UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

Resolución: RPC-SO-09-No.265-2021

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGISTER

Título del proyecto:
Diseño e implementación de un sistema de monitoreo de tráfico en tiempo real que utilice visión artificial y sensores IoT
Línea de Investigación:
Ciencias de la ingeniería aplicadas a la producción, sociedad y desarrollo sustentable
Campo amplio de conocimiento:
Ingeniería, industria y construcción
Autor/a:
Luis Eduardo Zurita Morales
Tutor/a:
Mg. René Ernesto Cortijo Leyva / PhD. Yolvy Javier Quintero Cordero

Quito – Ecuador

2024

APROBACIÓN DEL TUTOR

Nosotros, Mg. **René Ernesto Cortijo Leyva** con C.I: **1719010108 y** PhD**. Yolvy Javier Quintero Cordero** con C.I**: 1759715301,** en calidad de Tutores del proyecto de investigación titulado: **"Diseño e implementación de un sistema de monitoreo de tráfico en tiempo real que utilice visión artificial y sensores IoT"**.

Elaborado por: **Ing. Luis Eduardo Zurita Morales**, de C.I: **1723707996,** estudiante de la Maestría: **Electrónica y Automatización**, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, nos permitimos declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, se aprueba en todas sus partes.

Quito 17 de septiembre del 2024

Firmado electrónicamente por:
RENE ERNESTO
CORTIJO LEYVA

Firma

Tutor Técnico

Firmado electrónicamente por:
YOLVY JAVIER
QUINTERO CORDERO

Firma

Tutor Metodológico

DECLARACIÓN DE AUTORIZACIÓN POR PARTE DEL ESTUDIANTE

Yo, Luis Eduardo Zurita Morales con C.I: 1723707996, autor/a del proyecto de titulación denominado: **Diseño e implementación de un sistema de monitoreo de tráfico en tiempo real que utilice visión artificial y sensores IoT**. Previo a la obtención del título de Magister en Electrónica y Automatización.

1. Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar el respectivo trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Manifiesto mi voluntad de ceder a la Universidad Tecnológica Israel los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor@ del trabajo de titulación, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital como parte del acervo bibliográfico de la Universidad Tecnológica Israel.
3. Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de prosperidad intelectual vigentes.

Quito 17 de septiembre del 2024

Firmado electrónicamente por:
LUIS EDUARDO ZURITA MORALES

Firma

Tabla de contenidos

Índice de tablas

Índice de figuras

INFORMACIÓN GENERAL

Contextualización del tema

La visión artificial (VA) es un método enfocado en el estudio, captación y comprensión de imágenes y videos. Se relaciona con otros campos como el aprendizaje máquina, inteligencia artificial (IA), neurociencia, los procesamientos VPC y de señal, psicología y robótica (Domingo et al., 2024). La visión por computadora (VPC) se basa en la búsqueda de patrones o características dentro de las imágenes para extraer información útil. Esto puede incluir detección de objetos, seguimiento de personas o vehículos y clasificación de eventos. Por otra parte, se puede emplear para analizar patrones y tendencias en el comportamiento del tráfico, lo que puede ayudar a los urbanistas a diseñar ciudades más eficientes y seguras (Cortijo & Arellano, 2020). Los sensores IoT (por sus siglas en inglés, *Internet of Things*), son dispositivos conectados a internet que pueden medir datos sobre el entorno. Estos nos permitirán recopilar grandes cantidades de datos sobre la conducta del tráfico real, a fin de mejorar la gestión del tránsito urbano (Suárez et al., 2020).

Quito es una ciudad ubicada en los Andes ecuatorianos que enfrenta desafíos comunes relacionados con el tráfico urbano. La ciudad cuenta con un sistema de transporte público limitado y una gran cantidad de vehículos particulares, lo que genera congestión y problemas de acceso a la vía pública. El tráfico, por ejemplo, en San Rafael, es complicado durante las horas pico del día y fines de semana, que se produce cuando el flujo de personas y vehículos aumenta significativamente. La gestión efectiva del tráfico urbano es crucial para mejorar la calidad de vida de las personas y reducir la congestión y la contaminación ambiental. La Agencia Metropolitana de Tránsito (AMT) es la institución pública encargada de planificar, coordinar y controlar el tráfico en Quito.

Problema de investigación

La gestión del tráfico urbano es un desafío crítico para las ciudades modernas, y Quito no es la excepción. A pesar de ser una ciudad con un sistema de transporte público limitado y un gran número de vehículos particulares, no existe un banco de información preciso y accesible sobre el tráfico urbano que permita tomar decisiones informadas y efectivas para mejorar la movilidad en la ciudad. La falta de datos precisos sobre el flujo de tráfico, los tiempos de viaje y las condiciones del tráfico impide diseñar soluciones innovadoras y eficientes para reducir la congestión y mejorar la calidad de vida de los transeúntes. Por ejemplo, no se pueden implementar semáforos inteligentes que ajusten su duración según el flujo de tráfico en tiempo real. La presente investigación busca llenar esta brecha al proporcionar un banco de información preciso sobre el tráfico urbano en San Rafael y explorar cómo este conocimiento

puede ser utilizado para desarrollar soluciones innovadoras y efectivas para la gestión del tráfico, como semáforos inteligentes o sistemas de control de flujo de tráfico.

Objetivo general

- Diseñar e implementar un sistema de monitoreo de tráfico en tiempo real que utilice visión artificial y sensores IoT.

Objetivos específicos

- Establecer las herramientas disponibles y el estado del arte sobre visión artificial y sensores IoT en la gestión del tráfico urbano.
- Desarrollar un prototipo de sistema de monitoreo de tráfico en tiempo real.
- Validar el funcionamiento de la plataforma de monitoreo de tráfico mediante la presentación de resultados recopilados.

Vinculación con la sociedad y beneficiarios directos:

El presente trabajo de investigación busca generar un banco de información preciso sobre el tráfico urbano en el sector de San Rafael, lo que permitirá contribuir significativamente a la sociedad y beneficiar directamente a los ciudadanos. Al desarrollar este proyecto, se espera que los resultados sean para mejorar y reducir el nivel de tráfico en la ciudad, lo que a su vez puede tener un impacto positivo en la calidad de vida de los transeúntes, la economía local y el medio ambiente.

Es decir, esta investigación permitirá el acceso a información actualizada sobre el tráfico urbano para diferentes investigadores o estudiantes en futuras investigaciones. Esto beneficiará a los especialistas en el área de interés y a otros que estén relacionados en la industria del transporte público y privado, como empresas de consultoría o instituciones educativas que buscan mejorar sus procesos y tomar decisiones informadas.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1.1 Contextualización general del estado del arte

El proyecto se enfoca en la necesidad de diseñar un sistema de monitoreo de tráfico. Para ello se implementa un prototipo que utilice VA y se comunique con una plataforma web la cual permitirá crear un banco de datos con la información recolectada por el prototipo. En consecuencia, se seleccionará un área urbana con alta densidad de tráfico para utilizar el dispositivo mencionado. El uso de la plataforma permitirá la posibilidad de escalar el monitoreo a más puntos mediante la adición de más dispositivos de monitoreo en caso de generarse la necesidad de un monitoreo más extenso.

El campo de monitoreo inteligente de tráfico es actualmente un tema de investigación activo debido al uso de tecnologías IoT e IA (Sarrab et al., 2020). Entre los conceptos que se aplican durante la investigación se encuentran los que se relacionan con VA, gestión de tráfico e IoT.

En Rishabh (2023) menciona roles como el IoT en el manejo de tráfico al incremento de la capacidad de las calles sin necesidad de construir nuevos caminos, la optimización del flujo de tráfico y el uso de información en tiempo real para el control dinámico de los semáforos, como control de nivel lumínico en función de condiciones ambientales.

En la investigación de Kunekar et al. (2024) se utiliza el algoritmo You Only Look Once (YOLO) que está construido con redes neuronales convolucionales para maximizar el número de vehículos que cruzan por una intersección. Se utilizó aprendizaje de máquina para automatizar el tráfico y la conclusión a la que llegaron es que se podría obtener mejores pronósticos a medida que se obtenga mayor cantidad de datos recolectados.

En el trabajo realizado por Vennila et al. (2021) se aplica IoT para realizar control y monitoreo de tráfico, a partir de un sistema de comunicación, dispositivos móviles de usuarios con sensores magnéticos y microcontroladores compatibles con wifi para guardar los datos de la aplicación en la nube y obtener una alta precisión en la estimación de ocupación de los caminos.

En la investigación nacional de Vega y Parra Narváez (2014) se caracterizó la intensidad vehicular media y perfiles horarios en la ciudad de Quito en 335 puntos (conteos). Esta información es proporcionada por la Secretaría de Movilidad, al generar mapas de emisiones contaminantes causadas por el tráfico vehicular.

Caiza Oña (2016) presenta el trabajo de titulación: “DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE SISTEMA DE SEMAFORIZACIÓN INTELIGENTE”, en el cual se construye una maqueta de las calles Pérez Guerrero y Cristóbal Colón donde se simula el control de tráfico, se

simula también una cámara de detección vehicular, el proyecto utiliza microcontroladores Atmega 2560 sincronizados, por lo que un sistema de recolección de información real podría aportar a tener simulaciones más completas en las que se pueda llevar a cabo proyectos futuros.

El proyecto de grado titulado “SISTEMA DE SEMAFORIZACIÓN ACTUADO, MEDIANTE CÁMARA DE VIDEO DETECCIÓN VEHICULAR, PARA PROYECTOS DE MOVILIDAD EN LA EMPRESA INDUSTRIAS SEBLAN CIA. LTDA” presentado por Ganchala Quishpe (2019) trata sobre sistemas de semaforización actuados, donde mediante la utilización de una cámara de detección vehicular se logró reducir tiempos de espera innecesarios. Esto nos permite concluir que la disponibilidad de información de sensores en un caso de intersección simple puede contribuir con el mejoramiento del control de tráfico y podría extrapolarse a intersecciones con mayor cantidad de vehículos.

Finalmente, Quintana Tenorio (2024) con el proyecto de posgrado titulado “Implementación de una Interfaz HMI Basado en Software Libre Node Red para el Control, Monitoreo Local y Remoto del Sistema de Bombeo en el Conjunto Habitacional Rivotorto” implementa un sistema de monitoreo basado en IoT mediante el protocolo de comunicación TCP/IP, de tal manera que facilita el monitoreo y control desde cualquier dispositivo en línea.

Un sistema de monitoreo de tráfico real que integre en conjunto sensores IoT con capacidades de procesamiento de imagen ofrece un aporte valioso para la toma de decisiones en gestión de tráfico. Esto, facilita la implementación de soluciones novedosas, lo que potencialmente puede generar valor agregado al mejorar la calidad de vida y obtener desplazamientos en menor tiempo.

1.2 Proceso investigativo metodológico

La investigación es de tipo aplicada, ya que se centra en la resolución de un problema práctico. Se tiene como objetivo, construir un banco de información abierto preciso y de tiempo real de monitoreo del tráfico que sea escalable a nivel de ciudad. Por su parte también permitirá comprender en mayor medida técnicas emergentes como la VA.

Para la recolección de información se emplean técnicas bibliográficas para seleccionar métodos de procesamiento de imagen y librerías de código factibles de aplicar en la resolución del problema.

También se emplea investigación de campo para evaluar el prototipo desarrollado ubicándolo en un punto de alto tráfico con lo que se inicia la recolección de datos que podrán ser accesibles a investigaciones futuras.

CAPÍTULO II: PROPUESTA

2.1 Fundamentos teóricos aplicados

Dentro de la movilidad urbana, el tráfico se considera como un factor de gran importancia, debido a los accidentes y congestión habituales (Slimani et al., 2023). Ante ello, han surgido varias formas de elaborar sistemas de monitorización del tráfico. Esto conlleva varios elementos físicos y de programación (Swathi et al., 2024).

2.1.1 Microcomputadora

La Raspberry Pi es una microcomputadora que se utiliza en una variedad de proyectos, tales como adquisición de datos, robótica y cualquier otra rama tecnológica y de ingeniería que involucre programación (Halfacree, 2024). Existe una gran variedad de modelos, visualizados en la Figura 1.

Figura 1

Raspberry Pi

Nota. Tomado de (Raspberry Pi Ltd., 2024)

Al ser de tamaño compacto y de fácil configuración, requiere ciertos periféricos como cualquier ordenador, tales como (Halfacree, 2024):

- Cada Raspberry Pi necesita de una fuente de alimentación de 5VDC; sin embargo, el amperaje depende del modelo, tal como se indica en la Tabla 1.

Tabla 1

Capacidad de las fuentes de alimentación

Raspberry Pi versión	Conector	Corriente (A)
5	USB-C	5
4 modelo B o 400	USB-C	3
Zero 2 W	Micro USB	2,5

Nota. Adaptado de (Halfacree, 2024)

- Una tarjeta microSD, la cual actúa como almacenamiento de información como el

software a instalar y archivos deseados.

- El teclado y ratón para lograr operar la Raspberry Pi.
- Un cable HDMI (Interfaz Multimedia de Alta Definición) a fin de transmitir el sonido e imágenes de la microcomputadora hacia la televisión o monitor.
- Puede utilizar una cámara para la captura de imágenes y videos. Ante ello, se incorpora la biblioteca PiCamera.

2.1.2 Sensores IoT

El IoT sirve para la interconexión de objetos, en este caso de sensores, a través de internet. Por medio del cual permite actuar, almacenar e intercambiar información sin la necesidad de una intervención directa por parte del usuario (persona). En la que se involucra un ecosistema donde los sensores operan como un conjunto para monitorear datos en tiempo real y brindar información valiosa a la hora de tomar decisiones (Suárez et al., 2020). Generalmente, los medios de comunicación presentan diversas tecnologías clasificadas según el ancho de banda, consumo de energía, entre otros. A continuación, se detallan los protocolos de acceso a la red (Barbara, 2024; Suárez et al., 2020):

- Corto alcance: bluetooth o de bajo consumo (BLE), Wi-Fi, zigbee y Z-Wave.
- Largo alcance: LoRa, sigfox, NB-IoT, LTE-M, 5G y WiMAX.
- Redes de satélites.
- Redes privadas Ethernet y 6LoWPAN (IPv6).

El término sensor se refiere a un dispositivo capaz de medir una variable física, la cual puede ser interpretada como señales analógicas y digitales. Existe una variedad de sensores que se utilizan para medir temperatura, presión, humedad y otros. Las especificaciones más nombradas de los sensores suelen ser la precisión, resolución y sensibilidad. Cabe mencionar que la mayoría de estos dispositivos suelen ser de comportamiento lineal (Aguilar, 2021). Finalmente, en la Tabla 2 se designan diversos sensores capaces de medir el entorno ambiental.

Tabla 2

Sensores de medición ambiental

Sensores	Mediciones			Tipo de salida
	Temperatura	Presión	Humedad	
DT11 o DT12	X		X	Digital
BMP180	X	X		Comunicación serial síncrona I2C (circuito inter integrado)
BMP280	X	X		Comunicaciones seriales síncronas
BME280	X	X	X	I2C y SPI (interfaz periférica serie)
LM35	X			Analógica
Series MPX		X		Analógica

Nota. Elaboración Propia

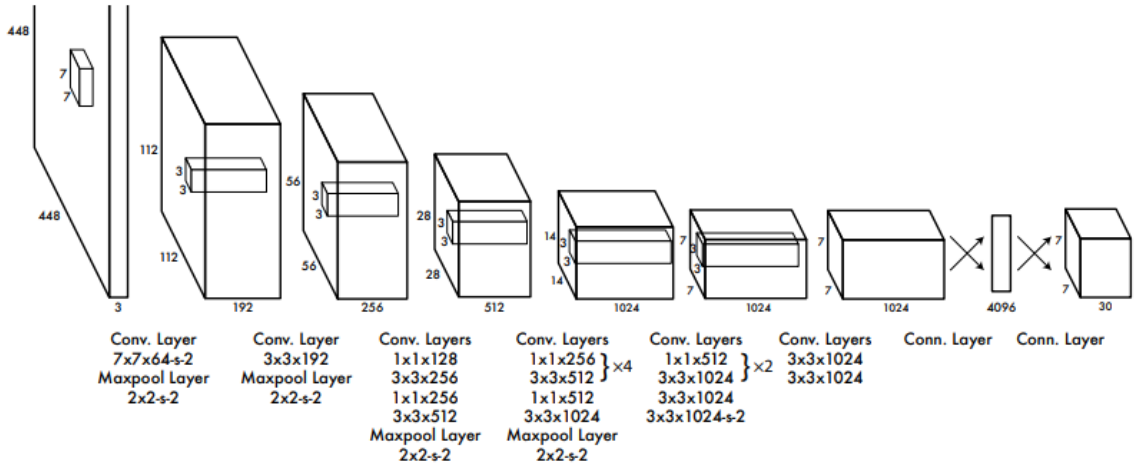
2.1.3 Bibliotecas y herramientas aplicadas

VA es un método enfocado en el estudio, captación y comprensión de imágenes y videos. Se relaciona con otros campos como el aprendizaje máquina, IA, neurociencia, los procesamientos VPC y de señal, psicología y robótica (Domingo et al., 2024). Existen varias bibliotecas y herramientas que se adaptan a la Raspberry Pi para el uso de VA, tales como:

- OpenCV (por sus siglas en inglés, *Open Source Computer Vision Library*). Incorpora una gran variedad de funciones para detección de objetos, procesamiento de videos, entre otros (Domingo et al., 2024). Esta librería se utiliza para mostrar las imágenes y videos en tiempo real. Aparte, de dibujar líneas y polígonos, superpone texto en las imágenes procesadas.
- TensorFlow es ideal para aplicar modelos de aprendizaje automático (ML, *machine learning*), como un clasificador y detección de imágenes. Otra herramienta que usa IA para el mismo propósito es PyTorch, el cual emplea aprendizaje profundo (DP, *deep learning*) para modelos ligeros (Tierra, 2020).
- Dlib es una biblioteca DL que incorpora herramientas para actividades de VPC.
- Yolo (por sus siglas en inglés, *You Only Look Once*) es un modelo VPC entrenado de detección de objetos en tiempo real, es decir, detecta, clasifica, segmenta y realiza un seguimiento de estos a partir de imágenes o videos. Por lo que incorpora 24, 4 y 2 capas convolucionales, agrupamiento máximo y totalmente conectadas; respectivamente, tal como se observa en la Figura 2 (Domingo et al., 2024; Ultralytics, 2024).

Figura 2

Arquitectura Yolo



Nota. Tomado de (Redmon et al., 2016)

- *Pickle* se utiliza para cargar lo almacenado en la configuración inicial.
- *Numpy* se encarga de definir las coordenadas en el área de interés para el procesamiento de imágenes.
- *Shapely.geometry*, aparte de limitar el polígono, también verifica si el centroide del vehículo detectado se encuentra dentro de este.
- *Ultralytics* (Yolo) se aplica para la detección y seguimiento de vehículos en el video capturado, luego los clasifica.
- La librería "time" se utiliza para actualizar los eventos de un proceso, como el envío de datos a una API cada minuto.
- "*threading*" ejecuta tareas en paralelo, como el contador de vehículos. De esta manera, se ejecuta el video, mientras se envía información a la API de URL: http://ticsnic.com/api/add_dato/.
- La librería *request* ayuda el envío de solicitudes HTTP y enviar la velocidad promedio, temperatura, presión y conteo de vehículos a una API externa.
- Se aplican las librerías *board* y *busio* para utilizar los pines y el protocolo I2C; respectivamente, entre el sensor BMP280 y la Raspberry Pi 5.
- "*adrafruit_bmp280*" se inicializa para la configuración del sensor.
- El historial de seguimientos de los vehículos necesita de un diccionario, por lo que se emplea la librería "*collections.defaultdict*".
- "*Picamera2*" se utiliza para la captura y manipulación de imágenes a través de la cámara.
- Finalmente, se emplean librerías de tiempo "*datetime*" y "timedelta" para obtener la

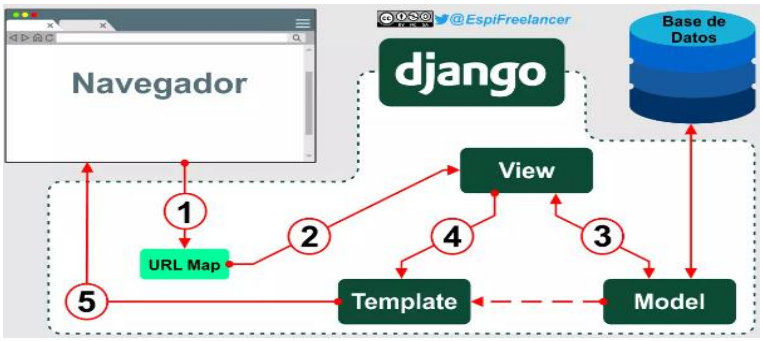
fecha y hora, luego se envía la información a la API.

2.1.4 Plataforma web

Una plataforma para facilitar la creación de aplicaciones web, es con Django en Python. Este es un framework de alto nivel, el cual maneja solicitudes HTTP y genera respuestas eficientes. El servidor Django utiliza una arquitectura basada en el patrón modelo – vista – plantilla (MVT), visualizado en la Figura 3. Este enruta las solicitudes a las vistas deseadas, las cuales procesan la secuencia lógica del modelo para interactuar con la base de datos y renderizar plantillas HTML para conceder una respuesta al navegador del usuario (W3Schools, 2024).

Figura 3

Funcionamiento MVT de Django



Nota. Tomado de (@EspiFreelancer, 2019)

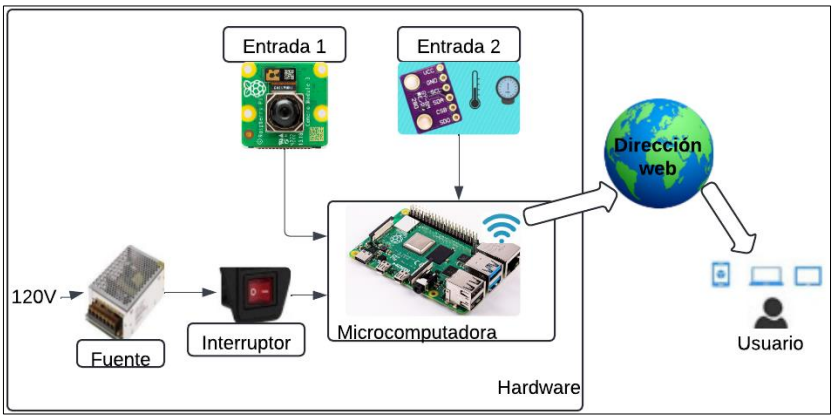
2.2 Descripción de la propuesta

2.2.1 Estructura general

En la Figura 4 se visualizan los dispositivos que se llevan a cabo para el desarrollo del sistema de monitoreo en tiempo real, con la utilización de VA y sensores IoT. En el Anexo 7 se ilustra a detalle el sistema eléctrico desde la conexión 120VAC hasta la microcomputadora.

Figura 4

Componentes del sistema de monitoreo del tráfico vehicular



Nota. Elaboración Propia

2.2.2 Explicación del aporte

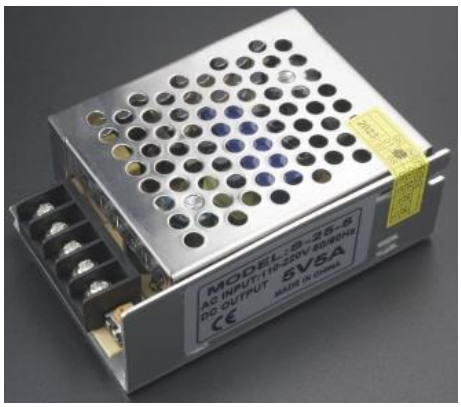
Funcionamiento de los componentes del sistema

Hardware

Como todo dispositivo electrónico, necesita de una fuente de alimentación (ver Figura 5). La entrada se conecta a una red monofásica (110 VAC), una salida de 5VDC y 5A. Adicional a ello, se dispone de un interruptor con fusible interno para encender/apagar la tarjeta Raspberry Pi, tal como se indica en la Figura 4.

Figura 5

Fuente de alimentación



Nota. Elaboración Propia

De la Tabla 2, se selecciona el sensor BMP280, el cual permite realizar mediciones de temperatura y presión. Presenta algunas características descritas en la Tabla 3.

Tabla 3

Características del sensor BMP 280

Parámetros	Valor	Unidad de medida	Figura
Presión (P)	300 – 1100	hPa	
Temperatura (T)	-40 a 85	°C	
Precisión P absoluta	± 1	hPa	
Precisión P relativa	± 0,12	hPa	
Precisión T	± 1	°C	
Nivel lógico	3,3	V	
Consumo	2,7	uA	
Interfaz	I2C, SPI		

Nota. Anexo 1

Se escoge una Raspberry Pi 5, la cual presenta un mejor rendimiento con respecto a las versiones anteriores. Por medio de la Tabla 4 se especifica los parámetros técnicos de esta tarjeta (Raspberry Pi Ltd., 2024). Esta microcomputadora se utiliza para desarrollar el sistema

de monitoreo del tráfico en tiempo real y recolectar, también, las mediciones ambientales de presión y temperatura en un punto de la vía en San Rafael.

Tabla 4

Características de la Raspberry Pi 5

Parámetros	Descripción	Figura
CPU Broadcom	BCM2717 de 4 núcleos	
SDRAM	2, 4 y 8 GB	
Wi-Fi	802.11ac doble banda	
Puertos USB 3.0/2.0	2/2	
Transceptores	2 de pantalla/cámara MIPI de 4 carriles	
Alimentación	5VDC, 5A (USB-C)	
Pines	40	
Sistema operativo	Raspberry Pi OS (basado en Linux)	
Almacenamiento	microSD	

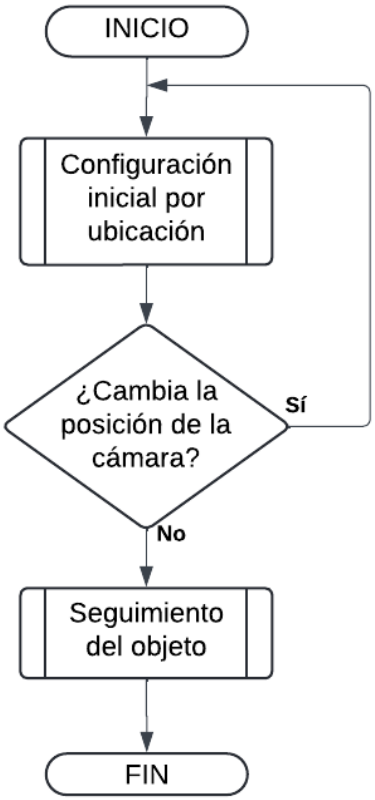
Nota. Adaptado de (Raspberry Pi Ltd., 2024)

Software

Este proyecto contempla 2 partes relacionadas al software, una referente a la programación en la Raspberry Pi y otra al servidor web. El primero consiste en aplicar un conjunto de librerías para realizar la detección del vehículo en tiempo real por medio de la cámara. En la Figura 6, se observa dos subrutinas, las cuales cada una posee un proceso interno. Cuando se coloca el dispositivo electrónico en cierta ubicación de una calle, por ejemplo, se lleva a cabo una configuración inicial. Luego, se realiza el seguimiento y obtención de la velocidad de los vehículos.

Figura 6

Programa general con la Raspberry Pi



Nota. Elaboración Propia

En la Figura 7 se muestra el procedimiento de la configuración inicial que se realiza ante una nueva posición. Este diagrama contiene el siguiente proceso de desarrollo:

Dibujar polígonos y crear ventanas en el primer fotograma

- El código lee el primer fotograma[1] del video y permite que el usuario dibuje líneas para delimitar el área de interés. Este sirve como referencia para la detección de vehículos y medición de las velocidades; para ello, se utilizan las librerías Picamera y OpenCV.
- Picamera que captura imágenes a través de la cámara de Raspberry Pi 5.

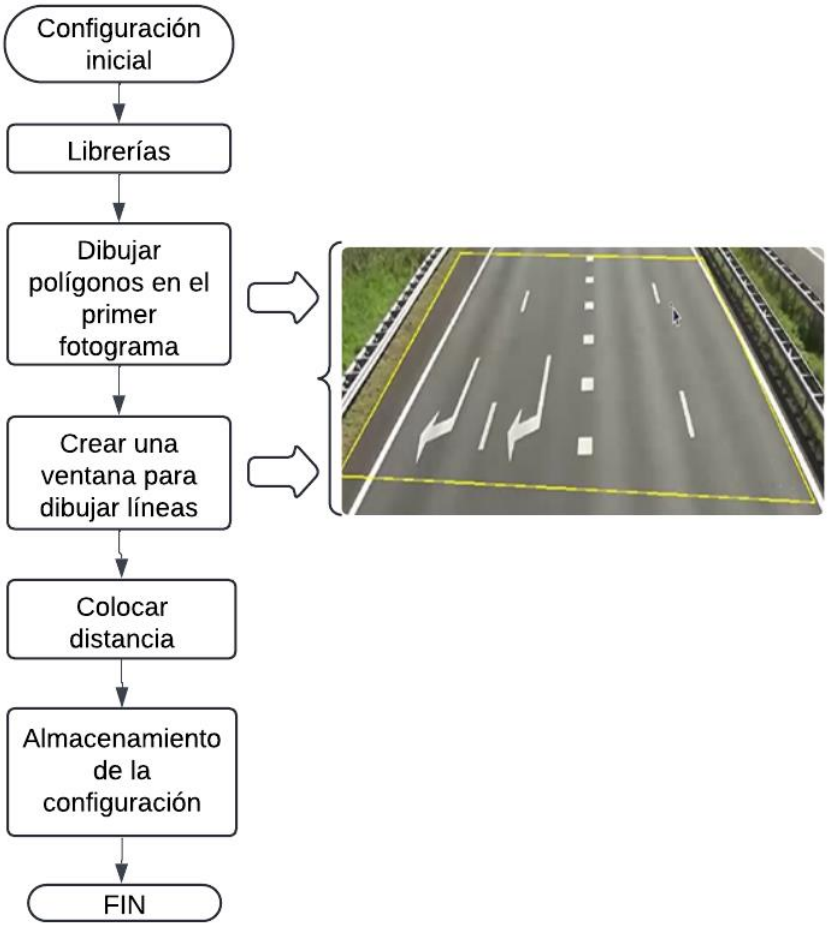
Colocar distancia y almacenamiento de la configuración

- Por consiguiente, el usuario ingresa la distancia física en metros entre los puntos de referencia del fotograma y se almacena esta información en un archivo con la librería Pickle. Por lo que se obtienen los parámetros necesarios para el proceso de seguimiento y cálculo de la velocidad del vehículo.

[1] Fotograma: secuencia del objeto en una imagen para formar un video.

Figura 7

Configuración inicial



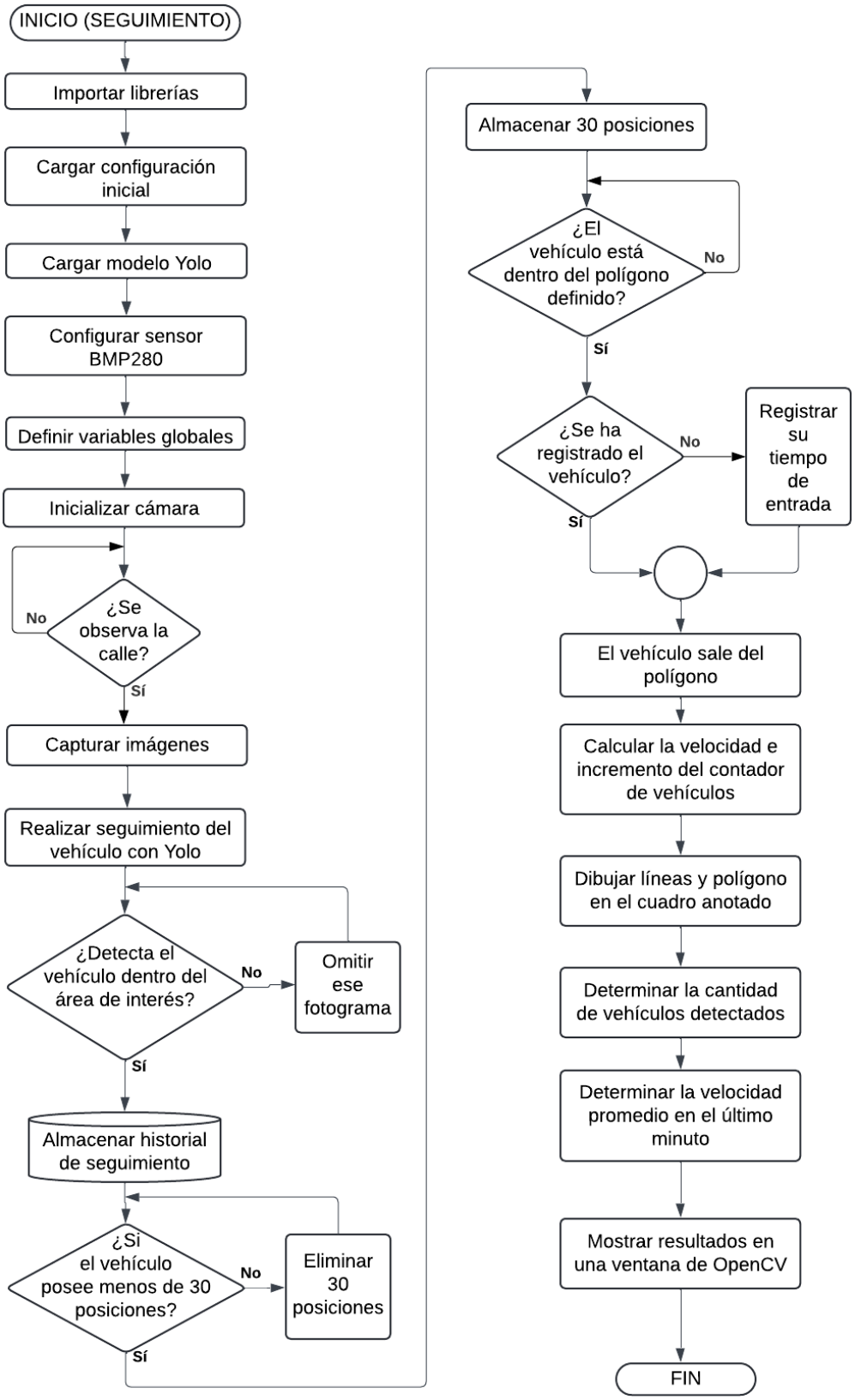
Nota. Elaboración Propia (ver Anexo 3)

En la Figura 8 se sigue un flujo de trabajo específico para obtener en esencia, dos parámetros ambientales y la velocidad de los vehículos que pasan a través de un polígono definido en la imagen de video. Para ello, se cargan las librerías necesarias para VA, captura de datos de sensores, procesamiento de imágenes y otras variables globales que ayudan al manejo de datos de los vehículos, su historial de posiciones y los parámetros del video. A continuación, se realiza el siguiente procedimiento:

- Se suben dos parámetros de la configuración inicial como el polígono del área de interés y las líneas de referencia desde un archivo con la librería pickle.
- Se carga el modelo ya entrenado Yolo, el cual sirve para la detección de los vehículos, por medio de la librería Ultralytics.
- Se configura el sensor BMP280 bajo la comunicación I2C para determinar los parámetros de presión y temperatura.
- Se inicializa la cámara y una vez que entra en funcionamiento, se capturan imágenes para ser procesadas en cada ciclo. Para ello, se aplican las librerías OpenCV y Picamera.

Figura 8

Seguimiento de vehículos en tiempo real



Nota. Elaboración Propia (ver Anexo 3)

- Con el modelo Yolo se realiza un seguimiento de los vehículos situados en las imágenes

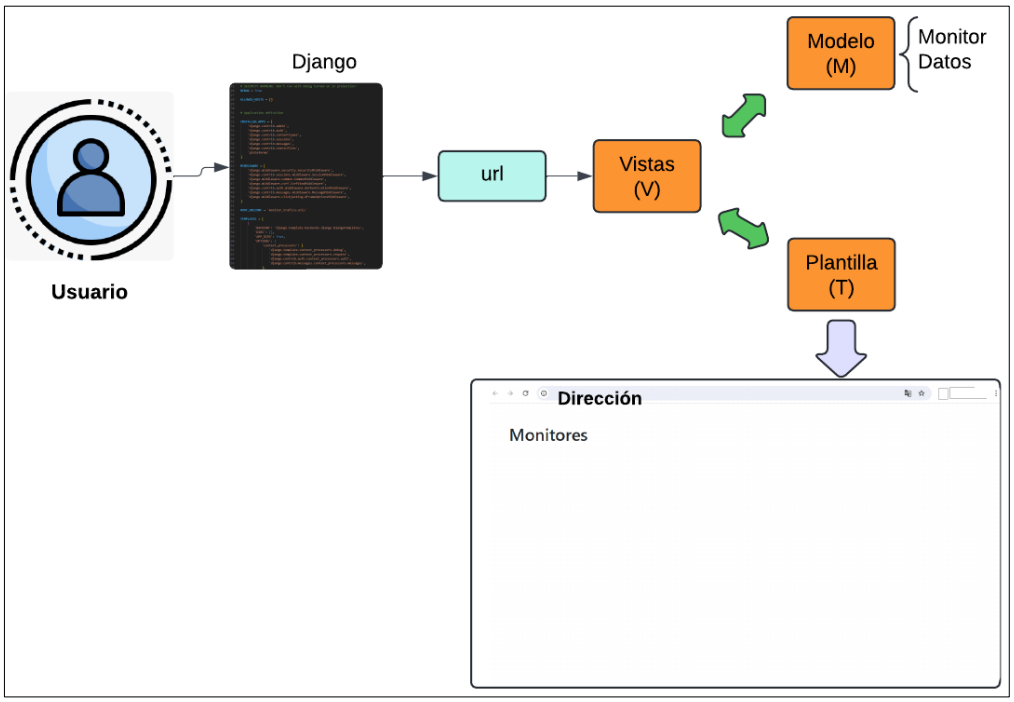
capturadas.

- Si el vehículo se encuentra dentro del polígono definido, se almacena el historial de posiciones en un diccionario para el rastreo de su desplazamiento con la librería "*collection.defaultdict*". Caso contrario, se omite ese fotograma y se pasa al siguiente hasta detectar los vehículos.
- Si el vehículo posee una cantidad menor a 30 posiciones, se almacena dicha información. De lo contrario, se eliminan las más antiguas para mantener un límite de registro. También se verifica nuevamente si el vehículo está dentro del polígono definido. Si no es así, se considera que ha salido del área de interés. Este último utiliza la librería "*shapely.geometry*".
- Una vez que se haya registrado el vehículo, este sale del polígono y se calcula el periodo transcurrido (librería "*time*") y la distancia recorrida.
- Se incrementa el contador de vehículos detectados y se calcula la velocidad una vez que haya salido del polígono. Estos utilizan las librerías "time" y "*collection.defaultdict*".
- Se dibujan las líneas y el polígono en la imagen procesada para su visualización. Esto se lleva a cabo, con las librerías *OpenCV* y *Numpy.*
- Se determina la cantidad de vehículos detectados y se promedia la velocidad. Por cada minuto, se envía la información recolectada a una API remota. Igualmente, las imágenes se indican en una pantalla, junto con información adicional como el conteo y velocidad promedio de los vehículos

El segundo programa (ver Figura 9) consiste en un servidor web, elaborado con Django en Python. Este aplica el modelo MVT para entregar la información del monitoreo de tráfico al usuario.

Figura 9

MVT propuesto



Nota. Elaboración Propia (ver Anexo 4)

Cada componente del modelo MVT presenta los siguientes atributos:

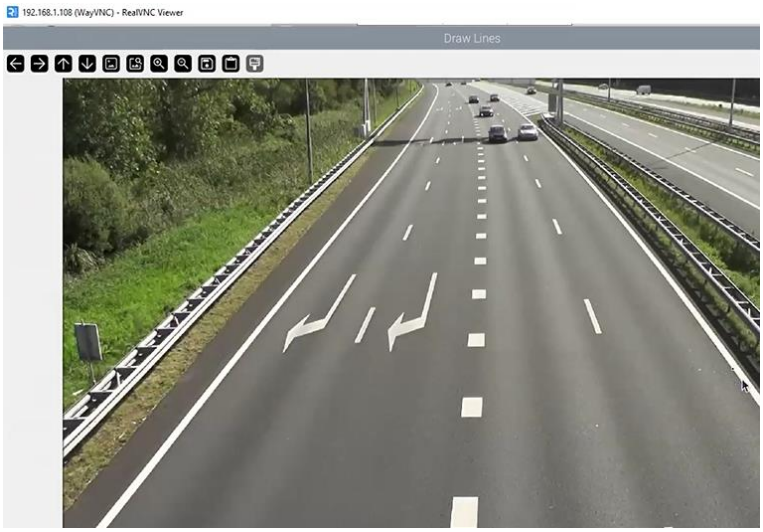
- Modelo
    - Monitor: capturan los datos de tráfico y se envía esta información a través de la vista “add_dato”. Los atributos son la ubicación (latitud, longitud), dirección y un identificador único.
    - Datos: hora, fecha, valor de velocidad, temperatura, presión y tráfico.
- Vistas
    - Add_dato: encargado de recibir los datos de tráfico del monitor. Almacena esta información en el monitor específico.
    - Proporciona una lista de todos los monitores registrados con la función “get_monitors”.
    - Devuelve los datos de tráfico en formato JSON, con la función “get_data”.
    - Tiene la capacidad de descargar los datos de tráfico en formato CSV, con la función “download_data”.
- Plantilla
    - El usuario ingresa a la página a través de una dirección HTTP.
    - Muestra la información del tráfico en tiempo real al usuario.
- Configura las rutas de acceso (URL) a las vistas.

- Como se habla de un servidor web, este dispone de un administrador, donde se gestionan los monitores y datos capturados. Conjuntamente, se configura en Ubuntu *DigitalOcean*, para APIs remota con Django y Python. Este entorno virtual viene optimizado para manejar solicitudes HTTP de manera robusta y segura gracias a las integraciones de Gunicorn y Nginx como servidores de aplicaciones y web; respectivamente, para distribuir la carga.

Para manejar la Raspberry Pi sin la necesidad de una pantalla, se recurre de forma remota con el VNC Viewer (ver Figura 10). Que utiliza el protocolo Virtual Network Computing (VNC), que transmite la pantalla del equipo remoto (celular o computador) al dispositivo del usuario (microcomputadora), el cual permite la interacción con el sistema operativo y las aplicaciones instaladas (Recalde Varela & Andrago Calvachi, 2019).

Figura 10

VNC Viewer



Nota. Elaboración Propia

Interactividad

Este diseño permite una interactividad significativa porque emplea la Raspberry Pi 5, cuya función consiste en detectar vehículos en movimiento en tiempo real con el uso de Ultralytics YOLOv8. Este proceso se genera a través de la detección y procesamiento de las entradas provenientes de los sensores y la cámara. A su vez, se transmite a los usuarios en una interfaz web, a fin de permitir el monitoreo remoto. Lo que resulta esencial para asegurar que los interesados puedan responder rápidamente a las detecciones y tomar decisiones informadas.

Recursos que apoyan los aprendizajes

El sistema utiliza varios recursos: una cámara como entrada 1 para capturar imágenes o videos de la escena, y un sensor adicional como entrada 2 (temperatura y presión) para

recopilar datos ambientales. La microcomputadora (Raspberry Pi) procesa estos datos, ejecuta el modelo YOLOv8 para la detección de vehículos, y transmite la información a través de una dirección web accesible para los usuarios. Estos recursos son cruciales para avalar que el sistema trabaje de manera eficiente y para poder acceder a los resultados en tiempo real.

Actividades de evaluación

Las actividades de evaluación contienen pruebas de velocidad del modelo YOLOv8 en la detección de vehículos. Donde se detecta un nivel de confianza, el cual se varía entre 0 a 1.

Construcción del conocimiento

Las actividades de construcción del conocimiento incluyen la programación y configuración del modelo YOLOv8 en la Raspberry Pi, la integración de los sensores, y la creación de un servidor web para la visualización de los resultados. Estas actividades permiten a los usuarios y desarrolladores entender cómo se puede aplicar la detección de objetos en un entorno de monitoreo en tiempo real.

2.3 Estrategias y/o técnicas

Estrategias metodológicas

Las estrategias metodológicas incluyen el aprendizaje profundo para la detección de vehículos mediante el modelo YOLOv8, que es reconocido por su alta precisión y eficiencia. Se selecciona la Raspberry Pi como microcomputadora por su capacidad de procesamiento adecuado para ejecutar modelos VPC y su compatibilidad con múltiples sensores y cámara. La metodología se enfoca en la integración de hardware y software para crear un sistema de monitoreo autónomo.

Herramientas tecnológicas

Las herramientas tecnológicas utilizadas incluyen la Raspberry Pi para el procesamiento central, una cámara de alta resolución para capturar imágenes, el sensor BMP280 para obtener mediciones de temperatura y presión en grados centígrados y hecto pascales; respectivamente, para recopilar datos ambientales. La selección de la Raspberry Pi 5 se basa en la accesibilidad y su capacidad para ejecutar modelos IA como YOLOv8. Se utiliza una fuente de alimentación de 110VAC y un interruptor para controlar el flujo de energía al sistema, para asegurar un funcionamiento continuo y confiable. Además, el servidor web (Ubuntu, *DigitalOcean*) permite a los usuarios acceder a los resultados desde cualquier dispositivo conectado a Internet y facilitar la supervisión remota.

2.4 Validación de la propuesta

Tabla 5

Perfil de los validadores

Nombres completos	Años de experiencia	Titulación	Cargo
Lenin Patricio Pacheco Tufiño	16	Magister en Gestión de Energías	Supervisor de Instrumentación y Control Ep Petroecuador
John Henry Carrión Palacios	16	Magister en Electrónica y Automatización	Técnico Líder en Instrumentación y Control Ep Petroecuador
Geovanny Alfonso Ashqui Carrasco	12	Ingeniero en Electrónica Automatización y Control	Supervisor de Instrumentación y Control Ep Petroecuador

Nota. Elaboración Propia

Para llevar a cabo la validación del proyecto de titulación, se evalúa su alto grado de importancia al aplicar la escala de Likert, en base a ciertos criterios de valuación. En la Tabla 6 se indica el grado de impacto, aplicabilidad, conceptualización, actualidad, calidad técnica, factibilidad y pertinencia, cuyos conceptos se definen en el Anexo 2.

Tabla 6

Escala de evaluación según el Mg. Lenin Pacheco

Criterios	Valuación				
	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Impacto				X	
Aplicabilidad				X	
Conceptualización				X	
Actualidad			X		
Calidad técnica				X	
Factibilidad				X	
Pertinencia				X	

Nota. Elaboración Propia

Tabla 7

Escala de evaluación según el Mg. John Carrión

Criterios	Valuación				
	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Impacto			X		
Aplicabilidad				X	
Conceptualización			X		
Actualidad				X	
Calidad técnica				X	
Factibilidad				X	
Pertinencia				X	

Nota. Elaboración Propia

Tabla 8

Escala de evaluación según el Ing. Geovanny Ashqui

Criterios	Valuación				
	Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Impacto				X	
Aplicabilidad				X	
Conceptualización			X		
Actualidad				X	
Calidad técnica				X	
Factibilidad				X	
Pertinencia			X		

Nota. Elaboración Propia

2.5 Matriz de articulación de la propuesta

A través de la Tabla 9 se detalla la matriz de articulación del tema “Diseño e implementación un sistema de monitoreo de tráfico en tiempo real que utilice visión artificial y sensores IoT”, el cual emplea todos los sustentos teórico – prácticos.

Tabla 9

Matriz de articulación

	Partes fundamentales	Descripción breve de los resultados	Sustento teórico	Metodologías y/o herramientas aplicadas
1	Definición de elementos (hardware) y software	1.1 Características de elementos hardware 1.2 Estructura de plataforma web 1.3 Técnicas VA	Se lleva a cabo una investigación bibliográfica sobre los componentes que conforman el "diseño e implementación un sistema de monitoreo de tráfico en tiempo real que utilice VA y sensores IoT", a nivel de hardware y software.	Investigación bibliográfica
2	Diseño (hardware y software)	2.1 Selección de dispositivos (hardware) 2.2 Comunicaciones entre dispositivos 2.3 Aplicación de técnicas VA	Se realiza una investigación bibliográfica en base a los elementos disponibles en el mercado y también lenguajes de programación y protocolos de comunicación que se adecúen a los dispositivos seleccionados.	Catálogos técnicos (BMP280 y Raspberry Pi), Python, OpenCV, Yolov8 OceanDigital, Django
3	Implementación del sistema de monitoreo de tráfico	3.1 Prototipo implementado 3.2 Pruebas experimentales 3.3 Validación del proyecto	Se realizan pruebas iniciales por partes, es decir, Raspberry Pi y servidor web. Luego, se realizan pruebas experimentales en campo del prototipo para determinar la velocidad promedio, temperatura, presión y tráfico.	Pruebas experimentales de varios días

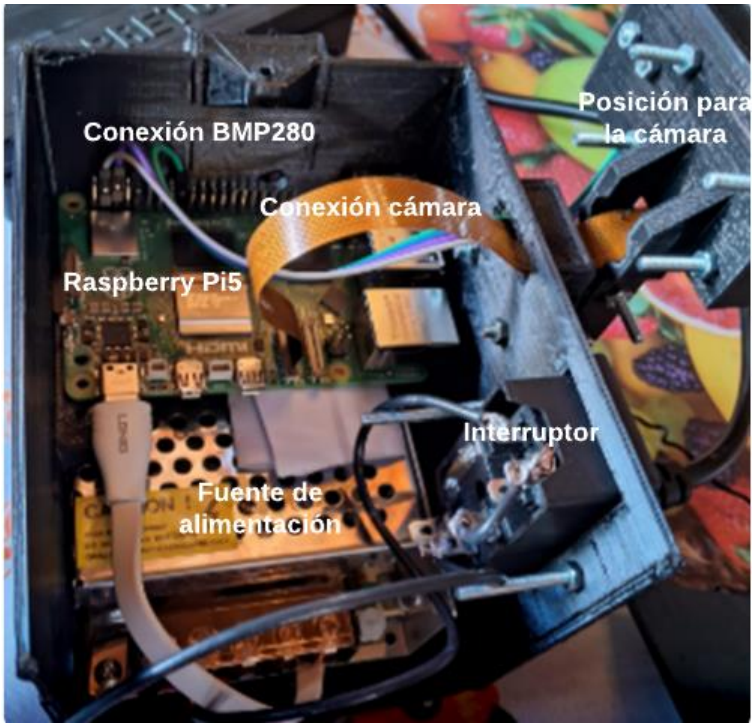
Nota. Elaboración Propia

2.6 Análisis de resultados

A través de la Figura 11 se visualiza los dispositivos electrónicos como la Raspberry Pi, fuente de alimentación, interruptor encendido/apagado, sensor bmp280 y cámara con sus respectivas conexiones.

Figura 11

Prototipo del sistema de monitoreo de tráfico vehicular

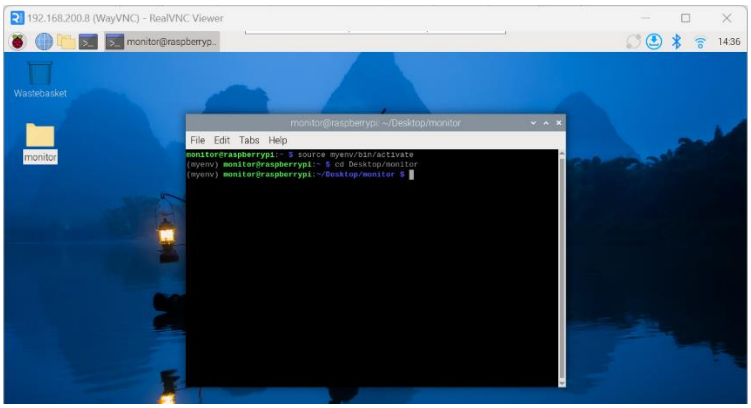


Nota. Anexo 6

Una vez que se encuentra el dispositivo armado, se ejecutan los scripts (líneas de comando) para efectuar el monitoreo del tráfico, tanto en la configuración inicial como en el programa principal, visto en la Figura 12 y descrito su procedimiento en el manual de usuario.

Figura 12

Ejecución de scripts en la Raspberry Pi 5

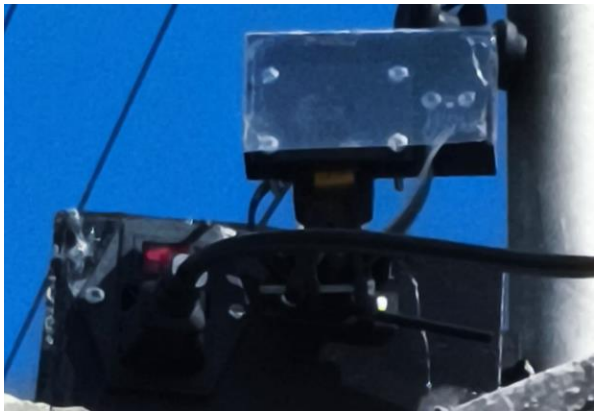


Nota. Elaboración Propia

Se sitúa el prototipo electrónico en el Conjunto San Nicolás, ubicado en la Av. General Rumiñahui 960, tal como se visualiza en la Figura 13.

Figura 13

Ubicación del prototipo en San Rafael



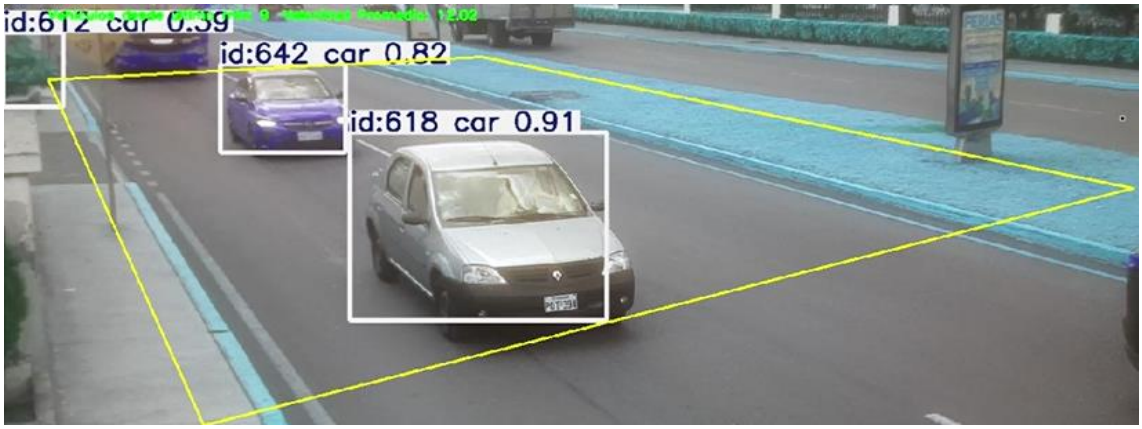
Nota. Elaboración Propia

La Figura 14 indica un esquema visual de la detección de vehículos en movimiento sobre la Av. General Rumiñahui, mediante el sistema de reconocimiento basado en Ultralytics YOLOv8. Cada vehículo viene etiquetado con un identificador único y un nivel de confianza. El que se sitúa en primer plano, reconocido como "id:618", tiene un nivel de confianza de 0,91, lo que muestra

una alta probabilidad de que sea más segura la detección del objeto. Mientras que otros 2 vehículos etiquetados como "id:642" e “id:612” poseen niveles de confianza iguales a 0,82 y 0,59.

Figura 14

Identificación de vehículos livianos



Nota. Elaboración Propia

Pruebas experimentales (datos por minuto)

Mediante la Raspberry Pi 5, se recopilan los parámetros de velocidad del vehículo, presión, temperatura y tráfico en un tramo de la Av. General Rumiñahui, a la altura del Conjunto San Nicolás. La microcomputadora recibe información cada minuto y genera un promedio de 5.590 datos diarios por los cuatro parámetros indicados en la Tabla 10. Estos se recolectan de forma continua durante los días 24 al 27 de agosto.

Tabla 10

Recepción de los parámetros

Agosto	Horas		Datos	Velocidad	Temperatura	Presión	Tráfico
2024	Inicio	Fin		Km/h	°C	hPa	Vehículos/min.
23	8:53	23:59	1.760	0 – 202	13 – 26	757 - 762	0 – 21
24	0:00	23:59	5.444	0 – 186	13 – 36	757 – 761	0 – 56
25	0:00	23:59	5.708	5 – 224	12 – 36	758 – 762	0 – 60
26	0:00	23:59	5.704	6 – 203	13 – 37	757 – 762	0 – 53
27	0:00	23:59	5.504	0 – 214	13 – 39	757 – 761	0 – 63
28	0:00	14:56	3.552	3 – 51	9 – 38	757 – 760	0 – 33

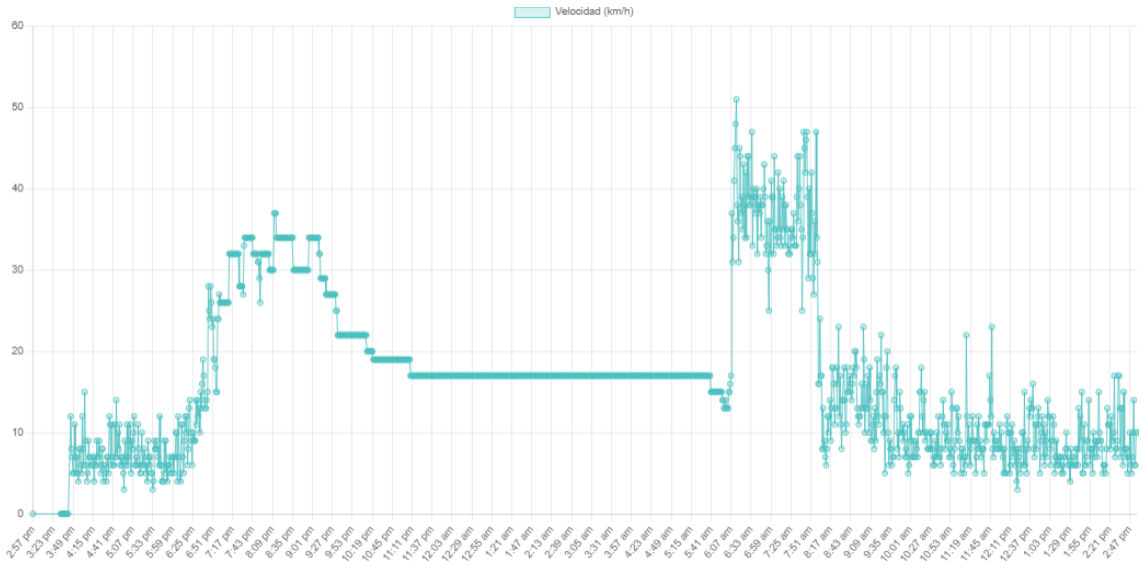
Nota. Elaboración Propia

De las pruebas realizadas, se escoge una muestra un día de prueba, en este caso se toma como referencia del día 27 al 28 de agosto.

La Figura 15 detalla la medición de la velocidad promedio expresada en km/h. En un inicio, la velocidad registrada es baja, cuyo valor es de 10 km/h aproximadamente. Posteriormente, hay un incremento en su valor que alcanza picos entre 30 - 40 km/h a las 6:51 y 9:27 pm del 27 de agosto. Después de este pico, la velocidad se estabiliza en un rango más bajo, y durante un período prolongado, se mantiene constante alrededor de 17 km/h, con pocas fluctuaciones. Sin embargo, a partir de las 6:00 am, se observa una nueva variabilidad con valores que suben y bajan de manera irregular hasta el final del período de observación. Esta variabilidad es especialmente pronunciada en horas de la mañana del día siguiente, donde se observan picos y descensos abruptos, para luego disminuir gradualmente sus valores hasta el final del periodo.

Figura 15

Monitoreo de la velocidad promedio de vehículos

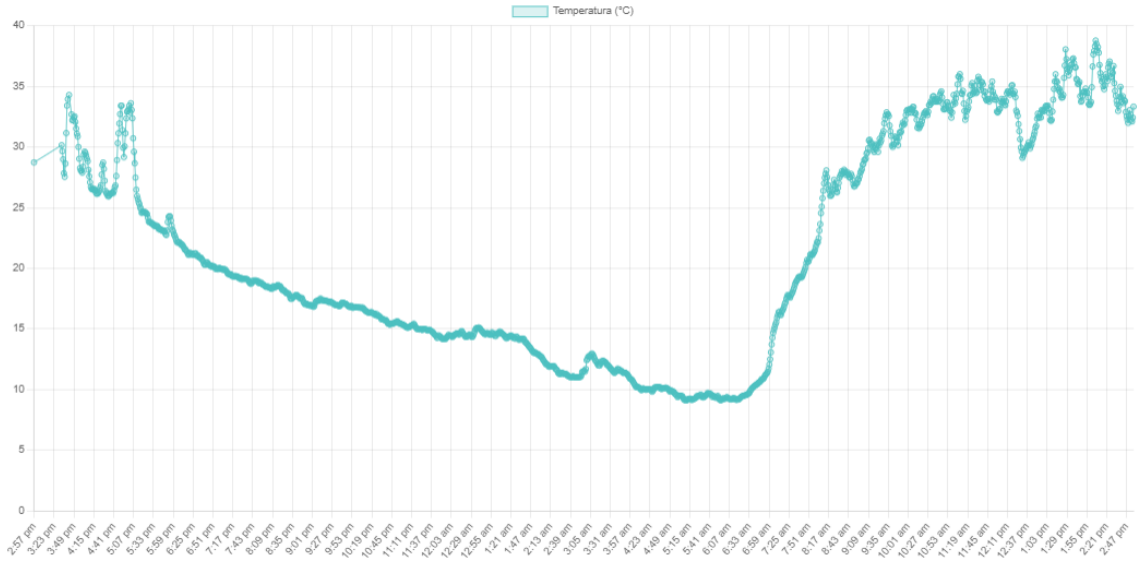


Nota. Elaboración Propia

La Figura 16 muestra el cambio de temperatura registrada por el monitor bajo las mismas horas de la medición anterior. Al inicio del periodo, la temperatura se encuentra alrededor de 30°C, con fluctuaciones leves. Posteriormente, se observa como baja la temperatura en el transcurso de la noche del 27 de agosto hasta llegar aproximadamente a los 10°C en horas de la mañana del 28 de agosto (7 am). A partir de ese punto, la temperatura comienza a aumentar de manera gradual y consistente hasta alcanzar un nuevo pico, que supera los 35°C alrededor de las 12 pm. Después de este aumento, la temperatura muestra pequeñas fluctuaciones en torno a dicho valor, manteniéndose relativamente estable hasta el final del periodo de observación. Los valores de temperatura superiores a 30°C pueden ocasionar dos consecuencias importantes. En primer lugar, el asfalto puede ablandarse debido a la fricción de los neumáticos contra la superficie, lo que incrementa el desgaste y provoca deformaciones en la carretera. En segundo lugar, estas condiciones pueden afectar el comportamiento de los conductores, quienes podrían sentirse fatigados o incómodos durante el manejo del vehículo.

Figura 16

Monitoreo de la temperatura ambiental



Nota. Elaboración Propia

Con respecto a la presión atmosférica se sitúa en un valor constante entre 757 a 762 hPa (ver Figura 17), equivalente a 0,747 a 0,752 atm; esto sucede durante el periodo de observación antes mencionado.

Figura 17

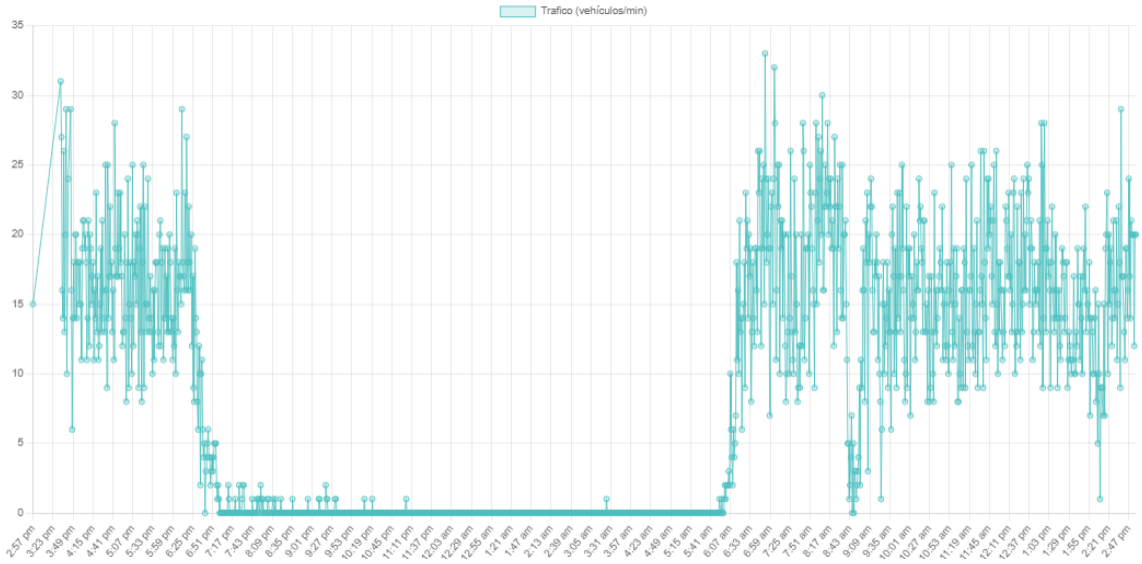
Monitoreo de la presión atmosférica

Nota. Elaboración Propia

En la Figura 18 se detalla el monitoreo del tráfico, como una representación de los vehículos por minuto que circulan en la Avenida. Se observa como el tráfico varía entre horas, por lo que se observa un tránsito ligero entre las 6pm a 6am. A partir de esta última hora se intensifica el tráfico hasta las 8:30 am, 9am – 1pm y 2pm – 6pm.

Figura 18

Monitoreo del tráfico

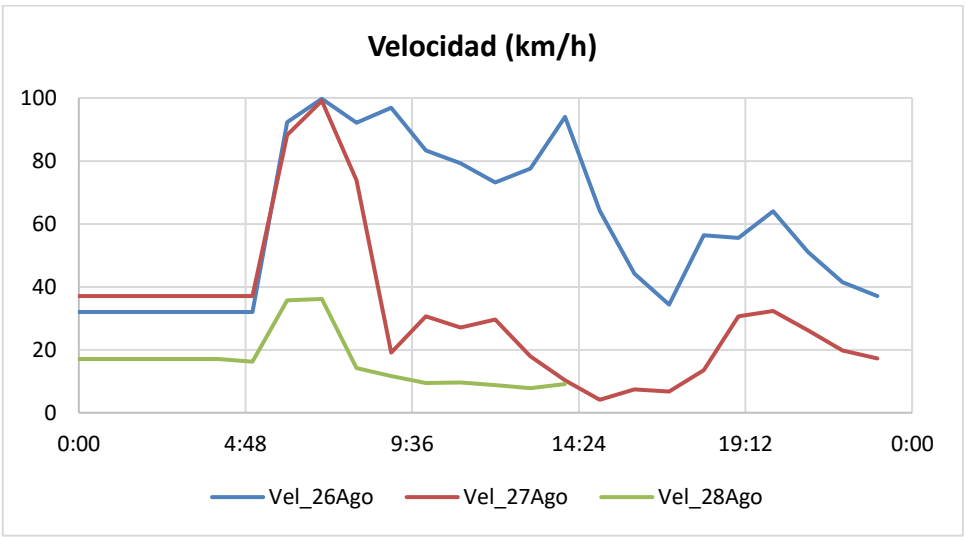


Nota. Elaboración Propia

En la Figura 19 se muestra la variación de velocidad dada en km/h desde las 0:00 del 26 de agosto hasta las 2pm del 28 de agosto. En los días 26 y 27, se observa cómo alcanza un promedio cercano a los 100 km/h a las 7am, mientras que para el resto de las horas difiere en sus velocidades y muestra una mayor constancia a altas velocidades, el día lunes 26 de agosto.

Figura 19

Comparación de la velocidad

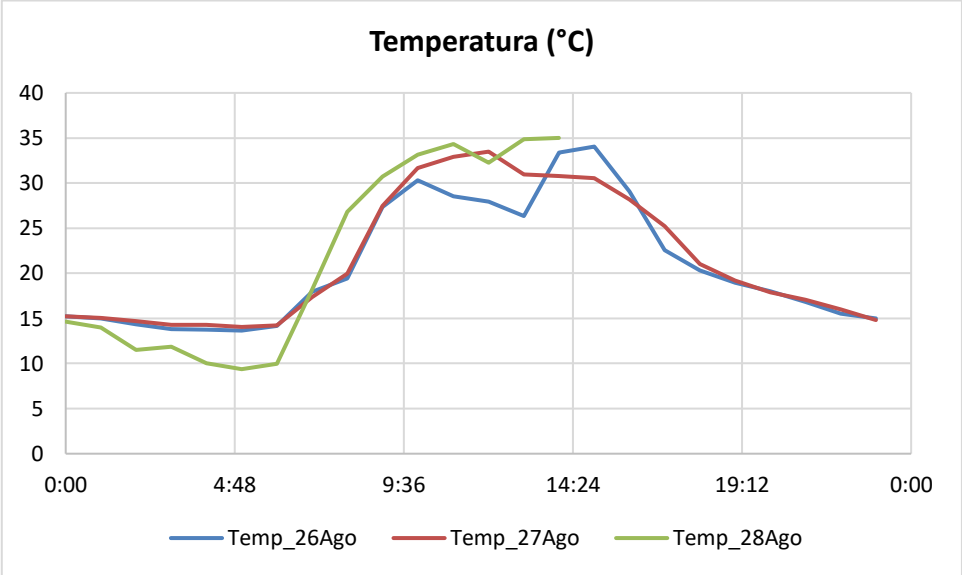


Nota. Elaboración Propia

Con respecto a la temperatura ambiente, se observa (ver Figura 20) cómo son de similares características para los tres días y se determina altas temperaturas mayores a 25°C de 8am a 5pm.

Figura 20

Comparación de temperatura

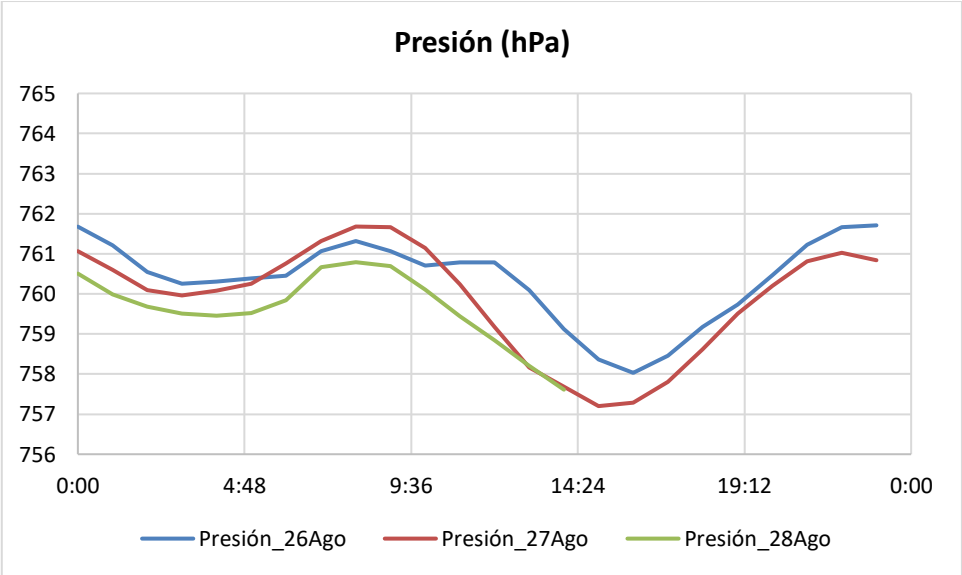


Nota. Elaboración Propia

La presión atmosférica oscila entre 757 a 762 hPa, cuya diferencia no es mayor a 5 hPa en los tres días establecidos, visto en la Figura 21.

Figura 21

Comparación de la presión atmosférica



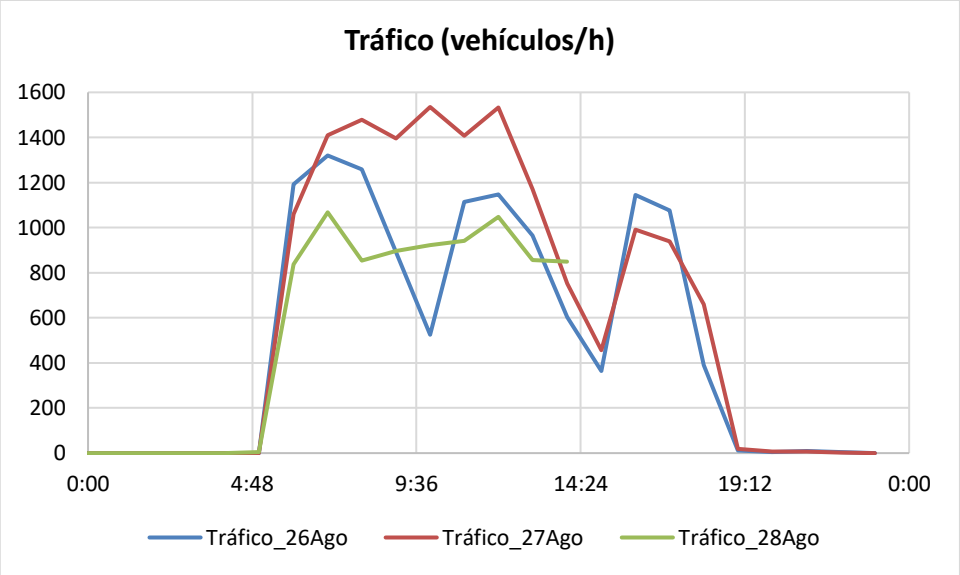
Nota. Elaboración Propia

Se muestra el tráfico vehicular en la Figura 22, donde se observa una gran cantidad de vehículos por hora que alcanza aproximadamente los 1500 vehículos entre las 7am a 12pm del día martes 27 de agosto. Luego, el tráfico desciende hasta los 1000 vehículos/h. Se visualiza

que para los días restantes el tránsito es menor. Por último, se detalla que existe un tránsito ligero de 7pm a 5am.

Figura 22

Comparación del tráfico vehicular



Nota. Elaboración Propia

A nivel estadístico, se determinan la varianza, valores promedio, mínimo y máximos por día (ver Tabla 11). A medida que la varianza se aleja de promedio, indica una mayor dispersión del tráfico ese día, es decir, están más dispersos. En los tres días sucede una gran dispersión, siendo el de mayor fluctuación el día 27 de agosto, el cual posee una varianza de 400.668 con un promedio de 618 vehículos por hora.

La media del tráfico típico varía por día, cuyo rango oscila entre 501 a 618 vehículos por hora del 26 al 28 de agosto. Se determinan valores nulos en horas tempranas y tardías; mientras que sus valores máximos son de 1.320, 1.535 y 1.068 vehículos/hora.

Tabla 11

Datos estadísticos del tráfico (vehículos/hora)

Datos	26/Agosto	27/Agosto	28/Agosto
Varianza	278.215	400.668	221.149
Promedio	501	618	552
Mínimo	0	0	0
Máximo	1.320	1.535	1.068

Nota. Elaboración Propia

CONCLUSIONES

La revisión del estado del arte ayuda a identificar y evaluar algunas herramientas que incluyen el uso del IoT, IA y visión artificial con Yolo. Este último, es un modelo que ocupa una arquitectura de capas neuronales y forma parte de la IA. Generalmente, los sensores IoT de bajo costo involucran una adaptación del sensor, sistema controlador (microcomputadora) y el medio de comunicación ya sea Wifi, bluetooth, LoRa, entre otros.

El prototipo electrónico se encuentra ubicado en la Av. General Rumiñahui. Este integra una serie de elementos como el sensor ambiental bmp280 para medir temperatura y presión atmosférica, Raspberry Pi 5 con un conjunto de herramientas para la detección de objetos y procesamiento de imágenes como lo es Yolo y OpenCV; respectivamente. Se utiliza Django para la gestión de la interfaz de usuario y el servidor web.

La validación del sistema se realiza a través de pruebas experimentales durante tres días, con el cual se puede verificar el grado de efectividad del prototipo desarrollado. Los datos recopilados, que incluyen la velocidad promedio de los vehículos, la temperatura, la presión y la densidad de tráfico, mostraron coherencia ya que fueron evaluados durante las 24 horas del día. Estos resultados evidencian que la plataforma es capaz de proporcionar información útil y fiable para la gestión del tráfico en tiempo real.

RECOMENDACIONES

Se recomienda a trabajos futuros revisar la literatura de los últimos avances tecnológicos para que puedan ser adaptados al presente proyecto, en caso de que se desee realizar mejoras, como, por ejemplo, una semaforización inteligente. Una implementación de este tipo, donde se ajusten los tiempos de cambio en función del flujo vehicular registrado mediante sensores y VA puede optimizar significativamente la movilidad en áreas urbanas congestionadas. Mediante el uso de algoritmos IA permitirá prever patrones de tráfico, anticipar picos de congestión y ajustar de manera dinámica la duración de las señales.

Al ser una cámara para la Raspberry Pi 5 cuya capacidad máxima es de 5 MPX, se recomienda realizar pruebas experimentales entre las 6am a 6pm.

Se recomienda ejecutar el programa de configuración inicial, cada vez que se cambie de posición el prototipo electrónico.

BIBLIOGRAFÍA

Aguilar, L. J. (2021). *Internet de las cosas: Un futuro hiperconectado: 5G, inteligencia artificial, Big Data, Cloud, Blockchain y ciberseguridad* (1ra Ed.). Marcombo.

Barbara. (2024). *Protocolos de comunicación IoT que debes conocer—Barbara*. https://www.barbara.tech/es/blog/protocolos-iot-que-deberias-conocer

Caiza Oña, D. W. (2016). *Diseño y construcción de un prototipo de sistema de semaforización inteligente* [bachelorThesis, Universidad Israel]. http://repositorio.uisrael.edu.ec/handle/47000/1210

Cortijo, R., & Arellano, L. (2020). *Sistema de video vigilancia mediante visión por computador para el Centro de Educación Inicial N°1 del Ministerio de Educación* [bachelorThesis, Universidad Israel]. http://repositorio.uisrael.edu.ec/handle/47000/2433

Domingo, J. D., García-Bermejo, J. G., & Casanova, E. Z. (2024). *Visión Artificial. Componentes de los sistemas de visión y nuevas tendencias en Deep Learning*. Ra-Ma Editorial.

@EspiFreelancer. (2019, agosto 3). *Que es el patrón MTV (Model Template View)*. EspiFreelancer. https://espifreelancer.com/mtv-django.html

Ganchala Quishpe, F. S. (2019). *Sistema de semaforización actuado, mediante cámara de video detección vehicular, para proyectos de movilidad en la Empresa Industrias Seblan Cia Ltda* [bachelorThesis, Universidad Israel]. http://repositorio.uisrael.edu.ec/handle/47000/2151

Halfacree, G. (2024). *La guía oficial de Raspberry Pi para principiantes: Cómo usar tu nuevo ordenador* (5ta Ed.). Raspberry Pi Press.

Quintana Tenorio, G. F. (2024). *Implementación de una interfaz HMI basado en software libre Node Red para el control, monitoreo de forma local y remota del sistema de bombeo en el conjunto habitacional Rivotorto.* [masterThesis, Universidad Israel]. http://repositorio.uisrael.edu.ec/handle/47000/4060

Raspberry Pi Ltd. (2024). *Raspberry Pi*. Raspberry Pi. https://www.raspberrypi.com/

Recalde Varela, P., & Andrago Calvachi, M. (2019). *Uso de reconocimiento facial de emociones basado en técnicas de Deep Learning para el mejoramiento de la educación.* [masterThesis, Universidad Israel]. http://repositorio.uisrael.edu.ec/handle/47000/2297

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection* (arXiv:1506.02640). arXiv. http://arxiv.org/abs/1506.02640

Slimani, I., Zaarane, A., & Atouf, I. (2023). Traffic Monitoring System for Vehicle Detection in Day and Night Conditions. *Transport and Telecommunication Journal*, *24*(3), 256–265. https://doi.org/10.2478/ttj-2023-0020

Suárez, C., Gaona, P., Soto, S., & Montenegro, C. (2020). *Modelo de comunicación basado en IoT para la transmisión de datos de estaciones meteorológicas* (1ra Ed.). Editorial Universidad Distrital Francisco José de Caldas.

Swathi, P., Tejaswi, D. S., Khan, M. A., Saishree, M., Rachapudi, V. B., & Anguraj, D. K. (2024). Real-Time Vehicle Detection for Traffic Monitoring: A Deep Learning Approach. *Data and Metadata*, *3*, 295–295. https://doi.org/10.56294/dm2024295

Tierra, J. (2020, julio 27). *Pytorch Vs Tensorflow Vs Keras: Here are the Difference You Should Know*. Simplilearn.com. https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article

Ultralytics. (2024). *Quick Start Guide: Raspberry Pi with Ultralytics YOLOv8*. https://docs.ultralytics.com/guides/raspberry-pi

W3Schools. (2024). *Introduction to Django*. https://www.w3schools.com/django/django_intro.php

ANEXOS

Anexo 1. Especificaciones técnicas del sensor BMP280

SENSOR BMP 280

BMP280

DIGITAL PRESSURE SENSOR

Key parameters

Pressure range	300 ... 1100 hPa (equiv. to +9000...-500 m above/below sea level)
Package	8-pin LGA metal-lid Footprint : 2.0 × 2.5 mm², height: 0.95 mm
Relative accuracy (700 ... 900hPa @25°C)	±0.12 hPa, equiv. to ±1 m
Absolute accuracy (950 ...1050 hPa, 0 ...+40 °C)	typ. ±1 hPa
Temperature coefficient offset 1.5 Pa/K, equiv. to 12.6 cm/K (25 ... 40°C @900hPa)	
Digital interfaces	I²C (up to 3.4 MHz) SPI (3 and 4 wire, up to 10 MHz)
Current consumption	2.7µA @ 1 Hz sampling rate
Temperature range	-40 ... +85 °C
RoHS compliant, halogen-free	
MSL 1	

Typical applications

- Enhancement of GPS navigation
  (e.g. time-to-first-fix improvement, dead-reckoning, slope detection)
- Indoor navigation (floor detection, elevator detection)
- Outdoor navigation, leisure and sports applications
- Weather forecast
- Vertical velocity indication (e.g. rise/sink speed)

Target devices

- Handsets such as mobile phones, tablet PCs, GPS devices
- Navigation systems
- Home weather stations
- Flying toys
- Watches

General Description

Robert Bosch is the world market leader for pressure sensors in automotive and consumer applications. Bosch's proprietary APSM (Advanced Porous Silicon Membrane) MEMS manufacturing process is fully CMOS compatible and allows a hermetic sealing of the cavity in an all silicon process. The BMP280 is based on Bosch's proven Piezo-resistive pressure sensor technology featuring high EMC robustness, high accuracy and linearity and long term stability.

The BMP280 is an absolute barometric pressure sensor especially designed for mobile applications. The sensor module is housed in an extremely compact 8-pin metal-lid LGA package with a footprint of only 2.0 × 2.5 mm² and 0.95 mm package height. Its small dimensions and its low power consumption of 2.7 µA @1Hz allow the implementation in battery driven devices such as mobile phones, GPS modules or watches.

As the successor to the widely adopted BMP180, the BMP280 delivers high performance in all applications that require precise pressure measurement. The BMP280 operates at lower noise, supports new filter modes and an SPI interface within a footprint 63% smaller than the BMP180.

The emerging applications of indoor navigation, fitness as well as GPS refinement require a high relative accuracy and a low TCO at the same time. BMP180 and BMP280 are perfectly suitable for applications like floor detection since both sensors feature excellent relative accuracy is ±0.12 hPa, which is equivalent to ±1 m difference in altitude. The very low offset temperature coefficient (TCO) of 1.5 Pa/K translates to a temperature drift of only 12.6 cm/K.
Please contact your regional Bosch Sensortec partner for more information about software packages enhancing the calculation of the altitude given by the BMP280 pressure reading.

Table 1: Comparison between BMP180 and BMP280

Parameter	BMP180	BMP280
Footprint	3.6 × 3.8 mm	2.0 × 2.5 mm
Minimum V_{DD}	1.80 V	1.71 V
Minimum V_{DDIO}	1.62 V	1.20 V
Current consumption @3 Pa RMS noise	12 µA	2.7 µA
RMS Noise	3 Pa	1.3 Pa
Pressure resolution	1 Pa	0.16 Pa
Temperature resolution	0.1°C	0.01°C
Interfaces	I²C	I²C & SPI (3 and 4 wire, mode '00' and '11')
Measurement modes	Only P or T, forced	P&T, forced or periodic
Measurement rate	up to 120 Hz	up to 157 Hz
Filter options	None	Five bandwidths

Anexo 2. Aspectos de los criterios de valuación

Criterios	Descripción
Impacto	Representa el alcance que tendrá el modelo de gestión y su representatividad en la generación de valor público.
Aplicabilidad	La capacidad de implementación del modelo considerando que los contenidos de la propuesta sean aplicables
Conceptualización	Los componentes de la propuesta tienen como base conceptos y teorías propias de la gestión por resultados de manera sistémica y articulada.
Actualidad	Los contenidos de la propuesta consideran los procedimientos actuales y los cambios científicos y tecnológicos que se producen en la nueva gestión pública.
Calidad Técnica	Miden los atributos cualitativos del contenido de la propuesta.
Factibilidad	Nivel de utilización del modelo propuesto por parte de la Entidad.
Pertinencia	Los contenidos de la propuesta son conducentes, concernientes y convenientes para solucionar el problema planteado.

Anexo 3. Software de la Raspberry

Configuración inicial

```
mport cv2
import numpy as np
from shapely.geometry import Point, Polygon
import time  # Import the time module
from picamera2 import Picamera2, Preview

# Initialize Picamera2
picam2 = Picamera2()
camera_config = picam2.create_still_configuration(main={"size": (1280, 720)},
controls={"FrameDurationLimits": (20000, 20000)})
picam2.configure(camera_config)
picam2.start()
image = picam2.capture_array()

# Variables for drawing the polygon
line1 = []
line2 = []
polygon = None

def draw_line(event, x, y, flags, param):
    global line1, line2, polygon
    if event == cv2.EVENT_LBUTTONDOWN:
        if not line1:
            line1.append((x, y))
        elif len(line1) == 1:
            line1.append((x, y))
        elif not line2:
            line2.append((x, y))
        elif len(line2) == 1:
            line2.append((x, y))
            polygon = Polygon([line1[0], line1[1], line2[1], line2[0]])
        print(f'Line points: {line1}, {line2}')
```

```
# Create a window to draw lines
cv2.imshow('Draw Lines', image)
cv2.setMouseCallback('Draw Lines', draw_line)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Prompt user for distance
distance = float(input("Ingresar distancia de medicion en m: "))

import pickle

# Save the distance and polygon to a pickle file
with open('config.pkl', 'wb') as f:
    pickle.dump({'distance': distance, 'polygon': polygon , 'line1':line1 ,'line2':line2   }, f)
print("Distance and polygon saved to config.pkl")
```

Seguimiento del vehículo

```
import pickle
import cv2
import numpy as np
from shapely.geometry import Point, Polygon
from ultralytics import YOLO
import time
import threading
import requests
import board
import busio
from adafruit_bmp280 import Adafruit_BMP280_I2C
from collections import defaultdict
from picamera2 import Picamera2, Preview
from datetime import datetime, timedelta
```

```
# Load the polygon and distance from pickle
with open('config.pkl', 'rb') as f:
    config = pickle.load(f)
distance = config['distance']
polygon = config['polygon']
line1 = config['line1']
line2 = config['line2']

# Load the exported NCNN model
ncnn_model = YOLO("yolov8n_ncnn_model")

# Initialize I2C bus and BMP280 sensor
i2c = busio.I2C(board.SCL, board.SDA)
bmp280 = Adafruit_BMP280_I2C(i2c, address=0x76)

# Shared variables for video frames and tracking
frame_buffer = None
frame_lock = threading.Lock()
track_history = defaultdict(lambda: [])
vehicle_speeds = {}
vehicle_counter = 0
avgSpeed=0

def update_vehicle_counter():
    global vehicle_counter,avgSpeed
    while True:
        time.sleep(60)  # Wait for 1 minute
        # Send vehicle_counter to the API
        api_url = "http://ticsnic.com/api/add_dato/"

        current_time = datetime.now()
        temperature = bmp280.temperature
        pressure = bmp280.pressure
```

```
        data = {
            "idmonitor": "mon01",  # Assuming you have a monitor with id "M123"
            "hora": current_time.time().strftime("%H:%M:%S"),
            "fecha": current_time.date().strftime("%Y-%m-%d"),
            "velocidad": round(avgSpeed, 2),
            "temperatura": round(temperature, 2),
            "presion": round(pressure, 2),
            "humedad_relativa": 0,
            "trafico": vehicle_counter
        }
        response = requests.post(api_url, json=data)
        if response.status_code == 201:
            print("Data added successfully:", data)
        else:
            print("Failed to add data:", response.text)
        # Reset the vehicle counter
        vehicle_counter = 0

# Start the vehicle counter reset thread
reset_thread = threading.Thread(target=update_vehicle_counter)
reset_thread.daemon = True
reset_thread.start()
#cap = cv2.VideoCapture(0)  # Use 0 for the default camera
# Initialize Picamera2
picam2 = Picamera2()
camera_config = picam2.create_still_configuration(main={"size": (1280, 720)},
controls={"FrameDurationLimits": (20000, 20000)})

picam2.configure(camera_config)
picam2.start()

while True:
    frame = picam2.capture_array()
    # Perform tracking
    results = ncnn_model.track(frame, persist=True)
```

```
if results[0].boxes.id is None:
    cv2.imshow('Draw Lines', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    continue  # Skip this frame if no objects are tracked

boxes = results[0].boxes.xywh.cpu()
track_ids = results[0].boxes.id.int().cpu().tolist()
annotated_frame = results[0].plot()
class_labels = results[0].boxes.cls.int().cpu().tolist()  # Assuming 'cls' contains class labels

# Update tracking history
for box, track_id,class_label  in zip(boxes, track_ids,class_labels):
    x, y, w, h = box
    if class_label != 2:
        print("LABEL is:", class_label)
        continue
    #print("LABEL CAR:"+ 'car')

    centroid = Point(float(x + w / 2), float(y + h / 2))
    track = track_history[track_id]
    track.append((float(x + w / 2), float(y + h / 2)))
    if len(track) > 30:
        track.pop(0)

    if centroid.within(polygon):
        if track_id not in vehicle_speeds:
            vehicle_speeds[track_id] = {'entry_time': time.time(), 'speed': None}
            vehicle_counter += 1
            #print(f"Vehicle {track_id} entered the polygon.")
        elif not vehicle_speeds[track_id]['speed']  == None:
            vehicle_speeds[track_id]['speed'] = None

keysdel=[]
```

```
    for eid in vehicle_speeds:

        if not(eid in track_ids):
            entry_time = vehicle_speeds[eid]['entry_time']
            time_diff = time.time() - entry_time
            if vehicle_speeds[eid]['speed']==None:
                speed = distance/1000 / (time_diff / 3600)
                vehicle_speeds[eid]['speed'] = speed
                vehicle_speeds[eid]['deleteTime']=time.time()
                print(f"Vehicle {track_id} exited the polygon, Speed: {speed:.2f} km/h")

            elif time.time()-vehicle_speeds[eid]['deleteTime']>30:
                keysdel.append(eid)
                if vehicle_speeds[eid]['speed'] <110:
                    avgSpeed=0.9*avgSpeed+vehicle_speeds[eid]['speed'] *0.1

    print('KEYSDEL',keysdel)

    for kd in keysdel:
        del vehicle_speeds[kd]

    # Draw the polygon and lines
    if line1 and line2:
        cv2.line(annotated_frame, line1[0], line1[1], (255, 0, 0), 2)
        cv2.line(annotated_frame, line2[0], line2[1], (255, 0, 0), 2)
        cv2.polylines(annotated_frame, [np.array([line1[0], line1[1], line2[1], line2[0]])],
isClosed=True, color=(0, 255, 255), thickness=2)

    cv2.putText(annotated_frame, 'Vehiculos desde ultimo min: ' + str(vehicle_counter) +"
Velocidad Promedio: " +str(round(avgSpeed, 2)) , (50, 50) , cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 255, 0), 2)
    cv2.imshow('Draw Lines', annotated_frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Anexo 4. Servidor web - Plataforma (Django – Python)

Modelo

```
from django.db import models
# Create your models here.
class Monitor(models.Model):
    ubicacion_lat = models.FloatField()
    ubicacion_lon = models.FloatField()
    direccion = models.CharField(max_length=255)
    idmonitor= models.CharField(max_length=255, unique=True)
    def __str__(self):
        return f"Monitor at {self.direccion}"

class Dato(models.Model):
    hora = models.TimeField()
    fecha = models.DateField()
    monitor = models.ForeignKey(Monitor, on_delete=models.CASCADE)
    velocidad = models.IntegerField()
    temperatura = models.FloatField()
    presion = models.FloatField()
    humedad_relativa = models.FloatField()
    trafico = models.IntegerField()

    def __str__(self):
        return f"Dato for Monitor {self.monitor} on {self.fecha} at {self.hora}"
```

Vistas

```
from django.shortcuts import render
# Create your views here.
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from .models import Monitor, Dato
import json
from django.http import JsonResponse
from datetime import datetime
```

```
@csrf_exempt
def add_dato(request):
    if request.method == 'POST':
        try:
            data = json.loads(request.body)
            monitor_id = data['idmonitor']
            monitor = Monitor.objects.get(idmonitor=monitor_id)
            dato = Dato.objects.create(
                hora=data['hora'],
                fecha=data['fecha'],
                monitor=monitor,
                velocidad=data['velocidad'],
                temperatura=data['temperatura'],
                presion=data['presion'],
                humedad_relativa=data['humedad_relativa'],
                trafico=data['trafico']
            )
            return JsonResponse({'message': 'Dato added successfully!'}, status=201)
        except Monitor.DoesNotExist:
            return JsonResponse({'error': 'Monitor not found'}, status=404)
        except KeyError as e:
            return JsonResponse({'error': f'Missing field: {str(e)}'}, status=400)
        except Exception as e:
            return JsonResponse({'error': str(e)}, status=500)
    return JsonResponse({'error': 'Invalid request method'}, status=405)

def frontpage(request):
    return render(request, 'frontpage.html')

def get_monitors(request):
    monitors = Monitor.objects.all()
    monitor_data = []
    for monitor in monitors:
        last_dato = Dato.objects.filter(monitor=monitor).order_by('-fecha', '-hora').first()
        monitor_data.append({
```

```
            'idmonitor': monitor.idmonitor,
            'ubicacion_x': monitor.ubicacion_lat,
            'ubicacion_y': monitor.ubicacion_lon,
            'direccion': monitor.direccion,
            'last_update': f"{last_dato.fecha} {last_dato.hora}" if last_dato else "No data"
        })
    return JsonResponse(monitor_data, safe=False)

def get_data(request, idmonitor):
    start_date = request.GET.get('start')
    end_date = request.GET.get('end')
    if start_date and end_date:
        start_date = datetime.strptime(start_date, '%Y-%m-%d')
        end_date = datetime.strptime(end_date, '%Y-%m-%d')
        datos = Dato.objects.filter(monitor__idmonitor=idmonitor, fecha__range=[start_date,
end_date]).order_by('fecha', 'hora')
    else:
        datos = Dato.objects.filter(monitor__idmonitor=idmonitor).order_by('fecha', 'hora')
    data_list = [{
        'fecha': dato.fecha,
        'hora': dato.hora,
        'velocidad': dato.velocidad,
        'temperatura': dato.temperatura,
        'presion': dato.presion,
        'humedad_relativa': dato.humedad_relativa,
        'trafico': dato.trafico,
    } for dato in datos]
    return JsonResponse(data_list, safe=False)
import csv
from django.http import HttpResponse

def download_data(request, idmonitor):
    monitor = Monitor.objects.get(idmonitor=idmonitor)
    datos = Dato.objects.filter(monitor=monitor).order_by('-fecha', '-hora')
```

```
    # Create a CSV response
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = f'attachment; filename="{monitor.idmonitor}_data.csv"'

    # Write CSV data
    writer = csv.writer(response)
    writer.writerow(['Fecha', 'Hora', 'Velocidad', 'Temperatura', 'Presion', 'Humedad Relativa',
'Trafico'])
    for dato in datos:
        writer.writerow([dato.fecha, dato.hora, dato.velocidad, dato.temperatura, dato.presion,
dato.humedad_relativa, dato.trafico])

    return response
```

URL

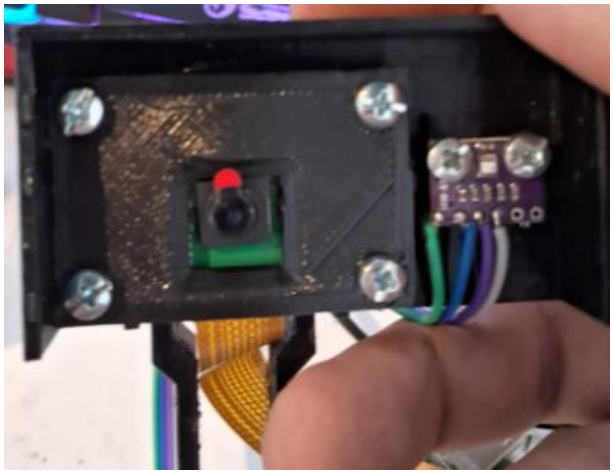
```
# monitor_trafico/plataforma/urls.py
from django.urls import path
from .views import add_dato
from .views import frontpage, get_monitors, download_data,get_data

urlpatterns = [
    path('add_dato/', add_dato, name='add_dato'),
    path('get_monitors/', get_monitors, name='get_monitors'),
    path('download_data/<str:idmonitor>/', download_data, name='download_data'),
    path('get_data/<str:idmonitor>/', get_data, name='get_data'),
]
```

Administrador

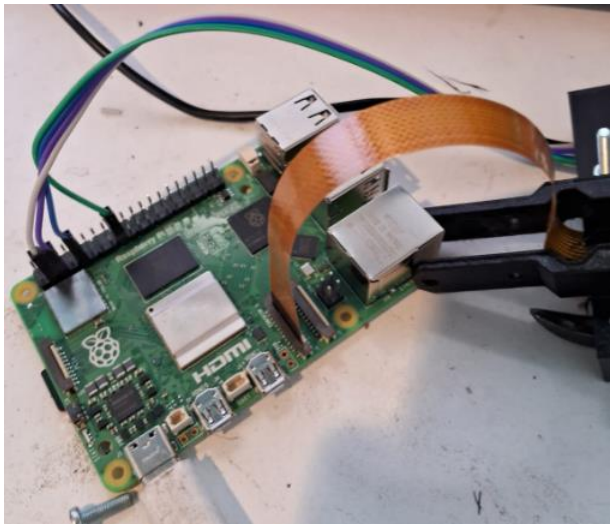
```
from django.contrib import admin
# Register your models here.
from django.contrib import admin
from .models import Monitor, Dato

admin.site.register(Monitor)
admin.site.register(Dato)
```

Anexo 5. Respaldos de los elementos electrónicos para montaje

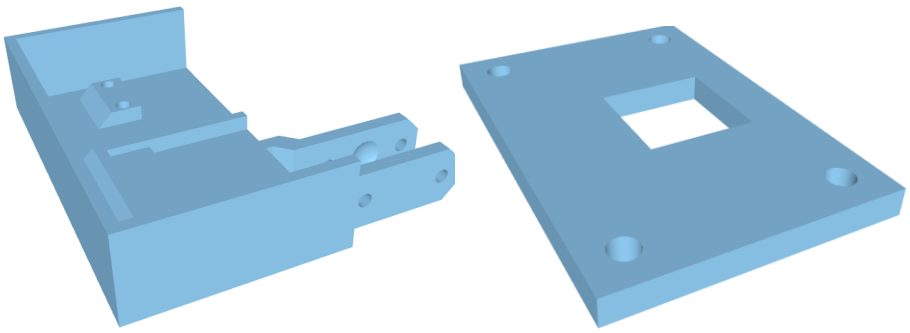
- Cámara y sensor BMP280 conectados

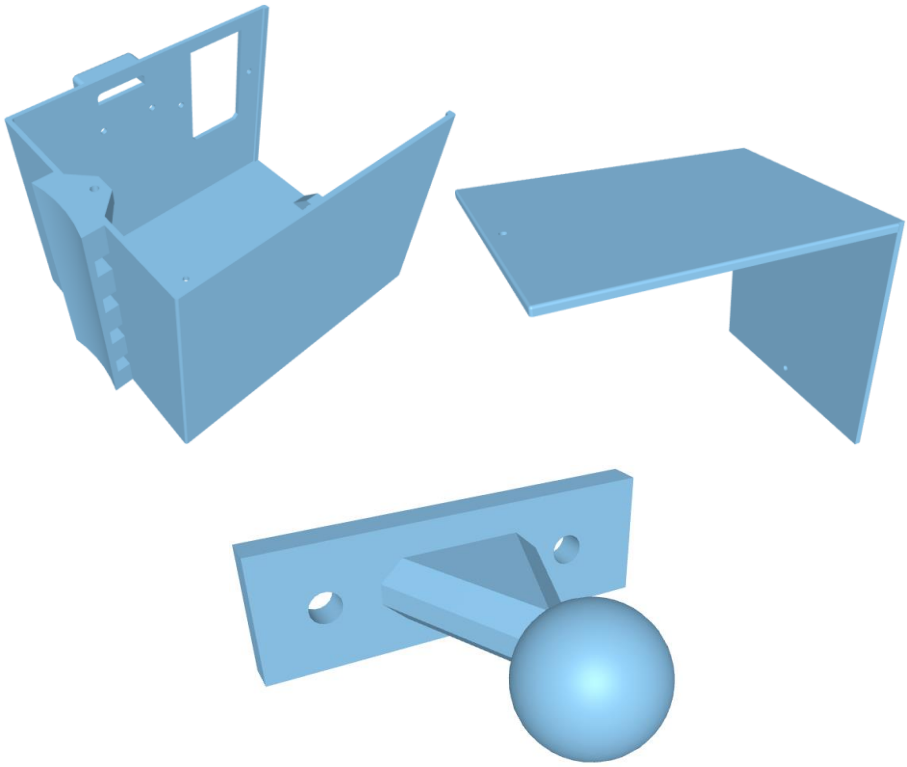


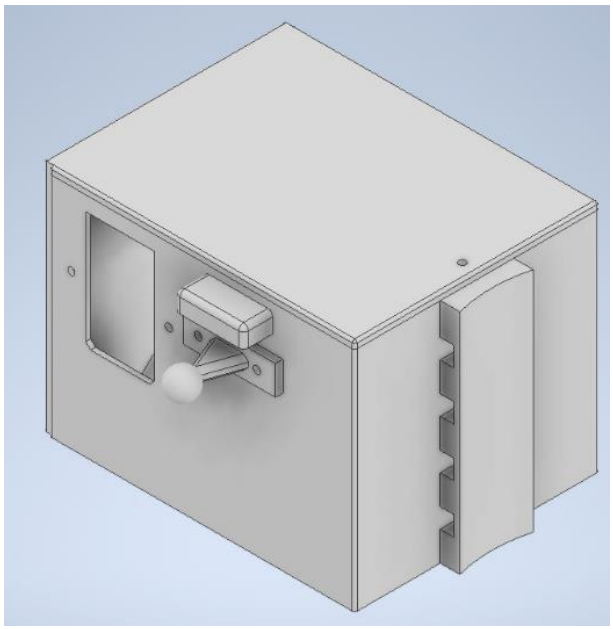
- Raspberry Pi 5



- Partes del modelo de diseño de la caja negra







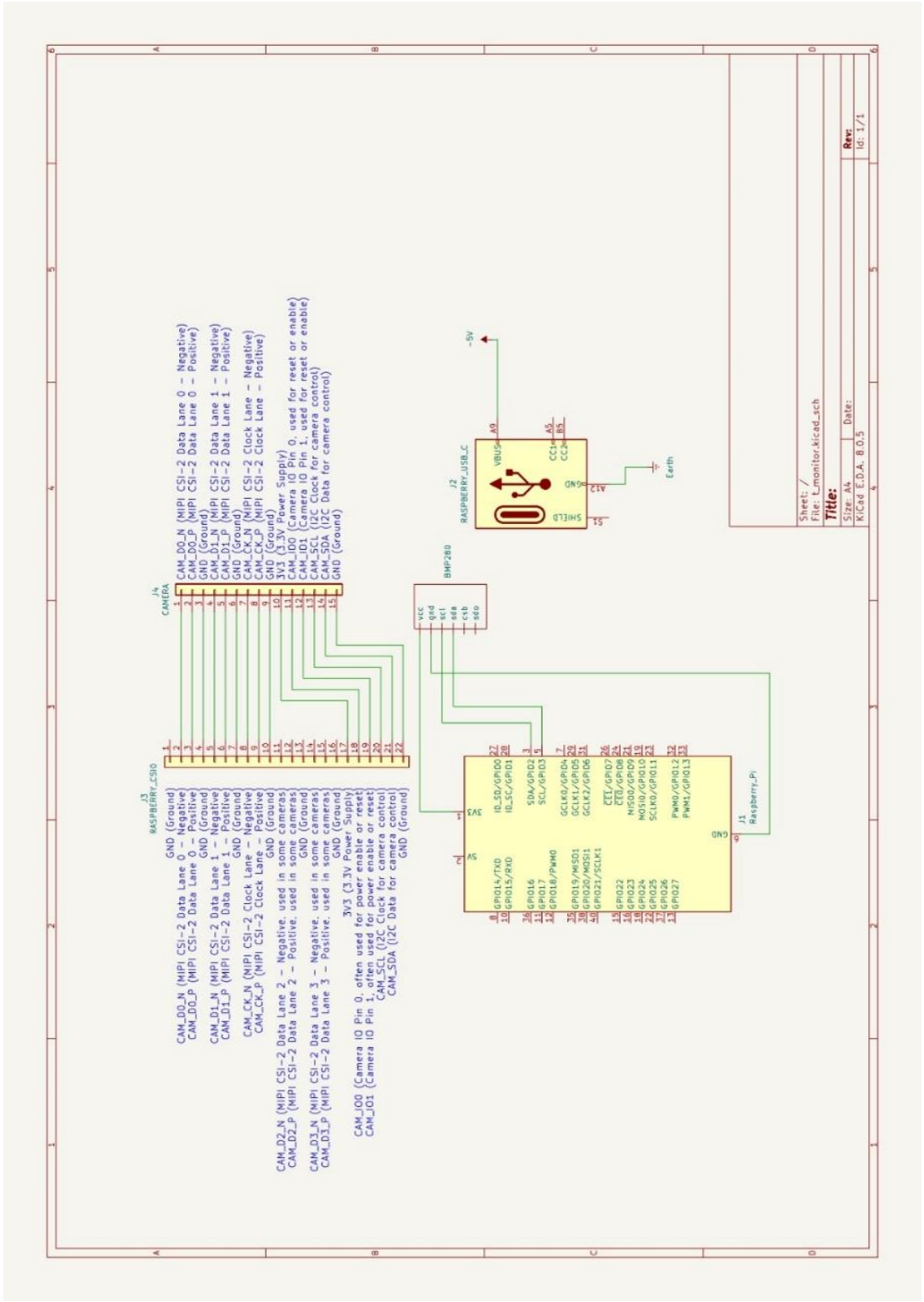
- Conexión conjunta del dispositivo electrónico

Anexo 6. Datos totales de los parámetros

Se coloca una muestra del total (27.672) de datos.

Fecha	Hora	Velocidad	Temperatu	Presion	Trafico
28/8/2024	0:00:29	17	14,45	760,59	0
28/8/2024	0:01:30	17	14,42	760,6	0
28/8/2024	0:02:30	17	14,44	760,61	0
28/8/2024	0:03:31	17	14,37	760,6	0
28/8/2024	0:04:31	17	14,32	760,56	0
28/8/2024	0:05:32	17	14,42	760,59	0
28/8/2024	0:06:33	17	14,46	760,63	0
28/8/2024	0:07:33	17	14,53	760,64	0
28/8/2024	0:08:34	17	14,57	760,63	0
28/8/2024	0:09:34	17	14,61	760,62	0
28/8/2024	0:10:35	17	14,64	760,66	0
28/8/2024	0:11:36	17	14,64	760,63	0
28/8/2024	0:12:36	17	14,54	760,65	0
28/8/2024	0:13:37	17	14,55	760,66	0
28/8/2024	0:14:37	17	14,64	760,71	0
28/8/2024	0:15:38	17	14,75	760,63	0
28/8/2024	0:16:38	17	14,83	760,67	0
28/8/2024	0:17:39	17	14,73	760,63	0
28/8/2024	0:18:39	17	14,6	760,62	0
28/8/2024	0:19:40	17	14,53	760,59	0
28/8/2024	0:20:41	17	14,42	760,59	0
28/8/2024	0:21:41	17	14,33	760,61	0
28/8/2024	0:22:42	17	14,34	760,54	0
28/8/2024	0:23:42	17	14,39	760,53	0
28/8/2024	0:24:43	17	14,38	760,49	0
28/8/2024	0:25:44	17	14,47	760,54	0
28/8/2024	0:26:44	17	14,54	760,49	0
28/8/2024	0:27:45	17	14,48	760,45	0
28/8/2024	0:28:46	17	14,42	760,47	0
28/8/2024	0:29:46	17	14,33	760,44	0
28/8/2024	0:30:47	17	14,34	760,48	0
28/8/2024	0:31:47	17	14,46	760,44	0
28/8/2024	0:32:48	17	14,63	760,48	0

Anexo 7. Circuito eléctrico del dispositivo



MANUAL DE USUARIO

Manual del usuario

Configuración e instalación del sistema de monitoreo de tráfico

1

Contenido

Para realizar la configuración del sistema de monitoreo de tráfico se deben seguir el siguiente procedimiento:

Configuración de energía y hardware

1. Hay que asegurar de que la Raspberry Pi esté conectada a una fuente de alimentación con una salida de 5 VDC y corriente suficiente (al menos 3 A, según el modelo). Además, utilizar un cable de alimentación USB-C y encender la microcomputadora con el interruptor.

2. Verificar que la tarjeta microSD esté ya instalada con el sistema operativo Raspberry Pi.
3. Verificar que las conexiones coincidan con el cableado especificado en el diagrama de cableado del dispositivo (ver Anexo 8).

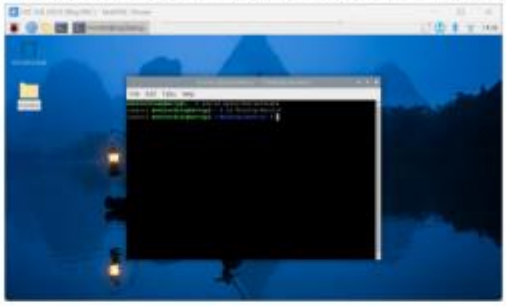
Configuración de la red y el acceso remoto

1. Conectar la Raspberry Pi a la red local:
   - Utilizar una conexión Wi-Fi.
   - Verificar que a la Raspberry Pi se le asigne una dirección IP en la red local (puede usar la opción ifconfig).
   - Instalar VNC Viewer para acceso remoto:
2. Instalar el VNC Viewer en su PC o dispositivo móvil.
   - Asegurar de que VNC esté habilitado en Raspberry Pi (sudo raspi-config-> Opciones de interfaz -> Habilitar VNC).
   - Abrir VNC Viewer en su PC/dispositivo móvil y conectarse a la Raspberry Pi con la dirección IP.
   - Introducir las siguientes credenciales:
     - Nombre de usuario: monitor.

- Contraseña: monitor01.

Configuración inicial en Raspberry Pi

1. Acceder a la línea de comandos de Raspberry Pi a través del visor VNC.
2. Activar entorno virtual:
   - Navegar hasta la carpeta que contiene sus scripts: cd Desktop/monitor



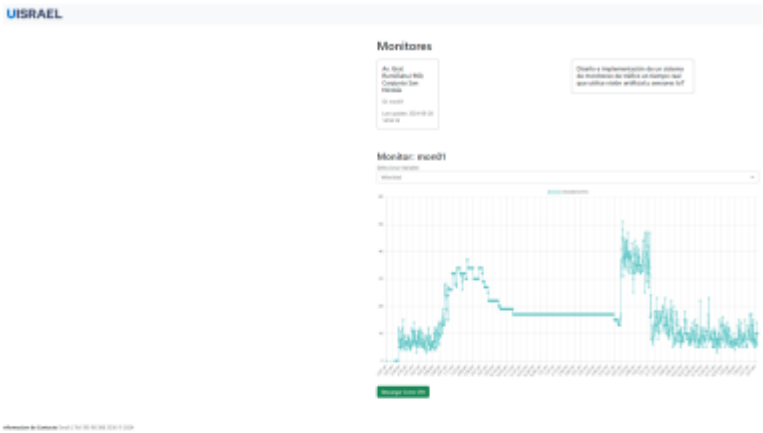
   - Activar el entorno Python: source myenv/bin/activate.

Ejecución de los scripts

1. Ejecutar script $ python traffic_track_YOLO
2. En caso de que se desee ejecutar el script de configuración inicial $ python startConfig.

Seguimiento y recopilación de datos

1. Monitoreo de datos:
   - El sistema rastreará los vehículos en tiempo real y medirá parámetros como la velocidad, la presión y la temperatura.
   - Los datos se cargarán a la API web para acceso remoto.
2. Acceso a los datos a través de la interfaz web:
   - Los datos recopilados por el sistema de monitoreo de tráfico se pueden ver a través de una interfaz web desarrollada por Django.
   - Inicie sesión en el servidor (DigitalOcean, Ubuntu) para comprobar los datos de monitoreo:
     - Dirección IP: ###.###.###.##
     - Nombre de usuario: monitor
     - Contraseña: monitor01
3. Finalmente, se pueden ver datos como la velocidad, la presión y la temperatura del vehículo directamente en la plataforma web.



5