UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN SISTEMAS

“Diseño y construcción de un prototipo para cobro de peajes con Visión Artificial”

Tesis previa a la Obtención del título de Ingeniero en Sistemas

Autor:

Ramiro Vladimir Vaca Moscoso.

Director:

Ing. Mario Andrés Palma Jaramillo, Mg. Sc

Loja – Ecuador

2018

CERTIFICACIÓN

Ing. Mario Andrés Palma Jaramillo, Mg. Sc

DOCENTE DE LA CARRERA DE INGENIERÍA EN SISTEMAS, DEL ÁREA DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES DE LA UNIVERSIDAD NACIONAL DE LOJA.

CERTIFICA:

Que el Sr. Ramiro Vladimir Vaca Moscoso, egresado de la carrera de Ingeniería en Sistemas y cuyo tema versa sobre: **"DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA COBRO DE PEAJES CON VISIÓN ARTIFICIAL",** ha sido monitoreado, revisado y orientado bajo mi asesoramiento, con pertinencia y con la rigurosidad científica que el trabajo de investigación debe cumplir, por lo cual autorizo su presentación y sustentación.

Loja, 30 de Marzo de 2018

Ing. Mario Andrés Palma Jaramillo Mg. Sc

Director del Proyecto de Titulación.

AUTORÍA

Yo, **RAMIRO VLADIMIR VACA MOSCOSO,** declaro ser autor del presente trabajo de tesis y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales, por el contenido de la misma.

Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional-Biblioteca Virtual.

Firma: …….

Cédula: 1718562919

Fecha: Loja, 30 de Marzo de 2018

CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DEL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

Yo **RAMIRO VLADIMIR VACA MOSCOSO**, declaro ser autor de la tesis titulada: **"DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA COBRO DE PEAJES CON VISIÓN ARTIFICIAL",** como requisito para optar al grado de**: INGENIERO EN SISTEMAS;** autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la Ciudad de Loja, a los veintiséis días del mes de julio del dos mil dieciocho.

Firma:…….

Autor: Ramiro Vladimir Vaca Moscoso

Cédula: 1718562919

Dirección: Loja (Dolores Cacuango A'29 y Transito Amaguaña)

Correo Electrónico: ramvlay@gmail.com

Teléfono: 072326431. **Celular:** 0994750077

DATOS COMPLEMENTARIOS:

Director de Tesis: Ing. Mario Andrés Palma Jaramillo, Mg. Sc

Tribunal de Grado: Ing. Edison Leonardo Coronel Romero, Mg. Sc

Ing. Oscar Miguel Cumbicus Pineda, Mg. Sc

Ing. Jose Luis Granda Sivisaca, Mg. Sc

DEDICATORIA

A mis Padres.

Por ser siempre mí fuerza, por su apoyo incondicional, por su guía y consejos siempre acertados, sus valores y motivación constante que me han permitido ser una persona correcta y de bien, pero sobre todo por su amor que siempre me acogió.

A mis hermanos.

Por alegrar mis días y ser siempre mis más fieles compañeros y amigos en la vida, por ser las estrellas que me han iluminado y me iluminan en este camino.

A mi novia.

Por su constante apoyo y comprensión, por su motivación y deseos siempre de superación. Por darme el regalo más grande de la vida, por hacerme tan feliz con un angelito que viene en camino.

EL AUTOR.

AGRADECIMIENTO

En primer lugar deseo agradecer a la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, por haberme permitido formarme en sus aulas como profesional. A todos los docentes que de una u otra manera supieron transmitir y enseñarme sus conocimientos.

Agradezco también a mis padres por su apoyo, porque sin ellos esto no sería posible, por estar a cada paso siempre a mi lado y porque este logro más que mío es de ellos.

Y de manera especial deseo agradecer al Ing. Mario Andrés Palma Jaramillo por brindarme su valioso tiempo y experiencia profesional, para que este trabajo sea realizado y finalizado de la mejor manera.

EL AUTOR.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. TÍTULO.

"DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA COBRO DE PEAJES CON VISIÓN ARTIFICIAL".

2. RESUMEN.

El presente trabajo de titulación tiene lugar en los espacios físicos destinados para efectuar el cobro por derecho para transitar por una vía; es decir, este proyecto software propone un prototipo para el cobro de peajes. La modalidad de negocio para el cobro de peajes en nuestro país son: a través de un sistema manual que se basa en el ojo humano para según la clasificación vehicular, imponer el rubro por concepto de peaje; y un sistema llamado Telepeaje que a través de un chip inteligente “TAG” ubicado en el parabrisas del vehículo, cuenta con saldo electrónico para descontarse cada vez que se pase por los radares de una estación de peaje, siendo así que este sistema se aproxima a una automatización que resulta costosa su implementación y poco confiable ya que los usuarios transfieren el chip y muchas veces circulan por las estaciones de peaje, vehículos sin saldo en sus chips.

El prototipo de peaje que propone este trabajo de titulación es usando visión artificial, esta propuesta es económica y demanda de trabajo en el software y poco en el hardware. A través de la visión artificial se logra un sistema basado en adquirir, procesar, analizar y comprender imágenes naturales (imágenes tomadas en escenas de la vida cotidiana) con el fin de producir información para ser tratados por un computador, en este caso, para a partir de una placa vehicular obtener su clasificación e imponer el cobro automático por concepto de peaje. Utilizando así, solo una cámara, un ordenador y una placa. El presente prototipo yace en una maqueta de una estación de peaje en general, el software de escritorio que modeló la forma de trabajo de las estaciones de peaje; y adicionalmente, un software web para la consulta por pagos de peajes y reportes para administradores de estación.

El sistema fue desarrollado bajo el patrón de diseño Modelo Vista Plantilla “MVT” y los métodos: científico, deductivo, inductivo y metodología de desarrollo de software “ICONIX”. En el proceso de desarrollo del sistema se consideró el ciclo de vida de software incremental, según como lo indica ICONIX. Se inició con la obtención de los requisitos para determinar la funcionalidad del sistema, a través de: la revisión documental, prototipado, diagrama de casos de uso y modelo de dominio. Se continuó con la especificación de casos de uso, para en la siguiente fase de diseño realizar los diagramas de secuencia y de clases. Finalmente, ICONIX culmina con la fase de codificación: arquitectura del sistema y generación de código.

Para concluir este proyecto de titulación y dar cumplimiento a sus objetivos, las pruebas de funcionalidad fueron satisfactorias, reflejando un nivel de confianza del 98% para la detección y reconocimiento de placas vehiculares a través del prototipo propuesto con entorno controlado.

2.1. SUMMARY.

The present degree work takes place in the physical spaces destined to effect the collection by right to transit by a road; that is, this software project proposes a prototype for the collection of tolls. The business modality for the collection of tolls in our country are: through a manual system that is based on the human eye according to the vehicle classification, to impose the item for the concept of toll; and a system called Telepeaje that through an intelligent chip "TAG" located in the windshield of the vehicle, has electronic balance to be deducted each time it passes through the radars of a toll station, being that this system approaches an automation that is costly to implement and unreliable since users transfer the chip and often circulate through the toll stations, vehicles without balance in their chips.

The prototype of toll that this degree work proposes is using artificial vision, this proposal is economic and demand of work in the software and little in the hardware. Through artificial vision a system is achieved based on acquiring, processing, analyzing and understanding natural images (images taken in scenes of daily life) in order to produce information to be treated by a computer, in this case, to Starting from a vehicular license plate, obtain its classification and impose automatic charging for tolls. Using this way, only one camera, one computer and one plate. The present prototype lies in a model of a toll station in general, the desktop software that modeled the way of working of the toll stations and additionally, a web software for consultation for payment of tolls and reports for station managers.

The system was developed under the design pattern Model Vista Template "MVT" and the methods: scientific, deductive, inductive and software development methodology "ICONIX". In the process of system development, the incremental software life cycle was considered, as indicated by ICONIX. It started with obtaining the requirements to determine the functionality of the system, through: documentary review, prototyping, use case diagram and domain model. We continued with the specification of use cases, and in the next design phase we made the sequence and class diagrams. Finally, ICONIX ends with the coding phase: system architecture and code generation.

To complete this titling project and meet its objectives, the functionality tests were satisfactory, reflecting a level of confidence of 98% for the detection and recognition of vehicle plates through the proposed prototype with controlled environment.

3. INTRODUCCIÓN.

Dentro de los campos de la inteligencia artificial está la visión artificial, cuyo propósito es programar un computador para que éste entienda la imagen de una escena natural (simulando al ojo humano), mediante sistemas de adquisición y dispositivos de cómputo. El propósito de éste es la detección y reconocimiento del objeto placa vehicular en imágenes, mediante el reconocimiento de patrones y procesamiento de imágenes.

Se aplica la visión artificial a la automatización de cobros de peaje, ya que el modelo de negocio actual no es óptimo, en el caso del sistema manual (el más utilizado en el país), se coligió que: **"El sistema de cobro de peajes en las carreteras del país no es el adecuado, generando congestionamiento vehicular e interrupción del tráfico vehicular".** Mientras que, en el caso de la automatización implementada en Quito, no es favorable porque los radares son costosos y el chip inteligente que compran los conductores, es transferible y muchas veces en las estaciones de peaje, circulan vehículos con saldo insuficiente para descontarse del chip. La constitución no considera como una infracción la transferencia del chip, ni la circulación sin saldo; mientras que, sí es delito la adulteración de placas y hay más regulación a las mismas. De modo que el prototipo propuesto es económico, ya que no requiere la adquisición de hardware para el conductor, sino tan solo con la placa se hará el cobro por peaje, siendo este sistema más controlado para las empresas concesionadas y para los conductores.

Esta solución todavía no ha sido implementada en ningún peaje del territorio nacional; en su primera versión efectúa el cobro de peaje y hace que los cobros sean transparentes al usuario a través de un sistema web, donde cualquier usuario con su número de placa puede consultar su historial de pagos por peaje; además, a los administradores de las estaciones se les permite el acceso a información estadística por flujo vehicular e ingresos por recaudaciones de peaje. El prototipo construido realiza la detección y reconocimiento de placas. En la parte de detección se utiliza a la librería OpenCV que: captura la imagen de la escena natural, obtiene la región de interés (ROI), mediante el clasificador HAAR CASCADE detecta la placa, con el método basado en ER (Extremal regions) del algoritmo basado en componente conexas se realiza la detección de regiones estables, se realiza el filtrado y agrupación utilizando el método de Neumann; finalmente se realiza la detección de bordes con el umbral adaptativo. En la siguiente parte referente al reconocimiento de placas se utiliza la librería Tesseract que: obtiene una imagen procesada, la binariza para analizar la estructura de la placa, realiza la segmentación para finalmente realizar el reconocimiento de caracteres.

Todo este trabajo y carga de procesamiento es transferida al ordenador, donde con mejoras de la programación se logra el reconocimiento óptico de caracteres de la placa, dejando así que el acondicionamiento del entorno de peaje se enfoque a una cámara de alta resolución específica para sistemas de visión artificial, correcta iluminación y un ordenador en la cabina de peaje.

Este prototipo de peaje tiene un software de escritorio programado en el lenguaje C++ que será ejecutado en cada ordenador de las diferentes cabinas de peaje que tenga la estación, las personas encargadas de su utilización son los usuarios: administrador y operador; básicamente este software automatiza todo el modelo de negocio de los peajes. Mientras que, también se desarrolló un software web programado en el lenguaje Python, que es accesible a través de la IP pública: http://167.99.6.160/, para los usuarios administrador y cualquier persona; en este sistema se proporciona información de consulta y una sección privada con información estadística para cada estación de peaje.

La estructura de esta memoria se enmarca en los lineamientos establecidos por la Universidad Nacional de Loja y el Área de la Energía, las Industrias y los Recursos Naturales No Renovables; como sigue a continuación: RESUMEN contiene un extracto del contenido general del Trabajo de Titulación; ÍNDICE describe la ubicación de los temas tratados con sus respectivas figuras y tablas, indicando el número de página a la que pertenece; INTRODUCCIÓN relata lo relevante que es el tema y su aplicabilidad en la investigación científica; REVISIÓN DE LITERATURA comprende la sustentación teórica de las temáticas que ayudan a la comprensión del Trabajo de Titulación; METODOLOGÍA realiza una descripción de los principales materiales empleados y métodos de investigación tanto científicos, experimentales y técnicas aplicadas que dieron solución al proyecto; RESULTADOS tiene como propósito la evaluación y el cumplimiento de los objetivos planteados; DISCUSIÓN se explica el uso de los métodos y técnicas utilizadas; CONCLUSIONES describe las ideas a las que se llegó; el proyecto finaliza con las RECOMENDACIONES, BIBLIOGRAFÍA y ANEXOS.

4. REVISIÓN DE LITERATURA.

4.1. Peaje.

Peaje es el pago que se efectúa como derecho para poder circular por un camino, es una tarifa que se cobra a un medio de transporte terrestre, fluvial o marítimo como derecho de tránsito para utilizar la infraestructura de la respectiva vía de comunicación; por ejemplo a los automóviles para poder circular por una autopista o carretera, o a los barcos para poder atravesar por un canal de navegación o una hidrovía. Además, el peaje ahorra a los usuarios tiempo de viaje y reduce costos de operación, con respecto al tránsito por vías o rutas alternas libres de peaje. [1]

El dinero recaudado a través de un peaje se destina normalmente a financiar la construcción, operación y mantenimiento de infraestructuras viarias (carreteras, túneles, canales de navegación o puentes). En el caso de carreteras sujetas a concesión o tercerizadas, el peaje permite al operador privado recuperar las inversiones realizadas y los costos futuros de administración, operación y mantenimiento. [2]

4.1.1. Tipos de peaje.

Hay varios tipos de peaje como se indican a continuación: [1]

- **Peaje abierto:** Cada cierta distancia hay una caseta de peaje donde se paga.
- **Peaje cerrado:** Al entrar en la carretera de peaje, se registra la entrada y se abona a la salida, según la longitud recorrida, sin más paradas intermedias.
- **Peaje anual:** En países como Suiza, los usuarios pagan anualmente una cantidad para circular por todas las autopistas libremente (se acredita mediante una pegatina en el parabrisas). Los turistas, pagan también la misma cantidad.
- **Peaje urbano de congestión:** Es un tipo de impuesto que se cobra en algunas ciudades como: Buenos Aires, Estocolmo, Londres, Milán y Singapur, bajo la política de tarifas de congestión. Con el propósito de: disminuir la cantidad de vehículos que acceden a una determinada zona del centro, reducir la congestión de tránsito y disminuir las emisiones de gases de efecto invernadero.
- **Peaje sombra:** En este tipo de peaje una empresa financia la construcción y mantenimiento de una carretera a cambio de una concesión de explotación por un largo periodo de tiempo, sin embargo el pago de dicha concesión está a cargo del presupuesto del Estado y no de los usuarios, estos pagos se realizan anualmente en función del tráfico vehicular.

4.1.2. Estación de cobro de peaje.

Es la infraestructura y equipos destinados a realizar las actividades relativas exclusivamente al cobro de peaje establecido mediante la normativa vigente.

La estación de peaje tiene usualmente la siguiente infraestructura mínima y necesaria: Estación de Peaje, Cubierta, Accesos a la estación de peaje, Área de Operación, Área de Administración y Servicios, Isletas de Protección, Cabinas de Peaje, Equipos de Operación, Señalización Horizontal y Vertical; y Nota de Venta (documento de pago).

4.1.3. Peaje en ecuador.

Durante el Gobierno de Sixto Durán Ballén se creó la figura legal de la concesión (permiso) para entregar a la empresa privada ciertos tramos de carretera que requerían de mantenimiento rutinario. Para retribuir al concesionario las inversiones por gastos administrativos, operativos y prestación de servicios, el Estado, a través del Ministerio de Transporte y Obras Públicas (MTOP), autorizó el cobro del peaje. [3]

4.1.3.1. Concesiones autorizados por el MTOP.

A continuación están las empresas autorizadas por el Ministerio de Transporte y Obras Públicas para el cobro de peajes a lo largo de todo el territorio ecuatoriano: [4]

- **PANAVIAL S.A.** está a cargo de la Troncal de la Panamericana en la Sierra que va desde Rumichaca a Riobamba, aproximadamente de 570 Kilómetros.
- **CONORTE S.A** se encarga de 266.20 Kilómetros en la región costa.
- **CONCEGUA S.A.** tiene 248.10 Kilómetros en la región amazónica.
- **CONVIAL S.A** (desde el 2017) está a cargo de 12 vías y tiene 3 estaciones.
- **LAS PREFECTURAS** por generalizar así, se encargan de otros peajes que suman 20 peajes (dichos consejos provinciales se entienden implícitamente): Pichincha, Azuay, El Oro, Los Ríos y Loja.

4.1.3.2. Tarifas de los peajes.

Cada proyecto tiene su propio análisis económico y el MTOP, como ente regulador de los valores del peaje, toma en cuenta la realidad social para poder implementarlos.

A modo de ejemplo, se exponen algunas tarifas vigentes del año 2017 de cada una de las empresas antes mencionadas:

Tabla 1 Tarifa de la concesionaria PANAVIAL S.A en el Peaje Oyacoto [5]

Automóviles, todoterrenos o camionetas	$ 1,00
Buses y camiones de 2 ejes	$ 2,00
Buses y camiones de 3 ejes	$ 3,00
Camiones de 4 ejes	$ 4,00
Camiones de 5 ejes	$ 5,00
Camiones de 6 ejes o más	$ 6,00
Motos	$ 0,20
Eje remolque en livianos	$ 0,50

Tabla 2 Tarifa de la concesionaria CONORTE S.A en el Peaje Samborondon [6]

Liviano (Automóviles, todoterrenos o camionetas)	$ 0,50
Pesado (Buses y camiones de 2 ejes)	$ 1,00
Extra Pesado 1 (Buses y camiones de 3 ejes)	$ 1,50
Extra Pesado 2 (Camiones de 4 ejes)	$ 2,00
Extra Pesado 3 (Camiones de 5 ejes)	$ 2,50
Extra Pesado 4 (Camiones de 6 ejes o más)	$ 3,00

Tabla 3 Tarifa de la concesionaria CONCEGUA S.A en el Peaje Milagro [7]

Liviano (Automóviles, todoterrenos o camionetas)	$ 1,00
Pesado (Buses y camiones de 2 ejes)	$ 2,00
Extra Pesado 1 (Buses y camiones de 3 ejes)	$ 3,00
Extra Pesado 2 (Camiones de 4 ejes)	$ 4,00
Extra Pesado 3 (Camiones de 5 ejes)	$ 5,00
Extra Pesado 4 (Camiones de 6 ejes o más)	$ 6,00

No se tienen tarifas oficiales de la empresa CONVIAL que está a cargo de la vialidad centro-sur manabita, ya que esta empresa es de reciente constitución y las tarifas que se pretenden cobrar no han sido aprobadas por el Ministerio de Transporte y Obras Públicas ante el rechazo de la ciudadanía a las tentativas tarifas.

4.1.3.3. Funcionamiento del Peaje.

Dos son las formas de pago por concepto del peaje: el pago manual que se hace en la cabina de peaje y el telepeaje. [8]

El pago manual se hace en todos los peajes del país, este sistema se caracteriza por disponer dentro de la estación de peaje de una caseta de cobro en la cual un operador

realiza el cobro en forma manual a cada uno de los automóviles o variantes que circulan. La forma de pago automático (telepeaje) está en correcto funcionamiento en la ciudad de Quito, siendo así que en la avenida Interoceánica Oswaldo Guayasamín se utiliza el Telepeaje y en la autopista Rumiñahui y vía Intervalles están los usuarios del Peaje Exprés. La idea de utilizar esta forma de pago en la ciudad de Quito, es una forma de incentivar el uso de estos dispositivos para agilizar el paso y reducir la congestión; ya que según la Unidad de Gestión de Peajes, explica que de las 70 000 pasadas que se registran diariamente en el peaje de la autopista Rumiñahui, el 60% pertenece al pago manual en comparación con el 40% que tiene el sistema automático. [9]

4.1.4. Telepeaje en Ecuador.

Figura 1 Telepeaje del túnel Oswaldo Guayasamín [10]

El Municipio de Quito tiene a disposición de los usuarios un nuevo TAG - TelePass (Dispositivo electrónico que contabiliza el número de pasadas por el Telepeaje – estación donde se cobra el peaje); que se puede usar en todos los peajes de la provincia e incluso en: Panavial en la Panamericana Norte, Concegua y Conorte. La tecnología 5.8 GHZ de los nuevos dispositivos TAG que se comercializan para el paso por el Telepeaje de la Vía Interoceánica es la versión que reemplaza a su predecesora (versión 2.4 GHZ). Este dispositivo cuenta con un chip inteligente que puede ser identificado fácilmente por radares una vez que el TAG se encuentre colocado en el parabrisas del vehículo. [11]

El TAG se puede conseguir en cualquiera de las oficinas de Servicio al Cliente de la Empresa Pública Metropolitana de Movilidad y Obras Públicas “EPMMOP”, para lo cual se deben cumplir algunos requisitos que se exponen en Adipiscor [12]

Los principales beneficios del Telepeaje son: la reducción del tiempo de circulación en la Av. Interoceánica, disminución del lapso de espera para el cobro del peaje. En una fila de 200 vehículos disminuye entre 10 y 15 minutos a comparación del cobro manual. Además, hay un 25% de descuento a los usuarios que usan Telepeaje [13]

El costo del TAG y de la primera carga es de $30.00 y la recarga mínima es de $28.50, la cual equivale a aproximadamente 100 pasadas y no tiene fecha de caducidad. El pago puede realizarse en las oficinas del peaje o mediante débito o crédito bancario. En este sentido, los usuarios llenarán en el formulario la autorización para que la EPMMOP realice el débito o a su vez, podrán realizar recargas en los centros de atención al cliente, farmacias Fybeca y www.telepeaje.ec. Para determinar el saldo del TAG, el color de las luces al momento de pasar por el peaje lo indican. La luz verde significa paso normal, el destello simultáneo de las luces roja y verde anuncia que se está acabando el saldo (menos de cinco dólares) y la luz roja indica que no hay saldo. En este caso, el usuario es considerado como un infractor y tendrá que pagar una multa de $4.00 la primera vez y $10.00 por reincidencia. La Policía cobrará las multas en la matriculación. [14]

4.1.5. Placas en ecuador.

Las placas en Ecuador tienen 154 mm de alto y 404 mm de ancho y son reflectivas con el fin de mejorar su visibilidad. Están formadas por tres letras y tres o cuatro dígitos, partiendo desde 000 al 9999, en formatos de ABC-123 y ABC-1234, con el nombre del país en mayúsculas en la parte superior. La primera letra indica la provincia en la que se matriculó por primera vez el vehículo, la segunda identifica el tipo de la placa y la tercera es correlativa. Ej: ABC-123, ABC-1234, HG-12L (para motos). Dependiendo del tipo de vehículo, las placas tienen colores diferentes. Para los vehículos no particulares, las nuevas placas conservarán el mismo color diferenciador pero ya no será aplicado en la totalidad de la placa, sino solo en el borde superior, siendo el resto de la placa de color blanco. Este cambio se realizó de manera que se mejore la visibilidad de las placas, en particular ante las cámaras y radares. [15]

Figura 2 Estándar de placas en Ecuador [15]

4.2. Reconocimiento de placas vehiculares.

Este proyecto se enfoca en el desarrollo de un sistema de detección y extracción de placas vehiculares en la zona de peajes, basado en visión artificial. La clasificación de este tipo de proyecto es un sistema de reconocimiento automático de matrículas también llamado sistema ANPR (Automatic Number Plate Recognition), el cual es un método que permite la vigilancia en masa a través del uso de un software de reconocimiento óptico de caracteres para escanear imágenes y leer las placas de los vehículos. [16]

Existen varios sistemas ANPR comerciales y cada uno utiliza diferentes tecnologías, algunas costosas en la adquisición del hardware indispensable y otras, como esta que pretende realizar mejoras en el software basados en visión artificial.

4.2.1. Técnicas de tratamiento digital de imágenes.

Las imágenes digitales a color de la actualidad están formadas a partir de 3 matrices de píxeles, cuya intensidad de píxel variará entre 0 y 1 o entre 0 y 255 dependiendo del formato. Estas tres matrices representan la intensidad del color rojo, verde y azul (RGB) respectivamente. Por ejemplo, en la matriz que representa el color rojo en la imagen y suponiendo un formato en el que la intensidad varía 0 y 255, un pixel con valor 255 en dicha matriz significará la presencia máxima del rojo en dicho punto, mientras que un píxel con valor 0 significará la ausencia de rojo. Si representamos cada una de estas matrices (o canales) individualmente podremos apreciar que cada una de ellas es una imagen de grises, y al combinar las 3 obtenemos una imagen a color. El objetivo de esta parte del algoritmo es obtener la matriz de grises que representará la imagen de entrada utilizando los tres canales explicados durante este párrafo. [17]

Solo se trabajó con imagen de grises ya que es monocanal, lo cual hizo que las operaciones posteriores sean 3 veces más rápidas, al procesar un canal en lugar de 3. Para obtener dicha imagen de grises, la librería de tratamiento de imagen utilizada (OpenCV) incorpora una función con la que realizar este proceso muy fácilmente.

4.2.1.1. Binarización.

La binarización de imágenes consiste en convertir una imagen de nivel de gris (4 bits/pixel u 8 bits/pixel) a una imagen de blanco y negro. También la binarización se entiende como el comparar los niveles de grises presentes en la imagen con un valor (umbral) predeterminado. Este proceso se realiza básicamente para separar objetos de interés desde de fondo y simplificar operaciones principales que siguen, tales como

reconocimiento o clasificación de objetos de interés. Para realizar la binarización, se necesita obtener un valor de pixel adecuado, el cual se llama valor umbral, y la operación de selección de este valor se llama umbralización (“Tresholding”). [18]

4.2.1.2. Umbralización.

Este concepto va de la mano con el anterior. La umbralización es uno de los métodos más simples y eficientes de segmentación, es decir, separar o extraer las regiones de una imagen que nos interesa estudiar o analizar, para lograr separar la región deseada se establece un valor que define el umbral, los pixeles cuya intensidad superen el umbral serán rechazados o aceptados, según sea el caso. [19]

Si el valor del píxel es mayor que un valor umbral, se le asigna un valor (puede ser blanco), de lo contrario se le asigna otro valor (puede ser negro). La función utilizada en OpenCV es cv.threshold. El primer argumento es la imagen de origen en escala de grises. El segundo argumento es el valor de umbral que se utiliza para clasificar los valores de píxel. El tercer argumento es maxVal, que representa el valor que se dará si el valor del píxel es más o menos el valor del umbral. OpenCV proporciona diferentes estilos de umbralización y se decide por el cuarto parámetro de la función. [20]

Umbral adaptativo

Puede no ser bueno tener un valor fijo para el umbral en todas las condiciones donde la imagen tiene diferentes condiciones de iluminación en diferentes áreas. En ese caso, se optó por umbrales adaptativos. El algoritmo calcula el umbral para una pequeña región de la imagen, así se obtuvo diferentes umbrales para diferentes regiones de la misma imagen que dio mejores resultados para las imágenes con iluminación variable.

Tiene tres parámetros de entrada 'especiales' y solo un argumento de salida. En resumen, el método adaptativo decide cómo se calcula el valor umbral. [20]

4.2.1.3. Obtención de los bordes.

Una vez obtenida la imagen binarizada, el siguiente paso que realiza el algoritmo creado es la detección de los bordes de la imagen, que serán las zonas en la que la imagen cambia bruscamente en pixeles cercanos, dichas zonas son conocidas también como altas frecuencias. Para la detección de bordes el algoritmo lo ha hecho implícitamente en la umbralización adaptativa con el método Gaussiano.

Estas características son conseguidas gracias a que es un algoritmo basado en la reducción de ruido, el cálculo del gradiente, la eliminación de pixeles no considerados parte de un borde y por último la histéresis. A continuación, se van a explicar en detalle dichos métodos. Es inevitable que las imágenes tomadas por una cámara tengan cierta cantidad de ruido, para prevenir que dicho ruido sea confundido con bordes debe de ser reducido en la medida de lo posible. El proceso que se encargará de reducir dicho ruido será el emborronado de la imagen mediante la aplicación de un filtro Gaussiano (también conocido como filtro paso bajo en tratamiento de imagen). Mediante la convolución de la imagen de grises obtenida en la sección anterior del algoritmo, obtendremos una imagen binarizada más borrosa, pero con menos ruido. El algoritmo encuentra bordes donde la intensidad en la imagen de grises sufre cambios más bruscos, dichos cambios son localizados mediante el cálculo de los gradientes de la imagen.

Una vez los bordes han sido clasificados como “fuertes” y “débiles”, el último paso a realizar para completar la ejecución del algoritmo es el seguimiento de bordes por histéresis. Que consiste en la eliminación de todos los bordes “débiles” salvo los que estén conectados a bordes “fuertes”, debido a que hay una alta probabilidad que los demás bordes “débiles” hayan sido producidos por el ruido existente en la imagen original. El seguimiento de bordes puede ser implementado mediante el “BLOB-analysis” (Binary Large OBject). Los píxeles de la imagen obtenida después de la aplicación de la doble umbralización son segmentados en BLOBs conexos, utilizando una clasificación de componentes conexas con vecindad 8 (Esto significa que se analizan como posibles componentes conexas a cada uno de los 8 píxeles adyacentes a cada píxel). Todos los BLOBs que contengan al menos un borde “fuerte” serán conservados, mientras que todos los demás serán suprimidos. [17]

4.2.2. Algoritmos de localización y reconocimiento de texto.

La localización y reconocimiento de texto en imágenes naturales, es decir, imágenes tomadas en escenas de la vida cotidiana ha sido un campo en el que se ha investigado mucho en la última década. A diferencia del reconocimiento de texto en documentos, tarea que se ha resuelto con éxito gracias a los algoritmos de estado del arte de OCR. La mayoría de los métodos de reconocimiento y localización de texto publicados están basados en un procesado secuencial que consta de 3 etapas: localización de

candidatos, segmentación del texto y por último reconocimiento de texto mediante sistemas OCR para documentos impresos. Según los algoritmos de localización de texto que implementen dichos métodos pueden ser clasificados en 2 grupos: métodos basados en ventana deslizante, métodos basados en componentes conexas.

En este proyecto de trabajo de titulación, se ha utilizado uno de los dos métodos basados en componentes conexas; ya que el método de ventana deslizante se basan en pasar una ventana deslizante a lo largo de toda la imagen para encontrar posibles candidatos a texto, para posteriormente utilizar técnicas de machine learning que identifican textos. Este tipo de métodos suelen ser lentos debido a que la imagen debe ser procesada en múltiples escalas y al ser este proyecto un prototipo de peaje, se requiere que el reconocimiento se haga rápidamente.

A esta razón se ha utilizado dentro de los métodos basados en componentes conexas el método basado en ER (Extremal regions). El cual obtienen candidatos a letras analizando características comunes en píxeles vecinos, para posteriormente agrupar dichos candidatos en texto, realizando comprobaciones para evitar falsos positivos.

Esta subsección abarca: la captura de la imagen que contiene el texto a reconocer (placa) y la forma en que se lee dicha imagen para obtener en texto su reconocimiento. El siguiente mapa mental resume la localización y reconocimiento de placas que realiza este prototipo de peaje. [17]

Figura 3 Detección y reconocimiento de texto en este prototipo de peaje

4.2.2.1. Método basado en Regiones ER (Extremal Region).

El método basado en regiones ER que se va a analizar fue propuesto por Jiri Matas y Lukas Neumann, en junio de 2012. Este método es superior al MSER (también basado en componentes conexas) debido a que utiliza menos memoria para su ejecución, muestra mejores resultados en imágenes borrosas, es resistente a la falta de nitidez, variación de la iluminación, el color, la textura y maneja bajo contraste de texto.

Además, esta técnica difiere de MSER en que la selección de ER adecuados se realiza mediante un clasificador secuencial capacitado para la detección de caracteres, es decir, eliminando el requisito de estabilidad de MSER y seleccionando regiones específicas de clase (no necesariamente estables). [17]

El primer paso del método que se está analizando es obtener las regiones ER, para posteriormente clasificar dichas regiones en caracter o no caracter a través de dos etapas bien diferenciadas. Primero se realizó un primer filtrado utilizando características que son poco exigentes computacionalmente (área, bounding box, perímetro, etc), que ayudaron a deshacerse de muchas regiones no deseadas. Posteriormente se obtuvo las regiones clasificadas como texto mediante un segundo filtro que evaluará características más exigentes computacionalmente (relación de casco convexo, número de puntos de inflexión de contorno exterior y relación de área de los orificios). Una vez obtenidos los caracteres se agruparon en palabras para dar paso a la etapa de reconocimiento a través de Tesseract, el cual mostró la interpretación obtenida de la imagen de procesada.

En la primera etapa de clasificación, la probabilidad de cada ER siendo un carácter se estima utilizando nuevas características calculadas con O(1) complejidad por región probada. Solo ER con probabilidad local máxima se seleccionan para el segundo etapa, donde la clasificación se mejora usando más características computacionalmente costosas. Una muy eficiente y exhaustiva búsqueda con bucles de retroalimentación se aplica luego al grupo ERs en palabras y para seleccionar en la segmentación el carácter más probable. Finalmente, el texto se reconoce en una etapa de OCR entrenado usando fuentes sintéticas.

En el método propuesto, cada canal se itera por separado (en las proyecciones originales e invertidas) y posteriormente ERs son detectados. Para reducir la alta falsa

tasa positiva y alta redundancia del detector ER, solo se seleccionan los ER que corresponden a los caracteres por un clasificador secuencial. La clasificación está rota en dos etapas para una mejor eficiencia computacional. En la primera etapa, un umbral se incrementa paso a paso de 0 a 255, descriptores incrementalmente computables, se computan (en O (1)) para cada ER r y los descriptores se utilizan como características para un clasificador que estima la probabilidad condicional de clase p (r | character). El valor de p (r | character) se rastrea utilizando la relación de inclusión de ER en todos los umbrales (ver siguiente figura) y solo el ERs que corresponden al máximo local de la probabilidad p (r | character) se seleccionan (si el máximo local de la probabilidad está por encima de un límite global p y la diferencia entre el máximo local y el mínimo local es mayor que Δ min)

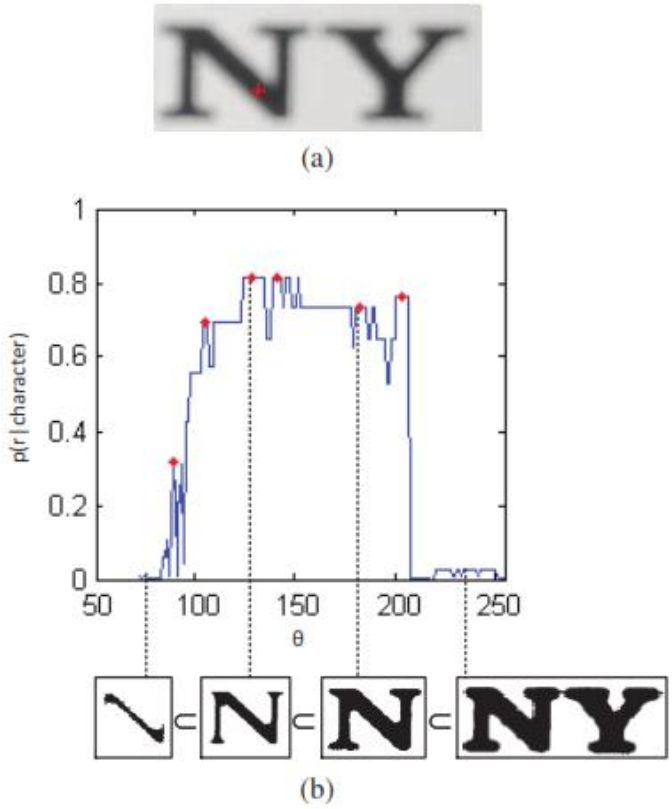


Figura 4 Clasificador secuencial del método basado en regiones

La Figura 4 representa: En la primera etapa de la clasificación secuencial, la probabilidad p(r | character) de cada ER se estima usando incrementalmente descriptores computables que explotan la relación de inclusión de ERs. (a) Un recorte de imagen de origen y la semilla inicial del ER secuencia de inclusión (marcada con una cruz roja). (b) El valor de p(r | character) en la secuencia de inclusión, ERs pasados a la segunda etapa marcada con color rojo [21]

4.2.3. Clasificador en cascada basado en características HAAR.

Un clasificador es un sistema capaz de proporcionar una predicción de pertenencia a una clase como salida a partir de un conjunto de características tomadas como entradas. En la rama de aprendizaje supervisado, construir un clasificador se basa en definir una regla que pueda asignar una etiqueta a cualquier otro dato de entrada a partir de un conjunto de ejemplos etiquetados. Una de las posibles combinaciones de un conjunto de clasificadores es la cascada y ésta dará un mejor resultado dependiendo la cantidad de datos que se introduzcan, el número de clasificadores, etc. La motivación para hacer uso de cascadas se debe a que éstas se usan para aprender conjuntos balanceados de datos. Por ejemplo, en el caso de querer aprender cabeza contra cualquier otro elemento del mundo visual, incluiremos en la cascada más imágenes de lo que no es cabeza. El uso de cascadas en este contexto permite hacer viable el aprendizaje desbalanceado a partir de la concatenación de clasificadores semi-Balanceados. [22]

La detección de objetos usando clasificadores en cascada basados en características Haar es un método efectivo de detección de objetos propuesto por Paul Viola y Michael Jones en 2001. Es un enfoque de aprendizaje automático donde la función de cascada está entrenada a partir de muchas imágenes positivas y negativas. Luego se usa para detectar objetos en otras imágenes. [23]

OpenCV cuenta con varios clasificadores en cascada entrenados que podemos encontrar en el directorio opencv\build\etc\haarcascades ó en opencv\build\etc\lbpcascades, además se puede entrenar propios clasificadores para que detecte lo que deseemos según lo requiera el proyecto que realicemos. [24]

Este procedimiento de detección de objetos clasifica las imágenes basándose en el valor de las características simples. Hay muchas motivaciones para usar las características en lugar de los pixeles directamente. La razón más común es que estas características pueden actuar para codificar ad-hoc el dominio de conocimiento que es difícil de aprender usando una cantidad finita de datos de entrenamiento. Para este sistema hay también una segunda motivación muy importante para exaltar por las características. Los sistemas basados en características operan mucho más rápido que un sistema basado en pixel. Lienhart y Maydt extienden el conjunto de características de Viola y Jones, añadiéndole las versiones rotadas de cada tipo de característica. [25]

4.3. Tecnologías y herramientas empleadas.

Figura 5 Tecnologías y herramientas utilizadas en el proyecto de peaje

Para desarrollar el prototipo de peaje cuyo campo de la visión artificial es la detección y reconocimiento de placas, se han utilizado diversas librerías, compiladores, programas y lenguajes de programación que son de libre acceso con disponibilidad ilimitada. Los recursos utilizados fueron seleccionados por su comportamiento en las tareas de visión computacional. Se dividió el proyecto: en una aplicación de escritorio y una aplicación web. La decisión de desarrollar una aplicación de escritorio en el lenguaje C++ es evidente por la facilidad de instalación en cualquier dispositivo y el cómodo mantenimiento que ello conllevaba, ya que la detección y reconocimiento de placa es de uso exclusivo en la estación de peaje. Una vez obtenida la placa, nace la aplicación web desarrollada en el lenguaje de programación Python, al ser necesario consultar el tipo del vehículo para el cobro de peaje y otras características propias del sistema web.

4.3.1. OpenCV.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones, actualmente tiene más de 2500 algoritmos desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos, calibración de cámaras, visión estérea y visión robótica; incluye funciones relacionadas con la detección y reconocimiento de texto, las cuales serán fundamentales para la creación del algoritmo creado a lo largo de este trabajo propuesto. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas. OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X, Windows, IOS y Android (de estas, se utilizó en la distribución de Linux: Ubuntu). [17]

La biblioteca Open Source Computer Vision utilizada fue en su versión 3.2.0 que presenta una estructura modular, es decir, sus diferentes funciones estas agrupadas dentro de módulos que pueden ser fácilmente adjuntadas a cualquier proyecto. [26]

4.3.2. Tesseract.

Tesseract OCR es un motor OCR open-source (libtesseract) creado para el reconocimiento óptico de caracteres (OCR). Inicialmente fue desarrollado en HP, que lo hizo open-source en 2005, desde el 2006 es desarrollado por Google bajo licencia Apache. Tesseract ofrece soporte unicode (UTF-8) y puede reconocer más de 100 lenguajes "out of the box". Además puede entrenarse para reconocer más lenguajes (Tesseract Training). Tesseract ofrece una línea de comandos, además existen otras aplicaciones con GUI (3rdParty) como el VietOCR en Java. [27] En este proyecto, se utilizó la versión 3.04.01 de un repositorio Github [28]. Su aplicación es posterior al tratamiento de las imágenes. En la siguiente figura se presenta un ejemplo de placa a reconocerse, la cual no tiene bordes ni la palabra innecesaria “Ecuador” que se presenta en todas las placas vehiculares ecuatorianas. Esta imagen sin más detalles que las letras se envía al API de Tesseract, caso contrario se pueden tener resultados menos precisos en el reconocimiento.

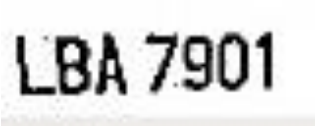


Figura 6 Placa procesada para ser reconocida por Tesseract

4.3.3. Leptonica Biblioteca Processing (LBP).

Este paquete es una dependencia de Tesseract que requiere ser incluida en los encabezados de desarrollo antes de compilar Tesseract. Leptónica realizó varias operaciones de procesamiento de imágenes internamente antes de realizar el OCR real. A continuación las versiones de Tesseract y de Leptónica requeridos: [29]

Tabla 4 Versión de Leptónica para Tesseract

TESSERACT	LEPTONICA	UBUNTU
4.00	1.74.2	Debe construir desde la fuente
3.05	1.74.0	Debe construir desde la fuente. **La versión 1.74.0 fue la usada en el proyecto.**
3.04	1.71	Ubuntu 16.04
3.03	1.70	Ubuntu 14.04
3.02	1.69	Ubuntu 12.04
3.01	1.67	

4.3.4. Qt.

En este proyecto se ha utilizado: Qt Framework y Qt Creator a la razón de la facilidad de uso e integración con el lenguaje C++ y OpenCV.

Qt es un framework multiplataforma en C++ de desarrollo de aplicaciones. Se utiliza fundamentalmente para desarrollar aplicaciones con interfaz gráfica, gracias al conjunto de controles independientes de la plataforma que ofrece, aunque también es usado para crear herramientas de línea de comando o consolas de gestión para servicios. Esto último es debido a que el API de la librería cuenta con clases para: Acceso a bases de datos mediante SQL, Procesamiento de XML, Gestión de hilos, Comunicaciones por red y Manejo de archivos.

A parte de las clases ya mencionadas para el desarrollo de aplicaciones con interfaz gráfica. Todo esto convierte a Qt en un framework muy valorado a la hora de desarrollar aplicaciones multiplataforma en C++. [30]

Qt Creator es un IDE multiplataforma programado en C++, JavaScript y QML. Creado por Trolltech el cual es parte de SDK para el desarrollo de aplicaciones con Interfaces Gráficas de Usuario (GUI) con las bibliotecas Qt, los sistemas operativos que soporta en forma oficial son: [31]

- GNU/Linux 2.6.x, para versiones de 32 y 64 bits con Qt 4.x instalado. Además hay una versión para Linux con gcc 3.3. (Se utilizó la versión de 64 bits)
- Mac OS X 10.4 o superior, requiriendo Qt 4.x
- Windows XP en adelante, requiere el compilador MinGW y Qt 4.4.3 para MinGW.

En este proyecto se utilizaron las versiones: Qt Framework 5.10.0 y Qt Creator 4.5.0.

4.3.5. Django.

Django es un marco de desarrollo en Python de código abierto y muy ligero que permite la creación rápida de páginas y aplicaciones web, sigue el patrón de diseño Modelo – Vista - Plantilla. Este framework web de alto nivel fomenta el desarrollo rápido, el diseño limpio y pragmático, se utilizó la versión 1.11.0.

Su objetivo esencial es la creación de aplicaciones web sin complicaciones. El programador se concentra en escribir su aplicación sin la necesidad de tener que reinventar la rueda. Es un resumen casi perfecto: programación rápida de páginas y

aplicaciones web. Esto es gracias a Python, un lenguaje sencillo, directo, de máxima eficiencia gracias a que la cantidad de código necesario para programar cualquier proyecto digital es realmente baja. Está basado en la filosofía DRY (Don't Repeat Yourself: No te repitas). Django es el marco de desarrollo de refactorización de código casi por excelencia, reutiliza programación de unas aplicaciones a otras sin la obligación de tener que repetir el mismo código. [32]

4.3.6. Django REST Framework (DRF).

Django REST framework (DRF) es una de las apps de terceros más usadas en Django y prácticamente se ha convertido en una herramienta obligada si lo que queremos es construir un API REST sobre Django, incluye gran cantidad de código para reutilizar (Views, Resources, etc.) y una interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP como lo son: POST y GET. [33]

Se utilizó en este proyecto la versión 3.7.7 de DRF porque permite que el sistema sea escalable, accede tener en un solo servidor el servicio web y la página, además su arquitectura permite la comunicación de datos para cualquier lenguaje ya que maneja un formato común nada específico para un dispositivo en particular.

4.3.7. Eclipse.

Como entorno de desarrollo para el sistema web se ha decidido utilizar Eclipse, debido a que es un IDE con el que se puede trabajar con mucha comodidad y rapidez, además de acoplarse perfectamente y con mucha facilidad al lenguaje Python para la creación de sitios web con Django. Algunas de sus características más destacables son:

- **Perspectivas, editores y vistas:** el concepto de trabajo está basado en las perspectivas, que son una preconfiguración de ventanas y editores relacionadas entre sí, permiten trabajar en un determinado entorno de trabajo óptimamente.
- **Gestión de proyectos:** el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios, etc.
- **Depurador de código:** incluye un potente depurador, de uso fácil e intuitivo, y que visualmente ayuda a mejorar el código.
- **Extensa colección de plug-ins**: están disponibles en una gran cantidad, unos publicados por Eclipse, otros por terceros, lo cual lo hace bastante ligero en comparación con otros IDEs.

5. MATERIALES Y MÉTODOS.

5.1. Materiales.

Este Proyecto de Trabajo de Titulación “PTT19” tuvo una planificación por fases, cada una de estas fases tuvo materiales asignados según la demanda de cada una de ellas. Los siguientes fueron los materiales utilizados y sus respectivos costos:

5.1.1. Talento Humano.

Este proyecto fue desarrollado por un Investigador con funciones de analista, desarrollado y testeador; el cual contó con la asesoría de un docente de la carrera, para lo cual se estimó un total de 480 horas, considerando costos estimados, los cuales se detallan en la siguiente tabla:

Tabla 5 Talento humano empleados en el PTT019

ROL	TIEMPO (HORAS)	($)PRECIO /HORA	($)VALOR TOTAL
Estudiante	120	5.00	600.00
Tutor	200	10.00	2000.00
		SUBTOTAL	2 600.00

El valor de $2 000.00 USD, que es costo del Director del PTT, es financiado por la Universidad Nacional de Loja, ya que los docentes tienen a su responsabilidad dirección de tesis. Así que el costo de talento humano es $ 600.00 USD.

5.1.2. Recursos Hardware y Software.

Estos recursos son imprescindibles, se emplearon para elaborar la propuesta y el desarrollo de este proyecto. Fueron los que siguen a continuación::

Tabla 6 Recursos hardware empleados en el PTT019

RECURSO	$ PRECIO UNITARIO	TIEMPO DE VIDA AÑOS	TIEMPO DE UTILIZACIÓN (MES)	($) DEPRECIACIÓN
Laptop Toshiba i7	1800	2	48	500.00
Impresora	70	1	5	50.00
Pendrive (16GB)	18	1	2	12.00
Cámara	400	0	0	400.00
Cables	20	0	0	20.00
			SUBTOTAL	982.00

Tabla 7 Recursos software empleados en el PTT019

RECURSO	DESCRIPCIÓN	($) TOTAL
Software	SW Utilitario	90.00
IDE	Eclipse	0.00
Gestor BD	MySQL	0.00
Tecnologías y herramientas	OpenCV, Tesseract, Qt, Django, Eclipse, MySQL	0.00
	SUBTOTAL	90.00

5.1.3. Servicios.

Fueron necesarios los servicios básicos de: Internet con el cual se realizó consultas y el transporte empleado para asistir a tutorías planificadas por el Director del PTT. En la siguiente tabla se detallan los costos de los servicios utilizados en el presente proyecto:

Tabla 8 Servicios empleados en el PTT019

SERVICIO	DESCRIPCIÓN	($)PRECIO UNITARIO	($) VALOR TOTAL
Internet	9 meses	35.00	315.00
Transporte	180 días	1.20	216.00
		SUBTOTAL	531.00

5.1.4. Materiales de Oficina.

En todo trabajo no se puede dejar a un lado las evidencias impresas, por lo cual surgen los costos que se detallan en la siguiente tabla:

Tabla 9 Materiales de oficina empleados en el PTT019

DESCRIPCIÓN	CANTIDAD	($)PRECIO UNITARIO	($) VALOR TOTAL
Resma de Papel	5	5	25.00
CD´s	8	1.65	13.20
Tinta de Impresora	3	26.00	78.00
Copias	3	30.00	90.00
Anillados	4	5.00	20.00
Empastados	1	50.00	50.00
Otros útiles	*	20.00	20.00
		SUBTOTAL	296.20

El presupuesto total empleado en el presente proyecto se lo ha calculado con un adicional para imprevistos que suma al 5% del costo total del presupuesto.

El presupuesto total se detalla en la tabla que se presenta a continuación:

Tabla 10 Presupuesto final empleado en el PTT019

RECURSO	DESCRIPCIÓN ($) TOTAL	
Talento Humano	600.00	
Recursos Hardware	982.00	
Recursos Software	90.00	
Servicios	531.00	
Materiales de Oficina	296.20	
	SUBTOTAL	2 499.20
	IMPREVISTOS	124.96
	TOTAL	2 624.16

5.2. Métodos.

El presente prototipo de cobro de peajes es un proyecto de software, el cual consta de una aplicación de escritorio y otra web. Su realización se acogió en métodos y técnicas integradas en una Esquematización Metodológica que facilitó al ciclo de vida de software iterativo e incremental; y a cada fase de este proyecto de fin de carrera.

5.2.1. Métodos Utilizados.

Los métodos utilizados en el desarrollo del proyecto de fin de carrera son los siguientes:

5.2.1.1. Método Científico.

Este proyecto aportó con conocimientos acerca de: fases y entregables de la metodología ICONIX, tecnologías y herramientas utilizadas en visión artificial; que posteriormente se integraron en la revisión literaria. El método científico se utilizó para formular el perfil del proyecto, revisión de literatura y entregables de ICONIX.

5.2.1.2. Método Deductivo.

Permitió a partir de aspectos generales llegar a especificaciones; tal es el caso que se estudiaron técnicas de visión artificial al campo de reconocimiento de placas. En esta generalidad se observaron librerías recomendadas para la visión por computador de la cual se eligió la biblioteca libre de mayor aceptación como es OpenCV.

Así, el marco de referencia se redujo a algoritmos y clasificadores que lograron un óptimo reconocimiento de placas en escenas naturales.Este método consiguió bosquejos de requerimientos que debía exhibir el software. Además, fue importante porque se entendió la generalización del modelo de negocio de peajes para ofrecer una solución que: se adapte a todas las empresas concesionadas, permita la adaptabilidad y escalabilidad del sistema.

5.2.1.3. Método Inductivo.

Este método complementario comprobó y ayudó al método deductivo para lograr la definición de un sistema escalable y parametrizable; se modeló los casos de uso que fueron los cimientos para la construcción de este prototipo, teniendo presente que para la metodología de desarrollo ICONIX, los casos de uso son la definición del sistema.
Se entregó una maqueta de peaje que idea una posible solución al cobro de peaje.

5.2.1.4. Metodología de desarrollo.

El marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo software de este proyecto fue: ICONIX. Esta metodología pesada-ligera se halla entre la complejidad del RUP (Rational Unified Processes) y la simplicidad y pragmatismo del XP (Extreme Programming), sin eliminar las tareas de análisis y diseño que XP no contempla [34]. Se eligió a ICONIX al ser este proyecto de mediana complejidad, corta duración, se tiene información puntual de los requerimientos a través del estado del arte e inferencias del analista, además se necesita un proceso práctico y ágil que no deje de lado la documentación y representación del comportamiento del sistema a través de UML (Lenguaje Unificado de Modelado). [35]
La herramienta CASE usada fue Visual Paradigm EC, por su extenso uso y modelamiento de los diagramas solicitados en ICONIX. Las fases y actividades fueron:

1. **Fase Análisis de Requisitos:** ICONIX en esta fase tiene como meta la revisión de requerimientos. El presente proyecto es un prototipo de peaje en general, que no es específico para un cliente en particular. Por ello, dentro de las fases del proyecto (más no de la metodología), se estableció que los requerimientos serán recolectados a partir de una revisión de literatura y de la pericia del analista.
   Según ICONIX, en esta primera fase se realizó un Modelo de Dominio, que no es más que un Diagrama de Clases extremadamente simplificado, que contiene objetos de la vida real cuyo comportamiento o datos deban ser almacenados en el sistema.
   En concordancia con la metodología se hizo en dos iteraciones un Prototipado rápido que fue presentado al director de tesis, este prototipado de diseño simuló la interfaz, funcionamiento y comportamiento del sistema.
   Finalmente, una vez establecido el prototipado se realizó el Modelo de casos de uso, para describir la manera que el usuario interactúa con el sistema. [36]
2. **Fase de Análisis y Diseño preliminar:** ICONIX consolida la primera fase mediante ilustraciones de las interacciones entre los objetos participantes de un caso de uso, a través de una especificación de casos de uso, como un flujo principal de acciones

pudiendo contener flujos alternos y excepciones; y a través de diagramas de robustez que analiza el texto narrativo de la especificación de cada caso de uso e identificar un conjunto inicial de objetos participantes de cada caso de uso [36] [37]
Como se puede inferir, ambas representaciones llegan a detallar mejor los casos de uso con la diferencia que uno es a través de texto y el otro es gráficamente. En este proyecto se utilizó la Especificación de casos de uso (descripción textual).

3. **Fase de Diseño:** ICONIX reconoce todos los elementos que forman parte del sistema y muestra todos los cursos alternos que pueden tomar los casos de uso, mediante la representación del diagrama de secuencia. [36] [38]
   Los diagramas de secuencia representados derivaron de las fichas de caso de uso, los cuales a su vez se relacionan con casos de uso que se relacionan con requisitos. Esto implicó que una vez finalizado el diseño, tras refinar nuevamente el Diagrama de clases, se pudo verificarlo directamente gracias a este factor de trazabilidad que expone ICONIX, para finalmente prepararnos para la última fase.
4. **Fase de Implementación:** a partir de un buen diseño ICONIX codifica el software para después entregar el sistema. Los entregables de esta fase fueron:
   Diagrama de componentes: mostró la distribución física de los elementos que componen la estructura interna del sistema y sus relaciones.
   Diagrama de despliegue: mostró los elementos físicos que componen el sistema, así como el hardware necesario para que se ejecute adecuadamente. [35] [36]

Los entregables de cada fase de ICONIX, se muestran en la sección 6. RESULTADOS.

5.2.2. Técnicas.

5.2.2.1. Documentación bibliográfica.

Se definió toda la información necesaria en el desarrollo de la propuesta, sustentó el estado del arte, requerimientos y entendimiento del modelo de negocio de peajes en Ecuador. Se hizo efectiva en consultas de: libros, artículos científicos, revistas indexadas y fuentes confiables de Internet.

5.2.2.2. Revisión de tutorías.

Esta técnica es necesaria en todo proyecto de trabajo de titulación. Hace alusión al control del cumplimiento de objetivos por parte del director del proyecto de trabajo de titulación al tesista. En este proceso se utilizaron medios de enseñanza - aprendizaje continuo que consiguieron obtener los resultados esperados de cada fase del proyecto.

6. RESULTADOS.

Tras haber presentado la revisión literaria, así como los métodos que permitieron culminar cada fase con éxito, se presentan ahora los resultados obtenidos al desarrollar el proyecto de fin de carrera: **Diseño y construcción de un prototipo para cobro de peajes con Visión Artificial.**

El desarrollo del sistema de peaje se acopla a la metodología ICONIX, adaptando cada uno de los entregables especificados en la guía del conocimiento de ICONIX con las fases del proyecto. Esto, con el propósito de alcanzar los resultados esperados según los objetivos de este proyecto de titulación.

6.1. Fase uno: Analizar el estado del arte de los peajes en nuestro país, para determinar los requerimientos que construirán el software.

El cumplimiento de este objetivo se ve reflejado en la sección 4. REVISIÓN DE LITERATURA, en específico en la sub-sección 4.1 Peaje. Se hace referencia a toda la sección ya que se consideran como requerimientos las librerías, técnicas, algoritmos, clasificadores, tecnologías y herramientas que consiguieron el desarrollo del sistema.

Esta subsección ilustra los entregables establecidos por la metodología ICONIX como son: Prototipado, Modelo de dominio, Diagramas de casos de uso, Especificación de casos de uso, Diagramas de secuencia y Diagrama de clases, además de los requerimientos funcionales y no funcionales.

6.1.1. Requerimientos funcionales.

Las siguientes funciones que realiza el sistema fueron obtenidas a partir de la documentación bibliográfica, trabajo conjunto del director de tesis y pericia del tesista; teniendo en cuenta que el sistema no tiene un usuario final, sino es un prototipo de peaje que podría ser utilizado por cualquiera de las empresas concesionadas autorizadas por el Ministerio de Transporte y Obras Públicas.

Tabla 11 Requerimientos funcionales del sistema

REF	REQUERIMIENTO	PRIORIDAD
RF01	Reconocer placas de vehículos utilizando una cámara	ALTA
RF02	Obtener caracteres de la placa detectada.	ALTA

RF03	Reconocimiento de Caracteres – OCR.	ALTA
RF04	Obtener datos de vehículo del servicio de la ANT.	ALTA
RF05	Mostrar placa detectada.	ALTA
RF06	Clasificar al vehículo según el tipo.	ALTA
RF07	Realizar cobro al vehículo según el tipo.	ALTA
RF08	Registrar fecha y hora de cuando se realizan los cobros	ALTA
RF09	Registrar usuarios.	MEDIA
RF10	Autenticar usuarios.	MEDIA
RF11	Administrar cámaras.	MEDIA
RF12	Administrar Tarifas de Cobro.	MEDIA
RF13	Permitir en el sistema web consultas por búsqueda de placas y fechas.	ALTA
RF14	Descargar reporte en formato PDF con los detalles del cobro.	ALTA
RF15	Generación de gráficas sobre los vehículos que transitan por los peajes.	MEDIA

6.1.2. Requerimientos no funcionales.

Las restricciones en el diseño del software, que describen las cualidades que el sistema debe tener son las siguientes:

Tabla 12 Requerimientos no funcionales del sistema

RNF01 REQUISITOS DE RENDIMIENTO
• Garantizar que el diseño de las consultas u otro proceso no afecte el desempeño de la base de datos, ni considerablemente el tráfico de la red. • El tiempo de respuesta (en tiempo real) a peticiones máximo es de 10 segundos.
RNF02 REQUISITOS DE SEGURIDAD
• Para consultas generales se permite acceso sin necesidad de loguearse. • La generación de reportes requiere la autenticación de usuarios administradores. • Según las características de los usuarios se establecerán sus privilegios de usuario y restricciones de acceso al sistema. • Las contraseñas serán encriptadas.
RNF03 REQUISITOS DE FIABILIDAD
• La única persona que tendrá acceso a la modificación de los datos será: el administrador y el equipo de mantenimiento del sistema. • El sistema debe controlar la autenticidad de placas vehiculares.
RNF04 REQUISITOS DE DISPONIBILIDAD
• La disponibilidad del sistema debe ser continua los 7 días de la semana por 24 horas.
RNF05 REQUISITOS DE MANTENIBILIDAD
• Entregar documentación para realizar operaciones de mantenimiento con el menor esfuerzo posible. • Entregar un manual de usuario del sistema que permita conocer su funcionamiento.

RNF06 REQUISITOS DE PORTABILIDAD
• El sistema utilizará una interfaz intuitiva y amigable. • El sistema tiene una arquitectura cliente servidor. • El sistema será multiplataforma siempre y cuando se disponga de un navegador web y en el caso de la aplicación de escritorio se ejecutará en una distribución de Linux.
RNF07 REQUISITOS DE FACILIDAD DE USO
• Se debe visualizar el texto fácilmente a una distancia de 40 centímetros. • Evitar colores asociados con formas comunes de daltonismo.
RNF08 RESTRICCIONES DE IMPLEMENTACIÓN
• El lenguaje de programación es: C++ para la detección y reconocimiento vehicular; y, para el sistema web de peaje el lenguaje de programación es Python. • El gestor de base de datos es MySQL. • OpenCV y Tesseract son las librerías para la detección y reconocimiento vehicular. • El framework de programación para interfaces es QT, en el web son: Django y Django REST.
RNF09 METODOLOGÍA DE DESARROLLO
El sistema debe exponer una metodología ágil y pesada para su correcta documentación. Por ello es altamente sugerente la metodología ICONIX.
RNF10 OTROS REQUISITOS
• La licencia del sistema debe ser open source. • Para garantizar la adaptabilidad a cualquier navegador web, se opta por el lenguaje de maquetación HTML5 y una estandarización con el estilo de hojas de CSS.

6.1.3. Prototipo de Interfaces.

En esta parte se da a conocer de manera general la distribución y apariencia que podrían tener las interfaces para el funcionamiento del sistema. Se realizaron 4 prototipos base de acuerdo a las funcionalidades que deberá tener el sistema, se distribuyeron de manera que el sistema fácil de usar y de rápido acceso a las diferentes funcionalidades que posee el sistema. A continuación se presentan las plantillas de ingreso y login para el usuario (ver Figura 7)

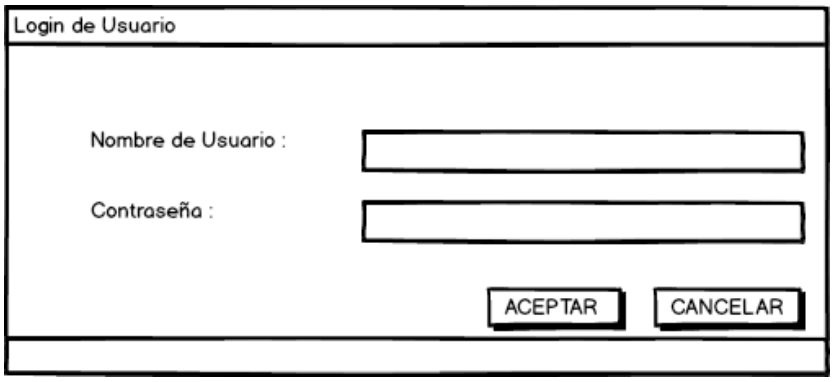


Figura 7 Plantilla para el ingreso y login de usuario.

Se realizó también la plantilla para la pantalla principal donde se tendrá acceso a todas las funcionalidades del sistema (ver Figura 8).

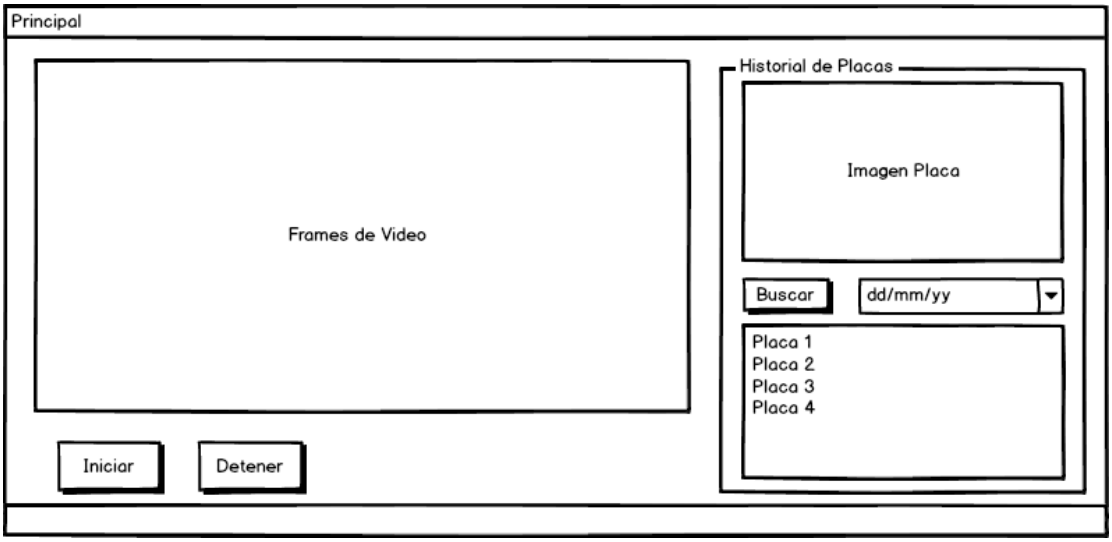


Figura 8 Plantilla para la pantalla principal del sistema.

En la plantilla para las diferentes administraciones se utilizaron diferentes tabs, para organizar de mejor manera cada uno de los diferentes módulos (ver Figura 9).

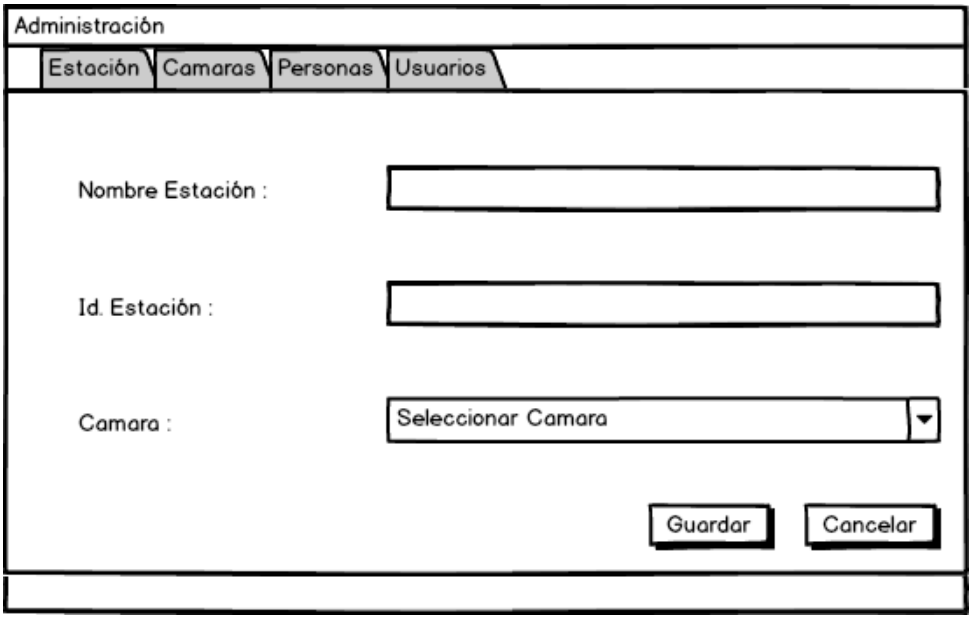


Figura 9 Prototipo de pantalla para administración.

Finalmente, la plantilla para la pantalla de generación de tickets de cobro (ver Figura 10).

Figura 10 Prototipo para pantalla de cobro de peaje.

Considerando las diferentes plantillas realizadas y los requerimientos funcionales obtenidos en la sección anterior, se procedió al prototipado de cada una de las ventanas, las cuales se muestran a continuación (ver Figura 11 a la 14).

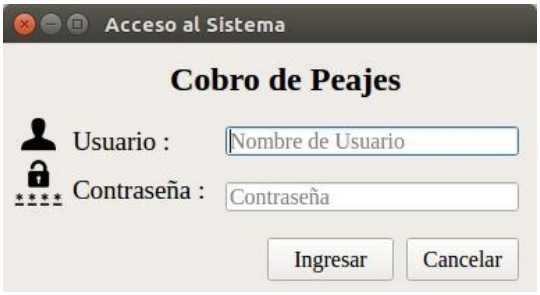


Figura 11 Pantalla de ingreso de usuarios

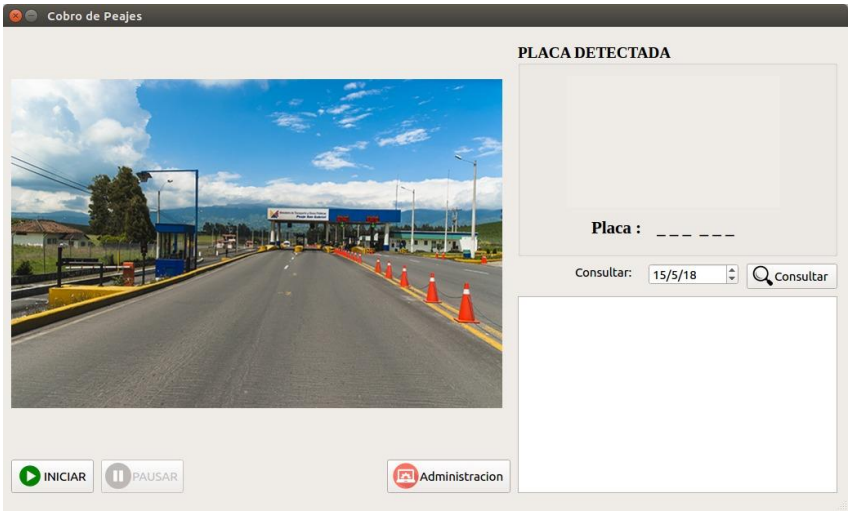


Figura 12 Pantalla principal del sistema

Figura 13 Pantalla para el cobro de tickets

Figura 14 Pantalla de administración para módulos

6.1.4. Modelo de dominio.

En concordancia con la metodología ICONIX, la cual enmarcó el desarrollo software de este proyecto de trabajo de titulación, se realizó uno de los modelos requeridos que permitió abstraer de una manera general el funcionamiento del sistema y las clases más importantes. A continuación se presenta el modelo del dominio y una breve descripción de las clases.

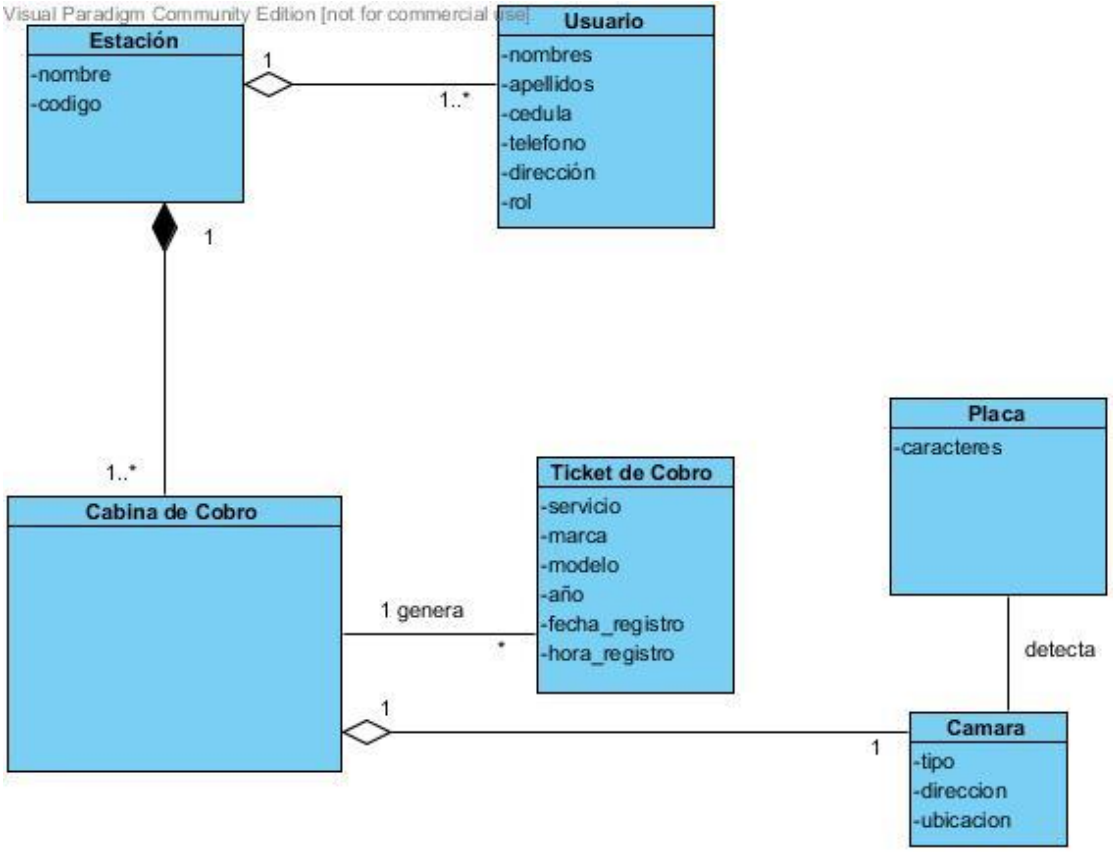


Figura 15 Modelo de Dominio

- **Estación:** Clase que abstrae y une todos los componentes del prototipo.
- **Usuario:** Clase que permite dar privilegios de acceso y uso al sistema.
- **Cabina:** Clase que nos permite dar funcionalidad y emisión de tickets.
- **Ticket de Cobro:** Esta clase contiene toda la información y métodos que permiten generar los tickets de cobro.
- **Cámara:** Clase que permite obtener los frames de video.
- **Placa:** Esta clase tiene los métodos e información que permite consultar los datos del vehículo.

6.1.5. Diagrama de Casos de Uso.

Siguiendo los lineamientos de ICONIX, para esta sección se identificaron todos los procesos que se deben realizar dentro del sistema basándose en el documento de requisitos, los cuales se extrajeron y se asignaron a cada actor del sistema.

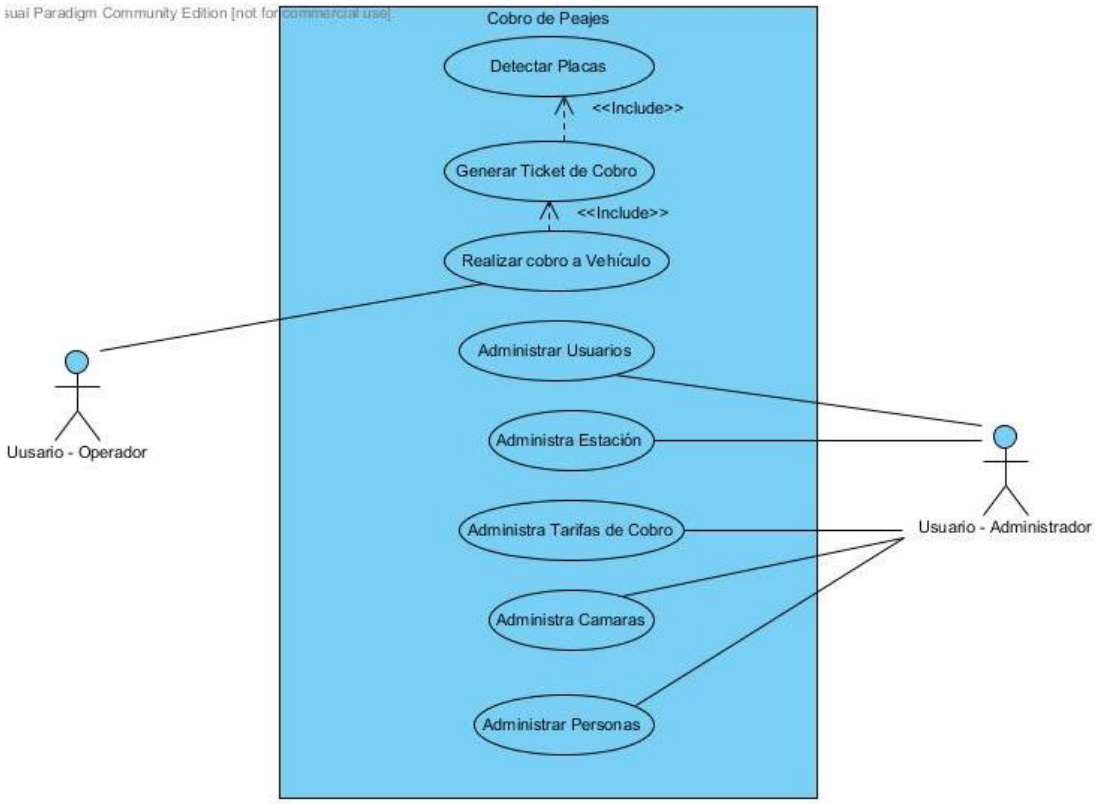


Figura 16 Diagrama de Casos de Uso

6.1.6. Especificación de casos de uso.

Con la especificación (descripción) de caso de uso, se pudo establecer cada uno de los procesos y funcionalidades que el sistema debe poder realizar.

6.1.6.1. Módulo de Personas.

A continuación se describen cada uno de los casos de uso que el módulo de personas podrá realizar.

- En la tabla 13 se describe el caso de uso: Crear persona:
- En la tabla 14 se muestra el caso de uso: Modificar persona.
- En la tabla 15 se muestra el caso de uso: Eliminar persona.

Tabla 13 Caso de Uso Crear persona

Nombre:	Crear Persona
Actor:	Administrador
Descripción:	
Detalla el flujo para la creación de una persona en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de personas.	
Flujo Normal:	
1. El actor da clic sobre el botón ingresar. 2. Se genera una nueva fila de registro. 3. El actor ingresa la información necesaria y solicitada. 4. Al finalizar el actor presiona la tecla “Enter”. 5. El registro es guardado.	
Flujo Alternativo:	
6. Si la información ingresada no es válida, el registro no es guardado.	
Poscondiciones:	
El sistema guardará la información ingresada en el registro.	

Tabla 14 Caso de Uso Modificar persona

Nombre:	Modificar Persona
Actor:	Administrador
Descripción:	
Detalla el flujo para la modificación de una persona en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de personas.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea modificar. 2. Se modifica la información necesaria. 3. Al finalizar de modificar el actor presiona la tecla “Enter”. 4. El registro modificado es guardado.	
Flujo Alternativo:	
5. Si la información ingresada no es válida, el registro no es modificado.	
Poscondiciones:	
El sistema guardará la información modificada en el registro.	

Tabla 15 Caso de Uso Eliminar persona

Nombre:	Eliminar Persona
Actor:	Administrador
Descripción:	
Detalla el flujo para la eliminación de una persona en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de personas.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea eliminar. 2. Hace clic sobre el botón “Eliminar”. 3. El registro es eliminado.	
Flujo Alternativo:	
4. Si la persona a eliminar, posee un usuario el registro no se eliminará. 5. El actor cierra la administración de personas, el registro no se elimina.	
Poscondiciones:	
El sistema elimina el registro de la persona.	

6.1.6.2. Módulo de Cámaras.

Crear, Modificar y Eliminar Cámara, son los casos de uso que este módulo realiza.

Tabla 16 Caso de Uso Crear cámara

Nombre:	Crear Cámara
Actor:	Administrador
Descripción:	
Detalla el flujo para la creación de una cámara en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de cámaras.	
Flujo Normal:	
1. El actor da clic sobre el botón ingresar. 2. Se genera una nueva fila de registro. 3. El actor ingresa la información necesaria y solicitada. 4. Al finalizar el actor presiona la tecla “Enter” y el registro es guardado.	
Flujo Alternativo:	
5. Si la información ingresada no es válida, el registro no es guardado.	
Poscondiciones:	
El sistema guardará la información ingresada de la nueva cámara en el registro.	

Tabla 17 Caso de Uso Modificar cámara

Nombre:	Modificar Cámara
Actor:	Administrador
Descripción:	
Detalla el flujo para la modificación de una cámara en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de Cámaras.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea modificar. 2. Se modifica la información necesaria. 3. Al finalizar de modificar el actor presiona la tecla “Enter”. 4. El registro modificado es guardado.	
Flujo Alternativo:	
5. Si la información ingresada no es válida, el registro no es modificado.	
Poscondiciones:	
El sistema guardará la información modificada de una cámara en el registro.	

Tabla 18 Caso de Uso Eliminar cámara

Nombre:	Eliminar Cámara
Actor:	Administrador
Descripción:	
Detalla el flujo para la eliminación de una cámara en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de cámaras.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea eliminar. 2. Hace clic sobre el botón “Eliminar”. 3. El registro es eliminado.	
Flujo Alternativo:	
4. Si la cámara a eliminar, está siendo usada por el sistema no se elimina. 5. El actor cierra la administración de cámaras, el registro no se elimina.	
Poscondiciones:	
El sistema elimina el registro de la cámara.	

6.1.6.3. Módulo de Tarifas de Cobro.

Agregar, Modificar y Eliminar Tarifa, son los casos de uso que este módulo realiza.

Tabla 19 Caso de Uso Agregar tarifa

Nombre:	Crear Tarifa
Actor:	Administrador
Descripción:	
Detalla el flujo para la creación de una tarifa en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de Tarifas.	
Flujo Normal:	
1. El actor da clic sobre el botón ingresar. 2. Se genera una nueva fila de registro. 3. El actor ingresa la información necesaria y solicitada. 4. Al finalizar el actor presiona la tecla “Enter” y el registro es guardado.	
Flujo Alternativo:	
5. Si la información ingresada no es válida, el registro no es guardado.	
Poscondiciones:	
El sistema guardara la información ingresada en el registro de tarifas.	

Tabla 20 Caso de Uso modificar tarifa

Nombre:	Modificar Tarifa
Actor:	Administrador
Descripción:	
Detalla el flujo para la modificación de una tarifa en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de Tarifas.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea modificar. 2. Se modifica la información necesaria. 3. Al finalizar de modificar el actor presiona la tecla “Enter”. 4. El registro modificado es guardado.	
Flujo Alternativo:	
5. Si la información ingresada no es válida, el registro no es modificado.	
Poscondiciones:	
El sistema guardara la información modificada en el registro.	

Tabla 21 Caso de Uso Eliminar Tarifa

Nombre:	Eliminar Tarifa
Actor:	Administrador
Descripción:	
Detalla el flujo para la eliminación de una tarifa en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de tarifas.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea eliminar. 2. Hace clic sobre el botón “Eliminar”. 3. El registro es eliminado.	
Flujo Alternativo:	
4. Si la tarifa a eliminar, está siendo usada por cobros previos no se eliminara. 5. El actor cierra la administración de tarifas, el registro no se elimina.	
Poscondiciones:	
El sistema elimina el registro de la tarifa.	

6.1.6.4. Módulo de Estación.

A continuación se describen cada uno de los casos de uso que el módulo de Estación podrá realizar:

- En la tabla 22 se muestra el caso de uso Configurar estación.
- En la tabla 23 se muestra el caso de uso modificar configuración de estación.

Tabla 22 Caso de Uso Configurar estación

Nombre:	Configurar estación
Actor:	Administrador
Descripción: Detalla el flujo para el ingreso de las configuraciones para una cabina de estación.	
Precondiciones: El actor ingresa al sistema y al módulo de Estación. Debe estar ingresada al menos una cámara.	
Flujo Normal: 1. El actor ingresa la información solicitada. 2. Se debe seleccionar una cámara. 3. El actor presiona en guardar configuración. 4. La configuración es guardada.	
Flujo Alternativo: 5. Si la información ingresada no es válida, la configuración no es guardada.	
Poscondiciones: El sistema guardará la información ingresada para la configuración de estación.	

Tabla 23 Caso de Uso Modificar estación

Nombre:	Modificar Estación
Actor:	Administrador
Descripción: Detalla el flujo para la modificación de la estación del sistema.	
Precondiciones: El actor ingresa al sistema y al módulo de administración de estación.	
Flujo Normal: 6. El actor modifica la información que desea cambiar. 7. Al finalizar de modificar el actor presiona en Guardar Configuración. 8. Los cambios son guardados.	
Flujo Alternativo: 9. Si la información ingresada no es modificada, el registro no es modificado.	
Poscondiciones: El sistema guardará la información modificada en la configuración de estación.	

6.1.6.5. Módulo de Usuarios.

A continuación se describen cada uno de los casos de uso que el módulo de usuarios podrá realizar.

En la tabla 24 se muestra el caso de uso Agregar Usuario.

Tabla 24 Caso de Uso Crear Usuario

Nombre:	Crear Usuario
Actor:	Administrador
Descripción:	
Detalla el flujo para la creación de un usuario en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de Usuarios. Debe haber ingresadas personas.	
Flujo Normal:	
1. El actor da clic sobre el botón ingresar. 2. Se genera una nueva fila de registro. 3. El actor ingresa la información necesaria y solicitada. 4. Al finalizar el actor presiona la tecla “Enter”. 5. El registro es guardado.	
Flujo Alternativo:	
6. Si la información ingresada no es válida, el registro no es guardado.	
Poscondiciones:	
El sistema guardará la información ingresada en el registro de usuarios.	

En la tabla 25 se muestra el caso de uso modificar Usuario.

Tabla 25 Caso de Uso Modificar Usuario

Nombre:	Modificar Usuario
Actor:	Administrador
Descripción:	
Detalla el flujo para la modificación de un usuario en el sistema.	
Precondiciones:	
El actor ingresa al sistema y al módulo de Administración de Usuarios.	
Flujo Normal:	
1. El actor da clic sobre el registro que desea modificar. 2. Se modifica la información necesaria. 3. Al finalizar de modificar el actor presiona la tecla “Enter”. 4. El registro modificado es guardado.	
Flujo Alternativo:	
5. Si la información ingresada no es válida, el registro no es modificado.	
Poscondiciones:	
El sistema guardará la información modificada en el registro.	

En la tabla 26 se muestra el caso de uso eliminar usuario.

Tabla 26 Caso de Uso Eliminar Usuario

Nombre:	Eliminar Usuario
Actor:	Administrador
Descripción: Detalla el flujo para la eliminación de un usuario en el sistema.	
Precondiciones: El actor ingresa al sistema y al módulo de Administración de Usuarios.	
Flujo Normal: 1. El actor da clic sobre el registro que desea eliminar. 2. Hace clic sobre el botón “Eliminar”. 3. El registro es eliminado.	
Flujo Alternativo: 4. El actor cierra la administración de usuarios, el registro no se elimina.	
Poscondiciones: El sistema elimina el registro del usuario.	

6.1.6.6. Módulo de Cobro.

A continuación se describe el caso de uso que el módulo de Cobro de peaje podrá realizar.

En la tabla 27 se muestra el caso de uso Cobrar Ticket.

Tabla 27 Caso de Uso Cobrar Ticket

Nombre:	Cobrar Ticket
Actor:	Operador
Descripción: Detalla el flujo para el cobro de un ticket generado.	
Precondiciones: El actor ingresa al sistema y debe empezar a capturar frames de video desde la cámara.	
Flujo Normal: 1. El sistema detecta una placa. 2. El sistema genera un formulario de cobro. 3. El actor presiona en cobrar y realiza el cobro del peaje.	
Flujo Alternativo: 4. El actor presiona en cancelar, el cobro no es realizado.	
Poscondiciones: El sistema guardará la información ingresada en el registro cobrar Ticket de peaje.	

6.1.7. Diagramas de Secuencia.

Con los diagramas de secuencia se logra identificar los mensajes entre todos los objetos definidos en el modelo del dominio, por lo que es importante definir todas sus interacciones. A continuación se presentan los diagramas de secuencia para el módulo de cobro: Cobrar Ticket y para el módulo tarifas de cobro: Agregar tarifa.

Se dejan como evidencia estos dos diagramas de secuencia a la razón que para los 5 módulos con excepción del de Cobro, todos representan un CRUD (Crear, Leer, Modificar y eliminar). La comprensión de estos dos diagramas dan una definición de la interacción de los objetos del sistema, cuyo primordial y vinculante es el cobro de peaje.

6.1.7.1. Módulo de Cobro.

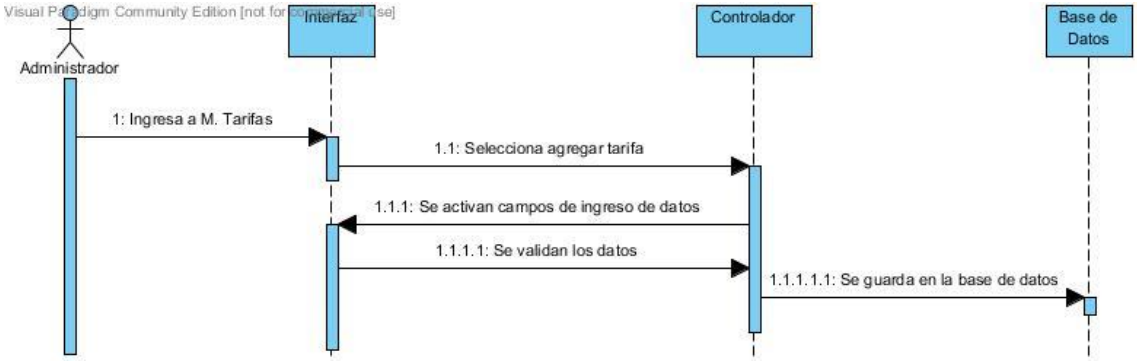


Figura 17 Diagrama de secuencia ingresar tarifa

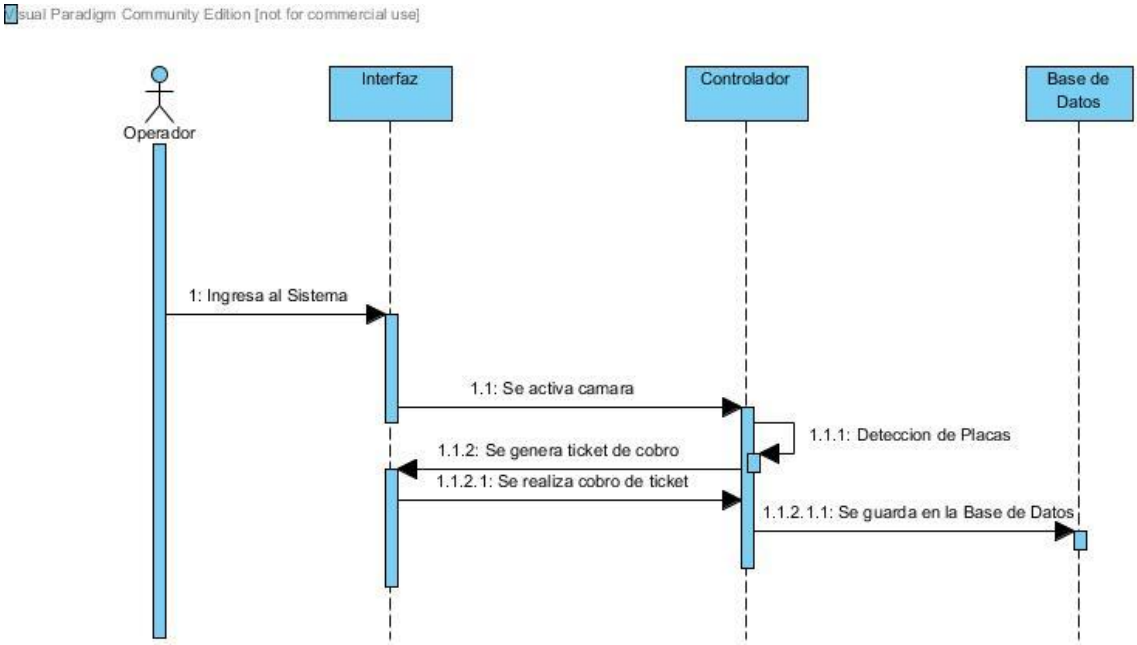


Figura 18 Diagrama de secuencia cobrar ticket

6.1.8. Actualización del Modelo del Dominio.

En la siguiente figura se presenta la actualización del modelo del dominio, en el cual se incluyen todos los atributos y relaciones entre objetos.

El modelo del dominio actualizado es el Diagrama de Clases que es solicitado como entregable en la fase de diseño (previo a la implementación) según la metodología de desarrollo seguida: ICONIX.

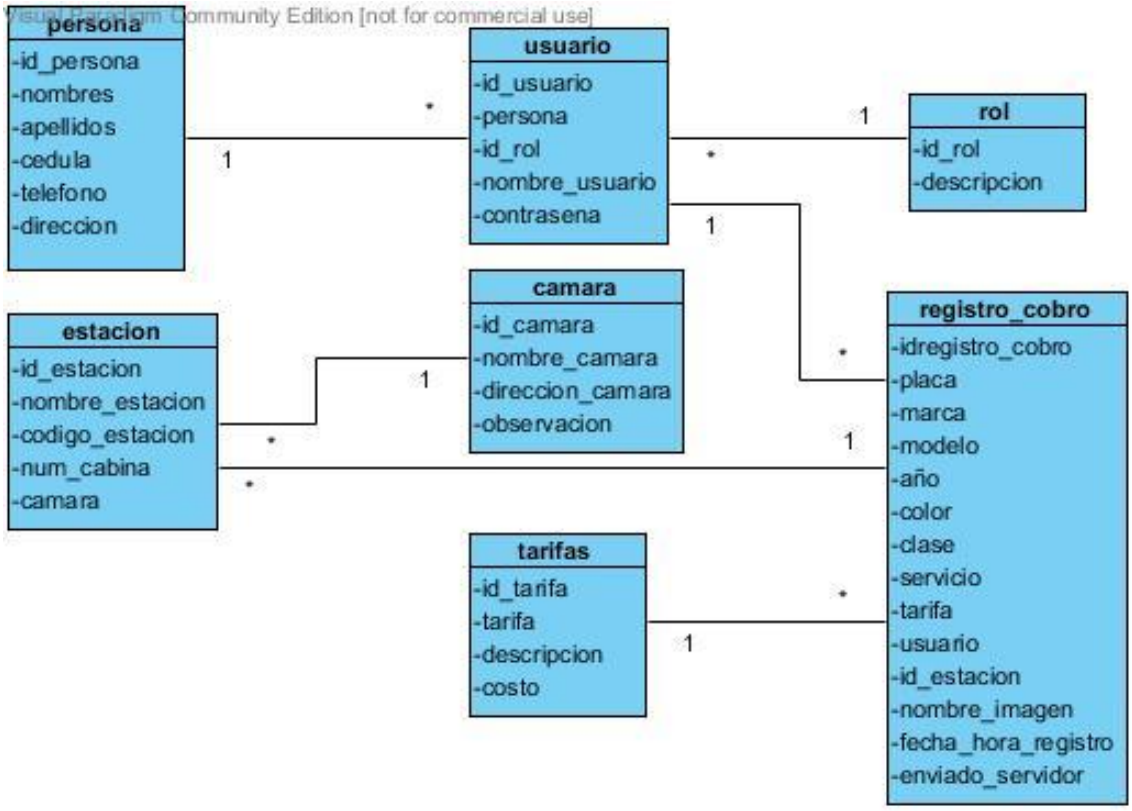


Figura 19 Actualización del modelo del dominio

6.2. Fase dos: Configurar y aplicar técnicas de Visión y Reconocimiento de Vehículos con OpenCV.

Toda esta sección está dedicada a la documentación esencial de: librerías base, OpenCV, Integración de OpenCV con QT, finalmente se documenta la creación e implementación del clasificador HAAR. Todas estas subsecciones permitieron la visión y reconocimiento de los vehículos para el presente prototipo de peaje.

Para el desarrollo del software se ha utilizado el Sistema Operativo Ubuntu en su versión 16.04, dada la rapidez, robustez y fácil integración que tiene con las herramientas que se utilizaron en el desarrollo del presente Trabajo de Titulación.

6.2.1. Instalación de Librerías Base.

Las herramientas previas que se deben instalar y configurar son Leptónica y Tesseract ya que son la base para realizar a continuación el OCR. Para la instalación de la librería leptónica se debe descomprimir el archivo, y configurar los archivos para su posterior instalación.

Ingresaremos al directorio de leptónica y su estructura se deberá presentar como en la siguiente imagen:

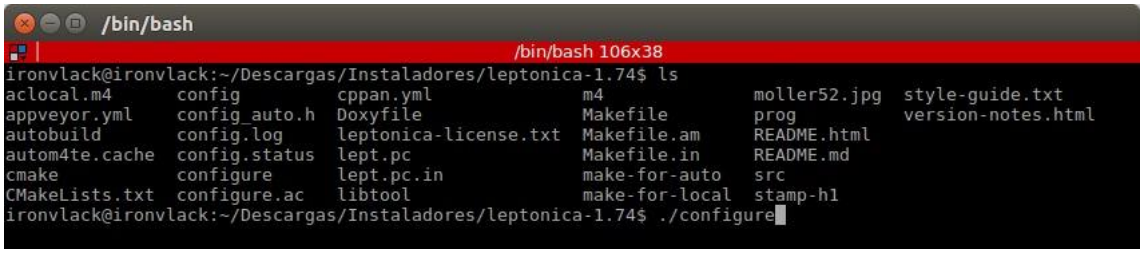


Figura 20 Directorio de librería leptónica

Una vez dentro del directorio escribiremos:

1. El comando ./configure, lo que configurará la librería con las dependencias que sean necesarias para su posterior compilación e instalación, luego
2. Procederemos a escribir el comando make, el cual nos permitirá compilar las librerías para poder instalarlas en el sistema;
3. Una vez termine la compilación escribiremos el comando make install, el cual empezará a realizar la instalación de la librería en nuestro sistema,
4. Finalizado todo el proceso deberá quedarnos de la siguiente manera:

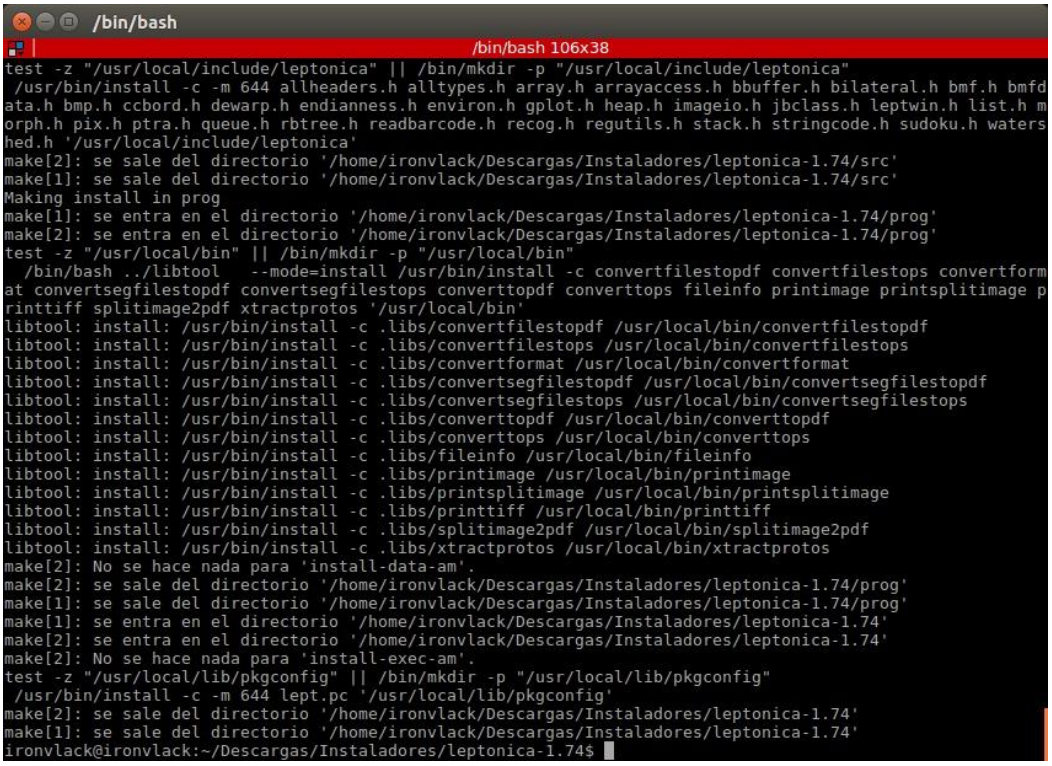


Figura 21 Finalización de la instalación de leptónica

De igual manera para la instalación de la librería Tesseract se deben realizar los mismos 4 pasos enunciados anteriormente. A continuación se muestra la imagen del directorio:

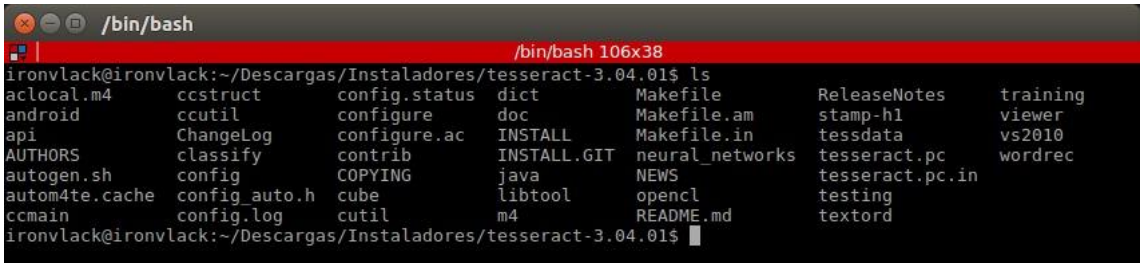


Figura 22 Directorio de la librería tesseract

Como se dijo, una vez dentro del directorio escribiremos:

1. El comando ./configure, lo que configurará la librería con las dependencias que sean necesarias para su posterior compilación e instalación, luego
2. Procederemos a escribir el comando make, el cual nos permitirá compilar las librerías para poder instalarlas en el sistema;
3. Una vez termine la compilación escribiremos el comando make install, el cual empezará a realizar la instalación de la librería en nuestro sistema,
4. Finalizado todo el proceso deberá quedarnos de la siguiente manera:

Y luego de compilar e instalar la librería tendremos el siguiente resultado:

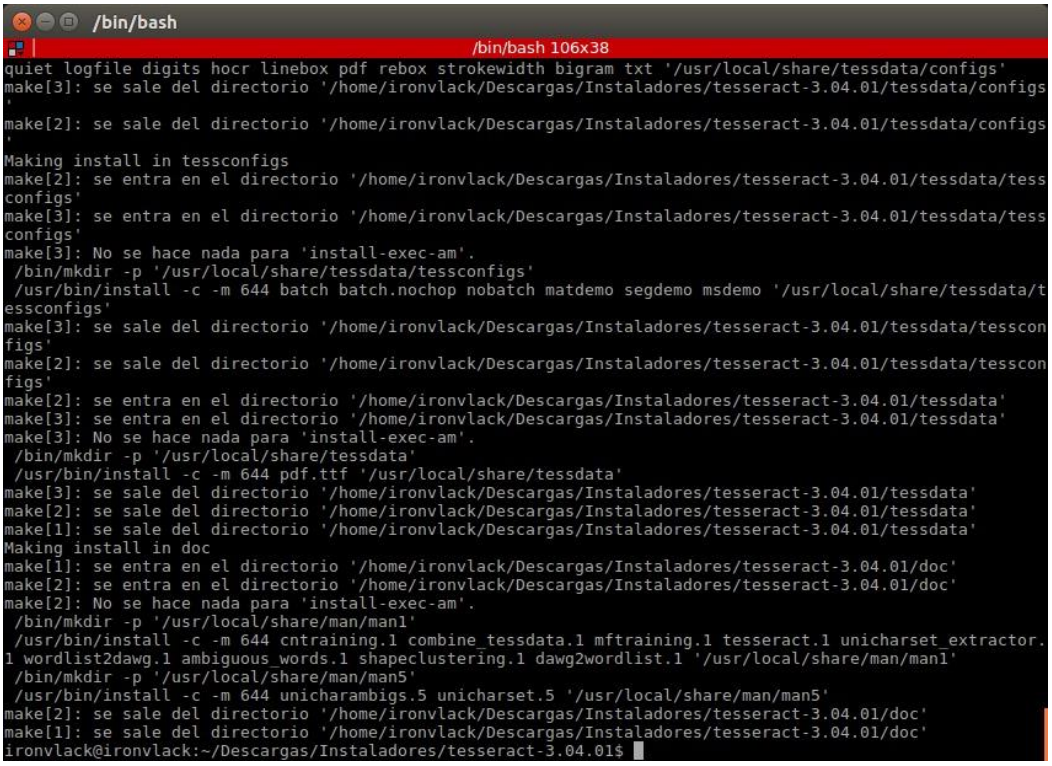


Figura 23 Finalización de la instalación de la librería Tesseract

De esta manera lo que se está realizando es configurar las librerías e instalar dependencias que se utilizarán conjuntamente con OpenCV, para que esta librería pueda trabajar correctamente.

6.2.2. Instalación y Compilación de OpenCV.

Para la instalación de OpenCV se utilizará una herramienta adicional llamada Cmake que nos permitirá generar y automatizar el código para que este pueda adicionalmente integrarse a trabajar con el lenguaje C++ y el Framework QT. Además deberemos obtener también los módulos adicionales llamados OpenCV-Contrib que no son módulos oficiales de la librería, pero que se los puede integrar.

Estos módulos no son oficiales porque han sido recientemente desarrollados o aún no han tenido una amplia aceptación y uso por la comunidad, por lo que los van agregando paulatinamente.

Antes de pasar a la compilación ubicaremos la carpeta donde realizamos la extracción de OpenCV y OpenCV-Contrib, crearemos también una carpeta llamada build que será

el directorio que se usará como directorio para almacenar el código resultante, la estructura de carpetas nos debe quedar de la siguiente manera:

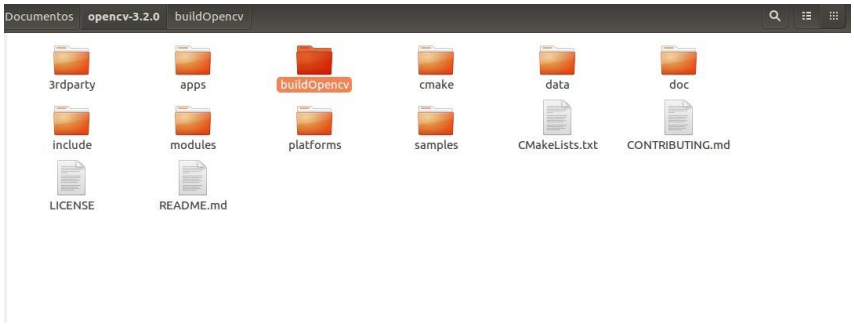


Figura 24 Estructura del directorio de OpenCV

Instalaremos la herramienta Cmake escribiendo en una terminal el siguiente comando: *sudo apt-get install cmake* el cual automáticamente instalará y configurará el programa para que lo podamos utilizar.

Cmake cuenta con dos maneras de trabajar, una por medio de consola en la cual se le pasan banderas y parámetros de configuración con las rutas de archivos a los cuales queremos compilar y configurar. Y otra de manera gráfica que nos permitirá realizar el mismo trabajo pero de una manera más sencilla. A continuación presentaremos la compilación de una manera gráfica.

Desde el dash de Ubuntu escribiremos cmake y abriremos el programa, nos aparecerá una interfaz como la siguiente:

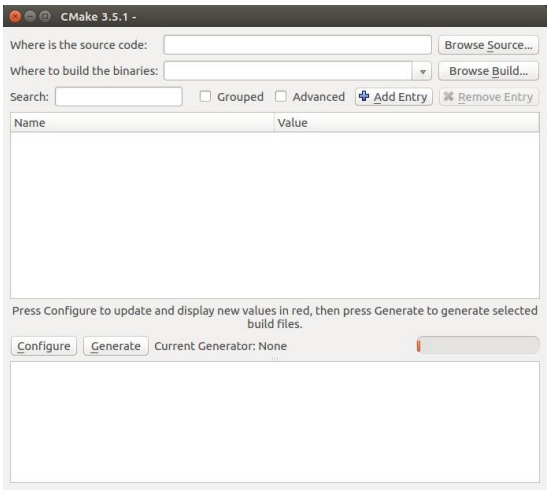


Figura 25 Interfaz herramienta Cmake

En la parte superior en la casilla de “Browse source” navegaremos hacia nuestro directorio de OpenCV y seleccionaremos la carpeta, en la segunda casilla “Browse

Build” deberemos colocar el directorio de nuestra carpeta llamada build donde irán los binarios que se generan. Una vez seleccionadas las carpetas fuente y destino seleccionaremos los compiladores a utilizarse, para lo cual dejaremos en los que vienen por defecto en el sistema:

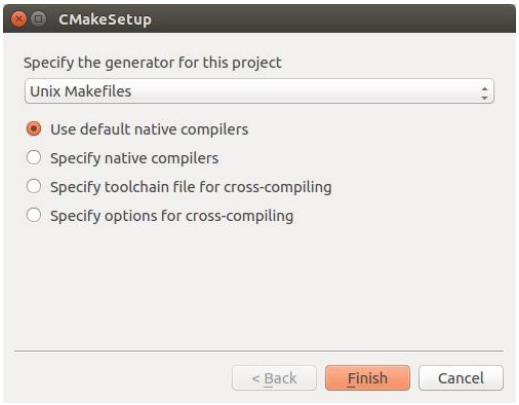


Figura 26 Selección de compiladores en CMake

En la parte inferior deberemos dar clic cobre el botón “Configure”, si hemos realizado bien los pasos anteriores deberemos ver la siguiente pantalla, además de la salida por consola “Configuring done” la cual nos indica que la configuración previa a la compilación se realizó de una manera exitosa:

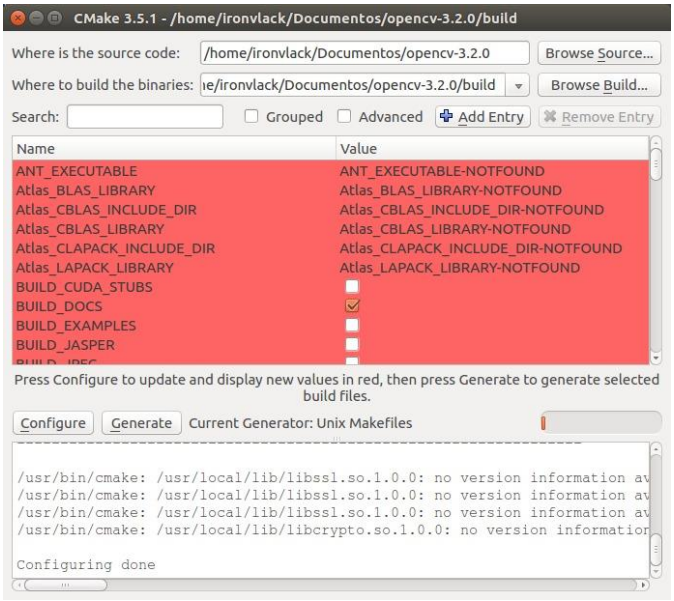


Figura 27 Configuración de archivos fuente a compilarse

Una vez tengamos cmake en ese estado deberemos incluir los módulos de OpenCV Contrib que contiene el modulo “Text” que nos permitirá realizar el OCR en nuestro sistema, lo agregaremos de la siguiente manera:

1. Buscaremos la entrada OPENCV_EXTRA_MODULES_PATH
2. Seleccionaremos la casilla y en nuestra carpeta seleccionamos la carpeta “Modules”

Una vez hayamos verificado y configurado las imágenes daremos clic nuevamente en “Configure” y finalmente en “Generate”. Con lo cual habremos terminado de configurar y generar nuestro OpenCV para ser utilizado en el Framework QT.

Las salidas por pantalla de cada una de las configuraciones se muestran a continuación:

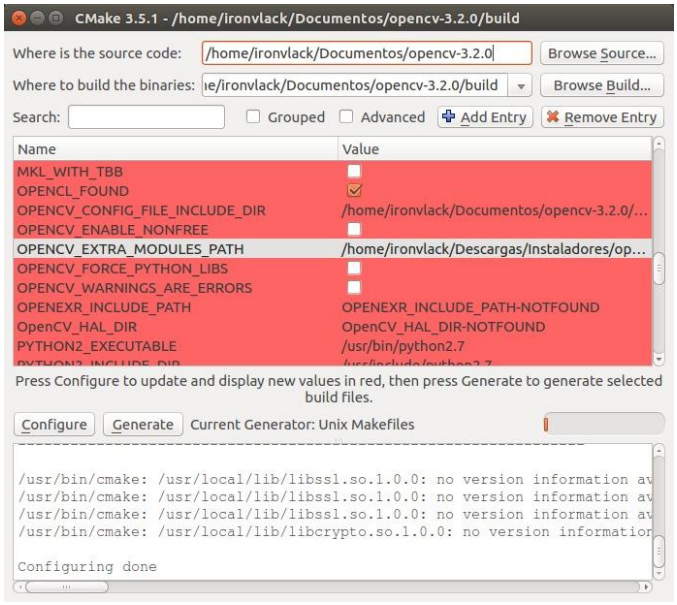


Figura 28 Agregación de los módulos OpenCV-Contrib

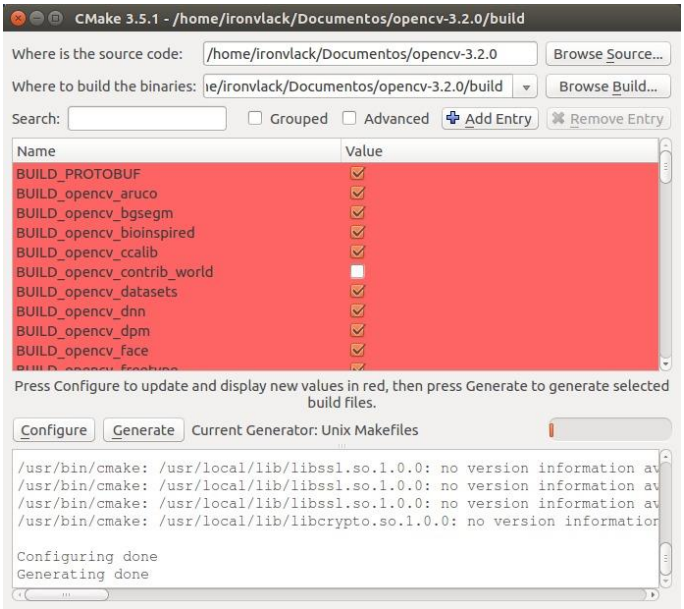


Figura 29 Imagen Final luego de compilar todos los módulos de OpenCV

6.2.3. Integración de OpenCV con QT.

Dentro del Framework QT y específicamente dentro de nuestro proyecto creado, observaremos que existe un archivo con extensión .pro en el cual se ingresan los paths o rutas hacia las librerías que utilizaremos.

Para el desarrollo de este proyecto se utilizaron todas las librerías de OpenCV y adicional el modulo “Text” que nos permitirá hacer el OCR.

El archivo .pro quedaría de la siguiente manera:

```
CONFIG += link_pkgconfig
PKGCONFIG += opencv

INCLUDEPATH += /usr/local/include/opencv

LIBS += -L/usr/local/lib
LIBS += -lopencv_core
LIBS += -lopencv_highgui
LIBS += -lopencv_features2d
LIBS += -lopencv_flann
LIBS += -lopencv_imgcodecs
LIBS += -lopencv_imgproc
LIBS += -lopencv_ml
LIBS += -lopencv_objdetect
LIBS += -lopencv_photo
LIBS += -lopencv_shape
LIBS += -lopencv_stitching
LIBS += -lopencv_superres
LIBS += -lopencv_video
LIBS += -lopencv_videoio
LIBS += -lopencv_videostab
LIBS += -lopencv_text

LIBS += -L$$PWD/../../../../usr/local/lib/ -ltesseract
LIBS += -L$$PWD/../../../../usr/local/lib/ -lcurl
LIBS += -L$$PWD/../../../../usr/local/lib/ -lcurlpp

INCLUDEPATH += $$PWD/../../../../usr/local/include
DEPENDPATH += $$PWD/../../../../usr/local/include

RESOURCES += \
    imagenes.qrc
```

Figura 30 Links en el archivo .pro de nuestro proyecto hacia las librerías compiladas e instaladas

Con esta configuración podremos hacer uso de todas las funciones que poseen las librerías dentro de nuestro proyecto, dentro de estas funciones se encuentra la captura y procesamiento de video, la clasificación de los objetos entre las más principales.

6.2.4. Clasificador HAAR Cascades.

Una de las funcionalidades que nos provee OpenCV es la carga de un archivo clasificador. Este archivo nos va a permitir la búsqueda de los objetos para lo cual lo hemos entrenado, se realizará el análisis dentro de la imagen que nosotros le enviemos en nuestro sistema.

OpenCV cuenta con un conjunto de herramientas que nos permite crear estos clasificadores, creado a base de un entrenamiento de imágenes negativas y positivas. El archivo en su estructura es un árbol, con extensión .xml lo que facilita su lectura dentro del programa que se implemente, permitiéndonos realizar la detección del objeto para lo cual se entrenó.

6.2.4.1. Creación del Clasificador.

Para la creación de este clasificador se utilizaron un conjunto de imágenes positivas y negativas, utilizando un total de 1200 imágenes positivas y 2000 imágenes negativas.

Deberemos organizar una carpeta de manera que estén clasificadas las imágenes respectivamente, para lo cual crearemos dos archivos en donde se encontrará toda la información de cada una de las imágenes clasificadas.

Crearemos un archivo denominado positivas.info el cual contendrá toda la información de las imágenes positivas, debemos realizar un listado dentro del archivo con todas las imágenes y con el nombre respectivo de cada una en formato JPEG.

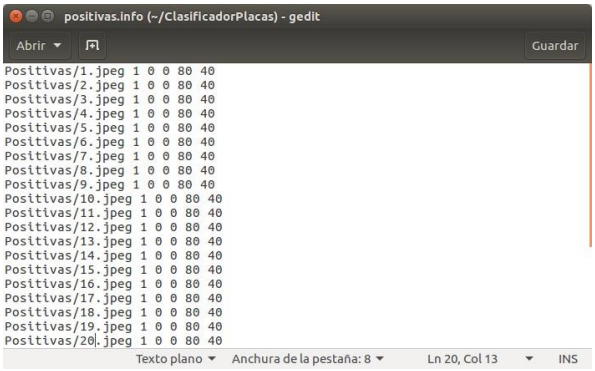


Figura 31 Estructura del archivo positivas.info

El segundo archivo que deberemos construir es el de imágenes negativas.txt el cual contendrá únicamente los nombres de las imágenes negativas, estos dos archivos son la base de la cual se construirá nuestro clasificador.

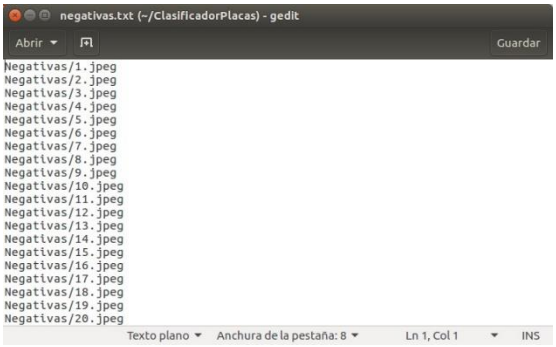


Figura 32 Estructura del archivo negativas.txt

Evitar esos falsos positivos es la razón por la que en el entrenamiento se deben brindar dos conjuntos de imágenes. Primer conjunto son los "ejemplos positivos" de la clase. El

segundo conjunto son los "ejemplos negativos", por ejemplo, el objeto placa con la condición no existe dentro de la imagen.

Una vez construido cada uno de los archivos utilizaremos la herramienta create_samples con la cual crearemos un archivo .vec que contiene la estructura de las imágenes y es a partir de esta que construiremos nuestro clasificador.

Para construir el archivo de vectores utilizaremos el comando como la indica la Figura 33. En el cual pasamos como parámetro el archivo de positivas y el nombre que le daremos al nuevo archivo.

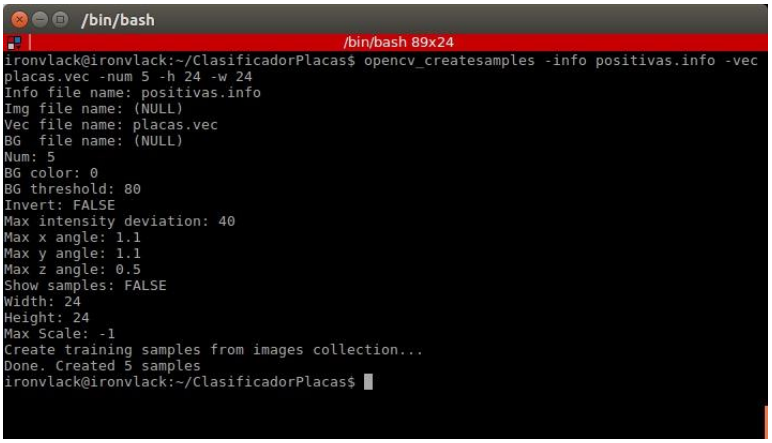


Figura 33 Creación del archivo de vectores

Una vez hemos construido el archivo de vectores pasaremos a crear nuestro clasificador, para lo cual utilizaremos la herramienta opencv_traincascade como lo muestra la figura 34.

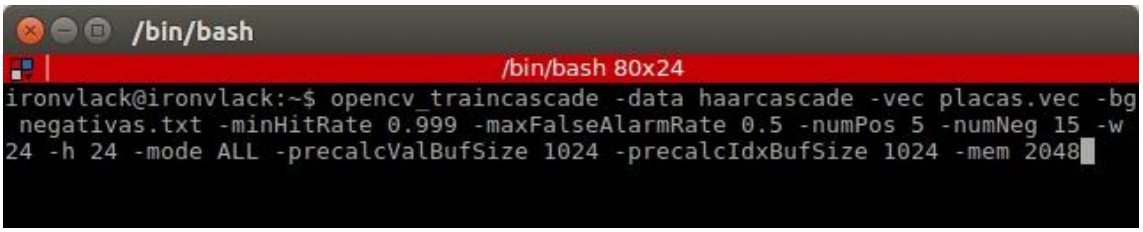


Figura 34 Comando de creación de clasificador haarcascade

Donde cada uno de los parámetros utilizados son:

- **-data:** Ubicación de salida donde se almacenaran los archivos generados
- **-vec:** Nombre del archivo de vectores creado anteriormente
- **-bg:** Lista de imágenes negativas
- **-minHitRate:** Tasa mínima de aciertos

- **-maxFalseAlarmRate:** Tasa máxima de falsa alarma
- **-numPos:** Numero de imágenes positivas
- **-numNeg:** Numero de imágenes negativas
- **-w –h:** Tamaño utilizado en la creación de las muestras
- **-mem:** Tamaño de la memoria asignada para el entrenamiento

Durante el entrenamiento veremos por salida de consola como se va realizando el proceso de entrenamiento por cada una de las imágenes como lo muestra Figura 35:

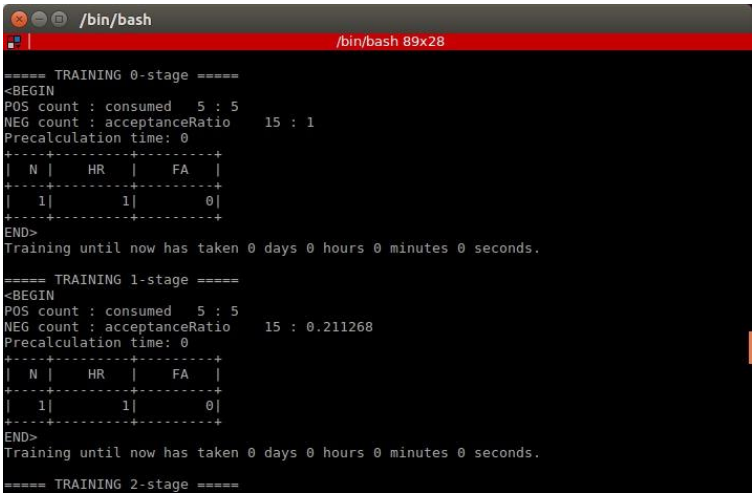


Figura 35 Proceso de entrenamiento del clasificador

Una vez terminado el entrenamiento tendremos como resultado un archivo llamado cascade.xml el cual será nuestro clasificador final que implementaremos en el desarrollo del sistema. La creación del archivo tuvo una duración aproximada de 2 días, tomando en cuenta que se utilizó una computadora con un procesador core i7 y una memoria de 6Gb. La carpeta final de salida haarcascade debe quedar como la Figura 36.

Es importante señalar que para la creación de un clasificador a más imágenes positivas el resultado será mejor, debido a que contendrá mayor cantidad de muestras y como salida se tendrá un clasificador mucho más robusto.

Figura 36 Estructura final de la carpeta y archivo cascade.xml

6.2.4.2. Implementación del clasificador.

Para la implementación del clasificador final obtenido, al momento de la inicialización de los objetos y de nuestro código, haremos la llamada a nuestro archivo y colocaremos la ruta donde se encuentra alojado. En este caso lo hemos renombrado y lo hemos llamado clasificadorPlacas.xml,

En la Figura 37 se muestra la implementación en el código del proyecto del clasificador ya entrenado.

```
66  void Principal::inicializar_objetos(){
67      this->progress->setValue(10);
68      inicializar_rol();
69      QThread::msleep(250);
70      setWindowIcon(QIcon(":/images/image/peaje.png"));
71      crear_directorio_imagenes();
72      this->progress->setValue(30);
73      QThread::msleep(250);
74      region_interes = cv::Rect(20, 280, 560, 180);
75      this->progress->setValue(40);
76      QThread::msleep(500);
77      this->progress->setValue(50);
78      conectar_camara();
79      cargar_imagen_label();
80      this->progress->setValue(60);
81      QThread::msleep(500);
82      this->progress->setValue(70);
83      this->lib_api = new Libanpr("/home/ironvlack/Documentos/Tesis/Clasificadores/clasificadorPlacas.xml",
84                                  "spa","ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789",7,8,7);
85      this->progress->setValue(80);
86      QThread::msleep(250);
87      this->progress->setValue(90);
88      QThread::msleep(250);
89      ui->btn_pause->setEnabled(false);
90      this->progress->setValue(100);
91  }
```

Figura 37 Implementación del clasificador haarcascades

Con la implementación de nuestro clasificador al momento de la detección y procesamiento de imágenes, nos devolverá a nivel de código un vector con todos los objetos que detecta para lo cual fue entrenado.

Con el vector resultante procederemos a realizar todos los procesos y operaciones necesarias para obtener de la imagen los caracteres que en él se encuentran.

6.3. Fase tres: Desarrollo del módulo para el cobro de peajes según el tipo de vehículo.

Para el core del sistema que nos permitirá realizar el OCR, se construyeron las clases y métodos de tal manera que sea una librería, es decir que cualquier sistema que la implemente podrá reutilizar y realizar el reconocimiento óptico de caracteres de una imagen, sin tener la necesidad de volver a programar los métodos que se utilizan.

Esta es la última fase de la metodología ICONIX, en la cual se realizó:

- Construcción de la arquitectura.
- Generación de código cuyo entregable es el proyecto codificado en C++ en QT y el proyecto codificado en Python en Django.

6.3.1. Arquitectura del Sistema.

Para comprender las decisiones arquitectónicas más relevantes tomadas a la hora de codificar el prototipo de peaje, se documenta en esta primera parte de la tercera fase: diagrama de componentes, diagrama de despliegue, tecnologías utilizadas y la estructura que tiene el proyecto.

6.3.1.1. Diagrama de componentes.

En la presente subsección se presenta una abstracción de la vista de desarrollo a través de un diagrama de componentes, que muestra la organización de cada uno de los elementos de diseño del prototipo de peaje en el sistema web. Como Django tiene una organización MVT “Model View Template – Modelo Vista Plantilla”, en el diagrama de componentes, se hace una abstracción de este patrón de diseño.

A continuación la representación de este patrón de diseño “MVT” en el proyecto.

- M significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

- T significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.

- V significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: se puede pensar en esto como un puente entre el modelo y las plantillas.

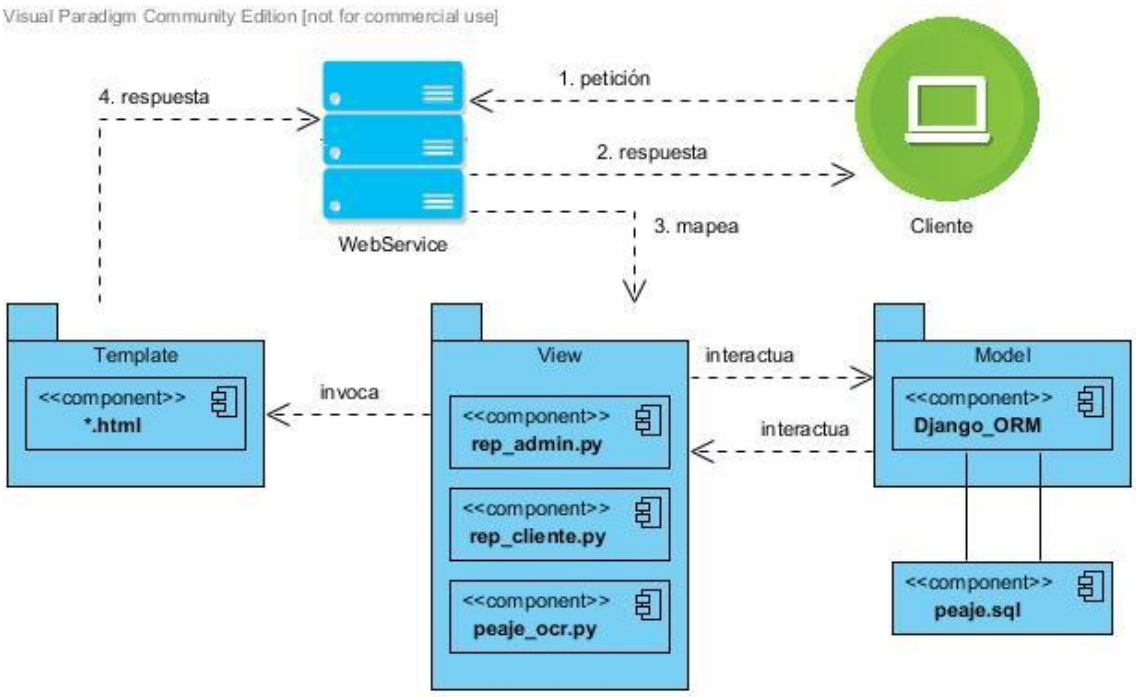


Figura 38 Diagrama de componentes del sistema

Template es la página HTML y el código que provee de datos dinámicos a la página, el Model es el Sistema de Gestión de Base de Datos y el View representa la Lógica de negocio.

6.3.1.2. Diagrama de despliege.

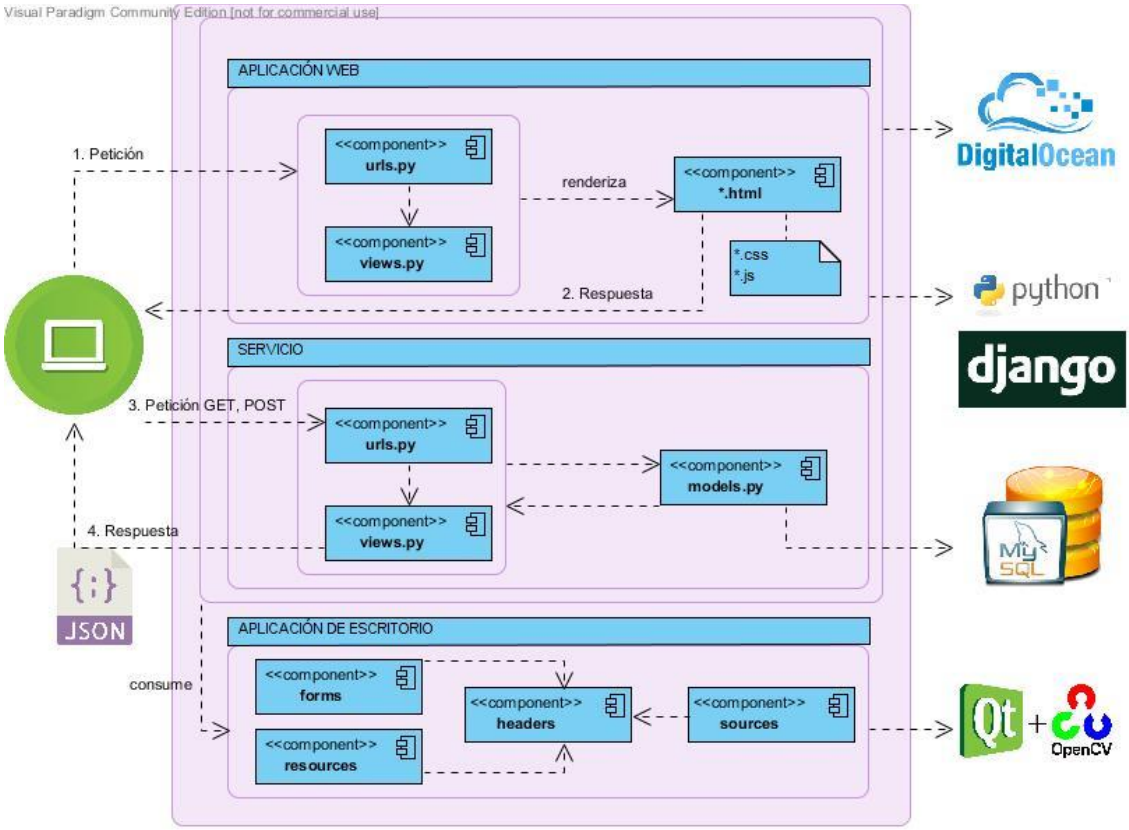


Figura 39 Diagrama de despliegue del sistema

El diseño de la arquitectura para el sistema de cobro de peajes, fue de manera general compuesto por un servidor de base de datos, un servidor web, una aplicación web y una aplicación de escritorio.

En el servidor de base de datos se encuentra la base de datos diseñada para poder gestionar y almacenar todos los cobros realizados por la aplicación de escritorio, la base de datos es gestionada a través de Mysql. La base de datos contiene toda la información sobre las tarifas que serán cobradas a los diferentes vehículos, además de todos los cobros realizados.

En el sistema web se encuentra replicada toda la información sobre los cobros, esto debido a que los usuarios de los peajes puedan consultar los cobros generados a través de cada estación, la comunicación se realizó a través del levantamiento de un web service que permite el ingreso de la información generada desde la aplicación de escritorio.

La aplicación de escritorio permitirá realizar todo el proceso de configuración y detección de las placas vehiculares, el cual luego de detectar una placa generará un ticket de cobro en el sistema. Para la obtención de los datos del vehículo a cobrar, se utilizó un servicio público de la ANT, el cual nos provee de los datos del vehículo, incluido la placa, el color y el tipo de vehículo, entre otros atributos, los cuales nos permiten generar el ticket de cobro el cual será almacenado y posteriormente enviado hacia el servidor de la aplicación web.

En la figura 40 se muestra el esquema general del funcionamiento, entre todos los componentes que forman el sistema.

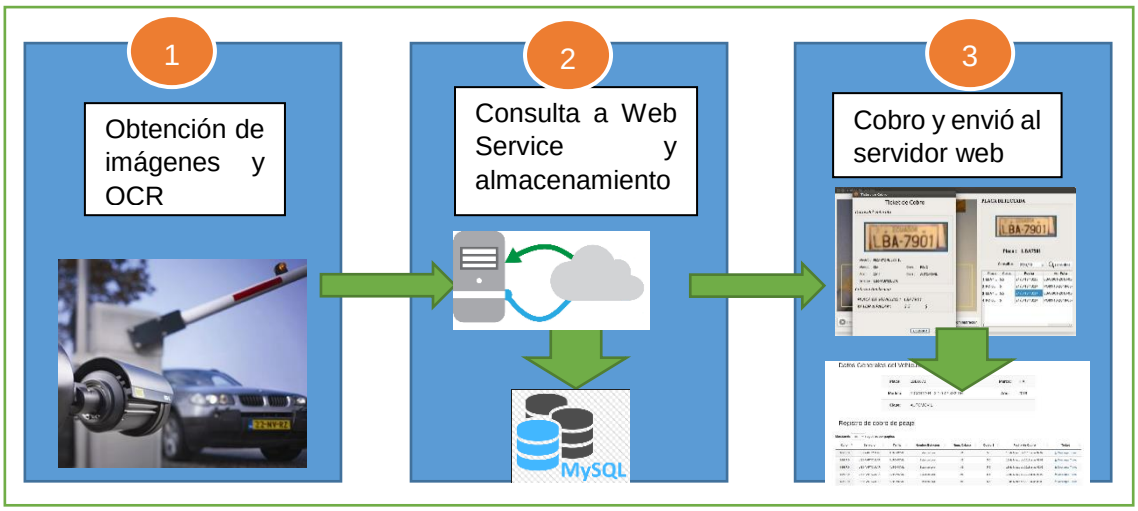


Figura 40 Arquitectura del sistema

1. **Obtención de los frames** a través de la cámara configurada, la cual nos permitirá detectar las placas.
2. **Consulta hacia el servicio de la ANT** para la obtención de los datos de la placa del vehiculó detectado.
3. **Realización del cobro y envió de datos hacia el servidor web** para que el cliente pueda realizar la revisión de su pago.

6.3.1.3. Tecnologías utilizadas.

En el desarrollo del proyecto se utilizaron las siguientes tecnologías:

Aplicación Web.

- Django Framework
- Django REST Framework

Aplicación de Escritorio.

- Qt Framework
- OpenCV

6.3.1.4. Estructura del proyecto.

El proyecto fue divido en una estructura de carpetas acorde a cada una de las funcionalidades y componentes:

Figura 41 Estructura del proyecto

Archivo de configuración:

Este archivo hace referencia a todas las dependencias y librerías que se utilizaron en el proyecto.

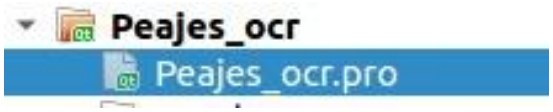


Figura 42 Archivo .pro de configuraciones del proyecto

Headers:

Los Headers contienen los nombres de las clases, las variables y métodos que se utilizaron en el proyecto.

Figura 43 Sección de Headers

Sources:

Aquí se encuentran todos los métodos y el código fuente de todos los métodos declarados anteriormente en Headers.

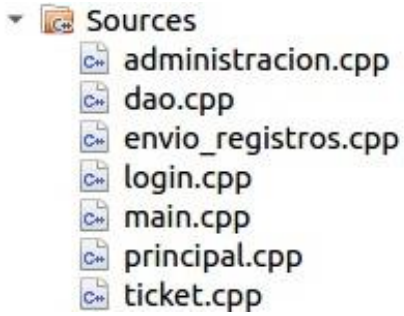


Figura 44 Sección de Sources

Forms:

En esta sección se encuentran las pantallas que se realizaron en el diseño.

Figura 45 Sección de Forms

Resources:

En esta sección se encuentran todos los archivos adicionales que se utilizaron en el proyecto, como son imágenes e iconos.

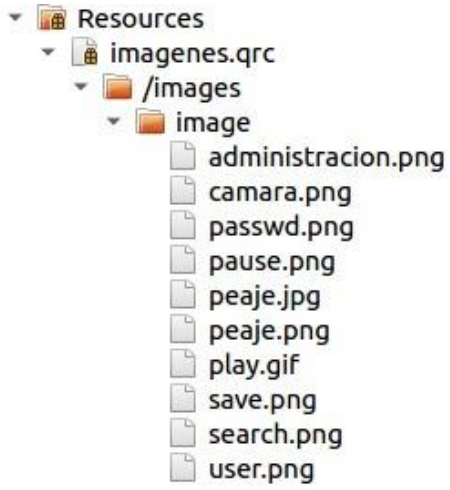


Figura 46 Sección Resources

6.3.2. Generación de Código.

Para esta etapa se utilizó como estándares de programación: QGis para la programación del software de escritorio en QT y PEP 8 para la programación del sistema web en Django; los cuales nos permitieron la escritura de un código funcional, limpio, fácil de mantener y entender al momento del desarrollo del sistema.

En la implementación del reconocimiento de placas se crearon varios métodos que nos permitieron realizar la detección en una imagen, y otro método adicional que nos permite realizar el OCR de la imagen que le pasamos como parámetro.

La estructura de los métodos y clases se los organizó de tal manera que se pudo construir una librería, la que posteriormente se utilizará en la interfaz del cliente.

La librería realiza la detección de texto en imágenes válidas de placas, se utilizó el lenguaje de programación C++ conjuntamente con la librería OpenCV para el procesamiento de imágenes y la detección del objeto deseado, en este caso placas. Para la extracción de las placas válidas se utilizó el clasificador construido previamente clasificadorPlacas.xml usando el algoritmo LBPH.

El tamaño de las imágenes para ser procesadas dependerá de la resolución de la cámara que se utilice, hay que considerar que para evitar la búsqueda en una zona muy grande de la imagen se determinó un ROI en la captura de las imágenes, lo que nos permitirá identificar los objetos en una zona específica la placa de los vehículos.

Los procesos de reconocimiento de placas y OCR se describen a continuación.

6.3.2.1. Obtención de Imágenes Válidas.

Usando el clasificador Haar Cascades clasificadorPlacas.xml, se realizó la extracción de imágenes de placas dentro de los frames capturados por la cámara. Considerando la posición de las placas en un vehículo, se optimizó la parte inferior de los frames como una zona de interés (ROI) para realizar una búsqueda rápida y eficiente.

El clasificador analiza por cada frame de video o imagen buscando el objeto para el cual fue entrenado, en este caso utilizaremos el método creado llamado process como se muestra en la Figura 47, este método está construido de tal manera que nos devuelve un vector con las placas detectadas; y por cada placa detectada un objeto placa que contiene los atributos, matriz de la imagen, porcentaje de confiabilidad, y el OCR de cada placa detectada en la imagen.

En el sistema se consideró la primera placa obtenida para su procesamiento, tomando en cuenta que por cada carril de peaje se tendrá un vehículo con su respectiva placa para su respectivo cobro.

```
116 void Principal::obtener_ocr_placa()
117 {
118     Placa placa_detectada;
119     QString placa;
120     cv::rectangle(matrizOriginal, region_interes, Scalar(232, 221, 76), 2, 2);
121     matrizRoi = matrizOriginal(region_interes);
122     lib_api->process(matrizRoi);
123     vector<Placa> placas = lib_api->getPlacas();
124     if(!placas.empty()){
125         placa_detectada = placas.at(0);
126         if(placa_detectada.isRecognize()){
127             placa = placa_detectada.getPlacaOcr();
128             cv::rectangle(matrizRoi, placa_detectada.rect, Scalar(43, 97, 128), 2, 1);
129             if(!placa.contains(placa_anterior.mid(0, 3))){
130                 if (placa_detectada.getPlacaOcr().size() == 6 ){
131                     placa = placa.mid(0,3).append("0").append(placa.mid(3,6));
132                 }
133                 mostrar_placa_detectada(placa_detectada);
134                 ui->label_placa->setText(placa);
135                 generar_ticket_cobro(placa, placa_detectada);
136             }
137             placa_anterior = placa_detectada.getPlacaOcr();
138         }
139     }
140 }
141
```

Figura 47 Método principal utilizado para realizar todo el proceso de cobro

En la primera parte del método process se realiza el tratamiento de la imagen, su conversión a escala de grises y la obtención del umbral adaptativo. Una vez realizado este proceso se obtiene un vector con los puntos del contorno y se realiza una búsqueda de todos los bordes, dándonos como resultado una imagen con todos los caracteres contenidos en blanco y negro; además de todas las secciones con los caracteres para su respectiva identificación.

```
82  void Ocr::process(){
83      resize(placa,placa,Size(200,75));
84      //imshow("color",placa);
85      vector<vector<ERStat> > regions(2);
86      cv::cvtColor(placa,grey,COLOR_RGB2GRAY);
87      adaptiveThreshold(grey, grey,255,CV_ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY,11,2);
88      bitwise_not(grey, grey);
89      Mat contorno;
90      grey.copyTo(contorno);
91      vector<vector<Point> > contours;
92      vector<Vec4i> hierarchy;
93      findContours( contorno, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);
94      vector<vector<Point> > contours_poly( contours.size() );
95      vector<Rect> boundRect( contours.size() );
96      Mat mask = Mat::zeros(contorno.rows, contorno.cols, CV_8UC1);
97      for( int i = 0; i < contours.size(); i++ ){
98          if(contourArea(contours[i],false)>10){
99              approxPolyDP( Mat(contours[i]), contours_poly[i], 3, true );
100             boundRect[i] = boundingRect( Mat(contours_poly[i]) );
101             if(boundRect[i].width>3&&boundRect[i].width<30 &&boundRect[i].height>27&&boundRect[i].height<50){
102                 Mat img_roi=grey(boundRect[i]);
103                 Mat roi(mask,boundRect[i]);
104                 img_roi.copyTo(roi);
105             }
106         }
107        }
108     bitwise_not(mask, mask);
109     mask.copyTo(grey);
```

Figura 48 Tratamiento de la imagen

Este tratamiento de la imagen se realiza previo al análisis de cada sección de los posibles caracteres, a continuación se describe la segunda parte del proceso.

6.3.2.2. Reconocimiento Óptico de Caracteres – OCR.

En la segunda parte del método se realizó el seccionamiento de los posibles caracteres contenidos en la imagen, se recorre cada una de las secciones obteniendo el mejor resultado de cada uno. A partir de este proceso se obtiene una cadena de caracteres del texto contenido en la imagen.

Internamente el modulo Text de OpenCV utiliza los métodos y procesos de las librerías instaladas en las secciones anteriores como son Leptónica y Tesseract, como pudimos darnos cuenta en el api de este módulo, todas las funciones y métodos los utiliza en sus propios métodos creando su propio listado de funciones que nos permiten llegar a un mejor resultado.

```
for (int i=0; i<(int)detections.size(); i++){
    outputs[i].erase(remove(outputs[i].begin(), outputs[i].end(), '\n'), outputs[i].end());
    if (outputs[i].size() < 3)
        continue;
  for (int j=0; j<(int)boxes[i].size(); j++){
        pplaca=words[i][j];
        if((pplaca.size()>2)&&(pplaca.size()<5)){
            if(confidences[i][j]>60){
              //  cout<<confidences[i][j]<<"____"<<pplaca<<"__||__";
                ocr=ocr+pplaca;
                cont++;
                if(this->confidence==0.0)
                    this->confidence=confidences[i][j];
                else
                    this->confidence=(this->confidence+confidences[i][j])/2;
            }
        }
  }
  if(cont==2){
     string placa_correct=corregirPlaca(ocr);
    if((placa_correct!="")&&(!isRepetitive(placa_correct))){
        this->is_sussceful=true;
        this->placa_ocr=placa_correct.c_str();
    }else{
        this->placa_ocr="";
        this->is_sussceful=false;
        this->placa.release();
        this->confidence=0.0;
    }
  }
```

Figura 49 Segunda parte del método, obtención de la cadena de texto

Como se pudo apreciar en la Figura 49 se tienen las funciones del módulo, en donde nos permite extraer cada uno de los caracteres y formar nuestro texto para luego ser utilizado respectivamente. Finalmente luego de haber realizado el OCR se realiza una validación de la placa obtenida, para poder controlar a nivel de programación posibles errores en la obtención de la placa vehicular.

6.3.2.3. Consulta de datos y generación de cobro.

Una vez realizado el procesamiento y obtenido el texto de la placa detectada, el siguiente proceso es realizar la consulta para poder obtener los datos del vehículo al que pertenece la placa.

La ANT posee un servicio público de consulta de infracciones, por medio de placas y/o cédula de los propietarios de los vehículos. Es un servicio al cual se puede acceder mediante un portal web, este portal al momento de realizar las consultas nos devuelve todos los datos del vehículo con la placa consultada.

El link para el acceso es el siguiente:

https://sistemaunico.ant.gob.ec:5038/PortalWEB/paginas/clientes/clp_criterio_consulta.jsp o en la siguiente url corta: https://goo.gl/C2XksM .

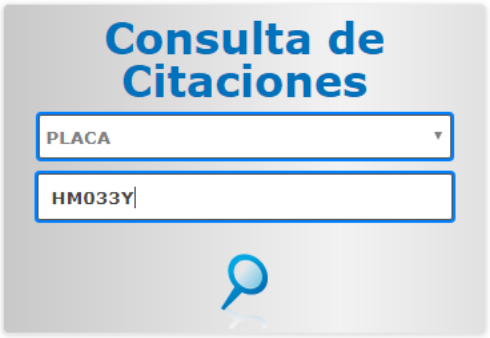


Figura 50 Pantalla de inicio del sistema de la ANT

Una vez dentro del portal seleccionaremos el criterio de consulta placa e ingresaremos la placa de nuestro vehículo, y nos desplegará una pantalla similar a la siguiente:

Figura 51 Despliegue de datos en el portal web de consultas de la ANT

En este servicio solamente nos interesa obtener los primeros datos mostrados como son: marca, modelo, año, color, clase y servicio del vehiculó del cual estamos consultado.

Para la obtención de datos según nuestro criterio y placa del vehículo que vamos a consultar se procedió a construir a nivel de código una url específica por cada consulta que realicemos de la siguiente manera:

Donde tendremos:

- **url:** Link del servicio que utilizaremos
- **placa:** datos de la placa vehicular a consultar

Cada vez que realicemos la consulta al servicio crearemos la url del servicio con la placa que deseamos consultar: **url + placa.**

En el código realizaremos el tratamiento de los datos obtenidos, para solo obtener los datos que son de nuestro interés, el método que se utiliza para la formación de la url y la obtención de los datos se lo muestra en la siguiente figura:

```
void ticket::consultar_datos_web_service(QString placa)
{
    try {
        ui->progressBarTicket->setValue(10);
        std::string s = placa.toStdString();
        ui->progressBarTicket->setValue(20);
        std::string url = "https://sistemaunico.ant.gob.ec:5038/PortalWEB/paginas/clientes/clp_grid_citaciones.jsp?"
                          "ps_tipo_identificacion=PLA&ps_identificacion="+s+"&ps_placa=";
        curlpp::Cleanup clean;
        ui->progressBarTicket->setValue(30);
        curlpp::Easy r;
        r.setOpt(new curlpp::options::Url(url));
        ui->progressBarTicket->setValue(50);
        r.setOpt(new curlpp::options::Timeout(7));
        r.setOpt(new curlpp::options::SslVerifyPeer(FALSE));
        ui->progressBarTicket->setValue(60);
        std::ostringstream response;
        r.setOpt(new curlpp::options::WriteStream(&response));
        ui->progressBarTicket->setValue(70);
        r.perform();

        ui->progressBarTicket->setValue(80);
        QString respuesta_web_service_aux = QString::fromStdString(std::string(response.str())).toUtf8();
        ui->progressBarTicket->setValue(100);
        respuesta_web_service = parsear_datos(respuesta_web_service_aux);
    }catch ( curlpp::LogicError &e ) {
        qDebug()<< e.what() ;
        respuesta_web_service = QString("-2");
    }
    catch ( curlpp::RuntimeError &e ) {
        qDebug()<< e.what() ;
        respuesta_web_service = QString("-2");
    }
}
```

Figura 52 Método para la obtención de datos del vehículo

Con los datos del vehículo que hemos consultado, se generará un cobro en base a las tarifas ingresadas en el sistema (ver Figura 54). Cada ticket de cobro será un Diálogo que contendrá los datos específicos del vehículo, la tarifa y el valor que se cobrará. Por cada placa vehicular detectada se generará un cobro, cada uno se ira colocando secuencialmente en la parte izquierda y en la parte derecha de la pantalla respectivamente.

```
void Principal::generar_ticket_cobro(QString placa, Placa placa_data)
{
        if (posicion_ticket == 1){
            posicion_ticket = 2;
        }else{
            posicion_ticket = 1;
        }
        qDebug()<< id_estacion;
        ticket *t =new ticket(placa, placa_data, id_usuario_prin, id_estacion, posicion_ticket);
        connect(t, SIGNAL(recargar_cobros()), this, SLOT(refrescar_tickets()));
}
```

Figura 53 Método para generar un ticket de cobro

Cuando el cobro es finalizado se almacenara el registro en la base de datos local y posteriormente será enviada al servidor web para que el cliente al que se le realizó el cobro pueda hacer las consultas en línea.

Figura 54 Ticket de cobro

El sistema realiza varios reconocimientos de placas vehiculares se haya o no realizado el cobro de peaje, es decir, el sistema debe contemplar la posibilidad que hayan varios tickets de peaje por cobrar sin que esto detenga el funcionamiento de la cabina de peaje.

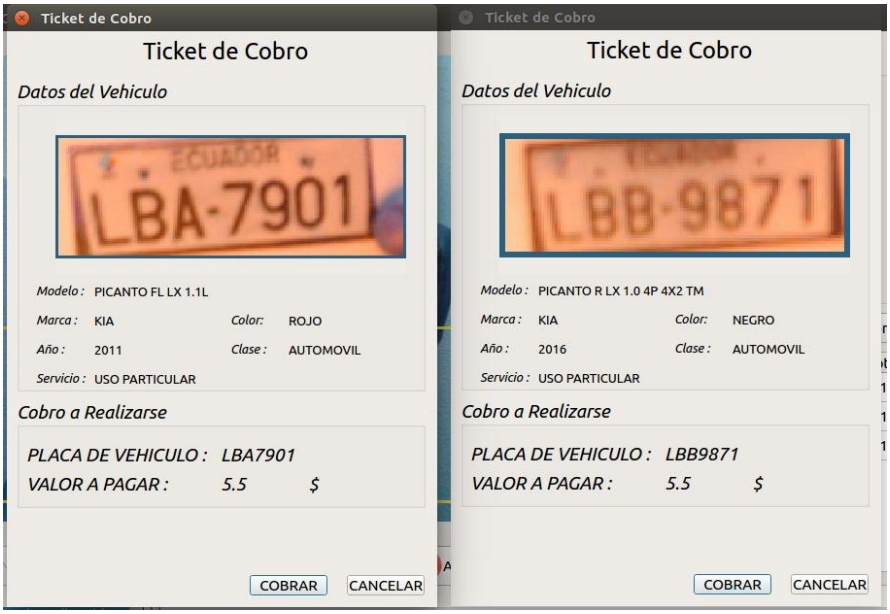


Figura 55 Distribución por acumulación de varios Ticket del cobro por peajes

6.4. Fase cuatro: Pruebas de funcionalidad en tiempo real del prototipo.

Esta etapa no está contemplada por la metodología ICONIX, pero es el cierre del cuarto objetivo del presente proyecto de trabajo de titulación. Las pruebas validan todas las fases anteriores justo después de la codificación del prototipo de peaje y cuando este sistema se encuentra ya alojado en un servicio de hosting como lo es DigitalOcean.

No hay una metodología específica para realizar las pruebas, por lo cual, para la ejecución de las mismas se ha tomado como referente el conocimiento aceptado y validado por la comunidad científica; es decir, esta sección será realizada según las directrices obtenidas en revisiones bibliográficas.

En esta fase se realizaron múltiples pruebas individuales como en conjunto de todo el sistema y sus componentes software, con la finalidad de encontrar vulnerabilidades de acuerdo a su ejecución y utilización.

6.4.1. Pruebas de caja blanca.

El código generado para la implementación de todos los métodos, declaración de variables y clases en el sistema fue realizado siguiendo los estándares mencionados en Generación de Código. Esto nos permitió obtener un código limpio, modular y reusable que a su vez será fácil de mantener e interpretar por personas que sean ajenas al proyecto y que deseen contribuir al mejoramiento del mismo.

Para la generación de pruebas de caja blanca se utilizó la librería presente en el Framework de desarrollo Qt llamada QtTest conjuntamente con la librería Unittest++.

QtTest: Es una librería integrada y presente en el Framework Qt que nos proporciona todas las clases necesarias para la realización de pruebas unitarias de aplicaciones y bibliotecas del mismo framework, así como también extensiones para probar interfaces gráficas de usuario. Esta librería se encuentra presente en todas las versiones del framework, ya que fue diseñado para facilitar la escritura de pruebas unitarias para aplicaciones y bibliotecas basadas en Qt.

Para su utilización es muy sencilla solo bastara agregar la directiva en la clase que se desee utilizar como lo muestra la Figura 56.

```
#include < QtTest >
```

Figura 56 Inclusión de las librerías en una clase

Y para poder incluir las librerías respectivas en el proyecto deberemos agregarla al archivo de configuración de nuestro proyecto tal como lo muestra la siguiente figura:

```
QT + = testlib
```

Figura 57 Inclusión de QtTest en nuestro proyecto

Unittest++: Es un marco liviano para realizar pruebas unitarias para C ++. Fue diseñado para hacer un desarrollo basado en pruebas en una amplia variedad de plataformas. La simplicidad, la portabilidad, la velocidad son todos aspectos muy importantes. Unittest++ es en su mayoría un estándar de C ++ y hace un uso mínimo de las características avanzadas de la biblioteca y el lenguaje, lo que significa que debe ser fácilmente portátil para casi cualquier plataforma por lo que se encuentra disponible para: Windows, Linux y Mac OS.

Su instalación es muy sencilla lo que nos facilita el fácil uso en cualquier clase C++ que se tenga implementada en cualquier proyecto, para su instalación se lo realizó desde los repositorios oficiales de Ubuntu con el comando:

apt-get install unittest++-dev.

A continuación se presenta la implementación y ejecución de las pruebas unitarias, en diferentes casos de prueba, de manera que se verifica que el software soporta entradas e ingreso de datos erróneos garantizando la calidad y el rendimiento del mismo.

6.4.1.1. Pruebas unitarias

Se realizaron pruebas unitarias únicamente en los métodos principales de las diferentes clases utilizadas, que dan la funcionalidad al sistema y que retornan valores que pueden ser predichos y posteriormente ser comparados respectivamente.

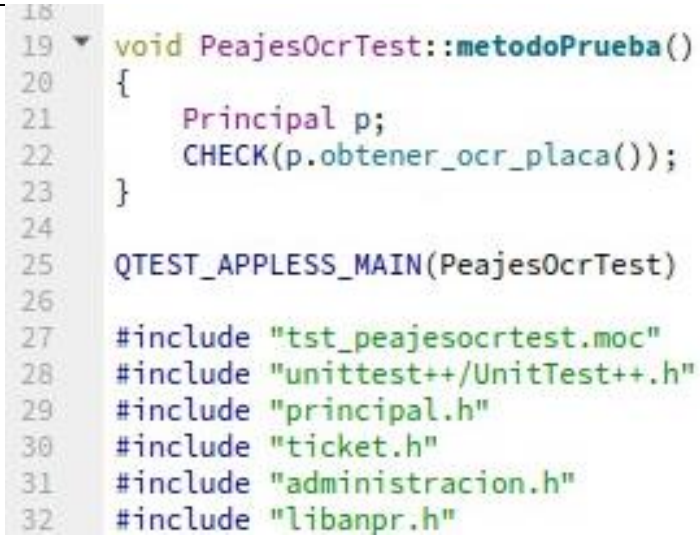
Clase Principal – Método obtener_ocr_placa

Este método es el que organiza y genera todas las operaciones dentro del sistema, es el método principal de la clase, está construido de manera que se organizan todos los métodos y se los va llamando a medida que se realiza el proceso.

Cada uno de los métodos que se llaman, están controlados de manera que si se tienen valores en blanco o nulos el sistema siga trabajando sin que se presenten excepciones o cortes en el flujo de trabajo.

Considerando los puntos mencionados anteriormente se realizaron las respectivas pruebas, utilizando diferentes muestras y quedando como resultados emitidos de la siguiente manera (ver Tabla 28).

Tabla 28 Implementación de pruebas unitarias en método obtener_ocr_placa

Implementación de pruebas unitarias **Método Principal**


```
19  void PeajesOcrTest::metodoPrueba()
20  {
21      Principal p;
22      CHECK(p.obtener_ocr_placa());
23  }
24
25  QTEST_APPLESS_MAIN(PeajesOcrTest)
26
27  #include "tst_peajesocrtest.moc"
28  #include "unittest++/UnitTest++.h"
29  #include "principal.h"
30  #include "ticket.h"
31  #include "administracion.h"
32  #include "libanpr.h"
```

Método: obtener_placa_ocr	
Resultado	

```
void Principal::obtener_ocr_placa()
{
    Placa placa_detectada;
    QString placa;
    cv::rectangle(matrizOriginal, region_interes, Scalar(232, 221, 76), 2, 2);
    matrizRoi = matrizOriginal(region_interes);
    lib_api->process(matrizRoi);
    vector<Placa> placas = lib_api->getPlacas();
    if(!placas.empty()){
        placa_detectada = placas.at(0);
        if(placa_detectada.isRecognize()){
            placa = placa_detectada.getPlacaOcr();
            qDebug()<< "Placa detectada " << placa;
            cv::rectangle(matrizRoi, placa_detectada.rect, Scalar(43, 97, 128), 2, 1);
            if(!placa.contains(placa_anterior.mid(0, 3))){
                if (placa_detectada.getPlacaOcr().size() == 6 ){
                    placa = placa.mid(0,3).append("0").append(placa.mid(3,6));
                }
                mostrar_placa_detectada(placa_detectada);
                ui->label_placa->setText(placa);
                generar_ticket_cobro(placa, placa_detectada);
            }
            placa_anterior = placa_detectada.getPlacaOcr();
        }
    }
}
```

```
PeajesOcr ✖
Starting /home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest...
******** Start testing of PeajesOcrTest ********
Config: Using QtTest library 5.9.1, Qt 5.9.1 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC 5.3.1
5.3.1-6))
PASS   : PeajesOcrTest::initTestCase()
PASS   : PeajesOcrTest::metodoPrueba()
PASS   : PeajesOcrTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
******** Finished testing of PeajesOcrTest ********
/home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest exited with code 0
```

Clase Principal – Método mostrar_placa_detectada

Este método es el que se encarga de mostrar y tratar la matriz de la placa detectada, convirtiéndola a una imagen y al tamaño determinado para ser mostrada en la interfaz de usuario.

Una vez convertida la imagen es representada en pixeles para ser mostrada en un widget de la interfaz.

Considerando los puntos mencionados anteriormente se realizaron las respectivas pruebas, utilizando diferentes muestras y quedando como resultados emitidos de la siguiente manera (ver Tabla 29).

Tabla 29 Implementación de pruebas unitarias en método mostrar placa_detectada

<table>
<tr><th colspan="2">Implementación de pruebas unitarias
Método mostrar placa detectada</th></tr>
<tr><td colspan="2">

```
void PeajesOcrTest::metodoPrueba()
{
    Principal p;
    CHECK(p.mostrar_placa_detectada(Placa *placa));
}

QTEST_APPLESS_MAIN(PeajesOcrTest)

#include "tst_peajesocrtest.moc"
#include "unittest++/UnitTest++.h"
#include "principal.h"
#include "ticket.h"
#include "placa.h"
#include "administracion.h"
#include "libanpr.h"
```

</td></tr>
<tr><td>Método:
mostrar_placa_
detectada</td><td>

```
void Principal::mostrar_placa_detectada(Placa placa)
{
    QImage imagenTemporal((uchar*)placa.placa.data, placa.placa.cols, placa.placa.rows,
                          placa.placa.step, QImage::Format_RGB888);
    imagenTemporal = imagenTemporal.scaled(261, 161, Qt::KeepAspectRatio);
    ui->labelPlaca->setPixmap(QPixmap::fromImage(imagenTemporal));
}
```

</td></tr>
<tr><td>Resultado</td><td>

```
PeajesOcr
Starting /home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest...
********* Start testing of PeajesOcrTest *********
Config: Using QtTest library 5.9.1, Qt 5.9.1 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC 5.3.1
5.3.1-6))
PASS   : PeajesOcrTest::initTestCase()
PASS   : PeajesOcrTest::metodoPrueba()
PASS   : PeajesOcrTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
********* Finished testing of PeajesOcrTest *********
/home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest exited with code 0
```

</td></tr>
</table>

Clase Principal – Método generar_ticket_cobro

Este método se encarga de generar los tickets de cobro, se debe manejar cuidadosamente cada generación ya que este se almacena en un puntero de memoria para cada ticket, el manejo inadecuado generara un colapso de memoria, como parámetros de entrada tendrán una cadena de la placa reconocida y un objeto placa que contiene la información sobre la placa detectada.

Considerando los puntos mencionados anteriormente se realizaron las respectivas pruebas, utilizando diferentes muestras y quedando como resultados emitidos de la siguiente manera (ver Tabla 30).

Tabla 30 Implementación de pruebas unitarias en método generar ticket de cobro

Implementación de pruebas unitarias
Método generar ticket de cobro

```
void PeajesOcrTest::metodoPrueba()
{
    Principal p;
    CHECK(p.generar_ticket_cobro(QString *placa, Placa *placa));
}

QTEST_APPLESS_MAIN(PeajesOcrTest)

#include "tst_peajesocrtest.moc"
#include "unittest++/UnitTest++.h"
#include "principal.h"
#include "ticket.h"
#include "placa.h"
#include "administracion.h"
#include "libanpr.h"
```

Método: generar_ticket_cobro

```
void Principal::generar_ticket_cobro(QString placa, Placa placa_data)
{
        if (posicion_ticket == 1){
            posicion_ticket = 2;
        }else{
            posicion_ticket = 1;
        }
        qDebug()<< id_estacion;
        ticket *t =new ticket(placa, placa_data, id_usuario_prin, id_estacion, posicion_ticket);
        connect(t, SIGNAL(recargar_cobros()), this, SLOT(refrescar_tickets()));
}
```

Resultado

```
PeajesOcr
Starting /home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest...
********* Start testing of PeajesOcrTest *********
Config: Using QtTest library 5.9.1, Qt 5.9.1 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC 5.3.1
5.3.1-6))
PASS   : PeajesOcrTest::initTestCase()
PASS   : PeajesOcrTest::metodoPrueba()
PASS   : PeajesOcrTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
********* Finished testing of PeajesOcrTest *********
/home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest exited with code 0
```

Clase Ticket – Método consultar_datos_webservice

Este método se encarga de realizar las consultas al webservice de la ANT, este método recibe como parámetro de entrada una cadena, los datos que ingresen no deben ser valores en blanco ni nulos, en caso de que el servicio no tenga resultados de la búsqueda devolverá un valor de -2, caso contrario devolverá la respuesta del servicio.

Considerando los puntos mencionados anteriormente se realizaron las respectivas pruebas, utilizando diferentes muestras y quedando como resultados emitidos de la siguiente manera (ver Tabla 31).

Tabla 31 Implementación de pruebas unitarias en método consultar datos en webservice

Implementación de pruebas unitarias
Método consultar datos webservice

```
void PeajesOcrTest::metodoPrueba()
{
    ticket t;
    CHECK(t.consultar_datos_web_service(QString *placa));
}

QTEST_APPLESS_MAIN(PeajesOcrTest)

#include "tst_peajesocrtest.moc"
#include "unittest++/UnitTest++.h"
#include "principal.h"
#include "ticket.h"
#include "placa.h"
#include "administracion.h"
#include "libanpr.h"
#include "QModelIndex"
```

Método:
consultar_datos
_webservice

```
void ticket::consultar_datos_web_service(QString placa)
{
    try {
        ui->progressBarTicket->setValue(10);
        std::string s = placa.toStdString();
        ui->progressBarTicket->setValue(20);
        std::string url = "https://sistemaunico.ant.gob.ec:5038/PortalWEB/paginas/clientes/clp_grid_citaciones.jsp?"
                          "ps_tipo_identificacion=PLA&ps_identificacion="+s+"&ps_placa=";
        curlpp::Cleanup clean;
        ui->progressBarTicket->setValue(30);
        curlpp::Easy r;
        r.setOpt(new curlpp::options::Url(url));
        ui->progressBarTicket->setValue(50);
        r.setOpt(new curlpp::options::Timeout(7));
        r.setOpt(new curlpp::options::SslVerifyPeer(FALSE));
        ui->progressBarTicket->setValue(60);
        std::ostringstream response;
        r.setOpt(new curlpp::options::WriteStream(&response));
        ui->progressBarTicket->setValue(70);
        r.perform();

        ui->progressBarTicket->setValue(80);
        QString respuesta_web_service_aux = QString::fromStdString(std::string(response.str())).toUtf8();
        ui->progressBarTicket->setValue(100);
        respuesta_web_service = parsear_datos(respuesta_web_service_aux);
    }catch ( curlpp::LogicError &e ) {
        qDebug()<< e.what() ;
        respuesta_web_service = QString("-2");
    }
    catch ( curlpp::RuntimeError &e ) {
        qDebug()<< e.what() ;
        respuesta_web_service = QString("-2");
    }
}
```

Resultado

```
PeajesOcr
Starting /home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest...
********* Start testing of PeajesOcrTest *********
Config: Using QtTest library 5.9.1, Qt 5.9.1 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC 5.3.1
5.3.1-6))
PASS   : PeajesOcrTest::initTestCase()
PASS   : PeajesOcrTest::metodoPrueba()
PASS   : PeajesOcrTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
********* Finished testing of PeajesOcrTest *********
/home/ironvlack/QtProjects/build-PeajesOcr-Desktop_Qt_5_9_1_GCC_64bit-Debug/tst_peajesocrtest exited with code 0
```

Una vez finalizadas todas las pruebas y acorde a los resultados obtenidos en cada uno de ellos, se puede decir que los métodos principales para que el sistema funcione correctamente están debidamente implementados.

6.4.2. Pruebas de carga.

Para la realización de las pruebas de carga en el sistema, se realizó un monitoreo a los recursos del sistema, tanto en uso de memoria como en procesamiento, durante la ejecución de sus métodos más importantes. A continuación se muestra los resultados obtenidos (ver Tabla 32, 33, 34, 35).

Tabla 32 Estado de la memoria al iniciar sesión

Estado de la memoria al iniciar sesión

Descripción **total:** memoria total del pc. **used:** cantidad de memoria que está siendo usada en ese momento. **free:** memoria libre y disponible. **shared:** memoria compartida. **buff/cache:** suma de memoria intermedios y caché. **available:** cantidad de memoria disponible para iniciar nuevos procesos. **swap:** memoria asignada al sistema operativo.
Conclusión Al iniciar sesión en el sistema se aprecia un uso del 15% de la memoria del computador.

Tabla 33 Estado de la memoria al ejecutar el sistema

Estado de la memoria al ejecutar el módulo de cobro

Descripción **total:** memoria total del pc. **used:** cantidad de memoria que está siendo usada en ese momento. **free:** memoria libre y disponible. **shared:** memoria compartida. **buff/cache:** suma de memoria intermedios y caché. **available:** cantidad de memoria disponible para iniciar nuevos procesos. **swap:** memoria asignada al sistema operativo.
Conclusión Al ejecutar el sistema e iniciar el módulo de cobro se considera un aumento de memoria al 35% lo que representa un aumento del 20% de consumo.

Tabla 34 Estado del CPU al ejecutar el módulo de cobros

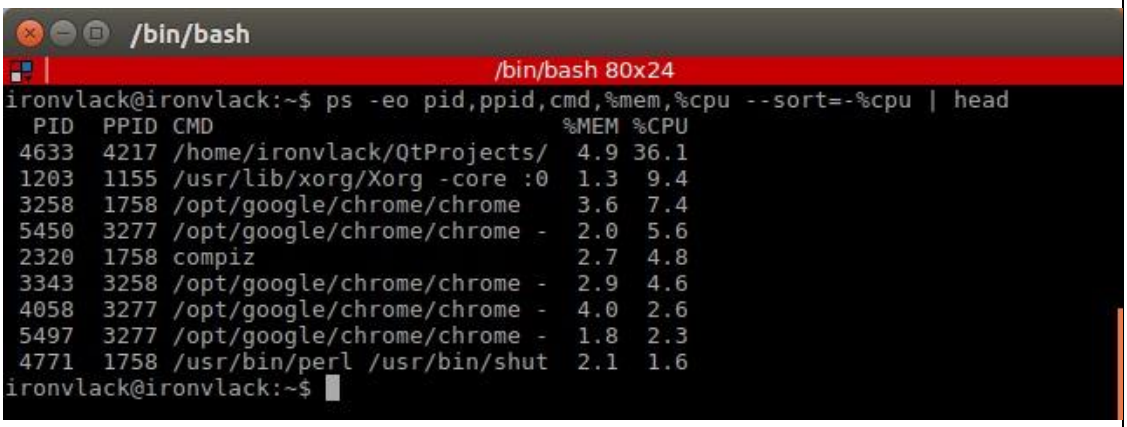
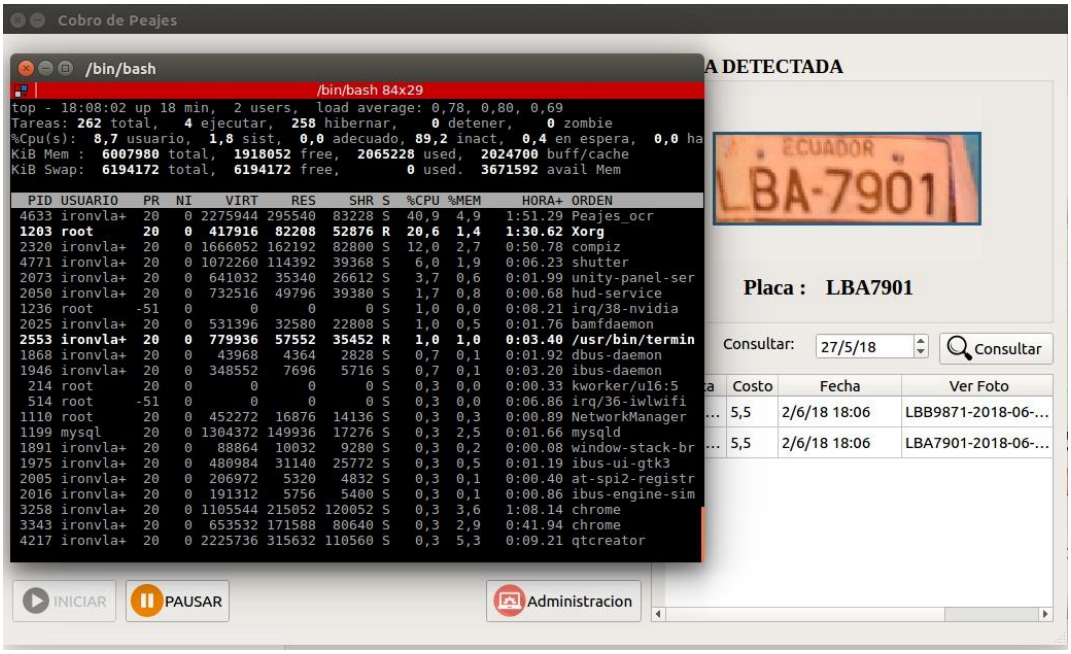
Estado conjunto de memoria y cpu al ejecutar el módulo de cobro

Descripción **%MEM:** memoria consumida por el proceso. **%CPU:** cantidad del procesador usada por el proceso.
Conclusión Al ejecutar el módulo de cobros se ve un uso del CPU del 36.1%

Tabla 35 Estado del CPU y memoria durante la ejecución del módulo de cobros

Estado conjunto de memoria y CPU durante la ejecución del módulo cobro

Descripción **%MEM:** memoria consumida por el proceso **%CPU:** cantidad del procesador usada por el proceso
Conclusión Al ejecutar el módulo de cobros por aproximadamente 2 horas se ve un uso del CPU del 40,9% y un uso de la memoria del 4,9%.

6.4.3. Pruebas de funcionalidad.

Para validar los módulos se realizaron pruebas de: unidad, integración y funcionamiento; con el objetivo de la verificación de que el sistema cumple y satisface los requisitos funcionales previamente definidos en la sección 6.1.1 Requerimientos funcionales.

6.4.3.1. Pruebas de unidad.

Las pruebas de unidad nos permitieron verificar el funcionamiento de los componentes individuales del sistema, ya que son la unidad más pequeña del diseño del software.

El proceso de desarrollo de la metodología ICONIX recomienda que para realizar las pruebas de unidad se lo debe hacer en base a los casos de uso, por lo que se utilizará esta pauta para realizar las diferentes pruebas. Para lo cual se siguió el siguiente código:

1. Descripción del caso de uso a probar
2. El caso de prueba deberá tener precondiciones, resultado esperado, resultado obtenido, casos de excepción.
3. Si el resultado obtenido no es el esperado en el caso de prueba se debe refactorizar el código.

A continuación se describen las pruebas de unidad realizadas.

Tabla 36 Caso de prueba crear persona

CASO DE PRUEBA : CREAR PERSONA	
Descripción: Prueba unitaria para el caso de uso crear persona.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de personas. **3.** Clic en el botón **Ingresar.**
Datos de entrada	**Nombres:** Ramiro Vladimir **Apellidos:** Vaca Moscoso **Cedula:** 1718562919 **Teléfono:** 0994750077 **Dirección:** Ciudad Victoria A' 29
Resultado esperado	Persona creada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	En caso de que exista una persona con el mismo número de cedula el registro no será ingresado.
Observación	Si no existen coincidencias con el campo **Cedula**, el usuario presiona la tecla **Enter** al finalizar de llenar los campos y el registro es ingresado.

Tabla 37 Caso de prueba modificar persona

CASO DE PRUEBA : MODIFICAR PERSONA	
Descripción: Prueba unitaria para el caso de uso modificar persona.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de personas. **3.** Clic en el registro a modificar.
Datos de entrada	**Nombres:** Ramiro Vladimir **Apellidos:** Vaca Moscoso
Resultado esperado	Persona modificada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	Si no existen coincidencias con el campo **Cedula**, el usuario presiona la tecla **Enter** al finalizar de llenar los campos y el registro es modificado.

Tabla 38 Caso de prueba eliminar persona

CASO DE PRUEBA : ELIMINAR PERSONA	
Descripción: Prueba unitaria para el caso de uso eliminar persona.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de personas. **3.** Clic en el registro a eliminar.
Datos de entrada	**Cedula:** 1718562919
Resultado esperado	Persona eliminada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	El usuario presiona el botón **"Eliminar"** al seleccionar el registro y no se elimina por estar asignado a un usuario.
Observación	El usuario presiona el botón **"Eliminar"** al seleccionar el registro y es eliminado.

Tabla 39 Caso de prueba crear cámara

CASO DE PRUEBA : CREAR CAMARA	
Descripción: Prueba unitaria para el caso de uso crear cámara.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de cámaras. **3.** Clic en el botón **Ingresar.**
Datos de entrada	**Nombre cámara:** Logitech c920 **Dirección :** 192.168.1.1(cámara local /media/sda/0) **Observación:** Cámara perteneciente al carril 1 de estación Loja.
Resultado esperado	Cámara creada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	El usuario presiona la tecla **Enter** al finalizar de llenar los campos y el registro es ingresado.
Observación	El usuario presiona la tecla **Enter** al finalizar de llenar los campos y el registro es creado.

Tabla 40 Caso de prueba modificar cámara

CASO DE PRUEBA : MODIFICAR CAMARA	
Descripción: Prueba unitaria para el caso de uso modificar cámara.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de cámaras. **3.** Clic en el registro a modificar.
Datos de entrada	**Nombre cámara:** Logitech c921
Resultado esperado	Cámara modificada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona la tecla **Enter** al finalizar de modificar los campos y el registro es modificado.

Tabla 41 Caso de prueba eliminar cámara

CASO DE PRUEBA : ELIMINAR CAMARA	
Descripción: Prueba unitaria para el caso de uso eliminar cámara.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de cámaras. **3.** Clic en el registro a eliminar.
Datos de entrada	**Nombre cámara:** Logitech c921
Resultado esperado	Cámara eliminada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona el botón **"Eliminar"** al seleccionar el registro y es eliminado.

Tabla 42 Caso de prueba crear tarifa

CASO DE PRUEBA : CREAR TARIFA	
Descripción: Prueba unitaria para el caso de uso crear tarifa.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de tarifas. **3.** Clic en el botón **Ingresar.**
Datos de entrada	**Tarifa:** Vehículo liviano **Descripción :** tarifa aplicada a vehículos livianos **Costo:** 2
Resultado esperado	Tarifa creada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona la tecla **Enter** al finalizar de llenar los campos y el registro es creado.

Tabla 43 Caso de prueba modificar tarifa

CASO DE PRUEBA : MODIFICAR TARIFA	
Descripción: Prueba unitaria para el caso de uso modificar tarifa.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de tarifas. **3.** Clic en el registro a modificar.
Datos de entrada	**Costo:** 2.50
Resultado esperado	Tarifa modificada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona la tecla **Enter** al finalizar de modificar los campos y el registro es modificado.

Tabla 44 Caso de prueba eliminar tarifa

CASO DE PRUEBA : ELIMINAR TARIFA	
Descripción: Prueba unitaria para el caso de uso eliminar tarifa.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de tarifas. **3.** Clic en el registro a eliminar.
Datos de entrada	**Tarifa:** Vehículo liviano
Resultado esperado	Tarifa eliminada correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona el botón **"Eliminar"** al seleccionar el registro y es eliminado.

Tabla 45 Caso de prueba crear usuario

CASO DE PRUEBA : CREAR USUARIO	
Descripción: Prueba unitaria para el caso de uso crear usuarios.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de usuarios. **3.** Clic en el botón **Ingresar.**
Datos de entrada	**Nombre Usuario:** rvladimir **Contraseña :** *****
Resultado esperado	Usuario creado correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona la tecla **Enter** al finalizar de llenar los campos y el registro es creado.

Tabla 46 Caso de prueba modificar usuario

CASO DE PRUEBA : MODIFICAR USUARIO	
Descripción: Prueba unitaria para el caso de uso modificar usuario.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de usuarios. **3.** Clic en el registro a modificar.
Datos de entrada	**Contraseña :** *******
Resultado esperado	Usuario modificado correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona la tecla **Enter** al finalizar de modificar los campos y el registro es modificado.

Tabla 47 Caso de prueba eliminar usuario

CASO DE PRUEBA : ELIMINAR USUARIO	
Descripción: Prueba unitaria para el caso de uso eliminar cámara.	
Precondiciones	**1.** Ingreso al módulo de administración. **2.** Ingreso a la sección de usuarios. **3.** Clic en el registro a eliminar.
Datos de entrada	**Nombre Usuario:** rvladimir
Resultado esperado	Usuario eliminado correctamente.
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona el botón **"Eliminar"** al seleccionar el registro y es eliminado.

Tabla 48 Caso de prueba cobrar ticket

CASO DE PRUEBA : COBRAR TICKET	
Descripción: Prueba unitaria para el caso de uso cobrar ticket.	
Precondiciones	**1.** Ingreso al sistema **2.** Inicio la detección de placas
Datos de entrada	**Entrada de video.** **Placa :** LBB9871
Resultado esperado	Ticket de cobro generado correctamente
Resultado obtenido	**Satisfactorio**
Casos de excepción	Ninguno.
Observación	El usuario presiona el botón **"Cobrar"** el cobro es realizado

6.4.3.2. Pruebas de integración.

Mediante la técnica de pruebas de integración, se pudo estructurar al sistema, con la finalidad de tomar los componentes que interactúan directamente en las pruebas de unidad y verificar su correcto funcionamiento en conjunto.

A continuación se describe las pruebas de integración realizadas en cada uno de los módulos.

Tabla 49 Pruebas de integración

CASO DE PRUEBA	PRUEBA UNITARIA	RESULTADOS ESPERADOS
Administración de personas	Crear Persona	Módulo Integrado
	Modificar Persona	
	Eliminar Persona	
Administración de Cámaras	Crear Cámara	Módulo Integrado
	Modificar Cámara	
	Eliminar Cámara	
Administración de Tarifas	Crear Tarifa	Módulo Integrado
	Modificar Tarifa	
	Eliminar Tarifa	
Administración de Usuarios	Crear Usuario	Módulo Integrado
	Modificar Usuario	
	Eliminar Usuario	
Módulo de Cobro	Detección de Placa	Módulo Integrado
	Obtención de Datos	
	Generación de Ticket	

6.4.3.3. Pruebas de funcionalidad.

Las pruebas de funcionalidad de este prototipo se realizaron en una maqueta a escala cuyo modelo de peaje es general, es decir, se realizó una maqueta acorde a los lineamientos recopilados en la revisión de literatura.

Figura 58 Maqueta de estación de peaje

Los pasos seguidos para comprobar que el software cumple con la función para la que se había pensado y su precisión son los siguientes:

1. ADECUACIÓN DE LA CÁMARA

La cámara utilizada en la maqueta es una cámara web de marca Logitech modelo C920, que mide 9.4 cm x 2.9 cm, peso de 162 gramos y cuyas especificaciones técnicas son:

- Tecnología UVC H.264 compatible con calidad Full HD 1080p (hasta 1920 x 1080 píxeles) a 30 fotogramas por segundo.
- Campo visual diagonal de 78°
- Captura de imágenes (W 16:9) de: 2,0 MP, 3 MP*, 6 MP*, 15 MP*
- Captura de video de: 360p, 480p, 720p, 1080p
- Compatibilidad con Windows, Linux y Mac.

Figura 59 Cámara Logitech C920

La cámara se encuentra ubicada a un costado del carril de la estación de peaje, la misma apunta hacia la posición donde se localiza la placa de cualquier vehículo.

2. CONFIGURACIÓN PARA CONEXIÓN DE LA CÁMARA

La cámara web estará conectada a un ordenador por conexión USB, este ordenador debe poseer cualquier distribución de Linux e instalado el programa de escritorio para el cobro de peajes.

Dentro de la interfaz del software de escritorio, para el usuario con rol administrador, consta la administración de cámaras, en la cual el administrador podrá agregar o eliminar una cámara. A continuación la interfaz de administración de cámaras:

Figura 60 Interfaz para administrar cámaras

El administrador, puede observar en esta interfaz lo siguiente según como corresponde la numeración de la Figura 60:

1) **Área del listado de cámaras,** este espacio muestra las cámaras que han sido ingresadas por el administrador, como se observa, en una configuración de cámara se necesita un nombre, dirección y observación (todos estos campos permiten el ingreso de datos de texto y números).
   Nombre: es el nombre de la cámara o cualquier otro identificador con el que el administrador asocie una cámara física en específico.
   Dirección: es el puerto en donde se reconoce la cámara y ésta puede trabajar, para capturar las placas vehiculares que posteriormente se les hará el cobro de peaje.
   Observación: es cualquier indicación que el administrador desee colocar para una cámara en específico.
2) **Área de testeo de cámara,** este espacio cuando no se ha seleccionado ninguna cámara aparece deshabilitado (el usuario no puede interactuar con él). Cuando el usuario ha seleccionado una cámara este espacio se habilita, siendo así que el usuario puede hacer clic en el botón Probar Cámara y automáticamente en el lugar donde se observa la imagen de una cámara, cambiará mostrando lo que la cámara seleccionada esté capturando en ese momento.
   Este espacio está dedicado para que el usuario pueda probar la funcionalidad de la cámara.
3) **Botón insertar,** permite ingresar nuevas configuraciones de cámara.
   El administrador para ingresar una nueva cámara, debe colocar el mouse debajo del último registro y automáticamente se crea una nueva fila en la cual debe registrar los campos solicitados en **1) Área del listado de cámaras.** Una vez completados estos 3 campos, el usuario debe hacer clic en el botón Insertar y se agregan automáticamente los datos previamente ingresados, (se refresca la interfaz añadiendo la nueva cámara).
4) **Botón eliminar,** permite al administrador eliminar la configuración de una cámara, por ende la cámara es eliminada del **1) Área del listado de cámaras** sin poder ya usarse.

Con esta configuración dada en 3) Botón insertar, la cámara queda conectada al módulo de cobro y está listo para usarse en la cabina de peaje dentro del sistema.

3. DETECCIÓN DE PLACAS VEHICULARES

En cuanto al entrenamiento del clasificador detallado en Clasificador HAAR Cascades. para evitar esos falsos positivos se entrenó al clasificador con un conjunto positivo de 1200 imágenes y un conjunto negativo de 2000 imágenes, con un tiempo de entrenamiento aproximado de 2 días.

Posteriormente se probó el nivel de confianza del sistema, que en condiciones de poca luz, ángulo de la cámara, placas con polvo o muy alejadas de la cámara, disminuye en 2% el reconocimiento del OCR; mas no de la detección y reconocimiento de la placa. A continuación se resumen las pruebas y resultados obtenidos:

Tabla 50 Predicción del sistema para la detección de placas vehiculares

Imágenes de muestra	Placas detectadas	Nivel de confianza	% Error
50	48	95%	5%

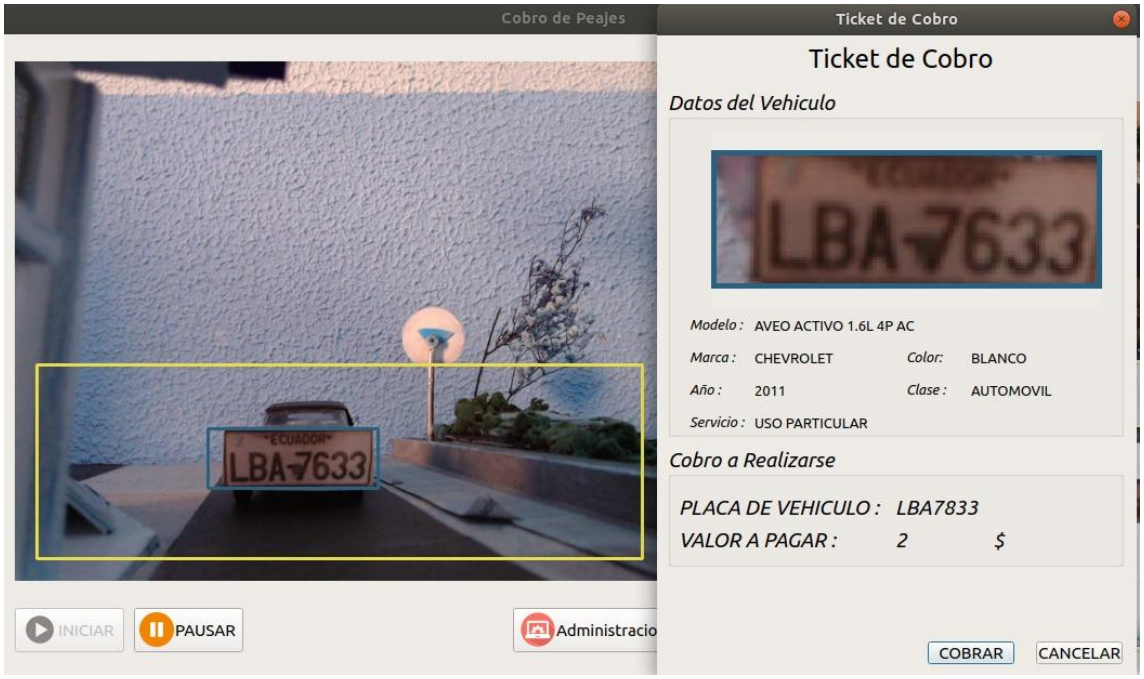


Figura 61 Entorno y resultado de la ejecución de pruebas del nivel de confianza del sistema

En la figura anterior se muestra el entorno donde se capturó y reconoció la placa vehicular, esta es la evidencia de 1 de las 50 imágenes de muestra para la realización de estas pruebas.

7. DISCUSIÓN.

Todas las secciones anteriores muestran el proceso para la realización del presente Proyecto de Trabajo de Titulación, tras esta documentación es necesaria una evaluación de los objetivos planteados inicialmente y determinar si se ha logrado cumplir con cada uno de ellos. Esta sección 7 explica el uso de los métodos y técnicas utilizadas en cada objetivo y el cumplimiento del mismo.

7.1.1. Objetivo específico 1: Analizar el estado del arte de los peajes en nuestro país, para determinar los requerimientos que construirán el software.

Para dar cumplimiento a este objetivo, primeramente se investigó detalladamente en la comunidad científica la situación actual de los peajes en todo el territorio nacional, como técnica se utilizó la documentación bibliográfica y la herramienta para dicha documentación fue Word 2013, el resultado de este análisis se evidencia en los 3 capítulos de 4. REVISIÓN DE LITERATURA..

Conjuntamente se capturó el modelo de negocio de los peajes para así determinar los requerimientos del sistema, cuyos entregables son: requerimientos funcionales y no funcionales, prototipo de interfaces, modelo de dominio, diagrama de casos de uso, diagramas de secuencia y diagrama de clases. (Ver Resultados: 6.1 Fase Uno).

Este objetivo tiene relación con las fases: Análisis de Requisitos, Análisis y Diseño preliminar y Diseño de la metodología de desarrollo de software ICONIX. Se utilizaron técnicas como: observación directa, documentación bibliográfica y revisión de tutorías, que establecieron una visión global del dominio del problema, además de varias alternativas para resolverlo. Las herramientas para la documentación de estas técnicas y entregables fueron: Visual Paradigm EC y Word 2013.

Con todos los entregables, técnicas y herramientas enunciados, se dio cumplimiento a este objetivo, recordando que el presente proyecto es un prototipo de peaje que no tiene un usuario final en específico, sino, es un modelo de peaje en general que puede ser implementado como solución al cobro de peaje; es necesario enunciar que en el país no sólo es una empresa la dedicada al cobro de peajes, sino, se conocen 4 empresas privadas y 5 prefecturas.

7.1.2. Objetivo específico 2: Configurar y aplicar técnicas de Visión y Reconocimiento de Vehículos con OpenCV.

Este segundo objetivo tuvo como finalidad la preparación del entorno de desarrollo para la consecución del tercer objetivo. De este modo, se sustentó en la técnica de documentación bibliográfica la evidencia mostrada en Resultados 6.2 Fase dos.

Además, este objetivo aplica ya las técnicas de Visión y Reconocimiento de Vehículos con OpenCV, esto en el proyecto de trabajo de titulación generó como evidencia un software de escritorio que será utilizado en los ordenadores de las cabinas de peaje. Para el entregable software de este objetivo se utilizaron herramientas como son: el lenguaje de programación C++, OpenCV, Tesseract y QT.

El lector puede remitirse a la evidencia simbólica en este documento del clasificador en cascada HAAR que utilizó para su entrenamiento un conjunto de 1200 imágenes positivas y 2000 imágenes negativas, documentado en: 6.2.4 Clasificador HAAR Cascades. Como puede inferir el lector, este segundo objetivo prepara para la codificación del sistema de peaje, ya que primeramente es necesario poder identificar la placa de un vehículo, capturarla y reconocer el texto de dicha placa, a la cual posteriormente se le hará el cobro por concepto de peaje.

7.1.3. Objetivo específico 3: Desarrollo del módulo para el cobro de peajes según el tipo de vehículo.

Este tercer objetivo acoge a la última fase de la metodología de desarrollo de software ICONIX, llamada: fase de implementación, la cual expone como entregables a: diagrama de componentes y diagrama de despliegue, que conjuntamente reflejan la arquitectura del sistema. (Ver 6.3.1 Arquitectura del sistema), la herramienta utilizada fue Visual Paradigm. EC. Además, como complemento se documentó el código de: obtención de imágenes válidas, reconocimiento de caracteres, consulta y generación de cobro; todo esto documentado en 6.3.2 Generación de Código.

Como se manifestó anteriormente, la finalidad de este objetivo es la realización del cobro de peaje, para lo cual es necesario ya tener el reconocimiento de placas vehiculares. En este objetivo se concluyó el software de escritorio, el cual con la placa ya reconocida

consulta en el sistema de la ANT una placa vehicular, este devuelve información del tipo de vehículo, para según esto imponer el cobro automático de peaje que será entregado en forma de Ticket a través de una ventana de dialogo para el usuario de la cabina de peaje. Además, como adicional se ha realizado un sistema web desarrollado en el lenguaje Python con el framework Django, para que los usuarios de las estaciones de peaje puedan consultar los rubros por pago de peaje y; los administradores de estaciones de peaje puedan dar seguimiento a cada estación.

Con el cumplimiento de este objetivo se tiene ya el software y maqueta que constituyen el prototipo de peaje, propuesto en este trabajo de titulación.

7.1.4. Objetivo específico 4: Pruebas de funcionalidad en tiempo real del prototipo.

Una vez desplegado el sistema en los servicios de hosting de DigitalOcean, se ha procedido a evaluar el desempeño del sistema mediante las pruebas de funcionalidad en tiempo real, las cuales abarcaron pruebas de: caja blanca, carga y funcionalidad (Resultados 6.4 Fase 4.). Las múltiples pruebas individuales como en conjunto de todo el sistema y sus componentes software, buscaron vulnerabilidades de acuerdo a su ejecución y utilización, así como la satisfacción de los requerimientos funcionales.

La metodología ICONIX dentro de sus fases, no contempla las pruebas del software, sin embargo, como el sustento de esta metodología son los casos de uso, la atención a las pruebas recaen en cada caso de uso.

Concluido este objetivo y con resultados de pruebas favorables, se colige que el proyecto software: **“Diseño y Construcción de un Prototipo para Cobro de Peajes con Visión Artificial”** es apto porque cumple con todos los objetivos planteados de este proyecto de trabajo de titulación y se ajusta a las necesidades generales de los peajes.

Cabe destacar que el nivel de confianza del sistema es del 98% y el 2% concerniente a la tasa de error, tiene lugar debido a: condiciones de iluminación porque cuando hay excesiva luz o contraluz, el sistema reconoce una placa pero no detecta correctamente el OCR. También si el ángulo entre la cámara y la región de la placa es muy amplio, no se reconoce la placa vehicular. La razón que se tenga una tasa de error tan pequeña es porque el entorno donde se ejecutó las pruebas es un entorno controlado, el prototipo de peaje tiene su propia maqueta y en ella se minimiza lo mejor posible situaciones de entorno que pudieran aumentar el porcentaje de error de detección del OCR.

8. CONCLUSIONES.

Lo expuesto a lo largo de este trabajo permite arribar a las siguientes conclusiones:

- La integración de herramientas para procesamiento de imágenes como OPENCV conjuntamente con la creación de un clasificador en cascada, permiten el OCR de placas con un 95% de confianza; automatizando así el modelo de negocio de las estaciones de peaje en general.
- La metodología de desarrollo de software pesada ligera: ICONIX, permitió realizar avances incrementales y con ello cada vez se fue mejorando el software hasta tener su refinamiento final, esto a partir del conocimiento del modelo de negocio de las estaciones de peaje que fue modelado en el diagrama de casos de uso, el cual es conocido como la base de desarrollo porque es el elemento que marca la trazabilidad de los demás artefactos del software.
- Las herramientas libres como: el framework Django generó código limpio y organizado, ayudando en la rápida construcción del sistema, modularidad y escalabilidad.
- Las pruebas de funcionalidad aseguraron el correcto cumplimiento de las necesidades del producto especificadas en los requisitos funcionales y casos de uso, además, demostraron que el software para su ejecución no demanda más que: 20% de consumo de memoria y 36.1% de consumo de CPU, como requisitos aproximados de procesamiento.
- Pese a que el sistema ha tenido una buena respuesta con una tasa de error de tan solo 5%, se considera importante destacar que el entorno de prueba ha sido un entorno controlado con algunas simulaciones de condiciones no tan favorables para el cobro de peajes.

9. RECOMENDACIONES.

Adicionalmente de las conclusiones planteadas anteriormente, es también necesario realizar algunas recomendaciones al momento de desarrollar sistemas basados en visión artificial:

- Es primordial el testeo recurrente del sistema para comprobar el estado de la memoria y evitar desbordamientos por errores de administración de hilos.
- Se recomienda una correcta iluminación, posicionamiento fijo de la cámara y ángulo con la placa del vehículo; también se recomienda una cámara cuyo propósito específico sea la captura de imágenes para proyectos de visión artificial; ya que de parte del software, éste está programado para tratar la imagen aún en condiciones poco optimistas, dejando así la mejora en hardware.
- Al construir clasificadores en cascada para reconocimiento de objetos se recomienda utilizar la mayor cantidad posible de imágenes positivas en diferentes condiciones de luz, aumentando así la robustez del clasificador y disminuyendo la tasa de error en la detección de los objetos.
- Investigar las herramientas utilizadas en el presente proyecto para no limitarlo a una sola plataforma y este pueda estar disponible como servicio en la nube y también para el sistema operativo Windows.
- Crear una base de datos de vehículos propia para no depender de servicios de terceros en caso de que estos llegaran a fallar y no afecten la funcionalidad de nuestro sistema.
- Siempre utilizar librerías de visión artificial robustas, como OPENCV que siendo software libre tiene un fuerte soporte y mantenimiento por parte de la comunidad de desarrollo y empresas que lo utilizan en sus proyectos.

TRABAJOS FUTUROS

La visión artificial es un campo muy extenso de investigación y desarrollo, por lo que este tipo de sistemas da lugar a nuevas líneas de investigación o mejoras del sistema, sirviendo de base para el desarrollo de otros sistemas más robustos. Se pueden considerar como trabajos futuros los siguientes:

- Desarrollo de un módulo para la realización de cobros electrónicos o facturación electrónica en el prototipo del sistema para cobro de peajes con visión artificial.
- Implementación de este sistema en un peaje real.
- Investigación para la migración del sistema a un dispositivo móvil para cobros en parqueaderos públicos o privados de las ciudades en base al reconocimiento de sus placas
- Minería de datos en la información recopilada para establecer mayor afluencia de rutas y aplicar descongestionamiento vehicular en base a señalización inteligente.

10. BIBLIOGRAFÍA.

[1] F. Rosales, «El peaje califica como un tributo,» Noviembre 2010. [En línea]. Available: http://www.derecho.usmp.edu.pe/7ciclo/derecho_tributario_I/articulos/2010/el_peaje.pdf. [Último acceso: 10 Septiembre 2017].

[2] eTrust Sistemas, «Peaje,» 2015. [En línea]. Available: http://its.etrustsistemas.com/index.php/articulos/2-uncategorised/3-peajes. [Último acceso: Septiembre 10 2017].

[3] F. Echeverría, «La concesión vial como aporte al desarrollo integral del país,» Junio 2001. [En línea]. Available: http://repositorio.iaen.edu.ec/bitstream/24000/310/1/IAEN-018-2001.pdf. [Último acceso: 10 Septiembre 2017].

[4] Diario La Hora, «Peaje: opción para mantener vías,» 3 Abril 2007.

[5] PANAVIAL S.A, «Tarifas,» [En línea]. Available: http://www.panavial.com/estaciones-de-peaje-panavial-administracion-red-autopistas-vias-carreteras-panamericana-vial-ecuador.php?tablajb=estaciones_de_peaje&p=24&t=Tarifas&. [Último acceso: 10 Septiembre 2017].

[6] CONORTE S.A, «Tarifario Samborondon,» [En línea]. Available: http://www.conortesa.com/cuadro-tarifario-samborondon/. [Último acceso: 10 Septiembre 2017].

[7] CONCEGUA S.A, «Tarifas de Peaje Milagro,» [En línea]. Available: http://www.concegua.com/cuadro-tarifario-milagro/. [Último acceso: 10 Septiembre 2017].

[8] El Telégrafo, «Peajes, un solo dispositivo y dos recargas diferentes,» 7 Marzo 2014. [En línea]. Available: https://www.eltelegrafo.com.ec/noticias/informacion-general/1/peajes-un-solo-dispositivo-y-dos-recargas-diferentes. [Último acceso: 10 Septiembre 2017].

[9] I. Alarcon, «Más facilidades para el pago electrónico en tres peajes,» 7 11 2015. [En línea]. Available: http://www.elcomercio.com/actualidad/movilidad-quito-pago-electronico-peajes.html. [Último acceso: 10 Septiembre 2017].

[10] Metro Ecuador, «Epmmop inicia entrega de tags para Telepeaje,» 25 Mayo 2017. [En línea]. Available: https://www.metroecuador.com.ec/ec/noticias/2017/05/25/epmmop-inicia-entrega-tags-telepeaje.html. [Último acceso: 10 Septiembre 2017].

[11] El Telégrafo, «Telepeajes fueron cambiados,» 4 Agosto 2012. [En línea]. Available: http://www.eltelegrafo.com.ec/noticias/quito/1/telepeajes-fueron-cambiados. [Último acceso: 10 Septiembre 2017].

[12] ADIPISCOR, «¿Cómo obtener el TAG para el auto en Quito?,» 6 Marzo 2015. [En línea]. Available: https://www.adipiscor.com/como/422/Como-obtener-el-TAG-para-el-auto-en-Quito. [Último acceso: 10 Septiembre 2017].

[13] EPMMOP, «Telepeaje: más de mil usuarios ya disponen de Tag,» 6 Junio 2017. [En línea]. Available: http://www.epmmop.gob.ec/epmmop/index.php/noticias/boletines/item/2810-telepeaje-mas-de-mil-usuarios-ya-disponen-de-tag. [Último acceso: 10 Septiembre 2017].

[14] El Comercio, «El dispositivo para usar el telepeaje cuesta USD 7,» [En línea]. Available: http://www.elcomercio.com/actualidad/quito/dispositivo-telepeaje-cuesta-usd.html. [Último acceso: 10 Septiembre 2017].

[15] Foros Ecuador, «Placas de Ecuador (por provincias):,» 7 Mayo 2013. [En línea]. Available: http://www.forosecuador.ec/forum/aficiones/autos-y-motos/212-placas-de-ecuador-por-provincias. [Último acceso: 10 Septiembre 2017].

[16] Revista tecnológica ESPOL, «Detección y extracción de placas de vehiculos en señales de video,» Octubre 2012. [En línea]. Available: https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwj8oKHPheDYAhVDXq0KHZ3dBFsQFgg2MAE&url=http%3A%2F%2Fwww.rte.espol.edu.ec%2Findex.php%2Ftecnologica%2Farticle%2FviewFile%2F95%2F63&usg=AOvVaw1tE8xvL528fT93_K1cGcF6. [Último acceso: 15 Septiembre 2017].

[17] R. López, «Detección de textos en escenas naturales,» 29 Junio 2016. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/92594/L%C3%93PEZ%20-%20Detecci%C3%B3n%20de%20texto%20en%20escenas%20naturales.pdf?sequence=1. [Último acceso: 18 Septiembre 2017].

[18] J. Delgado, «Reconocimiento de placas vehiculares,» 2010. [En línea]. Available: http://tesis.ipn.mx/jspui/bitstream/123456789/9880/1/221.pdf. [Último acceso: 13 Septiembre 2017].

[19] M. Elkan, «Umbralización en OpenCV,» 28 Julio 2017. [En línea]. Available: http://acodigo.blogspot.com/2017/07/umbralizacion-en-opencv.html. [Último acceso: 20 Septiembre 2017].

[20] OpenCV 3.4.0-dev , «Umbral de la imagen,» [En línea]. Available: https://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html. [Último acceso: 20 Septiembre 2017].

[21] L. Neumann y J. Matas, «Real-Time Scene Text Localization and Recognition,» [En línea]. Available: http://cmp.felk.cvut.cz/~matas/papers/neumann-2012-rt_text-cvpr.pdf. [Último acceso: 20 Septiembre 2017].

[22] D. Sánchez, «Códigos correctores de cascadas de clasificadores y graphs cuts para la segmentación multi-extremidad,» 25 Junio 2012. [En línea]. Available: http://www.maia.ub.es/~sergio/linked/memoriapfc_dani.pdf. [Último acceso: 18 Septiembre 2017].

[23] OpenCV, «Detección de rostros utilizando Cascadas Haar,» [En línea]. Available: https://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html. [Último acceso: 18 Septiembre 2017].

[24] M. Carmelo, «Entrenar OpenCV en Detección de Objetos,» 6 Diciembre 2015. [En línea]. Available: http://acodigo.blogspot.com/2015/12/entrenar-opencv-en-deteccion-de-objetos.html. [Último acceso: 20 Septiembre 2017].

[25] V. Navarro, «Seguimiento de características faciales en entornos controlados,» 19 Septiembre 2007. [En línea]. Available: http://www.maia.ub.es/~sergio/linked/vanessa07.pdf. [Último acceso: 21 Septiembre 2017].

[26] OpenCV, «Releases,» [En línea]. Available: https://opencv.org/releases.html. [Último acceso: 20 Septiembre 2017].

[27] L. Gracia, «¿Qué es Tesseract OCR?,» 19 Mayo 2016. [En línea]. Available: https://unpocodejava.com/2016/05/19/que-es-tesseract-ocr/. [Último acceso: 20 Septiembre 2017].

[28] GitHub, «Tesseract-Ocr,» [En línea]. Available: https://github.com/tesseract-ocr/. [Último acceso: 22 Septiembre 2017].

[29] GitHub, «tesseract-ocr / tesseract,» 10 Diciembre 2017. [En línea]. Available: https://github.com/tesseract-ocr/tesseract/wiki/Compiling. [Último acceso: 22 Diciembre 2017].

[30] J. Torres, «Proyecto Qt: Framework de desarrollo de aplicaciones,» 28 Junio 2013. [En línea]. Available: https://medium.com/jmtorres/proyecto-qt-framework-de-desarrollo-de-aplicaciones-2b2f895ac285. [Último acceso: 29 Septiembre 2017].

[31] The Qt Company, «Licensing,» [En línea]. Available: https://www1.qt.io/licensing/. [Último acceso: 29 Septiembre 2017].

[32] BBVA APIs, «Django: guía rápida para desarrollar páginas web con este framework,» 14 Enero 2016. [En línea]. Available: https://bbvaopen4u.com/es/actualidad/django-guia-rapida-para-desarrollar-paginas-web-con-este-framework. [Último acceso: 30 Septiembre 2017].

[33] A. Montenegro, «ntroducción a Django Rest Framework,» 21 Junio 2012. [En línea]. Available: https://axiacore.com/blog/2012/06/introduccion-a-django-rest-framework/. [Último acceso: 30 Septiembre 2017].

[34] Ecured, «Aplicación de la metodología semi-ágil ICONIX,» 24 Julio 2014. [En línea]. Available: http://laccei.org/LACCEI2014-Guayaquil/RefereedPapers/RP246.pdf. [Último acceso: 3 Octubre 2017].

[35] Ecured, «ICONIX,» [En línea]. Available: https://www.ecured.cu/ICONIX. [Último acceso: 3 Octubre 2017].

[36] C. De San Martín, «Metodología ICONIX,» [En línea]. Available: http://www.portalhuarpe.com.ar/Seminario09/archivos/MetodologiaICONIX.pdf. [Último acceso: 3 Octubre 2017].

[37] ICONIX, «Manual Introductorio de Iconix,» [En línea]. Available: http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesICONIX.pdf. [Último acceso: 3 Octubre 2017].

[38] M. Sumano, J. Fernández y J. Andrade, Octubre 2004. [En línea]. Available: https://www.uv.mx/personal/asumano/files/2010/07/iconix2.pdf. [Último acceso: 3 Octubre 2017].

11. ANEXOS.

ANEXO I: INSTALACIÓN DE QT 5.10.0 EN UBUNTU 16.04

Descargar QT desde https://www.qt.io/download#section-2. QT está disponible en dos versiones: comercial y libre; para este proyecto se utilizó su versión libre.

Ilustración 1 instalación de QT OpenSource

Una vez descargado el ejecutable. En la terminal se debe ajustar los permisos y ejecutar el instalador, siguiendo las instrucciones para completar la instalación. Ver ilustración 2 hasta la ilustración 6.

```
chmod +777 qt-opensource-linux-x64-5.7.0.run
```

```
./qt-opensource-linux-x64-5.10.0.run
```

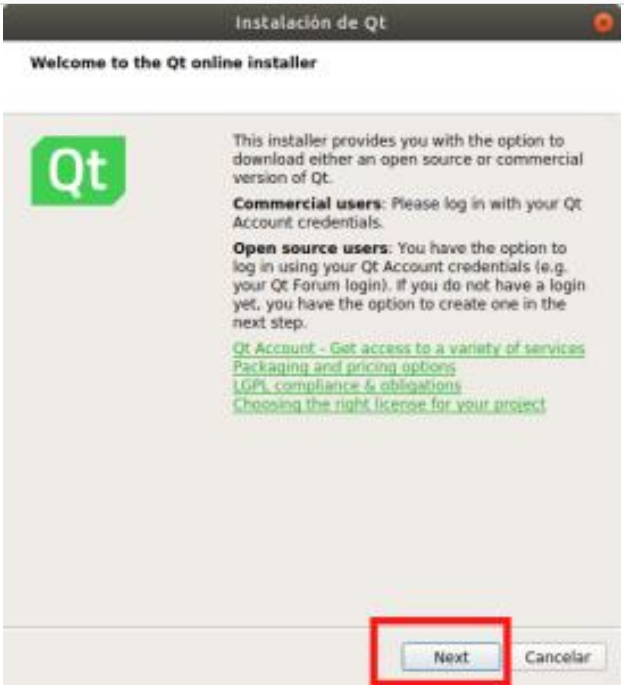


Ilustración 2 Inicialización el instalador de QT

Se debe ingresar datos de la cuenta o crear una cuenta

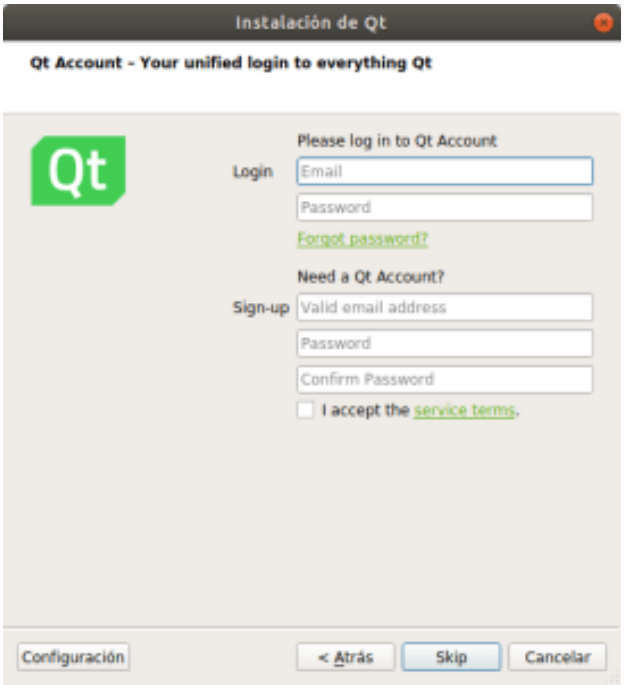


Ilustración 3 Inicio se sesión de cuenta de usuario en QT

Se inicia el proceso de instalación, para ello se selecciona la carpeta de instalación.

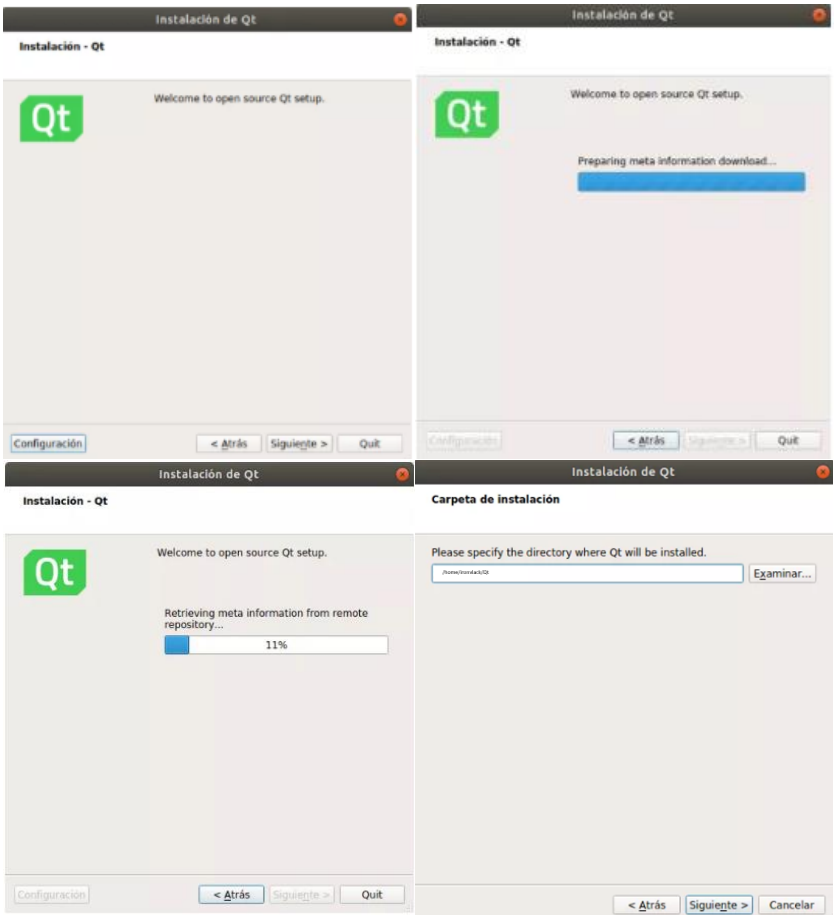


Ilustración 4 Proceso para la selección de la carpeta de instalación de QT

Se presenta una interfaz para seleccionar la versión a instalar, que será la versión 5.10.

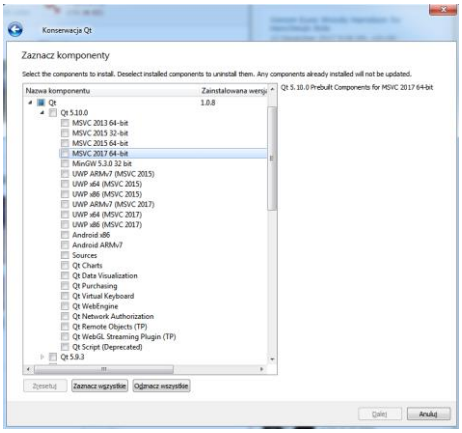


Ilustración 5 Menú de selección para instalar la versión 5.10 de QT

Se debe aceptar la licencia y esperar a que se termine de instalar QT. Finalizado el proceso de instalación de QT, al ejecutarlo se tiene la siguiente pantalla:

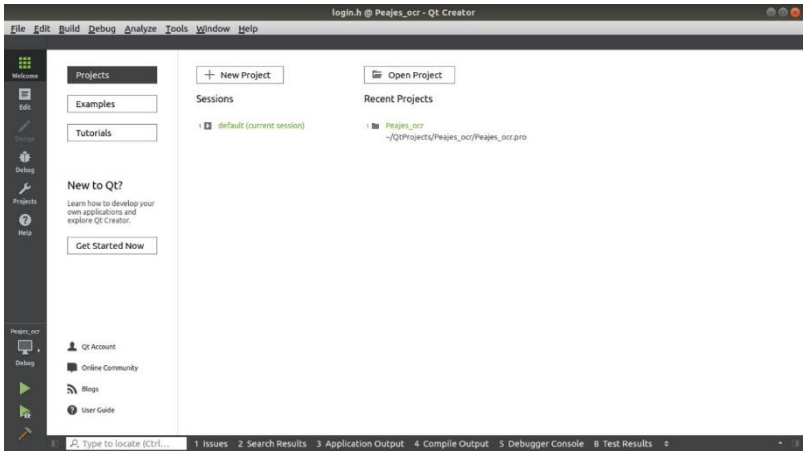


Ilustración 6 Pantalla principal de QT 5.10.0 en Ubuntu 16.04

Conjuntamente se debe instalar g++, para lo cual desde la terminal se escribirá el siguiente comando.

```
sudo apt-get install build-essential
```

Ya instalado g++, se debe iniciar QT, en Tools>Options. Clic en Build & Run, en la pestaña Kit, verificar que el compilador esté configurado, caso contrario se debe establecer el compilador g++.

Luego, se debe proceder a instalar la librería de configuración de fuentes genéricas – runtime, para lo cual desde la terminal se escribirá el siguiente comando.

```
sudo apt-get install libfontconfig1
```

Las librerías OpenGL se deben instalar con el siguiente comando en la terminal:

```
sudo apt-get install mesa-common-dev
sudo apt-get install libglu1-mesa-dev -y
```

ANEXO II: GLOSARIO DE TÉRMINOS

TÉRMINO	DESCRIPCIÓN
ANPR	Automatic Number Plate Recognition - Reconocimiento automático de matrículas.
ANT	Agencia Nacional de Tránsito.
CONCEGUA	Empresa que tiene concesionado el cobro de peaje en la región amazónica.
CONORTE	Empresa que tiene concesionado el cobro de peaje en la región costa.
CONVIAL	Empresa que tiene concesionado el cobro de peaje en la región centro sur Manabita.
CRUD	Create, Read, Update and Delete - Crear, Leer, Modificar y Eliminar.
DRF	Django REST Framework.
EC	Edición Comunitaria.
EPMMOP	Empresa Pública Metropolitana de Movilidad y Obras Públicas.
HTTP	Hypertext Transfer Protocol – Protocolo de Transferencia de Hipertexto.
ICONIX	Metodología de desarrollo de software pesada-ligera que se halla a medio camino entre un RUP (Rational Unified Process) y un XP (eXtreme Programming).
MTOP	Ministerio de Transporte y Obras Públicas.
MVT	Patrón de diseño Model View Template – Modelo Vista Plantilla.
OCR	Optical Character Recognition - Reconocimiento óptico de caracteres.
OPENCV	Open Source Computer Vision - Visión por computadora de código abierto. Es una biblioteca libre de visión artificial.
PANAVIAL	Empresa que tiene concesionado el cobro de peaje de la Panamericana en la Sierra que va desde Rumichaca a Riobamba.
PTT019	Proyecto de Trabajo de Titulación número 19. Es el número de perfil del presente proyecto de trabajo de titulación.
RF	Requerimiento Funcional.
RGB	Red, Green, Blue, en español «rojo, verde y azul». Es la composición del color en términos de la intensidad de los colores primarios de la luz.
RNF	Requerimiento No Funcional.
ROI	Región de Interés.
RUP	Metodología de desarrollo de software Rational Unified Processes - Procesos Racionales Unificados.
TELEPASS	Dispositivo electrónico que contabiliza el número de pasadas por el Telepeaje.
TELEPEAJE	Forma de pago automático por concepto de peaje. Denominada así a la estación donde se cobra el peaje.
TESSERACT	Tesseract OCR es un motor OCR open-source (libtesseract) creado para el reconocimiento óptico de caracteres (OCR).
UML	Unified Modeling Language - Lenguaje Unificado de Modelado.
XP	Metodología de desarrollo de software Extreme Programming - Programación extrema.

ANEXO III: LICENCIA CREATIVE COMMONS

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

You are free to:

Under the following terms:

Notices:

ANEXO IV: ARTÍCULO CIENTÍFICO

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO PARA COBRO DE PEAJES CON VISION ARTIFICIAL

V. Vaca, M. Palma

rvvacam@unl.edu.ec

mpalma@.unl.edu.ec

Universidad Nacional de Loja

Abstract—The present article shows the creation of a prototype for charging tolls, that is, a base system for the recognition of vehicle plates and, depending on the type of vehicle, a charge is made, using the artificial vision libraries OpenCV, the programming language C++ and the Integrated QT Development Environment. For the implementation of the system the heavy-light ICONIX methodology was used, because it will allow us to continue making iterations of the system, in addition to the reuse of the components due to the modularity that it establishes. The most important stages of development include the analysis, design and implementation of the system.

Keywords— **OpenCV, Artificial Vision, QT Ide, ICONIX, Haarcascade**

INTRODUCCIÓN

La inteligencia artificial en los últimos años ha dado grandes avances tecnológicos permitiendo al ser humano utilizar cada vez más a las maquinas en tareas que antes no eran posibles, como es el control de complejos sistemas, toma de decisiones respecto a resultados obtenidos en diferentes procesos, siendo estas actividades las más generales y populares para la aplicación de la inteligencia artificial. Una de las ramas más populares y usadas en los últimos años es la visión por computadora o visión artificial, esta nos ha permitido realizar tareas muy complejas a la hora de utilizarla en nuestros sistemas. Teniendo en cuenta que para el ser humano el sentido de la visión es en el que más confía, debido a que reaccionamos a todo tipo de estímulos y situaciones que logramos obtener del mundo real a través de la vista. En los sistemas funcionan de la misma manera, logrando obtener mucha más información de una imagen que de grandes cantidades de texto, siendo más fácil la interpretación de la imagen, además de que se puede estructurar de una mejor manera la información.

En nuestra sociedad actualmente la información se encuentra disponible en gran cantidad a través de imágenes, muchas de las veces no logramos interpretar esta información debido a que no contamos con las herramientas necesarias para esta tarea. Con una adecuada implementación de estos sistemas que nos permitan la interpretación de la información nos ahorrarían costos además de tiempo. En el sector vehicular y de movilidad la visión artificial está tomando un gran protagonismo, como es el uso de esta tecnología, en el control de semáforos inteligentes, vehículos autónomos, control vehicular, etc. En este último es donde se ha empezado a utilizar la detección de las placas o matriculas vehiculares para el control del flujo vehicular tanto también para control de robos, conteos, y muchos usos más.

Ahora teniendo esta información de los vehículos se pueden acoplar a muchas otras tecnologías como lo son de cobros, control de accesos, las posibilidades son muchas, dándonos a nosotros la capacidad de decidir en que deseamos y en que queremos mejorar nuestro trabajo utilizando todas las herramientas y tecnologías que tenemos hasta el día de hoy.

HERRAMIENTAS UTILIZADAS EN EL DESARROLLO

Los avances tecnológicos que ha tenido la informática son enormes, dándonos la capacidad de encontrar las herramientas para prácticamente cualquier cosa que deseemos construir, pero una cosa muy importante que debemos tener en cuenta o considerar al momento de usar una herramienta o librería es su licencia, aunque también es imperceptible porque siempre hay empresas u organizaciones además de desarrolladores en todo el mundo que contribuyen al mejoramiento y soporte de herramientas libres, como lo es el caso de OpenCV[1] una librería que contiene miles de algoritmos para visión artificial, y tiene una licencia abierta, es decir que lo podemos utilizar, modificar o usar como nosotros lo necesitemos.

Para la construcción del sistema es necesario poseer conocimientos sólidos en programación, ya que todo el sistema está basado en el lenguaje de programación C++ y la mayoría de herramientas usadas están basadas en él, como el entorno integrado de desarrollo QT Creator. Además de poseer conocimientos en bases de datos MySQL.

OpenCV se integra perfectamente con el lenguaje de programación C++ luego de realizar una compilación al mismo, en conjunto se logran obtener excelentes resultados, por lo que ha llegado a ser una de las librerías más usadas en lo que respecta a visión artificial, dada la

efectividad y resultados obtenidos sobre procesamientos en tiempo real.

Es importante notar que para el desarrollo de este tipo de sistemas el rendimiento siempre es un factor importante y este se obtiene de la integración de las diferentes herramientas[2] y lenguajes de programación, pero cabe recalcar que su importancia radica en saber aplicarlos de la manera correcta y acorde al resultado que se requiere obtener de ellos.

En la Fig. 1 se muestra la interfaz e inicio del entorno integrado de desarrollo integrado [3] utilizado para el desarrollo del sistema de cobro de peajes, en la cual también se muestra los formularios del sistema.

Figura 1. Interfaz principal del IDE QT Creator

Las librerías que contribuyeron en el desarrollo del sistema fueron llamadas como se muestran en la Fig. 2 previamente se debe tener instaladas todas las dependencias y las librerías OpenCV, además de un gestor de base de datos como lo es MySQL para la persistencia de la información generada, cabe indicar que el sistema operativo utilizado para el desarrollo e implementación es Ubuntu 16.04.

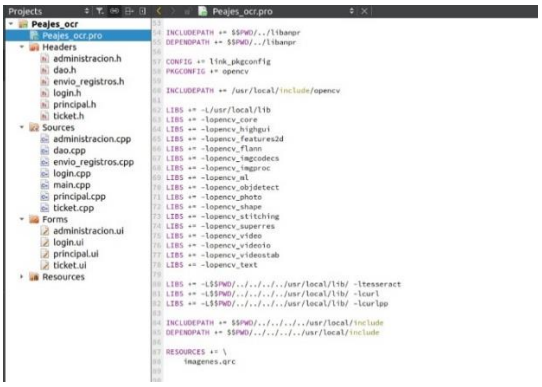


Figura 2. Archivo .pro incluyendo todas las librerías necesarias

IMPLEMENTACIÓN DEL SISTEMA

Para el desarrollo del sistema se utilizó la metodología ICONIX[4], debido a que su enfoque iterativo e incremental permite la integración de nuevas funcionalidades. A continuación se detallan cada una de las etapas correspondientes a la metodología:

a) Análisis de requisitos: Al tratarse el sistema de un prototipo se debió cumplir con los requerimientos base de un sistema de este tipo, es decir se debió recopilar la información necesaria para poder cumplir con este objetivo y de esta manera determinar los requerimientos con los que debe cumplir el sistema. Esto fue posible debido a la documentación bibliográfica y la observación directa.

b) Análisis y diseño preliminar: Luego de haber realizado un primer análisis para el desarrollo del sistema, el siguiente paso es la determinación de los requerimientos. Entre los cuales tenemos los funcionales y no funcionales :

- **Requerimientos Funcionales**
  - Reconocer placas utilizando una cámara.
  - Reconocimiento de Caracteres de la placa detectada (OCR)
  - Mostrar la placa detectada.
  - Generar un cobro según el tipo de vehículo.
- **Requerimientos no funcionales**
  - Poseer una interfaz amigable e intuitiva.
  - Fácil uso por parte del usuario.

c) Diseño: Una vez determinados los requerimientos con los que deberá cumplir el sistema, se procedió a realizar el reconocimiento de todos los elementos que serán parte del sistema. Para ello se construyó el diagrama de casos de usos final en base a los requerimientos y análisis respectivos analizados, del cual partirá todo el sistema, como se muestra en la figura 3.

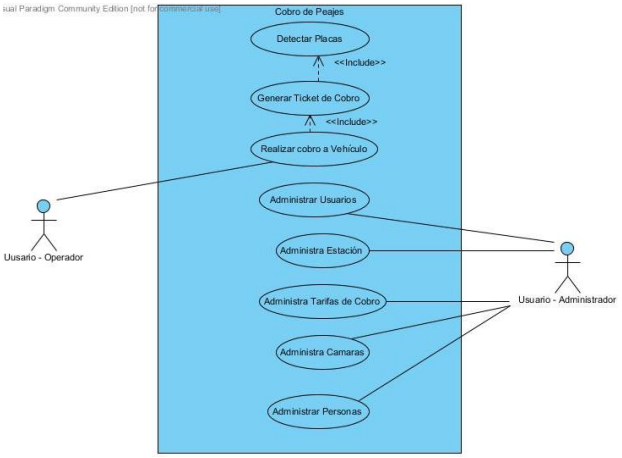


Figura 3. Diagrama de casos de uso

Además de este diagrama se construyeron artefactos de esta fase como son el diagrama de secuencia como lo muestra en la figura 4 y el modelo del dominio de la figura 5.

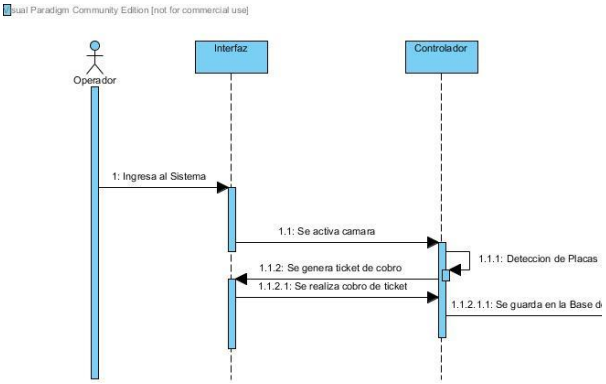


Figura 4. Diagrama de secuencia cobrar ticket

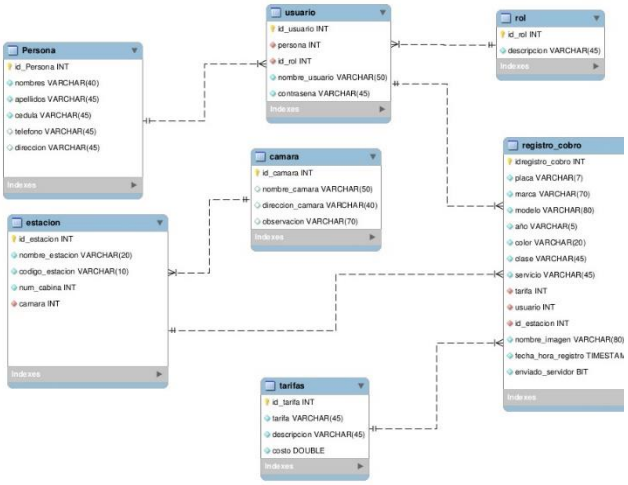


Figura 5. Modelo del dominio

d) Implementación: Con el análisis, el diseño y las funcionalidades del sistema claras, se puede empezar la construcción del software, se siguieron varias fases para la implementación del sistema, las cuales se detallan a continuación.

Integración de las herramientas

Como se mencionó anteriormente, para la construcción del software se usaron varias herramientas, entre las cuales se encuentra principalmente la librería OpenCV. Esta librería se encuentra escrita en el lenguaje de programación C++, pero para poder realizar la integración con el IDE Qt Creator se deben realizar una serie de pasos, y además para también poder utilizarlo en nuestro sistema Ubuntu, que fue el que se utilizó para el desarrollo.

La herramienta que nos permitió realizar la compilación de las librerías fue Cmake, un conjunto de librerías que nos permiten compilar librerías escritas en C++ para ser instaladas en nuestro sistema ya este Linux, Mac o Windows. Además de compilar nos permite integrar los módulos extra de OpenCV Contrib, estos módulos son desarrollados en paralelo, pero no son integrados de manera directa ya que son módulos aun no conocidos, o son desarrollados por la comunidad que mantienen el código.

A continuación en la figura 6 se muestra la interfaz de Cmake con la inclusión de los módulos de OpenCV Contrib.

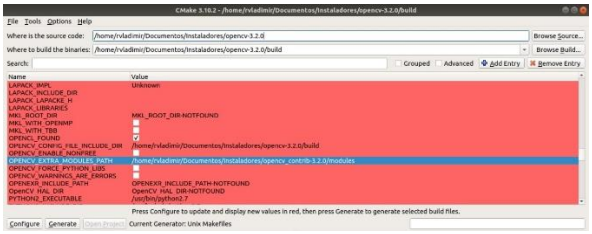


Figura 6. Interfaz de Cmake con path de módulos extra

Clasificador Haarcascade

Una de las herramientas más importantes que provee OpenCV es la posibilidad de crear, entrenar y utilizar un clasificador en cascada [5]. Este clasificador es parte fundamental ya que este contendrá la información necesaria que necesitaremos para realizar la detección de los objetos deseados en una imagen o secuencia de imágenes como lo es en un video. Se pueden utilizar varios clasificadores dentro de un mismo sistema, permitiéndonos ubicar no solo uno sino varios objetos de nuestro interés.

```
cargar_imagen_label();
this->progress->setValue(60);
QThread::msleep(500);
this->progress->setValue(70);
this->lib_api = new Libanpr("/home/ironvlack/Documentos/Tesis/Clasificadores/clasificadorPlacas.xml",
                            "spa","ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789",7,8,7);
this->progress->setValue(80);
QThread::msleep(250);
this->progress->setValue(90);
QThread::msleep(250);
ui->btn_pause->setEnabled(false);
this->progress->setValue(100);
```

Figura 7. Carga del clasificador haarcascade en el proyecto

Para la construcción del clasificador utilizado en este sistema, se utilizaron para el entrenamiento 1200 imágenes positivas que son las que encontrara positivamente en nuestras imágenes, y también se utilizaron 2000 imágenes negativas que caso contrario tomara como objetos erróneos dentro de nuestra escena y no las reconocerá.

Algoritmo de detección

En la figura 8 se puede apreciar los pasos a seguir para el tratamiento y detección de la placa vehicular y seguidamente el proceso para extraer el texto de la imagen[6].

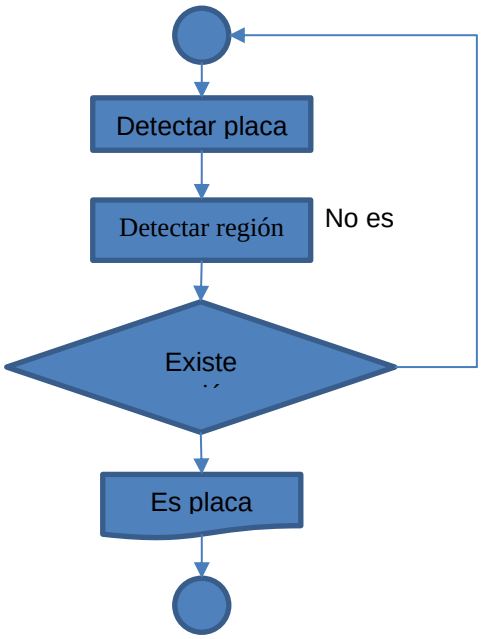


Figura 8. Diagrama de flujo empleado para la detección de placas

Es importante señalar que al ser un sistema en tiempo real se debió manejar los recursos del sistema de una manera adecuada, ya que todos los flujos se ejecutan muchas de las veces en paralelo al realizar procesamientos, y se puede llegar a provocar desbordamientos de memoria o ralentización en el funcionamiento del sistema.

Estos tipo de sistemas por lo general generan gran cantidad de información, y esta debe ser almacenada de alguna manera, por lo que su utilizo una base de datos local que esta sincronizada con una base de datos en la nube lo que permitirá mantener un respaldo de toda la información, como también poner a disponibilidad de los usuarios que deseen acceder a ella a través de un sitio web.

Implementación del código

El sistema está escrito completamente en código de C++, el cual conjúntame con el framework Qt nos permitió crear todo el sistema y los algoritmos necesarios para que este funcione correctamente.

Es importante indicar que dada la metodología y la estructura que se llevó en el proyecto, las funciones base para el OCR y detección de texto así como funciones validadoras de las placas se realizaron en forma de librería, lo que permitirá que esta sea usada en los proyectos que sean necesarios sin comprometer la interfaz o al sistema aquí construido. Esto conlleva varias ventajas como el mejoramiento y optimizaciones posteriores del código y así mismo el mejoramiento de la interfaz o integración de nuevas funcionalidades.

Todos los formularios de administración además del frame principal fueron diseñados con la misma herramienta, lo cual agilito el desarrollo e implementación. Teniendo en cuenta también la organización del código y la estructura del lenguaje mismo, se obtuvo una estructura de fácil comprensión y mantenimiento, a continuación en la figura 9 se presenta la estructura final del proyecto.

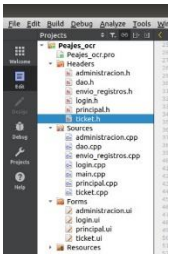


Figura 9. Estructura generada en la estructura del proyecto

Resultados

Una vez concluida las fases de análisis, diseño e implementación se debe realizar la verificación, obteniéndose los siguientes resultados:

1. **Simulación en tiempo real:** En el sur del país es evidente el no encontrar peajes que realicen sus actividades de concesión debido a las regulaciones del gobierno local. En vista de esto se vio la necesidad de construir una maqueta a escala, que nos permita realizar todas las pruebas necesarias. A continuación en la figura 10 se muestra la fase de construcción de la maqueta.

Figura 10. Proceso de construcción de la maqueta a escala

Una vez terminada la construcción de la maqueta queda de la manera como lo indica la figura 11, permitiéndonos posteriormente realizar todas las pruebas necesarias.

Figura 11. Prototipo final de maqueta de peaje

Se utilizaron además vehículos a escala conjuntamente con placas, para realizar las pruebas de funcionalidad.

Figura 12. Vehículo y placa a escala utilizados para las pruebas

2. **Detección de placas:** Evidentemente para la detección de los objetos, en este caso placas cabe indicar que el clasificador implementado es el pilar fundamental de todas las operaciones realizadas posteriormente ya que sin la ayuda de este no se hubiera podido extraer de las imágenes las placas que hubieren encontrado en ella. Teniendo resultados aceptables que corresponden al 95% siendo este un gran resultado, teniendo en cuenta las condiciones de prueba en el prototipo que su tuvieron para realizar las pruebas necesarias.

   Para la obtención de las imágenes se utilizó una cámara de alta definición Logitech C920 de 1080px, que nos permitió acercarnos a la calidad de las cámaras usadas propiamente para los sistemas de reconocimiento de placas vehiculares.

Figura 13. Cámara utilizada en el prototipo de peaje

En el sistema se puede evidenciar la detección de las placas, encerrándola en un recuadro para su fácil ubicación en la pantalla. Para minimizar el impacto y uso de memoria del computador se determinó una zona de interés en la captura de video debido a que un peaje es un entorno controlado y podemos suponer en que zona estará siempre ubicada la placa del vehículo así como lo muestra la figura 14.

Figura 14. Reconocimiento de placa y zona de interes

3. **Generación de cobros:** Con la información obtenida de cada placa se puede realizar una consulta a la página de la ANT donde podemos obtener la información de cada vehículo y así saber a qué tipo pertenece para poder realizar el respectivo cobro.
   Para este proceso se tomó en cuenta la clasificación actual de cada uno de los vehículos existentes, dándole nosotros un costo a cada uno en base al tipo de vehículo al que pertenece, el sistema también mostrara un ticket de cobro en la interfaz del usuario indicando los datos, el tipo de vehículo y el costo a cancelar. Cada uno de estos cobros realizados es almacenado en la base de datos local, para posteriormente ser enviado a la base de datos en la nube.

MÉTODOS UTILIZADOS

En el desarrollo de sistema es siempre importante utilizar esquemas metodológicos que nos permitan utilizar y recopilar toda la información necesaria para la construcción del sistema. Estos esquemas nos permitirán estructurar, organizar y clasificar el contenido que vayamos obteniendo, dándonos grandes posibilidades de éxito al desarrollar sistemas, los métodos que se utilizaron se describen a continuación.

1. **Método científico:** Este método permite obtener los conocimientos necesarios acerca de todo lo relacionado con la construcción del sistema, como fueron librerías de visión artificial, metodologías de desarrollo y estado del arte acerca de los peajes ayudándonos a comprender conceptos y plantear soluciones mediante procedimientos.
2. **Método deductivo:** Los temas y aplicaciones dentro de la visión artificial son extensos, lo que nos permite a través del método deductivo llegar a especificar los requerimientos con los que cumplen los sistemas de reconocimiento de placas y más aún del funcionamiento de los peajes.
3. **Método inductivo:** Este método permite a través de los conceptos encontrados e identificados realizar una complementación con el método deductivo, llevándonos a determinar el impacto que tendrá un sistema de este tipo tanto en las instituciones públicas o privadas que lo llegaran a implementar.

4. **Metodología de desarrollo:** Nos permite obtener una organización y obtener resultados a través del control del ciclo de vida del software creado permitiendo realizar mejoras o inclusión de funcionalidades.

CONCLUSIONES

Una vez terminado el desarrollo del presente artículo es preciso puntualizar algunas conclusiones a las que se llegó luego de haber realizado todos los procesos necesarios para la construcción del sistema, las mismas se describen a continuación:

La metodología ICONIX es importante debido a su robustez y a la agilidad con la que se realizan cada una de las fases. Otro punto importante de haber seguido esta metodología es la documentación que nos deja y que además nos permite realizar posteriores iteraciones para inclusión de nuevos módulos o refactorización y mejora del código ya existente gracias a su estructura modular con la que nos sugiere la construcción del software.

El correcto uso e integración de las librerías y herramientas seleccionadas presentan un factor importante en el desempeño del sistema, ya que depende en gran medida el acople y confiabilidad de que trabajen bien conjuntamente. El construir siempre un clasificador robusto permitirá obtener resultados más confiables, usando siempre una gran cantidad de imágenes positivas en relación a las imágenes negativas asegurara que el clasificador permita obtener más fácilmente las imágenes deseadas.

REFERENCIAS

OpenCV, «Releases,» [En línea]. Available: https://opencv.org/releases.html. [Último acceso: 20 Septiembre 2017].

GitHub, «Tesseract-Ocr,» [En línea]. Available: https://github.com/tesseract-ocr/. [Último acceso: 22 Septiembre 2017].

J. Torres, «Proyecto Qt: Framework de desarrollo de aplicaciones,» 28 Junio 2013. [En línea]. Available: https://medium.com/jmtorres/proyecto-qt-framework-de-desarrollo-de-aplicaciones-2b2f895ac285. [Último acceso: 29 Septiembre 2017].

Ecured, «ICONIX,» [En línea]. Available: https://www.ecured.cu/ICONIX. [Último acceso: 3 Octubre 2017].

M. Carmelo, «Entrenar OpenCV en Detección de Objetos,» 6 Diciembre 2015. [En línea]. Available: http://acodigo.blogspot.com/2015/12/entrenar-opencv-en-deteccion-de-objetos.html. [Último acceso: 20 Septiembre 2017].

M. Elkan, «Umbralización en OpenCV,» 28 Julio 2017. [En línea]. Available: http://acodigo.blogspot.com/2017/07/umbralizacion-en-opencv.html. [Último acceso: 20 Septiembre 2017].

Vladimir Vaca nació en Carchi/Montufar en el año de 1991. Sus estudios primarios los realizo en la escuela Julio María Matovelle de la ciudad de Loja, sus estudios secundarios los cumplió en el colegio San Francisco de Asís de la misma ciudad. Trabajo varios años para la empresa KRADAC como desarrollador de software

Mario Palma nació en Loja. Ingeniero en Sistema Informáticos y Computacionales, graduado en la Universidad Nacional de Loja, actualmente trabaja como docente investigador en dicha universidad. Posee maestrías en: Gerencia y Liderazgo Educacional en la Universidad Técnica Particular de Loja, Desarrollo de aplicaciones para dispositivos móviles, Desarrollo de aplicaciones móviles en Universitat Oberta de Catalunya.

ANEXO V: ANTEPROYECTO

UNIVERSIDAD NACIONAL DE LOJA

PPTT-CIS-IXB-019

“Diseño y construcción de un prototipo para cobro de peajes con Visión Artificial.”

PERFIL DEL PROYECTO DE TRABAJO DE TITULACIÓN

NOVENO CICLO B

Autor:

- [ECINF7683] Vaca-Moscoso, Ramiro-Vladimir

Revisor:

- Ing. Jácome-Galarza, Luis Roberto Mg. Sc

Loja, 26 de Enero de 2016

LOJA - ECUADOR

Índice

Índice de Figuras

Índice de Tablas

A. Tema.

"**Diseño y construcción de un prototipo para cobro de peajes con Visión Artificial**"

B. Problemática.

1. Situación Problemática

En el desarrollo de un país, una de las causas más importantes para que este proceso de crecimiento se vaya consolidando, son sus carreteras y vías de acceso a los diferentes puntos estratégicos del país. El buen estado de las carreteras contribuye al desarrollo turístico, industrial y comercial del país [1], porque la economía viaja a través de sus carreteras, hacia todos los rincones donde exista población.

El caso de nuestro país no es muy diferente al de los demás, las vías y carreteras son un punto importante en el crecimiento de nuestra economía, y aún más las vías que son troncales de paso hacia las regiones de nuestro país, y como también de acceso de otros países.

Por este motivo es importante darle la importancia necesaria a la infraestructura y estado de las carreteras de nuestro país, en los últimos años el gobierno de turno ha hecho una inversión sustancial en el mejoramiento de todas las carreteras y vías dentro de todo el país, pero este sería solo un paso hacia el crecimiento del desarrollo. Una vez construidas y mejoradas las carreteras, no se las puede dejar en el abandono, se les debe realizar un constante mantenimiento y mejora, así como también dentro de los tramos específicos y más afluentes de los recorridos, se debe prestar servicios específicos como servicios de grúa, ambulancia, monitoreo de los vehículos que circulan, conteo de vehículos, etc.

Los peajes dentro de nuestro país fueron dados en concesión a empresas privadas, donde el Estado y el Sector Privado trabajan conjuntamente para un mejor servicio a los usuarios. La concesión de carreteras es un mecanismo implementado exitosamente a nivel mundial que ha permitido una apropiada y necesaria asociación entre el Estado y el sector privado, facilitando la ejecución de obras y la prestación de servicios viales de forma permanente, con una importante inversión en infraestructura de carreteras para beneficio de la economía y bienestar de los usuarios. [2].

Todos los servicios prestados dentro de estos tramos más importantes, requieren de un costo, de alguien que las administre, de personal que esté constantemente al pendiente de todo lo que sucede, por lo que se estableció una tarifa de cobro hacia todos los vehículos que transitan por medio de las vías que contengan peajes.

El cobro de estas tarifas corresponden al tipo de vehículo que circule por la vía, si este es liviano, mediano, pesado, etc. en sus diferentes categorías [3]. Actualmente los

peajes realizan un cobro a los vehículos que circulan de forma manual, y en forma electrónica a través de tarjetas y stickers adhesivos.

Esta manera de realizar los cobros dentro de los peajes, produce la mayoría de veces aglomeración de vehículos, especialmente en feriados y temporada de vacaciones, por la gran movilización que existe en el país. Además de que no todos los usuarios cuentan con una tarjeta o un sticker que les permita pasar por los peajes sin detener su vehículo, por lo que en las cabinas del peaje se realiza un registro y facturación del paso del vehículo ocasionando pérdida de tiempo; tomando en cuenta la gran afluencia de vehículos que se da en estas carreteras.

Al tener este sistema tradicional no se puede establecer un control sobre todos los vehículos que transitan por las carreteras, ya que no se cuenta con un contador de vehículos que nos permita obtener estadísticas de los vehículos que transitan, si estos fueron livianos, pesados, etc. Lo que nos permitirá a través de Visión Artificial e Inteligencia Artificial realizar predicciones acerca de cómo sería el flujo vehicular en determinadas circunstancias o escenarios.

Estas situaciones traen consigo estrés para los usuarios que quieren llegar de una manera rápida y tranquila hacia sus lugares de destino, sin dejar de sentir el respaldo de una empresa a cargo de la seguridad en las vías si se presentara un percance o accidente.

2. Problema de Investigación.

Teniendo en cuenta los problemas causados por las actuales formas de cobro en los peajes, de las diferentes carreteras del país, se plantea el siguiente problema:

> **El sistema de cobro de peajes en las carreteras del país no es el adecuado, generando congestionamiento vehicular e interrupción del tráfico vehicular.**

Teniendo en cuenta el problema anteriormente planteado, se tiene el siguiente problema de investigación:

> *¿Una forma automatizada de cobro en los peajes a través de visión artificial mejorará la circulación vehicular en las carreteras del país?*

C. Justificación.

El presente proyecto a desarrollarse formará parte de una investigación dentro del ámbito universitario, que será de gran interés para muchos usuarios en general, y en específico al sector que cuenta como medio de transporte un vehículo, generando alternativas para mejorar el buen vivir de la sociedad, brindando mejores soluciones a los problemas de transporte que se encuentran en el día a día tanto en la ciudad de Loja como en el país, mejorando la comunicación entre ciudades y acelerando el proceso de desarrollo del país.

El proyecto tendrá gran impacto en la sociedad, tanto para el transporte público como el privado. El transporte público será beneficiado de manera sustancial, por lo que sus usuarios optarán más por sus servicios, por la rapidez y calidad del mismo. El transporte privado de igual manera tendrá los mismos beneficios, existen productores con flotas vehiculares que podrán sacar a tiempo sus productos y expenderlos. Los usuarios también reducirán el tiempo de traslado de un lugar a otro, se disminuirá la congestión vehicular, sin dejar de tener un servicio de calidad con atención permanente 24 horas del día de los 7 días de la semana; y soporte a los diferentes inconvenientes que se puedan presentar durante los viajes.

Como estudiantes y futuros profesionales de Ingeniería en Sistemas, adquirimos conocimientos durante todo el desarrollo de nuestra vida estudiantil, y está en nuestras manos aprovechar el conocimiento adquirido para mejorar la sociedad donde vivimos, solventando los diferentes problemas que existen y se detectan, solucionando los problemas a través de soluciones tecnológicas e innovadoras.

La Visión Artificial pertenece a la nueva malla curricular de la Carrera de Ciencias de la Computación que ofrecerá la Universidad Nacional de Loja próximamente, dejando ya un previo adelanto de conocimiento en esta asignatura. Además de poder ser participe y aportar con el desarrollo de una investigación que incrementará más nuestros conocimientos, nos dará una experiencia enriquecedora para cuando nos desarrollemos en nuestra vida profesional.

El proyecto se desarrollará aplicando las nuevas tecnologías que existen en el mundo, aportando mayores y mejores beneficios al momento de su implementación o puesta en marcha.

Las tecnologías crecen a cada día, como es el caso de librerías para Visión Artificial donde constantemente se mejoran los algoritmos de detección de patrones, y el caso de las cámaras que cada vez son más potentes y prestan una mejor resolución de imágenes que hará que el procesamiento a través del software sea más eficiente.

Las mejoras e implementaciones tecnológicas ayudan en gran parte al planeta y a su mejor conservación. Al disminuir el uso de papel al momento de facturar y emitir recibos, se está ayudando a la conservación del medio ambiente; también se beneficia al ahorro energético porque una cámara consumiría mucho menos recursos que una computadora, impresora, brazos mecánicos, etc (todos estos juntos generan un gran ahorro).

Por ser un proyecto previo a obtener el título profesional, el autor del proyecto solventará todos los gastos necesarios que se presenten durante el desarrollo del prototipo y las pruebas correspondientes en campo.

D. Objetivos.

1. Objetivo General.

Diseñar y construir un prototipo de cobro de peajes a través de Visión Artificial, mediante cámaras que detectan los vehículos en movimiento que transiten a través de las carreteras.

2. Objetivos Específicos.

- Analizar el estado del arte de los peajes en nuestro país, para determinar los requerimientos que construirán el software.
- Configurar y aplicar técnicas de Visión y Reconocimiento de Vehículos con OpenCV
- Desarrollo del módulo para el cobro de peajes según el tipo de vehículo.
- Pruebas de funcionalidad en tiempo real del prototipo.

E. Alcance.

El proyecto se desarrollará en un tiempo estimado de 6 meses, durante los cuales se prevé concluir todos los objetivos planteados, se ha planteado el desarrollo de un prototipo de software que permitirá a través de cámaras detectar los vehículos en movimiento que circulen por las carreteras, las cuales que cuenten con cabinas de peaje, para su detección y cobro respectivo según sea el tipo de vehículo.
Se han planificado cuatro fases para el desarrollo del proyecto, cada una basada en los objetivos específicos planteados para el desarrollo del proyecto, a continuación se detalla las fases y tareas:

Fase 1: Analizar el estado del arte de los peajes en nuestro país, para determinar los requerimientos que construirán el software.

1.1 Búsqueda de información a nivel nacional.
1.2 Funcionamiento de los peajes.
1.2.1 Ubicaciones estratégicas de los peajes en el país.
1.2.2 Concesiones y tarifas de los peajes
1.3 Determinación del estado actual de los peajes

Fase 2: Configurar y aplicar técnicas de Visión y Reconocimiento de Vehículos con OpenCV.

2.1 Cámaras para detección de vehículos
2.1.1 Cámaras de alta resolución
2.1.2 Adaptabilidad y streaming a computadores
2.1.3 Ubicación estratégica de las cámaras
2.2 Librerías para Visión Artificial
2.2.1 Análisis de librerías
2.2.2 OpenCV
2.2.3 Funciones y ventajas de OpenCV
2.3 Lenguajes de Programación
2.3.1 Elección del Lenguaje de Programación
2.3.2 Elección del Entorno Integrado de Desarrollo - IDE
2.3.3 Protocolos para comunicación y streaming de video
2.3.4 Recepción y análisis de streaming mediante el lenguaje de Programación.

2.4 Integración de todos los componentes

Fase 3: Desarrollo del módulo para el cobro de peajes según el tipo de vehículo.

3.1 Arquitectura del Software

3.1.1 Elección de una Metodología de Desarrollo

3.1.2 Patrones de Desarrollo

3.2 Implementación

3.2.1 Desarrollo del módulo para detección de vehículos

3.2.2 Desarrollo del módulo para cobro de tarifas

Fase 4: Pruebas de funcionalidad en tiempo real del prototipo.

4.1 Pruebas de Funcionamiento

4.1.1 Detección de Vehículos

4.1.2 Cobro de Tarifa según Vehículo

4.2 Pruebas en tiempo real

.

F. Metodología.

El presente trabajo de titulación utilizará métodos esenciales y específicos para su elaboración, siendo estos fundamentales para la sustanciación y desarrollo del mismo. Desde el planteamiento del tema hasta su finalización, se utilizaron los siguientes métodos y técnicas:

Métodos.

El **método deductivo** se utilizó para encontrar los problemas a resolver dentro del campo vehicular, específicamente en cobro de peajes; para lo cual nos referimos a varias temáticas generales correspondientes a problemas causados por el actual sistema de cobro de tarifas. Posteriormente se hizo la elección del problema particular: “Diseño y construcción de un prototipo para cobro de peajes con Visión Artificial”.

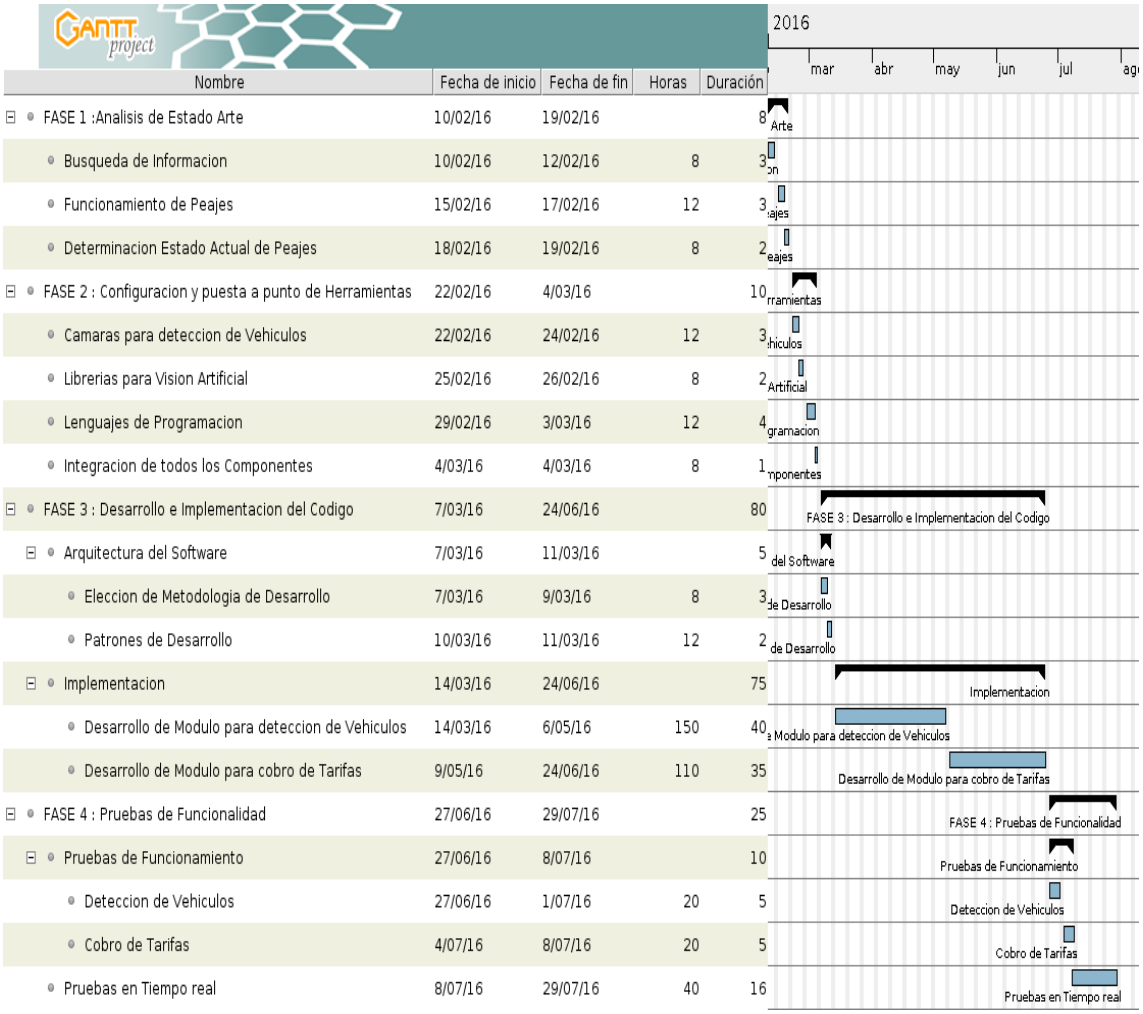
El **método inductivo** parte del tema ya definido y se lo utiliza para la búsqueda de información en el desarrollo del marco teórico e indagación de posibles soluciones en la elaboración del posterior trabajo de titulación.

El **método científico** ayudó a través de un proceso sistemático y planificado la obtención de los objetivos, ya que también el proyecto forma parte de una investigación.

Técnicas.

La **técnica de revisión bibliográfica** sirvió para obtener toda la información correspondiente e importante para el proyecto, obteniéndola de fuentes bibliográficas confiables como: libros, documentaciones oficiales e información confiable de internet.

G. Cronograma.



Nombre	Fecha de inicio	Fecha de fin	Horas	Duración
FASE 1 :Analisis de Estado Arte	10/02/16	19/02/16		8
Busqueda de Informacion	10/02/16	12/02/16	8	3
Funcionamiento de Peajes	15/02/16	17/02/16	12	3
Determinacion Estado Actual de Peajes	18/02/16	19/02/16	8	2
FASE 2 : Configuracion y puesta a punto de Herramientas	22/02/16	4/03/16		10
Camaras para deteccion de Vehiculos	22/02/16	24/02/16	12	3
Librerias para Vision Artificial	25/02/16	26/02/16	8	2
Lenguajes de Programacion	29/02/16	3/03/16	12	4
Integracion de todos los Componentes	4/03/16	4/03/16	8	1
FASE 3 : Desarrollo e Implementacion del Codigo	7/03/16	24/06/16		80
Arquitectura del Software	7/03/16	11/03/16		5
Eleccion de Metodologia de Desarrollo	7/03/16	9/03/16	8	3
Patrones de Desarrollo	10/03/16	11/03/16	12	2
Implementacion	14/03/16	24/06/16		75
Desarrollo de Modulo para deteccion de Vehiculos	14/03/16	6/05/16	150	40
Desarrollo de Modulo para cobro de Tarifas	9/05/16	24/06/16	110	35
FASE 4 : Pruebas de Funcionalidad	27/06/16	29/07/16		25
Pruebas de Funcionamiento	27/06/16	8/07/16		10
Deteccion de Vehiculos	27/06/16	1/07/16	20	5
Cobro de Tarifas	4/07/16	8/07/16	20	5
Pruebas en Tiempo real	8/07/16	29/07/16	40	16

Figura 9 Cronograma del PPT - Prototipo para cobro de peajes con Visión Artificial

La definición de la columna **duración** hace referencia a los días para cada una de las actividades y de los hitos, la suma de los días da un total de 120 días que cumplen con los 6 meses para realizar este proyecto.

La definición de la columna **horas** es la duración en horas de cada una de las actividades, la suma en horas de cada uno de los hitos da un total de **400 horas**. Cabe recalcar que hay actividades que se hacen en paralelo, por ello es que si se sumase todas las horas en separado no daría un resultado total de 400 horas; es por ello que en los hitos ya está considerado las actividades paralelas entre sí.

H. Presupuesto y Financiamiento.

Todos los recursos utilizados durante la realización del proyecto serán asumidos por el autor, a excepción de los gastos por tutor que serán asumidos por la Universidad.

1. TALENTO HUMANO.

Tabla 51 Talento humanos a disponer para el desarrollo del PFM

ROL	NÚMERO DE HORAS	NÚMERO DE HORAS($)	VALOR TOTAL ($)
Estudiante	120	5.00	600.00
Tutor	200	10.00	2000.00
		Total ($):	**2600.00**

2. BIENES.

Tabla 52 Bienes para el desarrollo del PFM

DETALLE	CANTIDAD	VALOR TOTAL ($)
Computador	1	500.00
Impresora	1	50.00
Cámara	1	400.00
Materiales (Cables)	1	20.00
	Total ($):	**970.00**

3. SERVICIOS.

Tabla 53 Servicios utilizados en el desarrollo del PFM

SERVICIO	VALOR($)	VALOR TOTAL ($)
Internet	0.60	200.00
Transporte	0.30	32.00
Impresión a color	0.10	100.00

Impresión B/N	0.05	50.00
Copias	0.02	40.00
Empastados	20.00	80.00
Anillados	1.50	10.00
	Total ($):	**51 2.00**

4. IMPREVISTOS.

Tabla 54 Bienes para el desarrollo del PFM

DETALLE	**VALOR TOTAL** ($)
Transporte	80.00
Materiales(Cables)	5.00
Otros	40.00
Total ($):	**85.00**

Costo Total del proyecto: $ 4167.00 USD

5. Financiamiento

El costo total del proyecto es de $ 4167.00, la universidad otorga la dirección del Trabajo de Titulación, por lo tanto el coste que el estudiante asume es del $ 2167.00 USD

I. Bibliografía.

[1] Panavial: Quienes Somos. [Online]. Available: http://panavial.com/quienes-somos-acerca-de-panamericana-vial-panavial-quito-ecuador.php. [Accessed: 21-Dec-2015].

[2] Panavial: La Concesión . [Online]. Available: http://panavial.com/concesion-administracion-red-autopistas-vias-carreteras-panamericana-vial-ecuador.php. [Accessed: 21-Dec-2015].

[3] Panavial: Tarifas . [Online]. Available: http://panavial.com/estaciones-de-peaje-panavial-administracion-red-autopistas-vias-carreteras-panamericana-vial-ecuador.php?tablajb=estaciones_de_peaje&p=24&t=Tarifas& .[Accessed: 21-Dec-2015].

[4] IBM: Transporte Inteligente. [Online]. Available: http://www-05.ibm.com/services/es/bcs/pdf/transporte-inteligente-como-mejorar-la-movilidad-en-las-ciudades.pdf . [Accessed: 25-Dec-2015].

[5] Department Transportation: New York. [Online]. Available: https://www.dot.ny.gov/index. [Accessed: 25-Dec-2015]

[6] Conorte.S.A [Online]. Available: http://www.conortesa.com/w/ .[Accessed: 25-Dec-2015]

[7] Telectronica: Cámaras IP Axis validan vehículos en peajes argentinos [Online]. Available: http://telectronicapeaje.com/camaras-ip-axis-validan-vehiculos-en-peajes-argentinos/ [Accessed: 25-Dec-2015]

[8] OpenCV [Online]. Available: http://opencv.org/ [Accessed: 25-Dec-2015].

J. Anexos.

ANEXO I: Árbol del problema

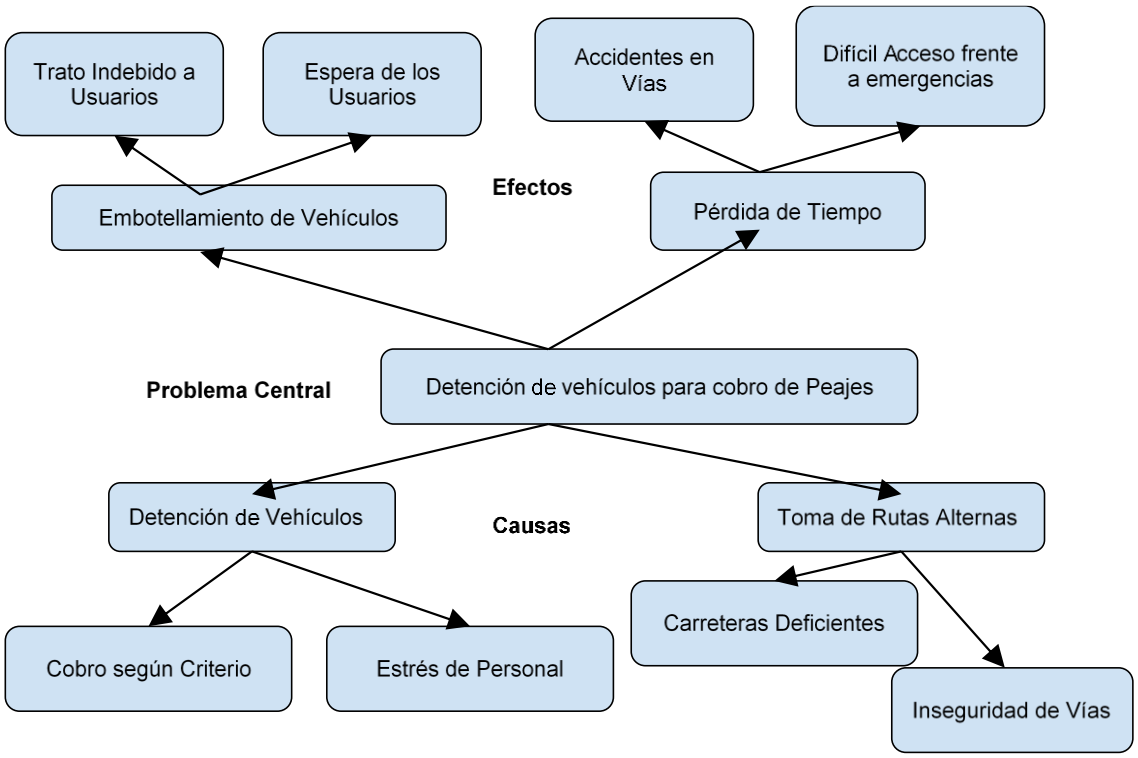
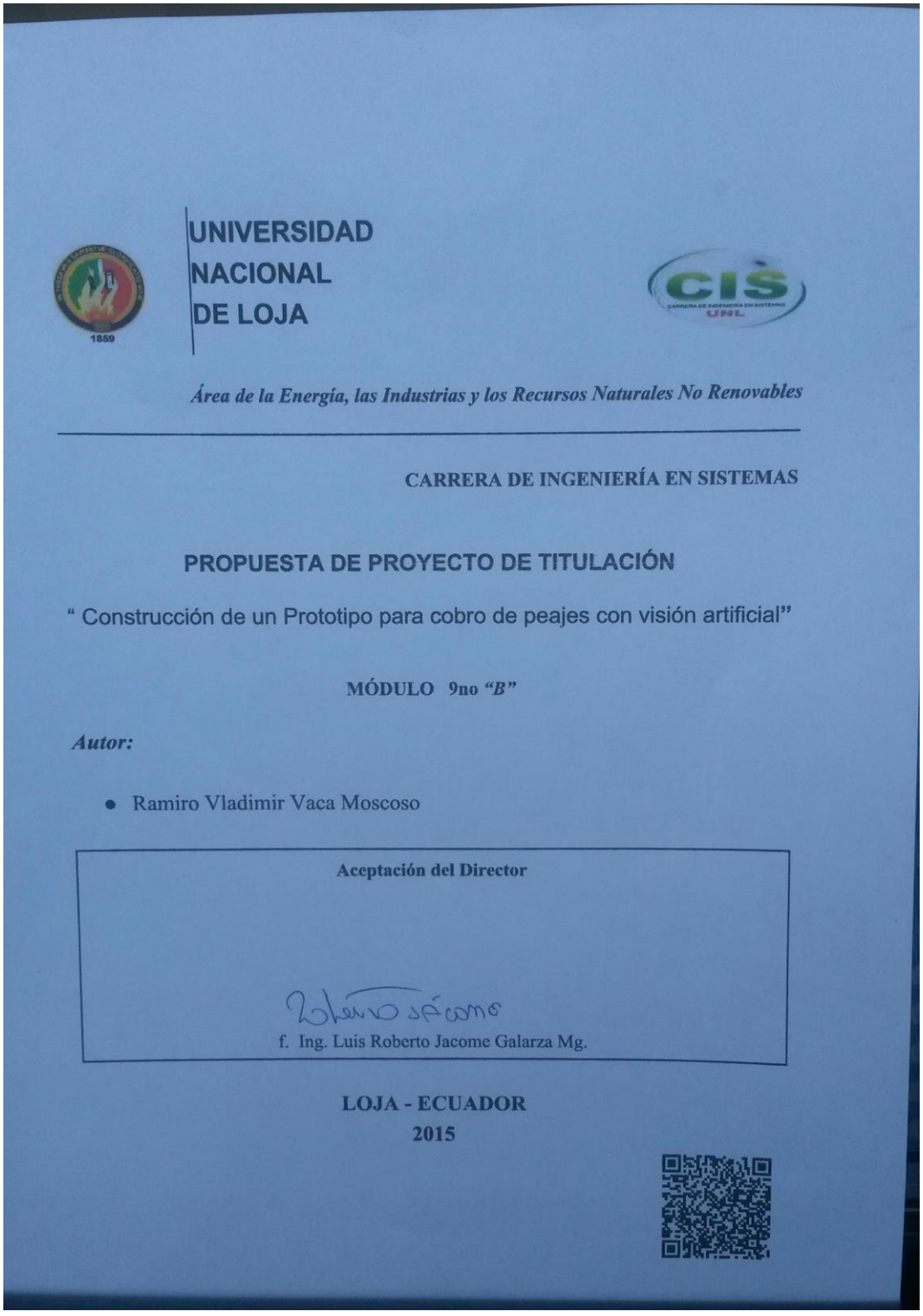


Figura 10 Árbol del Problema

ANEXO II: Propuesta



UNIVERSIDAD
NACIONAL
DE LOJA

1859

CIS
CARRERA DE INGENIERÍA EN SISTEMAS
UNL

Área de la Energía, las Industrias y los Recursos Naturales No Renovables

CARRERA DE INGENIERÍA EN SISTEMAS

PROPUESTA DE PROYECTO DE TITULACIÓN

" Construcción de un Prototipo para cobro de peajes con visión artificial"

MÓDULO 9no *"B"*

Autor:

- Ramiro Vladimir Vaca Moscoso

Aceptación del Director

f. Ing. Luis Roberto Jacome Galarza Mg.

LOJA - ECUADOR
2015

Figura 11 Propuesta firmada del proyecto de trabajo de titulación

I. INTRODUCCIÓN

El documento tiene como finalidad dar a conocer la propuesta que se planteó para la realización del trabajo de titulación, la elección del tema se realizó mediante la exposición de ofertas para trabajo de titulación expuestas por los docentes de la carrera de ingeniería en sistemas, donde se dieron pautas y temas específicos para su elaboración.

Una vez elegido el tema a desarrollar se utilizó técnicas de elección para poder tener una idea más clara del tema propuesto, por lo cual se desarrolló los diagramas de investigación: reticular y un árbol de relevancia en los cuales se aborda el contenido concerniente a los sistemas de transporte inteligente.

En el presente documento se detallarán aspectos de importancia como: el propósito, objetivos y resultados que se esperan en el desarrollo del trabajo de titulación, por lo que se aguarda una adecuada elección del respectivo director, que se encuentre inmerso en los conocimientos requeridos para la elaboración correcta del Trabajo de Titulación.

II. TÉCNICAS DE ELECCIÓN

A. Mapa de Investigación

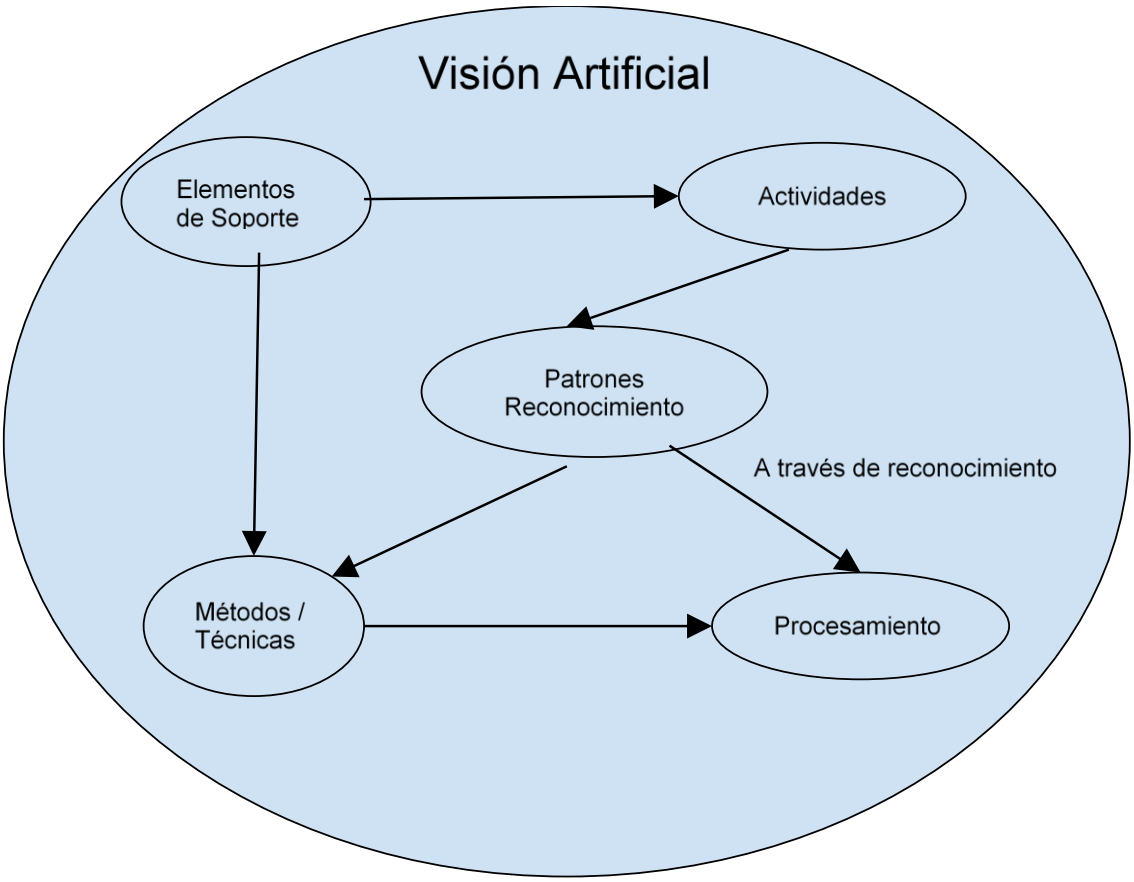


Figura 12 Mapa de investigación de la Visión Artificial

Los Sistemas de Visión Artificial permiten realizar actividades de reconocimiento con patrones predefinidos o adaptados a las necesidades requeridas. Estas actividades se realizan a través de la aplicación de Patrones de Reconocimiento con métodos y técnicas que facilitan el análisis, a través de elementos de soporte como: cámaras, sistemas de red, etc.

B. Árbol de relevancia

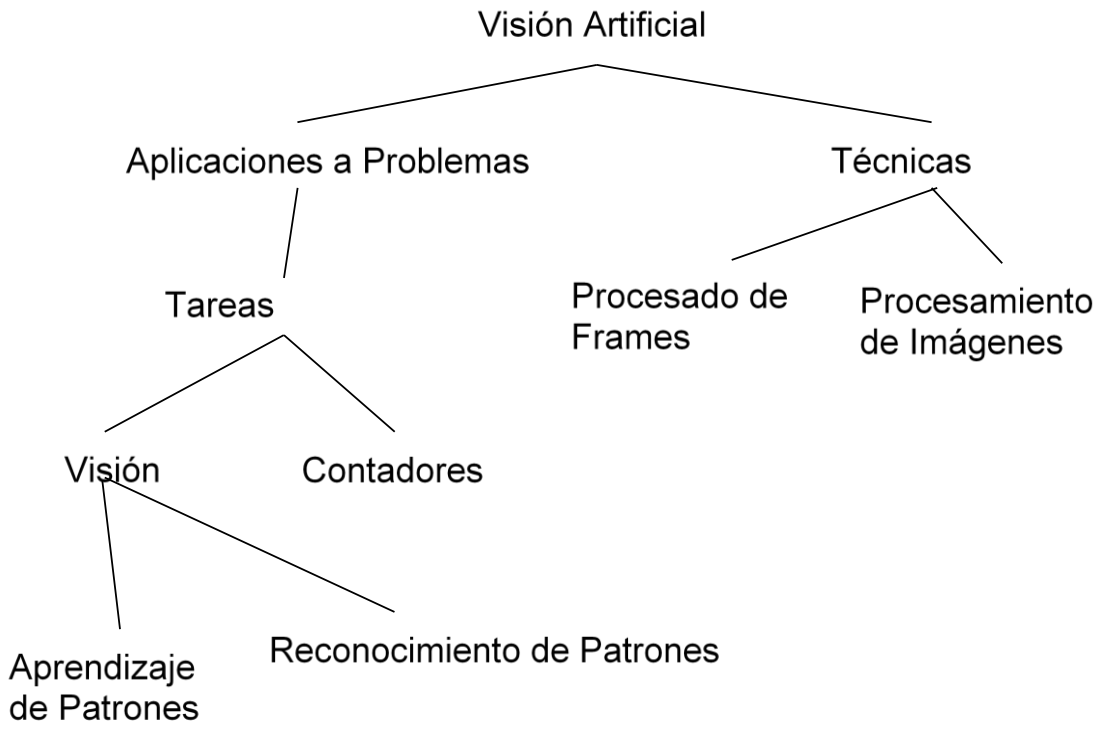


Figura 13 Árbol de relevancia de la Visión Artificial

En este gráfico se presenta como por medio de las aplicaciones de la visión artificial se pueden resolver problemas a tareas genéricas, como reconocimiento de personas, tráfico, etc. A través de estas aplicaciones podemos aprender a reconocer patrones, los que nos van a servir para toma de decisiones inteligentes; además, también de poder realizar conteos y recolección de datos de placas vehiculares, etc.

Por lo que se puede observar la visión artificial puede tener muchas aplicaciones que benefician a la sociedad, y en este caso en específico al tráfico de una ciudad.

III. PROPUESTA

a. Título

"Construcción de un prototipo para cobro de peajes con visión artificial"

b. Tipo de Proyecto

Desarrollo - Investigación - Tema Ofertado por la CIS

c. Propósito y Objetivos

Propósito.

Brindar a través de visión artificial una solución para el cobro de peajes, dadas las circunstancias del tráfico en un punto, para optimizar el tráfico y evitar las congestiones por la tardanza en cobros de forma manual.

Objetivos.

Objetivo General

Reconocer Vehículos en movimiento con visión artificial, para cobro de peajes.

Objetivos Específicos.

- Analizar y aplicar técnicas de visión y reconocimiento de objetos con OpenCV, para detección de vehículos.
- Desarrollar y probar en tiempo real el módulo para la detección de vehículos en movimiento.
- Adaptar el módulo a las necesidades requeridas para un cobro de peajes.

d. Resultados Esperados

Lograr la detección de vehículos a través de técnicas de reconocimiento de vehículos, y realizar un cobro adecuado de peajes utilizando las librerías OpenCV.

e. Metodología de Investigación

Estudio de Casos, investigación de bibliografía.

f. Elección del director

El presente tema se eligió de todas las propuestas realizadas por algunos docentes de la Carrera de Ing. en Sistema y varias propuestas de la UTI, por lo que específicamente se lo escogió de las propuestas realizadas por el Ing. Roberto Jácome y va enfocada como parte de una investigación y por ser parte de la misma. El ing. Roberto Jacome sería el director elegido para el desarrollo del futuro trabajo de titulación.

El Ing. Roberto Jácome ha dirigido alrededor de 30 trabajos de titulación, y ha tenido excelentes resultados en la dirección y tutoría de sus trabajos que han sido de gran ayuda en la investigación en las que se encuentra formando parte, algunos han tenido gran aceptación que se han logrado vender.

g. Referencias

[1] Opencv, Open Source Computer Vision, Disponible en la web: http://opencv.org

[2] IBM Institute for Business Value, Cómo mejorar la movilidad en las ciudades, Disponible en la Web: http://www-05.ibm.com/services/es/bcs/pdf/transporte-inteligente-como-mejorar-la-movilidad-en-las-ciudades.pdf

[3] Domingo Mery ,Departamento de Ciencia de la Computación, Universidad Católica de Chile, 2004. Disponible en: http://web.ing.puc.cl/~dmery/Prints/Books/2004-ApuntesVision.pdf

[4] Enrique Onieva Caracuel,Departamento de Ciencias de la Computación e Inteligencia Artificial, UNIVERSIDAD DE GRANADA, Disponible en: http://digital.csic.es/bitstream/10261/38851/1/TESIS%20Enrique%20Onieva.pdf

ANEXO III: Acta - Historial de revisiones

	UNIVERSIDAD NACIONAL DE LOJA CARRERA DE INGENIERÍA EN SISTEMAS	**FORMULARIO** PRESENTACIÓN DE ACTAS
		UNL-CIS-001-2016
		Fecha: 11/01/2016

Tema de la Reunión: REVISIÓN DEL PERFIL DE TRABAJO DE TITULACIÓN		
Lugar: Cubículo del docente de la CIS del AEIRNNR, ubicado en el bloque 3 y bloque 5	**Área o Dependencia que convoca:** Software/Telecomunicaciones/Inteligencia/Visión Artificial Artificial/Trabajo de Investigación	
ACTA No.: 001	**Fechas de Revisiones:**	
Orden de Revisiones: 1. Revisión de Perfil de Proyecto 2. Revisión de Planteamiento Anteproyecto 3. Revisión de Correcciones Situación Problemática 4. Revisión de Correcciones Metodología 5. Revisión General		**CITADOS:** • Ing. Roberto Jácome • Vladimir Vaca
Desarrollo: • Formular correctamente la problemática encontrada y sus problemas, mejorar la redacción y justificar con evidencias cada problema. • Aprobación sin inconveniente alguno de la propuesta. • Anexar el acta de revisiones del perfil del proyecto de trabajo de titulación.		
Anexos: • Revisión del perfil de trabajo de titulación		

f. Ing. Luis Roberto Jácome Galarza **Docente CIS**	**f. Ramiro Vladimir Vaca Moscoso** **Estudiante CIS**

ANEXO IV: Licencia

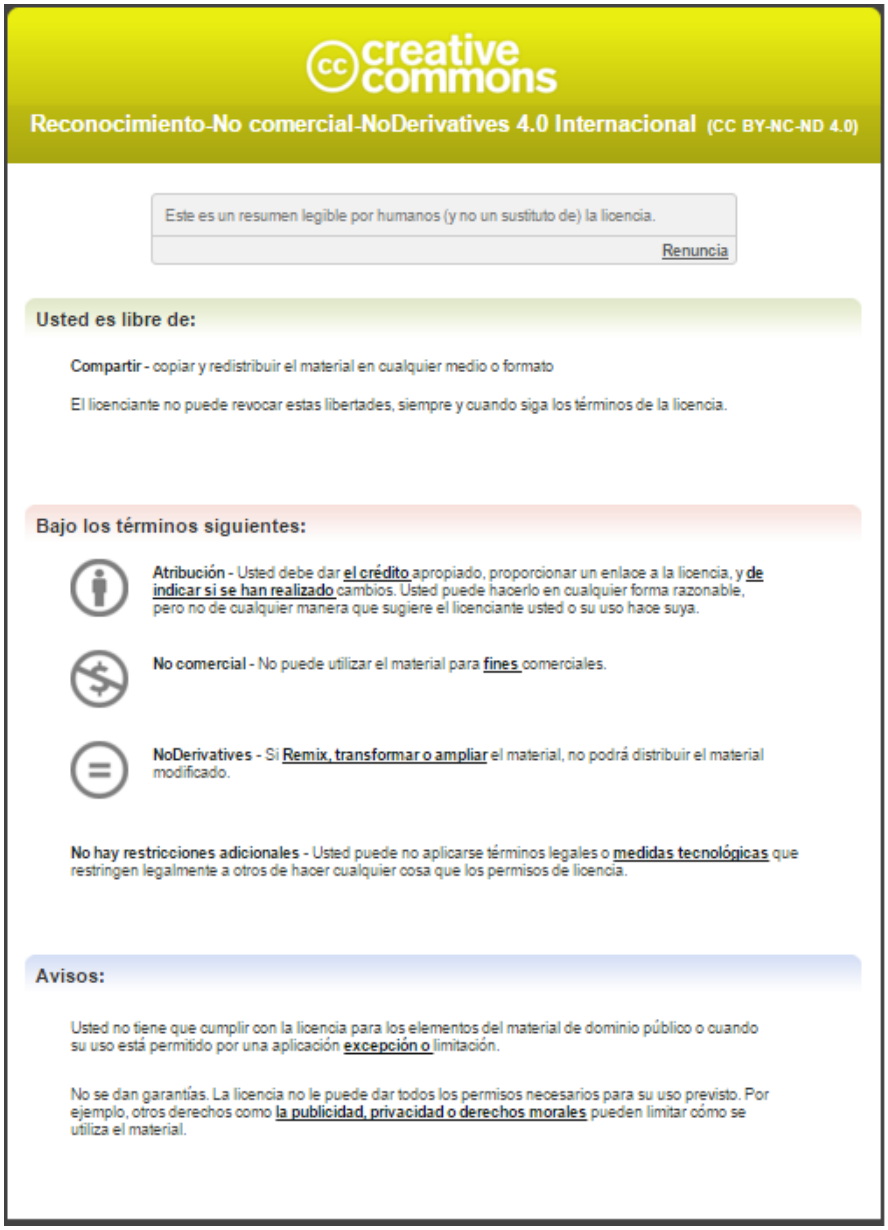


Figura 14 Licencia Creative Commons, Fuente: http://creativecommons.org/licenses/by-nc-nd/4.0/