UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA DE SISTEMAS

Desarrollo de un prototipo de semáforo inteligente con visión por computador.

Trabajo de titulación previo a la obtención del
Título de Ingeniero de Sistemas

AUTOR: Marcelo Omar Castro Quishpe

TUTOR: Holger Raúl Ortega Martínez

Quito - Ecuador

2022

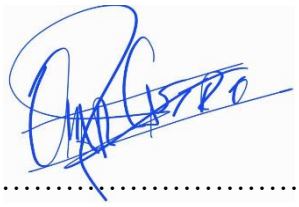
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Marcelo Omar Castro Quishpe con documento de identificación N.º 1720821766 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 14 de septiembre del 2022

Atentamente,



……..

Marcelo Omar Castro Quishpe

1720821766

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Marcelo Omar Castro Quishpe con documento de identificación N.º 1720821766, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "Desarrollo De Un Prototipo De Semáforo Inteligente Con Visión Por Computador", el cual ha sido desarrollado para optar por el título de: Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 14 de septiembre del 2022

Atentamente,

……..

Marcelo Omar Castro Quishpe

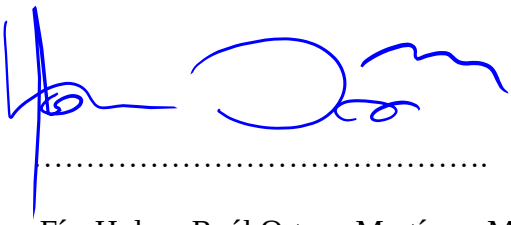
1720821766

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACÍON

Yo, Ortega Martínez Holger Raúl con documento de identificación N.º 1716975501, docente dela Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado trabajo de titulación: DESARROLLO DE UN PROTOTIPO DE SEMÁFORO INTELIGENTE CON VISIÓN POR COMPUTADOR realizado por Marcelo Omar Castro Quishpe con documento de identificación N.º 1720821766, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinadospor la Universidad Politécnica Salesiana.

Quito, 14 de septiembre del 2022

Atentamente,



…….

Fís. Holger Raúl Ortega Martínez, MSc

1716975501

DEDICATORIA

Dedico el presente trabajo principalmente a Dios por ser mi eje fundamental, y haberme guiado y brindarme la sabiduría en cada paso tanto de mi carrera como de mi vida cotidiana.

Con gran amor y enorme gratitud, a mi madre Marcela Quishpe por brindarme su apoyo incondicional y estar en todo momento de mi vida y por su motivación para lograr convertirme en un gran profesional.

A mi prometida Nataly Torres y a mi hijo Leonardo Castro, que han sido una gran motivación para seguir creciendo y culminar esta etapa tan importante en mi vida.

Al Sr. Patricio Arcos y Sra. Zoila Cañar por su apoyo incondicional por verme triunfar y por quienes he podido seguir la carrera, a quienes agradezco su motivación, su guía, por el cariño y aprecio hacia mí familia por haberme apreciado como un hijo.

A todos quienes han sido parte de este propósito y sueño y que han servido de inspiración, motivación y ayuda a lo largo de la carrera, a amigos y familiares que han aportado para que pueda culminar mi carrera.

Marcelo Omar Castro Quishpe

AGRADECIMIENTO

Agradezco a Dios, por permitirme gozar de buena salud, por darme la sabiduría y ayudarme en cada etapa de mi vida.

Agradezco a mi madre por cada día inculcarme buenos valores y ayudarme a ser un profesional con la correcta formación desde el hogar.

A mis compañeros y amigos por cada momento y experiencias compartidas a lo largo de la carrera.

A la Universidad Politécnica Salesiana por permitirme ser parte de esta gran familia y culminar mi carrera profesional.

Finalmente agradecer a mi tutor Holger Ortega, por transmitir su conocimiento, por su ayuda, tiempo y dedicación para que la realización del proyecto sea posible.

Marcelo Omar Castro Quishpe

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURA

RESUMEN

El proyecto técnico se desarrolló mediante el enfoque de realizar el prototipo de un semáforo inteligente con visión por computador; con un módulo de detección de autos, mediante el cual se cuenta, en tiempo real, los autos de dos vías de una intersección, para dar prioridad a la vía con mayor número de vehículos.

Con esta información, con la ayuda de un módulo controlador de semáforo, se determina el tiempo de cambio de luz dinámicamente, priorizando la circulación, a la vía con mayor cantidad de vehículos, disminuyendo la congestión vehicular y demora en la movilidad en dicha intersección.

Para programar los módulos de conteo de vehículos y de control del semáforo se utilizó el lenguaje Python, fundamental para el reconocimiento de objetos, y para conseguir desarrollar el módulo de detección de vehículos; una de sus librerías llamada OpenCV, que por medio de la captura y procesamiento de las imágenes obtenidas por la cámara se consigue detectar al vehículo entre los demás objetos para su conteo y con dicha información mediante el módulo controlador del semáforo distribuir el tiempo para el cambio de luz, los datos obtenidos se almacenan en una base de datos, para ello se usó PostgreSQL.

Palabras Clave: Semáforo inteligente, visión por computador, Raspberry, Python, OpenCV, Procesamiento de Imágenes, Inteligencia Artificial.

ABSTRACT

The technical project was developed through the approach of making the prototype of an intelligent traffic light with computer vision; with a car detection module, through which two-way cars at an intersection are counted in real time, to give priority to the road with the largest number of vehicles.

With this information, with the help of a traffic light controller module, the time of change of light is determined dynamically, giving priority to circulate, to the road with the largest number of vehicles, reducing traffic congestion and delay in mobility at that intersection.

To program the vehicle counting and traffic light control modules, the Python language was used, essential for object recognition, and to develop the vehicle detection module; one of its libraries called OpenCV. Capturing and processing the images obtained by the camera is able to detect the vehicle among the other objects for counting and with said information through the traffic light controller module distribute the time for the change of light, the data obtained is stored in a postgres database.

To carry out the project, in addition to the research, computer vision has been applied mainly, which is a branch of study of Artificial Intelligence, and which, added to the programming implemented in the modules, makes the developed prototype differ from a conventional traffic light.

Keywords: Intelligent traffic light, computer vision, Raspberry, Python, OpenCV, Image Processing, Artificial Intelligence.

INTRODUCCIÓN

El procesamiento de imágenes, el reconocimiento de patrones, la visión por computadora y la Inteligencia Artificial forman parte esencial de una gran variedad de programas que tienen distintas áreas de aplicación como son la medicina, el sector industrial, la ciencia, la seguridad entre otros.

El proyecto técnico que se presenta a continuación lleva por título: “Desarrollo de un prototipo de semáforo inteligente con visión por computador.” Está constituido por un módulo de visión por computador y un módulo de control del semáforo. En el desarrollo, el lenguaje de programación que se ha utilizado es Python, debido a su eficiencia con el estudio de datos, reconocimiento de objetos e Inteligencia Artificial.

ANTECEDENTES

La movilidad vehicular en una intersección se encuentra condicionada por los semáforos convencionales con un tiempo definido para cada cambio de fase que es independiente de la cantidad existente de tráfico, lo cual es una desventaja en relación un semáforo inteligente que cuenta con la capacidad de configurar los cambios de fase de manera dinámica en base al análisis en tiempo real y conteo de los autos que se encuentran circulando en la vía para disminuir el tiempo promedio que tardan los vehículos en cruzar una intersección y para optimizar el tráfico.

El municipio de Quito mediante de la Empresa Pública Metropolitana de Movilidad y Obras Públicas (EPMMOP) ha indicado que existen 1021 intersecciones semaforizadas que administra actualmente, de las cuales 311 se encuentran aisladas y no se encuentran centralizadas mientras que 710 se operan desde el Centro de Gestión de Movilidad (Secretaría de Movilidad, 2022).

Al no contar con todas las intersecciones controladas por un semáforo inteligente se dificulta el control de la movilidad debido a la falta de análisis en tiempo real para que el tiempo de cambio de fase se adapte más a la realidad y sea controlado en base a ese análisis.

PROBLEMA

El tráfico vehicular es un inconveniente diario en varias ciudades tal como lo mencionan Pérez, Bautista, Salazar y Macias (2014) en la página 36 de su publicación: Análisis del flujo de tráfico vehicular a través de un modelo macroscópico, “La mayoría de las ciudades se enfrentan hoy en día a varios inconvenientes provocados por el tráfico vehicular a causa de la creciente cantidad de vehículos en circulación, como: contaminación del medio ambiente, congestión, exceso de ruido, incremento del número de accidentes viales, etc”.

De acuerdo con la información recopilada por el Instituto Nacional Ecuatoriano de Estadística y Censos (INEC), Quito es la ciudad más poblada del Ecuador. En el 2018, tenía aproximadamente 2.690.150 millones de habitantes y de acuerdo con la proyección del 2020 se incrementó esta cantidad a los 2.781.461 millones de habitantes, es decir que, ocupa el 16 % de la población nacional (CAMICON, 2020).

El transporte urbano desempeña un papel significativo en la movilidad. (Secretaría de Movilidad, 2021). El tráfico vehicular es con lo que muchos conductores deben lidiar a diario, al no tener un control eficiente para la optimización de la movilidad se toma demasiado tiempo en movilizarse dentro de la ciudad (Secretaría de Movilidad, 2021).

Se conoce que, la movilidad ha crecido, según los estudios técnicos que realizó la Asociación de Empresas Automotrices del Ecuador (AEADE), dicho incremento ha sido de 10% entre 2019 y 2020, y se ha mantenido en el primer semestre de 2021 (Mendoza, 2021).

En muchas ocasiones, cuando los conductores llegan a las intersecciones se encuentran con el semáforo en luz roja, teniendo que esperar al cambio de luz, pese a que en la intersección no se encuentra ningún vehículo. Esto ocasiona que aumente el tráfico vehicular y que disminuya

la capacidad de movilidad más fluida, en especial en las vías principales (Secretaría de Movilidad, 2021).

JUSTIFICACIÓN

El desarrollo del tema propuesto es importante ya que representaría una optimización de recursos, como serían: disminuir los agentes de tránsito para controlar el tráfico, disminuir la contaminación auditiva y las emisiones de CO_2, y solucionar el problema de la delincuencia que se da en los semáforos con frecuencia.

Un semáforo inteligente tiene control sobre la movilidad por medio de la visión por computador y gestiona el tiempo, de cada fase del semáforo, dinámicamente con la información obtenida, y con la ayuda de algoritmos procesa esa información con la finalidad de reducir la congestión que se da, sobre todo en las calles principales, reduciendo las aglomeraciones en las intersecciones. La implementación de este sería un aspecto positivo para el turismo ya que tendría mayor facilidad de movilidad, menos tráfico y una mejor imagen de la ciudad.

Llevar a cabo la implementación del sistema tendría un costo moderado porque se puede implementar en dispositivos Raspberry Pi que es de precio accesible y además las herramientas de software utilizadas en el desarrollo son Open Source.

El área de trabajo y estudio está relacionada con la visión por computador que viene a ser parte de Inteligencia Artificial y Machine Learning que son ramas de la Ingeniería de Sistemas, por lo cual tenemos las herramientas conceptuales para abordar el desarrollo y la investigación.

OBJETIVOS

General

Desarrollar un prototipo de semáforo y su módulo controlador que utilice visión por computador en una intersección de avenidas para optimizar el flujo de circulación de los vehículos.

Específicos

Definir la arquitectura del prototipo

Desarrollar el módulo de visión por computador que realice la detección de vehículos en el video capturado.

Armar el prototipo del semáforo y del módulo controlador.

Realizar las pruebas del prototipo.

METODOLOGÍA

Se ha desarrollado el prototipo enfocado en cada aspecto de su funcionamiento y la funcionalidad brindada, se optó por manejar herramientas Open Source, un minicomputador Raspberry por su capacidad de proceso, poco espacio que utiliza y su bajo coste, se ha utilizado también 2 cámaras web y dos circuitos de semáforos que son de bajo coste y el espacio requerido para su implementación es mínimo. El lenguaje utilizado es Python para los dos módulos que lo componen.

Para poder adaptar a la realidad se ha realizado el desarrollo basándose en un video de tráfico real en la ciudad, durante el día y posicionando la cámara sobre un puente peatonal.

El semáforo ha sido configurado y puesto a prueba en fases de tiempo predeterminadas y fases de tiempo dinámicas.

En base a una circulación rápida por parte de los motociclistas que no respetan el carril, se ha tenido que adaptar la captación del video a la velocidad de circulación para poder detectar de manera más eficiente.

Para la implementación del módulo controlador del semáforo se investigó acerca de la semaforización y las normas INEN existentes en ecuador, y fueron de gran ayuda para conocer nociones básicas acerca de los semáforos.

Se planteo un cronograma para conocer que tiempo tomaría realizar las actividades necesarias en cada uno de los módulos.

Tabla 1 Cronograma de actividades

N°	**ACTIVIDAD**	**Duración**	**Observación**
1	Análisis y definición de la arquitectura del prototipo.	40 horas	

2	Construcción de la maqueta del semáforo.	24 horas	
3	Investigación bibliográfica de algoritmos de detección de vehículos.	39 horas	
4	Implementación y corrección de errores en el algoritmo de detección.	140 horas	Actividad pesada
5	Implementación del módulo de visión por computador.	23 horas	
6	Implementación del módulo controlador.	50 horas	Actividad pesada
7	Pruebas	20 horas	
8	Depuración de los errores que pueda tener el prototipo.	13 horas	
9	Redacción del primer borrador	45	Actividad pesada
10	Corrección del primer borrador	14 horas	

Nota. Cronograma de actividades. Elaborado por: Marcelo Castro

CAPÍTULO I
MARCO TEÓRICO

1.1 EVOLUCIÓN DE LOS CONTROLES DE TRÁFICO VEHICULAR

A partir de la creación de un sistema vial se han venido desarrollando diversos métodos de control de tráfico que permitan la disminución de accidentes de tránsito y atascamiento. Desde que en el año 1914 en la ciudad de Berlín se implementó el primer sistema de semaforización con funcionamiento a gas, hasta llegar al sistema de semaforización actual, basado en programas y sistemas informáticos. El control de tráfico vehicular está centrado en detectar y organizar la circulación de los vehículos que transitan con el objetivo de prevenir congestión vehicular o accidentes, hay múltiples aplicaciones para dar una solución al problema de la movilidad, estos sistemas utilizan técnicas de visión artificial; con el avance tecnológico se han venido presentando nuevos paradigmas con respecto al control de flujo vehicular y siendo preponderante el desarrollo de programación que detecte una mejor capacidad en el desarrollo de la movilidad humana (Garcia, Sarrazola, 2018).

En el año de 1989 el California Institute of Technology en conjunto con el Jet Propulsion Laboratory de la Nasa desarrollaron un sistema que detecta vehículos en tiempo real que además de detectarlos también podía contarlos y medir su velocidad con la ayuda del método de diferenciación de marcos (Iñigo, 1989).

La investigación desarrollada por Hoose (1991) señala que, entre 1989 y 1991, se desarrolló un sistema con la capacidad de seguir los vehículos y realizar la detección de las cualidades en la imagen; el sistema comienza dividiendo en celdas la imagen captada, luego dividiendo en grupos las celdas obtenidas: vacía, con vehículo en movimiento y con vehículo estático. Al final los agrupa en conjuntos. Ubicando la forma de sus bordes en las celdas agrupadas los vehículos fijos son encontrados y comparando las celdas en tiempos diferentes los vehículos

en movimiento son encontrados.

Según el desarrollo realizado por Betke, Haritaoglu, y Davis (1996), en el año de 1996 se desarrolló e implementó un sistema de rastreo y reconocimiento de vehículos partiendo de videos en escala de grises, tomados desde un vehículo diferente en movimiento, una de las complejidades de este desarrollo representa la cámara en movimiento a diferencia de la que se encuentra ubicada en un punto estático, esto se debe al desplazamiento ejecutado, al paisaje en constante cambio que debe evaluar y a la variable iluminación. La detección vehicular se realizó estudiando los bordes horizontales y verticales en las imágenes obtenidas, buscando objetos rectangulares y buscando las imágenes que cumplen una relación entre sus medidas.

En Gupte, Masoud, Martin y Papanikolopoulos, (2002), existen algoritmos para el desarrollo y avance de la visión por computador, basados en la detección, visión y clasificación vehicular en series de imágenes monoculares de cuadros de tráfico captadas por una cámara fija. La transformación es realizada en tres grados: imágenes sin transformar, nivel de área y nivel de vehículo.

El desarrollo realizado por Moreno, Sánchez, Valcárcel y Chávez, (2012) detallan de la utilización de lógica difusa aplicada en un controlador de tráfico con la capacidad de coordinar y diferenciar centralizada y autónomamente el flujo vehicular en varias intersecciones con el objetivo de dar prioridad a vehículos de emergencia en la vía, mediante el análisis de secuencias de video y de técnicas de procesamiento de imágenes.

Los autores (Youyang, Liang, Xinping, 2020), en su investigación, presentan la detección de vehículos en movimiento desde vehículos aéreos no tripulados y proponen un detector de vehículos en movimiento preciso.

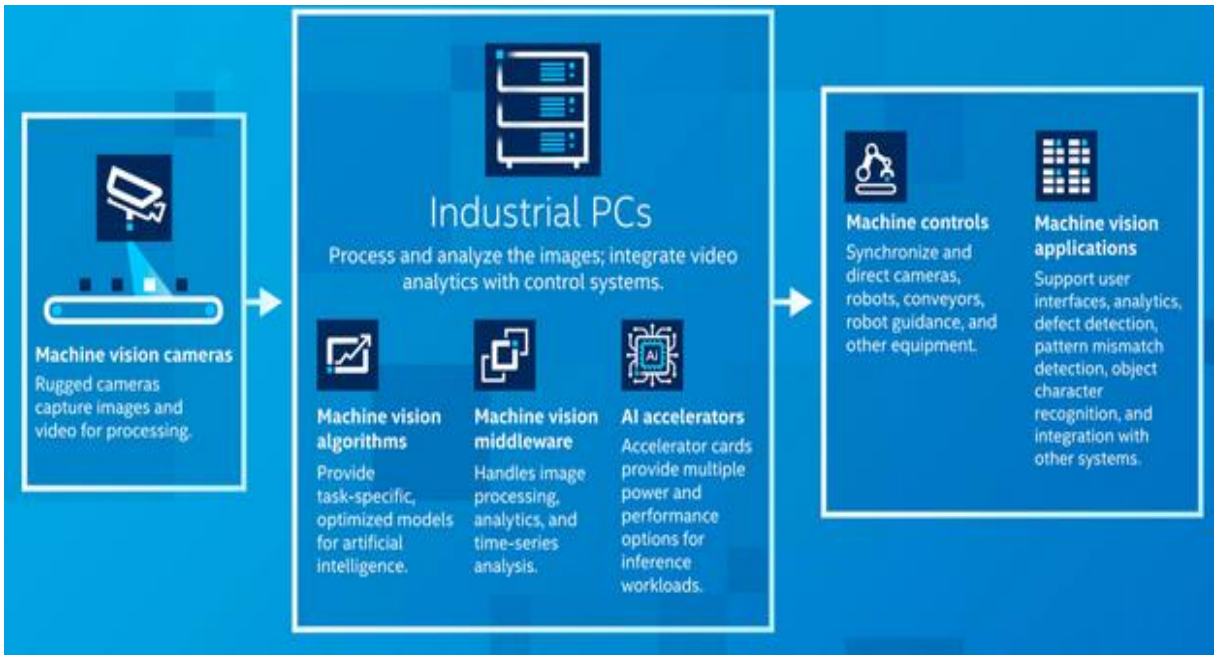
1.2 VISIÓN ARTIFICIAL

1.2.1 Definición de visión artificial

Se define como un campo de la inteligencia artificial que hace posible que el computador y los sistemas recopilen importante información de videos, imágenes y otras entradas visuales, y basado en esa información hagan ciertas recomendaciones o se tomen acciones (IBM, 2022).

Es una de las fundamentales tecnologías en la era de la automatización industrial. Ha sido de ayuda en el aceleramiento de la producción, en el mejoramiento de los productos en base a la calidad, y la optimización en la fabricación. Actualmente se fusiona con la inteligencia artificial para una transición liderada hacia la industria 4.0 (Intel, 2022).

Figura 1 *Arquitectura de sistemas de visión artificial*



Nota. Diagrama de los sistemas de visión artificial y los componentes trabajando en conjunto para la transformación de las operaciones. Fuente (Intel, 2022)

El funcionamiento de la visión artificial y la visión humana tienen bastante similitud, salvo la

ventaja que tienen los humanos. La vista de los humanos posee la ventaja del aprendizaje de los contextos y de las experiencias para diferenciar entre los objetos, la distancia, el movimiento o las diferencias en una imagen.

Para ejecutar estas funciones la visión artificial adiestra a las máquinas, pero tiene que realizarlo en un tiempo considerablemente menor y ayudado de cámaras, datos y algoritmos y no con la ayuda de nervios ópticos retinas y una corteza visual (IBM, 2022).

1.2.2 Historia de visión artificial

Tanto científicos como ingenieros durante aproximadamente 60 años han intentado evolucionar la manera en que las máquinas ven y comprenden visualmente los datos. En 1959 la experimentación comenzó, cuando le indicaron a un gato varias imágenes un grupo de neurofisiólogos, tratando de asociar una respuesta en su cerebro. Encontraron que respondía inicialmente a líneas sólidas o bordes, y esto, científicamente, representaba que el procesamiento de imágenes comienza con formas sencillas.

Simultáneamente, la primera tecnología para el escaneo artificial de imágenes se desarrolló, la cual permite al computador obtener imágenes y digitalizar. A partir de la década de 1960, la utilización de la Inteligencia Artificial marcó el inicio para resolver el problema de la visión humana y también apareció como un campo de estudio académico. En el año de 1963, consiguieron que las computadoras pudieran convertir imágenes de dos dimensiones en figuras tridimensionales.

Ya para 1974 se presentaba la tecnología que en cualquier fuente o tipo de letra permitía reconocer el texto impreso, llamada reconocimiento óptico de caracteres (OCR). De igual manera, utilizando redes neuronales, el reconocimiento inteligente de caracteres (ICR) lograría decodificar el texto escrito a mano. A partir de aquel momento, OCR e ICR se han extendido en funcionalidad en la identificación de placas de los vehículos, la traducción de

idiomas automáticamente, el procesamiento de documentos y más aplicaciones.

En el año de 1982, David Marr, un neurocientífico, determinó que la visión artificial operaba de manera jerárquica, e instauró algoritmos con el propósito de que las máquinas detecten bordes, esquinas y formas básicas parecidas. Simultáneamente, una red de células con la capacidad para el reconocimiento de patrones fue desarrollada por el científico informático Kunihiko Fukushima. Con el nombre de Neocognitron se conoció a la red, esta contenía capas convolucionales en una red neuronal.

En el 2000, el centro de estudio era la identificación de objetos, y en el 2001 las primeras aplicaciones para reconocer rostros en tiempo real surgieron. La normalización para la anotación y etiquetado de las series de datos visuales nació durante la década del año 2000. Para el 2010, se encontró utilizable la serie de datos de ImageNet., lo que estableció una base en las CNN y para los modelos de Deep Learning que actualmente se utilizan, ya que contenía millones de imágenes con etiquetas en diferentes clases de objetos. Para el año 2012, un modelo de CNN, llamado AlexNet, fue presentado en un concurso de reconocimiento de imágenes por un grupo de la Universidad de Toronto. Dicho modelo, redujo la tasa de error de manera significativa en el reconocimiento de imágenes. Luego de este enorme avance, se han reducido los porcentajes de las tasas de error (IBM, 2022).

1.2.3 Finalidad de la visión artificial

La visión artificial posee como propósito principal el proveer a la máquina de “ojos” en la toma de decisiones para la automatización de cualquier proceso viendo lo que sucede en el mundo real (Contaval, 2016).

Se requiere muchos datos en la visión artificial y se realizan, por varias ocasiones, los análisis de datos necesarios hasta ubicar las diferencias y, por último, reconocer imágenes. La

fabricación de vehículos basados en visión artificial, con el objetivo de dar relevancia a las cámaras y otros sensores de un automóvil convirtiéndolo en autónomo son un gran ejemplo. Toda señal e información visual que se encuentre en la carretera, y se pueda reconocer como señales de tráfico, automóviles, peatones, son indispensables y es posible gracias a la visión por computador (IBM, 2022).

1.2.4 Algoritmos de detección y reconocimiento de patrones.

En el libro Patter Reconigtion los autores Theodoris y Koutroumbas (2008) señalan que, con el objetivo de categorizar objetos de varios tipos y clases, existe a manera de disciplina científica el reconocimiento de patrones. Varían según la utilización, pueden ser olores, sonidos, imágenes, datos que producto de cálculos deben ser clasificados.

Se han realizado infinidad de algoritmos para la detección de tendencias, estadística, imagen, posicionamiento u otras en combinación con el reconocimiento de patrones que fue altamente desarrollado en el lenguaje de programación.

El siglo XX es una época de gran desarrollo tecnológico en la que se ha evidenciado el uso de algoritmos para la optimización y aplicación de soluciones a nivel industrial y social, como, por ejemplo:

“Los usuarios mediante la cámara de un smartphone pueden apuntar hacia el texto para traducir inmediatamente en el idioma que se eligió gracias a Google Translate” (IBM, 2022).

“Los fabricantes de vehículos pueden identificar los autos defectuosos previo a su salida al mercado gracias a la visión artificial, tecnología que IBM en asociación con Verizon han aplicado” (IBM, 2022).

En el año 2018 en el torneo de golf Master se creó My Moments utilizando visión artificial por parte de IBM Watson que con la recopilación de cientos de horas de imágenes pudo

clasificar las tomas importantes de Masters. La recopilación de imágenes personalizadas con las mejores tomas se entregó a los fanáticos (IBM, 2022).

1.2.5 Etapas en la detección de patrones

La estructura del reconocimiento de patrones se compone de algunas fases asociadas entre sí, (los efectos de una fase pueden transformar las variables de fases previas) se debe tener en cuenta las siguientes fases:

- Recopilación de Datos
- Pre-Procesamiento
- Extracción de características
- Reconocimiento

1.2.6 Clasificación Supervisada

Se utilizan algoritmos de aprendizaje supervisado para la detección de patrones, en la clasificación supervisada, con los datos de entrada para igualar los datos de preparación etiquetados de forma manual con el producto deseado.

En visión artificial, los métodos supervisados para detección de patrones son utilizados para el reconocimiento óptico de caracteres (OCR), la detección de objetos y la clasificación de objetos.

1.2.7 Clasificación No Supervisada

Para clasificar sin supervisión los procedimientos pretenden localizar esquemas ocultos en datos no etiquetados con métodos de segmentación o clustering. Los procedimientos habituales de clasificación no supervisada son los siguientes:

- Modelos ocultos de Markov
- Modelos de mezcla Gaussiana
- Clustering de K-Means

1.2.8 Aplicación de la visión artificial

La visión artificial, gracias a su utilidad y eficacia, haría impensable la industria 4.0 al no tener una herramienta que sea robusta, aplicable y principalmente, constituida en todas las fases del proceso productivo (Revista Unir, 2022)

Con el objetivo de analizar la información que se consigue mediante cámaras y partiendo de la misma, crear un modelo del mundo real simulado en un computador empleando una rama de la inteligencia artificial la cual es la visión artificial. (Nagabhushana, 2005, pag. 2)

En distintas Áreas de la industria y las operaciones la aplicación de visión artificial es de gran utilidad; las aplicaciones en el mundo demuestran que la visión artificial es muy importante para las compañías, en el transporte, el entretenimiento, la medicina, y en el día a día. Un importante elemento para el crecimiento de dichas aplicaciones es la exorbitante información de categoría visual obtenida gracias a los smartphones, cámaras en el tránsito, sistemas para seguridad, entre otros. (IBM, 2022)

Las aplicaciones en el día a día evidencian lo fundamental que es la visión artificial para el transporte, las empresas, la atención médica, el entretenimiento y la vida cotidiana. La elevada cantidad de información visual es un elemento indispensable para que estas aplicaciones incrementen, dicha información proviene de cámaras de tráfico, sistemas de seguridad, teléfonos inteligentes y otros dispositivos visuales. En la industria esta información es valiosa para las operaciones, pero no se está aprovechando del todo hoy en día (IBM, 2022).

Tabla 2 Aplicaciones de la Visión artificial en la Industria

ÁREA DE LA INDUSTRIA	APLICACIÓN
Astronomía	Exploración espacial
Supervisión de calidad	Identificación de productos Etiquetados Inspección del estado de las piezas
Monitoreo del tráfico	Detección de placas Medición de velocidad Reconocimiento de vehículos
Robótica	Robots guiados
Agroindustria	Verificación en plantaciones Análisis en fotografías
Biomedicina	Investigación en imágenes de microscopía Resonancia Magnética

Nota. Las aplicaciones de visión artificial. Elaborado por: Marcelo Castro

1.2.7 Ventajas de la aplicación de visión artificial

- Incremento de la productividad
- Reducción de tiempos en el proceso productivo.
- Suprime la subjetividad en los sistemas de control de calidad.
- Examina la trazabilidad del producto en todo el proceso.
- Facilita una precisión mayor en el diagnóstico de los errores.
- Reduce el índice de residuos.
- Maximiza los niveles de seguridad disminuyendo riesgos.
- Previene la contaminación cruzada.

1.2.8 Diferencia de la visión por computador fundamentada en algoritmos y visión por computador fundamentada en Inteligencia Artificial

La visión por computador que emplea algoritmos dirigidos para reconocimiento de determinadas formas tiene madurez, es suficientemente robusta, y es ideal para la localización de objetos fáciles de reconocer y distinguir. (Intel,2022)

La visión por computador que se fundamenta en inteligencia artificial emplea prototipos de aprendizaje profundo (redes neuronales entrenadas) en la detección de elementos difíciles de reconocer y distinguir. (Intel, 2022)

1.3 IMPLEMENTACIÓN DE LA VISIÓN ARTIFICIAL

1.3.1 Principales etapas de una aplicación de visión artificial

1 Obtención de imágenes e importancia de las características visuales de los objetos

mediante técnicas fotográficas.

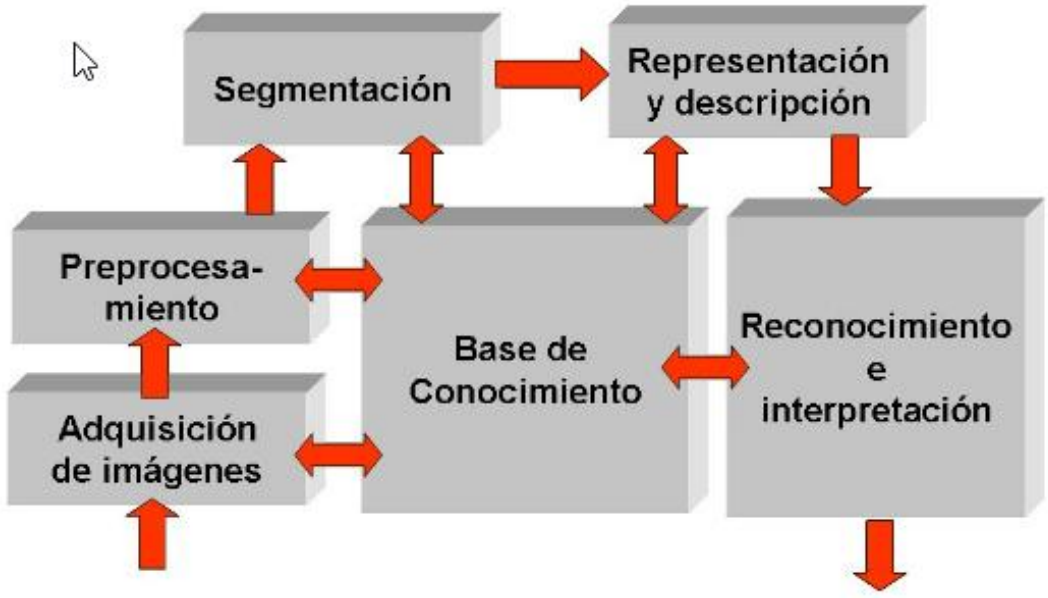
2 Preprocesamiento para optimizar la información de la imagen obtenida en donde se incluyen cálculos de desarrollo de la conexión de ruido-señal (denoising)

3 Segmentación de la imagen en zonas con concepto. Por ejemplo, las zonas de cultivo, de agua, carreteras, urbanas, determinadas en una imagen satelital.

4 Sustracción de los atributos de la imagen segmentada, principalmente de clase morfológica, como pueden ser perímetro, área, períodos de inercia, particularidad, estructuras, de igual forma, se puede asignar atributos basados en el color o la textura.

5 Organizar e interpretar los atributos analizados de cada región. Por lo tanto, serán diseñados organizadores para dar a cada área dividida una etiqueta que sea de alto nivel. Por ejemplo, en una imagen aérea cuales son las zonas que son áreas urbanas, tierras de cultivo, entre otras.

Figura 2 *Etapas de una aplicación artificial*



Nota. Fases en una aplicación de visión artificial. Fuente: (Platero, 2009)

Hay un grupo de técnicas para la clasificación:

- Sistemas Expertos
- Redes Neuronales
- Clasificadores Estadísticos
- Lógica Difusa

1.3.2 Métodos de aplicación en la visión artificial.

1.3.3.1 Fotografía y óptica: Procurar una iluminación que beneficie en la recopilación de las imágenes, la mayor parte de ocasiones necesita utilizar métodos profesionales para fotografía y video.

1.3.3.2 Procesamiento digital de la imagen: Se refiere a los algoritmos computacionales para convertir la imagen digital a una con mayor importancia.

1.3.3.3 Reconocimiento de patrones: dedicado a la correcta organización de las señales y dentro de estas, a la respectiva búsqueda de patrones existentes.

1.3.3.4 Computación de Imágenes: enfocada en la exposición visual de modelos geométricos. (Platero C, 2009)

1.4 OPENCV

1.4.1 Definición de OpenCV

Se le define como una Biblioteca de visión artificial de Código Abierto (Open Source Computer Vision Library) que permite la manipulación de imágenes y videos para realizar

varias tareas enfocadas en la identificación de objetos, rostros, reconocimiento de objetos en tiempo real y diferenciación de imágenes. (team OpenCV, 2022)

1.4.2 Aplicación de OpenCV

Está compuesta por más de 2500 algoritmos optimizados conforman un conjunto completo de algoritmos de aprendizaje automático y visión por computadora. Los algoritmos se pueden utilizar para identificación de rostros, detectar elementos, clasificación de imágenes, reconocimiento y seguimiento de objetos, modelado 3D, unir imágenes para producir una imagen de alta resolución, seguir los movimientos de los ojos, reducir los ojos rojos, utilización de la realidad aumentada.

1.5 RASPBERRY PI

La Raspberry Pi es considerada como un computador costo reducido que posee ciertas dimensiones, similar al tamaño de una credencial o una tarjeta de crédito, que además posee la característica que se puede conectar al monitor del computador o también una TV, y conectarle un teclado y mouse. En este pequeño computador se puede correr un Sistema Operativo Linux con la capacidad de que las personas que lo deseen puedan aprender a programar lenguajes como Scratch y Python o conocer y explorar más acerca de la computación.

Posee la capacidad de realizar la mayor parte de las tareas básicas del computador de escritorio, como la navegación, reproducción de vídeos de alta calidad, manipulación de documentos en ofimática, incluida la reproducción de videojuegos.

También, la Raspberry Pi es capaz de ser utilizada en muchos proyectos digitales, como reproductores de música y video, control de cámaras para seguridad. Se espera que sea notable que la Raspberry Pi puede usarse por adultos y niños de cualquier parte del mundo,

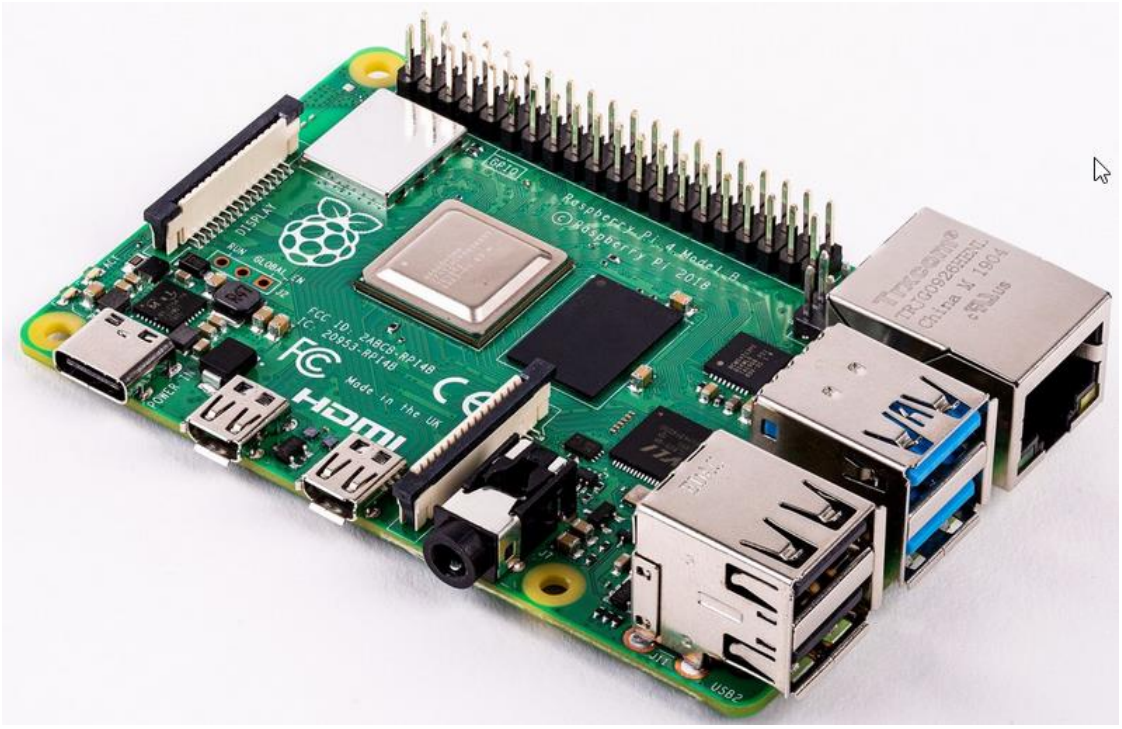
para conocimientos de programación y un entendimiento básico de cómo funciona el computador. (Web, 2021)

Tabla 3 Información Técnica de la Raspberry PI 4

RASPBERRY PI	
PROCESADOR	Procesador ARM Cortex-A72
GPU	VideoCore VI (Con soporte para OpenGL ES 3.x)
MEMORIA	4GB / 8GB LPDDR4 SSDRAM
CONEXIONES	Wi-Fi 802.11ac, Bluetooth 5.0, Gigabit Ethernet
VARIOS PUERTOS	GPIO 40 pines 2 x micro HDMI CSI (puerto para una cámara Raspberry Pi) DSI (puerto para una pantalla Táctil) 2 x USB 3.0 2 x USB 2.0 Micro SD Conector de audio Jack USB-C (entrada de energía)

Nota. Detalles y especificaciones basados en el último modelo de Raspberry. Fuente (Cristian Rus, 2019)

Figura 3 *Imagen de Raspberry Pi*



Nota. Imagen de la Raspberry Pi 4 Modelo B. Fuente (Cristian Rus, 2019)

1.6 PRINCIPALES LIBRERÍAS UTILIZADAS

1.6.1 Python

El algoritmo utilizado en la detección de vehículos en tiempo real y el módulo de control del semáforo han sido realizados completamente en Python, en los cuales se ha utilizado:

1.6.1.1 Numpy: por su eficacia con los arreglos, la recolección y procesamiento de datos de las imágenes recibidas, en el módulo controlador del semáforo se utilizó

1.6.1.2 RPI.GPIO: que permite la comunicación entre los puertos de la Raspberry con el circuito del semáforo.

1.6.1.3 psycopg2: librería utilizada en la conexión a la Base de Datos PostgreSQL.

1.6.2 PostgreSQL

Los datos recopilados de la detección y conteo de vehículos fueron guardados en una base de datos PostgreSQL.

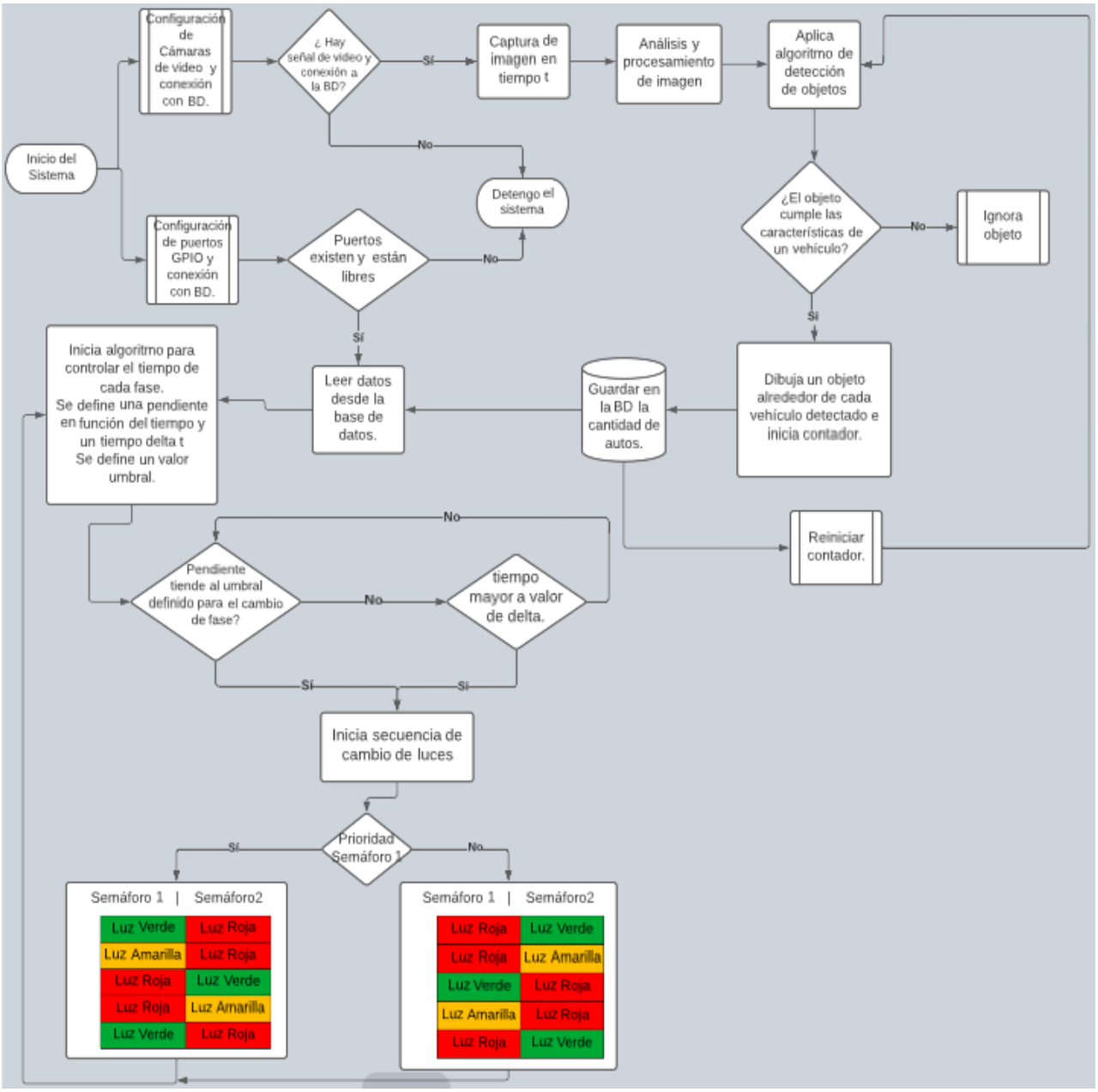
CAPITULO II
ANÁLISIS Y DISEÑO

2.1 Análisis para el desarrollo

En este capítulo se analiza y presenta las características técnicas del proyecto, los requerimientos, la viabilidad para la estructura y la construcción de un prototipo de semáforo inteligente.

Para este proyecto se estableció el siguiente diagrama de flujo del funcionamiento:

Figura 4 *Diagrama de flujo del funcionamiento*



2.1.1 Requerimientos funcionales

Este prototipo de semáforo inteligente está formado por dos módulos principales que son:

1 Módulo de visión por computador

2 Módulo controlador del semáforo

El Módulo de visión por computador debe:

- Mediante una primera cámara captar en video a la vía y con la ayuda de una segunda cámara captar el video de la intersección de la vía.
- Definir el área de interés sobre la cual se realizará la detección.
- Preprocesar la imagen captada para eliminar ruido y obtener datos de la imagen ordenados.
- Procesar la información segmentada y aplicar técnicas para reconocimiento de patrones.
- Utilizar algoritmos para detectar a los vehículos en la imagen luego del procesamiento de datos.
- Detectar los vehículos existentes en el video recibido y guardar la información en la base de datos.
- Realizar el conteo de vehículos en tiempo real y guardar esa información.
- Gestionar correctamente los recursos para el procesamiento en tiempo real sin ralentizar el sistema.

El Módulo controlador del semáforo debe:

- Leer la información almacenada en la base de datos con el fin de utilizar en el algoritmo que define el tiempo de cambio de la luz del prototipo del semáforo.
- Mediante el algoritmo de tiempo determinar el cambio de fase de cada semáforo de manera dinámica en base a los datos recibidos por el módulo de visión por

computador.

- Configurar y asignar a cada semáforo la luz correcta para controlar el tráfico existente en cada vía en tiempo real.
- Conectarse correctamente con el circuito del semáforo.
- Cambiar dinámicamente dando prioridad a la vía con mayor número de vehículos.
- Se realiza la simulación para comprobar el correcto funcionamiento del módulo controlador del semáforo.
- Tener la capacidad de calcular de manera rápida los datos recibidos en tiempo real.

2.1.2 Requerimientos no funcionales

- Las herramientas para utilizar en el desarrollo serán Open Source.
- El lenguaje utilizado será Python
- Para la conexión entre el computador y la Raspberry se utilizará el protocolo SSH
- El circuito deberá ser probado con números de vehículos determinados y aleatorios.

2.1.3 Factibilidad técnica del software

Para el desarrollo de los módulos del proyecto y toda la programación realizada se utilizaron las siguientes herramientas a nivel de Software:

- Lenguaje de Programación Python.
- Base de Datos Relacional Postgresql
- PgAdmin4
- Biblioteca OpenCV
- Sistema Operativo Ubuntu
- Editor de Código Visual Studio Code
- MATLAB Online

2.1.4 Factibilidad técnica del hardware

El siguiente es en base a los costos de los productos adquiridos

Tabla 4 Análisis de factibilidad a nivel de Hardware

Recurso	Detalle Técnico	Descripción
Laptop	SSD de 1TB 16 GB de RAM Procesador Intel Core i7	Para desarrollar la programación.
Raspberry Pi 4	8GB de RAM 32Gb de tarjeta de memoria.	Para implementar el módulo controlador del semáforo
Cámara web	Resolución 1080p	Para captar las imágenes
Circuito del semáforo	3 diodos LED por circuito, 2 resistencias	Para representar un semáforo real

Nota. Detalles técnicos para el hardware requerido. Elaborado por: Marcelo Castro

CAPITULO III

DESARROLLO

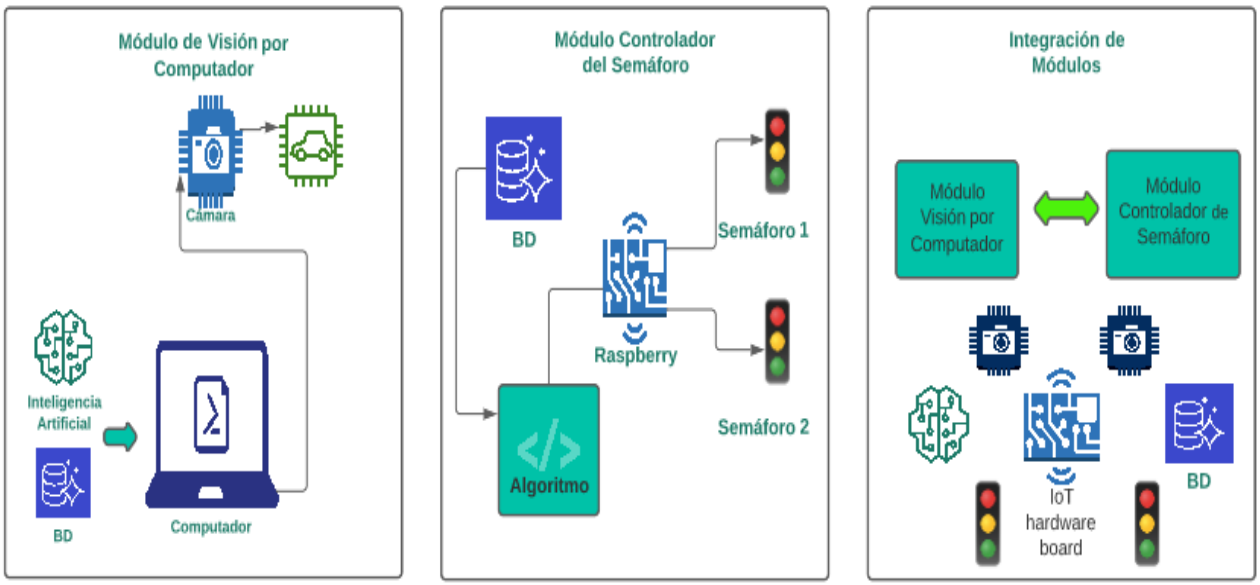
3.1 INICIO DEL DESARROLLO

A partir de un programa desarrollado en el lenguaje Python en la Herramienta Visual Studio Code tuvo inicio esta metodología de investigación en la cual se utilizó varias librerías entre ellas las librerías Numpy y las librerías de OpenCV que son importantes para el desarrollo del proyecto por la eficacia para el procesamiento de datos y manipulación de imágenes, con mayor relevancia esta última puesto que es muy utilizada para la detección de objetos mediante visión por computador e inteligencia artificial con lo cual se ha llevado a cabo la obtención de imágenes, detectando errores, definiendo las diferentes variables que son pertinentes para el objeto de estudio como son el tiempo de cada fase del semáforo, la cantidad de vehículos y el tiempo que necesita cada vehículo para cruzar la intersección.

3.2 FASES DEL DESARROLLO

Las fases del desarrollo son las que se muestran a continuación:

Figura 5 *Fases del desarrollo del prototipo.*



3.3 MODULO DE VISIÓN POR COMPUTADOR

3.3.1 Obtención de la imagen

Se ha realizado la obtención de video enviada desde cada cámara web mediante OpenCV con el algoritmo VideoCapture y en cada paréntesis se debe colocar el número de cámara que se encuentra conectado al computador iniciando desde 0, en este caso se ha utilizado los números 1 y 2 ya que el número 0 es la cámara web fija que viene de fábrica en el computador portátil, en el anexo 1 se presenta el código de lo anteriormente descrito.

3.3.2 Configuración del área a procesar

Con la ayuda de la librería Numpy de Python, que es ideal para realizar cálculos matemáticos y de mucha utilidad en el manejo y utilización de vectores y matrices, se va creando los arreglos con los valores de cada imagen captada en tiempo real, y posteriormente se va ordenando los datos de entrada como arreglos asignados a una variable y mediante este proceso se consigue optimizar el proceso de segmentación de la información para tratar los datos que se obtienen de la imagen recibida así como se muestra en el anexo 1.

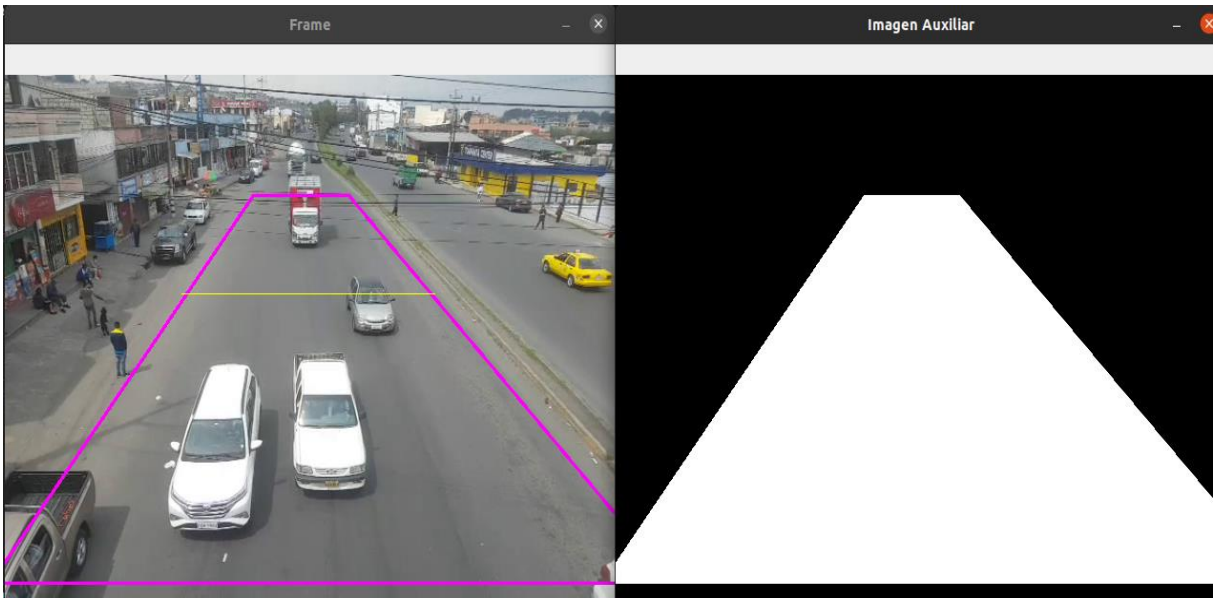
3.3.3 Procesamiento de la imagen

Posterior al almacenamiento y la captura de la imagen mediante las distintas técnicas hasta la obtención de una imagen binaria generada correctamente para el estudio, se prepara esta imagen para el procesamiento, para centrar la detección solo en determinada área y no en toda la imagen surge la necesidad de manipular y extraer parte de la imagen recibida y así disminuir el procesamiento y ahorrar recursos, para lo cual con la librería numpy se ha declarado y guardado en un arreglo lleno de ceros que será utilizado como una imagen auxiliar, del mismo ancho y alto de la imagen inicial recibida, en esta imagen auxiliar se determinó el área sobre la cual actuará el detector de vehículos la cual primero la dibujamos en color blanco.

Para definir el área de interés y dibujar esta área sobre la cual se va a trabajar con la detección

de los vehículos que ingresen en la imagen, se utilizó el algoritmo *drawCountours()* de OpenCV, tal como se muestra en el anexo 1.

Figura 6 *Imagen auxiliar*

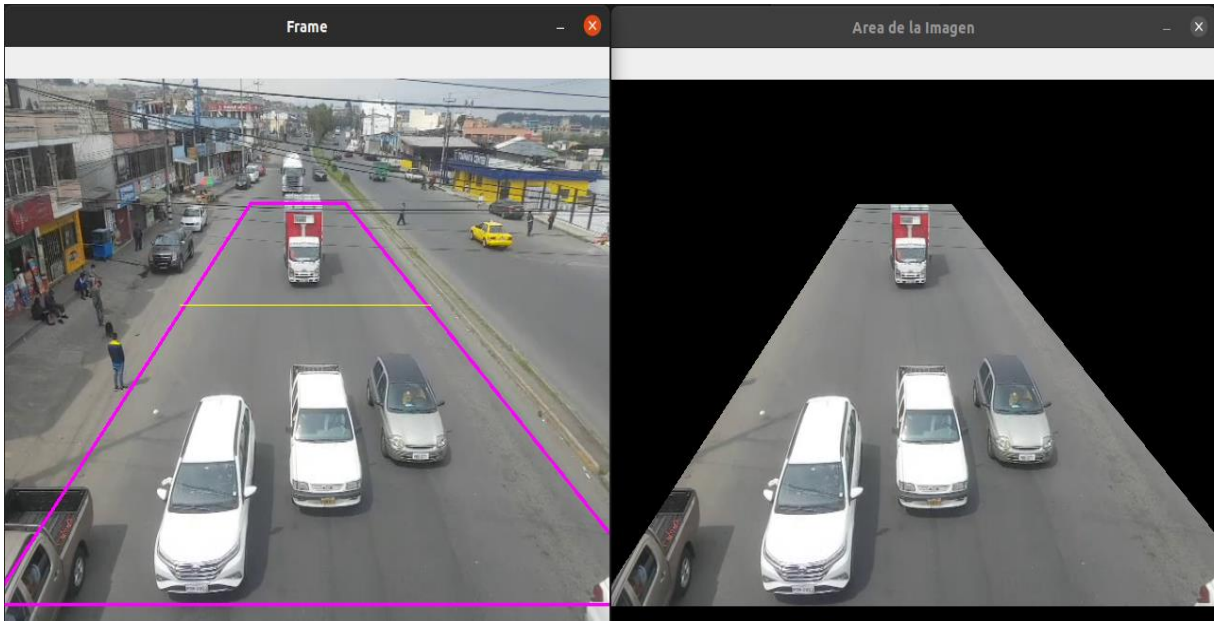


Nota. *Muestra de imagen auxiliar y recorte del área de interés.*

Elaborado por: *Marcelo Castro*

El algoritmo *bitwise_and* se utilizó para que la imagen guardada se muestre en esa área blanca y visualizar únicamente el área de interés con los vehículos que ingresan en esa área.

Figura 7 *Área de Interés*



Nota. *Presentación del área de interés en la imagen auxiliar. Elaborado por: Marcelo Castro*

Para el procesamiento se ha utilizado la técnica de sustracción de fondo con lo cual se elimina el fondo de la imagen, para ello el algoritmo extrae el primer plano en movimiento del fondo estático. Como resultado de esto se obtiene una imagen binaria que será procesada luego con kernel que mejorará dicha imagen binaria.

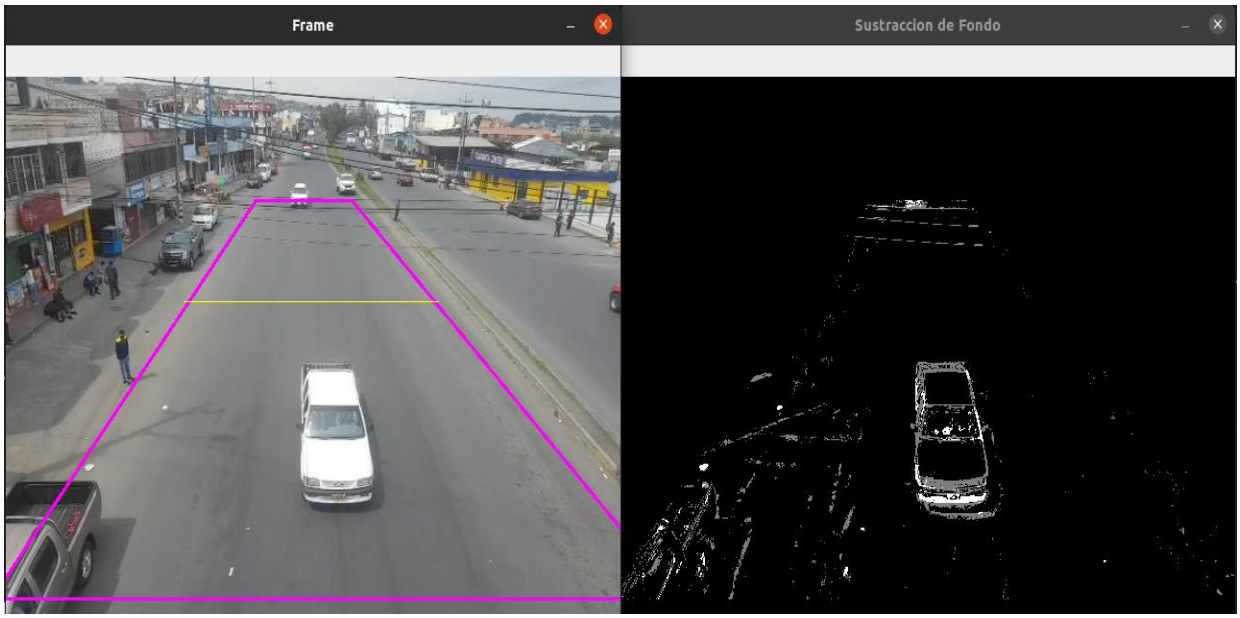
Utilizamos el algoritmo createBackgroundSubstractorMOG2 para aplicar la sustracción de fondo y el algoritmo getStructuringElement para el mejoramiento de la imagen binaria, los dos algoritmos son de la librería cv2 (anexo 1)

Al área definida anteriormente se le ha aplicado la sustracción de fondo para un óptimo procesamiento de un área definida específicamente y no a toda la imagen con lo que mejora el proceso ya que es solo cierta parte de la imagen lo que implica menos datos que procesar y menos recursos que utilizará el módulo de visión por computador.

Con morphologyEx mediante una simple operación basada en la forma de la imagen, se consigue utilizar una imagen binaria sobre la cual se aplica una transformación morfológica del objeto detectado, y se alcanza un mejoramiento del objeto. Consta de dos entradas, la imagen original y un elemento estructurante o kernel con el que se decide la naturaleza de la operación, para el proyecto se ha aplicado las variantes de entrada (cv2.MORPH_OPEN) y salida (cv2.MORPH_CLOSE), con la variante de entrada se logra eliminar el ruido de la imagen y con la variante de salida se logra eliminar pequeños agujeros dentro del objeto.

Con el algoritmo cv2.dilate, que su declaración y utilización se puede ver en el anexo 1, se consigue que nuestro objeto se agrande o dilate, lo cual es necesario dado que nuestro objeto u objetos en cuestión han sido reducidos después de la eliminación de ruido, ya que luego de esto todo objeto detectado y procesado tiende a ser más pequeño, este proceso se repite 5 veces ya que con esto se ha obtenido un mejor resultado.

Figura 8 *Imagen de sustracción de Fondo*



Nota. Imagen original a la izquierda / Imagen aplicada la sustracción de fondo a la izquierda.
Elaborado por: Marcelo Castro

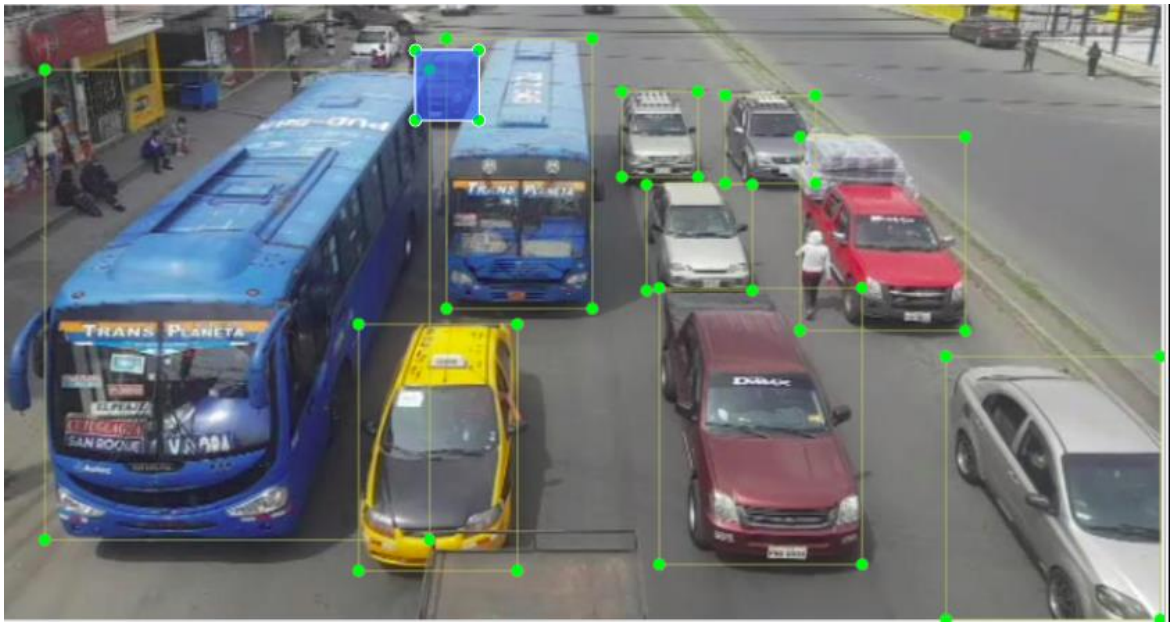
3.3.4 Detección de los Vehículos

Para detectar los vehículos se realizó varios intentos para que el algoritmo logre detectar correctamente y con el menor valor posible de errores, primero se realizó una práctica con varios algoritmos de detección de objetos como por ejemplo el sistema YOLO (You Only Look Once) que significa que detecta un objeto a la primera visualización. Pero la desventaja de este método fue que la necesidad de poder entrenar al algoritmo con los vehículos de carga liviana y pesada que circulan en la ciudad y la demanda de recursos necesarios para procesar la información.

Otro algoritmo que se utilizó es el algoritmo de Haar Cascades pero está desarrollado y enfocado especialmente para detección de rostros y reconocimiento facial.

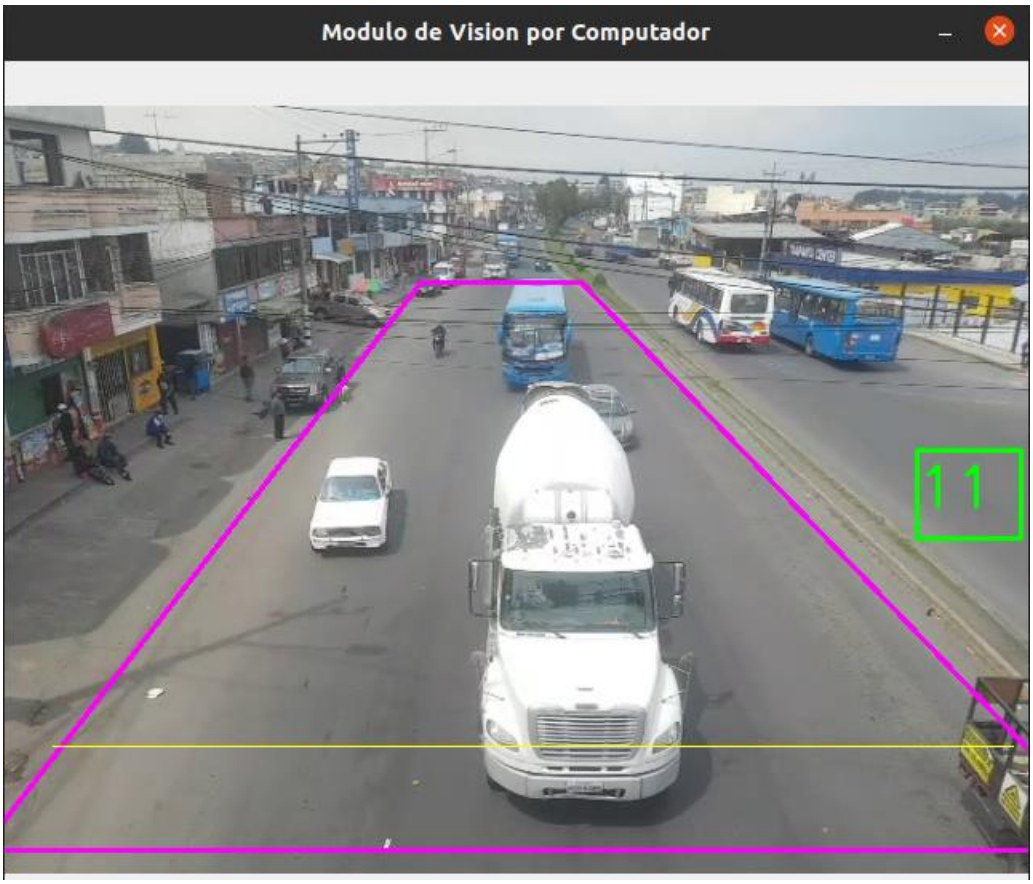
La detección de los vehículos se ha realizado con OpenCV, así, mediante el procesamiento de imagen que se le aplica a cada imagen recibida se logra la identificación de los autos tal como se puede observar en la imagen.

Figura 9 *Detección de vehículos*



Nota. Imagen de reconocimiento de vehículos por medio del módulo de detección creado.
Elaborado por: Marcelo Castro

Figura 10 *Módulo de visión por computador*



3.4 MÓDULO DE CONTROL DEL SEMÁFORO

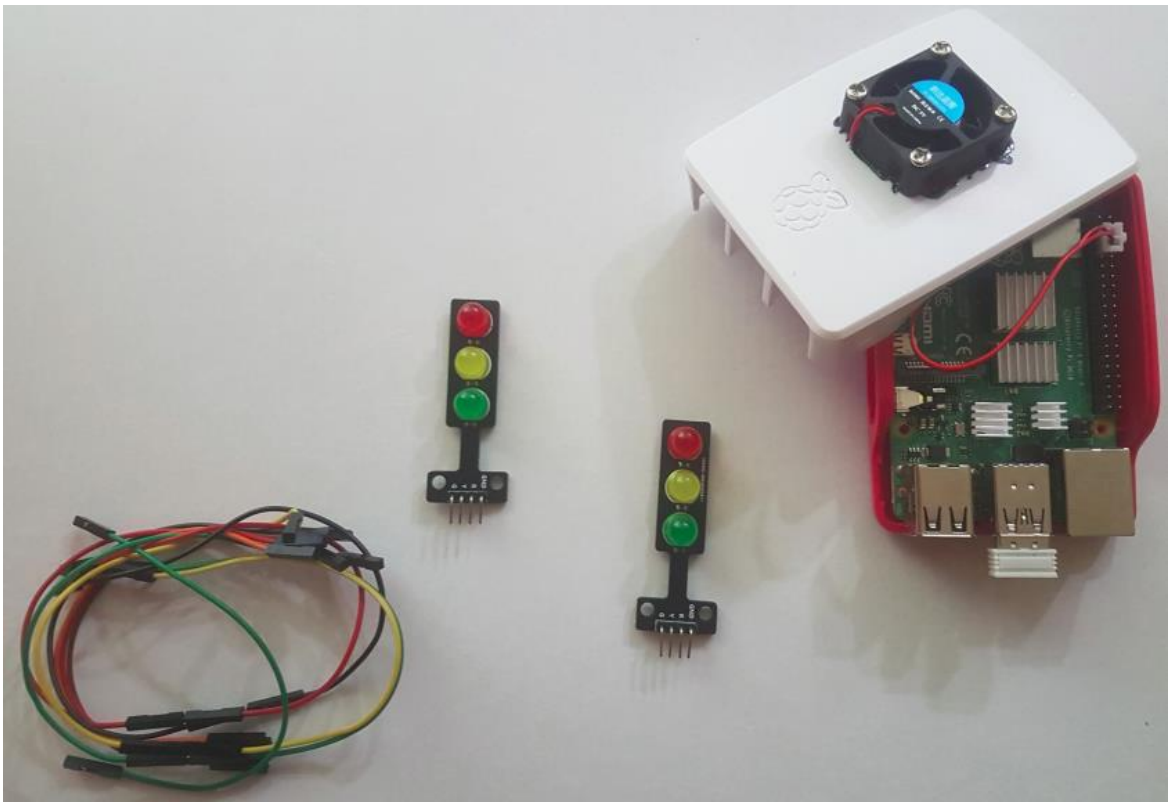
Este módulo se ha realizado para gestionar el tiempo dinámicamente en base de los datos obtenidos a partir del módulo de visión por computador, definir un tiempo en el cual cada fase del semáforo va a ejecutarse, lo cual será en tiempo real y servirá para que el flujo de cada vía se vaya disminuyendo, así como el tiempo en el que los vehículos crucen dicha intersección disminuya, mejorando el tránsito en la intersección en donde se coloque el prototipo del semáforo inteligente.

3.4.1 Enlace a la Base de datos

Se realizó una conexión con la base de datos en la que se usó la librería psycopg2 para extraer los datos de la detección de vehículos han sido cargados en la base de datos y así llevar un control de registro de cada instante mientras estuvo en ejecución el módulo de visión por computador y que posteriormente estos datos serán utilizados por el módulo de control del semáforo.

3.4.2 Implementación del semáforo en la Raspberry

Figura 11 *Raspberry con circuitos de los dos semáforos.*



Nota. Raspberry Pi 4 Modelo B y dos circuitos de semáforo. Elaborado por: Marcelo Castro

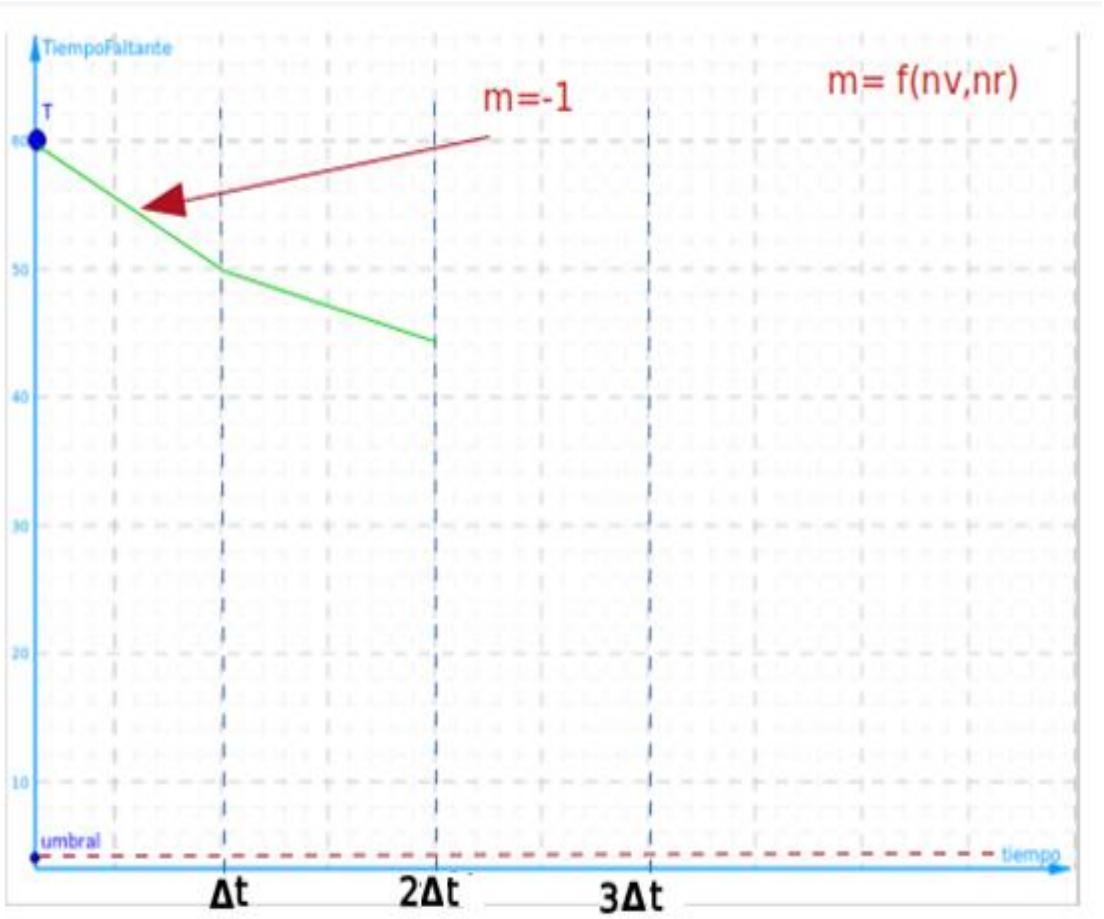
Para conseguir que el circuito del semáforo funcione en la Raspberry fue necesario configurar sus puertos llamados **GPIO** (General-Porpouse Input/Ouput), que son la interfaz física entre la Raspberry y el circuito del semáforo, los cuales trabajaran en base a la luz del semáforo a la cual se la va a asignar para su ejecución en el código, ver anexo 1.

3.4.3 Fases de tiempo del semáforo

Al ejecutarse en tiempo real el módulo de visión por computador se recopila datos de las cantidades de autos de cada vía en todo momento, por lo que fue necesario contar con la capacidad de consultar y procesar los datos también en tiempo real por parte del módulo controlador del semáforo.

Para conseguir dicho objetivo ha sido necesario el desarrollo de un algoritmo que estará constantemente calculando y comparando la cantidad de vehículos en las dos vías todo el tiempo y en base a esa información define el tiempo faltante para el cambio de fase, y que gracias a un umbral definido y un valor delta también definido, consigue calcular el valor de una pendiente con la que se definirá el cambio de luz verde y la duración de esta fase para dar prioridad a la vía con mayor cantidad de vehículos y así permitir optimizar la circulación.

Figura 12 *Gráfica de la función T(t)*



Nota. Elaborado por: Marcelo Castro

Tabla 5

Símbolo	Descripción
T	Tiempo que falta para cambiar de fase en cada semáforo y que irá disminuyendo en base a la cantidad de vehículos existente en la vía.
t	Período de tiempo en el que se evaluará la cantidad de autos.
m	La pendiente que será dinámica será una función que dependerá del número de autos en la vía que tiene la luz en verde y el número de autos de la vía con el semáforo en rojo.
Umbral	Valor definido al que nuestra pendiente **m** tenderá en caso de no haber vehículos o existir menos vehículos en la vía con el semáforo en verde, para dar paso a los vehículos de la otra vía.

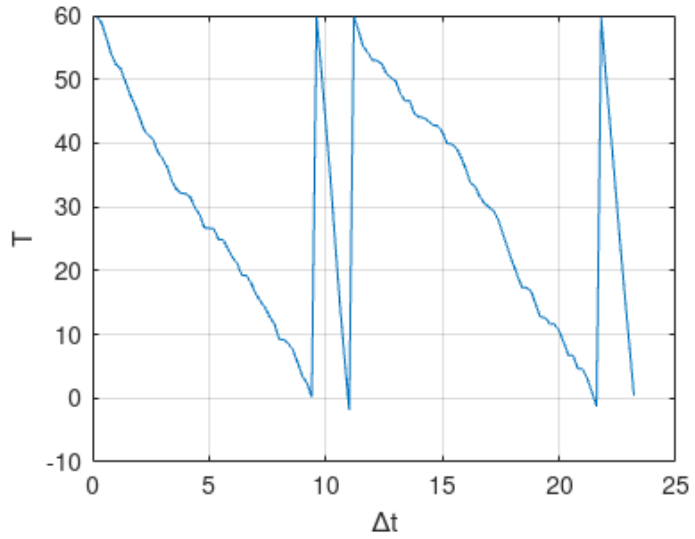
Elaborado por: Marcelo Castro

De esta manera, en cada intervalo **t** se evaluará la cantidad de autos para actualizar nuestra pendiente y ver si se mantiene o tiende al umbral definido para permitirle al semáforo cambiar de fase.

3.4.3.1 Programación en MATLAB: fue utilizada para el desarrollado del algoritmo y conseguir graficar adicionalmente los datos obtenidos del algoritmo en 2 gráficas, la primera fue la gráfica de la función T(t), en esta gráfica muestra a la pendiente *m* en un período de varios cambios de fase del semáforo.

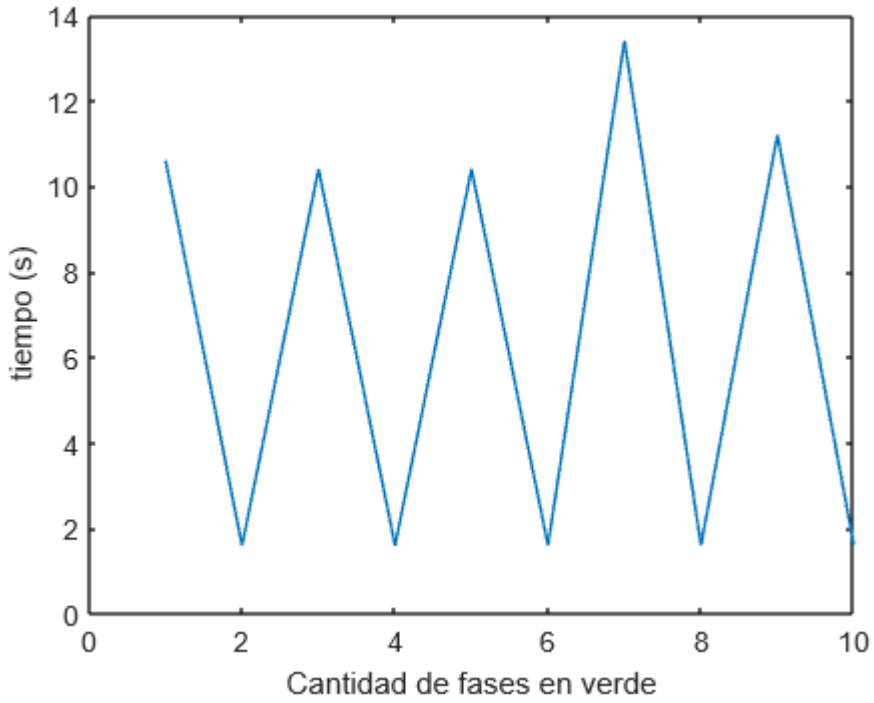
La segunda es la gráfica que muestra el número de veces que el semáforo se mantuvo en luz verde en cada período de tiempo mientras se ejecutaba el programa.

Figura 13 *Función grafica T(t)*



Elaborado por: Marcelo Castro

Figura 14 *Cantidad de fases en verde durante la ejecución en un tiempo t*



Nota. Cantidad de fases en verde para 20 cambios de fase en base al tráfico. Elaborado por: Marcelo Castro

El código desarrollado en MATLAB se encuentra en el anexo 2.

3.4.3.2 Programación en Python: fue utilizada para aplicar el código en el módulo de control del semáforo y poder realizar las pruebas de funcionamiento del algoritmo aplicado en una animación con tráfico y flujo controlado para ver lo dinámico del tiempo en cada fase, ver anexo 3.

CAPITULO IV
PRUEBAS

4.1 PRUEBAS DE FUNCIONALIDAD DE LOS MÓDULOS

Para ejecutar las pruebas de funcionalidad se segmentó la funcionalidad del prototipo en dos partes para probar cada módulo independientemente y determinar los posibles escenarios en los que deberá funcionar.

Se desarrolló las pruebas con la finalidad de disminuir los errores que se pueden presentar y para obtener una madurez en el algoritmo tanto del módulo de visión por computador como con el módulo de control de semáforo.

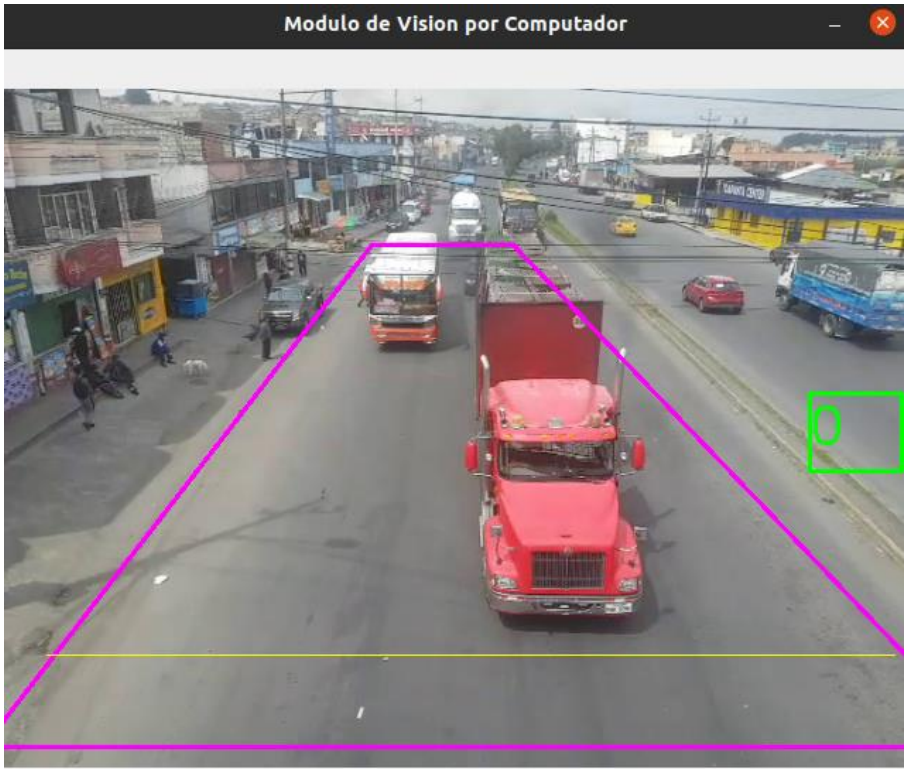
4.1.1 Módulo de visión por computador

En el módulo de visión por computador se realizó pruebas con un video pregrabado en el cual circula el tráfico normalmente en la vía para probar la detección, primero se realizó la detección sin técnicas de procesamiento a cada cuadro de video recibido, y posteriormente se realizó la detección pero con una imagen ya aplicada las técnicas para ver qué tan efectivo fue, y tuvo mejor rendimiento el módulo de visión por computador, pero adicional se determinó luego de prueba y error, la limitación que se tiene en la calidad de video, al moverse la cámara y del exceso de movimiento, en cada uno de estos limitantes puede influir las condiciones climáticas para obtención optima o decadente del video, por lo cual se tuvo que probar en varios videos para tener la seguridad que funciona correctamente y con el menor margen de error.

Adicionalmente de comprobó con autos de juguete para comprobar el conteo del módulo de visión por computador para probar el módulo con visión en tiempo real.

4.1.1.1 Caso 1. ***Detección Inicial sin técnicas de eliminación de ruido:*** Detección de vehículos en el módulo de visión por computador, se realiza la prueba con video pregrabado en una intersección, la detección resulta ser más efectiva en un 80% al aplicar las diferentes técnicas para eliminar el ruido de la imagen. Por ejemplo, en la Figura 22 se realizó una prueba sin eliminar el ruido de la imagen y no detectó al tráiler rojo.

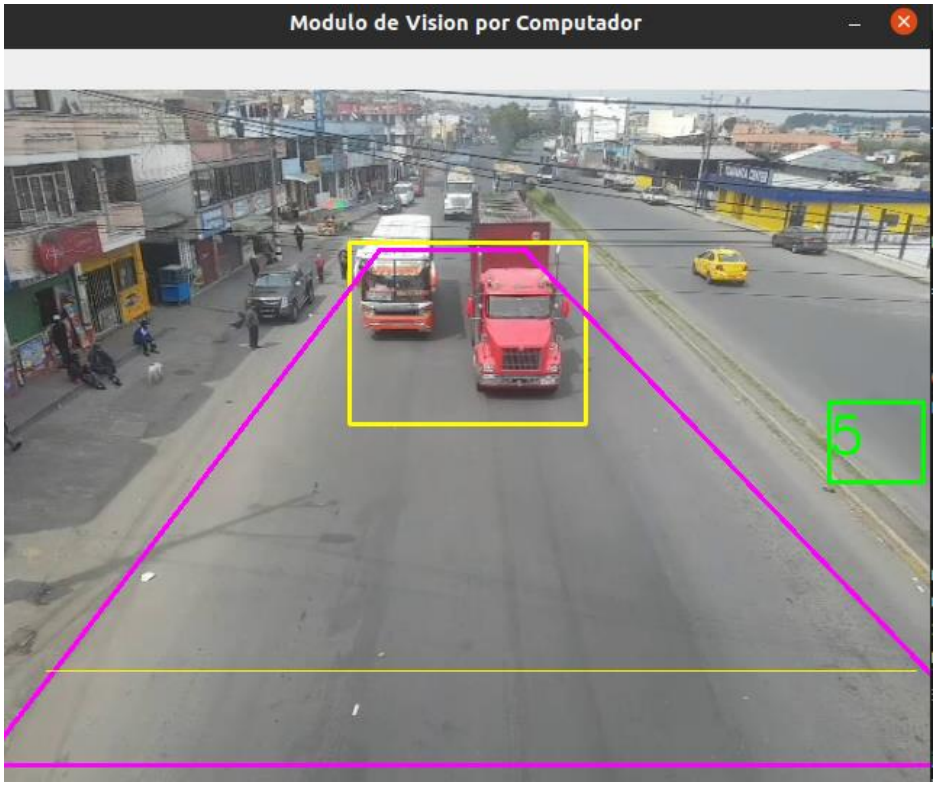
Figura 15 *Prueba de detección sin limpieza de ruido de la imagen*



Se logra una detección más efectiva en relación con el procesado de imagen sin aplicar las técnicas para la eliminación de ruido de los cuadros del video, como son:

- Sustracción de fondo
- Morfología del objeto
- Dilatación de la imagen

Figura 16 *Prueba de Detección de vehículos con las técnicas de limpieza de ruido aplicadas.*



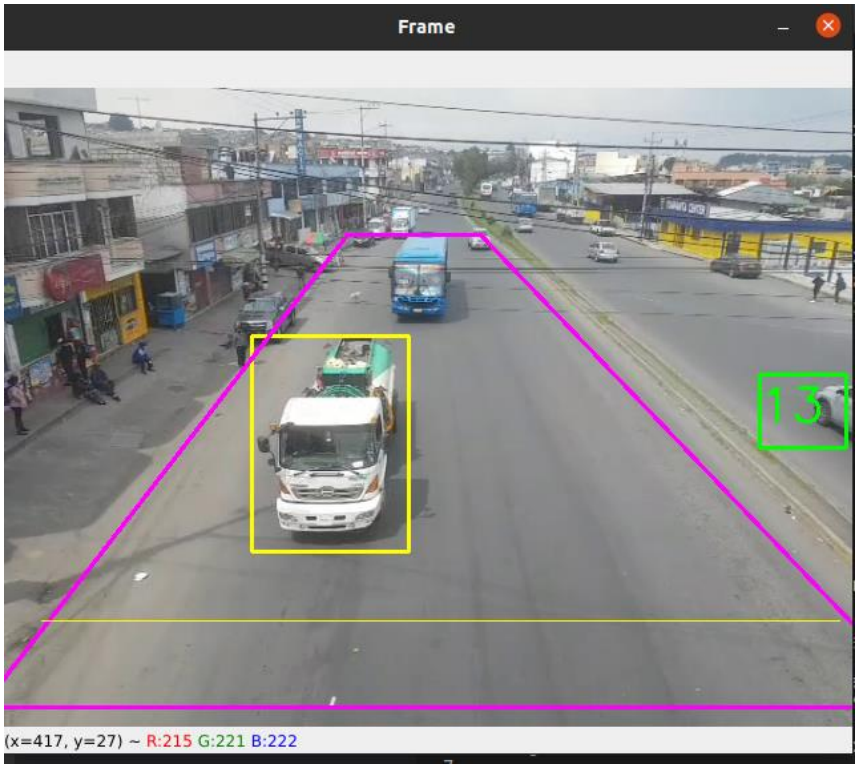
Nota. Se detecta al vehículo luego de aplicar la eliminación de ruido. Elaborado por: Marcelo Castro.

4.1.1.2 Prueba de Detección de vehículos tipo liviano y de carga: Fue un reto detectar los autos grandes ya que por su dimensión el módulo no los detecta de la misma manera y no es tan efectivo al detectar un automóvil que un camión, del tipo de vehículo depende mucho la efectividad de la detección del módulo, ya que con los vehículos pequeños no hay inconveniente pero con los vehículos de tipo pesado existía inconvenientes, una de las cosas que hay que tomar en cuenta es la calidad de video y que la cámara se encuentre estática ya que al definir una área trazada por pixeles, al menor movimiento esa área cambiaba y el movimiento en la imagen presenta falsos positivos y falsos negativos en la detección y para ello se trabajó con varias repeticiones hasta lograr un margen de error del 20 % menos.

Figura 17 *Prueba de Detección de un vehículo tipo liviano.*

Nota. Detección de un vehículo tipo liviano. Elaborado por: Marcelo Castro

Figura 18 *Prueba de Detección de un vehículo tipo pesado.*



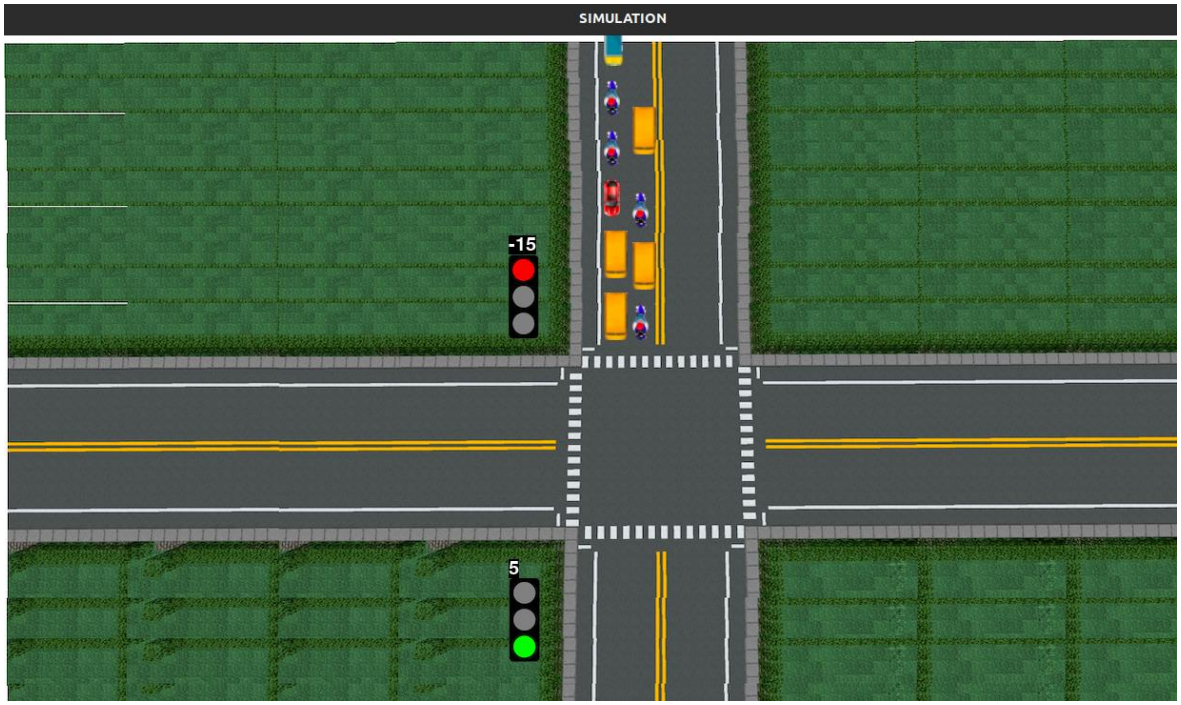
Nota. Detección de un vehículo tipo pesado. Elaborado por: Marcelo Castro

4.1.2 Módulo de Control de Semáforo

Se realizó las pruebas con una simulación desarrollada en Python para verificar la eficacia del funcionamiento del módulo de control del semáforo con tráfico controlado que sirvió para probar los cambios de fase y comprobar que sean dinámicamente en base al tráfico existente en cada vía, en donde también se calculó el tiempo promedio de todos los vehículos desde que empezaron a circular por la vía, primero con el tiempo predeterminado en cada semáforo y luego con el algoritmo aplicado para hacer el cambio de fase dinámicamente.

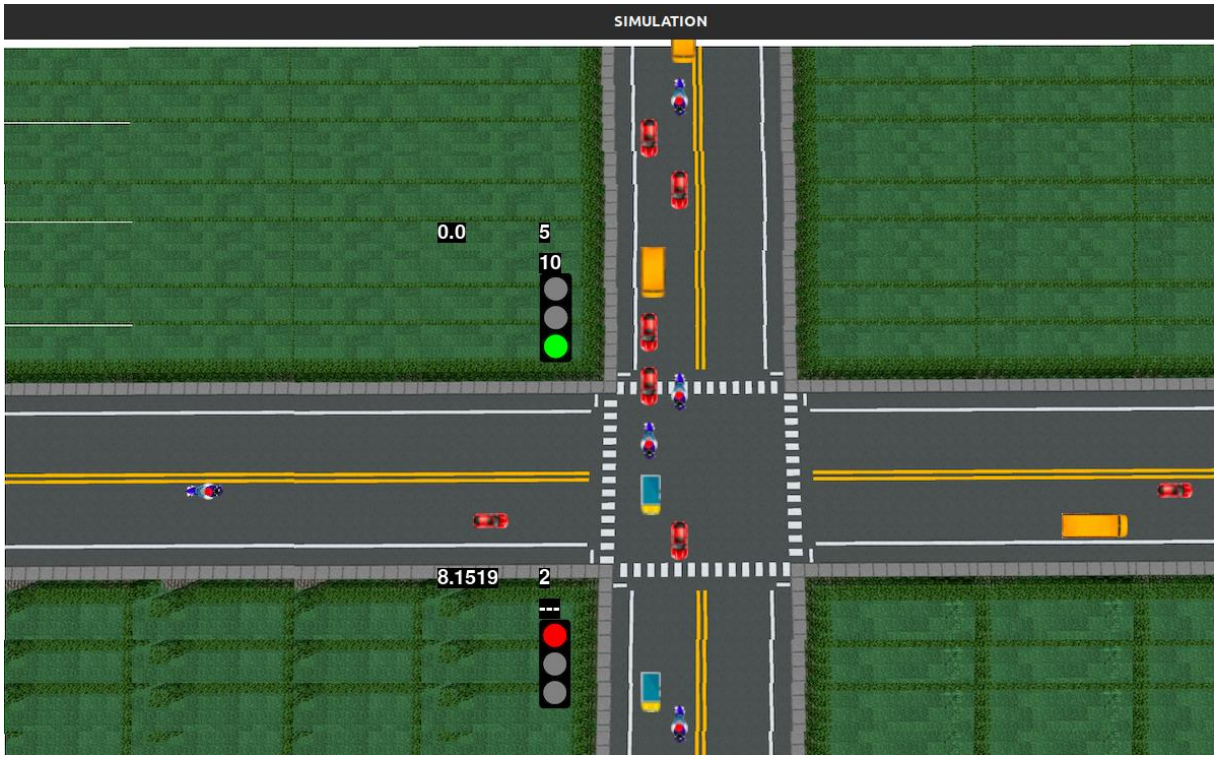
4.1.2.1Semáforo convencional: tiene el tiempo predeterminado para cada fase siendo notable la diferencia con el prototipo del semáforo inteligente, puesto que con el semáforo convencional sus cambios de fase no actúan dinámicamente para dar preferencia a la vía con más vehículos, no realizan un conteo de vehículos para su análisis en tiempo real, y no toma el tiempo promedio que tardan en cruzar la intersección.

Figura 19 *Simulación utilizando el módulo controlador del semáforo*



4.1.2.2 Prueba del módulo controlador del semáforo: con el algoritmo para el tiempo dinámico ya aplicado para que el semáforo decida dar más tiempo en su fase de color verde a la vía de la intersección con más cantidad de vehículos, se obtuvo una mejora del 100% y el promedio de tiempo que tardan los vehículos disminuye notablemente y de esta manera se diferencia del semáforo convencional teniendo un tiempo de cambio de fase dinámico y dependiente y también calculando el promedio de tiempo que tardan en pasar los vehículos.

Figura 20 *Simulación utilizando el módulo controlador del semáforo*



Nota. Simulación de una intersección con tráfico implementado el semáforo inteligente. Elaborado por: Marcelo Castro

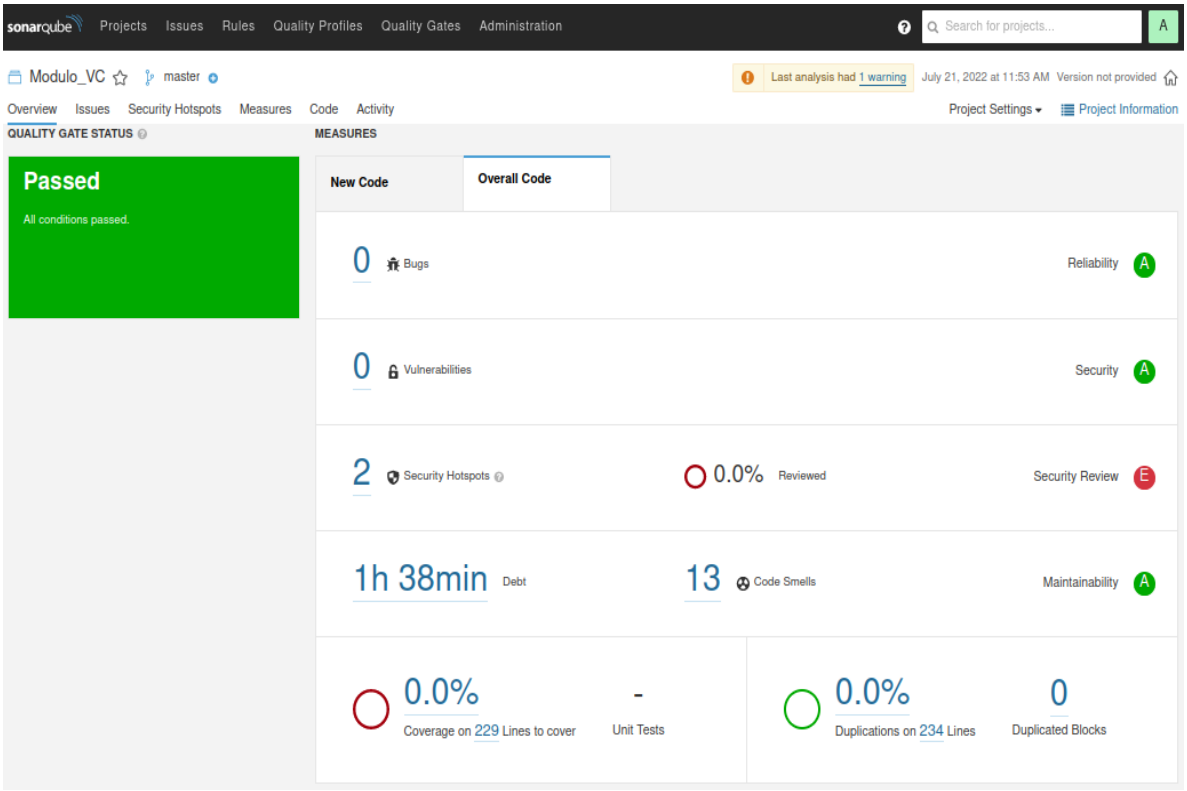
4.2 RESULTADOS

4.2.1 Resultado de pruebas de código.

Se han realizado las pruebas del código desarrollado, con la ayuda de la herramienta Sonarqube, la cual sirve para evaluar y determinar el estado de las distintas métricas de desarrollo en cuanto a la calidad, el rendimiento, la seguridad y la mantenibilidad del código, con lo cual luego de las correcciones se logra un código eficiente, seguro, mantenible y que puede ser escalable.

El log de ejecución se puede ver en el Anexo 4.

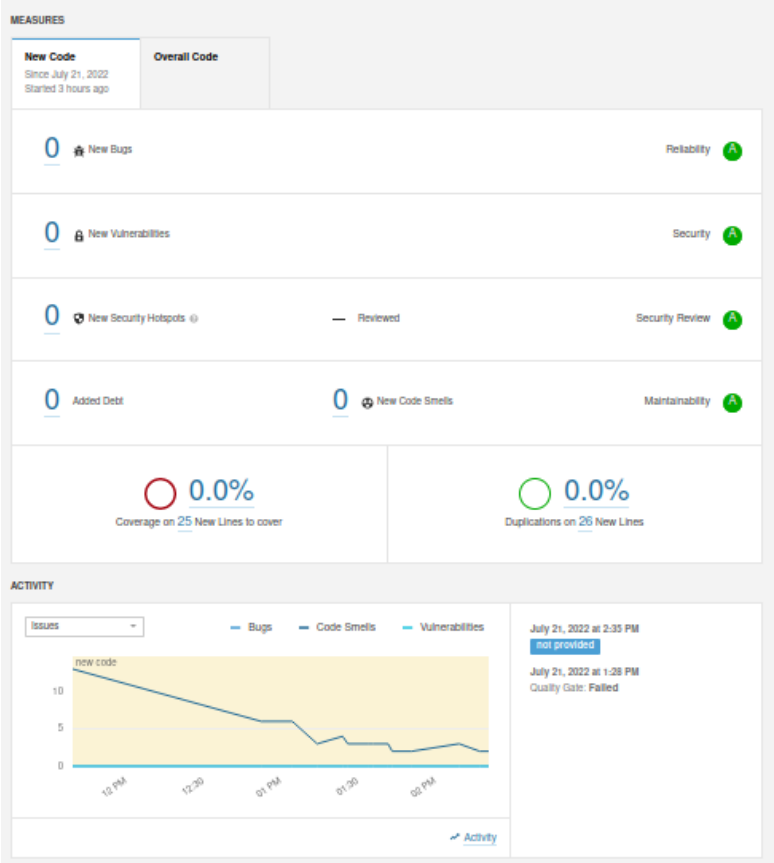
Figura 21 *Resultado de la ejecución en Sonarqube.*



Luego de realizar las mejoras en cuanto a mantenibilidad del código, eliminación de duplicados, eliminar variables sin usar, corregir las excepciones para el manejo de errores, de optimizar el código en cada ejecución realizada de la prueba, se obtuvo el resultado que se

muestra en la Figura 22.

Figura 22 Resultados luego de las modificaciones



4.2.2 Resultado de pruebas funcionales.

Al realizar las pruebas funcionales se consiguió el resultado esperado en cada uno de los procesos implementados para probar la funcionalidad, cumpliendo así el objetivo y con los requerimientos deseados.

Al realizar las pruebas funcionales se consiguió el resultado esperado en cada uno de los procesos implementados para probar la funcionalidad, cumpliendo así el objetivo y con los requerimientos deseados.

Tabla 6

CASO	PRUEBAS	REPETICIÓN	OBSERVACIÓN	ESTADO
1	Verificar que capture correctamente el video.	125		Aprobado
2	Verificar que se pueda leer la imagen guardada.	122		Aprobado
3	Definir el área en cuestión para la detección.	122		Aprobado
4	Aplicación de las técnicas para mejoramiento de la imagen	160		Aprobado
5	Aplicación de técnicas para detección de los vehículos.	190		Aprobado
6	Salida de la imagen con detección de los vehículos.	180		Aprobado

CONCLUSIONES

Mediante el desarrollo del proyecto se concluyó que el tiempo que tardan los autos en cruzar la intersección disminuyó de 10 a 6 segundos en un período de 1 minuto determinando que el prototipo cumple con el objetivo de disminuir el tiempo de movilidad en una intersección.

Con el prototipo fue posible llevar un conteo de los autos que cruzaron en la intersección y se comparó con el número de autos que cruzaron con el uso del semáforo convencional que se tuvo que contar manualmente y se determinó que en un minuto cruzaron 33 autos con el semáforo inteligente y con el semáforo normal cruzaron 21 autos lo que demuestra que el prototipo mejora la circulación y además lleva un registro de los autos que cruzan por la intersección.

Para el procesamiento de la imagen y la detección de los vehículos la memoria RAM del computador (DELL CORE i7de 11va Generación con 16GB de RAM) se demoró 5 segundos más con el algoritmo YOLO y ralentizaba el video en relación con el algoritmo Haar Cascades que para un video de 30 segundos no tuvo demora extra ni ralentización del video.

Al momento de desarrollar el módulo de visión por computador se tuvo que lidiar con varios temas como la iluminación, el movimiento de la cámara, el clima, el polvo,

entre otros, ya que dificultaban a la detección correcta de los vehículos, fue uno de los inconvenientes más comunes con los que en varios proyectos similares tuvieron que tratar.

Para desarrollar la detección de vehículos existen varias maneras cada una tiene sus ventajas y particularidades, sin embargo, cuando se aplica la detección en lugares en donde no circulan solo vehículos de tipo liviano, sino además buses y transporte

pesado o ciertas motocicletas que circulan en distintos carriles el algoritmo tenía falsos positivos en 19 de 25 ocasiones que se realizó las pruebas.

Se obtuvo un algoritmo que permite la identificación de afluencia vehicular en determinado carril, cambiando ágilmente la luz del semáforo para ceder el paso a los vehículos del carril que tiene afluencia, evitando esperar que cambie el color del semáforo en un carril sin vehículos.

RECOMENDACIONES

Debido a la facilidad de implementación y a su bajo costo, es recomendable que se aplique en varios sectores de la ciudad, para ir ajustando a las necesidades locales de optimización de movilidad.

Se recomienda gestionar una reunión con el Municipio del DMQ para revisar la capacidad operativa de los sistemas de semaforización y mencionar los cuellos de botella generados en el tráfico vehicular en toda la ciudad y dar a conocer el prototipo de semáforo inteligente desarrollado.

Al realizar un proyecto relacionado al manejo de imágenes adquiridas desde una cámara se recomienda tener en cuenta la ubicación de la cámara para poder obtener una mejor toma ya que influye mucho la iluminación, el movimiento y la posición de la cámara.

Se debe tener en cuenta que la capacidad de procesamiento al manejar imágenes es alta por lo cual se recomienda utilizar el modelo de Raspberry Pi 4 de 8GB de RAM por su demanda de memoria de procesamiento y además es recomendable tener la tarjeta de memoria que sea de categoría 10 para no sufrir perdida de la información, debido al número de información a procesar.

LISTA DE REFERENCIAS

Baker , K. D., & Sullivan, G. D. (1992). *"Performance assessment of model based tracking," in Proc. IEEE Workshop Applications of Computer Vision, Palm Springs.* California.

Garber , J., & Hoel, L. (2015). *Traffic & Highway engineering* (5 ed.). Standford.

García, A. (2012). *Inteligencia Artificial. Fundamentos, práctica y aplicaciones.* Madrid, España: RC Libros.

Youyang Qu;Liang Jiang;Xinping Guo 2016 2nd International Conference on Control, Automation and Robotics (ICCAR)

team, O. (2022). *OpenCV.org*. Obtenido de https://opencv.org/ateam, O. (2022). OpenCV.org. Obtenido de https://opencv.org/about/: https://opencv.org/about/

Arroyo Menéndez, D. (2005). *Inteligencia Artificial (I) aplicaciones prácticas.* Mundo Linux: Sólo programadores Linux, 24-28.

Cheah Wai Zhao. (2015). *Exploring IOT Application Using Raspberry Pi. International Journal of Computer Networks and Applications*, 27 - 34.

Eben Upton, & Gareth Halfacree. (2014). *Raspberry Pi User Guide*. Great Britain: Google drive

García Serrano, A. (2012). *Inteligencia Artificial. Fundamentos, práctica y aplicaciones*. Madrid Vilena Artes Gráficas.

Jurado Lozada, M., & Chávez Fuentes, C. (2015). *Sistema de semaforización inteligente para el control de flujo vehicular mediante el procesamiento digital de imágenes*. Ambato: Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Carrera de Ingeniería en Electrónica y Comunicaciones.

Matt Richardson, & Shawn Wallace. (2013). *Getting Started with Raspberry Pi.* USA: O'REILLY .

Rauch - Hindin, W. (1989). *Aplicaciones de la inteligencia artificial en la actividad empresarial, la ciencia y la industria*. Madrid: Díaz de Santos, S.A.

Salcedo, O., Pedraza, L., & Hernández , C. (2006). *Modelo de Semaforización Inteligente para la ciudad de Bogotá*. Bogotá : Universidad Distrital Francisco José de Caldas.
bout/: https://opencv.org/about/

ANEXOS

Anexo 1: Código utilizado para el módulo de detección de vehículos

```python
import time
import cv2
import numpy as np
import imutils
import threading
import psycopg2

cap = cv2.VideoCapture('minuto.mp4')

fgbg = cv2.createBackgroundSubtractorMOG2()
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

def countdown():
    car_counter = 0
    inicio = time.time()
    cant_anterior = 0

    while True:
        ret, frame = cap.read()
        if ret == False:break
        frame = imutils.resize(frame, width= 640)

        #especificar area
        area_pts = np.array([[-15, 465], [frame.shape[1]+60, 465], [frame.shape[1]-280, 110],[260, 110]])

        #ayuda de imagen auxiliar
        imAux = np.zeros(shape=(frame.shape[:2]), dtype=np.uint8)
        imAux = cv2.drawContours(imAux, [area_pts], -1, (255), -1)
        image_area = cv2.bitwise_and(frame, frame, mask=imAux)
```

```python
        #Aplicamos sustracción de fondo

        fgmask= fgbg.apply(image_area)
        fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
        fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_CLOSE, kernel)
        fgmask = cv2.dilate(fgmask, None, iterations=5)

        #Encontramos los contornos
        cnts = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
        for cnt in cnts:
            if 17000 > cv2.contourArea(cnt) >6500:
                x, y, w, h = cv2.boundingRect(cnt)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255), 2)
                #print (cv2.contourArea(cnt))

                if 390 < (y+h) < 410:
                    car_counter = car_counter +1
                    cv2.line(frame, (30,400), (630,400), (0, 255,0), 3)

        #Visualización
        cv2.drawContours(frame, [area_pts], -1, (255, 0, 255), 2)
        cv2.line(frame, (30,400), (630,400),(0,255,255), 1)

        cv2.rectangle(frame, (frame.shape[1]-70, 215), (frame.shape[1]-5, 270), (0, 255, 0),2)
        cv2.putText(frame, str(car_counter), (frame.shape[1]-70, 250), cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 255, 0), 2)

        cv2.imshow('Modulo de Vision por Computador', frame)
        k = cv2.waitKey(30) & 0xFF
        if k == 27:
            print(car_counter)
            break
```

```python
        timer = int(time.time()-inicio)
        tiempoLuzVerde = int((querySelect()[0]*2)+5)
        if timer == (tiempoLuzVerde):
            cant_vehiculos = car_counter
            queryInsert(cant_vehiculos)
            car_counter = 0
            print(cant_vehiculos)
            inicio = time.time()
    cap.release()
    cv2.destroyAllWindows()
```

Anexo 2: Código en MATLAB, Algoritmo para cambio de luz del semáforo

```
function m = actualizar_m(n_v, n_r)
m_0 = 50;
c = n_v / (n_v + n_r+0.1);
m = m_0*(1-c);
```

```
clear; clc; close all
h = 1; % factor de conversión de unidades temporales
T = 60*h;
umbral = h;
delta_t = 0.2*h;
supert_0 = tic;
tiempos = [];
t_que_faltan = [];
t_en_verde = [];
n_cambios = 0;
i = 1;
while n_cambios < 4
    m = 1; tf = T; t_0 = tic; n = 1;
    while tf > umbral
        t = toc(t_0);
        if t > n*delta_t
            n_autos = [autos_via1, autos_via2];
            tf = tf - m*delta_t;
            tiempos = [tiempos; toc(supert_0)];
            t_que_faltan = [t_que_faltan; tf];
            n_v = n_autos(i);
            n_r = n_autos(j);
            fprintf("t=%f, Autos en verde=%d, En rojo=%d, m=%f\n", t, n_v, n_r, m)
            m = actualizar_m(n_v, n_r);
            n = n+1;
        end
    end
    disp("Cambiar a rojo")
    i = j;
    n_cambios = n_cambios + 1;
    t_en_verde = [t_en_verde; toc(t_0)];
end
```

Anexo 3: Código del módulo controlador del semáforo

```python
import random
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.OUT) ## GPIO 4 ROJO
GPIO.setup(17, GPIO.OUT) ## GPIO 17 AMARILLO
GPIO.setup(27, GPIO.OUT) ## GPIO 27 VERDE
GPIO.setup(22, GPIO.OUT) ## GPIO 22 ROJO
GPIO.setup(23, GPIO.OUT) ## GPIO 23 AMARILLO
GPIO.setup(24, GPIO.OUT) ## GPIO 24 VERDE

rojo = 4
amarillo = 17
verde = 27

rojo2 = 22
amarillo2 = 23
verde2 = 24

def actualizar_m(n_v, n_r):
    m_0 = 50
    c = n_v / (n_v + n_r+0.1)
    return m_0*(1-c)
def tiempo():
    h = 1 # factor de conversión de unidades temporales
    T = 60*h
    umbral = h
    delta_t = 0.2*h
    supert_0 = time.time()
    tiempos = []
    t_que_faltan = []
```

```python
    t_en_verde = []
    n_cambios = 0
    swich = 1 #quitar, solo sirve para elegir entre números randómicos
    i = 0
    while n_cambios<5:
        m = 1
        tf = T
        t_0 = time.time()
        n = 1
        swich = swich + 1
        j = i + (-1)**swich
        while tf > umbral:
            t = int(time.time()- t_0)
            if t > n*delta_t:
                n_autos = [random.randint(11,20), random.randint(0,9)]
                tf = tf - m*delta_t
                tiempos.append(int(time.time()- supert_0))
                t_que_faltan.append(tf)
                n_v = n_autos[i]
                n_r = n_autos[j]
                #print("t=%i, Autos en verde=%i, En rojo=%i, m=%f\n" %( t, n_v, n_r, m))

                if i == 0 and j == 1:
                    print ("Ejecución de la función de semaforo...")
                    GPIO.output(verde,True)
                    GPIO.output(rojo, False)
                    GPIO.output(amarillo, False)
                    print('VERDE en semáforo 1')
                    GPIO.output(rojo2, True) # En rojo el segundo semaforo
                    GPIO.output(amarillo2, False)
                    GPIO.output(verde2, False)
```

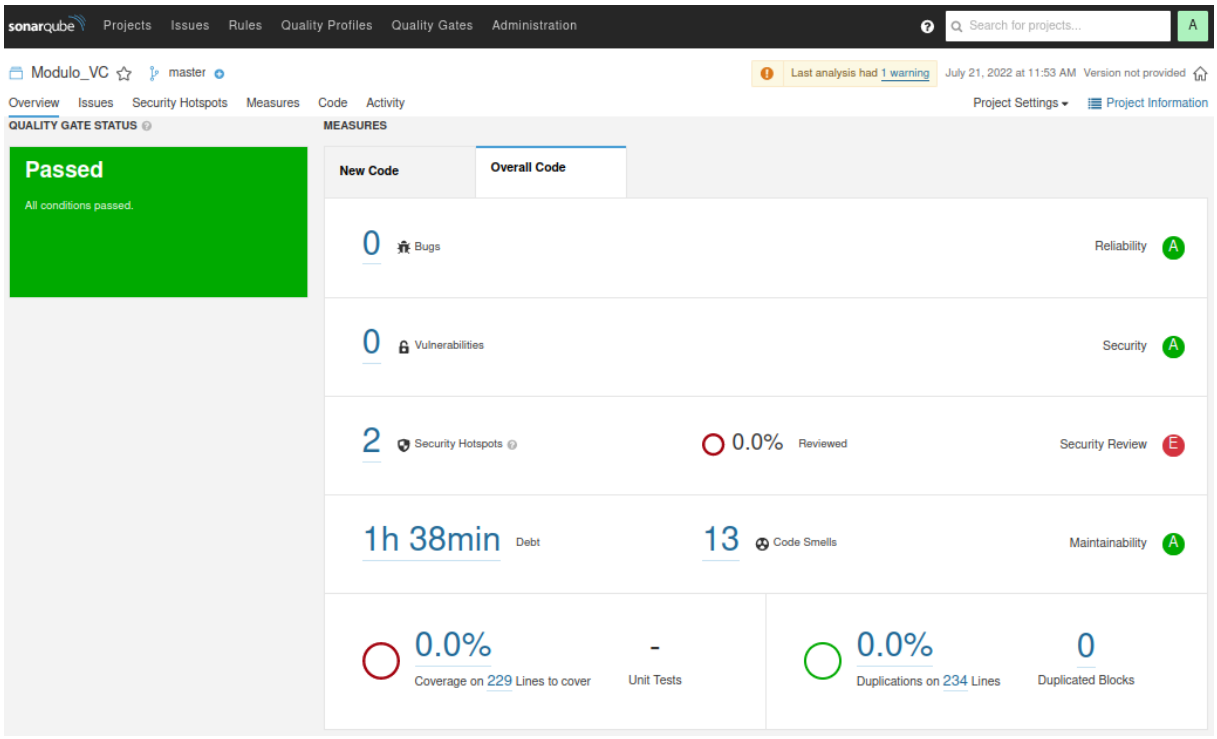
```
                else:
                    GPIO.output(amarillo2,False)
                    GPIO.output(verde2,True)
                    GPIO.output(rojo2, False)

                    GPIO.output(amarillo,False)
                    GPIO.output(verde,False)
                    GPIO.output(rojo, True)
                    print('VERDE en semaforo 2')
                m = actualizar_m(n_v, n_r)
                n = n+1
        if i == 0 and j == 1:
            GPIO.output(amarillo,True) # En naranja el primero y sigue encencido el rojo el segundo
            GPIO.output(verde, False)
            print('amarillo')
            time.sleep(3)
            GPIO.output(amarillo, False) # Se apaga el naranja
            GPIO.output(rojo, True)      # Se prende el rojo  del primero semaforo
            GPIO.output(rojo2, False)    # Se apaga el rojo del segundo semaforo y
            GPIO.output(verde2, True)    # Se prende el verde del segundo semaforo
            print('tiempo para que pasen los %i autos en rojo es: %f segundos' %(n_r, n_r * 0.3))
            print("Cambiar a rojo")
        else:
            GPIO.output(amarillo2,True) # En naranja el primero y sigue encencido el rojo el segundo
            GPIO.output(verde2, False)
        i = j
        n_cambios = n_cambios + 1

tiempo()
print("Limpiando la configuración de los GPIO")
GPIO.cleanup() ## Hago una limpieza de los GPIO
```

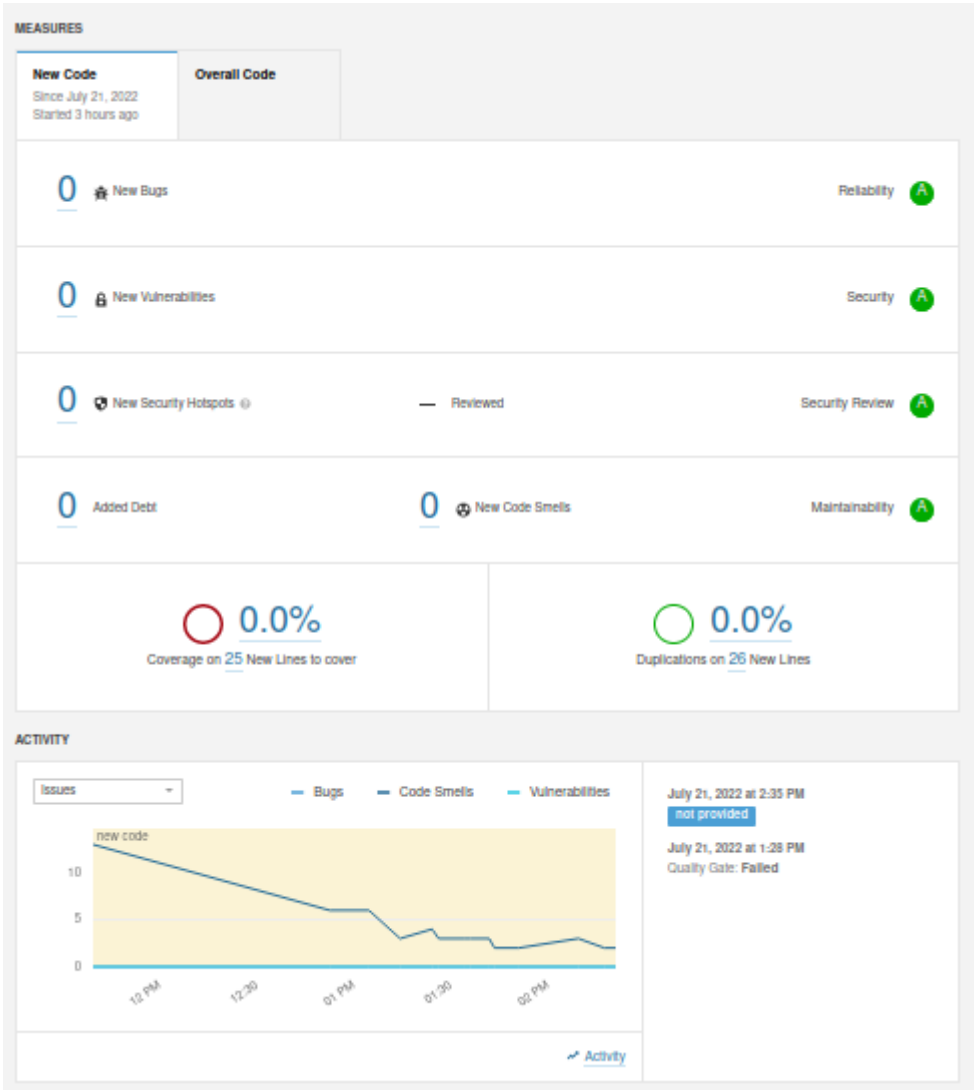
Anexo 4: Logs de ejecución de la herramienta Sonarqube

```
omar@omar-SVE14A27CLS:~/Documentos/docs/tesis/Modulos$ sonar-scanner   -Dsonar.projectKey=Modulo_VC   -Dsonar.sources=.   -Dsonar.host.url=http://127.
0.0.1:9000   -Dsonar.login=admin -Dsonar.password=sonarpwd
INFO: Scanner configuration file: /home/omar/Descargas/scanner/sonar-scanner-cli-4.7.0.2747-linux/sonar-scanner-4.7.0.2747-linux/conf/sonar-scanner.pr
operties
INFO: Project root configuration file: /home/omar/Documentos/docs/tesis/Modulos/sonar-project.properties
INFO: SonarScanner 4.7.0.2747
INFO: Java 11.0.14.1 Eclipse Adoptium (64-bit)
INFO: Linux 5.4.0-122-generic amd64
INFO: User cache: /home/omar/.sonar/cache
INFO: Scanner configuration file: /home/omar/Descargas/scanner/sonar-scanner-cli-4.7.0.2747-linux/sonar-scanner-4.7.0.2747-linux/conf/sonar-scanner.pr
operties
INFO: Project root configuration file: /home/omar/Documentos/docs/tesis/Modulos/sonar-project.properties
INFO: Analyzing on SonarQube server 9.0.1
INFO: Default locale: "es_ES", source code encoding: "UTF-8" (analysis is platform dependent)
INFO: Load global settings
INFO: Load global settings (done) | time=583ms
INFO: Server id: D0BA8F50-AYId7aCjkaJrT2R11BnI
INFO: User cache: /home/omar/.sonar/cache
INFO: Load/download plugins
INFO: Load plugins index
INFO: Load plugins index (done) | time=455ms
INFO: Load/download plugins (done) | time=1359ms
INFO: Process project properties
INFO: Process project properties (done) | time=13ms
INFO: Execute project builders
INFO: Execute project builders (done) | time=2ms
INFO: Project key: Modulo_VC
INFO: Base dir: /home/omar/Documentos/docs/tesis/Modulos
INFO: Working dir: /home/omar/Documentos/docs/tesis/Modulos/.scannerwork
INFO: Load project settings for component key: 'Modulo_VC'
INFO: Load project settings for component key: 'Modulo_VC' (done) | time=324ms
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=588ms
INFO: Load active rules
INFO: Load active rules (done) | time=7480ms
```

```
INFO: Indexing files...
INFO: Project configuration:
INFO: 9 files indexed
INFO: Quality profile for py: Sonar way
INFO: ------------ Run sensors on module Modulo_VC
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=486ms
INFO: Sensor Python Sensor [python]
INFO: Starting global symbols computation
INFO: 4 source files to be analyzed
INFO: Load project repositories
INFO: Load project repositories (done) | time=584ms
INFO: 4/4 source files have been analyzed
INFO: Starting rules execution
INFO: 4 source files to be analyzed
INFO: 4/4 source files have been analyzed
INFO: Sensor Python Sensor [python] (done) | time=10281ms
INFO: Sensor Cobertura Sensor for Python coverage [python]
INFO: Sensor Cobertura Sensor for Python coverage [python] (done) | time=17ms
INFO: Sensor PythonXUnitSensor [python]
INFO: Sensor PythonXUnitSensor [python] (done) | time=1ms
INFO: Sensor CSS Rules [cssfamily]
INFO: No CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
INFO: Sensor CSS Rules [cssfamily] (done) | time=1ms
INFO: Sensor JaCoCo XML Report Importer [jacoco]
INFO: 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations: target/site/jacoco/jacoco.xml,target/site/jacoco-it/jacoco.xml,b
uild/reports/jacoco/test/jacocoTestReport.xml
INFO: No report imported, no coverage information will be imported by JaCoCo XML Report Importer
INFO: Sensor JaCoCo XML Report Importer [jacoco] (done) | time=4ms
INFO: Sensor C# Project Type Information [csharp]
INFO: Sensor C# Project Type Information [csharp] (done) | time=1ms
INFO: Sensor C# Properties [csharp]
INFO: Sensor C# Properties [csharp] (done) | time=1ms
INFO: Sensor JavaXmlSensor [java]
INFO: Sensor JavaXmlSensor [java] (done) | time=2ms
INFO: Sensor HTML [web]
INFO: Sensor HTML [web] (done) | time=5ms
INFO: Sensor VB.NET Project Type Information [vbnet]
```

```
INFO: Sensor VB.NET Properties [vbnet] (done) | time=1ms
INFO: ------------ Run sensors on project
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=31ms
INFO: SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
INFO: CPD Executor Calculating CPD for 4 files
INFO: CPD Executor CPD calculation finished (done) | time=16ms
INFO: Analysis report generated in 114ms, dir size=123,3 kB
INFO: Analysis report compressed in 57ms, zip size=26,4 kB
INFO: Analysis report uploaded in 528ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://127.0.0.1:9000/dashboard?id=Modulo_VC
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://127.0.0.1:9000/api/ce/task?id=AYIhrku-fXrLYN-Askmi
INFO: Analysis total time: 25.794 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 29.393s
INFO: Final Memory: 7M/34M
INFO: ------------------------------------------------------------------------
```




MEASURES
New Code
Since July 21, 2022
Started 3 hours ago
Overall Code
0
New Bugs
Reliability
A
0
New Vulnerabilities
Security
A
0
New Security Hotspots
Reviewed
Security Review
A
0
Added Debt
0
New Code Smells
Maintainability
A
0.0%
Coverage on 25 New Lines to cover
0.0%
Duplications on 26 New Lines
ACTIVITY
Issues
Bugs
Code Smells
Vulnerabilities
new code
10
5
0
12 PM
12:30
01 PM
01:30
02 PM
Activity
July 21, 2022 at 2:35 PM
not provided
July 21, 2022 at 1:28 PM
Quality Gate: Failed