenamiento de los datos y visualización de los históricos generados por el sistema de control de tráfico vehicular, existen en la actualidad tres empresas conocidas que dan servicios en la nube como son: Amazon Web Service, Google Cloud Platform y Microsoft Azure, cada una con sus respectivas especificaciones detalladas en la Tabla 5.

Tabla 5: Comparación entre las empresas de servicios en la nube [49], [50].

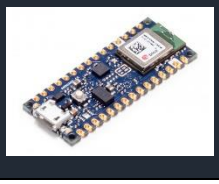
Empresas	Amazon Web Service	Google Cloud Platform	Microsoft Azure
Características	-Posición dominante en el mercado -Oferta amplia y madura	- Compromiso con el código abierto y la portabilidad	- Segundo proveedor más grande - Integración con herramientas y software de Microsoft
Servicios	- Nube de computación elástica o EC2 -Contenedores de aplicaciones	- Computo de ingeniería - Enfoque en Kubernetes	- Máquinas virtuales - Arquitectura de microservicios
Geografía	Ubicados en 206 países en América del Norte, América del Sur, Europa, Asia, Australia y África.	Consta de 200 países ubicados en América del Norte, América del Sur, Europa, Asia y Australia.	Ubicados en 140 países en América del Norte, América del Sur, Europa, Asia, Australia y África
Precios	- Plan para desarrollador de $29,00 mensuales - Prueba gratuita limitada.	- Plan para desarrollador de $22 mensuales - Prueba gratuita limitada.	- Plan para desarrollador $0,075/hora - Prueba gratuita limitada.

Según las especificaciones que tienen cada empresa para brindar servicios en la nube se eligió la nube de Amazon Web Service (AWS) ya que es la empresa que domina el mercado y tiene mayor alcance geográfico para minimizar latencias, ofrece el servicio de “nube de computación elástica o EC2” que es un servicio web que proporciona capacidad de informática y una gran variedad de instancias compatibles con Linux. Además AWS también ofrece un plan gratuito para EC2 limitado durante un máximo de doce meses, tiempo adecuado para el desarrollo del sistema.

3.7.4 Microcontrolador para el semáforo inteligente

Para la implementación del semáforo inteligente el cual requiere la conexión inalámbrica mediante Wifi con el servidor en la niebla para recibir datos correspondientes al cambio en los intervalos de tiempo de las luces de semaforización, se analiza las características que tiene los diferente modelos de microcontroladores que existen en el mercado como se ve en la Tabla 6, mediante el cual permite convertir al semáforo en un dispositivo IoT y de esta manera comunicarse con el servidor de niebla para su funcionamiento.

Tabla 6: Comparación de los modelos de microcontroladores [51], [52].

Modelos de microcontroladores	**NodeMCU ESP8266** 	**Arduino Nano 33** 	**Teensy 3.2** 
CPU	Xtensa LX106 de 32 bits	SAMD21 Cortex de 32 bits	ARM Cortex-M4 de 32 bits
RAM	96 KB	32 KB	64 KB
Voltaje de alimentación	2.3 – 3.6 Voltios	2.3 – 3.6 Voltios	2.3 – 3.6 Voltios
Voltaje de Operación	3.3 Voltios	3.3 Voltios	3.3 Voltios
Entradas y salidas digitales	32	14	34
Entradas analógicas	1	8	21
Memoria flash	1 MB	256 KB	128 KB
Estándar de comunicación	802.11 b/g/n	802.11 b/g/n Bluetooth	-
Dimensiones	1,8 cm x 2,5 cm	4,5 cm x 1,8 cm	3,5 cm x 1,8 cm
Costo	$15	$18	$25

Para la implementación del semáforo inteligente el cual se debe conectar al servidor en la niebla de manera inalámbrica, se eligió el microcontrolador NodeMCU modelo ESP8266 ya que posee un número de salidas digitales útiles para el encendido y apagado de las luces del semáforo, además trabaja con el estándar inalámbrico 802.11 o Wifi, tiene un costo de asequible y se puede programar mediante software libre y sus dimensiones son bastante pequeñas, lo que permite que se utilice en la implementación de prototipos en proyectos IoT.

3.8 Esquema general del sistema basado en la arquitectura fog computing

En la Figura 11, se detalla gráficamente las tres capas que tiene la arquitectura fog computing como son la capa borde, niebla y nube empleados para el control de tráfico vehicular, se especifican los respectivos elementos seleccionados anteriormente tanto en hardware como en software para la implementación del sistema, y las tecnologías de comunicación adecuadas para el correcto funcionamiento a través del protocolo de comunicación para dispositivos IoT denominado MQTT.

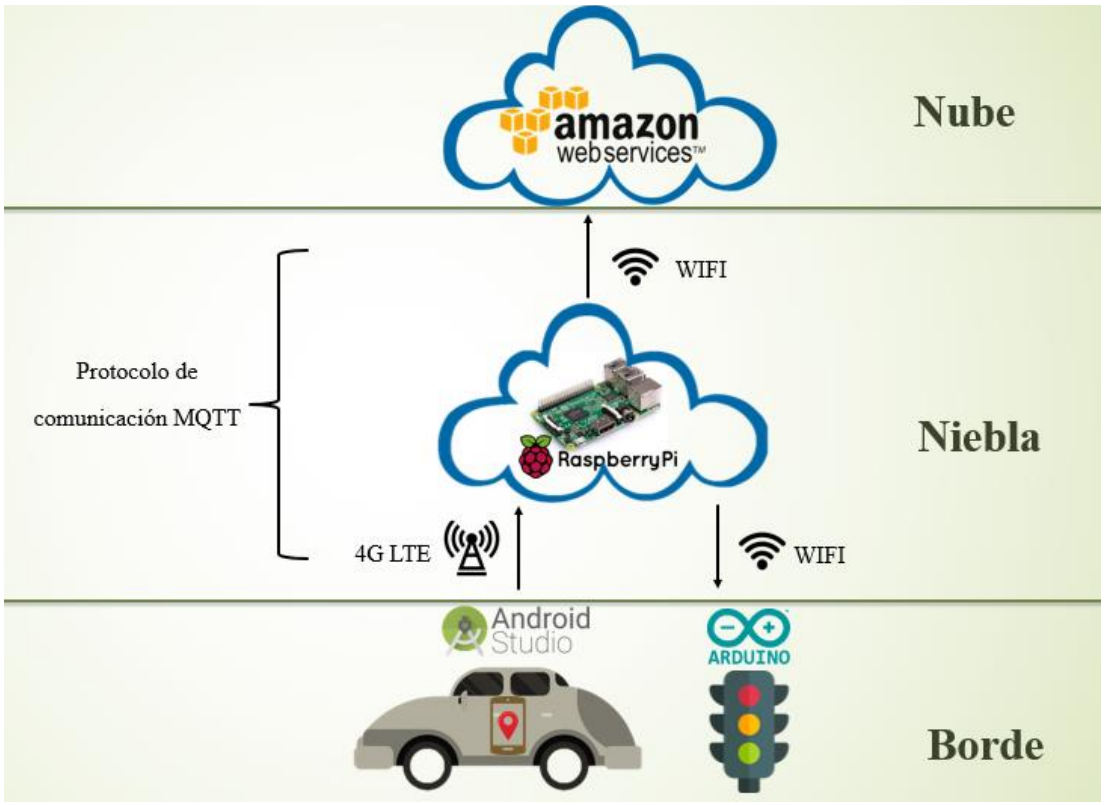


Figura 11: Esquema general del sistema basado en la arquitectura fog computing.

Elaborado por: El investigador.

3.9 Implementación del servidor en la niebla

El servidor en la niebla implementado mediante la placa Raspberry Pi actúa como un nodo fog o servidor local, el cual comunica los dispositivos IoT que se encuentran en el borde y el servidor en la nube de aws, almacena los datos enviados de la ubicación

de cada vehículo, procesa dicha información para actuar en el comportamiento de los intervalos de tiempo de los semáforos como se detalla en la Figura 12, y a través de una interfaz web se monitorea en tiempo real el estado del sistema, la ubicación de los vehículos y el cambio de los intervalos de tiempo de las fases de semaforización.

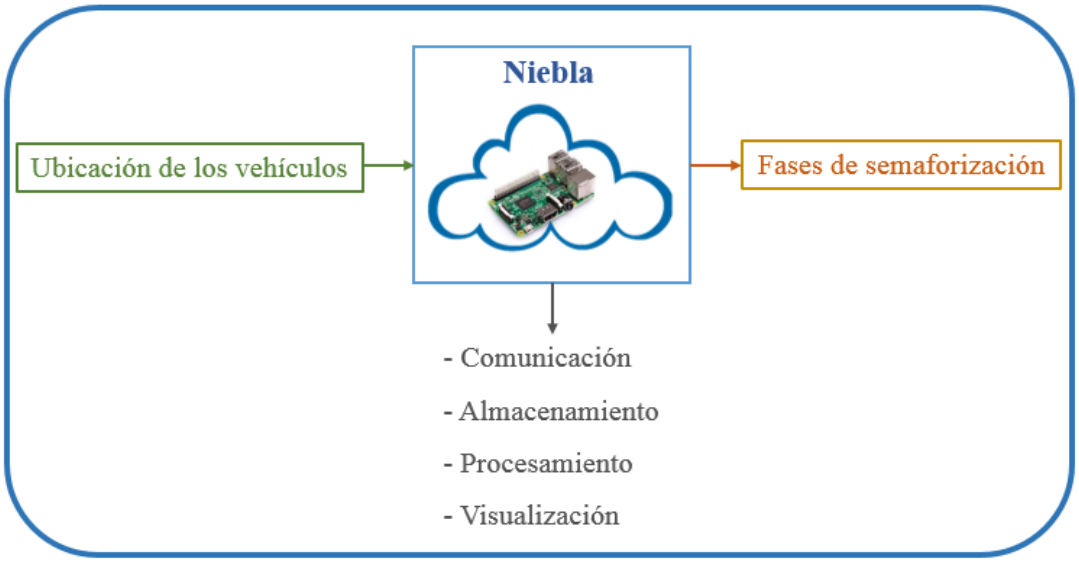


Figura 12: Diagrama del proceso de funcionamiento del servidor en la niebla.

Elaborado por: El investigador.

3.9.1 Instalación del sistema operativo en la Raspberry Pi

Para la implementación del servidor en la niebla se eligió el sistema operativo oficial de Raspberry Pi denominado Raspbian que es una distribución del sistema operativo GNU/Linux basado en Debian en la versión Stretch por su mayor estabilidad en comparación a otras versiones. Mediante el software Raspberry Pi Imager descargado de la página oficial, se instaló el sistema operativo en una memoria SD de 32 GB como muestra la Figura 13.

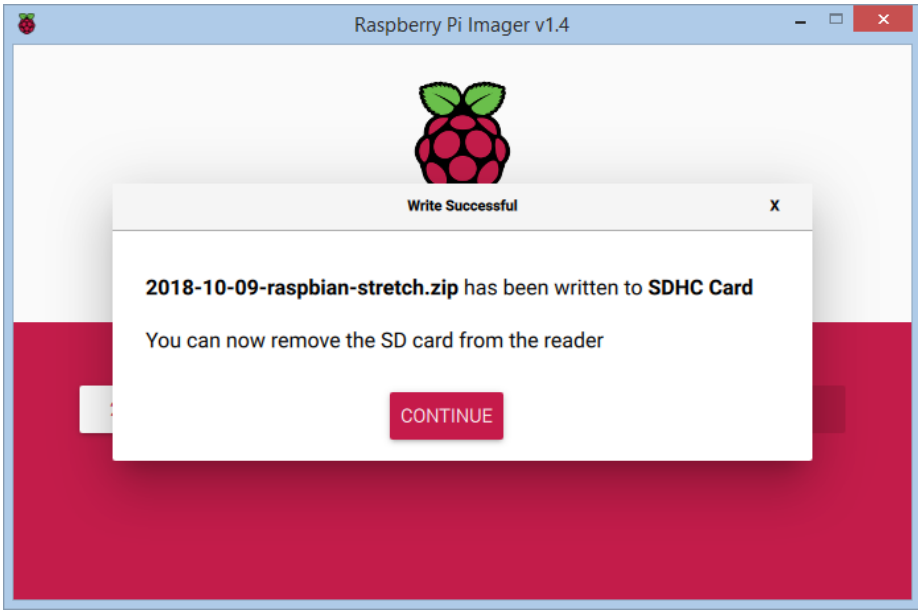


Figura 13: Instalación del sistema operativo Raspbian usando el software Raspberry Pi Imager**.**

Elaborado por: El investigador.

3.9.2 Configuración de la Raspberry Pi

Con el sistema operativo ejecutando en la Raspberry Pi, se realiza las configuraciones iniciales necesarias para habilitar la interfaz de comunicación: SSH para el acceso remoto a la terminal de comandos y la interfaz VNC para observar las acciones de la Raspberry Pi remotamente a través de un ordenador.

Además se establece una dirección ip estática en la placa Raspberry Pi para la conexión mediante Wifi especificada por la interfaz wlan0. La dirección ip estática asignada es la dirección 192.168.1.20 con la puerta de enlace predeterminada por la dirección ip 192.168.1.1.

3.9.3 Configuración del usuario y contraseña de la Raspberry Pi

Para la configuración del usuario y contraseña se ingresa a la Raspberry Pi utilizando la herramienta PuTTY que es un cliente SSH que permite conectarse y ejecutar

comandos en la terminal de la Raspberry Pi siempre y cuando estén habilitadas dichas interfaces como se realizó anteriormente en la configuración inicial. En la Figura 14, se observa la conexión a la dirección ip 192.168.1.20, mediante el puerto SSH que opera en el puerto TCP 22 de forma predeterminada.

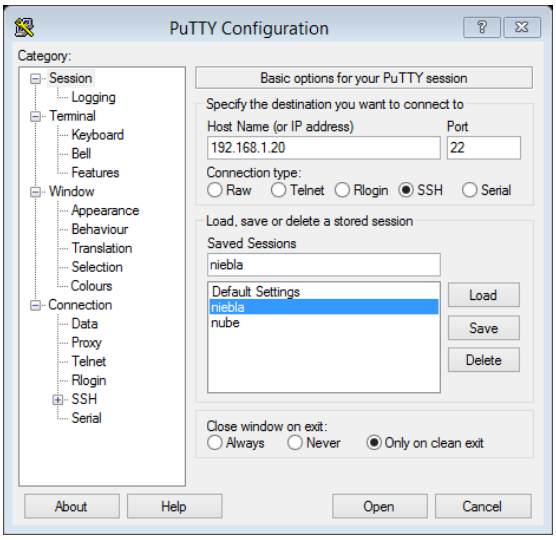


Figura 14: Configuración de PuTTY para acceder a la terminal de la Raspberry Pi.

Elaborado por: El investigador.

El usuario y contraseña configurada por defecto en Raspbian es usuario: pi y su contraseña: raspberry, es necesario cambiar esta configuración para mejorar la seguridad de la Raspberry Pi a través de los siguientes pasos:

1. Iniciar sesión en la raíz como super usuario con el comando:
$sudo su

2. Ingresar al directorio "etc" encargado de almacenar los archivos de configuración:
#cd /etc

3. Crear el usuario niebla utilizando el siguiente comando:
#adduser niebla

4. Luego se añade el usuario de niebla al grupo "sudo" que tienen permisos de super usuario, editando el archivo de configuración /etc/group.

5. Finalmente se deshabilita el usuario de pi configurado por defecto para dar mayor seguridad al dispositivo, para esto se edita el archivo /etc/passwd, en la cuenta de pi se coloca un signo de "*" para deshabilitarla.

3.9.4 Configuración del protocolo de comunicación MQTT y Websocket para el servidor en la niebla

Para la comunicación del servidor en la niebla a través del protocolo Mqtt se instala el broker mosquitto y mosquitto clientes mediante las siguientes líneas de comando:

#apt-get install mosquitto

#apt-get install mosquitto-clients

Luego se crear un archivo de configuración para el protocolo de comunicación websockets en el la dirección /etc/mosquitto/conf.d, agregando las siguientes líneas de código para definir los puertos de comunicación a través del protocolo Mqtt y Websocket:

listener 1883, protocol mqtt

listener 9001, protocol websockets

3.9.5 Configuración de la conexión del servidor en la niebla a Internet

Para conectar el servidor en la niebla con los dispositivos móviles que se encuentran en los vehículos ubicados fuera de nuestra red de área local inalámbrica o WLAN, es necesario utilizar técnicas para el re direccionamiento del tráfico de datos, esto se puede realizar atreves de métodos de reenvío de puertos que normalmente presentan vulnerabilidades o mediante servicios de acceso remoto seguro como es Remote.it que elimina la exposición a ataques de puertos abiertos.

Para configurar el servicio de Remote.it en el servidor de niebla se realiza los siguientes pasos detallados a continuación:

1. Crear una cuenta en el sitio oficial para acceder a la consola de remote.it.

2. Instalar el servicio de remote.it en el servidor de niebla mediante los siguientes comandos:

   #apt install connectd

   #connectd_installer

3. Registrar el servidor en la niebla en remote.it, utilizando el email y la contraseña ingresados en el paso 1 para poder conectar el servidor de niebla a internet.

4. Añadir las reglas de control de tráfico de datos para el servidor en la niebla, definiendo los protocolos de comunicación en este caso SSH para el acceso remoto mediante el puerto 22, HTTP para el servicio web por el puerto 80, los protocolos de comunicaciones MQTT para mosquitto mediante el puerto 1883 y Websocket definido por el puerto 9001, como se ve en la Figura 15.

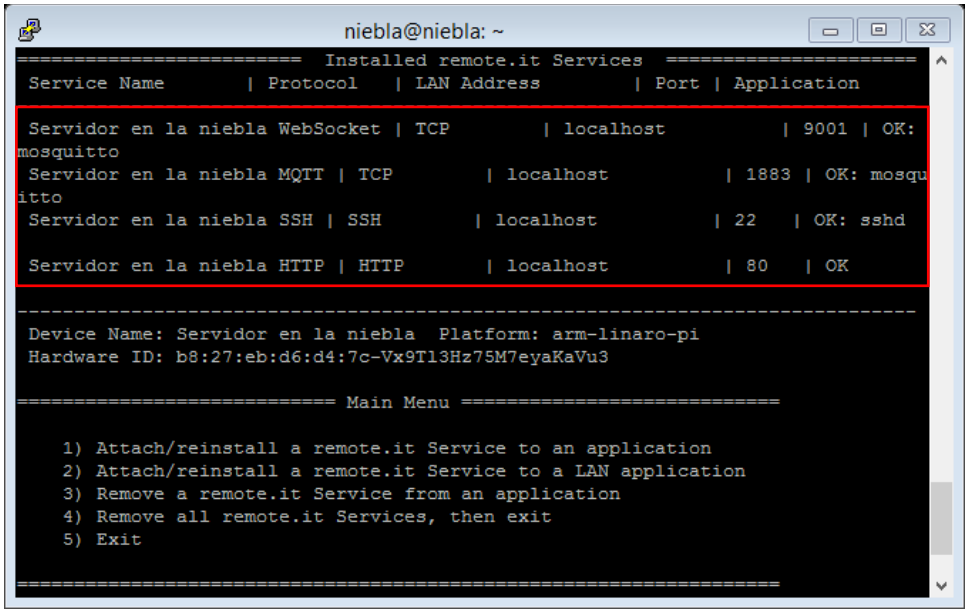


Figura 15: Reglas de control de tráfico de datos para el servidor en la niebla.

Elaborado por: El investigador.

5. Se comprueba en el panel de control de remote.it que el servidor en la niebla está conectado correctamente a internet, la dirección ip interna definida anteriormente es la 192.168.1.20 y la dirección ip externa es la 190.152.130.184 como se ve en la Figura 16.

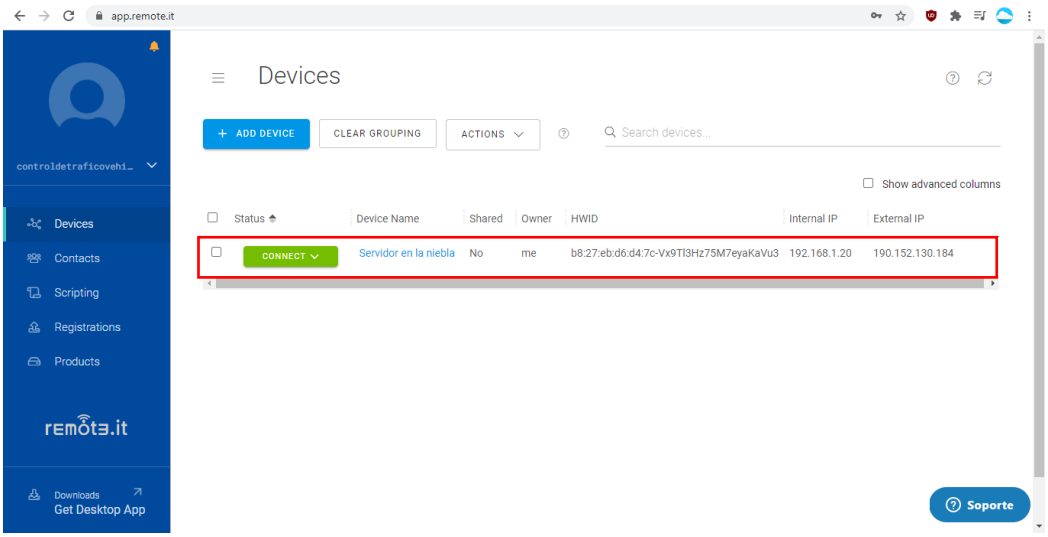


Figura 16: Detalles de la conexión del servidor en la nube la consola de Remote.it.

Elaborado por: El investigador.

Para la comunicación del servidor en la niebla con los dispositivos IoT mediante el protocolo MQTT se utiliza el proxy generado por el servicio de Remote.it "proxy19.rt3.io:39155" como se ve en la Figura 17:

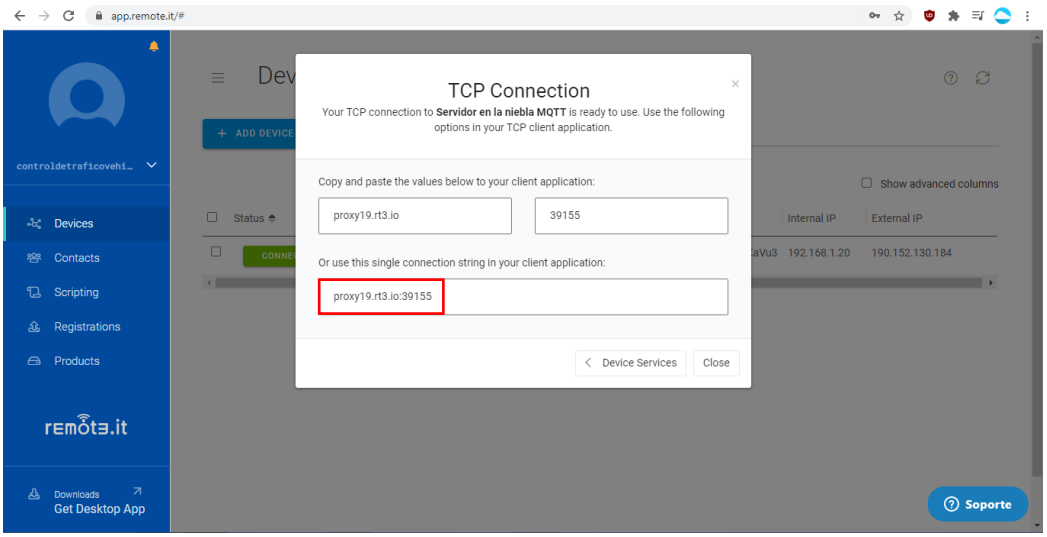


Figura 17: Proxy generado por el servicio de Remote.it para la comunicación MQTT.

Elaborado por: El investigador.

3.10 Implementación del servidor en la nube

El servidor en la nube implementado mediante el servicio de Amazon Web Services (AWS) para el sistema de control de tráfico vehicular actúa como un servidor remoto ubicado en América del Sur específicamente en Sao Paulo Brasil el cual se comunica con el servidor en la niebla o servidor local y almacena datos correspondientes a registros históricos del sistema de control de tráfico vehicular relacionados el conteo de vehículos y fases de semaforización comos se detalla en la Figura 18, y además los datos son visualizados a través de una interfaz web con la fecha y hora en el cual se produjo dicho registro.

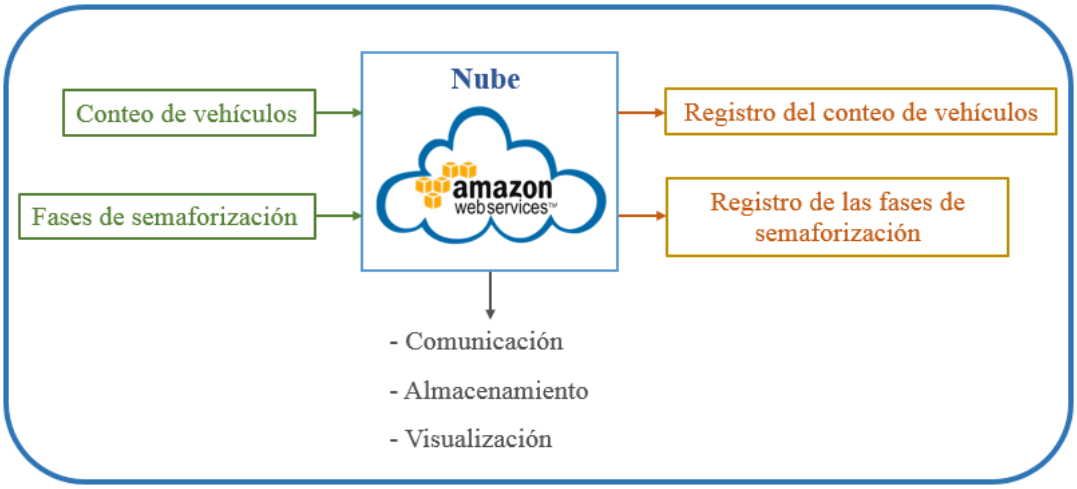


Figura 18: Diagrama del proceso de funcionamiento del servidor en la nube.
Elaborado por: El investigador.

3.10.1 Creación de la cuenta en AWS para el servidor en la nube

La creación de la cuenta en AWS para el servidor en la nube se realiza a través de la página oficial llenando el formulario de registro, la cuenta es gratuita e incluye 12 meses para desarrollar pruebas de funcionamiento, además se pedirá un número de cuenta de tarjeta de crédito o débito para verificar la identidad del usuario, se hará el cargo de USD 1 esta cantidad se muestra pendiente durante un periodo de 3 a 5 días hasta completar la verificación de la cuenta entonces el cargo se eliminará.

3.10.2 Configuración del servidor en la nube mediante Amazon Elastic Compute Cloud o EC2

Para configurar el servicio EC2 de AWS que permite alquilar computadores virtuales para implementar el servidor en la nube, se siguen los siguientes pasos:

1. Seleccionar el sistema operativo de la máquina virtual que corresponde al servidor en la nube en este caso será Ubuntu Server 16.04 de nivel gratuito.

2. Configurar la capacidad de almacenamiento del disco duro que tiene el servidor en la nube, en este caso se define la capacidad máxima que es 30 GB.

3. En la figura 16, se añade las reglas de control de tráfico de datos para el servidor en la nube, se define los protocolos de comunicación mediante TCP: SSH mediante el puerto 22, HTTP por el puerto 80, los protocolos de comunicaciones MQTT para mosquitto mediante el puerto 1883 y Websocket definido por el puerto 9001, como se ve en la Figura 19.

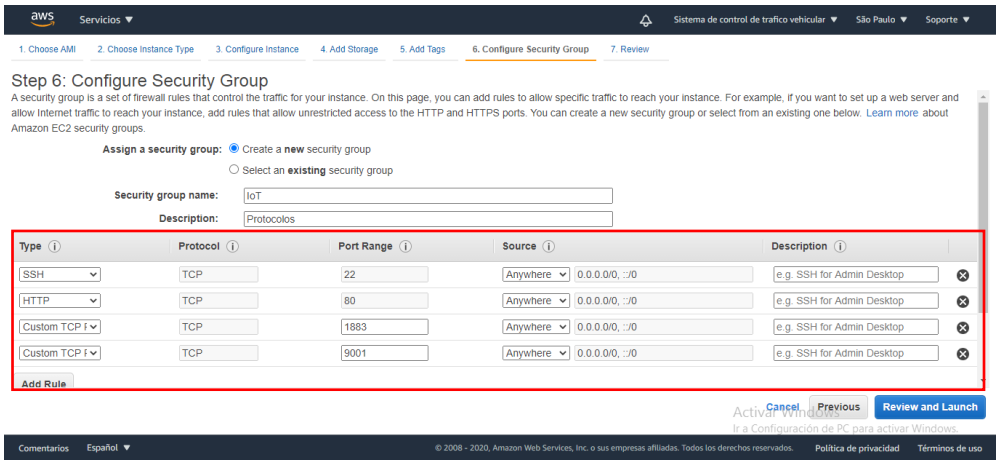


Figura 19: Reglas de control de tráfico de datos para el servidor en la nube.

Elaborado por: El investigador.

4. Generar una par de claves que consta de una clave pública que almacena AWS y un archivo de clave privada que se almacena en nuestro ordenador para conectarse

al servidor en la nube de forma segura. La clave generada para el servidor en la nube tiene el nombre de “Llave nube aws”.

5. En la Figura 20, se asigna al servidor en la nube una dirección ip estática privada 172.31.10.233 y una dirección ip estática pública 18.230.182.193, para su correcta ejecución.

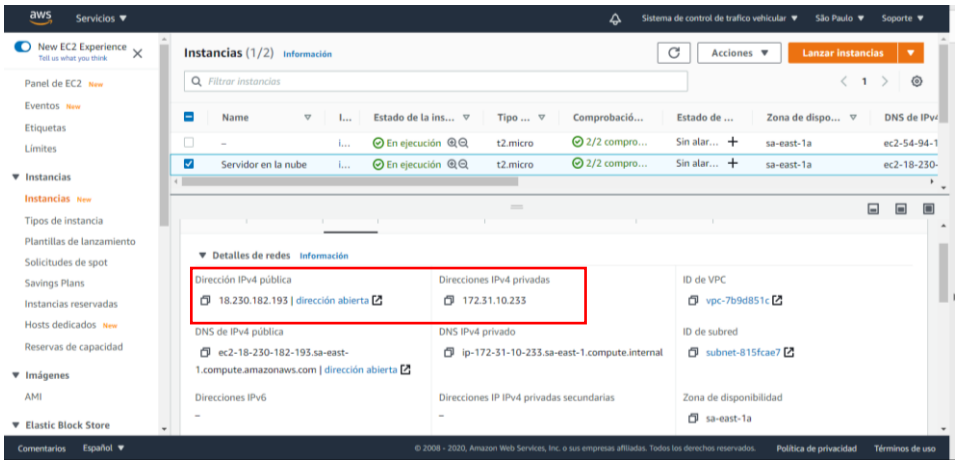


Figura 20: Dirección ip privada y pública del servidor en la nube.

Elaborado por: El investigador.

6. Finalmente se selecciona la ubicación de servidor en la nube, se elige el lugar más cercano a Ecuador con el fin de evitar latencias, la única opción que se encuentra en América del Sur es la ciudad São Paulo en Brasil.

3.10.3 Configuración de la conexión remota al servidor en la nube de aws mediante SSH

Para la conexión remota al servidor en la nube de aws utilizamos el par de claves generados en el paso 4 dedicado a la configuración del servidor en la nube mediante EC2 y la herramienta PuTTY. En la Figura 21, se observa la conexión a la dirección ip 18.230.182.193, mediante el puerto 22.

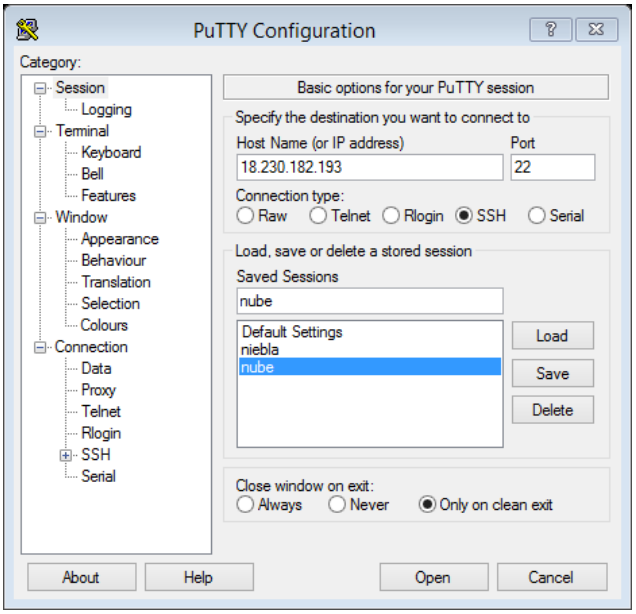


Figura 21: Configuración de PuTTY para acceder al servidor en la nube.

Elaborado por: El investigador.

3.10.4 Configuración del almacenamiento del servidor en la nube

Para el almacenamiento de datos en la nube acerca de los registros históricos del sistema de control de tráfico vehicular, se utiliza el gestor de base de datos MySQL mediante la implementación de un servidor LAMP que está compuesto por las iniciales de sus cuatro componentes: Linux, Apache, MySQL y PHP que combinados definen la infraestructura de un servicio web necesarios para el desarrollo del sistema, la instalación de un servidor LAMP ser realiza mediante los siguientes pasos:

1. Instalar Apache mediante el siguiente comando:
   #apt-get install apache2

2. Luego instalar MySQL y phpmyadmin mediante las siguientes líneas de código, en este paso se pedirá una contraseña para acceder a la base de datos:
   #apt-get install mysql-server mysql-client libmysqlclient-dev
   #apt-get install phpmyadmin

3. Posteriormente instalar php y sus librerías mediante el siguiente comando:

#apt-get install libapache2-mod-php php7.0-mbstring php-gettext php7.0-dev

4. Crear un acceso directo de phpmyadmin en Apache mediante la siguiente línea de código:
   #ln -s /usr/share/phpmyadmin /var/www/html

5. Reiniciar el servicio de Apache mediante:
   #service apache2 restart

6. Finalmente se ingresa a la interfaz de phpmyadmin, con la dirección ip del servidor en la niebla 192.168.1.20 en un navegador web para comprobar la correcta instalación de la base de datos MySQL.

3.10.5 Configuración del protocolo de comunicación Websocket para el servidor en la nube

Para la configuración del protocolo de comunicación Websocket de igual manera que en el servidor en la niebla se crea el archivo de configuración websockets.conf en la dirección /etc./mosquitto/conf.d, agregando las siguientes líneas de código para definir los puertos disponibles para la comunicación del servidor en la nube a través del protocolo Websocket y Mqtt:

listener 9001, protocol websockets

listener 1883, protocol mqtt

3.10.6 Integración de MQTT con MySQL a través de PHP para el servidor en la niebla

Para integrar el protocolo de comunicación MQTT con la base de datos MySQL mediante el lenguaje de programación PHP en el servidor de la nueva realizamos los siguientes pasos:

1. Instalar el paquete de mosquitto utilizando el siguiente comando:

#apt- get install libmosquitto-dev

2. Instalar la extensión PHP para mosquito mediante el comando que se encuentra a continuación, el paquete se instalara en su versión más reciente v0.4.0:
   #pecl install Mosquitto-alpha.

3. Finalmente de sebe agregar la "extension=mosquitto.so" al archivo de configuración de php.ini, para su correcto funcionamiento.

3.10.7 Creación de la base de datos MySQL para el servidor en la nube

Para la base de datos de MySQL en el cual se almacena los registros históricos del sistema se crea una tabla de datos denominada "Contenido" en la interfaz web de phpMyAdmin el cual tiene una tabla con 4 columnas para el Id de tipo entero, Tópico y Mensaje de tipo varchar con una longitud máxima de 30 caracteres y Recibido de tipo timestamp para obtener la fecha del registro como se ve en la Figura 22:

Figura 22: Estructura de la base de datos MySQL del servidor en la niebla.

Elaborado por: El investigador.

Para almacenar los datos en la tabla contenido se realiza un script denominado datos_nube.php el cual permite ejecutar el broker mosquitto para suscribir a los tópicos del sistema y enviar a la base de datos de MySQL mediante las siguientes líneas código:

- Especificar la base de datos para a conectar el broker mosquitto especificando el usuario root y contraseña servidoraws:

$db = new PDO('mysql:host=localhost;dbname=Datos;charset=utf8', 'root', 'servidoraws');

- Definir el protocolo de comunicación mqtt y la dirección ip del servidor en la nube:

  $mqtt->connect('18.230.182.193',1883);

- Suscripción a los tópicos del sistema:

  $mqtt->subscribe($topic, $qos);

- Insertar los datos recibidos en la tabla contenido:

  'INSERT INTO Contenido (Id, Topico, Mensaje, Recibido) VALUES (NULL, ?, ?, CURRENT_TIMESTAMP);'

3.11 Elaboración de la aplicación móvil con Android Studio

Para realizar la aplicación móvil utilizando el software de Android Studio que permita obtener datos correspondientes a la ubicación (latitud, longitud) e información del vehículo (placa) y luego enviar estos datos al servidor en la niebla para su procesamiento, se especifica a continuación la estructura de la aplicación móvil, características del proveedor de ubicación fusionado y del protocolo de comunicación Mqtt utilizado, además el diseño de la interfaz de usuario de la aplicación móvil.

3.11.1 Estructura de la aplicación móvil

La estructura de la aplicación móvil elaborada consta de seis archivos relacionados a entornos de programación, interfaces de usuario y archivos de configuración explicados a continuación:

1. **SplashActivity.java:** El SplashActivity.java es la actividad que se ejecuta al inicio mientras se cargan todos los elementos de la aplicación.
2. **LoginActivity.java:** El LoginActivity.java es el entorno de programación de la información correspondiente a la placa del vehículo.

3. **MainActivity.java:** El MainActivity.java es el entorno de programación de la aplicación móvil en el cual se define el proveedor de ubicación fusionado y el protocolo de comunicación Mqtt.
4. **activity_login.xml:** El activity_login.xml corresponde a la interfaz de la aplicación en donde se pide al usuario ingresar la placa del vehículo.
5. **activity_main.xml:** El activity_main.xml corresponde a la interfaz de usuario de la aplicación en donde se muestra datos del proveedor de ubicación fusionado y el funcionamiento de los semáforos.
6. **AndroidManifest.xml:** El AndroidManifest.xml es el archivo de configuración en el que se define los permisos que requiere la aplicación.
7. **Build.gradle**: El Build.gradle corresponde al archivo de configuración en el cual se especifica las librerías y dependencias que utiliza la aplicación.

3.11.2 Proveedor de ubicación fusionada para la aplicación móvil

El proveedor de ubicación fusionada es una Api estándar de Android desarrollado por Google que permite obtener la ubicación del usuario a través del sistema de posicionamiento global GPS, ubicaciones de torres celulares y ubicaciones Wifi conocidas. En este caso se utiliza la señal GPS para obtener la ubicación del vehículo en función de sus coordenadas geográficas latitud y longitud por su mayor precisión.

A continuación se especifican los requerimientos para utilizar el proveedor de ubicación fusionada en la aplicación móvil:

- Incluir en el archivo Build.gradle los servicios de Google Play como una dependencia:
  implementation 'com.google.android.gms:play-services-location:17.0.0'

- Otorgar permisos en el archivo AndroidManifest.xml para que la aplicación rastree la ubicación del móvil de la siguiente forma:
  <uses-permission
  android:name="android.permission.ACCESS_FINE_LOCATION"/>

- Definir la clase que corresponde al proveedor de ubicación fusionada:

FusedLocationProviderClient fusedLocationProviderClient;

- Especificar los parámetros de calidad de servicio para que la aplicación obtenga la ubicación en tiempo real.

  Tiempo que se realiza la verificación de ubicación predeterminada, cada 1 segundos:
  locationRequest.setInterval(1000);
  Tiempo que se realiza la actualización de la ubicación, cada 5 segundos:
  locationRequest.setFastestInterval(5000);
  Prioridad alta para obtener la ubicación mediante el sensor Gps del dispositivo móvil:
  locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

3.11.3 Protocolo de comunicación Mqtt para la aplicación móvil

El protocolo de comunicación Mqtt utilizado en la aplicación móvil permite encapsular la conexión dentro de un servicio de Android y poder enviar, recibir mensajes de manera confiable con el servidor en la niebla, mediante la librería "Cliente Paho Java Mqtt" desarrollada por el proyecto Paho de Eclipse.

A continuación se especifican los requerimientos para utilizar el protocolo de comunicación Mqtt en la aplicación:

- Agregar el servicio de Paho Android en el archivo de configuración Build.gradle:

  implementation('org.eclipse.paho:org.eclipse.paho.android.service:1.0.2')

- Añadir el servicio de Android Paho en el archivo de configuración AndroidManifest.xml para acceder a los servicios de comunicación de Mqtt en la aplicación y definir los permisos para que la aplicación móvil pueda conectarse utilizando el protocolo de comunicación Mqtt con el servidor en la niebla.

  Servicio de Android Paho para utilizar el protocolo Mqtt:
  <service android:name="org.eclipse.paho.android.service.MqttService">

</service>

Permiso para que la aplicación trabaje también en segundo plano:

<uses-permission android:name="android.permission.WAKE_LOCK" />

Permiso para que la aplicación utilice el servicio de internet:

<uses-permission android:name="android.permission.INTERNET" />

Permiso para que la aplicación pueda verificar el estado de la conexión de red:

<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />

Permiso para obtener la identificación de dispositivo:

<uses-permission android:name="android.permission.READ_PHONE_STATE" />

- Crear el cliente Mqtt para la aplicación y especificar la conexión con el servidor en la niebla mediante:
  Client = new MqttAndroidClient(this.getApplicationContext(),
  "tcp:// proxy19.rt3.io:39155", clientId);
- Publicar los mensajes a través del método client.publish con el tópico, mensaje y calidad de servicio qos, para enviar los datos de la placa, latitud y longitud del vehículo al servidor en la niebla.

  Publicación del tópico que tiene el dato correspondiente a la placa del vehículo con qos igual a cero:
  client.publish("Placa",Placa.getBytes(),0);
  Publicación del tópico que tiene el dato de la latitud del vehículo con qos igual a cero:
  client.publish("Latitud",String.valueOf(location.getLatitude()).getBytes(),0);
  Publicación del tópico que tiene el dato de la longitud del vehículo con qos igual a cero:
  client.publish("Longitud",String.valueOf(location.getLongitude()).getBytes(),0);

- Suscribir los mensajes a través del método client.subscribe con el tópico y la calidad de servicio qos, para generar los intervalos de semaforización en la aplicación móvil:

  Suscripción al tópico Semaforo/A con qos igual a cero:
  client.subscribe("Semaforo/A",0);
  Suscripción al tópico SemaforoB con qos igual a cero:
  client.subscribe("Semaforo/B",0);

  Los intervalos de semaforización en la aplicación móvil se generan mediante los mensajes recibidos al suscribirse en los tópicos: "Semaforo/A" y "Semaforo/B", siguiendo las siguientes condiciones para generar los intervalos de semaforización:

  Condición para generar el intervalo de semaforización uno:
  if(mensajeRecibido.equals("Uno")){ FaseUno(); }
  Condición para generar el intervalo de semaforización dos:
  if(mensajeRecibido.equals("Dos")){ FaseDos();}
  Condición para generar el intervalo semaforización tres:
  if(mensajeRecibido.equals("Tres")){FaseTres();}

3.11.4 Diseño de la interfaz de la aplicación móvil

El diseño de la interfaz de la aplicación móvil costa de tres pantallas, las cuales permiten al usuario interactuar con la aplicación, la primera corresponde a la pantalla de bienvenida, luego aparece la pantalla de inicio y finalmente se muestra la interfaz del sistema de control de tráfico vehicular, como se detalla a continuación:

1. La primera pantalla es el "splash screen" o pantalla de bienvenida definido en la actividad SplashActivity.java mediante el archivo xml background_splash el cual sirve para visualizar un elemento gráfico mientras se carga la aplicación, el elemento grafico es el ImageView del logotipo de la Universidad Técnica de Ambato, como se ve en la Figura 23:

Figura 23: Pantalla de bienvenida de la aplicación móvil.

Elaborado por: El investigador.

2. La segunda pantalla definido por el archivo activity_login.xml corresponde a la ventana de inicio el cual tiene dos TextView para el título "Sistema de control de tráfico vehicular" y para el título "Ingreso del número de la placa", un ImageView para el logo de la Facultad de Ingeniería en Sistemas Electrónica e Industrial, además un EditText para ingresar la placa del vehículo por parte del usuario y un Button para iniciar con el sistema como se muestra en la Figura 24:

Figura 24: Pantalla de inicio de la aplicación móvil.

Elaborado por: El investigador.

3. La tercera pantalla definida por el archivo activity_login.xml al inicio muestra una ventana de notificación que pide el permiso necesario para que la aplicación acceda a la ubicación del dispositivo, una vez otorgado el permiso se puede visualizar la interfaz de usuario del sistema de control de tráfico vehicular, como se muestra en la Figura 25, la interfaz está divida en tres secciones, en la primera sección se tiene un TextView con la placa ingresada anteriormente por parte del usuario y un ImageView que contiene un logotipo de un vehículo en general. En la segunda sección se encuentra los datos que corresponde a la ubicación de vehículo específicamente el tipo de sensor utilizado, este se encuentra configurado inicialmente para el uso del sensor Gps, estado de la ubicación ya sea obtenida o no y la latitud, longitud del vehículo, definidos por los TextView correspondientes. En la tercera parte se observa dos ImageView para los semáforos A y B en la Av. Cevallos y Calle Eugenio Espejo respectivamente que permite al usuario visualizar el cambio de los intervalos de semaforización del sistema de control de tráfico vehicular.

Figura 25: Pantalla de la interfaz de usuario del sistema de Control de tráfico vehicular de la aplicación móvil.
Elaborado por: El investigador.

3.12 Diseño del algoritmo para el control de tráfico vehicular en el servidor de niebla

Para el diseño del algoritmo que permita el procesamiento de datos en el servidor de niebla o nodo fog del flujo vehicular con la finalidad de cambiar los intervalos de tiempo de semaforización y así controlar de manera inteligente el tráfico vehicular, se han tomado en cuenta los siguientes aspectos detallados a continuación:

3.12.1 Análisis de los intervalos de semaforización actual y conteo del número de vehículos que circulan en la intersección ubicada en la Avenida Cevallos y Calle Eugenio Espejo

El semáforo A ubicado en la Avenida Cevallos regula la circulación de vehículos en dos sentidos Este y Oste y el semáforo B ubicado en la Calle Eugenio Espejo regula la circulación de vehículos en el sentido Norte como se ve en la Figura 26, funcionan de manera sincronizada con un ciclo de 90 segundos y tienen los siguientes intervalos

de tiempo de semaforización: luz roja 48 segundos, luz verde 39 segundos y luz amarilla 3 segundos, además se registró el número de vehículos que circulan en el intervalo de tiempo en verde que corresponde a los 39 segundos, se tiene que el número de vehículos contados es de 12 a 15 frecuentemente en horas pico, datos que se utilizan a continuación para el diseño del algoritmo adecuado.

Figura 26: Semáforo A ubicado en la Av. Cevallos y semáforo B ubicado en la Calle Eugenio Espejo.

Elaborado por: El investigador.

3.12.2 Conteo de vehículos en el servidor de niebla mediante el uso de geovallas

Para el conteo de vehículos en el servidor de niebla se define tres geovallas A, B y C, que son perímetros virtuales de un área geográfica con la finalidad de determinar si un vehículo se encuentra ubicado en la Avenida Cevallos en cualquiera de sus sentidos de circulación o en la Calle Eugenio Espejo y así realizar el conteo del volumen de tráfico en las vías mediante contadores A, B y C respectivamente a cada geovalla como se ve en la Tabla 7:

Tabla 7: Características de las geovallas A, B y C.

Geovalla	Latitud	Longitud	Función
A	-1.2401836	-78.6253437	Contar vehículos en la Av. Cevallos en el sentido de circulación Oeste
B	-1.240779	-78.625994	Contar vehículos en la Av. Cevallos en el sentido de circulación Este
C	-1.2409819	-78.6252571	Contar vehículos en la Calle Eugenio Espejo.

Elaborado por: El investigador.

Para la aplicación de las geovallas en el servidor de niebla que permite el conteo de los vehículos se siguen los siguientes pasos:

- Obtener la poción del vehículo en el servidor de niebla, enviando previamente por la aplicación móvil.
- Calcular la distancia que existe entre la posición del vehículo y la geovalla.
- Calcular el radio de la geovalla.
- Comprobar si la distancia calculada es menor o igual que el radio de la geovalla para determinar la presencia del vehículo en la geovalla y finalmente realizar el conteo del vehiculó.

3.12.3 Se establece las fases de semaforización adecuados para el control de tráfico vehicular:

Para determinar las fases de semaforización que respondan de manera eficiente al volumen de tráfico en las vías se tomó en cuenta las tres posibles alternativas que tiene los vehículos para circular por la intersección entre en la Av. Cevallos y Calle Eugenio Espejo las cuales son: por la Geovalla A, B y C definidas anteriormente que corresponden a las 3 variables para las $2^3 = 8$ posibilidades y mediante el número de vehículos contados en este caso si es mayor, igual a 12 o menor a 12 vehículos que

corresponden al dato obtenido en el paso 1, se realiza una tabla de verdad como se muestra a continuación en la Tabla 8:

Tabla 8: Tabla de verdad para determinar las fases de semaforización.

	Geovalla A	Geovalla B	Geovalla C	Fase de semaforización
Posibilidad 1	$A \geq 12$	$B \geq 12$	$C \geq 12$	Uno
Posibilidad 2	$A \geq 12$	$B \geq 12$	$C < 12$	Dos
Posibilidad 3	$A \geq 12$	$B < 12$	$C \geq 12$	Uno
Posibilidad 4	$A \geq 12$	$B < 12$	$C < 12$	Dos
Posibilidad 5	$A < 12$	$B \geq 12$	$C \geq 12$	Uno
Posibilidad 6	$A < 12$	$B \geq 12$	$C < 12$	Dos
Posibilidad 7	$A < 12$	$B < 12$	$C \geq 12$	Tres
Posibilidad 8	$A < 12$	$B < 12$	$C < 12$	Uno

Elaborado por: El investigador.

Para determinar el número de fases de semaforización que permitan cumplir con las condiciones dadas se considera lo siguiente:

- La fase “Uno” se cumple cuando existe una circulación regular en cualquier sentido ya sea por la geovalla A, B y C.
- La fase “Dos” se cumple cuando existe una circulación mayor en la Geovalla A y B que corresponde a la Av. Cevallos, permitiendo descongestionar esta avenida.
- La fase “Tres” se cumple cuando existe una circulación mayor en la Geovalla C que corresponde a la Calle Eugenio Espejo, permitiendo descongestionar esta calle.

3.12.4 Se establece el tiempo en los intervalos de semaforización de cada fase para el control de tráfico vehicular:

Para definir los intervalos de tiempo de semaforización que permitan circulación vehicular de manera adecuada en cada fase de semaforización se han tomado en cuenta los siguientes aspectos:

- **Fase de semaforización uno:**

La fase de semaforización uno permite la circulación vehicular de forma regular en cualquier sentido es por ello que basa su funcionamiento en el sistema actual de semaforización el cual tiene un ciclo de semaforización igual 90 segundos, un intervalo de tiempo en luz roja similar al intervalo de tiempo en luz verde y un intervalo de luz en amarillo modificado el cual cumple las siguientes condiciones:

Según estudios realizados internacionalmente se recomienda que el tiempo de semaforización para la luz amarilla varié según la velocidad permitida en cada calle y así evitar colisiones que se presentan por el rápido cambio de la luz, a velocidad de 40 km/h se recomienda un intervalo de tiempo de 3,0 segundos, a velocidad de 50 km/h un tiempo de 3,5 segundos, a velocidad de55 km/h un tiempo de 4,0 segundos y a velocidad de 65 km/h un tiempo de 4,5 segundos en el intervalo de luz amarilla [53].

Según la Dirección de Gestión de Movilidad, Tránsito y Transporte, el rango de velocidad moderado en el sector urbano para vehículos livianos es de 50 a 60 kilómetros por hora y para el transporte público el rango de velocidad moderado es de 40 a 50 kilómetros por hora [42].

Tomando en cuenta las condiciones se determinó un tiempo adecuado de 4 segundos para el intervalo de tiempo de luz en amarillo y un intervalo de tiempo en luz en verde igual al intervalo de tiempo de luz en rojo de 41 segundos, como se ve en la Figura 27, que permite una circulación vehicular regular en el ciclo de semaforización igual a 90 segundos tanto para el sémaforo A y semáforo B.

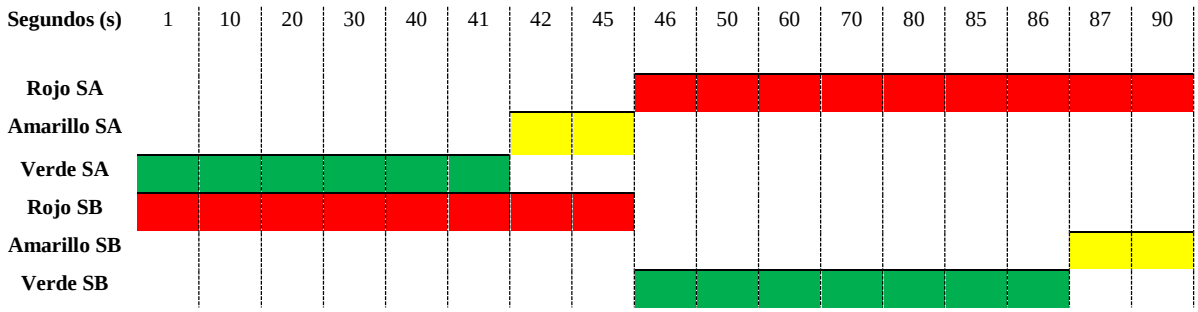


Figura 27: Intervalos de tiempo de la fase de semaforización Uno.

Elaborado por: El investigador.

- **Fase de semaforización dos:**

La fase de semaforización dos descongestiona el tráfico vehicular en la Av. Cevallos, tiene un ciclo de semaforización igual a 90 segundos, se establece un intervalo de tiempo en luz verde mayor al intervalo de tiempo en luz rojo que en este caso será mínimo en el semáforo A, y un intervalo de tiempo de luz en amarillo modificado.

Para el cálculo del intervalo mínimo de tiempo de semaforización en rojo se tomaron en cuenta las siguientes condiciones:

Según estudios realizados internacionalmente se recomienda que el tiempo de semaforización para la luz en rojo varié según según el ancho de la calle, para un ancho de 16 m el tiempo de semaforización en rojo recomendado es de 10 segundos para un peatón joven y 4 segundos más para un anciano en cruzar la calle.

Según los datos de la Dirección de Gestión de Movilidad, Tránsito y Transporte el ancho de la Av. Cevallos es igual a 13 metros y ancho de la Calle Eugenio Espejo igual a 6.5 metros.

Tomando en cuenta las condiciones se determinó un tiempo mínimo de 16 segundos para el intervalo de tiempo en luz roja, un intervalo de tiempo en modificado de luz amarilla de 4 segundos y un intervalo de tiempo de luz verde de 70 segundos para el sémaforo A, como se ve en la Figura 28, el cual permite tener una mayor circulación vehicular en la Av. Cevallos.

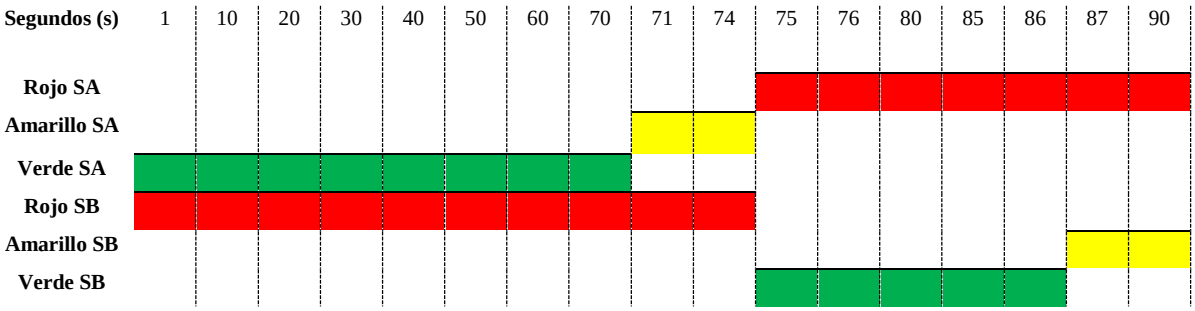


Figura 28: Intervalos de tiempo de la fase de semaforización Dos.

Elaborado por: El investigador.

- **Fase de semaforización tres:**

La fase de semaforización tres descongestiona el tráfico vehicular de la Calle Eugenio Espejo, tiene un ciclo de semaforización de 90 segundos, se establece un intervalo de tiempo en luz verde de 70 segundos mayor al tiempo de intervalo en luz rojo mínimo de 16 segundos para el semáforo B y un intervalo de tiempo de luz en amarillo modificado de 4 segundos, como se ve en la Figura 29, con la finalidad de brindar mayor circulación vehicular en la Calle Eugenio Espejo.

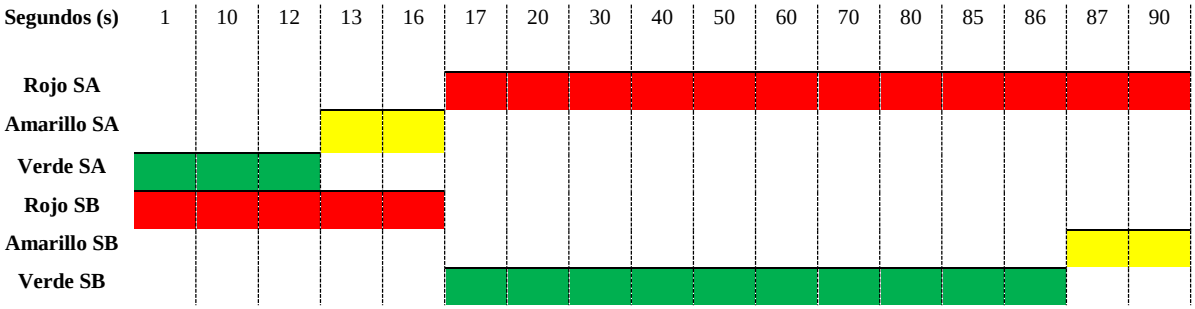


Figura 29: Intervalos de tiempo de la fase de semaforización Tres.

Elaborado por: El investigador.

3.12.5 Desarrollo del algoritmo para el control de tráfico vehicular en el servidor de niebla.

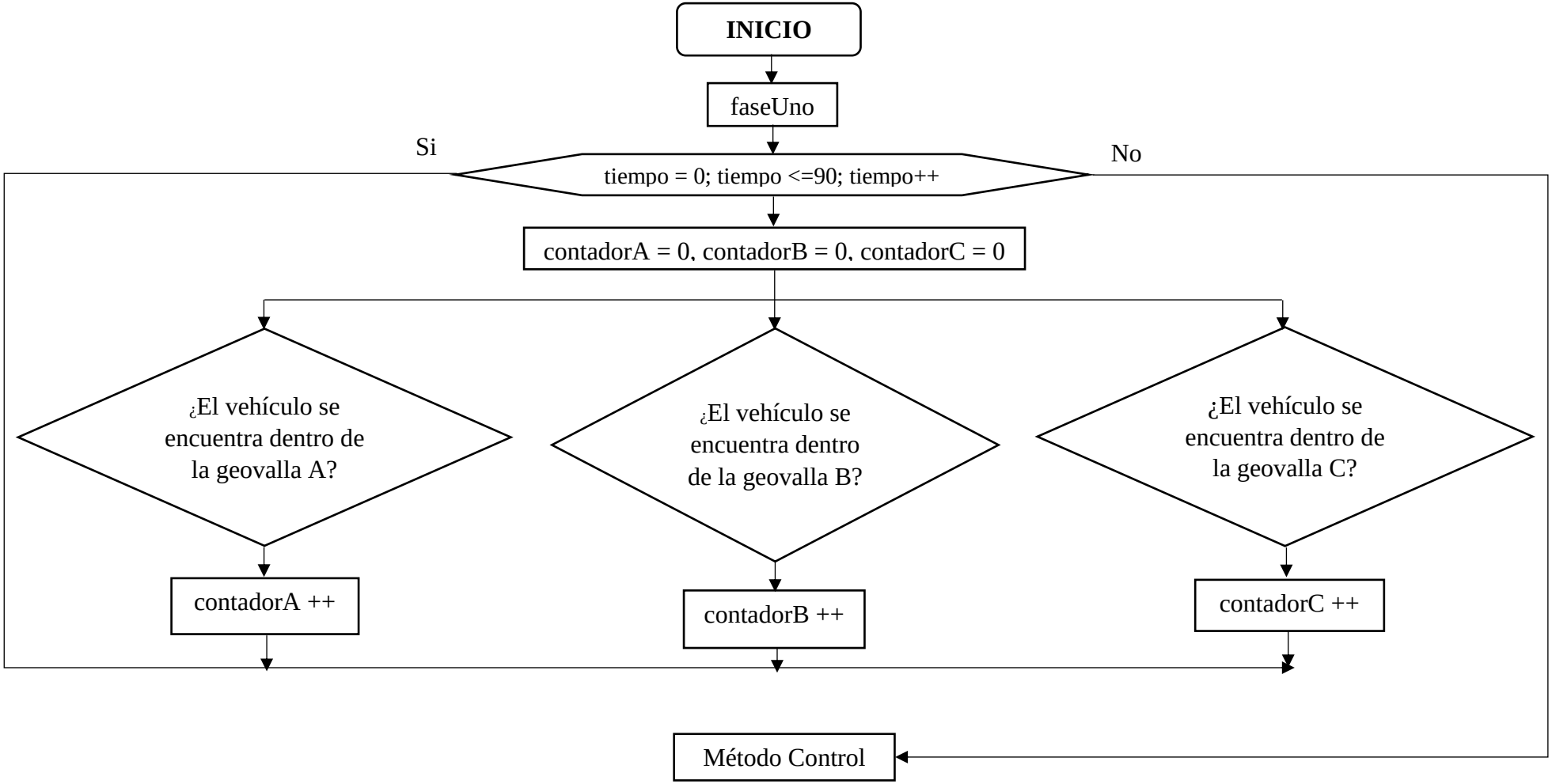
Para el desarrollo del algoritmo que permite una gestión eficiente de los datos en el servidor de niebla o nodo fog con el objetivo de contar el número de vehículos en la intersección y cambiar los intervalos de tiempo de semaforización se ha realizado el esquema de la Figura 30.

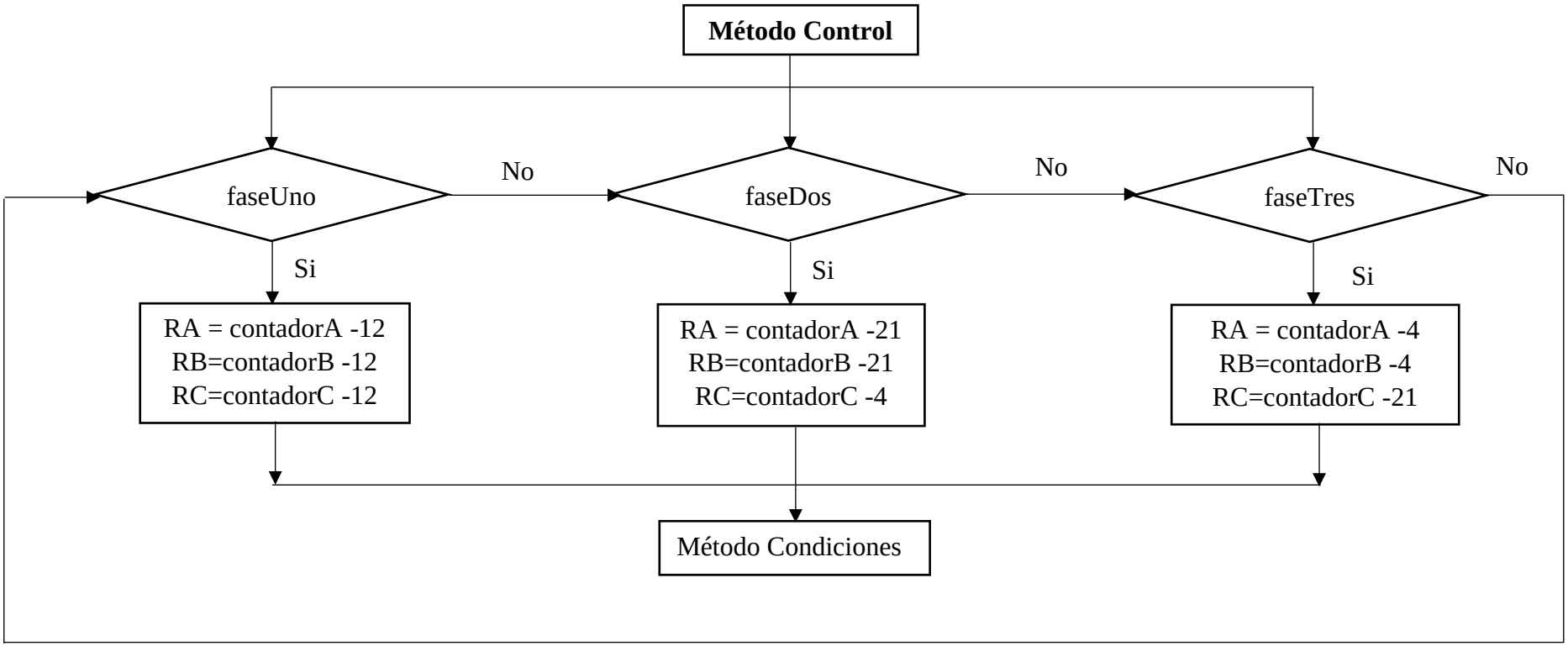
El algoritmo empieza ejecutando la fase "Uno" que tiene un ciclo de 90 segundos para finalizar, tiempo en el cual se realiza el conteo de los vehículos en la geovalla A, B, C y se representan por tres variables: contador A, contador B y contador C, luego se realiza un control de manera cíclica para saber cuántos vehículos se encuentran esperando un cambio de fase de semaforización de la siguiente manera:

- El control que se realiza cuando se ejecuta la fase "Uno" es restar a cada contador el valor de 12 que representa el número de vehículos que cruzan en 40 segundos en el intervalo de tiempo en luz verde del semáforo A y B, para poder determinar una de las 8 posibilidades mediante las variables RA, RB y RC que permiten volver a la fase "Uno" o cambiar a la fase "Dos" o "Tres.

- Cuando se ejecuta la fase "Dos" se resta al contador A y B el valor de 21 que representa el número de vehículos que cruzan en 70 segundos en el intervalo de tiempo en luz verde del semáforo A y restar el valor de 4 al contador C que representa los 12 segundos en el intervalo de tiempo en luz verde del semáforo B, para determinar una de las posibilidades y ejecutar nuevamente la fase "Dos" o cambiar a la fase "Uno" o "Tres".

- El control que se realiza cuando se ejecuta la fase "Tres" es restar al contador C el valor de 21 que representa el número de vehículos que cruzan en 70 segundos en el intervalo de tiempo en luz verde en el semáforo B y restar el valor de 4 al contador A y B que es el intervalo de tiempo de luz verde del semáforo A, para poder determinar una de las 8 posibilidades y volver a la fase "Tres" o cambiar a la fase "Uno " o "Dos.


INICIO
faseUno
Si
tiempo = 0; tiempo <=90; tiempo++
No
contadorA = 0, contadorB = 0, contadorC = 0
¿El vehículo se encuentra dentro de la geovalla A?
¿El vehículo se encuentra dentro de la geovalla B?
¿El vehículo se encuentra dentro de la geovalla C?
contadorA ++
contadorB ++
contadorC ++
Método Control


Método Control
faseUno
No
faseDos
No
faseTres
No
Si
Si
Si
RA = contadorA -12
RB=contadorB -12
RC=contadorC -12
RA = contadorA -21
RB=contadorB -21
RC=contadorC -4
RA = contadorA -4
RB=contadorB -4
RC=contadorC -21
Método Condiciones

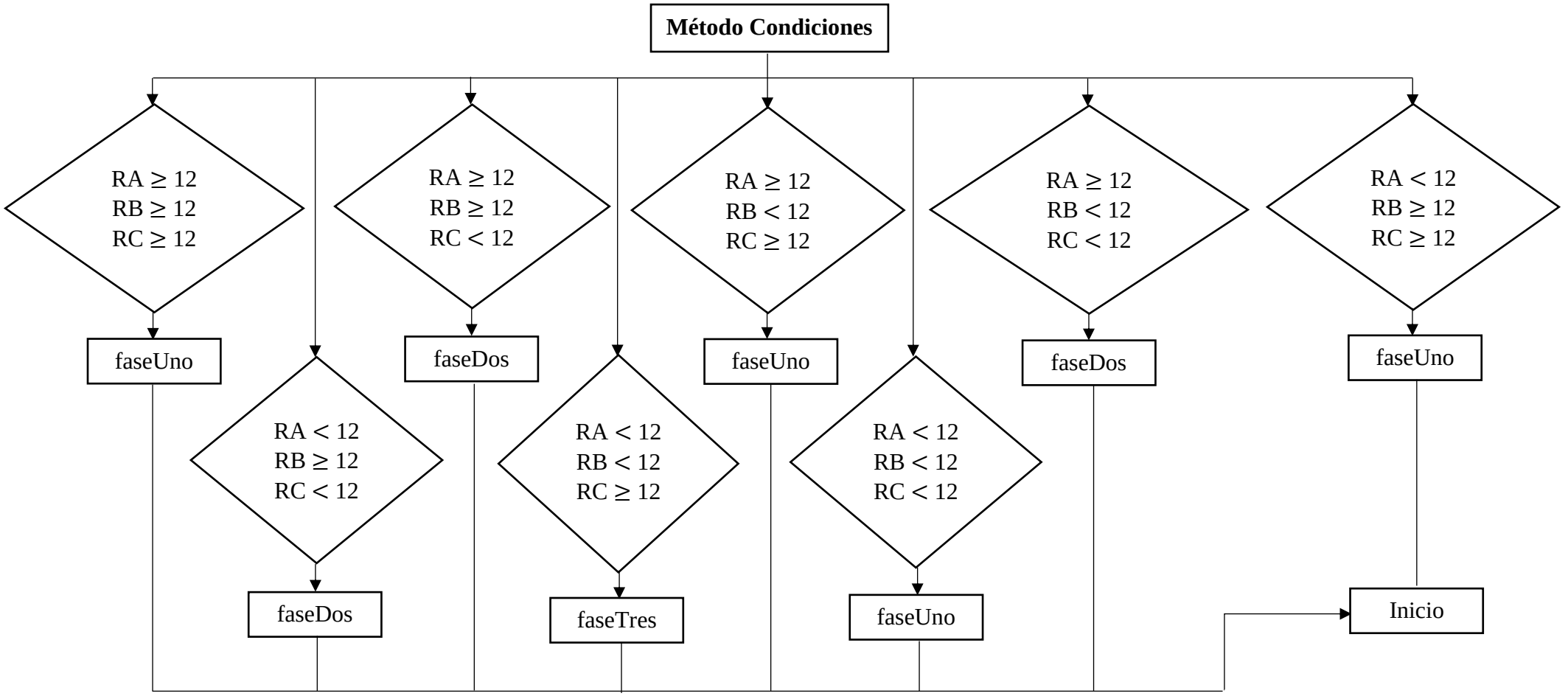


Figura 30: Algoritmo diseñado para el control de tráfico vehicular.

Elaborado por: El investigador.

3.13 Diseño de la interfaz web en el servidor de niebla

El diseño de la interfaz web en el servidor de niebla tiene como objetivo monitorear el funcionamiento de los semáforos A y B, estado del sistema y la posición de los vehículos en tiempo real, es realizado mediante los lenguajes de programación HTML y JavaScript con sus respectivas librerías de comunicación para Mqtt sobre Websockets y el diseño web utilizando las librerías de Bootstrap y estilos CSS, la interfaz web consta de una página de inicio y una página de monitoreo del sistema de control de tráfico vehicular aplicando el algoritmo diseñado anteriormente.

3.13.1 Página de inicio de la interfaz web en el servidor de niebla

La paginita de inicio de la interfaz web en el servidor de niebla se realiza mediante HTML y CSS el cual permite al usuario autenticarse para acceder a la página de monitoreo del sistema de control de tráfico vehicular mediante el archivo index.html, el usuario y contraseña son establecidos en el archivo de configuración de JavaScript llamado “inicio.js”, con el fin de realizar un control para el ingreso del sistema como se ve a en la Figura 31:

Figura 31: Página de inicio de la interfaz web en el servidor de niebla.

Elaborado por: El investigador.

3.13.2 Monitoreo del sistema de control de tráfico vehicular

Para realizar el monitoreo del sistema de control de tráfico vehicular se siguen los siguientes pasos:

1. Establecer la comunicación para la suscripción y publicación de tópicos utilizando la librería de JavaScript para Mqtt definido por el archivo "mqttws31.js", y crear una instancia en el archivo de programación denominado niebla.js que permite la conexión Mqtt sobre WebSocket en el servidor de niebla mediante la siguiente línea de código:

   client = new Paho.MQTT.Client("192.168.1.20", 9001, clientId);

- Suscripción de tópicos en el servidor de niebla para obtener la información del vehículo mediante las siguientes líneas de código:

  Para suscribirse al tópico placa:
  client.subscribe("Placa");
  Para suscribirse al tópico latitud:
  client.subscribe("Latitud");
  Para suscribirse al tópico longitud:
  client.subscribe("Longitud");

- Publicación de tópicos en el servidor de niebla correspondientes a las fases de semaforización para el semáforo A y B :

  Para definir los mensajes que tienen el dato correspondiente al número de fase del semáforo, se utiliza las siguientes líneas de código:

  message1 = new Paho.MQTT.Message("Número de Fase");

  message2 = new Paho.MQTT.Message("Número de Fase");

  Para publicar los mensajes en el tópico Semáforo/A y semáforo/B, se realiza mediante la siguiente línea de código:

  message1.destinationName = 'Semaforo/A'

message2.destinationName = 'Semaforo/B'

2. Añadir el mapa de leaflet en la interfaz web el cual permite monitorear la ubicación de los vehículos en tiempo real, añadiendo marcadores para cada nueva posición en el mapa, mediante la siguiente línea de código en el archivo interfaz_de _monitoreo.html:

   <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"> </script>

   Para insertar un marcador o icono en el mapa que represente la ubicación del vehículo en tiempo real se utiliza el código que se muestra a continuación en el archivo niebla.js:

   vehiculo = L.marker([latitud, longitud], { icon: iconMarker2 }).addTo(mapa);

   Para los marcadores que representan los semáforos A y B ubicados en el mapa de leaflet se realiza de la misma manera:

   semaforoA = L.marker([-1.240581,-78.6257843],{ icon:iconMarker1 }).addTo(mapa);

   semaforoB = L.marker([-1.2403109, -78.625827], { icon: iconMarker1 }).addTo(mapa);

3. Agregar las geovallas en el mapa de leaflet definidas en el algoritmo diseñado anteriormente para el realizar el conteo de vehículos mediante las siguientes líneas de código en el archivo niebla.js:

- Definir la posición geográfica respecto a la latitud y longitud de la geovalla A, B y C en el mapa de leaflet:

geovallaA = L.circle([-1.2401836, -78.6253437], }).addTo(mapa);
geovallaB = L.circle([-1.240779, -78.625994], }).addTo(mapa);
geovallaC = L.circle([-1.2409819, -78.6252571], }).addTo(mapa);

- Especificar las características de las geovallas mediante las siguientes líneas de código:

Añadir color al borde de la geovalla mediante:
color: '#454545',
Especificar el nivel de color de opacidad de la geovalla:
fillOpacity: 0.2,
Definir el radio de la geovalla mediante:
radius:24,

4. Programar las fases de semaforización se utilizó la función setTimeout el cual se encarga de ejecutar otra función después de un tiempo determinado y de esta manera realizar los intervalos de tiempo de semaforización del sistema en la interfaz web.

Primero se coloca dos imágenes mediante el atributo src utilizando la siguiente línea de código, el cual encenderá la luz vede para el semáforo A y la luz roja para el semáforo B:

img1.src = 'imagenes/verde.png';
img2.src = 'imagenes/rojo.png';

Luego se ejecuta la función setTimeout para que encienda la luz en amarillo a los 42 segundos para el semáforo A, dejando la luz roja del semáforo B sin realizar ningún cambio:

setTimeout(function(){img1.src = 'imagenes/amarillo.png' }, 42000);

A los 46 segundos ejecuta nuevamente otra función setTimeout para que encienda la luz roja en el semáforo A y la luz verde en el semáforo B:

setTimeout(function(){img1.src = 'imagenes/rojo.png',

img2.src = 'imagenes/verde.png' }, 46000);

Finalmente se ejecuta nuevamente la función setTimeout a los 87 segundos para encender la luz amarilla del semáforo B sin realizar ningún cambio en el semáforo A.

setTimeout(function(){img2.src = 'imagenes/amarillo.png' }, 87000);

5. Monitorear el estado y la actividad del sistema mediante un cuadro de información en la interfaz web y botón para salir del sistema:

   Se obtiene el estado conectado o desconectado del sistema mediante:

   document.getElementById("estado").value = estado;

   Para obtener la actividad del sistema que puede ser: Circulación normal, Descongestión Av. Cevallos y Descongestión Calle Espejo que responde al tipo de fase que se ejecuta en el servidor de niebla, se realiza mediante:

   document.getElementById("actividad").value = tipoDeActividad;

   Añadir la función salir,cuando se seleccione el botón el cual regresa a la página de inicio, programado mediante la siguiente línea de código:

   window.location = "index.html";

6. Implementación de la interfaz web para el monitoreo del sistema de control de tráfico vehicular utilizando el diseño web de bootstrap como se ve en la Figura 32, para organizar los elementos en cuadrículas definidas por columnas y filas, además el uso de componentes como tarjetas el cual permite dar una buena presentación a través de colores de fondo y estilos de fuente, todo esto definido en el archivo interfaz_de _monitoreo.html:

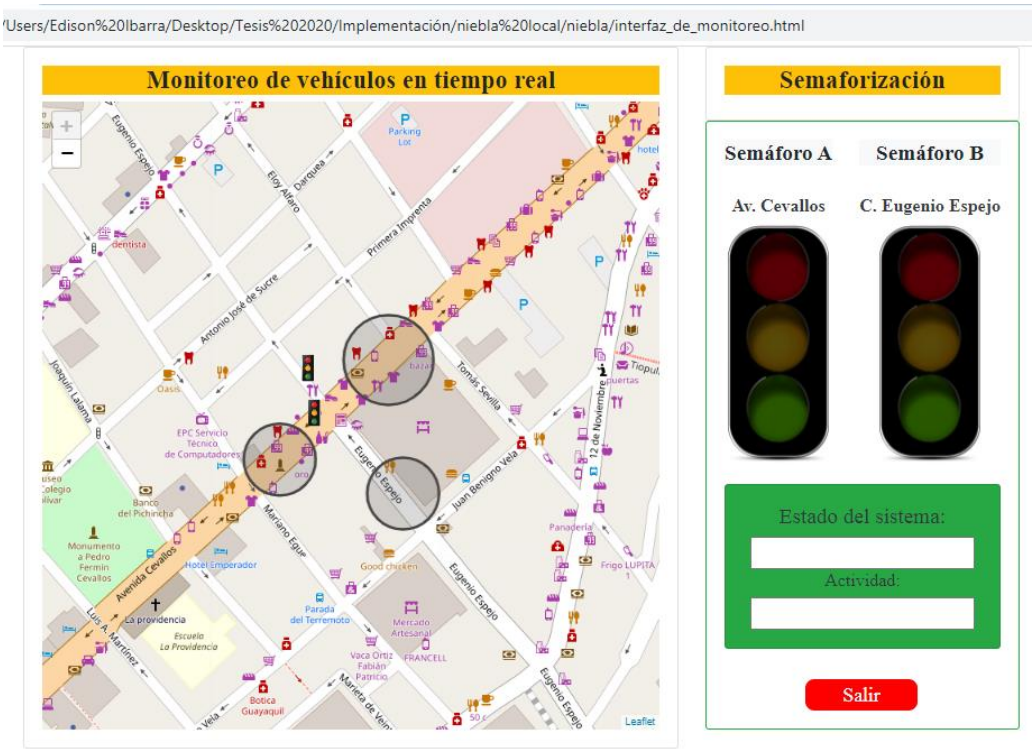


Figura 32: Página de monitoreo del sistema de control de tráfico vehicular de la interfaz web en el servidor de niebla.

Elaborado por: El investigador.

3.14 Diseño de la interfaz web para el servidor en la nube

El diseño de la interfaz web para el servidor en la nube tiene como finalidad obtener registros históricos del sistema de control de tráfico vehicular, mediante gráficas y tablas de reportes sobre el conteo de vehículos y fases de semaforización generadas en determinada fecha, es realizado mediante los lenguajes de programación HTML, PHP y JavaScript con sus respectivas librerías de comunicación para Mqtt sobre

Websockets y el diseño web utilizando las librerías de Bootstrap, Highcharts y estilos CSS, la interfaz web del servidor en la nube consta de una página de inicio y una página de registro de históricos del sistema de control de tráfico vehicular.

3.14.1 Página de inicio de la interfaz web del servidor en la nube

La paginita de inicio de la interfaz web para el servidor en la nube se realiza mediante HTML y CSS el cual permite al usuario autenticarse para acceder a la página de registro de históricos del sistema de control de tráfico vehicular mediante el archivo index.html, el usuario y contraseña de igual manera que en el servidor de niebla son establecidos en el archivo de configuración de JavaScript llamado "inicio.js", con el fin de realizar un control para el ingreso del sistema como se ve a en la Figura 33:

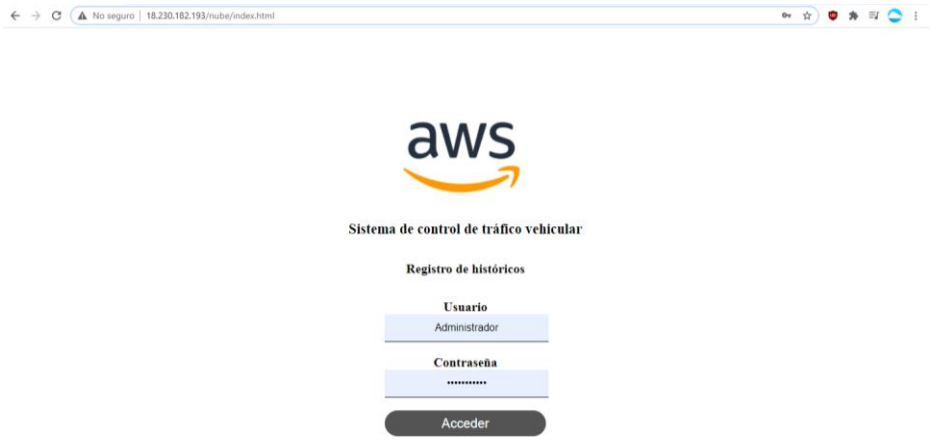


Figura 33: Página de inicio de la interfaz web para el servidor en la nube.

Elaborado por: El investigador.

3.14.2 Registro de históricos del sistema de control de tráfico vehicular

Para realizar el registro de históricos del sistema de control de tráfico vehicular se siguen los siguientes pasos:

1. Establecer la comunicación para la suscripción y publicación de tópicos utilizando la librería de JavaScript para Mqtt definido por el archivo "mqttws31.js", y crear

una instancia en el archivo de programación denominado nube.js que permite la conexión Mqtt sobre WebSocket para el servidor en la nube mediante la siguiente línea de código:

client = new Paho.MQTT.Client("18.230.182.193", 9001, clientId);

- Suscripción de tópicos en el servidor en la nube para obtener información del conteo de vehículos mediante las siguientes líneas de código:

  Para suscribirse al tópico Av. Cevallos sentido de circulación Oeste con calidad de servicio igual a uno:
  client.subscribe('Av.Cevallos/sent.Oeste', {qos: 1});
  Para suscribirse al tópico Av. Cevallos sentido de circulación Este con calidad de servicio igual a uno:
  client.subscribe('Av.Cevallos/sent.Este', {qos: 1});
  Para suscribirse al tópico Calle Eugenio Espejo con calidad de servicio igual a uno:
  client.subscribe('Calle/EugenioEspejo', {qos: 1});

2. Realizar el grafico 1 que permite registrar el conteo de vehículos en tiempo real utilizando la librería de highcharts, en el cual se representa los datos recibidos en un plano de dos ejes (x, y), en el eje "x" se muestra el tiempo en horas minutos y segundos en el cual se generó el conteo de vehículos y en el eje "y" el número de vehículos contados, se implementa través de las siguientes líneas de código en el archivo nube.js:

   Definir la variable para el grafico 1 de highcharts:
   chart = new Highcharts.Chart();

   Determinar el eje x para representar la variable en función del tiempo:
   xAxis: { type: 'datetime',},

   Determinar el eje y para representar el número de vehículos obtenidos mediante una serie de datos:
   yAxis: { text: 'Número de vehículos', series: [ ]},

3. Realizar las diferentes consultas a la base de datos phpmyadmin del servidor en la nube para generar la gráfica 2 sobre consultas, y las tablas 1 y 2 de reportes del sistema, mediante la siguientes líneas de código PHP:

- Conexión a la base de datos de phpmyadmin del servidor en la nube de aws:

  $connection = mysqli_connect("localhost", "root", "servidoraws","Datos");

- Consulta a la base de datos específicamente a la tabla denominada contenido para obtener información sobre la fecha y el dato recibido en el tópico 'Av.Cevallos/sent.Oeste', el cual tiene el mensaje del número de vehículos contados en la geovalla A:

  $geovallaA=mysqli_query($connection,"SELECT UNIX_TIMESTAMP(Recibido, Mensaje FROM Contenido WHERE year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day' AND `Topico`= 'Av.Cevallos/sent.Oeste'");

- Consulta a la base de datos para obtener información sobre la fecha y el dato recibido en el tópico 'Av.Cevallos/sent.Este', el cual tiene el mensaje del número de vehículos contados en la geovalla B:

  $geovallaB=mysqli_query($connection,"SELECT UNIX_TIMESTAMP(Recibido, Mensaje FROM Contenido WHERE year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day' AND `Topico`= 'Av.Cevallos/sent.Este'");

- Consulta a la base de datos para obtener información sobre la fecha y el dato recibido en el tópico 'Calle/EugenioEspejo', el cual tiene el mensaje del número de vehículos contados en la geovalla C:

$geovallaC=mysqli_query($connection,"SELECT UNIX_TIMESTAMP(Recibido, Mensaje FROM Contenido WHERE year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day' AND `Topico`= 'Calle/EugenioEspejo'");

- Consulta a la base de datos para obtener información sobre el identificador, tópico, mensaje y la fecha en el cual se recibió los datos correspondientes para generar el reporte del conteo de vehículos del sistema:

  $reporteVehiculos = mysqli_query($connection, "SELECT Id, Topico, Mensaje, Recibido FROM Contenido WHERE year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day' AND `Topico`!= 'Semaforo/A' AND `Topico`!= 'Semaforo/B'");

- Consulta a la base de datos para obtener información sobre el identificador, tópico, mensaje y la fecha en el cual se recibió los datos correspondientes para generar el reporte de las fases de semaforización del sistema:

  $reporteFases = mysqli_query($connection, "SELECT Id, Topico, Mensaje, Recibido FROM Contenido WHERE year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day' AND `Topico`!= 'Av.Cevallos/sent.Oeste' AND `Topico`!= 'Av.Cevallos/sent.Este' AND `Topico`!= 'Calle/EugenioEspejo'");

4. Realizar el grafico 2 de highchart sobre consultas del número de vehículos, y las tablas 1 y 2 correspondientes a reportes de las fases de semaforización y del conteo de vehículos del sistema, generadas mediante una fecha seleccionada por el administrador del sistema en la interfaz web:

- Para realizar la gráfica 2 sobre consultas del conteo de vehículos del sistema, se utilizan las siguientes líneas de código:

  Definir la variable para la gráfico 2 de highcharts:

$('#grafica2').highcharts();

Determinar el eje x para representar la variable en función del tiempo:
xAxis: { type: 'datetime',},

Determinar el eje y para representar el número de vehículos obtenidos mediante los datos generados por las consultas realizadas anteriormente a la base de datos phpmyadmin denominadas “geovallaA, geovallaB y geovallaC”:

Obtener el dato del número de vehículos de la geovalla A:
while($registrosA = mysqli_fetch_array($geovallaA))
Obtener el dato del número de vehículos de la geovalla B:
while($registrosB = mysqli_fetch_array($geovallaB))
Obtener el dato del número de vehículos de la geovalla C:
while($registrosC = mysqli_fetch_array($geovallaC))

- Para realizar la tabla 1 la cual permite generar reportes del conteo de vehículos del sistema de control de tráfico vehicular se utiliza las siguientes líneas de código:

Definir las cuatro columnas y una fila que componen la tabla 1 para el identificador, el detalle, la cantidad y la fecha de recibido mediante las etiquetas “td” y “tr” respectivamente:

<tr>
<td>Identificador</td>
<td>Detalle</td>
<td>Cantidad</td>
<td>Recibido</td>
</tr>

Llenar la tabla 1 mediante los datos generados por la consulta realizada anteriormente a la base de datos phpmyadmin denominada “reporteVehiculos”:

while($tabla1=mysqli_fetch_array($reporteVehiculos))

- Para realizar la tabla 2 la cual permite generar reportes de las fases de semaforización del sistema de control de tráfico vehicular se utiliza las siguientes líneas de código:

  Definir las cuatro columnas y una fila que componen la tabla 2 para el identificador, el detalle, la fase y la fecha de recibido mediante las etiquetas “td” y “tr” respectivamente:

  <tr>
  <td>Identificador</td>
  <td>Detalle</td>
  <td>Fase</td>
  <td>Recibido</td>
  </tr>

  De igual manera para llenar la tabla 2 se utilizan los datos generados por la consulta realizada anteriormente a la base de datos phpmyadmin denominada “reporteFases”:

  while($tabla2=mysqli_fetch_array($reporteFases))

  Finalmente añadir la función salir, cuando se seleccione el botón el cual regresa a la página de inicio, programado mediante la siguiente línea de código en la interfaz web de la nube:

  window.location = "index.html";

5. Implementación de la interfaz web para el registro de históricos del sistema de control de tráfico vehicular utilizando el diseño web de bootstrap para organizar los elementos en la pantalla y estilos css para el diseño de las gráficas y tablas

como se ve en la Figura 34, todo esto definido en el archivo interfaz_de _históricos.php:

Figura 34: Página de registro de históricos del sistema de control de tráfico vehicular de la interfaz web del servidor en la nube.

Elaborado por: El investigador.

3.15 Comunicación del servidor en la niebla con el servidor en la nube

Para realizar la comunicación del servidor en la niebla local con el servidor en la nube de aws, se configura un "bridge" o puente el cual permite conectar el bróker Mqtt local a otro bróker Mqtt en la nube, para esto se debe modificar el archivo de configuración de mosquitto en uno de los brókers, en este caso se configura el bróker del servidor siguiendo los siguientes pasos que se explican a continuación:

1. Abrir el archivo de configuración mosquitto.conf mediante el siguiente comando:

   nano /etc/mosquitto/mosquitto.conf

2. Ingresar el siguiente comando para establecer el puente de comunicación entre la nube y la niebla:

   connection bridge-01

3. Definir la dirección ip del servidor en la nube y el puerto de comunicación Mqtt a cual se va a conectar el servidor en la niebla:

   address 18.230.182.193:1883

4. Definir los tópicos que se van a enviar al servidor en la nube de aws:

   Tópicos para enviar los mensajes correspondientes a las fases de semaforización para el semáforo A y B con calidad de servicio igual a 0:

   topic Semaforo/A out 0,

   topic Semaforo/B out 0,

   Tópicos para enviar los mensajes correspondientes al conteo de vehículos en la Av. Cevallos en sus dos sentidos de circulación y en la Calle Eugenio Espejo de igual manera con calidad de servicio igual a 0:

   topic Av.Cevallos/sent.Oeste out 0,

   topic Av.Cevallos/sent.Este out 0,

   topic Calle/EugenioEspejo out 0,

5. Finalmente reiniciar el servicio de mosquitto para que se apliquen los cambios realizados mediante el siguiente comando:

   systemctl restart mosquitto

3.16 Construcción del prototipo de semáforo inteligente

La construcción del prototipo de semáforo inteligente se divide en tres etapas, en la primera etapa se fabrica la placa electrónica, en la segunda etapa se realiza la programación de las funciones del semáforo inteligente y en la tercera etapa se diseña la estructura física para el semáforo A y B del sistema de control de tráfico vehicular.

3.16.1 Fabricación de la placa electrónica del semáforo inteligente

Para la fabricación de la placa electrónica del semáforo inteligente el cual tiene como función conectarse al servidor en la niebla mediante un microcontrolador encender y apagar luces de semaforización dependiendo de las fases que se requiera, se sigue los siguientes pasos:

1. Realizar el circuito electrónico para el semáforo inteligente:

El circuito electrónico del semáforo inteligente está compuesta por la fuente de alimentación, el circuito de control y el circuito de potencia. La fuente de alimentación se encarga de entregar la energía para que los componentes del circuito electrónico puedan funcionar, está compuesta por una fuente de corriente alterna AC de 120 V y un convertidor de corriente alterna a corriente directa AC DC HKLP1, el circuito de control permite dar las órdenes que se deben realizar mediante el microcontrolador NodeMCU y el circuito de potencia recibe las señales digitales del circuito de control y activa una salida de alto voltaje como son las luces del semáforo mediante transistores y relés, en la Figura 35 se puede observar el diseño del circuito del semáforo inteligente realizado mediante el software de diseño de placas de circuito impreso Eagle:

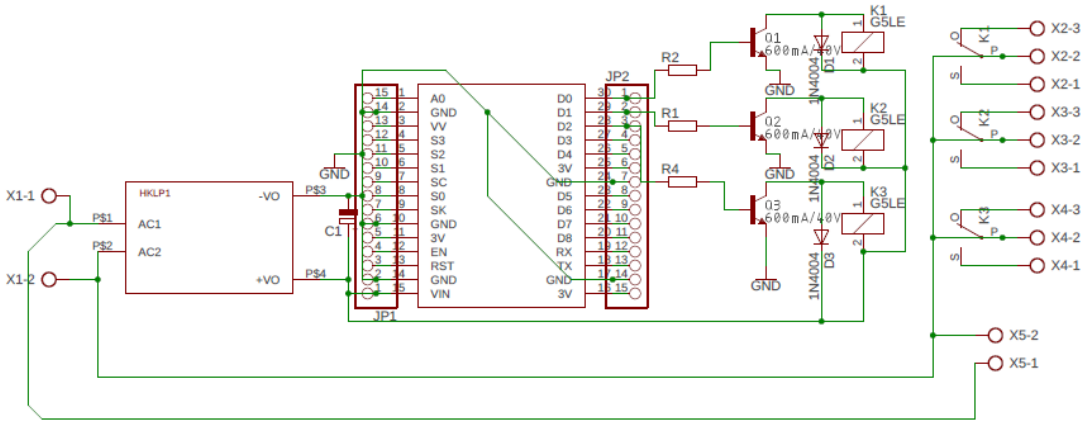


Figura 35: Circuito electrónico del semáforo inteligente realizado en el software Eagle.

Elaborado por: El investigador.

El funcionamiento del circuito electrónico de la Figura 35, se basa en encender y apagar tres luces de 120 V mediante las salidas digitales D0, D1 y D2 de acuerda a las funciones programadas en el microcontrolador NodeMCU. El circuito de control al enviar un voltaje de nivele lógico alto o 5 V en el pin digital D0 el circuito de potencia recibe dicha señal y enciente la luz roja mediante el relé K1, cuando el nivel lógico del pin digital D0 es bajo o 0 V la luz roja se apaga, para encender la luz amarilla mediante el relé K2 el pin digital D1 debe tener un nivel lógico alto o nivel bajo para apagar la luz amarilla y para encender la luz verde a través del relé K3 se debe tener un nivel lógico alto en el pin digital D2 por el contrario el nivel lógico bajo para apagar la luz verde.

El funcionamiento que tiene cada componente del circuito electrónico se explica a continuación:

Para la fuente de alimentación del circuito se utilizó un convertidor AC DC modelo HKLP1 el cual transforma la tensión de entrada de 100 - 220 V de corriente altera a una salida de 5 V en corriente continua máxima de 0.6 A, es utilizado para alimentar el microcontrolador mediante el pin VIN el cual requiere un voltaje de 5 V con un consumo de corriente mínima 0.07 A, y se conecta a un capacitor electrolítico de 100uF el cual se usa para eliminar las sobretensiones y mantener una corriente continua y estabilizada a 5 V en la fuente de alimentación del microcontrolador.

Para el circuito de control se usa un microcontrolador NodeMCU modelo ESP8266 el cual permite obtener niveles lógicos “alto” y “bajo”, representados por 0 V y 5 V en los pines digitales de salida obedecen a la programación de las funciones del semáforo inteligente, además se encuentra conectada a resistencias de 1KΩ que se usa para controlar la corriente de salida de los pines digitales y no superar los 0.07 A de corriente en la fuente alimentación.

El circuito de potencia está conformado por los transistores NPN modelo 2N3904 que se utilizan para activar los relés en el circuito electrónico, su funcionamiento se basa en utilizar la salida del circuito de control de nivel lógico alto 5 V, para superar la tensión de umbral de la base del transistor que es de 0.6 V y de esta manera hacer circular la corriente entre la base y la masa, llevando al transistor al estado de conducción entre el colector y el emisor por lo tanto se cierra el circuito eléctrico del relé y finalmente se activa. Para proteger al transistor de los picos de tensión que se presenta al desactivar el relé interrumpiendo la corriente que pasa por la bobina, se conecta en paralelo un diodo rectificador modelo 1N4004 inversamente polarizado para controlar estos picos de tensión con polaridad opuesta.

Para activar las cargas resistivas o luces de 120 V se utilizan relés modelo SRD-06VDC-SL-C el cual permite conmutar cargas con bajo voltaje 5 V, el control del circuito eléctrico del relé está compuesto por una bobina que se activa y produce un campo electromagnético que hace que el contacto del relé que esta normalmente abierto se cierre y permita el paso de la corriente y se encienda las luces, por otro lado cuando dejamos de suministrar corriente a la bobina el campo electromagnético desaparece y el contacto del relé se vuelve abrir dejando sin corriente a las luces.

2. Ubicación de los componentes en la placa electrónica para generar la pista del circuito electrónica del semáforo inteligente:

La ubicación de los componentes de la placa electrónica se puede realizar mediante el software Eagle como se puede ver en la Figura 36 a la izquierda, con el fin colocar los elementos adecuadamente y de esta manera utilizar el menor espacio posible, y establecer las entradas y salidas del circuito electrónico a la derecha e izquierda

correspondientemente para obtener una trayectoria de corriente eléctrica en un circuito cerrado. En la Figura 36 a la derecha se observa la pista de la placa electrónica generada para el semáforo inteligente.

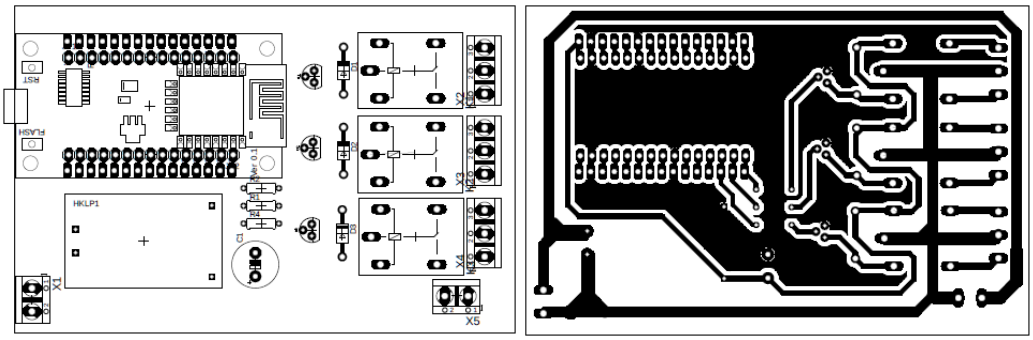


Figura 36: Ubicación de los componentes y pista de la placa electrónica del semáforo inteligente.

Elaborado por: El investigador.

3. Montaje y prueba de funcionamiento de los elementos del circuito electrónico en la placa diseñada anteriormente para el semáforo A y B como se ve en la Figura 37:

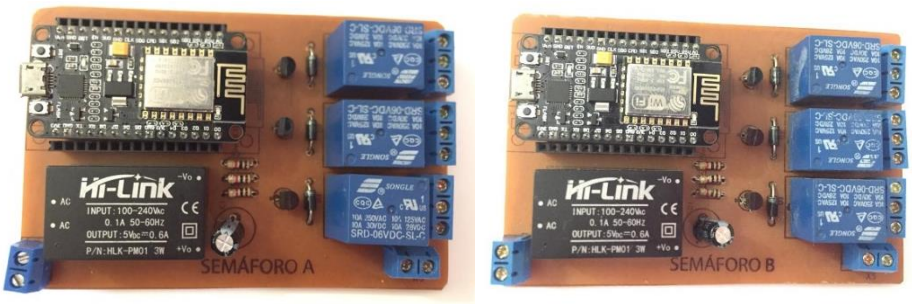


Figura 37: Montaje de los elementos del circuito para el semáforo A y B.

Elaborado por: El investigador.

Para verificar el funcionamiento de la fuente de alimentación en la placa del circuito del semáforo se utiliza un multímetro digital para registrar el voltaje y corriente de entrada y salida del convertidor AC DC HKLP1, los datos obtenidos se registran en la Tabla 9.

Tabla 9: Medidas de voltaje y corriente en el convertidor AC DC HKLP.

HKLP1	Entrada	Salida
Voltaje (V)	131.8	5.02
Corriente (A)	0.1	0.6

Elaborado por: El investigador.

Los datos obtenidos de corriente y voltaje en la salida del convertidor AC DC HKLP permiten dar la energía suficiente para el funcionamiento adecuado del microcontrolador NodeMCU el cual opera a 5 V y 0.07 A mediante el pin de alimentación VIN del microcontrolador.

Para verificar funcionamiento del microcontrolador NodeMCU se mide el voltaje y corriente en las salidas digitales utilizadas D0, D1 y D2 al tener un nivel lógico alto y bajo, los datos obtenidos se muestra en la Tabla 10.

Tabla 10: Medidas de voltaje y corriente en los pines digitales del microcontrolador NodeMCU.

NodeMCU	D0	D0	D1	D1	D2	D2
Nivel lógico	alto	bajo	alto	bajo	alto	Bajo
Voltaje (V)	3.20	0	3.20	0	3.20	0
Corriente (A)	0.04	0	0.04	0	0.04	0

Elaborado por: El investigador.

Los datos obtenidos en los pines digitales D0, D1 y D2 del microcontrolador NodeMCU permiten determinar el valor de voltaje y corriente para el nivel lógico alto en el cual se registra un valor de 3.20 V y 0.04 A adecuados para poder activar un pin digital de salida en la placa electrónica y un nivel lógico bajo de 0 V y 0 A para poder desactivar un pin digital de salida en la placa electrónica.

En el circuito de potencia de igual manera se verifica los voltajes y corrientes en los transistores, diodos y relés del circuito electrónico con el fin de comprobar su correcto funcionamiento en nivel lógico alto y bajo como se muestra en la Tabla 11.

Tabla 11: Medidas de voltaje y corriente en los elementos del circuito de potencia.

Componentes	Transistor	Transistor	Diodo	Diodo	Relé	Relé
Nivel lógico	Alto	Bajo	alto	bajo	Alto	Bajo
Voltaje (V)	0.76	0.16	0.66	0.04	131.1	12.0
Corriente (A)	0.025	0	0	0	0.053	0

Elaborado por: El investigador.

Los datos obtenidos en los elementos del circuito de potencia permiten determinar el valor de voltaje y corriente necesarios para que el transistor pueda funcionar como conmutador, al registrarse un voltaje mayor a la tensión umbral de 0.76 V en nivel lógico alto permitiendo circular una corriente y activar el relé en el cual se registra un voltaje de corriente alterna de 131.1 V y 0.053 A, necesarios para encender una luz como se ve en la Figura 38, además se obtiene un voltaje en el diodo de 0.04 V y 0 A en nivel lógico bajo el cual permite determinar que no existe conducción de corriente en sentido opuesto al desactivar el relé o apagar una luz.

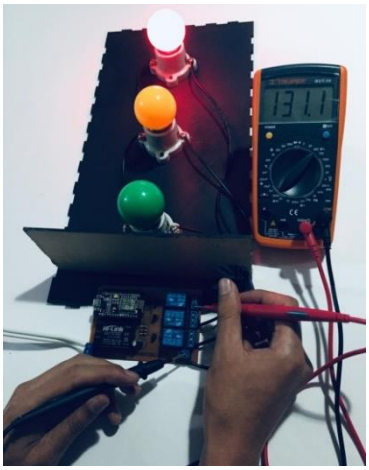


Figura 38: Pruebas de funcionamiento de los elementos del circuito en la placa electrónica mediante el multímetro digital.

Elaborado por: El investigador.

3.16.2 Programación de las funciones del semáforo inteligente

Para la programación de las funciones del semáforo inteligente mediante el microcontrolador NodeMCU modelo ESP8266 se utilizó el IDE de Arduino el cual es un software de código abierto que permite compilar programas escritos mediante el lenguaje estándar C++ en placas o microcontroladores NodeMCU.

A continuación se detallan los pasos realizados para programar las funciones del semáforo inteligente A y B:

1. Se importa las librerías requeridas del microcontrolador NodeMCU para la comunicación inalámbrica mediante Wifi y para utilizar el protocolo de comunicación Mqtt, utilizando los siguientes comandos:

   Librería para la comunicación Wifi:

   #include <ESP8266WiFi.h>

   Librería para la comunicación Mqtt:

   #include <PubSubClient.h>

2. Para conectar el microcontrolador NodeMCU con el servidor en la niebla con el objetivo de recibir las instrucciones de las diferentes fases de semaforización a ejecutarse, se utiliza los siguientes comandos:

   Definir la variable para el nombre de la red local:
   const char ssid = "FogComputing";*

   Especificar la variable para la contraseña de la red local:
   const char password = "Sistema2020";*

   Definir la variable para la conexión con el servidor de niebla:
   const char mqtt_server = " proxy19.rt3.io:39155";*

   Especificar el puerto de comunicación con el servidor de niebla mediante el protocolo Mqtt:

const int mqttPort = 1883;

3. Suscripción a los tópicos tanto para el semáforo A y semáforo B, utilizando los siguientes comandos:

   Suscripción para el tópico del semáforo A:
   client.subscribe("Semaforo/A");

   Suscripción para el tópico del semáforo B:
   client.subscribe("Semaforo/B");

4. Programación de las fases de semaforización según el mensaje recibido Uno, Dos o Tres en el tópico correspondiente, para generar los intervalos de tiempo de semaforización, a continuación se explica cómo se realiza la fase de semaforización uno:

   Primero se realiza una condición para el mensaje recibido en el tópico Semáforo/A o Semáforo/B sea igual a Uno, comparando a través del siguiente comando el primer valor del mensaje igual a “n”:

   if(mensaje[1] == 'n')

   Se escribe un nivel lógico alto al pin digital D2 definido por el valor 4 para encender la luz en color verde y un nivel lógico bajo al pin digital D0 definido por valor 16 para apagar la luz en color roja:

   Pin digital D2 en nivel lógico alto:
   digitalWrite(4, HIGH);

   Pin digital D0 en nivel lógico bajo:
   digitalWrite(16, LOW);
   Realizar un retardo de 42 segundos que representa al intervalo de semaforización en luz verde mediante el siguiente comando:

delay(42000);

Y se escribe un nivel lógico bajo en el pin digital D2 para apagar la luz en color verde y al mismo tiempo se escribe un estado lógico en alto el pin digital D1 definido por el valor 5 para encender la luz en color amarilla:

Pin digital D2 en nivel lógico bajo:

digitalWrite(4, LOW);

Pin digital D1 en nivel lógico alto:

digitalWrite(5, HIGH);

Nuevamente realizar otro retado de 4 segundos que corresponde al intervalo de tiempo de semaforización en luz amarilla mediante:

delay(4000);

Finalmente se escribe un nivel lógico bajo en el pin digital D1 para apagar la luz en color amarillo y un nivel lógico alto en el pin D0 para encender la luz en color rojo:

Pin digital D1 en nivel lógico bajo:

digitalWrite(5, LOW);

Pin digital D0 en nivel lógico alto:

digitalWrite(16, HIGH);

3.16.3 Diseño de la estructura física del prototipo de semáforo inteligente

Para el diseño de la estructura física del prototipo de semáforo inteligente se tomó en cuenta las características que tienen los semáforos instalados actualmente en la intersección ubicada en la Avenida Cevallos y Calle Eugenio Espejo, la construcción del prototipo se realiza utilizando el material de fibra de densidad media o MDF para

las diferentes partes que componen la estructura física del semáforo como son cabeza, cara, visera y lente como se explica a continuación:

- **Cabeza:** La cabeza del semáforo es la estructura que contiene a todas las partes visibles compuesta por las lámparas y boquillas, cuyas dimensiones son de 40 cm de alto, 20 cm de largo y 8.5 cm de ancho para el protótipo de semáforo inteligente, además cuenta con un orificio en la parte inferior de 1 cm de diámetro para el cable conductor como se ve en la Figura 39.

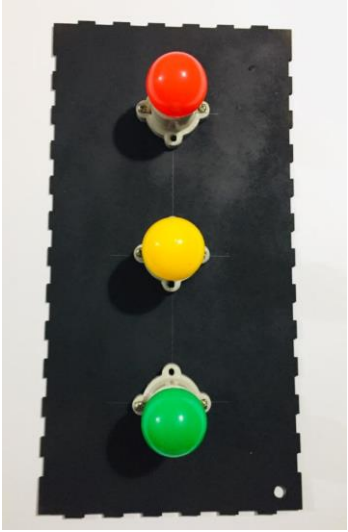


Figura 39: Cabeza de la estructura física del prototipo de semáforo inteligente.

Elaborado por: El investigador.

- **Cara:** La cara del semáforo es el conjunto de las tres unidades ópticas para la luz roja, amarilla y verde, cuyas dimensiones son de 10 cm de diámetro para cada unidad óptica y separadas 2 cm entre cada una para el protótipo de semáforo inteligente como se ve en la Figura 40.

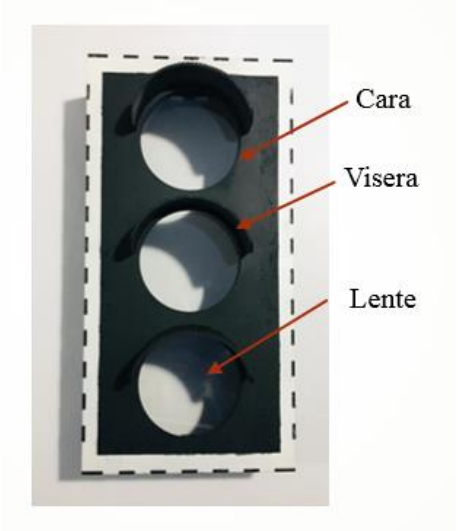


Figura 40: Cara, visera y lente de la estructura física del prototipo de semáforo inteligente.

Elaborado por: El investigador.

- **Visera:** La visera del semáforo es la parte superior de cada unidad óptica utilizada para evitar que los rayos solares incidan sobre estas, sus dimensiones son de 10 cm de largo y 4 cm de ancho.

- **Lente:** El lente del semáforo es la parte de la unidad óptica que dirige la luz proveniente de la lámpara en la dirección deseada, se utilizo una lamina de acetato para construir el lente correspondiente a cada unidad óptica para el protótipo de semáforo inteligente.

- **Separador:** El separador del semáforo sirve para proteccion de la placa eletrónica ubicada en la parte inferior con dimensiones de 20 cm de largo, 8.5 cm de ancho, además cuenta con un orifício de diámetro de 3 cm para atravessar el cable conductor calibre 14 utilizado para la conexión de la placa electrónica con las diferentes lamparas del protótipo de semáforo inteligente como se ve en la Figura 41.

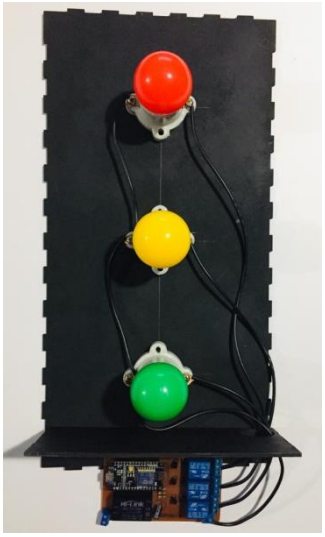


Figura 41: Separador de la estructura física del prototipo de semáforo inteligente.

Elaborado por: El investigador.

Finalmente en la Figura 42, se puede observar el diseño de la estrutura física del protótipo de semáforo inteligente A y B construído para el sistema de control de trafico vehicular:

Figura 42: Diseño de la estructura física del prototipo de semáforo inteligente A y B.

Elaborado por: El investigador.

3.17 Pruebas de funcionamiento del sistema

Para realizar las pruebas de funcionamiento del sistema de control de tráfico vehicular aplicando la arquitectura fog computing se realiza el esquema de red implementado el cual permite identificar las direcciones IP y los medios de comunicación de cada componente de la arquitectura como se ve en la Figura 43:

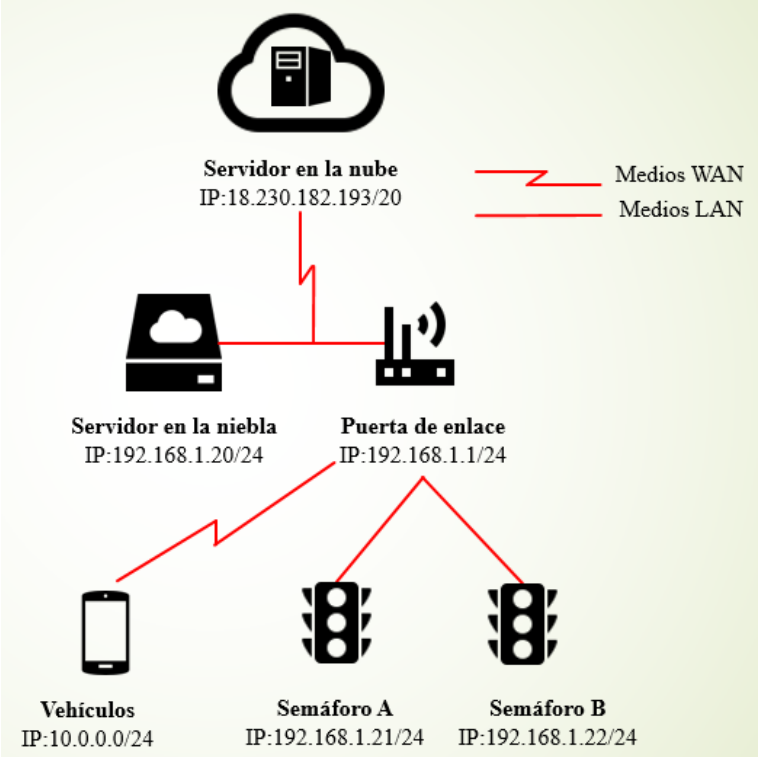


Figura 43: Esquema de red del sistema de control de tráfico vehicular aplicando la arquitectura fog computing.

Elaborado por: El investigador.

El medio de comunicación del servidor en la niebla el cual tiene dirección IP 192.168.1.20/24 con el servidor en la nube con dirección IP 18.230.182.193/20 se realiza en una red de área amplia WAN de igual manera con los vehículos conectados al sistema, mientras que el medio de comunicación de los semáforos A y B con el servidor en la niebla se realiza en una red de área local o LAN a través de la puerta de enlace 192.168.1.1/24.

La Tabla 12, resume los tópicos del sistema de control de tráfico vehicular implementado para los diferentes componentes de la arquitectura fog computing especificando la calidad de servicio Qos que se utiliza para cada tópico.

Tabla 12: Tabla de resumen de los tópicos del sistema de control de tráfico vehicular aplicando la arquitectura fog computing.

Componente	Tópico de publicación	Qos	Tópico de suscripción	Qos
Servidor en la niebla	Av.Cevallos/sent.Oeste Av.Cevallos/sent.Este Calle/EugenioEspejo Semaforo/A Semaforo/B	0 0 0 0 0	Placa Latitud Longitud	0 0 0
Servidor en la nube			Av.Cevallos/sent.Oeste Av.Cevallos/sent.Este Calle/EugenioEspejo Semaforo/A Semaforo/B	1 1 1 1 1
Vehículos	Placa Latitud Longitud	0 0 0	Semaforo/A Semaforo/B	0 0
Semáforo A			Semaforo/A	0
Semáforo B			Semaforo/B	0

Elaborado por: El investigador.

El servidor en la niebla tiene tópicos de publicación y suscripción que corresponden a datos para el conteo de vehículos en cada sentido de circulación, para el funcionamiento de los semáforos y tópicos de suscripción para obtener datos de los vehículos con la calidad de servicio igual a 0 para procesar los datos recibidos en tiempo real es decir el mensaje se envía una sola vez sin determinar el estado de entrega y así se evita latencias en la comunicación. El servidor en la nube tiene tópicos de

suscripción para realizar registros del conteo de vehículos y del funcionamiento de los semáforos con calidad de servicio igual a 1 ya que no es necesaria la comunicación en tiempo real pero asegura el estado de entrega. Los vehículos tienen tópicos de publicación y suscripción acerca de los datos del vehículo y del funcionamiento de los semáforos con calidad de servicio igual a 0 para de igual manera mantener una comunicación en tiempo real con el servidor en la niebla. Y los semáforos solo tienen tópicos de suscripción para el funcionamiento de los intervalos de semaforización en tiempo real es decir con calidad de servicio igual a 0.

A continuación se realiza las pruebas de funcionamiento de cada componente de la arquitectura fog computing para el sistema de control de tráfico vehicular:

3.17.1 Pruebas de funcionamiento del servidor en la niebla

Para realizar las pruebas de funcionamiento del servidor en la niebla se comprueba el algoritmo implementado para el control de tráfico vehicular a través de la interfaz web de monitoreo del sistema el cual permite la simulación de la ubicación de los vehículos en el mapa, se determina la comunicación exitosa del servidor en la niebla mediante un mensaje de estado igual a “Conectado”, los datos son obtenidos en la consola de la interfaz web el cual permite gestionar y verificar el correcto funcionamiento del algoritmo implementado para cumplir con las diferentes fases de semaforización del sistema según el número de vehículos contados en el servidor de niebla.

Para comprobar que se cumpla la fase de semaforización “Uno” o circulación normal como se ve en la Figura 44, que se ejecuta en una de las condiciones del algoritmo cuando existe un valor menor a 12 vehículos obtenidos por el método control del algoritmo en cualquier sentido de circulación ya sea en la Av. Cevallos sentido Oeste, Av. Cevallos sentido Este o Calle Eugenio Espejo.

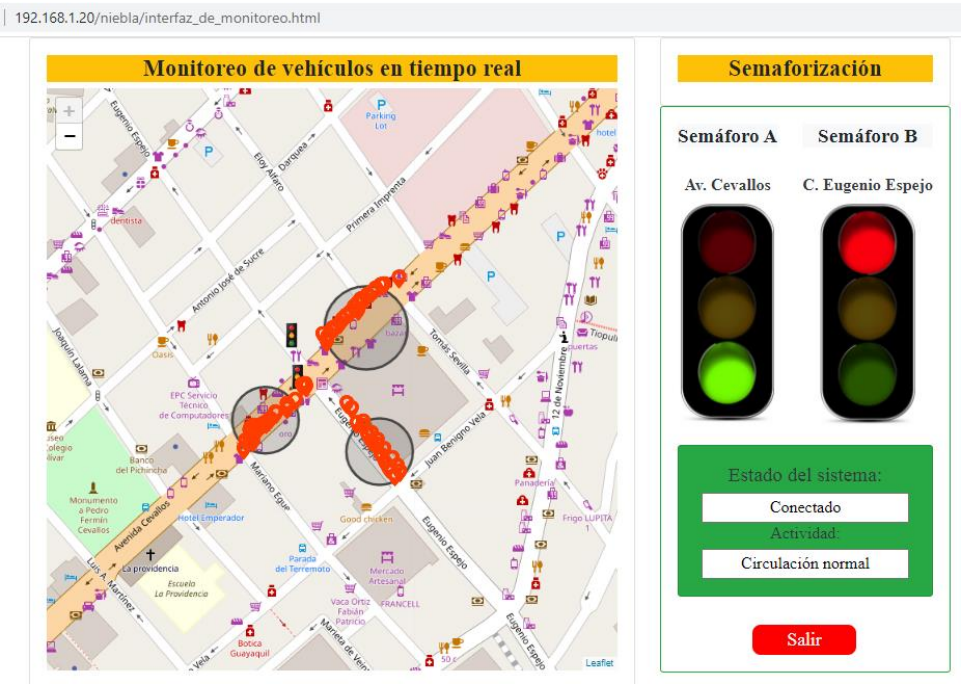


Figura 44: Verificación de la fase de semaforización uno o circulación normal mediante la interfaz web del servidor en la niebla.

Elaborado por: El investigador.

Los datos obtenidos en la consola de la interfaz web al ejecutar la fase de semaforización uno se muestran en la Tabla 13:

Tabla 13: Datos obtenidos en la consola de la interfaz web para la fase de semaforización uno.

Fase Uno	
Variables	N° de vehículos
Contador A	17
Contador B	15
Contador C	14
RA	5
RB	3
RC	2

Elaborado por: El investigador.

Las variables para los contadores A, B y C corresponden al número de vehículos contados respectivamente en la Av. Cevallos sentido Oeste, Av. Cevallos sentido Este y Calle Eugenio Espejo igual a 17, 15 y 14 vehículos, los cuales son procesadas mediante el método de control que permite determinar la fase de semaforización uno para las variables RA igual a 5 vehículos, RB igual a 3 vehículos y RC igual a 2 vehículos, cuyos valores son menores a 12 para verificar esta fase de semaforización.

Para comprobar que se cumpla la fase de semaforización “Dos” o descongestión de la Av. Cevallos como se ve en la Figura 45, el cual se ejecuta en una de las condiciones del algoritmo cuando el valor obtenido por el método de control es mayor a 12 vehículos en la Av. Cevallos sentido Oeste, Av. Cevallos sentido Este, y además cuando el valor obtenido de igual manera por el método de control es menor a 12 vehículos en la Calle Eugenio Espejo.

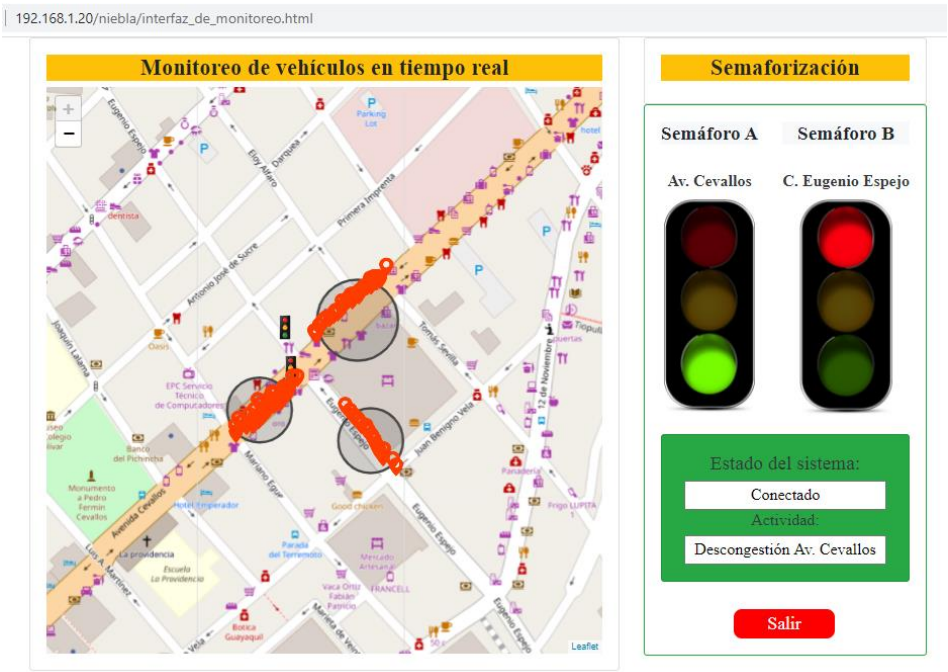


Figura 45: Verificación de la fase de semaforización dos o descongestión de la Av. Cevallos mediante la interfaz web del servidor en la niebla.

Elaborado por: El investigador.

Los datos obtenidos en la consola de la interfaz web al ejecutar la fase de semaforización dos se muestran en la Tabla 14:

Tabla 14: Datos obtenidos en la consola de la interfaz web para la fase de semaforización dos.

Fase Dos	
Variables	N° de vehículos
Contador A	28
Contador B	29
Contador C	15
RA	16
RB	17
RC	3

Elaborado por: El investigador.

Las variable contador A corresponde al número de vehículos contados en la Av. Cevallos sentido Oeste igual a 28 vehículos, en la Av. Cevallos sentido Este igual a 29 vehículos para el contador B y en la Calle Eugenio Espejo el contador C igual a 15 vehículos, mediante el método control del algoritmo se obtiene las variables RA , RB y RC que permiten determinar la fase de semaforización dos para las variables RA igual a 16 vehículos, RB igual a 17 vehículos cuyos valores con mayores a 12 y RC igual a 3 vehículos cuyo valor es menor a 12 para comprobar que se cumple esta fase de semaforización.

Para comprobar que se cumple la fase de semaforización “Tres” o descongestión de la Calle Eugenio Espejo como se ve en la Figura 46, el cual se ejecuta en una de las condiciones del algoritmo cuando existe un valor mayor a 12 vehículos obtenidos por el método control en la Calle Eugenio Espejo y un valor menor a 12 vehículos en la Av. Cevallos sentido Oeste y Av. Cevallos sentido Este:

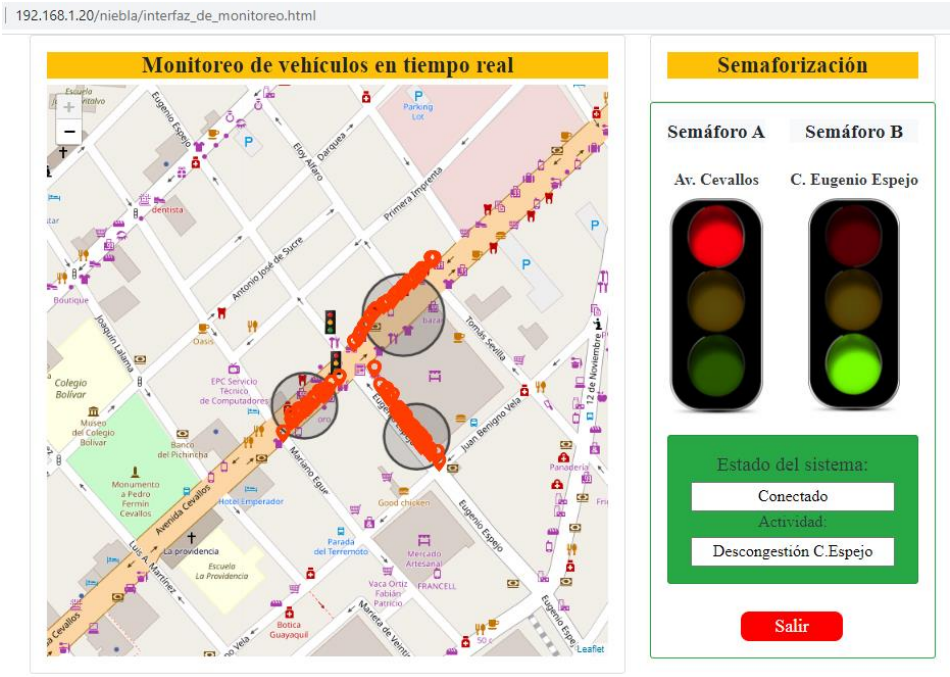


Figura 46: Verificación de la fase de semaforización tres o descongestión de la Calle Eugenio Espejo mediante la interfaz web del servidor en la niebla.

Elaborado por: El investigador.

Los datos obtenidos en la consola de la interfaz web al ejecutar la fase de semaforización tres se muestran en la Tabla 15:

Tabla 15: Datos obtenidos en la consola de la interfaz web para la fase de semaforización tres.

Fase Tres	
Variables	N° de vehículos
Contador A	17
Contador B	16
Contador C	28
RA	5
RB	4
RC	16

Elaborado por: El investigador.

Se obtuvo 17 vehículos para el contador A en la Av. Cevallos sentido Oeste, en la Av. Cevallos sentido Este igual a 16 vehículos para el contador B y en la Calle Eugenio Espejo el contador C igual a 28 vehículos, mediante el método control del algoritmo se obtiene las variables RA , RB y RC que permiten determinar la fase de semaforización tres para las variables RA igual a 5 vehículos, RB igual a 4 vehículos cuyos valores con menores a 12 y RC igual a 16 vehículos cuyo valor es mayor a 12 para comprobar que se cumple esta fase de semaforización.

3.17.2 Pruebas de funcionamiento del servidor en la nube

Para realizar las pruebas de funcionamiento del servidor en la nube se verifica el registro de históricos del sistema obtenido en la interfaz web:

El registro del coteo de vehículos obtenidos al ejecutarse la fase de semaforización uno se muestra en la Tabla 16:

Tabla 16: Registro del coteo de vehículos obtenidos al ejecutarse la fase de semaforización uno.

Fase Uno Fecha: 2021/01/04 Hora: 16:04	
Av. Cevallos sentido Oeste	17
Av. Cevallos sentido Este	15
Calle Eugenio Espejo	14

Elaborado por: El investigador.

El número de vehículos contados el día 4 de enero del 2021 a las 16 horas y 4 minutos en el cual se ejecuta la fase de sámaforización uno es igual a 17 vehículos en la Av. Cevallos sentido Oeste, 15 vehículos en la Av. Cevallos sentido Este y 14 vehículos en la Calle Eugenio Espejo como se oberva en la gráfica del conteo de vehículos de la interfaz web en la Figura 47:

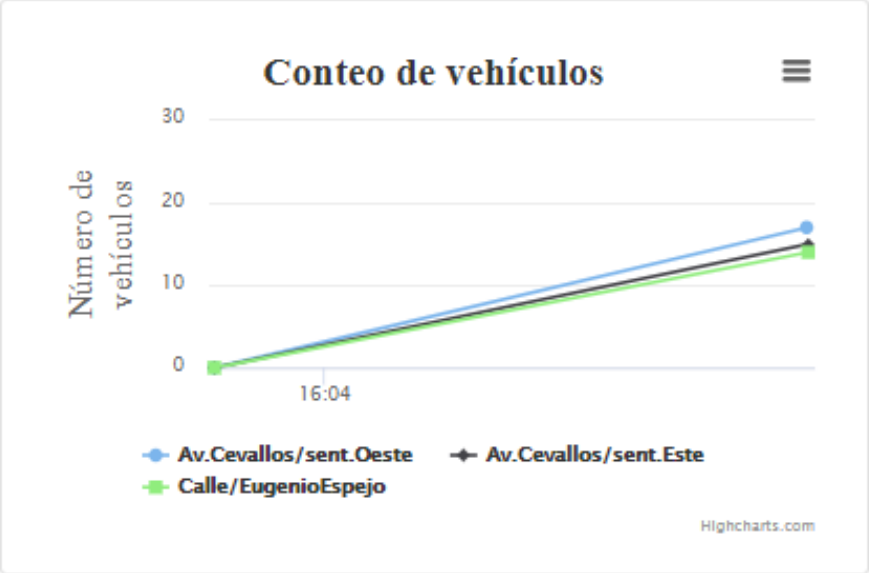


Figura 47: Registro del conteo de vehículos registrados en la fase de semaforización uno en la interfaz web**.**

Elaborado por: El investigador.

El registro del coteo de vehículos obtenidos al ejecutarse la fase de semaforización dos se muestra en la Tabla 17:

Tabla 17: Registro del coteo de vehículos obtenidos al ejecutarse la fase de semaforización dos.

Fase Dos Fecha: 2021/01/04 Hora: 16:11	
Av. Cevallos sentido Oeste	28
Av. Cevallos sentido Este	29
Calle Eugenio Espejo	15

Elaborado por: El investigador.

El número de vehículos contados el día 4 de enero del 2021 a las 16 horas y 11 minutos en el cual se ejecuta la fase de sámaforización dos es igual a 28 vehículos en la Av. Cevallos sentido Oeste, 29 vehículos en la Av. Cevallos sentido Este y 15 vehículos

en la Calle Eugenio Espejo como se oberva en la gráfica del conteo de vehículos de la interfaz web en la Figura 48:

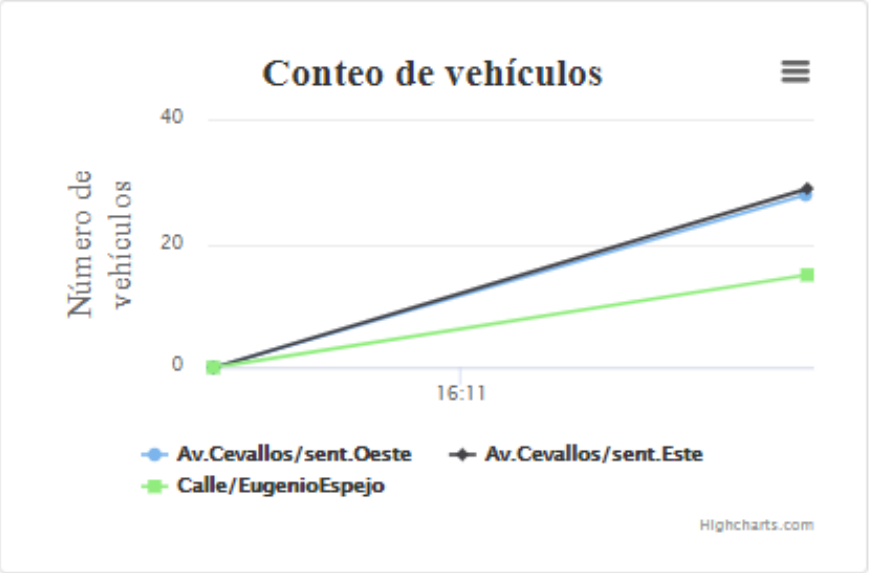


Figura 48: Registro del conteo de vehículos registrados en la fase de semaforización dos en la interfaz web.

Elaborado por: El investigador.

El registro del coteo de vehículos obtenidos al ejecutarse la fase de semaforización tres se muestra en la Tabla 18:

Tabla 18: Registro del coteo de vehículos obtenidos al ejecutarse la fase de semaforización tres.

Fase Tres Fecha: 2021/01/04 Hora: 16:16	
Av. Cevallos sentido Oeste	17
Av. Cevallos sentido Este	16
Calle Eugenio Espejo	28

Elaborado por: El investigador.

El número de vehículos contados el día 4 de enero del 2021 a las 16 horas y 16 minutos en el cual se ejecuta la fase de sámaforización tres es igual a 17 vehículos en la Av. Cevallos sentido Oeste, 16 vehículos en la Av. Cevallos sentido Este y 28 vehículos en la Calle Eugenio Espejo como se oberva en la gráfica del conteo de vehículos de la interfaz web en la Figura 49:

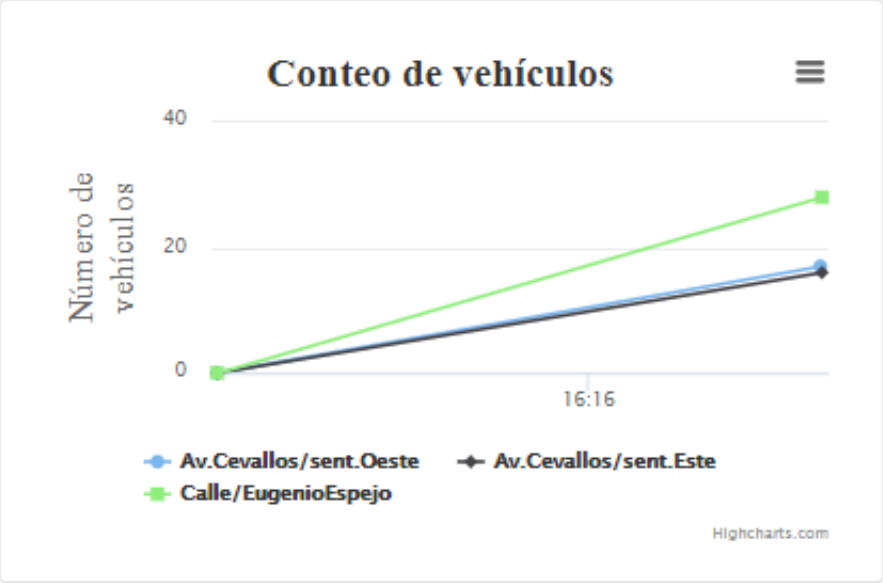


Figura 49: Conteo de vehículos registrados en la fase de semaforización tres.

Elaborado por: El investigador.

En la Figura 50, se observa la consulta realizada a la base de datos del servidor en la nube para generar la gráfica que representa el resumen del conteo de vehículos registrados en la Av. Cevallos sentido Oeste, Av. Cevallos sentido Este y la Calle Eugenio Espejo, el 4 de enero del 2021 desde las 16 horas 4 minutos hasta las 16 horas 16 minutos en el cual se generaron las tres fases de semaforización:

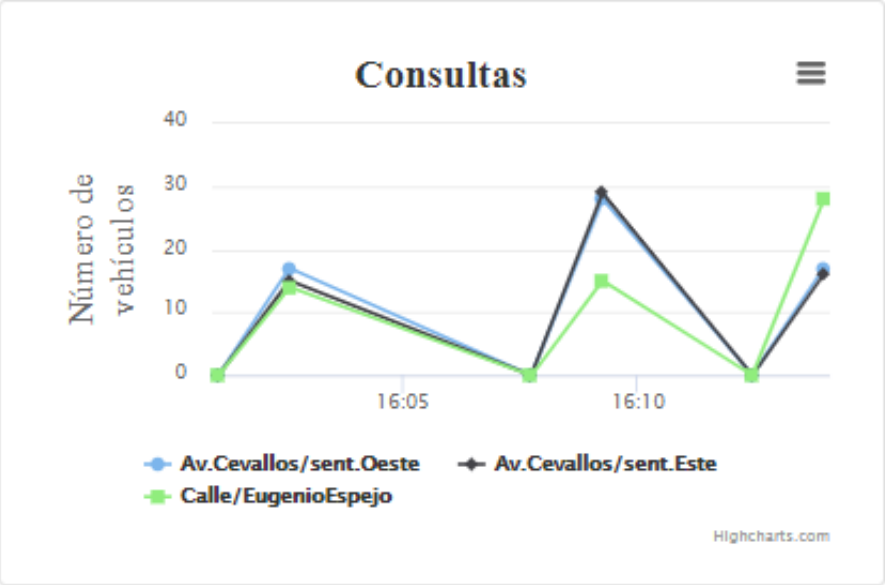


Figura 50: Conteo de vehículos obtenido mediante la consulta a la base de datos del servidor en la nube.

Elaborado por: El investigador.

Además se verifica en la interfaz web de la nube los reportes generados por el registro de históricos del sistema de control de tráfico vehicular correspondientes al 4 de enero del 2021 acerca del conteo de vehículos en la Av. Cevallos sentido Oeste, Av. Cevallos sentido Este y la Calle Eugenio Espejo y las fases de semaforización registradas tanto para el semáforo A y B, como se muestran en la Figura 51:

Reporte del conteo de vehículos

Identificador	Detalle	Cantidad	Recibido
3	Av.Cevallos/sent.Oeste	0	2021-01-04 16:01:05
4	Av.Cevallos/sent.Este	0	2021-01-04 16:01:05
5	Calle/EugenioEspejo	0	2021-01-04 16:01:05
6	Av.Cevallos/sent.Oeste	17	2021-01-04 16:02:36
7	Av.Cevallos/sent.Este	15	2021-01-04 16:02:36
8	Calle/EugenioEspejo	14	2021-01-04 16:02:36
13	Av.Cevallos/sent.Oeste	0	2021-01-04 16:07:44
14	Av.Cevallos/sent.Este	0	2021-01-04 16:07:44
15	Calle/EugenioEspejo	0	2021-01-04 16:07:44
16	Av.Cevallos/sent.Oeste	28	2021-01-04 16:09:15
17	Av.Cevallos/sent.Este	29	2021-01-04 16:09:15
18	Calle/EugenioEspejo	15	2021-01-04 16:09:15
23	Av.Cevallos/sent.Oeste	0	2021-01-04 16:12:26
24	Av.Cevallos/sent.Este	0	2021-01-04 16:12:26
25	Calle/EugenioEspejo	0	2021-01-04 16:12:26
26	Av.Cevallos/sent.Oeste	17	2021-01-04 16:13:57
27	Av.Cevallos/sent.Este	16	2021-01-04 16:13:57
28	Calle/EugenioEspejo	28	2021-01-04 16:13:57

Reporte de las fases de semaforización

Identificador	Detalle	Fase	Recibido
1	Semaforo/A	Uno	2021-01-04 16:01:05
2	Semaforo/B	Uno	2021-01-04 16:01:05
9	Semaforo/A	Uno	2021-01-04 16:02:36
10	Semaforo/B	Uno	2021-01-04 16:02:36
11	Semaforo/A	Uno	2021-01-04 16:07:44
12	Semaforo/B	Uno	2021-01-04 16:07:44
19	Semaforo/A	Dos	2021-01-04 16:09:15
20	Semaforo/B	Dos	2021-01-04 16:09:15
21	Semaforo/A	Uno	2021-01-04 16:12:25
22	Semaforo/B	Uno	2021-01-04 16:12:26
29	Semaforo/A	Tres	2021-01-04 16:13:57
30	Semaforo/B	Tres	2021-01-04 16:13:57

Figura 51: Reportes del conteo de vehículos y las fases de semaforización en la interfaz web del servidor en la nube.

Elaborado por: El investigador.

3.17.3 Pruebas de funcionamiento de la aplicación móvil

Para realizar las pruebas de funcionamiento de la aplicación móvil se instala la apk o archivo de ejecución de la app en un dispositivo móvil con el objetivo de verificar los requerimientos de la aplicación diseñada para el sistema de control de tráfico vehicular como son obtener la ubicación del dispositivo móvil ubicado en el vehículo y comunicar con el servidor en la niebla.

En la Figura 52, se observa una captura de pantalla de la aplicación móvil en funcionamiento en la que se muestra la ubicación obtenida mediante el sensor GPS del dispositivo para el vehículo de placa HBA6900 con latitud -1.2386839 y longitud -78.619304, además se verifica el correcto funcionamiento de las luces de

semaforización en la aplicación, la Av. Cevallos se encuentra en luz verde para el semáforo A y la Calle Eugenio Espejo en luz roja para el semáforo B.

Figura 52: Captura de pantalla de la aplicación móvil en funcionamiento.

Elaborado por: El investigador.

Mediante la consola de la interfaz web de monitoreo del servidor de niebla se verifica los datos enviados por la aplicación móvil con un tiempo de actualización de 5 segundos, obteniendo en la línea nueve de la consola los mismos valores enviados por la aplicación: placa HBA6900, latitud -1.2386839 y longitud -78.619304 y señalando por medio de un marcador en el mapa como se ve en la Figura 53, de igual manera se comprueba el funcionamiento de las luces de semaforización, semáforo A en luz verde y semáforo B en luz roja.

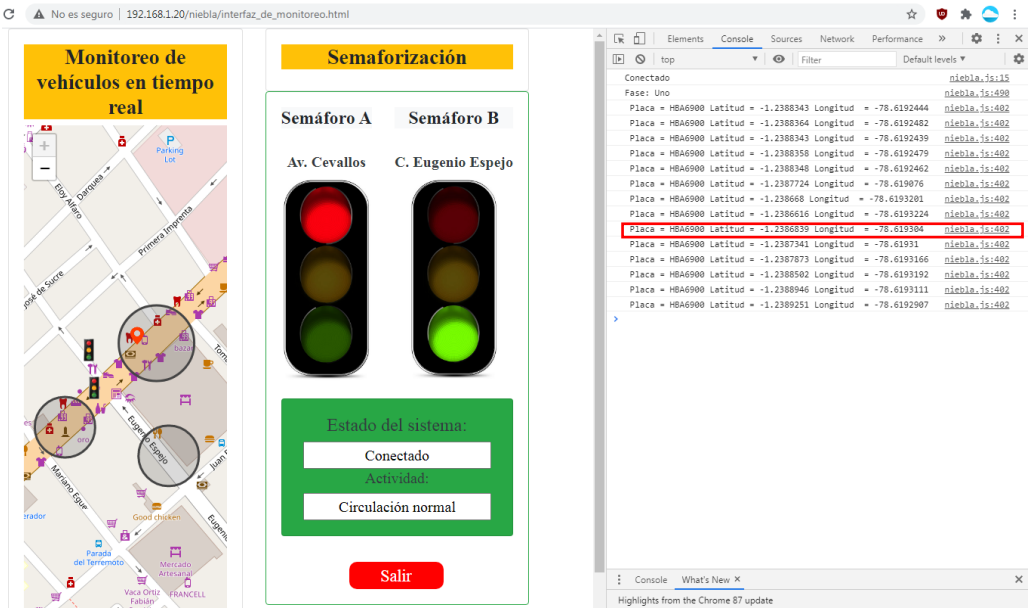


Figura 53: Verificación de los datos enviados por la aplicación móvil en la interfaz web de monitoreo del sistema.

Elaborado por: El investigador.

3.17.4 Pruebas de funcionamiento de los semáforos inteligentes

Para verificar el funcionamiento de los prototipos de semáforos inteligentes A y B al estar al estar conectados al sistema de control de tráfico vehicular, se comprueba los intervalos de tiempo que tienen las diferentes fases de semaforización como se ve en la Figura 54:

Figura 54: Pruebas de funcionamiento de los semáforos inteligentes A y B conectados al sistema de control de tráfico vehicular.

Elaborado por: El investigador.

Los datos obtenidos al ejecutarse la fase de semaforización “Uno” en los semáforos A y B se muestran en la Tabla 19:

Tabla 19: Datos obtenidos de los intervalos de tiempo para la fase de semaforización uno.

Fase Uno			
Intervalo de tiempo (s)	Rojo	Amarillo	Verde
Semáforo A	45	4	41
Semáforo B	45	4	41

Elaborado por: El investigador.

Los intervalos de tiempo obtenidos para el semáforo A y B en la fase de semaforización uno o circulación normal son 45 segundos en luz roja, 4 segundos en luz amarilla y 41 segundos en luz verde verificando que se cumple el ciclo de semaforización de 90 segundos y el tiempo definido anteriormente para la fase de semaforización uno.

Los datos obtenidos al ejecutarse la fase de semaforización “Dos” en los semáforos A y B se muestran en la Tabla 20:

Tabla 20: Datos obtenidos de los intervalos de tiempo para la fase de semaforización dos.

Fase Dos			
Intervalo de tiempo (s)	Rojo	Amarillo	Verde
Semáforo A	16	4	70
Semáforo B	74	4	12

Elaborado por: El investigador.

Los intervalos de tiempo obtenidos para el semáforo A en la fase de semaforización dos o descongestión de la Av. Cevallos son 16 segundos en luz roja, 4 segundos en luz amarilla y 70 segundos en luz verde verificando que se cumple el ciclo de

semaforización de 90 segundos, el tiempo definido anteriormente para la fase de semaforización dos y el funcionamiento contrario de las luces de semaforización del semáforo B con respecto al semáforo A.

Los datos obtenidos al ejecutarse la fase de semaforización “Tres” en los semáforos A y B se muestran en la Tabla 21:

Tabla 21: Datos obtenidos de los intervalos de tiempo para la fase de semaforización tres.

Fase Tres			
Intervalo de tiempo (s)	Rojo	Amarillo	Verde
Semáforo A	74	4	12
Semáforo B	16	4	70

Elaborado por: El investigador.

Se obtiene los intervalos de tiempo para el semáforo B en la fase de semaforización tres o descongestión de la Calle Eugenio Espejo igual a 16 segundos en luz roja, 4 segundos en luz amarilla y 70 segundos en luz verde verificando que se cumple el ciclo de semaforización de 90 segundos, el tiempo definido anteriormente para la fase de semaforización tres y el funcionamiento contrario de las luces de semaforización del semáforo A con respecto al semáforo B.

Finalmente, se realizo pruebas de funcionamiento del prototipo de sistema de control de tráfico vehicular simulando la ubicación de los vehículos en la interfaz web de monitoreo en diferentes días y horarios durante 30 minutos, tiempo en el cual se ejecutan 20 fases de semaforización cuyo ciclo es de 90 segundos, los reportes generados por el registro de históricos del sistema correspondientes al día 09 de enero del 2021 acerca del conteo de vehículos y fases de semaforización se muestra en la Tabla 22.

Tabla 22: Datos obtenidos el día 09 de enero del 2021 desde las 10 h 12 min 6 seg, hasta las 10 h 40 min 36 seg.

ID	Fecha	Hora	Av. Cevallos sentido Oeste	Av. Cevallos sentido Este	Calle Eugenio Espejo	Fase
1	09/01/2021	10:12:06	25	18	24	Uno
2	09/01/2021	10:13:36	15	26	25	Uno
3	09/01/2021	10:15:06	26	15	13	Dos
4	09/01/2021	10:16:36	34	35	13	Dos
5	09/01/2021	10:18:06	30	28	19	Tres
6	09/01/2021	10:19:36	15	13	30	Uno
7	09/01/2021	10:21:06	20	16	25	Tres
8	09/01/2021	10:22:36	17	18	35	Uno
9	09/01/2021	10:24:06	23	19	23	Uno
10	09/01/2021	10:25:36	16	25	25	Uno
11	09/01/2021	10:27:06	16	14	30	Tres
12	09/01/2021	10:28:36	11	8	35	Tres
13	09/01/2021	10:30:06	18	14	29	Uno
14	09/01/2021	10:31:36	23	28	24	Uno
15	09/01/2021	10:33:06	18	28	15	Dos
16	09/01/2021	10:34:36	34	14	18	Uno
17	09/01/2021	10:36:06	20	26	24	Uno
18	09/01/2021	10:37:36	13	12	10	Uno
19	09/01/2021	10:39:06	25	25	27	Uno
20	09/01/2021	10:40:36	9	13	26	Tres

Elaborado por: El investigador.

El datos obtenidos el día 09 de enero del 2021 desde las 10 h 12 min 6 seg hasta las 10 h 40 min 36 seg, dan como resultado un promedio de vehículos contados igual a 20 en la Av. Cevallos sentido Oeste, 20 en la Av. Cevallos sentido Este y 24 en la Calle Eugenio Espejo, mientras que el número de veces que se ejecutaron las fases de semaforización son 12 la para la fase uno, 3 para la fase dos y 5 para la fase de semaforización tres obteniendo un total de 20 fases de semaforización, con un tiempo para cada fase de 1 min 30 degundos y 30 minutos totales en el cual se realizó las pruebas de funcionamiento.

Los reportes generados por el registro de históricos del sistema correspondiente a las pruebas de fucionamiento del día 10 de enero del 2021 acerca del conteo de vehículos y fases de semaforización se muestran a continuación en la Tabla 23:

Tabla 23: Datos obtenidos el día 10 de enero del 2021 desde las 15 h 15 min 48 seg, hasta las 15 h 44 min 18 seg.

ID	Fecha	Hora	Av. Cevallos sentido Oeste	Av. Cevallos sentido Este	Calle Eugenio Espejo	Fase
1	10/01/2021	15:15:48	26	11	24	Uno
2	10/01/2021	15:17:18	10	12	25	Tres
3	10/01/2021	15:18:48	34	30	36	Uno
4	10/01/2021	15:20:18	8	26	26	Uno
5	10/01/2021	15:21:48	13	15	9	Uno
6	10/01/2021	15:23:18	11	13	25	Tres
7	10/01/2021	15:24:48	12	9	35	Tres
8	10/01/2021	15:26:18	34	30	36	Uno
9	10/01/2021	15:27:48	8	26	26	Uno
10	10/01/2021	15:29:18	13	15	9	Uno
11	10/01/2021	15:30:48	11	13	25	Tres
12	10/01/2021	15:32:18	12	9	35	Tres
13	10/01/2021	15:33:48	10	5	8	Uno
14	10/01/2021	15:35:18	11	10	26	Tres
15	10/01/2021	15:36:48	12	8	35	Tres
16	10/01/2021	15:38:18	11	18	36	Uno
17	10/01/2021	15:39:48	5	10	28	Tres
18	10/01/2021	15:41:18	6	6	36	Tres
19	10/01/2021	15:42:48	14	8	36	Tres
20	10/01/2021	15:44:18	12	18	28	Dos

Elaborado por: El investigador.

Los datos obtenidos el día 10 de enero del 2021 desde las 15 h 15 min 48 seg hasta las 15 h 44 min 18 seg, dan como resultado un promedio de vehículos contados igual a 14 en la Av. Cevallos sentido Oeste, 15 en la Av. Cevallos sentido Este y 27 en la Calle Eugenio Espejo, mientras que el número de veces que se ejecutaron las fases de

semaforización son 9 la para la fase uno, 1 para la fase dos y 10 para la fase de semaforización tres obteniendo un total de 20 fases de semaforización, con un tiempo para cada fase de 1 min 30 degundos y 30 minutos totales en el cual se realizó las pruebas de funcionamiento.

Finalmente se realiza la Tabla 24 en donde se muestra los reportes generados por el registro de históricos del sistema correspondiente a las pruebas de funcionamiento realizadas el día 11 de enero del 2021:

Tabla 24: Datos obtenidos el día 11 de enero del 2021 desde las 20 h 12 min 38 seg, hasta las 20 h 41 min 8 seg.

ID	Fecha	Hora	Av. Cevallos sentido Oeste	Av. Cevallos sentido Este	Calle Eugenio Espejo	Fase
1	11/01/2021	20:12:38	15	26	25	Uno
2	11/01/2021	20:14:08	26	15	13	Dos
3	11/01/2021	20:15:38	34	35	13	Dos
4	11/01/2021	20:17:08	34	14	18	Uno
5	11/01/2021	20:18:38	20	26	24	Uno
6	11/01/2021	20:20:08	13	12	10	Uno
7	11/01/2021	20:21:38	25	25	27	Uno
8	11/01/2021	20:23:08	11	10	26	Tres
9	11/01/2021	20:24:38	12	8	35	Tres
10	11/01/2021	20:26:08	11	18	36	Uno
11	11/01/2021	20:27:38	5	10	28	Tres
12	11/01/2021	20:29:08	6	6	36	Tres
13	11/01/2021	20:30:38	18	14	29	Uno
14	11/01/2021	20:32:08	23	28	24	Uno
15	11/01/2021	20:33:38	18	28	15	Dos
16	11/01/2021	20:35:08	30	28	19	Tres
17	11/01/2021	20:36:38	11	18	36	Uno
18	11/01/2021	20:38:08	5	10	28	Tres
19	11/01/2021	20:39:38	34	30	36	Uno
20	11/01/2021	20:41:08	8	26	26	Uno

Elaborado por: El investigador.

Los datos obtenidos el día 11 de enero del 2021 desde las 20 h 12 min 38 seg hasta las 20 h 41 min 8 seg, dan como resultado un promedio de vehículos contados igual a 18 en la Av. Cevallos sentido Oeste, 19 en la Av. Cevallos sentido Este y 25 en la Calle Eugenio Espejo, mientras que el número de veces que se ejecutaron las fases de semaforización son 11 la para la fase uno, 3 para la fase dos y 6 para la fase de semaforización tres obteniendo un total de 20 fases de semaforización, con un tiempo para cada fase de 1 min 30 degundos y 30 minutos totales en el cual se realizó las pruebas de funcionamiento.

3.17.5 Analisis de errores obtenidos

Los errores obtenidos en la práctica se deben al procesamiento de datos del sistema mediante el lenguaje de programación JavaScript el cual para la ejecución de la interfaz web de monitoreo del sistema utiliza un solo hilo de trabajo es decir el navegador puede realizar una sola tarea o proceso a la vez, teniendo la necesidad de usar alternativas para realizar multiples procesos creando varios hilos de ejecución como son los Thread & Runnable, excepciones para llevar a cabo eventos fuera del flujo normal de instrucciones y WebWorkers para ejecutar procesos en segundo plano. Las tres alternativas existentes están limitadas por el hardware en este caso de las unidades de procesamiento de la placa Raspberry Pi que se utilizó para implementación del servidor en la niebla y de la sincronización de las tareas a ejecutarse en el flujo de control del programa para que el procesamiento de los datos no se vea afectado y la interfaz web de monitoreo del sistema de control de tráfico vehicular muestre los datos eficientemente todo el tiempo en el que se encuentra en operación.

En la Tabla 25 se observa el porcentaje de error obtenido en los diferentes días en el cual se realizó las pruebas de funcionamiento, calculado en base al retardo en segundos (s) que tienen los intervalos de tiempo de semaforización en la interfaz de monitoreo del sistema en las tres fases establecidas.

Tabla 25: Porcentaje de errores obtenidos en los intervalos de tiempo de semaforización de la interfaz de monitoreo del sistema.

Fecha	Fase Uno (s)			Fase Dos (s)			Fase Tres (s)			Porcentaje (%)
	R	A	V	R	A	V	R	A	V	
09/01/2021	0	0	0.5	0.5	0.5	0	0.5	0	1	33
10/01/2021	0	0	0.5	0	0.5	0.5	0	0	0.5	22
11/01/2021	0	0.5	1	0	1	0	0	0.5	0.5	38

Elaborado por: El investigador.

Para calcular el porcentaje de error en los diferentes días se determino que el valor máximo de retardo que tienen los intervalos de tiempo de semaforización ya sea en luz roja (R), amarilla (A) o verde (V) es de 1 segundo en la interfaz de monitoreo del sistema igual a 9 segundos o 100% para las tres fases de semaforización establecidas en el caso de presentar errores, entonces se calcula el porcentaje de error obtenido para el día 9 de enero del 2021 igual a 33% equivalente a 3 segundos de retardo, para el diá 10 de enero un porcentaje de 22% igual a 2 segundos de retardo y para el 11 de enero un 38% igual a 3.5 segundos de retardo.

3.18 Comparación de los parámetros técnicos de la arquitectura fog con cloud computing

Para comparar los parámetros técnicos de la arquitectura fog computing con cloud computing se realizó el diagnóstico del servidor en la niebla (fog) y al servidor en la nube (cloud) ya implementados para determinar los valores de latencia y ancho de banda, además se analiza las características de almacenamiento y procesamiento que poseen.

Para medir la latencia se utiliza la herramienta de diagnóstico de redes denominada ping que sirve para obtener el tiempo en milisegundos que tarda en comunicarse la conexión local con un servidor.

En la Figura 55 se muestra el ping realizado al servidor en la niebla mediante el envío de paquetes ICMP de solicitud y de respuesta a la dirección ip 192.168.1.20:

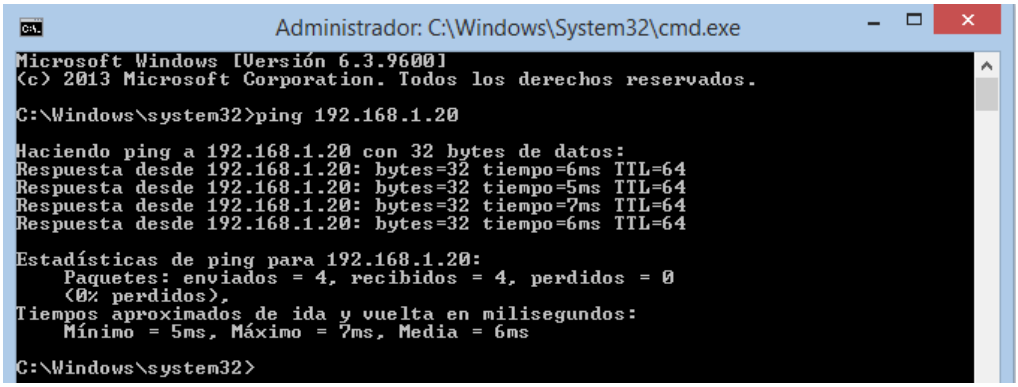


Figura 55: Ping realizado al servidor en la niebla.

Elaborado por: El investigador.

El ping realizado al servidor en la nube se observa en la Figura 56, mediante el envío de paquetes ICMP a la dirección ip 18.230.182.193:

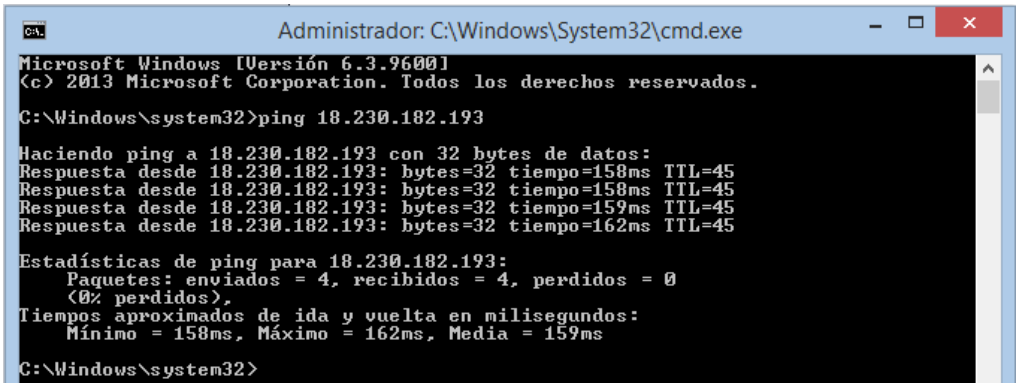


Figura 56: Ping realizado al servidor en la nube.

Elaborado por: El investigador.

Los datos obtenidos de latencia para el servidor en la niebla y nube se muestran en la Tabla 26:

Tabla 26: Medidas de latencia de los servidores en la niebla y nube.

Latencia (ms)			
Servidor	Mínimo	Máximo	Media
Niebla	5	7	6
Nube	158	162	159

Elaborado por: El investigador.

La nube se ve afectada por la latencia ya que el paquete de datos recorre mayor cantidad de distancia hasta llegar a los servidores de AWS ubicados específicamente en Sao Paulo - Brasil en comparación con la niebla el cual es un servidor local ubicado en la ciudad de Ambato - Ecuador, los niveles de latencia obtenidos en la niebla permiten utilizar servicios con calidad de servicio avanzados en tiempo real ya que cumple con la condición de tener latencia menor a 100 ms, mientras que los servicios en la nube no cumple con esta condición permitiendo solo utilizar servicios de nivel intermedio es decir aplicaciones que no requieren el procesamiento de datos en tiempo real con latencia de entre 159 y 100 ms.

Para medir el ancho de banda se utilizó la herramienta speedtest el cual mide la velocidad de conexión de subida y bajada en Mbit/s del servidor a diagnosticar con otro servidor de prueba.

En la Figura 57 se observa el speedtest realizado mediante la línea de comando speedtest-cli en servidor de niebla:

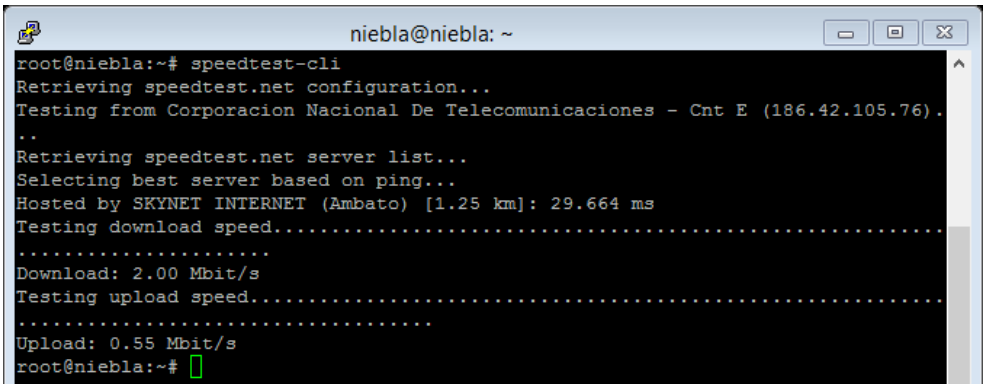


Figura 57: Speedtest realizado al servidor en la niebla.

Elaborado por: El investigador.

El speedtest realizado al servidor en la nube se observa en la Figura 58, realizado mediante la línea de comando speedtest-cli:

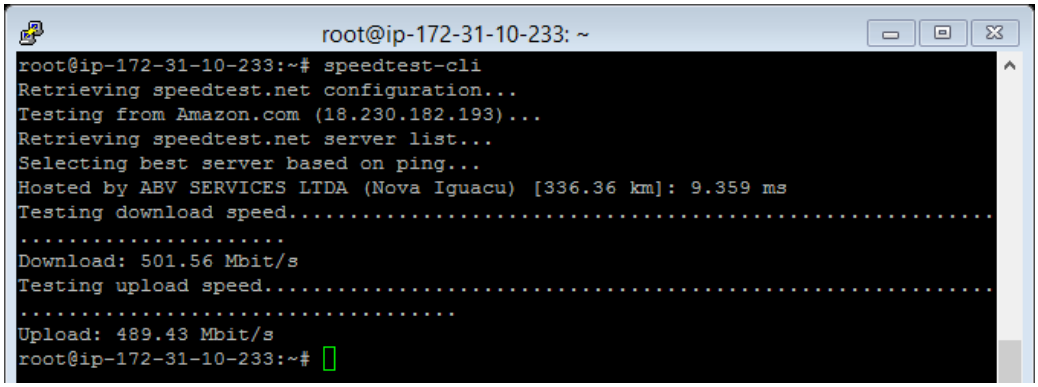


Figura 58: Speedtest realizado al servidor en la nube.

Elaborado por: El investigador.

Los datos obtenidos de ancho de banda para el servidor en la niebla y nube se muestran en la Tabla 27:

Tabla 27: Medidas de ancho de banda de los servidores en la niebla y nube.

Ancho de banda (Mbit/s)		
Servidor	Bajada	Subida
Niebla	2.00	0.55
Nube	501.56	489.43

Elaborado por: El investigador.

La niebla presenta un bajo rendimiento respecto al ancho de banda ya que la conexión depende del proveedor y el plan de internet contratando localmente en este caso de 2.00 Mbit/s de bajada y 0.55 Mbit/s de subida, mientras que la nube mejora el rendimiento con valores de conexión de subida y bajada muy altos de 501.56 Mbit/s y 489.43 Mbit/s ya que Aws tiene una mejor infraestructura de comunicaciones el cual permite a los servicios dotar de un mayor tráfico de datos.

La capacidad de almacenamiento depende de las unidades de almacenamiento físico para el caso del servidor en la niebla o unidades de almacenamiento virtual para el servidor en la niebla, mientras que el procesamiento dependen del microcontrolador y memoria RAM que tiene cada servidor, para la nube se contrató una instancia de AWS tipo t2.micro, mientras que para la niebla se adquirió una placa Raspberry Pi modelo 3 B, en la Tabla 28 se observa los valores de almacenamiento, CPU y memoria RAM para los la niebla y nube implementados:

Tabla 28: Características de almacenamiento y procesamiento del de los servidores en la niebla y nube.

Servidor	Almacenamiento	Procesador	Memoria RAM
Niebla	30 GB	ARM Cortex A-53	1 GB
Nube	30 GB	Intel Xeon	1 GB

Elaborado por: El investigador.

Los datos en la Tabla 28 muestran que el almacenamiento tanto de la niebla y la nube son iguales ya que dependen del plan contratado en el caso de la nube de aws permite utilizar unidades de almacenamiento virtual máximo de 30 GB en el plan gratuito, mientras que para la niebla el almacenamiento físico depende del tamaño de la tarjeta SD adquirida que es igual a 30 GB, los procesadores utilizados son el ARM Cortex A-53 con 4 núcleos y el Intel Xeon con 1 núcleo en el nivel gratuito y de igual memoria RAM, teniendo una mayor capacidad de procesamiento en la niebla ya que al aumentar el número de núcleos se tiene mayor unidades de procesamiento.

3.19 Presupuesto

A continuación se detalla el costo que tiene el diseño e implementación del sistema tomando en cuenta el costo de los implementos necesarios para la arquitectura fog computing y construir los prototipos de semáforos inteligentes.

Los cálculos realizados para obtener el costo del dieño del sistema se muestra en la Tabla 29:

Tabla 29: Costo del diseño del sistema de control de tráfico vehicular.

Valor Hora	Horas Empleadas	Valor del diseño
2.52 $	180	453,60

Elaborado por: El investigador.

Para calcular el costo del diseño del sistema primero se obtiene el valor de la hora de trabajo utilizando el salario mensual de un Ingeniero en Electrónica y Comunicaciones en el Ecuador según el Ministerio de Trabajo, tomando en cuenta las horas y días laborales en la Ecuación 3:

$$Valor\ Hora = \frac{Salario\ Mensual}{Horas\ Laborables\ \times\ Días\ Laborables} \quad (\boldsymbol{Ecuación\ 3})$$

$$Valor\ Hora = \frac{424}{8\ \times\ 21}$$

$$Valor\ Hora = \ 2{,}52\ \$$$

Con el valor calculado de la hora de trabajo igual a 2,52 $, se obtiene el costo del diseño tomando en cuenta las horas empleadas para la realización del proyecto mediante la Ecuación 4:

$$Valor\ Diseño = \ Valor\ Hora\ \times Horas\ Empleadas \quad (\boldsymbol{Ecuación\ 4})$$

$$Valor\ Diseño = \ 2{,}52\ \times 180$$

$$Valor\ Diseño = \ 453{,}60\ \$$$

El costo del diseño igual a 453,60 $ cuyo valor se suma al subtotal que corresponde a los costos de los implementos necesarios para la arquitectura fog computing y construir

los prototipos de semáforos inteligentes igual a 444,60 $, dando como total el valor de 898,20 $ como se puede ver en la Tabla 30.

Tabla 30: Presupuesto de la implementación del sistema de control de tráfico vehicular.

Nº	Implemento	Unidad	Cantidad	Precio Unitario $	Precio Total $
1	Raspberry Pi 3 B	c/u	1	80,00	80,00
2	Router Huawei Hg532c	c/u	1	50,00	50,00
3	Dispositivo Móvil	c/u	1	120,00	120,00
3	ESP8266 NodeMCU	c/u	2	15,00	30,00
4	Convertidor AC DC HKLP1	c/u	2	6,00	12,00
5	Relé SRD-06VDC-SL-C	c/u	6	1,00	6,00
6	Transistor 2N3904	c/u	6	0,25	1,50
7	Diodo 1N4004	c/u	6	0,50	3,00
8	Capacitor	c/u	2	1,00	2,00
9	Resistencia	c/u	6	0,10	0,60
10	Bornera	c/u	10	0,50	5,00
11	Zocalo	c/u	2	2,00	4,00
12	Elaboración de la placa	c/u	2	15,00	30,00
13	Lámpara	c/u	6	2,50	15,00
14	Enchufe	c/u	2	0,75	1,50
15	Cable conductor	c/u	8	0,50	4,00
16	Estructura del semáforo	c/u	2	40,00	80,00
Subtotal					444,60
Diseño del sistema					453,60
TOTAL					898,20

Elaborado por: El investigador.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La arquitectura fog computing permite implementar sistemas que requieren del procesamiento de información en tiempo real, ya que se encuentra ubicado muy cerca de donde se originan los datos, se lleva a cabo mediante tecnologías de comunicación inalámbricas muy utilizadas actualmente como es el estándar WIFI y redes de telefonía móvil celular 4G LTE utilizando protocolos de mensajería simple y ligera como son MQTT y WebSocket compatibles con dispositivos IoT, hardware de desarrollo como la Raspberry Pi y servicios en la nube de AWS, con la ventaja de utilizar software de código abierto.

- La implementación del algoritmo de programación diseñado en la niebla para el prototipo del sistema de control de tráfico vehicular para la intersección ubicada en la Avenida Cevallos y Calle Eugenio Espejo, cuenta los vehículos en los tres sentidos de circulación y realiza operaciones aritméticas para determinar la fase de semaforización que se debe ejecutar en los prototipos de semáforos inteligentes, verificando los resultados mediante las interfaz web de monitoreo y en la nube obteniendo solo los registros históricos del sistema.

- Al comparar los parámetros técnicos de la arquitectura fog con cloud computing mediante los servidores instalados en el sistema se determina que el fog tiene mayor ventaja con respecto al cloud ya que presenta menor latencia al estar más cerca a la fuente de información con un rendimiento del 94 % en comparación al cloud el cual tiene el 41 %, además el fog posee mayor capacidad de procesamiento por las mejores características del hardware adquirido, mientras que la nube tiene mejores cualidades respecto al ancho de banda al ser AWS una empresa mundial con una mejor infraestructura de comunicaciones.

4.2 Recomendaciones

- Para realizar el control de tráfico vehicular aplicando la computación en la niebla es necesario utilizar hadware con mayores unidades de procesamiento e infraestructuras de comunicaciones adecuadas, para el caso futuro de un sistema funcional completo implementado en la Avenida Cevallos y Calle Eugenio Espejo, en el cual se necesita procesar una gran cantidad de información.

- Para investigaciones futuras se recomienda combinar el algoritmo diseñado para el control de tráfico vehicular con modelos de programación que permitan aumentar la eficiencia del sistema, cómo por ejemplo modelos de algoritmos que ya se utilizan actualmente en diferentes servicios a través de redes neuronales, inteligencia artificial o aprendizaje profundo.

- Además es necesario implementar la arquitectura fog computing mediante servidores ubicados lo mas cerca a la fuente de datos para evitar latencias, es decir para los servidores en la nube se debe escoger adecuadamente la plataforma que brinde computacion en la nube en la región o área geográfica en donde se va a implementar el sistema.

BIBLIOGRAFÍA

[1] C. Chávez, «Sistema de semaforización inteligente para el control de flujo vehicular mediante,» 2015. [En línea]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/13061.

[2] OpenFog, «OpenFog Reference Architecturefor Fog Computing,» 2017. [En línea]. Available: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf.

[3] A. Bhardwaj y S. Goundar, «IoT enabled Smart Fog Computing for Vehicular Traffic Control,» 2019. [En línea]. Available: https://eudl.eu/doi/10.4108/eai.31-10-2018.162221.

[4] J. Pérez, «Análisis de tráfico vehicular mediante visión artificial,» 2019. [En línea]. Available: https://repositorio.uta.edu.ec/jspui/handle/123456789/29182.

[5] E. Jalew, «Fog Computing based Traffic Safety for Connected Vulnerable Road Users,» Bourgogne, 2019.

[6] M. Chiang and T. Zhang, «Fog and IoT: An Overview of Research Opportunities,» *IEEE,* June 2016.

[7] Cisco, «Cisco Annual Internet Report (2018-2023),» March 2020.

[8] Cisco, «El valor de IoT: cómo pasar de conectar cosas a obtener información,» 2015.

[9] «Fog computing: Nuevo paradigma para las nubes del IoT,» IONOS, 5 Diciembre 2019. [En línea]. Available: https://www.ionos.es/digitalguide/servidores/know-how/fog-computing/.

[10] N. Unidas, «Informe sobre la economía de la información,» *UNCTAD,* 2016.

[11] C. Mínguez, «IoT, sacando partido al mundo hiperconectado,» 28 Agosto 2019. [En línea]. Available: https://www.interempresas.net/TIC/Articulos/252927-IoT-sacando-partido-al-mundo-hiperconectado.html.

[12] Cisco, «Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are,» April 2015 .

[13] GADMA, «Causas de congestión vehicular,» 30 Julio 2014. [En línea]. Available: https://ambato.gob.ec/boletin-n-16-causas-de-congestion-vehicular.

[14] M. Ruiz, C. Mayorga, D. Aldaz y J. Reyes, «El costo y la percepción en la sociedad por congestión vehicular causada por el transporte público urbano en la ciudad de Ambato, Ecuador,» *Espacios,* Diciembre 2019.

[15] NOTIMEX, «Internet de las Cosas ayudaría a mejorar movilidad y cuidado de ambiente,» 5 Marzo 2019. [En línea]. Available: https://www.20minutos.com.mx/noticia/488248/0/internet-de-las-cosas-ayudaria-a-mejorar-movilidad-y-cuidado-de-ambiente/.

[16] R. Cal y J. Cárdenas, Ingeniería de Tránsito, Fundamentos y Aplicaciones, Séptima ed., México: Alfaomega, 1994.

[17] L. Nieto, J. Roque, H. Sánchez y G. Valdéz, «Sistema Móvil de Información Vial,» 2012.

[18] C. a. Mayor, Ingeniería de Tránsito, Fundamentos y Aplicaciones, México, 2000.

[19] M. Tamayo, «GUÍA DOCENTE PARA TRABAJAR LA EDUCACIÓN VIAL EN EL AULA,» MINISTRO DE EDUCACIÓN , 2019. [En línea]. Available: https://educacion.gob.ec/wp-content/uploads/downloads/2019/10/Guia-de-educacion-vial.pdf.

[20] M. Martínez, «Semáforos Inteligentes,» Asunción - Paraguay.

[21] J. Camarena, L. Contreras, K. Moreno, M. Rodríguez y C. Salazar , «Aplicaciones del IoT para el control de congestión vehicular,» Junio 2018.

[22] E. Arias, «Demostrador IoT-Cloud en tiempo real,» *Universidad Abierta de Cataluña,* Junio 2016.

[23] C. Hervas, «Análisis de rendimiento de protocolos de Publicación/Subscripción en comunicación con una Red de Sensores Inalámbricos Zigbee,» Universidad Nacional de La Plata, 2018. [En línea]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/69435/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y.

[24] J. Martinez, «CALIDAD DE SERVICIO (QoS),» Pontificia Universidad Javeriana de Cali, [En línea]. Available: http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:daysenr:daysenr_-_calidad_de_servicio_qos_.pdf.

[25] IETF , «IETF Home,» Diciembre 2011. [En línea]. Available: https://tools.ietf.org/html/rfc6455#section-1.6. [Último acceso: 2020].

[26] HiveMQ, «HiveMQ,» HiveMQ GmbH, 2015. [En línea]. Available: https://www.hivemq.com/blog/mqtt-essentials-special-mqtt-over-websockets/.

[27] I. Belcic, «Avast,» Fundación Avast, 15 Junio 2020. [En línea]. Available: https://www.avast.com/es-es/c-what-is-a-proxy-server.

[28] C. Villagómez, «Protocolo TCP,» 2017. [En línea]. Available: https://es.ccm.net/contents/281-protocolo-tcp.

[29] C. Moreno, «Telefonía móvil celular, GSM,» Universidad Rey Juan Carlos, Madrid.

[30] C. Campo y C. García, «Arquitecturas y tecnologías inalámbricas,» Universidad Carlos III de Madrid, Madrid.

[31] A. Fernández, D. González y A. Rubio, «Transmisión y redes de Datos,» de *Telefonía Móvil*, Universidad de Huelva, 2002, p. 7.

[32] J. Fernández, «4G LTE, LA NUEVA ERA DE LAS COMUNICACIONES TÁCTICAS,» ieee.es, 2015. [En línea]. Available: http://www.ieee.es/Galerias/fichero/docs_opinion/2015/DIEEEO87-2015_4G_LTE_ComunicacionesTacticas_J.AbrahamFdez.pdf.

[33] P. Florido y G. Navea, «Estudio y caracterización de la integración e implementación del estándar LTE (Long Term Evolution) a las redes existentes de la Corporación DIGITEL C.A.,» 2012. [En línea]. Available: http://biblioteca2.ucab.edu.ve/anexos/biblioteca/marc/texto/AAS7538.pdf.

[34] Thales, «Presentando la tecnología y redes 5G (definición, características, 5G vs 4G y casos de uso),» 2019. [En línea]. Available: https://www.thalesgroup.com/es/countries/americas/latin-america/dis/movil/inspiracion/5g.

[35] P. Jara y P. Nazar, «Editorial Universitaria de la Universidad Tecnológica Nacional,» [En línea]. Available: http://www.edutecne.utn.edu.ar/monografias/standard_802_11.pdf.

[36] A. M. y. G. N. E. Huerta, GPS Posicionamiento Satelital, Universidad Nacional de Rosario: UNR EDITORA, 2005.

[37] e-park, «El móvil como nuevo GPS,» 2019. [En línea]. Available: http://www.e-park.es/es/blog/el-movil-como-nuevo-gps.

[38] AristaSur, «Sistema de Coordenadas Geográficas: Longitud y Latitud,» 2010. [En línea]. Available: https://www.aristasur.com/contenido/sistema-de-coordenadas-geograficas-longitud-y-latitud.

[39] geohistoriablog., «2.4.- Coordenadas Geográficas,» [En línea]. Available: https://sites.google.com/site/geohistoriaenlaces/geografia-fisica/el-planeta-tierra/2-4---coordenadas-geograficas.

[40] IBM Cloud Education, «IBM,» 9 Mayo 2019. [En línea]. Available: https://www.ibm.com/cloud/learn/lamp-stack-explained.

[41] J. V. M. y. M. L. S. J. M. Vara Mesa, «Desarrollo web en entorno servidor,» Madrid, RA-MA, 2015, p. 234.

[42] T. T. y. S. V. Dirección de Tránsito, «Solicitud de Información,» Ambato, 2020.

[43] A. Ruiz, «Empieza a desarrollar en Android con Android Studio,» 5 Marzo 2018. [En línea]. Available: https://www.geekno.com/empieza-a-desarrollar-en-android-con-android-studio.html.

[44] «Instalación Visual Studio con Xamarin,» Software Guru, 2018. [En línea]. Available: https://sg.com.mx/buzz/instalaci-n-visual-studio-xamarin.

[45] «Instalación básica del IDE Eclipse en Windows,» PC Resumen, 2 Mayo 2019. [En línea]. Available: https://www.pcresumen.com/menu-tutoriales/28-instalacion-basica-del-ide-eclipse-en-windows.

[46] M. Mena, «Android e iOS dominan el mercado de los smartphones,» 30 Julio 2020. [En línea]. Available: https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo/.

[47] F. R. Pi, «Productos,» 8 Octubre 2020. [En línea]. Available: https://www.raspberrypi.org/products/.

[48] R. Alonso, «Las mejores alternativas a Raspberry Pi 4 que puedes comprar,» 11 Diciembre 2020. [En línea]. Available: https://hardzone.es/reportajes/listas/alternativas-raspberry-pi-4/.

[49] C. Harvey and A. Patrizio, «AWS vs. Azure vs. Google: Cloud Comparison,» 17 March 2020. [En línea]. Available: https://www.datamation.com/cloud-computing/aws-vs-azure-vs-google-cloud-comparison.html.

[50] S. Grinaker, «The cloud battle: Google vs. Microsoft vs. Amazon,» enonic, 20 November 2019. [En línea]. Available: https://enonic.com/blog/cloud-battle-google-microsoft-amazon.

[51] N. Team, «NodeMcu Connect Things EASY,» 2020. [En línea]. Available: https://www.nodemcu.com/index_en.html.

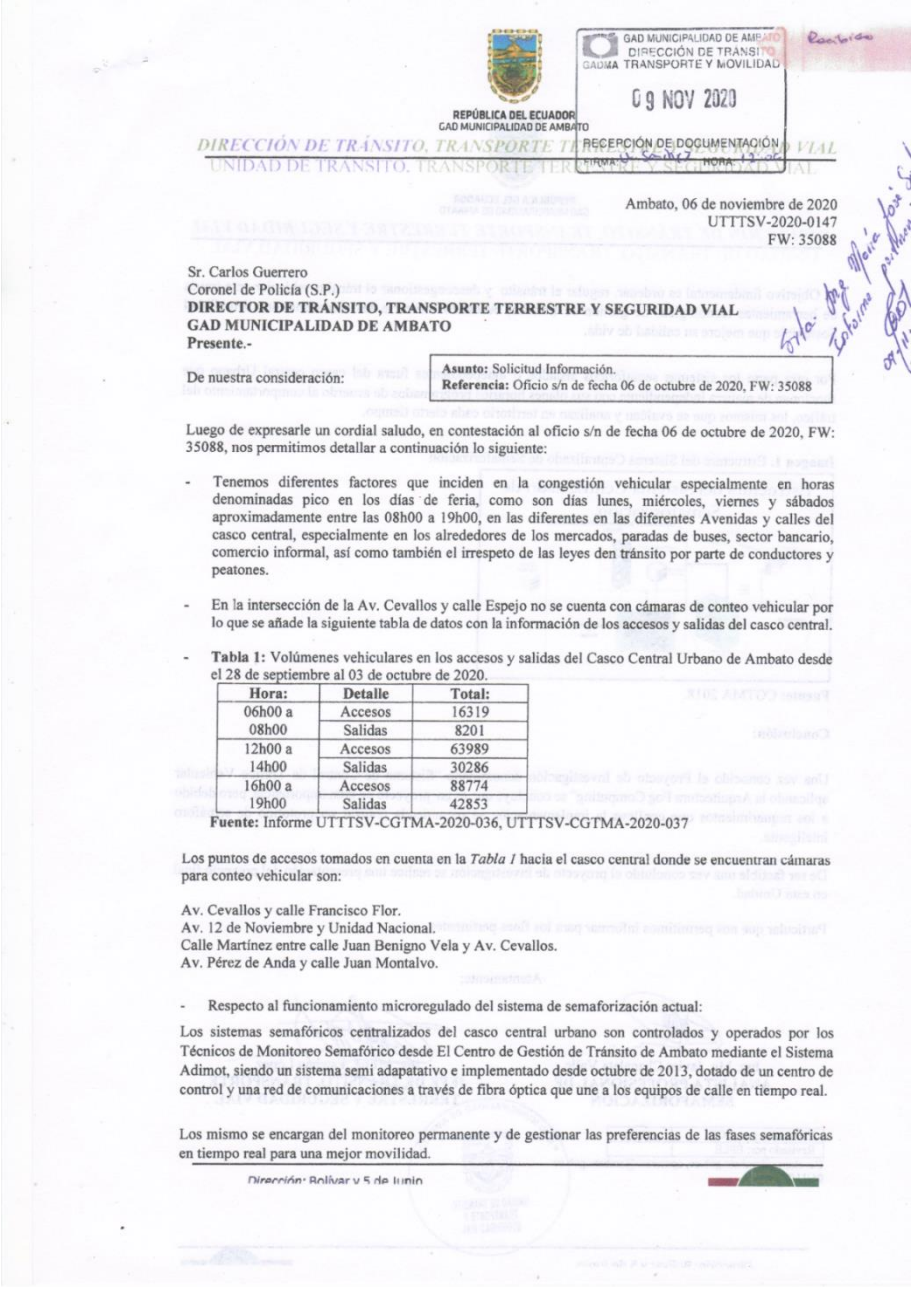
[52] M. Electronics, «Arduino Every vs.Arduino 33: Comparación de placas Arduino,» 2021. [En línea]. Available: https://arduino.cl/arduino-every-vs-arduino-33-comparacion-de-placas-arduino/.

[53] A. F. Navarrete, «Accidentes y rápido cambio de la luz amarilla. Regla de tiempos de las luces de semáforos,» 2018. [En línea]. Available: https://www.eluniverso.com/opinion/2018/12/08/nota/7087257/accidentes-rapido-cambio-luz-amarilla-regla-tiempos-luces-semaforos.

ANEXOS

ANEXO A

En la Figura 59 y Figura 60 se puede observar la información brindada por parte de la Dirección de Tránsito, Transporte Terrestre y Seguridad Vial de la ciudad de Ambato sobre la congestión vehicular y el funcionamiento del sistema de semaforización actual.



REPÚBLICA DEL ECUADOR
GAD MUNICIPALIDAD DE AMBATO

GAD MUNICIPALIDAD DE AMBATO
DIRECCIÓN DE TRÁNSITO
GADMA TRANSPORTE Y MOVILIDAD
09 NOV 2020
RECEPCIÓN DE DOCUMENTACIÓN

DIRECCIÓN DE TRÁNSITO, TRANSPORTE TERRESTRE Y SEGURIDAD VIAL
UNIDAD DE TRÁNSITO, TRANSPORTE TERRESTRE Y SEGURIDAD VIAL

Ambato, 06 de noviembre de 2020
UTTTSV-2020-0147
FW: 35088

Sr. Carlos Guerrero
Coronel de Policía (S.P.)
DIRECTOR DE TRÁNSITO, TRANSPORTE TERRESTRE Y SEGURIDAD VIAL
GAD MUNICIPALIDAD DE AMBATO
Presente.-

De nuestra consideración:

Asunto: Solicitud Información.
Referencia: Oficio s/n de fecha 06 de octubre de 2020, FW: 35088

Luego de expresarle un cordial saludo, en contestación al oficio s/n de fecha 06 de octubre de 2020, FW: 35088, nos permitimos detallar a continuación lo siguiente:

- Tenemos diferentes factores que inciden en la congestión vehicular especialmente en horas denominadas pico en los días de feria, como son días lunes, miércoles, viernes y sábados aproximadamente entre las 08h00 a 19h00, en las diferentes en las diferentes Avenidas y calles del casco central, especialmente en los alrededores de los mercados, paradas de buses, sector bancario, comercio informal, así como también el irrespeto de las leyes den tránsito por parte de conductores y peatones.

- En la intersección de la Av. Cevallos y calle Espejo no se cuenta con cámaras de conteo vehicular por lo que se añade la siguiente tabla de datos con la información de los accesos y salidas del casco central.

- **Tabla 1:** Volúmenes vehiculares en los accesos y salidas del Casco Central Urbano de Ambato desde el 28 de septiembre al 03 de octubre de 2020.

Hora:	Detalle	Total:
06h00 a	Accesos	16319
08h00	Salidas	8201
12h00 a	Accesos	63989
14h00	Salidas	30286
16h00 a	Accesos	88774
19h00	Salidas	42853

Fuente: Informe UTTTSV-CGTMA-2020-036, UTTTSV-CGTMA-2020-037

Los puntos de accesos tomados en cuenta en la *Tabla 1* hacia el casco central donde se encuentran cámaras para conteo vehicular son:

Av. Cevallos y calle Francisco Flor.
Av. 12 de Noviembre y Unidad Nacional.
Calle Martínez entre calle Juan Benigno Vela y Av. Cevallos.
Av. Pérez de Anda y calle Juan Montalvo.

- Respecto al funcionamiento microregulado del sistema de semaforización actual:

Los sistemas semafóricos centralizados del casco central urbano son controlados y operados por los Técnicos de Monitoreo Semafórico desde El Centro de Gestión de Tránsito de Ambato mediante el Sistema Adimot, siendo un sistema semi adapatativo e implementado desde octubre de 2013, dotado de un centro de control y una red de comunicaciones a través de fibra óptica que une a los equipos de calle en tiempo real.

Los mismo se encargan del monitoreo permanente y de gestionar las preferencias de las fases semafóricas en tiempo real para una mejor movilidad.

Dirección: Bolívar y 5 de Junio

Figura 59: Información sobre la congestión vehicular en la ciudad de Ambato [42].

REPÚBLICA DEL ECUADOR
GAD MUNICIPALIDAD DE AMBATO

DIRECCIÓN DE TRÁNSITO, TRANSPORTE TERRESTRE Y SEGURIDAD VIAL

UNIDAD DE TRÁNSITO, TRANSPORTE TERRESTRE Y SEGURIDAD VIAL

El Objetivo fundamental es ordenar, regular el tránsito y descongestionar el tránsito vehicular en a través de herramientas tecnológicas de gestión de tránsito, buscando ofrecer a los ciudadanos una Movilidad Sostenible que mejore su calidad de vida.

Por otra parte los sistemas semafóricos aislados o independientes fuera del casco central Urbano que funcionan de manera independiente con sus planes horarios programados de acuerdo al comportamiento del tráfico, los mismos que se evalúan y analizan en territorio cada cierto tiempo.

Imagen 1. Estructura del Sistema Centralizado de Semaforización

Fuente: CGTMA 2018.

Conclusión:

Una vez conocido el Proyecto de Investigación denominado "Sistema de Control de Tráfico Vehicular aplicando la Arquitectura Fog Computing" se concluye que es un proyecto de gran importancia, pero debido a los requerimientos que conlleva la implementación se recomienda realizar un prototipo de semáforo inteligente.

De ser factible una vez concluido el proyecto de investigación se realice una presentación del proyecto final en esta Unidad.

Particular que nos permitimos informar para los fines pertinentes.

Atentamente;

Ing. Mg. Javier Chanalata Valle
ANALISTA PROFESIONAL DE SEMAFORIZACIÓN

Ing. MEng. Esteban López
JEFE DE TRÁNSITO, TRANSPORTE TERRESTRE Y SEGURIDAD VIAL

Elaborador por: JJCH.	
Revisado por: EFLE	

Cc: elopez@ambato.gob.ec, cguerrero@ambato.gob.ec
06/11/2020

GAD MUNICIPALIDAD DE AMBATO
UNIDAD DE TRÁNSITO TRANSPORTE Y SEGURIDAD VIAL

Dirección: Bolívar y 5 de Junio

Figura 60: Información sobre el funcionamiento del sistema de semáforizacion actual en la ciudad de Ambato [42].

ANEXO B

El siguiente código fuente denominado index.html se utilizó para realizar la interfaz gráfica de inicio o autenticación para el servidor de niebla.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Monitoreo del sistema de control de tráfico vehicular</title>
<!-- Estilo css-->
<link rel="stylesheet" href="css/estilos.css">
<!-- Archivo javaScript para validar el usuario y contraseña-->
<script src="js/inicio.js"></script>
</head>
<body>
<div class="inicio">
<img src="imagenes/logo_fisei.png" class="icono">
<h1>Sistema de control de tráfico vehicular</h1>
<h2>Interfaz de Monitoreo</h2>
<form >
<!-- Usuario-->
<label for="username">Usuario</label>
<input type="text" id="username"placeholder="Ingresar Usuario">
<!-- Contraseña -->
<label for="password">Contraseña</label>
<input type="password" id="password" placeholder="Ingresar Contraseña">
<!-- Acceder -->
<input type="button" value="Acceder" onclick="validate()">
</form>
</div>
</body>
</html>
```

ANEXO C

A través del siguiente código fuente se realizó la interfaz gráfica de monitoreo del sistema para el servidor de niebla, el cual se denomina interfaz_de_monitoreo.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Monitoreo del sistema de control de tráfico vehicular</title>
<!-- Estilo css-->
<link rel="stylesheet" href="css/estilos.css">
<!-- Libreria de Bootstrap-->
<link
rel="stylesheet"href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.s
">
<!-- Librerias de JavaScript para Mqtt-->
<script src="js/mqttws31.js"></script>
<!-- Librerias de leaflet para el mapa -->
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZ
MZ19scR4PsZChSR7A=="
crossorigin=""/>
</head>
<body>
<div class="container" >
<div class="row" >
<div class="col col-lg-8">
<div class="card">
<div class="card-body">
<h3 class ="text-center bg-warning" style="font-family: Times New Roman" >
<b>Monitoreo de vehículos en tiempo real</b></h3>
```

<!-- Añadir el mapa de leaflet-->

* *

<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"

integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA=="

crossorigin=""></script>

* *

* *

* *

* *

<h3 class ="text-center bg-warning " style="font-family: Times New Roman" > Semaforización </h3>

* *

* *

* *

* *

* *

*<h4 class ="text-center bg-light" style="font-family: Times New Roman" > Semáforo A</h4>
*

<h5 class ="text-center text-dark border-warning" style="font-family: Times New Roman"> Av. Cevallos</h5>

<img src="imagenes/semaforoA.png" id='semaforoA' width="117px" class="mx-auto d-block">

* *

* *

*<h4 class ="text-center bg-light" style="font-family: Times New Roman"> Semáforo B</h4>
*

<h5 class ="text-center text-dark" style="font-family: Times New Roman"> C. Eugenio Espejo</h5>

<img src="imagenes/semaforoB.png" id='semaforoB' width="117px" class="mx-auto d-block" >

*
*

* *

* *

<h4 class ="text-center text-dark" style="font-family: Times New Roman">Estado del sistema: </h4>

<input type="text" style="font-family: Times New Roman; font-size: 15pt; text-align:center" id="estado" class="mx-auto d-block">

<h5 class ="text-center text-dark" style="font-family: Times New Roman">Actividad:</h5>

<input type="text" style="font-family: Times New Roman; font-size: 15pt; text-align:center" id="actividad" class="mx-auto d-block">

*
*

* *

<input type="button" value="Salir" style="font-family: Times New Roman" onclick="salir()">

* *

<!-- Archivo javaScript para el procesamiento de datos en la niebla-->

<script src="js/niebla.js"></script>

</body>

</html>

ANEXO D

Mediante el siguiente código fuente inicio.js se realizó la validación del usuario y cotraseña para acceder a la interfaz de monitoreo del servidor de niebla.

function validate(){

var username = document.getElementById("username").value;

var password = document.getElementById("password").value;

if (username == "Administrador" && password == "eibarra9373"){

//Redirigiendo a la siguiente página

window.location = "interfaz_de_monitoreo.html";

}

else{

alert("Porfavor ingrese, nombre de usuario y contraseña correctos.");

}

}

ANEXO E

El siguiente código fuente denominado niebla.js se utilizó para implementar el algoritmo del sistema de control de tráfico vehicular en el servidor de niebla.

```
// Declarar una variable para el servidor de niebla
var clientId = "ws" + Math.random();
// Crear una instancia para el servidor de niebla
var client = new Paho.MQTT.Client("192.168.1.20", 9001, clientId);
// Establecer controladores de devolución de llamada
client.onConnectionLost = onConnectionLost;
client.onMessageArrived = onMessageArrived;
// Conectar al cliente de niebla
client.connect({onSuccess:onConnect});
// Llamada cuando el cliente de niebla se conecta
function onConnect() {
// Mensaje de conectado en la consola una vez que se haya establecido la conexión
estado = "Conectado";
console.log(estado);
document.getElementById("estado").value = estado;
// Llamar al método FaseUno para iniciar el sistema
faseUno();
// Envio de mesajes correspondientes al conteo de vehículos
n1 = contadorA.toString();
n2 = contadorB.toString();
n3 = contadorC.toString();
message3 = new Paho.MQTT.Message(n1)
message4 = new Paho.MQTT.Message(n2)
message5 = new Paho.MQTT.Message(n3)
message3.destinationName = 'Av.Cevallos/sent.Oeste'
message4.destinationName = 'Av.Cevallos/sent.Este'
message5.destinationName = 'Calle/EugenioEspejo'
client.send(message3)
client.send(message4)
```

```
client.send(message5)
// Tópicos a suscribirse
client.subscribe("PlacaA");
client.subscribe("LatitudA");
client.subscribe("LongitudA");
client.subscribe("PlacaN");
client.subscribe("LatitudN");
client.subscribe("LongitudN");}
// Llamada cuando el cliente de niebla pierde la conexión
function onConnectionLost(responseObject) {
if (responseObject.errorCode !== 0) {
estado = "Desconectado";
console.log(estado);
console.log(estado+responseObject.errorMessage);
document.getElementById("estado").value = estado;
}}
// Mapa leaflet
const tilesProvider = 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png'
// Acercamiento del mapa en la interseccion Av. Cevallos y Eugenio Espejp
let mapa = L.map('mapa').setView([-1.240517, -78.6255646], 30)
L.tileLayer(tilesProvider, {
maxZoom: 18,
}).addTo(mapa)
// Icono para los semaforos en el mapa
let iconMarker1 = L.icon({
iconUrl: 'imagenes/semaforo.png',
iconSize: [30, 30],
iconAnchor: [15, 30]
})
// Icono para los vehículos en el mapa
let iconMarker2 = L.icon({
iconUrl: 'imagenes/marca.png',
iconSize: [24, 24],
```

```
iconAnchor: [8, 24]

})

// Marcadores ubicador en el mapa

let semaforoA = L.marker([-1.240581, -78.6257843], { icon: iconMarker1 }).bindPopup("Semáforo A").openPopup().addTo(mapa)

let semaforoB = L.marker([-1.2403109, -78.625827], { icon: iconMarker1 }).bindPopup("Semáforo B").openPopup().addTo(mapa)

// Variables Globales

var geovallaA = null;

var geovallaB = null;

var geovallaC = null;

var vehiculo;

var contadorA = 0;

var contadorB = 0;

var contadorC = 0;

var RA, RB, RC;

var tipoDeActividad = null;

var numeroDeFase = null;

var n1,n2,n3;

// Variables para el vehiculo 1

var placa1;

var latitud1;

var longitud1;

var vehiculo1 = null;

var contador1A = 0;

var contador1B = 0;

var contador1C = 0;

// Variables para el vehiculo n

var placaN;

var latitudN;

var longitudN;

var vehiculoN = null;

var contadorNA = 0;

var contadorNB = 0;
```

```
var contadorNC = 0;
// Variable de control para realizar el conteo de los vehiculos
var control = "c";
// Geovalla A
geovallaA = L.circle([-1.2401836, -78.6253437], {
color: '#454545',
fillOpacity: 0.2,
radius:30
}).addTo(mapa);
// Geovalla B
geovallaB = L.circle([-1.240779, -78.625994], {
color: '#454545',
fillOpacity: 0.2,
radius:24
}).addTo(mapa);
// Geovalla C
geovallaC = L.circle([-1.2409819, -78.6252571], {
color: '#454545',
fillOpacity: 0.2,
radius:24
}).addTo(mapa);
// Intervalo de tiempo para volver a contar el vehículo
var t1 = setInterval('reset()',91000);
// Método ingresar ubicaciones de prueba
mapa.doubleClickZoom.disable()
mapa.on('dblclick', e => {
let latLng = mapa.mouseEventToLatLng(e.originalEvent)
vehiculo = L.marker([latLng.lat, latLng.lng], { icon: iconMarker2 }).addTo(mapa)
// Llamar al método contador
metodocontadorP();
})
// Método contador para añadir ubicaciónes
function metodocontadorP(){
```

```
let pos = vehiculo.getLatLng();
let dA = mapa.distance(pos, geovallaA.getLatLng());
let radioA = geovallaA.getRadius();
if(dA <= radioA){
contadorA ++;
}
let dB = mapa.distance(pos, geovallaB.getLatLng());
let radioB = geovallaB.getRadius();
if(dB <= radioB ){
contadorB ++;
}
let dC = mapa.distance(pos, geovallaC.getLatLng());
let radioC = geovallaC.getRadius();
if(dC <= radioC ){
contadorC ++;
}}
function metodocontador1(){
let pos1 = vehiculo1.getLatLng();
let d1A = mapa.distance(pos1, geovallaA.getLatLng());
let radio1A = geovallaA.getRadius();
if(d1A <= radio1A){
contador1A++;
control = "c";
if (contador1A == 1 && control == "c"){
contadorA ++;
}}
let d1B = mapa.distance(pos1, geovallaB.getLatLng());
let radio1B = geovallaB.getRadius();
if(d1B <= radio1B){
contador1B++;
control = "c";
if (contador1B == 1 && control == "c"){
contadorB ++;
```

```
}}
let d1C = mapa.distance(pos1, geovallaC.getLatLng());
let radio1C = geovallaC.getRadius();
if(d1C <= radio1C){
contador1C++;
control = "c";
if (contador1C == 1 && control == "c"){
contadorC ++;
}}}
function metodocontadorN(){
let posN = vehiculoN.getLatLng();
let dNA = mapa.distance(posN, geovallaA.getLatLng());
let radioNA = geovallaA.getRadius();
if(dNA <= radioNA){
contadorNA++;
control = "c";
if (contadorNA == 1 && control == "c"){
contadorA ++;
}}
let dNB = mapa.distance(posN, geovallaB.getLatLng());
let radioNB = geovallaB.getRadius();
if(dNB <= radioNB){
contadorNB++;
control = "c";
if (contadorNB == 1 && control == "c"){
contadorB ++;
}}
let dNC = mapa.distance(posN, geovallaC.getLatLng());
let radioNC = geovallaC.getRadius();
if(dNC <= radioNC){
contadorNC++;
control = "c";
if (contadorNC == 1 && control == "c"){
```

```
contadorC ++;
}}}
// Método de control para la fase uno
function controlFaseUno(){
console.log("Contador A = " + contadorA + " vehículos");
console.log("Contador B = " + contadorB + " vehículos");
console.log("Contador C = " + contadorC + " vehículos");
// Envio de mesajes correspondientes al conteo de vehículos
n1 = contadorA.toString();
n2 = contadorB.toString();
n3 = contadorC.toString();
message3 = new Paho.MQTT.Message(n1)
message4 = new Paho.MQTT.Message(n2)
message5 = new Paho.MQTT.Message(n3)
message3.destinationName = 'Av.Cevallos/sent.Oeste'
message4.destinationName = 'Av.Cevallos/sent.Este'
message5.destinationName = 'Calle/EugenioEspejo'
client.send(message3)
client.send(message4)
client.send(message5)
RA = contadorA - 12;
RB = contadorB - 12;
RC = contadorC - 12;
console.log("Control: Uno ");
console.log("RA = " + RA + " vehículos");
console.log("RB = " + RB + " vehículos");
console.log("RC = " + RC + " vehículos");
condiciones();
contadorA = 0;
contadorB = 0;
contadorC = 0;
}
// Método de control para la fase dos
```

```
function controlFaseDos(){
console.log("Contador A = " + contadorA + " vehículos");
console.log("Contador B = " + contadorB + " vehículos");
console.log("Contador C = " + contadorC + " vehículos");
// Envio de mesajes correspondientes al conteo de vehículos
n1 = contadorA.toString();
n2 = contadorB.toString();
n3 = contadorC.toString();
message3 = new Paho.MQTT.Message(n1)
message4 = new Paho.MQTT.Message(n2)
message5 = new Paho.MQTT.Message(n3)
message3.destinationName = 'Av.Cevallos/sent.Oeste'
message4.destinationName = 'Av.Cevallos/sent.Este'
message5.destinationName = 'Calle/EugenioEspejo'
client.send(message3)
client.send(message4)
client.send(message5)
RA = contadorA - 21;
RB = contadorB - 21;
RC = contadorC - 4;
console.log("Control: Dos ");
console.log("RA = " + RA + " vehículos");
console.log("RB = " + RB + " vehículos");
console.log("RC = " + RC + " vehículos");
condiciones();
contadorA = 0;
contadorB = 0;
contadorC = 0;
}
// Método de control para el fase tres
function controlFaseTres(){
console.log("Contador A = " + contadorA + " vehículos");
console.log("Contador B = " + contadorB + " vehículos");
```

```
console.log("Contador C = " + contadorC + " vehículos");
n1 = contadorA.toString();
n2 = contadorB.toString();
n3 = contadorC.toString();
message3 = new Paho.MQTT.Message(n1)
message4 = new Paho.MQTT.Message(n2)
message5 = new Paho.MQTT.Message(n3)
message3.destinationName = 'Av.Cevallos/sent.Oeste'
message4.destinationName = 'Av.Cevallos/sent.Este'
message5.destinationName = 'Calle/EugenioEspejo'
client.send(message3)
client.send(message4)
client.send(message5)
RA = contadorA - 4;
RB = contadorB - 4;
RC = contadorC - 21;
console.log("Control: Tres ");
console.log("RA = " + RA + " vehículos");
console.log("RB = " + RB + " vehículos");
console.log("RC = " + RC + " vehículos");
condiciones();
contadorA = 0;
contadorB = 0;
contadorC = 0;
}
// Método para los mensajes recibidos en los distintos tópicos
function onMessageArrived(message) {
// Manejo de datos recibidos para el vehículo 1
if(message.destinationName == 'PlacaA'){
placa1 = (message.payloadString)
}
if(message.destinationName == 'LatitudA'){
latitud1 = parseFloat(message.payloadString)
```

```
}
if(message.destinationName == 'LongitudA'){
longitud1 = parseFloat(message.payloadString)
console.log(" Placa = " + placa1 +" Latitud = " + latitud1 + " Longitud  = " + longitud1);
// Marcador para el vehiculo 1
if (vehiculo1) {
mapa.removeLayer(vehiculo1)
}
vehiculo1 = L.marker([latitud1, longitud1], { icon: iconMarker2 }).addTo(mapa)
metodocontador1();
}
// Manejo de datos recibidos para el vehículo n
if(message.destinationName == 'PlacaN'){
placaN = (message.payloadString)
}
if(message.destinationName == 'LatitudN'){
latitudN = parseFloat(message.payloadString)
}
if(message.destinationName == 'LongitudN'){
longitudN = parseFloat(message.payloadString)
// Marcador para el vehiculo n
if (vehiculoN) {
mapa.removeLayer(vehiculoN);
}
vehiculoN = L.marker([latitudN, longitudN], { icon: iconMarker2 }).addTo(mapa)
metodocontadorN();
}}
// Semáforos A y B para la interfaz
const img1 = document.getElementById( 'semaforoA' );
const img2 = document.getElementById( 'semaforoB' );
// Método para las condiciones del sistema
function condiciones(){
if(RA >= 12 && RB >= 12 && RC >= 12){
```

```
faseUno();
}
if(RA >= 12 && RB >= 12 && RC < 12){
faseDos();
}
if(RA >= 12 && RB < 12 && RC >= 12){
faseUno();
}
if(RA >= 12 && RB < 12 && RC < 12){
faseDos();
}
if(RA < 12 && RB >= 12 && RC >= 12){
faseUno();
}
if(RA < 12 && RB >= 12 && RC < 12){
faseDos();
}
if(RA < 12 && RB < 12 && RC >= 12){
faseTres();
}
if(RA < 12 && RB < 12 && RC < 12){
faseUno();
}
}
// Fase de semaforización uno
function faseUno(){
message1 = new Paho.MQTT.Message("Uno")
message2 = new Paho.MQTT.Message("Uno")
message1.destinationName = 'Semaforo/A'
message2.destinationName = 'Semaforo/B'
client.send(message1)
client.send(message2)
tipoDeActividad = "Circulación normal"
```

```
document.getElementById("actividad").value =  tipoDeActividad;
numeroDeFase = "Fase: Uno"
console.log(numeroDeFase);
img1.src = 'imagenes/verde.png'
img2.src = 'imagenes/rojo.png'
setTimeout(function(){img1.src = 'imagenes/amarillo.png'
}, 42000)
setTimeout(function(){img1.src = 'imagenes/rojo.png', img2.src = 'imagenes/verde.png'
}, 46000)
setTimeout(function(){img2.src = 'imagenes/amarillo.png'
}, 87000)
setTimeout(function(){ controlFaseUno()
}, 91000);
}
// Fase de semaforización dos
function faseDos(){
message1 = new Paho.MQTT.Message("Dos")
message2 = new Paho.MQTT.Message("Dos")
message1.destinationName = 'Semaforo/A'
message2.destinationName = 'Semaforo/B'
client.send(message1)
client.send(message2)
tipoDeActividad = "Descongestión Av. Cevallos"
document.getElementById("actividad").value =  tipoDeActividad;
numeroDeFase = "Fase: Dos"
console.log(numeroDeFase);
img1.src = 'imagenes/verde.png'
img2.src = 'imagenes/rojo.png'
setTimeout(function(){img1.src = 'imagenes/amarillo.png'
}, 71000)
setTimeout(function(){img1.src = 'imagenes/rojo.png', img2.src = 'imagenes/verde.png'
}, 75000)
setTimeout(function(){img2.src = 'imagenes/amarillo.png'
```

```
}, 87000);
setTimeout(function(){controlFaseDos()
}, 91000);
}
// Fase de semaforización tres
function faseTres(){
message1 = new Paho.MQTT.Message("Tres")
message2 = new Paho.MQTT.Message("Tres")
message1.destinationName = 'Semaforo/A'
message2.destinationName = 'Semaforo/B'
client.send(message1)
client.send(message2)
tipoDeActividad = "Descongestión C.Espejo"
document.getElementById("actividad").value =  tipoDeActividad;
numeroDeFase = "Fase: Tres"
console.log(numeroDeFase);
img1.src = 'imagenes/verde.png'
img2.src = 'imagenes/rojo.png'
setTimeout(function(){img1.src = 'imagenes/amarillo.png'
}, 13000)
setTimeout(function(){img1.src = 'imagenes/rojo.png', img2.src = 'imagenes/verde.png'
}, 20000)
setTimeout(function(){img2.src = 'imagenes/amarillo.png'
}, 87000)
setTimeout(function(){controlFaseTres()
}, 91000);
}
// Método para resetear las variables de conteo
function reset(){
contador1A = 0;
contador1B = 0;
contador1C = 0;
contadorNA = 0;
```

```
contadorNB = 0;
contadorNC = 0;
}
// Método para redirigir a la página de inicio
function salir(){
window.location = "index.html";}
```

ANEXO F

A través del siguiente código fuente estilos.css se agrego diseño gráfico a la página de inicio y a la interfaz de monitoreo del sistema para el servidor de niebla.

```
body {
   margin: 0;
   padding: 0;
   background: #ffffff;
   font-family: 'Times New Roman';
}
.inicio {
   width: 460px;
   height: 420px;
   color: #000;
   background: #fff;
   top: 50%;
   left: 50%;
   position: absolute;
   transform: translate(-50%, -50%);
   box-sizing: border-box;
   padding: 100px 30px;
 }
 .inicio .icono {
   width: 200px;
   height: 200px;
```

```
  position: absolute;
  top: -100px;
  left: calc(39% - 50px);
}
.inicio h1 {
  margin: 20;
  padding: 0 0 20px;
  color: #000000;
  text-align: center;
  font-size: 22px;
}
.inicio h2 {
  margin: 20;
  padding: 0 0 20px;
  color: #000000;
  text-align: center;
  font-size: 20px;
}
.inicio label {
  margin: 0;
  padding: 0;
  font-weight: bold;
  display: block;
  text-align: center;
  font-size: 20px;
}
.inicio input {
  width: 100%;
  margin-bottom: 20px;
}
.inicio input[type="text"], .inicio input[type="password"] {
  border: none;
```

```
    border-bottom: 1px solid #000000;
    background: transparent;
    outline: none;
    height: 40px;
    color: #000000;
    font-size: 16px;
    text-align: center;
  }
.inicio input[type="button"] {
    border: none;
    outline: none;
    height: 40px;
    background:#FF0000;
    color: #fff;
    font-size: 20px;
    border-radius: 20px;
  }
.boton input[type="button"] {
  border: none;
  outline: none;
  width: 125px;
  height: 36px;
  background:#FF0000;
  color: #fff;
  font-size: 22px;
  border-radius: 10px;
  margin-top: 10px;
  margin-left: 90px;
}
```

ANEXO G

El siguiente código fuente denominado index.html se utilizó para realizar la interfaz gráfica de inicio o autenticación para el servidor en la nube.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Registro de históricos del sistema de control de tráfico vehicular</title>
<!-- Estilo css-->
<link rel="stylesheet" href="css/estilos.css">
<!-- Archivo javaScript para validar el usuario y contraseña-->
<script src="js/inicio.js"></script>
</head>
<body>
<div class="inicio">
<img src="imagenes/aws.png" class="icono">
<h1>Sistema de control de tráfico vehicular</h1>
<h2>Registro de históricos</h2>
<form >
<!-- Usuario-->
<label for="username">Usuario</label>
<input type="text" id="username"placeholder="Ingresar Usuario">
<!-- Contraseña -->
<label for="password">Contraseña</label>
<input type="password" id="password" placeholder="Ingresar Contraseña">
<!-- Acceder -->
<input type="button" value="Acceder" onclick="validate()">
</form>
</div>
</body>

</html>
```

ANEXO H

A través del siguiente código fuente se realizó la interfaz gráfica de registro de históricos del sistema para el servidor en la nube, el cual se denomina interfaz_de_historicos.php.

```
<?php
require 'consultas.php';
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Registro de históricos del sistema de control de tráfico vehicula</title>
<!-- Estilo css-->
<link rel="stylesheet" href="css/estilos.css">
<!-- Libreria de Bootstrap-->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css">
<!-- Librerias de JavaScript para Mqtt-->
<script src="js/mqttws31.js"></script>
<!-- Librerias de Highcharts-->
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/stock/modules/exporting.js"></script>
</head>
<body>
<div class="container" >
<div class="row" >
<div class="col-md-12">
<div class="card">
<div class="card-body">
```

<h3 class ="text-center bg-info" style="font-family: Times New Roman" > Registro de históricos del sistema de control de tráfico vehicular</h3>

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

* *

<hr color="gray" size=2>

* *

* *

* *

* *

* *

<h4 class ="text-center bg-light" style="font-family: Times New Roman" > Reportes </h4>

<h5 class ="text-center bg-light" style="font-family: Times New Roman"> Seleccione una fecha</h5>

<form method="post">

<input type="date" name="FechaIngresada">

<input type="submit" name="Enviar" >

</form>

* *

* *

* *

```
<div class="row">
<div class="col-md-7">
<div class="card">
<div class="card-body1" >
<h5 class ="text-center bg-success" style="font-family: Times New Roman" > <b>Reporte del conteo de vehículos</b></h5>
<table>
<tr><td>Identificador</td>
<td>Detalle</td>
<td>Cantidad</td>
<td>Recibido</td></tr>
<?php
while($tabla1=mysqli_fetch_array($reporteVehiculos )){
?>
<tr><td> <?php echo $tabla1['Id'] ?> </td>
<td> <?php echo $tabla1['Topico'] ?> </td>
<td> <?php echo $tabla1['Mensaje'] ?> </td>
<td><?php echo $tabla1['Recibido'] ?></td></tr>
<?php}
?>
</table></div></div></div>
<div class="col-md-5">
<div class="card">
<div class="card-body1">
<h5 class ="text-center bg-success" style="font-family: Times New Roman" > <b>Reporte de las fases de semaforización</b></h5>
<table>
<tr><td>Identificador</td>
<td>Detalle</td>
<td>Fase</td>
<td>Recibido</td></tr>
<?php
while($tabla2=mysqli_fetch_array($reporteFases)){
?>
```

```
<tr><td> <?php echo $tabla2['Id'] ?> </td>
<td> <?php echo $tabla2['Topico'] ?> </td>
<td> <?php echo $tabla2['Mensaje'] ?> </td>
<td><?php echo $tabla2['Recibido'] ?></td></tr>
<?php}
?>
</table></div></div></div>
</div></div></div></div></div></div></div><br>
<div class="boton">
<input type="button" value="Salir" style="font-family: Times New Roman" onclick="salir()"></div>
<!-- Archivo javaScript para el procesamiento de datos en la nube-->
<script src="js/nube.js"></script>
<?php
require 'graficas.php';
?>
</body  onload="init();">
</html>
```

ANEXO I

Mediante el siguiente código fuente inicio.js se realizó la validación del usuario y cotraseña para acceder a la interfaz de registro de históricos del servidor en la nube.

```
function validate(){
var username = document.getElementById("username").value;
var password = document.getElementById("password").value;
if ( username == "Administrador" && password == "eibarra9373"){
//Redirigiendo a la siguiente página
window.location = "interfaz_de_historicos.php";
}
else{
alert("Porfavor ingrese, nombre de usuario y contraseña correctos.");}}
```

ANEXO J

El siguiente código fuente denominado nube.js se utilizó para realizar la gráfica del conteo de vehículos en la interfaz web del registro de históricos del servidor en la nube.

```
// Declarar una variable para el servidor de niebla
var clientId = "ws" + Math.random();
// Crear una instancia para el servidor de niebla
var client = new Paho.MQTT.Client("18.230.182.193", 9001, clientId);
// Establecer controladores de devolución de llamada
client.onConnectionLost = onConnectionLost;
client.onMessageArrived = onMessageArrived;
// Conectar al cliente de niebla
client.connect({onSuccess:onConnect});
// Llamada cuando el cliente de niebla se conecta
function onConnect() {
// Mensaje de conectado en la consola una vez que se haya establecido la conexión
estado = "Conectado";
console.log(estado);
// Tópicos a suscribirse
client.subscribe('Av.Cevallos/sent.Oeste', {qos: 1});
client.subscribe('Av.Cevallos/sent.Este', {qos: 1});
client.subscribe('Calle/EugenioEspejo', {qos: 1});
}
// Llamada cuando el cliente de niebla pierde la conexión
function onConnectionLost(responseObject) {
if (responseObject.errorCode !== 0) {
estado = "Desconectado";
console.log(estado);
console.log(estado+responseObject.errorMessage);
}}
// Variables globales
var chart;
```

```
var datosTopicos = new Array();
// Método para los mensajes recibidos en los distintos tópicos
function onMessageArrived(message) {
console.log(message.destinationName +": " + message.payloadString);
// Verificar si el es tema nuevo y agregrar a la matriz
if (datosTopicos.indexOf(message.destinationName) < 0 ){
// Agregar un nuevo tema a la matriz
datosTopicos.push(message.destinationName);
// Obtener el índice del nuevo tema
var indice = datosTopicos.indexOf(message.destinationName);
// Crear una nueva serie de datos para el gráfico 1
var newseries = {
id: indice,
name: message.destinationName,
data: []};
// Agregar la serie de datos
chart.addSeries(newseries);};
// Obtener el número de índice del tema de la matriz
var indice = datosTopicos.indexOf(message.destinationName);
// Obtener el tiempo del dato
var tiempo = new Date().getTime();
// Eliminar cualquier espacio de texto del mensaje
var eliminar = message.payloadString.replace( /^\D+/g, '');
// Crear la matriz
var matriz = [tiempo, Number(eliminar)];
// Comprobar si es un número real y no un string
if (isNumber(eliminar)) {
// Enviar a la función plot
plot(matriz, indice);
};};
function isNumber(n) {
return !isNaN(parseFloat(n)) && isFinite(n);};
// Función que se ejecuta cuando se carga el documento
```

```
function init() {
// Controlar zonas horarias
Highcharts.setOptions({
});
// Conéctese al broker Mqtt
client.connect(options);};
// Funcion para definir el gráfico 1
function plot(point, chartno) {
var series = chart.series[0],
// Cambiar si la serie es mayor a 20
shift = series.data.length > 20;
// Agrega el dato mediante un punto
chart.series[chartno].addPoint(point, true, shift);  };
// Configuración del gráfico 1
$(document).ready(function() {
chart = new Highcharts.Chart({
time: {
useUTC: false},
chart: {
renderTo: 'grafica1',
type: 'line',
zoomType: 'x'},
title: {
text: 'Conteo de vehículos',
style:{
fontFamily: 'Times New Roman',
fontWeight: 'bold',
fontSize: '24px'} },
xAxis: {
type: 'datetime',
tickPixelInterval: 150,
maxZoom: 20 * 1000,},
yAxis: {
```

```
minPadding: 0.2,
maxPadding: 0.2,
title: {
text: 'Número de vehículos',
margin: 20,
style:{
fontFamily: 'Times New Roman',
fontSize: '20px'}}},
series: []
});});
function salir(){
//Redirigiendo a la siguiente página
window.location = "index.html";
}
```

ANEXO K

Mediante el siguiente código fuente se realiza las consultas a la base de datos mysql para el servidor en la nube, el cual se denomina consultas.php.

```
<?php
$connection = mysqli_connect("localhost", "root", "servidoraws","Datos");
$data = $_POST ['FechaIngresada'];
$year = substr("$data", 0, 4);
$month = substr("$data", 5, 2);
$day = substr("$data", 8, 2);
$geovallaA = mysqli_query($connection, "SELECT UNIX_TIMESTAMP(Recibido),
Mensaje FROM Contenido WHERE
year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day'
AND `Topico`= 'Av.Cevallos/sent.Oeste'");
$geovallaB = mysqli_query($connection, "SELECT UNIX_TIMESTAMP(Recibido),
Mensaje FROM Contenido WHERE
year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day'
AND `Topico`= 'Av.Cevallos/sent.Este'");
```

```
$geovallaC = mysqli_query($connection, "SELECT UNIX_TIMESTAMP(Recibido),
Mensaje FROM Contenido WHERE
year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day'
AND `Topico`= 'Calle/EugenioEspejo'");
$reporteVehiculos = mysqli_query($connection, "SELECT Id, Topico, Mensaje, Recibido
FROM Contenido WHERE
year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day'
AND `Topico`!= 'Semaforo/A' AND `Topico`!= 'Semaforo/B'");
$reporteFases = mysqli_query($connection, "SELECT Id, Topico, Mensaje, Recibido FROM
Contenido WHERE
year(`Recibido`)= '$year' AND month(`Recibido`) = '$month' AND day(`Recibido`) = '$day'
AND `Topico`!= 'Av.Cevallos/sent.Oeste' AND `Topico`!= 'Av.Cevallos/sent.Este' AND
`Topico`!= 'Calle/EugenioEspejo'");?>
```

ANEXO L

A través del siguiente código fuente se realiza la gráfica de consultas correspondiete al conteo de vehículos en la interfaz web del registro de históricos del servidor en la nube, denominada graficas.php.

```
<script>
$(function () {
$('#grafica2').highcharts({
time: {
useUTC: false},
chart: {
type: 'line',
zoomType: 'x'},
title: {
text: 'Consultas',
style:{
fontFamily: 'Times New Roman',
fontWeight: 'bold',
fontSize: '24px'} },
xAxis: {
```

```
type: 'datetime',},
yAxis: {
title: {
text: 'Número de vehículos',
margin: 20,
style:{
fontFamily: 'Times New Roman',
fontSize: '20px'
}}},
series: [
{name: 'Av.Cevallos/sent.Oeste', data:[<?php
 while($registrosA = mysqli_fetch_array($geovallaA)){
echo "[";
echo ($registrosA[0]*1000);
echo ",";
echo $registrosA[1];
echo "],";}
?>]},
{name: 'Av.Cevallos/sent.Este', data:[<?php
while($registrosB = mysqli_fetch_array($geovallaB)){
echo "[";
echo ($registrosB[0]*1000);
echo ",";
echo $registrosB[1];
echo "],";}?>]},
{name: 'Calle/EugenioEspejo', data:[<?php
while($registrosC = mysqli_fetch_array($geovallaC)){
echo "[";
echo ($registrosC[0]*1000);
echo ",";
echo $registrosC[1];
echo "],";}
?>]}]});});</script>
```

ANEXO M

El siguiente código fuente denominado datos_nube.php se utilizó para conectar la base de datos mysql con el broker mosquitto en el servidor de la nube.

```
<?php
use Mosquitto\Client;
use Mosquitto\Message;
class MqttToDb {
private $mqtt;
private $db;
private $topics = [];
private $insertMessage;
public function __construct(Client $mqtt, PDO $db){
$this->mqtt = $mqtt;
$this->db = $db;
$this->insertMessage = $this->db->prepare(
'INSERT INTO Contenido (Id, Topico, Mensaje, Recibido) VALUES (NULL, ?, ?, CURRENT_TIMESTAMP);');
$this->mqtt->onConnect([$this, 'subscribeToTopics']);
$this->mqtt->onMessage([$this, 'handleMessage']);}
public function setTopics(array $topics){
$this->topics = $topics;}
public function subscribeToTopics() {
foreach ($this->topics as $topic => $qos) {
$this->mqtt->subscribe($topic, $qos);}}
public function handleMessage($message){
$this->insertMessage->execute([$message->topic, $message->payload]);}
public function start(){
$this->mqtt->loopForever();}}
$db = new PDO('mysql:host=localhost;dbname=Datos;charset=utf8', 'root', 'servidoraws');
$mqtt = new Client();
$mqtt->connect('18.230.182.193',1883);
$logger = new MqttToDb($mqtt, $db);
$logger->setTopics([
```

```
'#' => 0, ]);
$logger->start();
```

ANEXO N

A través del siguiente código fuente estilos.css se agrego diseño gráfico a la página de inicio y a la interfaz de registro de históricos del sistema para el servidor en la nuble.

```
body {
    margin: 0;
    padding: 0;
    background: #ffffff;
    font-family: 'Times New Roman';
}
.inicio {
    width: 460px;
    height: 420px;
    color: #000;
    background: #fff;
    top: 50%;
    left: 50%;
    position: absolute;
    transform: translate(-50%, -50%);
    box-sizing: border-box;
    padding: 100px 30px;
  }
  .inicio .icono {
    width: 180px;
    height: 180px;
    position: absolute;
    top: -80px;
    left: calc(40% - 50px);
  }
```

```
.inicio h1 {
  margin: 20;
  padding: 0 0 20px;
  color: #000000;
  text-align: center;
  font-size: 22px;
}
.inicio h2 {
  margin: 20;
  padding: 0 0 20px;
  color: #000000;
  text-align: center;
  font-size: 20px;
}
.inicio label {
  margin: 0;
  padding: 0;
  font-weight: bold;
  display: block;
  text-align: center;
  font-size: 20px;
}
.inicio input {
  width: 100%;
  margin-bottom: 20px;
}
.inicio input[type="text"], .inicio input[type="password"] {
  border: none;
  border-bottom: 1px solid #000000;
  background: transparent;
  outline: none;
  height: 40px;
  color: #000000;
```

```
    font-size: 16px;
    text-align: center;
  }
.inicio input[type="button"] {
    border: none;
    outline: none;
    height: 40px;
    background:#545454;
    color: #fff;
    font-size: 20px;
    border-radius: 20px;
  }
.boton input[type="button"] {
  border: none;
  outline: none;
  width: 125px;
  height: 36px;
  background:#545454;
  color: #fff;
  font-size: 22px;
  border-radius: 10px;
  margin-top: 10px;
  margin-left: 500px;
}
form {
  margin: 0 auto;
  width:250px;
  }
  .card-body1{
    margin-left: auto;
    margin-right: auto;
  }
  table{
```

```
  background-color: white;
  text-align: center;
  border-collapse: collapse;
  width: 100%;
  font-family: 'Times New Roman';
 }
 td{
  padding: 10px;
 }
 tr:nth-child(even){
  background-color: #ddd;
 }
 tr:hover td{
  background-color:#A9A9A9;
  color: black;
 }
 table tr:nth-child(1) {
  font-weight: bold;
 }
```

ANEXO Ñ

A través del siguiente código fuente SplashActivity.java se realizó para configurar la pantalla de bienvenida en aplicación móvil.

```
package uta.edu.ec;
import android.content.Intent;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
public class SplashActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      Intent intent = new Intent(this, LoginActivity.class);
```

```
        startActivity(intent);
        finish();
    }
}
```

ANEXO O

El siguiente código fuente se realizó para configurar la pantalla de inicio en aplicación móvil, el cual se denomina LoginActivity.java.

```
package uta.edu.ec;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import com.google.android.material.button.MaterialButton;
public class LoginActivity extends AppCompatActivity {
    //Instanciar los elementos del layout activity_login
    MaterialButton btIniciar;
    EditText  editTextPlaca;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        //Intent para pasar de actividad
        // btIniciar = (MaterialButton) findViewById(R.id.btIniciar);
        editTextPlaca = (EditText)findViewById(R.id.editTextPlaca);
        btIniciar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(),MainActivity.class);
                //enviar la placa ingresada
                intent.putExtra("Placa", editTextPlaca.getText().toString());
```

```
            startActivity(intent);
        }
    });
  }
}
```

ANEXO P

A través del siguiente código fuente MainActivity.java se configuró las funciones principanes que tiene la aplicación móvil las cuales son obtener la ubicación y la comunicación con el sevidor en la niebla.

```
package uta.edu.ec;
import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.ImageView;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.tasks.OnSuccessListener;
import org.eclipse.paho.android.service.MqttAndroidClient;
```

```
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
public class MainActivity extends AppCompatActivity {
    // Constantes para la petición de ubicación
    public static final int DEFAULT_UPDATE_INTERVAL = 1;
    public static final int FAST_UPDATE_INTERVAL = 5;
    // Constante para dar permiso de ubicación, numero arbitrario
    public static final int PERMISSIONS_FINE_LOCATION = 100;
    // Referenciar a las variables de la interfaz de usuario
    TextView tv_lat, tv_lon, tv_sensor, tv_updates;
    Switch sw_gps;
    Switch sw_locationupdates;
    // Variable que indica si estamos obteniendo la ubicación o no
    boolean updateOn = false;
    // Define la clase del proveedor de ubicación fusionada
    FusedLocationProviderClient fusedLocationProviderClient;
    // Define la clase para la solicitud de ubicaón
    LocationRequest locationRequest;
    // Declarar la devolucion de solitud de nueva ubicación
    LocationCallback locationCallback;
    private static final String TAG = "MyTag";
    // Declarar cliente como variables globales
    MqttAndroidClient client;
    private String mensajeRecibido;
    // Declarar los elementos del layout activity_main
    TextView tv_placa;
    String Placa;
    private ImageView iv_semaforoUno;
```

```
    private ImageView iv_semaforoDos;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Instanciar los elementos del layout activity_main
        tv_placa = (TextView) findViewById(R.id.tv_placa);
        // Obtener la placa ingresada
        Bundle bundle = getIntent().getExtras();
        Placa = bundle.getString("Placa").toString();
        tv_placa.setText(Placa);
        // Generar el cliente Mqtt
        String clientId = MqttClient.generateClientId();
        client = new MqttAndroidClient(this.getApplicationContext(),
"tcp://proxy19.rt3.io:39155", clientId);
        try {
            IMqttToken token = client.connect();
            token.setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken) {
                    // Si es exitosa Llamar al metodo suscription de topicos
                    Toast.makeText(MainActivity.this, "Conectado al Sistema",
Toast.LENGTH_LONG).show();
                    suscriptionTopic();
                }
                @Override
                public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
                    Toast.makeText(MainActivity.this, "Error de Conexión",
Toast.LENGTH_LONG).show();
                }
            });
        } catch (MqttException e) {
            e.printStackTrace();
        }
```

```
// Parte del callback para recibir los mensajes
client.setCallback(new MqttCallback() {
   @Override
   public void connectionLost(Throwable cause) {
   }
   @Override
   public void messageArrived(String topic, MqttMessage message) throws Exception {
      mensajeRecibido = new String(message.getPayload());
      if(mensajeRecibido.equals("Uno") ){
         faseUno();
      }
      if(mensajeRecibido.equals("Dos") ){
         faseDos();
      }
      if(mensajeRecibido.equals("Tres") ){
         faseTres();
      }
   }
   @Override
   public void deliveryComplete(IMqttDeliveryToken token) {
   }
});
// Valor para cada variable de la interfaz de usuario
tv_lat = findViewById(R.id.tv_lat);
tv_lon = findViewById(R.id.tv_lon);
tv_sensor = findViewById(R.id.tv_sensor);
tv_updates = findViewById(R.id.tv_updates);
sw_gps = findViewById(R.id.sw_gps);
sw_locationupdates = findViewById(R.id.sw_locationsupdates);
// Definir las propiedades de la solicitud de ubicación
locationRequest = new LocationRequest();
// Especificar la frecuencia que se realiza la verificación de ubicación predeterminada,
cada 1 segundos
```

```
        locationRequest.setInterval(1000 * DEFAULT_UPDATE_INTERVAL);
        // Especificar la frecuencia que se realiza la actualizacion de la ubicación, cada 5 segundos
        locationRequest.setFastestInterval(1000 * FAST_UPDATE_INTERVAL);
        // Especificar la categoria de prioridad para obener la ubicación
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        // Evento que se activa siempre que se cumple el intervalo de actualización por lo menos 5 segundos
        locationCallback = new LocationCallback() {
            // Método para obtener los resultados de la nueva actualizacion de ubicación
            @Override
            public void onLocationResult(LocationResult locationResult) {
                super.onLocationResult(locationResult);
                //guardar la ubicación en una variable llamada location y pasar a la interfaz de usuario de actualización
                //método para que vuelva a mostrar todos los valores más nuevos
                updateUIValues(locationResult.getLastLocation());
            }
        };
        // Evento del boton para seleccionar el tipo de sensor
        sw_gps.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Condición para comprobar el estado del switch
                if (sw_gps.isChecked()) {
                    // Estado activado usa el sensor GPS con prioridad alta
                    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
                    tv_sensor.setText("Usando Sensor GPS");
                } else {
                    // Estado desactivado usa las torres celulares y Wifi con prioridad balanceada
                    locationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
                    tv_sensor.setText("Usando Torres + Wifi");
                }
```

```
        }
    });
    // Boton para actualizar las ubicaciones
    sw_locationupdates.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //condición para comprobar el estado del switch
            if (sw_locationupdates.isChecked()) {
                // Llamar al metodo inicar la actualizacion de ubicación
                startLocationUpdates();
            } else {
                // Llamar al metodo detener la actualizacion de ubicación
                stopLocationUpdates();
            }
        }
    });
    // Llamar al método actualizar Gps
    updateGPS();
    // Iniciar Semáforos
    init();
}
// Método iniciar la actualizacion de ubicación
private void startLocationUpdates() {
    tv_updates.setText("Ubicación obtenida");
    //llamamos al proveedor de ubicación fusionada para actualizar
    fusedLocationProviderClient.requestLocationUpdates(locationRequest,
locationCallback, null);
    updateGPS();
}
// Método detener la actualizacion de ubicación
private void stopLocationUpdates() {
    // Actualizar las variables
    tv_updates.setText("La ubicación no obtenida");
```

```
        tv_lat.setText("Ubicación no obtenida");
        tv_lon.setText("Ubicación no obtenida");
        tv_sensor.setText("Ubicación no obtenida");
        //Eliminamos las actualizaciones nuevas de ubicación
        fusedLocationProviderClient.removeLocationUpdates(locationCallback);
    }
    // Método para reconocer la solicitud de permiso
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        // Condición para la solicitud de permiso
        switch (requestCode) {
            // Caso para un permiso otorgado
            case PERMISSIONS_FINE_LOCATION:
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    // Llama al método actualizar GPS
                    updateGPS();
                } else {
                    // Mostrar un mensaje de alerta
                    Toast.makeText(this, "This app requires permission to be granted in order to
work properly", Toast.LENGTH_SHORT).show();
                    finish();
                }
                break;
        }
    }
    // Método para actualizar el GPS
    private void updateGPS() {
        // Definir el proveedor de servicio para obtener la ubicación
        fusedLocationProviderClient                                    =
LocationServices.getFusedLocationProviderClient(MainActivity.this);
        // Obtener permisos para usar laubicación por GPS
```

```
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        // Permiso otrogado por el usuario
        // Llamar al proveedor de servicio para obtener la ubicación
        fusedLocationProviderClient.getLastLocation().addOnSuccessListener(this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Llamar al metodo actualizar las variables de la interfaz de usuario
                updateUIValues(location);
            }
        });
    } else {
        // Permiso no concedido todavía
        // Verificar si la versión del Sistema Operativo es suficiente para obtener la ubicación
        // Version de compilació adecuada M = 23
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            //petición para otorgar permisos
            requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSIONS_FINE_LOCATION);
        }
    }
  }
  // Método Actualizar las variables de la interfaz de usuario
  private void updateUIValues(Location location) {
    // Obtener la latitud en String e imprimimos en el Text View
    tv_lat.setText(String.valueOf(location.getLatitude()));
    tv_lon.setText(String.valueOf(location.getLongitude()));
    try {
        client.publish("PlacaA",Placa.getBytes(),0,false);
        client.publish("LatitudA",String.valueOf(location.getLatitude()).getBytes(),0,false);

client.publish("LongitudA",String.valueOf(location.getLongitude()).getBytes(),0,false);
```

```
    }catch (Exception e)
    {
        e.printStackTrace();
    }
}
private void init(){
    iv_semaforoUno = findViewById(R.id.iv_semaforoUno);
    iv_semaforoDos = findViewById(R.id.iv_semaforoDos);
    iv_semaforoUno.setImageResource(R.drawable.semaforo);
    iv_semaforoDos.setImageResource(R.drawable.semaforo);
}
// Fase de semaforización Uno
private void faseUno(){
    // Bucle hilo secundario
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            faseUnoIntervaloUno();
            faseUnoIntervaloDos();
            faseUnoIntervaloTres();
            faseUnoIntervaloCuatro();
        }
    });
}
public void faseUnoIntervaloUno(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.verde);
            iv_semaforoDos.setImageResource(R.drawable.rojo);
        }
    },0);
```

```
}
public void faseUnoIntervaloDos(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.amarillo);
        }
    },42000);
}
public void faseUnoIntervaloTres(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.rojo);
            iv_semaforoDos.setImageResource(R.drawable.verde);
        }
    },46000);
}
public void faseUnoIntervaloCuatro(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoDos.setImageResource(R.drawable.amarillo);
        }
    },87000);
}
// Fase de semaforización Dos
private void faseDos(){
    //bucle hilo secundario
    runOnUiThread(new Runnable() {
```

```
        @Override
        public void run() {
            faseDosIntervaloUno();
            faseDosIntervaloDos();
            faseDosIntervaloTres();
            faseDosIntervaloCuatro();
        }
    });
}
public void faseDosIntervaloUno(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.verde);
            iv_semaforoDos.setImageResource(R.drawable.rojo);
        }
    },0);
}
public void faseDosIntervaloDos(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.amarillo);
        }
    },71000);
}
public void faseDosIntervaloTres(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
```

```
            iv_semaforoUno.setImageResource(R.drawable.rojo);
            iv_semaforoDos.setImageResource(R.drawable.verde);
        }
    },75000);
}
public void faseDosIntervaloCuatro(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoDos.setImageResource(R.drawable.amarillo);
        }
    },87000);
}
// Fase de semaforización Tres
private void faseTres(){
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            faseTresIntervaloUno();
            faseTresIntervaloDos();
            faseTresIntervaloTres();
            faseTresIntervaloCuatro();
        }
    });
}
public void faseTresIntervaloUno(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.verde);
            iv_semaforoDos.setImageResource(R.drawable.rojo);
```

```
        }
    },0);
}
public void faseTresIntervaloDos(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.amarillo);
        }
    },13000);
}
public void faseTresIntervaloTres(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoUno.setImageResource(R.drawable.rojo);
            iv_semaforoDos.setImageResource(R.drawable.verde);
        }
    },20000);
}
public void faseTresIntervaloCuatro(){
    final Handler handler = new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            iv_semaforoDos.setImageResource(R.drawable.amarillo);
        }
    },87000);
}
// Suscripción de tópicos
private void suscriptionTopic(){
```

```
        try {
            client.subscribe("Semaforo/A",0);
            client.subscribe("Semaforo/B",0);
        }catch (MqttException e){
            e.printStackTrace();
        }
    }
}
```

ANEXO Q

El siguiente código fuente denominado activity_login.xml se utilizó para realizar la interfaz gráfica de inicio de la aplicación móvil.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginActivity"
    android:background="@color/colorSplash">
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.08" />
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.3" />
```

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.55" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.6" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.7045144" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.8016416" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.84" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline10"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.95" />
<TextView
    android:id="@+id/textViewTituloUno"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:gravity="center"
    android:text="Sistema de control de tráfico vehicular"
    android:textColor="@color/colorTextIconsPrimary"
    android:textSize="@dimen/Grande"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline" />
<ImageView
    android:id="@+id/ivInicio"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline4"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline2"
    app:srcCompat="@drawable/logo_fisei" />
<TextView
    android:id="@+id/textViewTituloDos"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:gravity="center"
    android:text="Ingrese el número de la placa"
    android:textColor="@color/colorTextIconsPrimary"
    android:textSize="@dimen/Mediano"
```

```
    app:layout_constraintBottom_toTopOf="@+id/guideline6"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline5" />
  <LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:gravity="center"
    app:layout_constraintBottom_toTopOf="@+id/guideline7"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline6">
    <EditText
      android:id="@+id/editTextPlaca"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:ems="10"
      android:gravity="center"
      android:fontFamily="@font/glegoo_bold"
      android:inputType="textCapCharacters" />
  </LinearLayout>
  <com.google.android.material.button.MaterialButton
    android:id="@+id/btIniciar"
    style="?android:attr/borderlessButtonStyle"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:elevation="6dp"
    android:fontFamily="@font/glegoo_regular"
    android:text="Iniciar"
    android:textSize="@dimen/Mediano"
```

```
    app:cornerRadius="500dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline10"
    app:layout_constraintEnd_toStartOf="@+id/guideline9"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="@+id/guideline3"
    app:layout_constraintTop_toTopOf="@+id/guideline8" />
  <androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.1" />
  <androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.9" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

ANEXO R

A través del siguiente código fuente denominado activity_main.xml se realizó la interfaz gráfica principal de la apliación móvil para la interacción con el usuario.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
  <TextView
```

```
    android:id="@+id/tv_labelplaca"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:gravity="center"
    android:text="Placa:"
    android:textColor="@color/colorAccent"
    android:textSize="@dimen/Grande"
    app:layout_constraintEnd_toStartOf="@+id/guideline11"
    app:layout_constraintHorizontal_bias="0.178"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toTopOf="@+id/guideline15" />
<TextView
    android:id="@+id/tv_placa"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:gravity="center"
    android:text="ABC-0000"
    android:textColor="@color/colorTextIconsPrimary"
    android:textSize="@dimen/Grande"
    app:layout_constraintStart_toEndOf="@+id/tv_labelplaca"
    app:layout_constraintTop_toTopOf="@+id/guideline15" />
<ImageView
    android:id="@+id/ivInicio"
    android:layout_width="315dp"
    android:layout_height="100dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline14"
    app:layout_constraintEnd_toStartOf="@+id/guideline11"
    app:layout_constraintHorizontal_bias="0.466"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toBottomOf="@+id/tv_placa"
    app:layout_constraintVertical_bias="0.676"
```

```
    app:srcCompat="@drawable/vehiculo" />
<View
    android:id="@+id/divider1"
    android:layout_width="409dp"
    android:layout_height="1dp"
    android:background="?android:attr/listDivider"
    app:layout_constraintBottom_toTopOf="@+id/sw_gps"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:ignore="MissingConstraints" />
<TextView
    android:id="@+id/tv_labelsensor"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:text="Sensor:"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toBottomOf="@+id/sw_gps" />
<TextView
    android:id="@+id/tv_sensor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_regular"
    android:text="Usando Sensor Gps"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toEndOf="@+id/tv_labelsensor"
    app:layout_constraintTop_toTopOf="@+id/tv_labelsensor" />
<Switch
    android:id="@+id/sw_gps"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
```

```
    android:text="Torres + Wifi / Gps"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.59"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline16" />
<TextView
    android:id="@+id/tv_labelupdates"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:text="Estado:"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toBottomOf="@+id/sw_locationsupdates" />
<TextView
    android:id="@+id/tv_updates"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_regular"
    android:text="Ubicación Obtenida"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toEndOf="@+id/tv_labelupdates"
    app:layout_constraintTop_toTopOf="@+id/tv_labelupdates" />
<Switch
    android:id="@+id/sw_locationsupdates"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:checked="true"
    android:text="Ubicación"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.485"
    app:layout_constraintStart_toStartOf="parent"
```

```
    app:layout_constraintTop_toBottomOf="@+id/tv_labelsensor" />
<TextView
    android:id="@+id/tv_labellat"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:text="Latitud:"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toTopOf="@+id/guideline13" />
<TextView
    android:id="@+id/tv_lat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_regular"
    android:text="0.00"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toEndOf="@+id/tv_labellat"
    app:layout_constraintTop_toTopOf="@+id/guideline13" />
<TextView
    android:id="@+id/tv_labellon"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:fontFamily="@font/glegoo_bold"
    android:text="Longitud:"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toBottomOf="@+id/tv_labellat" />
<TextView
    android:id="@+id/tv_lon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_marginTop="16dp"
    android:fontFamily="@font/glegoo_regular"
    android:text="0.00"
    android:textSize="@dimen/Mediano"
    app:layout_constraintStart_toEndOf="@+id/tv_labellon"
    app:layout_constraintTop_toBottomOf="@+id/tv_lat" />
<View
    android:id="@+id/divider2"
    android:layout_width="409dp"
    android:layout_height="1dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:background="?android:attr/listDivider"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tv_labellon"
    tools:ignore="MissingConstraints" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline11"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.96" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline12"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_begin="20dp" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline13"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
```

```
    app:layout_constraintGuide_percent="0.48" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline14"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_begin="140dp"
    app:layout_constraintGuide_end="561dp" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline15"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.021887826" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline16"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.27" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline17"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.7" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline18"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.98" />
<TextView
```

```
    android:id="@+id/tv_Titulo1"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:gravity="center"
    android:text="Semáforo A"
    android:textSize="@dimen/Mediano"
    app:layout_constraintEnd_toStartOf="@+id/guideline20"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toTopOf="@+id/guideline23" />
<TextView
    android:id="@+id/tv_Titulo2"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_bold"
    android:gravity="center"
    android:text="Semáforo B"
    android:textSize="@dimen/Mediano"
    app:layout_constraintEnd_toStartOf="@+id/guideline11"
    app:layout_constraintHorizontal_bias="0.494"
    app:layout_constraintStart_toStartOf="@+id/guideline20"
    app:layout_constraintTop_toTopOf="@+id/guideline23" />
<TextView
    android:id="@+id/tv_Titulo3"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_regular"
    android:gravity="center"
    android:text="Av. Cevallos"
    android:textSize="@dimen/Pequeño"
    app:layout_constraintEnd_toStartOf="@+id/guideline20"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toBottomOf="@+id/tv_Titulo1" />
```

```
<TextView
    android:id="@+id/tv_Titulo4"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:fontFamily="@font/glegoo_regular"
    android:gravity="center"
    android:text="C. Eugenio Espejo"
    android:textSize="@dimen/Pequeño"
    app:layout_constraintEnd_toStartOf="@+id/guideline11"
    app:layout_constraintStart_toStartOf="@+id/guideline20"
    app:layout_constraintTop_toBottomOf="@+id/tv_Titulo2" />
<ImageView
    android:id="@+id/iv_semaforoUno"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline18"
    app:layout_constraintEnd_toStartOf="@+id/iv_semaforoDos"
    app:layout_constraintStart_toStartOf="@+id/guideline12"
    app:layout_constraintTop_toTopOf="@+id/guideline17"
    app:srcCompat="@drawable/semaforo" />
<ImageView
    android:id="@+id/iv_semaforoDos"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline18"
    app:layout_constraintEnd_toStartOf="@+id/guideline11"
    app:layout_constraintStart_toStartOf="@+id/guideline20"
    app:layout_constraintTop_toTopOf="@+id/guideline17"
    app:srcCompat="@drawable/semaforo" />
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline20"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.5" />
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline19"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_begin="431dp" />
    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline23"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_begin="401dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

ANEXO S

El siguiente código fuente se realizó para configurar las funciones del semáforo inteligente A.

```
// Programación del microcontrolador ESP8266 para el semáforo A
// Libreria para la comunicación Wifi
#include <ESP8266WiFi.h>
// Libreria para la comunicación Mqtt-WebSocket
#include <PubSubClient.h>
// Variables globales para la comunicación con el servidor de niebla
int contconexion = 0;
const char* ssid = "Peluqueria Mary";
const char* password = "mary2020";
const char* server_niebla = "192.168.1.20";
const int mqttPort = 1883;
WiFiClient espClient;
```

```
PubSubClient client(espClient);
// Configuración del la llamada de retorno
void callback(char* topic, byte* payload, unsigned int length) {
char PAYLOAD[5] = "    ";
// Condiciones para los topics y mensajes del semáforo A
if( String(topic) == "Semaforo/A"){
//Cuando el mensaje recibido es la fase "Uno"
if(payload[1] == 'n'){
digitalWrite(16, LOW);
digitalWrite(4, HIGH);
delay(42000);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
delay(4000);
digitalWrite(5, LOW);
digitalWrite(16, HIGH);
}
//Cuando el mensaje recibido es la fase "Dos"
if(payload[1] == 'o'){
digitalWrite(16, LOW);
digitalWrite(4, HIGH);
delay(71000);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
delay(4000);
digitalWrite(5, LOW);
digitalWrite(16, HIGH);
}
//Cuando el mensaje recibido es la fase "Tres"
if(payload[1] == 'r'){
digitalWrite(16, LOW);
digitalWrite(4, HIGH);
delay(13000);
```

```
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
delay(4000);
digitalWrite(5, LOW);
digitalWrite(16, HIGH);
}}}
// Función para reconectarse al servidor de niebla
void reconnect() {
while (!client.connected()){
if (client.connect("Semaforo/A")){
client.subscribe("Semaforo/A");
} }}
void setup() {
//Especificar los pines digitales para las luces del semáforo
// Pin D0 como salida para la luz roja
pinMode(16,OUTPUT);
digitalWrite(16, LOW);
// Pin D1 como salida para la luz amarilla
pinMode(5,OUTPUT);
digitalWrite(5, LOW);
// Pin D2 como salida para la luz verde
pinMode(4,OUTPUT);
digitalWrite(4, LOW);
// Conexión ianlámbrica mediante WIFI
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED and contconexion <50)
{ //Cuenta hasta 50 si no se puede conectar al servidor cancela la conexón
++contconexion;
delay(500);
}
client.setServer(server_niebla, mqttPort);
client.setCallback(callback);
}
```

```
// Bucle para ejecutar la comunicación
void loop() {
if (!client.connected())
{
reconnect();
}
client.loop();
}
```

ANEXO T

El siguiente código fuente se realizó para configurar las funciones del semáforo inteligente B.

```
// Programación del microcontrolador ESP8266 para el semáforo B
// Libreria para la comunicación Wifi
#include <ESP8266WiFi.h>
// Libreria para la comunicación Mqtt-WebSocket
#include <PubSubClient.h>
// Variables globales para la comunicación con el servidor de niebla
int contconexion = 0;
const char* ssid = "Peluqueria Mary";
const char* password = "mary2020";
const char* server_niebla = "192.168.1.20";
const int mqttPort = 1883;
WiFiClient espClient;
PubSubClient client(espClient);
// Configuración del la llamada de retorno
void callback(char* topic, byte* payload, unsigned int length) {
char PAYLOAD[5] = "    ";
// Condiciones para los topics y mensajes del semáforo B
if( String(topic) == "Semaforo/B"){
//Cuando el mensaje recibido es la fase "Uno"
if(payload[1] == 'n'){
```

```
digitalWrite(5, LOW);
digitalWrite(16, HIGH);
delay(46000);
digitalWrite(4, HIGH);
digitalWrite(16, LOW);
delay(41000);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
}
//Cuando el mensaje recibido es la fase "Dos"
if(payload[1] == 'o'){
digitalWrite(5, LOW);
digitalWrite(16, HIGH);
delay(75000);
digitalWrite(4, HIGH);
digitalWrite(16, LOW);
delay(12000);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
}
//Cuando el mensaje recibido es la fase "Tres"
if(payload[1] == 'r'){
digitalWrite(5, LOW);
digitalWrite(16, HIGH);
delay(20000);
digitalWrite(4, HIGH);
digitalWrite(16, LOW);
delay(67000);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
}}}
// Función para reconectarse al servidor de niebla
void reconnect() {
```

```
while (!client.connected()){
if (client.connect("Semaforo/B")){
client.subscribe("Semaforo/B");
}}}
void setup() {
//Especificar los pines digitales para las luces del semáforo
// Pin D0 como salida para la luz roja
pinMode(16,OUTPUT);
digitalWrite(16, LOW);
// Pin D1 como salida para la luz amarilla
pinMode(5,OUTPUT);
digitalWrite(5, LOW);
// Pin D2 como salida para la luz verde
pinMode(4,OUTPUT);
digitalWrite(4, LOW);
// Conexión ianlámbrica mediante WIFI
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED and contconexion <50)
{ //Cuenta hasta 50 si no se puede conectar al servidor cancela la conexón
++contconexion;
delay(500);
}
client.setServer(server_niebla, mqttPort);
client.setCallback(callback);
}
// Bucle para ejecutar la comunicación
void loop() {
if (!client.connected())
{
reconnect();
}
client.loop();}
```