ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE PROTOTIPO DE UN SISTEMA PARA COMPARTIR CARRIL DE LA METROVÍA CON VEHÍCULOS LIVIANOS APLICANDO UNA TARIFA, CON EL PROPÓSITO DE DISMINUIR LAS INVASIONES AL CARRIL EXCLUSIVO"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO/A EN ELECTRÓNICA Y TELECOMUNICACIONES

ARNALDO ANDRÉ ABAD GÓMEZ

LISSETTE ESTEFANIA SALAS GUERRERO

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTO

En primer lugar, un agradecimiento especial a Dios por haberme permitido llegar hasta este punto, demostrando su guía a pesar de cualquier adversidad u obstáculo presentado durante toda mi vida, a mi familia, hermanas (Angeoline y Valentina), abuelos (Ignacio e Hilda), tías (Melanie, Patricia y María) y amigos, sin ellos tal vez no sería la persona que soy realmente ahora. Ayudándome en momentos difíciles, siendo mi pañuelo de lágrimas, dándome consejos y alegría mediante bromas y risas causadas por las anécdotas que se nos han presentado pero que las hemos podido superar juntos.

Agradecido a la vez por los valiosos profesores que durante mi trayecto por la vida universitaria encontré y fueron fuente de inspiración, en especial a nuestro profesor colaborador, el Ing. Vladimir Sánchez y al profesor de la materia integradora el Ing. Ronald Ponguillo que pusieron su confianza en nosotros para realizar este proyecto, dándonos apoyo con temas que nos resultaban desconocidos y alentándonos para finalizar la carrera de obtener la tan anhelada mención de ser Ingeniero.

Arnaldo Abad

Agradezco principalmente a Dios por darme la vida y por guiarme con paso firme en cada uno de mis propósitos, a mis padres, por cada uno de sus esfuerzos para que mis hermanos y yo siguiéramos adelante en nuestros estudios, por su dedicación a nosotros y por siempre brindarme el apoyo, cariño y confianza necesaria para siempre levantarnos de cada tropiezo y no convertirlo en fracaso.

A mi novio por ser mi compañero, mejor amigo, quien me acompaña en todo momento, por su paciencia, colaboración y motivación durante toda mi carrera, por siempre estar cuando lo necesito, a mis hermanos por brindarme su apoyo incondicional, a mi hija por ser mi inspiración para lograr llegar lejos, a mi mejor amiga Gabriela Paredes por su confianza y motivación, al Ing Ronald Ponguillo y al Ing. Vladimir Sánchez por su guía y apoyo en el proyecto hasta el último momento.

Lissette Salas

DEDICATORIA

Dedico este logro a todas las personas que han existido durante mi vida, estén estas vivas o muertas. A los que han hecho poco o mucho para que esté donde estoy.

En especial y no quiero dejar de nombrarlos a Ivan Cedeño, Pedro Solís y Cristopher Terranova, un grupo especial de amigos, que, aunque la Universidad no la hayamos podido cursar juntos al mismo nivel como en el colegio, fue como si estuvieran allí siempre en cada materia que veía. Apuntes, ayuda y conocimiento fueron lo más preciado que me pudieron dar y por eso quedo eternamente agradecido.

Arnaldo Abad

Este proyecto va dedicado principalmente a Dios y a mis padres Fernando Salas y Gloria Guerrero, por su gran esfuerzo, guianza y educación.

A mi novio Kevin Astudillo, mi hija Luciana, mis hermanos Steve, Mafer y Mishell por su apoyo, confianza y disponibilidad en todo momento.

Y a todas las personas que pusieron un granito de arena para ayudarnos ya sea emocionalmente como con sus conocimientos.

Lissette Salas

TRIBUNAL DE EVALUACIÓN

Mgtr. Ronald Ponguillo	**Mgtr. Vladimir Sánchez**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Lissette Estefania Salas Guerrero

Arnaldo André Abad Gómez

RESUMEN

El proyecto consiste en implementar un prototipo de sistema de control tarifado para el acceso al carril de la metrovía mediante el reconocimiento de placa a través de una cámara, algoritmos de procesamiento de imágenes y una base de datos con el estado de los permisos.

Se inicia con la verificación de los permisos mediante la captura de una fotografía del vehículo que hace uso del carril, se procesa la imagen y se busca los datos asociados a la placa en la plataforma MySQL, para luego dependiendo del estado del permiso, habilitar la creación de una multa y su respectiva notificación vía correo electrónico y mensaje de texto, al propietario del vehículo.

Este proyecto se divide en 4 etapas:

Detección del vehículo en el carril de la metrovía.

El objetivo de esta parte del proyecto es detectar al vehículo que está haciendo uso del carril de la metrovía por medio de un circuito infrarrojo, conformado por un emisor y receptor, los mismos que se activan al momento en que el vehículo interrumpe su comunicación continua, enviando una señal a la Raspberry.

Captura de la imagen y extracción de la placa del vehículo

Con la señal emitida en la etapa anterior, la cámara Raspberry Pi se activa y captura la respectiva imagen, la misma que pasa por una serie de algoritmos que extraen la placa del vehículo.

Búsqueda de los datos asociados a la placa en la base de datos

Una vez que se tiene la placa en texto, inicia la búsqueda de la misma en la base de datos, con ayuda de algoritmos de búsqueda en lenguaje Python, que leen las fechas de caducidad de los permisos en la plataforma y las compara con la fecha actual.

Validación de multas y notificaciones.

Una vez verificado el estado del permiso se procede a borrar la imagen en caso de que el permiso se encuentre vigente, caso contrario se guarda la imagen con la hora y fecha de invasión inscrita, y se procede a enviar automáticamente vía correo electrónico y mensaje de texto la respectiva multa al propietario del vehículo infractor.

ÍNDICE GENERAL

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema.

Uno de los problemas más comunes en la ciudad de Guayaquil es la intensidad de tráfico en las zonas más concurridas, que a medida que la población aumenta ésta se intensifica, provocando que los ciudadanos no logren llegar a tiempo a su lugar de destino o un sin número de infracciones, que no solo nos producen males, sino también nos afectan económicamente.

A medida que el tiempo pasa se han optado por un sin número de soluciones a este problema, pero no han sido del todo efectivas, una de ellas fue la implementación de la metrovía que buscaba proporcionar un transporte rápido para los ciudadanos, ya que ésta cuenta con su propio carril, el mismo que ha sido invadido por muchas ocasiones por vehículos que buscan evitar el intenso tráfico.

La invasión cada vez fue creciendo por lo que la Autoridad Municipal de Tránsito (ATM), optó por crear multas por invasión de carril utilizando la misma tecnología de los radares, pero en vista de que estas invasiones no disminuían en un gran porcentaje su valor fue variando y en la actualidad está en $366, lo mismo que representan un salario básico en nuestro país.

Así mismo otra solución para evitar estas invasiones fue quitarles la exclusividad a los carriles de la metrovía en las zonas más concurridas como por ejemplo el carril de la Estación del Colegio 28 de mayo, el mismo que al principio era compartido por ciertos horarios y luego exclusivo en todo momento.

La mayoría de los carriles son exclusivos, pero a medida que el tráfico se intensifica, los vigilantes se han visto en la necesidad de permitir el paso de los vehículos, logrando disminuir la congestión vehicular en ciertas ocasiones y en otras intensificando el mismo, y consigo derrumbando el objetivo de la metrovía,

es decir, proporcionar un transporte rápido para los ciudadanos que la utilizan, por la congestión que se produce en toda la vía.

Estas medidas nos llevan a diseñar un proyecto donde el acceso a dicho carril sea de uso compartido entre la metrovía y los vehículos livianos que quieran acceder al servicio por una tarifa, que les permitirá el acceso por un tiempo definido, donde ya ni los vigilantes tengan la potestad para permitir el paso por el carril, ya que contará con un sistema que se mantendrá funcionando durante las 24 horas del día, el mismo que se encargará de verificar el estado de los permisos de todos los vehículos que circulan por este carril y notificando al propietario del vehículo de cualquier multa obtenida con su respectiva fotografía, como muestra de su infracción.

1.2 Justificación

La adquisición de vehículos en Guayaquil se hace cada vez más evidente, ya que el tráfico en algunas zonas no es intenso solo en horas picos, sino durante cualquier hora del día, lo que ha producido un incremento excesivo en el número de infractores por invasión al carril exclusivo de la metrovía.

De acuerdo a una publicación del Telégrafo [1], del 16 de julio del 2016, la Autoridad de Tránsito Municipal proporcionó los datos de multas por invasión del carril de la Metrovía, de agosto del 2015 hasta junio del 2016, mostrando que en ese lapso 10.990 vehículos fueron sancionados, lo que nos demuestra que es necesario buscar una solución no solo para el tráfico sino también para disminuir el número de infractores por usar este carril.

1.3 Objetivos

1.3.1 Objetivos generales

- Desarrollar un prototipo para el uso tarifado y compartido del carril de la metrovía, que permita disminuir el tráfico y el índice de multas por invasión al mismo en la ciudad de Guayaquil.

1.3.2 Objetivos específicos

- Desarrollar el prototipo utilizando el lenguaje de programación Python y la interfaz Rasperry Pi 3.
- Implementar un circuito con Sensores infrarrojos para la detección del vehículo en el carril de la metrovía.
- Proporcionar una solución de fácil implementación y de bajo costo.

1.4 Metodología:

- Detección del vehículo en el carril de la metrovía.
- Captura de la imagen y extracción de la placa del vehículo
- Búsqueda de los datos asociados a la placa en la base de datos
- Validación de multas y notificaciones.

1.5 Resultados esperados

- Asegurar una disminución de tráfico notable en la ciudad de Guayaquil.
- Lograr una disminución del índice de multas por acceso al carril de la metrovía.
- Lograr un beneficio tanto de los conductores como de la Alcaldía de Guayaquil.

CAPÍTULO 2

2. FUNDAMENTO TEÓRICO

2.1. Metrovía

La metrovía es un sistema de Bus de Tránsito Rápido (BRT), implementado en la ciudad de Guayaquil en el año 2006, en la administración del alcalde Jaime Nebot [2].

La implementación del sistema se realizó en base a otros países que ya contaban con este servicio como Bogotá, con el propósito de proporcionar a los ciudadanos de Guayaquil un transporte que por un solo pasaje les permita llegar a distintos puntos y de una manera rápida, sin tráfico, accediendo al servicio mediante tarjetas recargables.

La metrovía tiene 3 rutas principales para sus buses alimentadores interurbanos, las mismas que son:

- Troncal 1 que va del Guasmo al Río Daule
- Troncal 2 que parte de la Av. 25 de Julio y llega hasta el Río Daule
- Troncal 3 Que sale de Bastión Popular hasta el Centro.

Así mismo en varias de sus paradas salen alimentadores para cubrir otras zonas de Guayaquil como lo son: Sauces, Garzota, Alborada, entre otras.

2.1.1. Multas por invasión al carril de la Metrovía

La metrovía es de gran beneficio para las personas que no cuentan con carro propio y buscan llegar rápido de un punto a otro, sin embargo, los conductores tomaron este carril exclusivo, como una vía para evitar tráficos, lo que trajo consigo que la Autoridad de Tránsito Municipal impongan multas excesivas por invasión al carril exclusivo descritas en la “Tercera Reforma a la ordenanza reformatoria y codificación de la ordenanza que crea y reglamenta el sistema integrado de transporte masivo urbano de la ciudad de Guayaquil” en su artículo 5.

Este artículo nos dice que cualquier conductor en general que invada el carril de la metrovía será sancionado con una multa equivalente a un salario básico y en caso de reincidencia el doble [12].

En la figura 2.1 podemos observar la multa por invasión del carril en la Av. de las Américas

Figura 2.1 Carril de la Metrovía Av. De las Américas [13].

2.2. Raspberry pi

2.2.1. Definición

Placa de desarrollo que apareció en el año 2012, pero se empezó a desarrollar desde el año 2006 por una fundación sin fines de lucros llamada “Raspberry Pi Fundation”.

Este ordenador se desarrolló con el objetivo de proporcionar una enseñanza más fácil a bajo costo de la informática en la educación de nivel medio [4]**.**

2.2.2. Hardware

Esta placa tiene unas dimensiones de 8.5x5.3 cm, y cuenta con un chip integrado Broadcom BCM2835, con un procesador ARMv8, WIFI 802.11 b/g/n, bluetooth 4.1, 4 puertos HDMI, slot para MicroSD

Ethernet, conexión HDMI para poder conectarlo a un monitor, entre otras como se observa en la figura 2.2.

Sus principales características es su alta disponibilidad, baja potencia, alta fiabilidad y su bajo costo, además de eso cuenta con 40 pines los mismos que nos permitir adicionar dispositivos como cámaras, teclado, mouse, leds, auriculares, etc.

Entre sus pines más usados tenemos los GPIO, SPI, UART, HDMI, NAND [5].

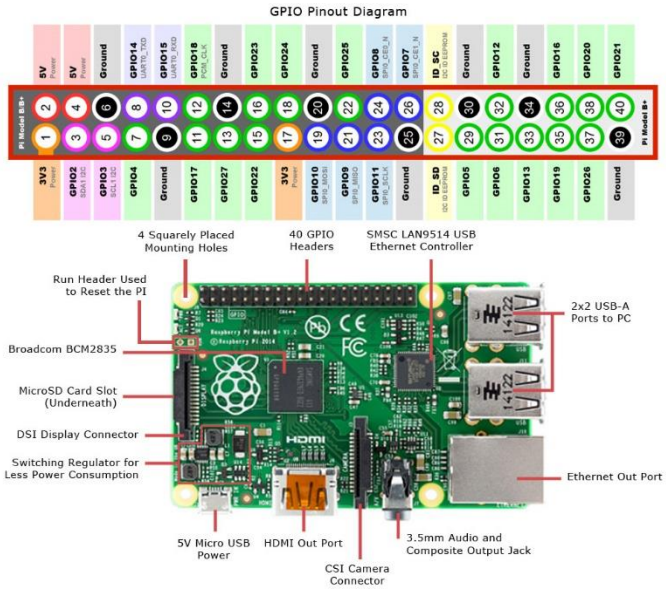

Figura 2.2: Puertos y pines Raspberry Pi 3 [10].

2.3. Python

Lenguaje de programación de fácil aprendizaje creado al principio de los años 90, como sucesor del lenguaje de programación ABC.

Entre sus características tenemos que utiliza un programa interprete ya que este no compila el código a lenguaje de máquina, es de tipado dinámico, es decir no es necesario declarar las variables que se van a utilizar, ya que estas se adaptan automáticamente en la ejecución del programa, es fuertemente

tipado, por lo que no podemos combinar variables de distintos tipos, es multiplataforma, es decir, podemos usar Linux, Windows, Mac.

Cuenta con librerías que nos permiten añadir funciones especiales al lenguaje como OpenCv que comprende el tratado de imágenes [6].

2.4. OpenCV

OpenCv (Open Source Computer Vision Library), es una librería de uso gratuito en el área académica y comercial, ya que se publica bajo la licencia BSD, cuenta con interfaz en varios lenguajes de programación como C, C++, Python y Java, puede ser usado en los distintos sistemas operativos que existen, es decir MAC, Windows, Linux, Android.

Fue diseñada principalmente para el enfoque en aplicaciones de tiempo real, cuenta con más de 500 funciones en el área del procesamiento de visión, que nos facilitan la calibración de cámaras, reconocimiento de objetos, visión robótica y visión estéreo [7].

2.5. MySQL

Base de datos de código abierto que proporciona confiabilidad, alto desempeño y facilidad de uso, permite realizar hojas de datos, informes, demostraciones, boletines.

Considerada la base de datos Open Source más usada en el mundo, ya que cuenta con bases de datos de hasta 50 millones de registros y puede ser usada por distintos lenguajes de programación como Python, Ruby, REALbasic, Tcl. [11].

2.6. Twilio

Servicio que permite usar las aplicaciones telefónicas a través de páginas web, aplicaciones, lenguajes de programación, etc.

Trabaja mediante la asignación de un número telefónico perteneciente al servicio, con el cual se puede realizar llamadas, enviar mensajes de texto, registrar mensajes de voz, pero su servicio es limitado, ya que solo se puede usar con cuentas pertenecientes a Twilio [14].

CAPÍTULO 3

3. IMPLEMENTACIÓN

3.1. Diseño

El proyecto consiste en un circuito electrónico con diodos infrarrojos para la detección del vehículo, un módulo Raspberry NoIR Camera para la captura de la imagen y una Raspberry Pi 3 model (b) para la programación, los mismo que se conectan como se observa en la figura 3.1.

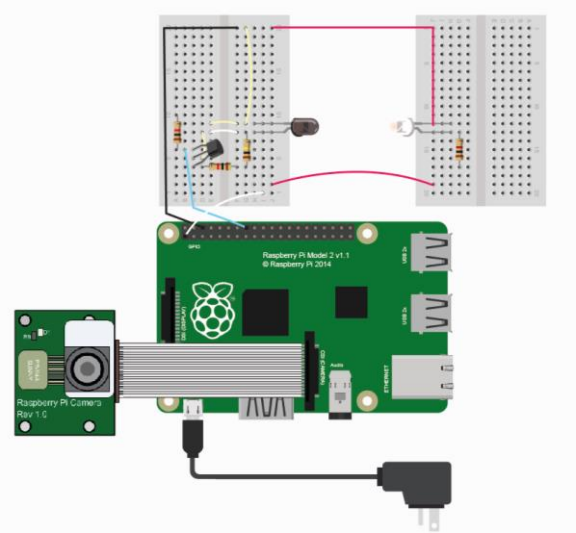

Figura 3.1: Conexiones de los elementos del proyecto

3.2. Programación

Este proyecto inicia importando todas las librerías y archivos necesarios para la manipulación de la cámara de la Raspberry Pi y el procesamiento de imágenes para detectar la placa y reconocer los caracteres en dicha sección, así mismo se importa las librerías necesarias para manipular bases de datos de MySQL y enviar mensaje de texto a través de la plataforma Twilio. Cabe recalcar que los archivos donde se realiza todo el proceso de la imagen son:

DetectChars, DetectPlates, PossiblePlate, en ellos hay otros archivos que son necesarios como PossibleChar, PossiblePlate, Preprocess.

Luego se crean arreglos de tres dimensiones que representan los colores con los que se forma los rectángulos, que indican donde se encuentra la placa y el color de fuente de las letras que se usa, para sobrescribir en las imágenes generadas.

El prototipo permanece censando una señal correspondiente a un circuito electrónico con diodos IR (emisor y receptor), que actúan como barrera infrarroja para detectar cuando un vehículo se acerca y atraviesa el carril, para iniciar la captura de la imagen desde una zona específica con el fin captar al vehículo.

Para la señal emitida por el infrarrojo es necesario asignar un pin de la Raspberry Pi 3, para el prototipo usamos el GPIO 23, el mismo que se debe setear antes de recibir la señal correspondiente, además se crean variables para el conteo de multas y crear anti rebotes en el circuito electrónico.

Luego se ajusta la el módulo de la cámara de la Raspberry, con un objeto que la represente, en este caso el objeto en el código lleva el nombre de “camera” y se la trabaja como arreglo para captar las imágenes con las funciones de seek(0) y truncate(), de este arreglo se puede tener una sucesión de imágenes en tiempo real como si se tratara de un video, hacemos esto para luego poder usar la función capture() que nos permitirá tomar la foto en el momento justo cuando nos indique la barrera.

Para obtener una buena imagen se debe ajustar los parámetros de la cámara como la resolución (320, 240), el ángulo de visión de la imagen (en este caso 180º) y el framerate al máximo para que pueda capturar lo más rápido la imagen (en este caso lo máximo es 15 fps).

Se entra en un lazo “while” de forma infinita en donde se usa la cámara para que esté tomando las imágenes a cada momento y la esté mostrando en la ventana llamada “frame”.

Se hace un pequeño código de anti rebote para cuando el vehículo pase por la barrera se tome la foto y se haga el proceso una sola vez, hasta que el vehículo haya terminado completamente de pasar la barrera.

Cuando el vehículo pase la barrera se hace la captura de la imagen como podemos observar en la figura 3.2, que luego se guardará de manera provisional como ‘imagen.jpg’, para luego ser llamada en el proceso de reconocimiento de la placa.

Figura 3.2: Captura del vehículo luego de pasar la barrera infrarroja.

El código primero obtiene de archivos que han sido cargados por un entrenamiento mediante redes neuronales, la información necesaria para el reconocimiento de los caracteres en la placa.

Para esto usa la función loadKNNDataAndTrainKNN(), que se encuentra en el archivo DetectChars.py, la cual devuelve un valor booleano, si este es falso es porque hubo alguna complicación en el proceso de carga de los archivos y si es verdadero el proceso continúa llamando a la imagen que se ha capturado anteriormente al atravesar la barrera infrarroja, esta imagen se encuentra localizada dentro de los archivos que contienen el código, debido a que es sólo una imagen que permite identificar la placa del vehículo, y de la cual se procederá a crear copias en caso de que se haya generado una multa para guardarla en la carpeta correspondiente, si llegase el caso de haber otro vehículo censado se sobrescribe la imagen con la foto capturada al momento.

Para este proceso de cargar la imagen se utilizó la función cv2.imread() de la librería OpenCV.

Al poder ser cargada la imagen para el reconocimiento de la placa se llama a la función detectPlatesInScene() que se encuentra en el fichero DetectPlates.py, la cual devuelve una lista de las posibles regiones donde se encuentran caracteres alfanuméricos en un área rectangular, dentro de este fichero se usa la función preprocess() del archivo Preprocess.py en la cual se realiza una conversión de color a HSV de la imagen mediante la función extractValue() del mismo fichero que usa algunas herramientas de la librería OpenCV tal como cv2.cvtColor() para realizar dicha conversión y se unen los canales de la imagen mediante la función cv2.split(), así mismo mediante el uso de la función maximizeContrast() se hace la eliminación de ruidos de la imagen en escala de grises anteriormente convertida mediante una erosión y dilatación de la misma con la ayuda de cv2.morphologyEx() con parámetro “cv2.MORPH_TOPHAT y cv2.MORPH_BLACKHAT” y operaciones de añadidura y de sustracción con cv2.add() y cv2.subtract(), finalmente con cv2.GaussianBlur() se elimina el ruido gaussiano después del primer filtro.

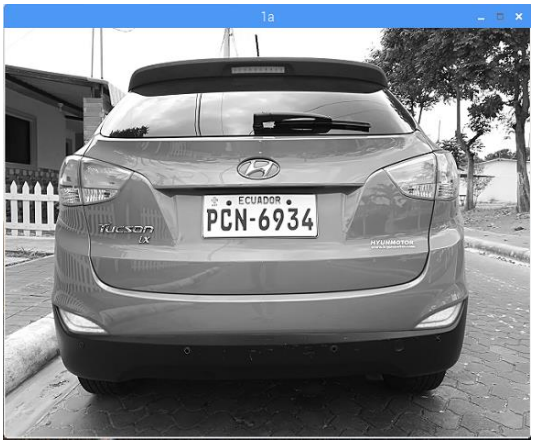

Figura 3.3: Imagen del vehículo en escala de grises.

Con estos pasos logramos cambiar nuestra imagen a escala de grises como se muestra en la figura 3.3, para luego con la función cv2.adaptiveThreshold() poder comparar cada pixel del umbral y obtener su forma binarizada.

Figura 3.4: Imagen del vehículo binarizada.

Con la imagen binarizada como se muestra en la figura 3,4, el algoritmo inicia la búsqueda de la placa en la imagen, variando la escala de grises de la imagen y marcando rectángulos en ella hasta encontrar el más grande como lo muestra la figura 3.5.

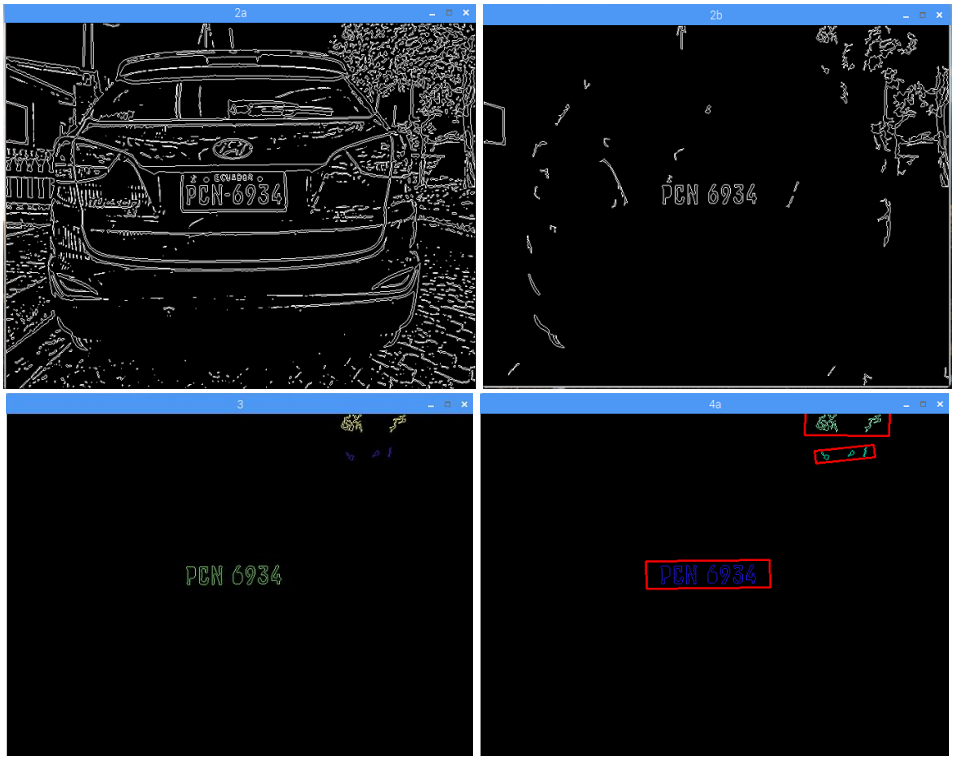

Figura 3.5: Proceso de detección de los caracteres de la placa.

Una vez detectada la placa, ésta se extrae y se la procesa hasta tenerla en escala de grises como se observa en la figura 3.6.

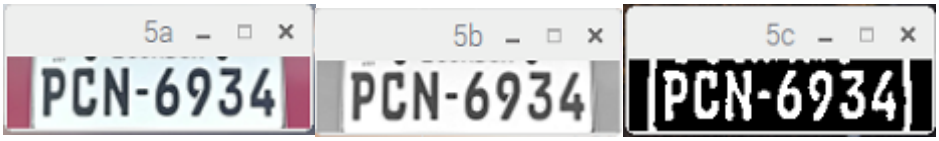

Figura 3.6: Placa extraída y conversión a escala de grises.

Luego se inicia el reconocimiento de caracteres con la función findPossibleCharsInScene() como se observa en la figura 3.7, antes de esto es necesario crear una copia debido a que esta imagen binarizada en el proceso será alterada con la función cv2.findContours(), la cual entrega una lista según lo especificado en los parámetros de la función con todos los contornos encontrados, para luego analizar si son o no caracteres mediante la función checkIfPossibleChar() del fichero DetectChars.py, la cual analiza si es un carácter válido en el aprendizaje de redes neuronales provisto en los archivos cargados al comienzo del proceso y si se encuentra en el área delimitada de la placa, devolviendo una lista de caracteres que hayan sido analizados y reconocidos.

Figura 3.7: Detección de caracteres.

Si la lista de posibles placas no es nula, se procede a analizar los caracteres que posee, si estos no son nulos al igual, se dibuja en la región de la placa un rectángulo alrededor de la zona que contiene dichos caracteres mediante la función drawRedRectangleAroundPlate(), que se encuentra en el mismo fichero del código principal, aparte se presenta en texto dicha placa tanto en consola como en la imagen usando la función writeLicensePlateCharsOnImage(), junto con la fecha actual del momento del uso del carril como se observa en la figura 3.8.

Esta imagen se muestra en una ventana creada llamada "imgOriginalScene" además de guardarla en una carpeta en específico si es que en el proceso de análisis de vigencia del permiso en la base de datos muestra que se encuentra en estado de caducidad.

Figura 3.8: Imagen con la placa y fecha de invasión sobrescrita.

El proceso de verificación en la base de datos de la placa obtenida es el siguiente:

Se crea el conector de MySql con el nombre del host (dirección de la base de datos donde guardamos la información), el usuario y la contraseña para poder manejarla también a través de phpMyAdmin como lo muestra la figura 3.9, se especifica además el nombre de la base de datos donde se encuentre la tabla con la información de los conductores que han accedido al permiso del carril de la Metrovía.

Figura 3.9: Inicio de plataforma MySQL con phpMyAdmin.

Para poder mostrar por consola los datos de la persona que ha pasado por el carril, cuya placa ha sido captada y hallada en la base de datos se hace uso de la creación de un cursor, donde se especifica el parámetro que se quiere leer de la base mediante los comandos SELECT __ FROM __ WHERE, y se ejecuta la acción por medio de la instrucción cursor.execute() y cursor.fetchone() para seleccionar el dato y mostrarlo como string por consola como en la figura 3.10, y ver todos los parámetros como NOMBRES, APELLIDOS, MODELO, INICIO DEL PERMISO, FIN DEL PERMISO Y USOS DEL CARRIL.

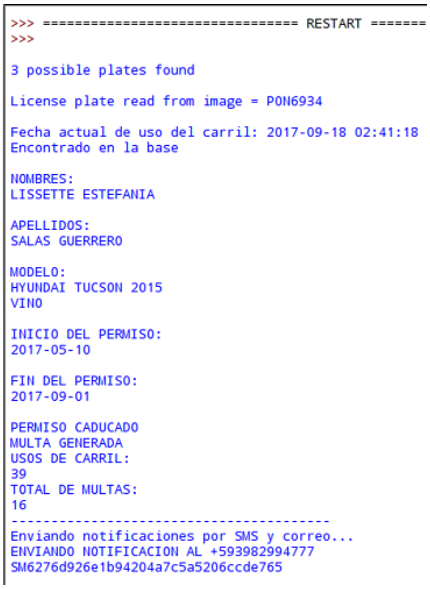

Figura 3.10: Lectura de la base de datos de una placa.

Con las fechas de inicio y fin del permiso se verifica si está vigente o caducado, comparándolas con la fecha actual del sistema.

En el caso de modificar o actualizar un dato de la base como el número de multas o la cantidad de veces usadas del carril, se usa los comandos UPDATE __ SET __ WHERE con la instrucción de cursor.execute() y db.commit() para que se actualice dentro de la base de datos utilizada.

En la figura 3.11 se observa la base de datos, donde se registran los vehículos que acceden al uso del carril a través de una tarifa, registrando los datos del

propietario del vehículo como nombres, apellidos, correo, teléfono y las fechas de inicio y fin del permiso adquirido.

PLACA	MODELO	COLOR	NOMBRES	APELLIDOS	INICIO_PERMISO	FIN_PERMISO	USOS	MULTAS	NUID	CEL
P0N6934	HYUNDAI TUCSON 2015	VINO	LISSETTE ESTEFANIA	SALAS GUERRERO	2017-05-10	2017-09-01	39	16	214748	+59
RIPLS1	FORD 2015	GRIS	ARNALDO ANDRE	ABAD GOMEZ	2017-03-15	2017-05-15	67	65	123456	+59

Figura 3.11: Base de datos de los vehículos registrados.

El proceso de las notificaciones por correo electrónico o por SMS es el siguiente:

Se crea un objeto MIMEMultipart del cual partiremos para poder usar los argumentos de ‘From’, ‘To’ y ‘Subject’ para poder enviar un correo electrónico, se debe tener en cuenta que cuando se usa el argumento de nuestro objeto multipart con el parámetro ‘From’ se debe usar una cuenta de correo propia de Google (Gmail).

Para poder enviar el mensaje que contiene la notificación con los detalles del acto se crea un objeto MIMEText, y para cargar la imagen capturada con las modificaciones realizadas en el proceso de análisis de la placa se crea un objeto MIMEImage, indicando la ruta de donde se está cargando la imagen, añadiendo en la cabecera otro nombre en este caso “placa.png”.

Se continua con un proceso de autenticación creando un objeto de sesión de cliente SMTP para poder enviar el correo con la cuenta de Gmail que se haya puesto como administrador, usando el usuario y la contraseña del mismo, se envía el correo haciendo uso del comando mailServer.sendmail() cuyos parámetros serán los correos del remitente (administrador) y el destinatario (obtenido de la base de datos) y se cierra el objeto creado anteriormente.

Este correo contara con la imagen correspondiente a la multa con la fecha de invasión sobrescrita, y un mensaje con información relevante como se observa en la figura 3.12.

Figura 3.12: Correo con aviso de multa por invasión de carril de la Metrovía.

Para la notificación mediante SMS se procede a obtener ciertos parámetros registrados en la base de datos que son información otorgada por la cuenta de Twilio como el ID de la cuenta (account_sid) y su autorización (auth_token) y se crea un objeto con la función Client(), con los parámetros antes mencionados se crea un mensaje donde se especifica a quién se va a enviar y de qué número otorgado por Twilio se va a realizar el envío.

En el mensaje, se indica al conductor que ha usado el carril con el permiso en estado caducado como se observa en la figura 3.13.

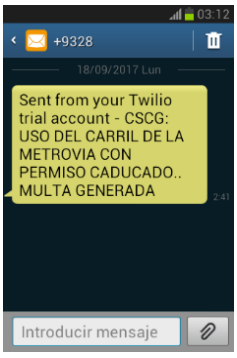

Figura 3.13: SMS con aviso de multa por invasión de carril de la Metrovía.

Al finalizar cuando se ha enviado las notificaciones se añade en otra tabla en la misma base de datos, el número de multas generadas, la fecha actual de la infracción, el nombre de la imagen captada, el nombre y apellido del usuario del servicio y el número de placa capturada como tenemos en la figura 3.14.

NUMERO_MULTA	NOMBRE_IMAGEN	FECHA_MULTA	NOMBRES	PLACA ▲ 1
1	imgen	12/08/2017	arnaldo abad	ABC1234
1	imgen	12/08/2017	arnaldo abad	ABC1234
1	2017-08-07_09:04:53	2017-08-07 09:04:53	ARNALDO ANDRE ABAD GOMEZ	M0B250
1	2017-08-07_12:57:18	2017-08-07 12:57:18	ARNALDO ANDRE ABAD GOMEZ	M0B250
1	2017-08-29_19:51:55	2017-08-29 19:51:55	LISSETTE ESTEFANIA SALAS GUERRERO	PCN6934
1	2017-08-07_13:17:59	2017-08-07 13:17:59	LISSETTE ESTEFANIA SALAS GUERRERO	RPLS1

Figura 3.14: Base de datos con las multas generadas.

Finalmente se puede observar en la figura 3.15 y 3.16 el prototipo que se creó para el Sistema de Metrovía y la captura de la placa del vehículo.

Figura 3.15: Prototipo del Sistema de la Metrovía.

Figura 3.16: Captura de la placa del vehículo con el prototipo.

CAPÍTULO 4

4. RESULTADOS OBTENIDOS

Se realizan pruebas con imágenes de placas de vehículos capturadas con otra cámara distinta a la usada en el prototipo para comprobar la eficiencia del código y se obtienen los siguientes resultados:

1. En ciertas imágenes al ser vueltas a leer en el proceso luego de que se han cargado otras anteriormente, el programa reconoce caracteres extras que no existen, es decir presenta un falso positivo como se observa en las figuras 4.1 y 4.2.

Figura 4.1: Matrícula capturada (seis caracteres reconocidos)

Figura 4.2: Matrícula capturada (falso positivo presentado)

2. Letras que tiene forma cerrada como la letra G y C dependiendo del tipo de fuente usada en la placa, tienen problemas al ser reconocidas como se ve en la figura 4.3.

Figura 4.3: Matrículas con caracteres con forma cerrada.

3. Dependiendo del ángulo de captura de la imagen, se ve una alteración en el reconocimiento, causando que se limite la zona donde se reconoce el área de la placa o se altere caracteres en dichos límites como se observa en la figura 4.4.

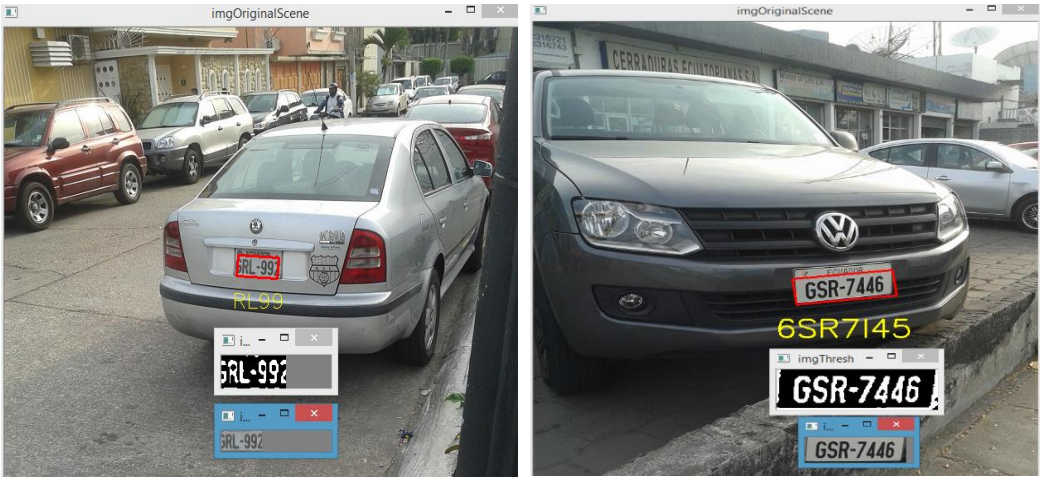

Figura 4.4: Matrículas capturadas a cierto ángulo de inclinación lateral por la cámara.

4. Existen casos exitosos en donde se corrobora las acotaciones anteriores, es decir cuando la imagen se captura con perfil frontal o posterior del vehículo sin inclinaciones, y con letras en una fuente estándar como en la figura 4.5.

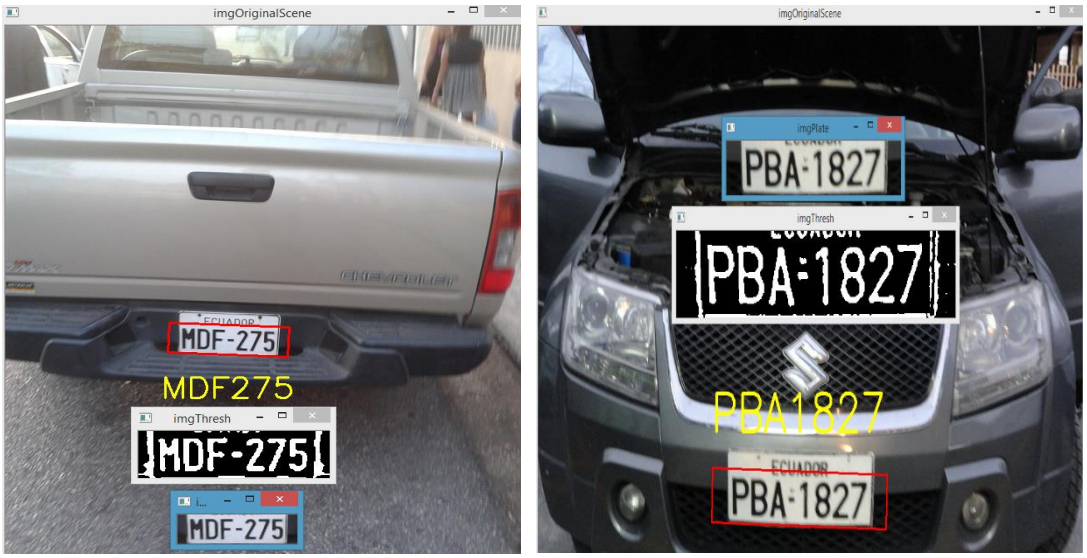

Figura 4.5: Matrículas con reconocimiento de caracteres exitoso.

La tabla 1 muestra el porcentaje de exactitud del algoritmo al reconocer los caracteres en las imágenes mostradas.

Nº Figura	Matrícula real del vehículo	Matrícula reconocida	Caracteres erróneos/total	Porcentaje de exactitud (%)
Figura 4.1	GSM-9640	OSM-9640	1/7	85.71
Figura 4.3	GSN-5505	OSN-5505	1/7	85.71
	GOY-515	COYSI5	3/6	50.00
	PCN-6934	PON-6934	1/7	85.71
	GSP-7329	8SP-73Z9	2/7	71.43
Figura 4.4	GRL-992	RL-99	2/6	66.67
	GSR-7446	6SR-7145	3/7	57.14
Figura 4.5	MDF-275	MDF-275	0/6	100.00
	PBA-1827	PBA-1827	0/7	100.00

Tabla 1: Exactitud del algoritmo.

CONCLUSIONES Y RECOMENDACIONES

Se construyó un prototipo utilizando el lenguaje de programación Python y la interfaz Raspberry Pi 3, con algoritmos de redes neuronales que nos proporcionan un porcentaje de exactitud de alrededor del 78.04%, lo cual no es suficiente para esta aplicación, por lo que se recomienda probar con nuevos algoritmos que no solo funcionen con imágenes en perfil frontal y posterior, sino también que corrija inclinaciones y haga una mejor limpieza de la imagen antes de la extracción de caracteres.

Los diodos infrarrojos proporcionan una buena detección del vehículo al momento que este interrumpe su comunicación, sin embargo, se puede reducir el costo de implementación haciendo la detección mediante códigos en lenguaje de programación Python, reduciendo también el tiempo para la captura de la imagen del vehículo.

El módulo de cámara NoIR de la raspberry es una buena herramienta para la captura de imágenes, ya que cuenta con una resolución de 5 Megapíxeles y puede tomar imágenes estáticas de hasta 2592x1944 pixeles, sin embargo, no es la mejor opción para captar imágenes en movimiento, y este sistema necesita una captura instantánea y de buena resolución, para poder captar rápidamente la imagen sin límites de velocidad y con buen alcance para que en las zonas donde exista doble carril no sea necesario usar 2 cámaras.

Twilio es una herramienta simple de usar y cuenta con las funciones necesarias para el envío de notificaciones a través de mensajes de texto, sin embargo, su alcance es limitado ya que aún no llega a ciertos países y solo funciona con números pertenecientes a su red, por lo que se puede cambiar esta herramienta por módulos GSM con el fin de abarcar a todos los usuarios del sistema.

Este proyecto funciona a través de internet por lo que es necesario contar con una línea directa conectada al sistema y no a través de redes wireless, ya que con estas su procesamiento no es tan rápido y eso demora las notificaciones a los usuarios.

BIBLIOGRAFÍA

[1] Telégrafo, E. (2017). Las multas por invadir el carril de la Metrovía crecieron en 400%. [online] El Telégrafo. Disponible en: http://www.eltelegrafo.com.ec/noticias/guayaquil/10/las-multas-por-invadir-el-carril-de-la-metrovia-crecieron-en-400 [Acceso 10 Jul. 2017].

[2] Metrovia-gye.com.ec. (2017). Fundación Metrovia | Sistema Integrado de Transporte Masivo Urbano de la ciudad de Guayaquil. [online] Disponible en: http://www.metrovia-gye.com.ec/ [Acceso 10 Jul. 2017].

[3] Metrovia-gye.com.ec. (2017). Fundación Metrovia | Sistema Integrado de Transporte Masivo Urbano de la ciudad de Guayaquil. [online] Disponible en: http://www.metrovia-gye.com.ec/mapaderutas. [Acceso 10 Jul. 2017].

[4] Raspberry Pi. (2017). Raspberry Pi Foundation - About Us. [online] Disponible en: https://www.raspberrypi.org/about/ [Acceso 12 Ago. 2017].

[5] Anon, (2017). [online] Disponible en: https://www.raspberrypi.org/documentation/hardware/computemodule/RPI-CM-DATASHEET-V1_0.pdf. [Acceso 12 Ago. 2017].

[6] Raspberry Pi. (2017). Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. [online] Disponible en: https://www.raspberrypi.org/[Acceso 20 Ago. 2017].

[7] Opencv.org. (2017). OpenCV library. [online] Disponible en: http://opencv.org/[Acceso 20 Ago. 2017].

[8] contributors, p. (2017). phpMyAdmin. [online] phpMyAdmin. Disponible en: https://www.phpmyadmin.net/ [Acceso 12 Sep. 2017].

[9] Twilio.com. (2017). Communication APIs for SMS, Voice, Video and Authentication. [online] Disponible en: https://www.twilio.com/ [Acceso 12 Sep. 2017].

[10] Blog electronica y ciencia. (2017). Conexión GPIO de Raspberry Pi 3. [online] Disponible en: http://electronicayciencia.blogspot.com/2016/11/conexion-gpio-de-raspberry-pi-3.html [Acceso 12 Sep. 2017].

[11] Oracle.com. (2017). MySQL | La base de datos de código abierto más popular | Oracle América Latina. [online] Disponible en: https://www.oracle.com/lad/mysql/index.html [Acceso 15 Sep. 2017].

[12] Anon, (2017). [online] Disponible en: http://guayaquil.gob.ec/Gacetas/Periodo%202014-2019/Gaceta%2017.pdf [Acceso 15 Sep. 2017].

[13] El Universo. (2017). Desde hoy, multa por invadir carril de Metrovía en avenida de las Américas. [online] Disponible en: http://www.eluniverso.com/noticias/2017/06/12/nota/6227956/hoy-multa-invadir-carril [Acceso 26 Sep. 2017].

[14] Twilio.com. (2017). Communication APIs for SMS, Voice, Video and Authentication. [online] Disponible en: https://www.twilio.com/ [Acceso 26 Sep. 2017].