



# ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA  
CARRERA DE INGENIERÍA EN ELECTRÓNICA E  
INSTRUMENTACIÓN**

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL  
TÍTULO DE INGENIERO ELECTRÓNICO EN  
INSTRUMENTACIÓN**

**TEMA: “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA  
DETECCIÓN DE VEHÍCULOS ROBADOS EN MOVIMIENTO,  
EMPLEANDO TECNOLOGÍA BEAGLEBONE, POR MEDIO DE  
SOFTWARE LIBRE”**

**AUTORES: GARZÓN CANCHIGNIA ROBERTO CARLOS  
PACHECO GAVILÁNEZ JONATHAN ALEXANDER**

**DIRECTOR: ING. EDDIE GALARZA ZAMBRANO**

**LATACUNGA**

**2016**



**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA ELECTRÓNICA  
CARRERA DE ELECTRÓNICA E INSTRUMENTACIÓN**

**CERTIFICACIÓN**

Certifico que el trabajo de titulación, **“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA DETECCIÓN DE VEHÍCULOS ROBADOS EN MOVIMIENTO, EMPLEANDO TECNOLOGÍA BEAGLEBONE, POR MEDIO DE SOFTWARE LIBRE”** realizado por los señores **ROBERTO CARLOS GARZÓN CANCHIGNIA, JONATHAN ALEXANDER PACHECO GAVILÁNEZ**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditar y autorizar a los señores **ROBERTO CARLOS GARZÓN CANCHIGNIA, JONATHAN ALEXANDER PACHECO GAVILÁNEZ** para que lo sustenten públicamente.

**Latacunga, 18 de agosto del 2016**



Ing. Eddie Galarza Zambrano  
**DIRECTOR**



**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA ELECTRÓNICA  
CARRERA DE ELECTRÓNICA E INSTRUMENTACIÓN**

**AUTORÍA DE RESPONSABILIDAD**

Nosotros, **ROBERTO CARLOS GARZÓN CANCHIGNIA, JONATHAN ALEXANDER PACHECO GAVILÁNEZ**, con cédula de identidad N°0503314692, 0503374787 respectivamente, declaramos que este trabajo de titulación “**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA DETECCIÓN DE VEHÍCULOS ROBADOS EN MOVIMIENTO, EMPLEANDO TECNOLOGÍA BEAGLEBONE, POR MEDIO DE SOFTWARE LIBRE**”, ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance de la investigación mencionada.

**Latacunga, 18 de agosto del 2016**

Roberto Carlos Garzón Canchignia

**C.C. 0503314692**

Jonathan Alexander Pacheco Gavilánez

**C.C. 0503374787**



**DEPARTAMENTO DE ELÉCTRICA ELECTRÓNICA  
CARRERA DE ELECTRÓNICA E INSTRUMENTACIÓN**

**AUTORIZACIÓN**

Nosotros, **ROBERTO CARLOS GARZÓN CANCHIGNIA, JONATHAN ALEXANDER PACHECO GAVILÁNEZ**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA DETECCIÓN DE VEHÍCULOS ROBADOS EN MOVIMIENTO, EMPLEANDO TECNOLOGÍA BEAGLEBONE, POR MEDIO DE SOFTWARE LIBRE”** cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría respetando autoría de terceros.

**Latacunga, 18 de agosto del 2016**

Roberto Carlos Garzón Canchignia

**C.C. 0503314692**

Jonathan Alexander Pacheco Gavilánez

**C.C. 0503374787**

## **DEDICATORÍA**

Dedico este trabajo a mi familia, quienes me brindaron su apoyo diario a lo largo de estos años llenos de constancia y sacrificio.

También dedico este proyecto al Ingeniero Eddie Galarza que supo guiarnos con sus conocimientos y consejos para el desarrollo de la tesis, y por último dedico este trabajo a todos mis amigos los cuales me brindaron su apoyo incondicional durante estos 6 años de vida universitaria.

**Jonathan**

## DEDICATORÍA

A Dios por haberme dado la sabiduría y la paciencia para sobrellevar los problemas y adversidades de la vida, además por poner en mi camino a personas que con su apoyo, aprecio y bondad me brindaron fuerza para seguir adelante sin desmayo.

A mis padres quienes con su apoyo incondicional, sus consejos y más que nada su cariño y comprensión me impulsaron a seguir adelante y me demostraron que el éxito se consigue solo a través del sacrificio constante.

**Roberto**

## **AGRADECIMIENTO**

Agradezco de una manera muy especial a toda mi familia, a mi abuelita Leticia Mazón por su ejemplo de cariño y constancia, a mi madre Blanca Gavilánez por su apoyo incondicional que siempre estuvo a mi lado en las buenas y malas, dándome la motivación diaria con su ejemplo, para poder alcanzar los objetivos trazados en mi vida.

Agradezco de manera especial a todos mis amigos que he conseguido en todos estos años de estudio con los cuales se ha pasado momentos buenos y malos, pero siempre hemos podido salir adelante en especial a Roberto Garzón, Kevin Bedón.

Gracias también a los maestros que supieron brindarme su invaluable conocimiento, consejos y experiencias de manera especial al Ingeniero Eddie Galarza por guiarnos en el desarrollo del proyecto.

**Jonathan**

## **AGRADECIMIENTO**

A mi familia que ha estado presente a lo largo de mi vida, demostrando que el cariño y esfuerzo permite alcanzar nuestros más grandes sueños

A mi padre Carlos Garzón y mi madre Beatriz Canchignia quienes con su ejemplo me inculcaron los valores de perseverancia, respeto y honestidad, mostrándome que no existen caminos fáciles sin esfuerzo.

A mis amigos que me brindaron su apoyo incondicional siempre que lo necesite, además agradecer su ayuda, su comprensión y consejos en momentos difíciles, sinceramente gracias amigos Jonathan, Kevin y Sebastián, muchas gracias por su apoyo.

También agradezco sinceramente al Ingeniero Eddie Galarza quien con sus conocimientos, responsabilidad y amistad, supo guiarnos a lo largo de este proyecto, compartiendo su enseñanzas y motivándonos a concluir con éxito nuestras metas.

**Roberto**



## ÍNDICE DE CONTENIDO

### CARÁTULA

<b>CERTIFICADO</b> .....	II
<b>AUTORÍA DE RESPONSABILIDAD</b> .....	III
<b>AUTORIZACIÓN</b> .....	IV
<b>DEDICATORÍA</b> .....	V
<b>AGRADECIMIENTO</b> .....	VII
<b>ÍNDICE DE TABLAS</b> .....	XIII
<b>ÍNDICE DE FIGURAS</b> .....	XIV
<b>RESUMEN</b> .....	XVII
<b>ABSTRACT</b> .....	XVIII

### CAPÍTULO I .....

<b>1. INTRODUCCIÓN</b> .....	- 1 -
1.1. Prólogo .....	- 1 -
1.2. Antecedentes .....	- 1 -
1.3. Planteamiento del problema .....	- 6 -
1.4. Justificación .....	- 6 -
1.5. Objetivos del proyecto .....	- 7 -
1.5.1. General .....	- 7 -
1.5.2. Específicos .....	- 7 -
1.6. Software libre .....	- 8 -
1.6.1. Tipos de licencias de software libre .....	- 8 -
1.6.2. Ventajas y desventase del software libre .....	- 9 -
1.7. Visión artificial .....	- 10 -
1.7.1. Ventajas y Beneficios de la visión artificial .....	- 11 -
1.7.2. Componentes de un sistema de visión artificial .....	- 12 -
1.7.3. Software para tratamiento de imágenes por visión Artificial .....	- 12 -
1.8. Opencv .....	- 13 -
1.8.1. Origen y Uso .....	- 14 -
1.8.2. Características de opencv .....	- 14 -
1.8.3. Aplicaciones de OpenCV. ....	- 15 -

1.8.4.	Estructura y contenido de OpenCV .....	- 16 -
1.9.	Selección de región de interés.....	- 17 -
1.10.	Cámara logitech c920.....	- 18 -
1.11.	Procesamiento digital de imágenes .....	- 19 -
1.11.1.	Adquisición de la Imagen .....	- 19 -
1.11.2.	Procesamiento.....	- 20 -
1.11.3.	Segmentación.....	- 20 -
1.11.4.	Binarización.....	- 21 -
1.11.5.	Reconocimiento Óptico de Caracteres .....	- 21 -
1.11.6.	Tesseract.....	- 22 -
1.12.	Beaglebone black rev c.....	- 23 -
1.12.1.	Características Generales .....	- 24 -
1.12.2.	Softwares Compatibles.....	- 25 -
1.13.	Python .....	- 25 -
1.13.1.	Antecedentes .....	- 26 -
1.13.2.	Características.....	- 26 -
1.13.3.	Principales Librerías en Python .....	- 27 -
1.14.	Base de datos mysql .....	- 28 -
1.14.1.	Características de MYSL .....	- 28 -
1.15.	Placas Vehiculares.....	- 29 -
<b>CAPÍTULO II .....</b>		<b>- 32 -</b>
<b>2. DESARROLLO .....</b>		<b>- 32 -</b>
2.1.	Instalación del sistema operativo en beaglebone black rev c.....	- 32 -
2.1.1.	Expansión de la Memoria en una MicroSD.....	- 33 -
2.1.2.	Instalación de OpenCV 2.4.9.....	- 34 -
2.2.	Lectura de imágenes con opencv y python.....	- 36 -
2.2.1.	Librería para la Adquisición de Imágenes .....	- 37 -
2.2.2.	Funciones para Lecturas de Imágenes .....	- 37 -
2.2.3.	Presentación de Imágenes .....	- 39 -
2.2.4.	Ejemplos .....	- 40 -
2.3.	Adquisición de imágenes en tiempo real con opencv y python .....	- 42 -
2.3.1.	Inicialización de la Cámara.....	- 42 -
2.3.2.	Lectura de la Cámara .....	- 43 -

2.4.	Captura de imágenes con opencv y python.....	- 43 -
2.5.	Desarrollo del algoritmo para el reconocimiento de placas vehiculares	- 45 -
2.5.1.	Ubicación de la Placa en una Región de Interés .....	- 47 -
2.5.2.	Extracción y Corrección del Ángulo de la Placa.....	- 49 -
2.5.3.	Algoritmo de Reconocimiento Óptico de Caracteres Tesseract .....	- 50 -
2.5.4.	Manejo de OCR Tesseract a través de Python .....	- 52 -
2.6.	Métodos de validación de caracteres.....	- 53 -
2.6.1.	El Método isalnum() .....	- 53 -
2.6.2.	El Método isalpha() .....	- 54 -
2.6.3.	El Método isdigit().....	- 54 -
2.7.	Base de datos python-mysql .....	- 58 -
2.7.1.	Crear Base de Datos .....	- 58 -
2.7.2.	Ingreso de Información en Base Mysql.....	- 60 -
2.7.3.	Desplegar Base de Datos .....	- 60 -
2.8.	Manejo de datos en mysql.....	- 61 -
2.8.1.	Acceso a Base de Datos.....	- 61 -
2.9.	Interfaz de usuario python .....	- 65 -
2.9.1.	Crear Ventanas con Tkinter.....	- 68 -
2.9.2.	Manejo de Etiquetas .....	- 69 -
2.9.3.	Gestión de la geometría.....	- 69 -
2.9.4.	Desarrollo de la Interfaz .....	- 70 -
	<b>CAPÍTULO III.....</b>	<b>- 74 -</b>
	<b>3. PRUEBAS Y RESULTADOS .....</b>	<b>- 74 -</b>
3.1.	Descripción de pruebas realizadas .....	- 74 -
3.2.	Descripción de pruebas realizadas .....	- 75 -
3.2.1.	Pruebas Realizadas a Diferentes Vehículos con la Tarjeta Beaglebone Black .....	- 75 -
3.2.2.	Pruebas Realizadas en el Computador .....	- 88 -
3.2.4.	Pruebas Realizadas a Vehículos Específicos con una Tarjeta Beaglebone Black y un Computador .....	- 92 -
3.2.5.	Análisis de los Resultados Obtenidos en las Diferente Pruebas.....	- 107 -
3.3.	Verificación de la hipótesis .....	- 111 -

<b>CAPÍTULO IV</b> .....	- 115 -
<b>4. CONCLUSIONES Y RECOMENDACIONES</b> .....	- 115 -
4.1. Conclusiones.....	- 115 -
4.2. Recomendaciones .....	- 116 -
<b>BIBLIOGRAFÍA</b> .....	- 119 -
<b>Anexo</b> .....	- 122 -
ANEXO A: IMPORTACIÓN DE LIBRERÍAS A TRABAJARSE PARA EL ALGORITMO DE RECONOCIMIENTO DE PLACAS VEHICULARES EN MOVIMIENTO EMPLEANDO LA TARJETA BEAGLEBONE BLACK	
ANEXO B: ALGORITMO DE RECONOCIMIENTO DE PLACAS VEHICULARES EN MOVIMIENTO EMPLEANDO LA TARJETA BEAGLEBONE BLACK	

## ÍNDICE DE TABLAS

TABLA 1 VEHÍCULOS REPORTADOS COMO ROBADOS EN LOS ÚLTIMOS 4 AÑOS..	- 3 -
TABLA 2 TIPOS DE PLACAS POR SERVICIOS .....	- 30 -
TABLA 3 FUNCIONES DISPONIBLE PARA MANEJO DE TKINTER.....	- 66 -
TABLA 4 PLACAS VEHICULARES DETECTADAS POR EL SISTEMA CON UN PORCENTAJE DE 100%.....	- 76 -
TABLA 5 EJEMPLOS DE VEHÍCULOS DETECTADOS CON UN PORCENTAJE DE 100% .....	- 78 -
TABLA 6 PLACAS VEHICULARES DETECTADAS CON UN PORCENTAJE MENOR AL 100% .....	- 80 -
TABLA 7 EJEMPLOS DE VEHÍCULOS DETECTADOS CON UN PORCENTAJE MENOR AL 100% .....	- 81 -
TABLA 8 ESPECIFICACIONES TÉCNICAS DEL COMPUTADOR .....	- 89 -
TABLA 9 RESULTADOS OBTENIDOS REALIZANDO EL PROCESAMIENTO MEDIANTE EL USO DE UN COMPUTADOR .....	- 89 -
TABLA 10 VEHÍCULO DE PRUEBA N° 1.....	- 93 -
TABLA 11 VEHÍCULO DE PRUEBA N° 2.....	- 94 -
TABLA 12 VEHÍCULO DE PRUEBA N° 3.....	- 95 -
TABLA 13 VEHÍCULO DE PRUEBA N° 4.....	- 96 -
TABLA 14 VEHÍCULO DE PRUEBA N° 5.....	- 97 -
TABLA 15 VEHÍCULO DE PRUEBA N° 6.....	- 98 -
TABLA 16 VEHÍCULO DE PRUEBA N° 7.....	- 99 -
TABLA 17 VEHÍCULO DE PRUEBA N° 8.....	- 100 -
TABLA 18 VEHÍCULO DE PRUEBA N° 9.....	- 101 -
TABLA 19 VEHÍCULO DE PRUEBA N° 10.....	- 102 -
TABLA 20 VEHÍCULO DE PRUEBA N° 11.....	- 103 -
TABLA 21 VEHÍCULO DE PRUEBA N° 12.....	- 104 -
TABLA 22 VEHÍCULO DE PRUEBA N° 13.....	- 105 -
TABLA 23 VEHÍCULO DE PRUEBA N° 14.....	- 106 -
TABLA 24 TIEMPOS DE PROCESAMIENTO ENTRE SISTEMAS .....	- 109 -
TABLA 25 DATOS OBTENIDOS DE LAS PRUEBAS REALIZADAS CON LA TARJETA BEAGLEBONE BLACK REVC.....	- 111 -
TABLA 26 MODELO DE TABLA DE FRECUENCIAS ESPERADAS.....	- 112 -
TABLA 27 VALORES CALCULADOS DE FRECUENCIAS ESPERADAS .....	- 112 -

## ÍNDICE DE FIGURAS

FIGURA 1 PORCENTAJE DE VEHÍCULOS REPORTADOS COMO ROBADOS EN 2012 -----	4 -
FIGURA 2 PORCENTAJE DE VEHÍCULOS REPORTADOS COMO ROBADOS EN 2013 -----	4 -
FIGURA 3 PORCENTAJE DE VEHÍCULOS REPORTADOS COMO ROBADOS EN 2012 -----	5 -
FIGURA 4 PORCENTAJE DE VEHÍCULOS REPORTADOS COMO ROBADOS EN 2015 -----	5 -
FIGURA 5 LIBERTADES DE UN SOFTWARE LIBRE -----	10 -
FIGURA 6 VISIÓN ARTIFICIAL EMPLEADA A LA DETECCIÓN DE OBJETOS -----	11 -
FIGURA 7 PARTES DE UN SISTEMA DE ADQUISICIÓN -----	12 -
FIGURA 8 LOGO SOFTWARE LIBRE OPENCV -----	13 -
FIGURA 9 ESTRUCTURA DE OPENCV -----	17 -
FIGURA 10 EJEMPLO SELECCIÓN DE REGIÓN DE INTERÉS -----	18 -
FIGURA 11 CÁMARA LOGITECH C920 -----	18 -
FIGURA 12 ETAPAS PARA EL PROCESAMIENTO DE IMÁGENES -----	19 -
FIGURA 13 ADQUISICIÓN DE LA IMAGEN PLACA VEHICULAR -----	20 -
FIGURA 14 SEGMENTACIÓN DE UNA IMAGEN -----	21 -
FIGURA 15 BINARIZACIÓN DE UNA IMAGEN -----	21 -
FIGURA 16 RECONOCIMIENTO DE CARACTERES -----	22 -
FIGURA 17 COMPARACIÓN DE MOTORES OCR -----	22 -
FIGURA 18 TARJETA BEAGLEBONE BLACK -----	24 -
FIGURA 19 LOGO PYTHON -----	25 -
FIGURA 20 LOGO MYSQL FUENTE (MYSQL, 2016) -----	28 -
FIGURA 21 PLACA VEHICULAR PERTENECIENTE A ECUADOR -----	30 -
FIGURA 22 EXTRACCIÓN DEL SISTEMA OPERATIVO DEBIAN 7.9 CON 7 ZIP -----	32 -
FIGURA 23 INSTALACIÓN DEL SISTEMA OPERATIVO EN LA MICROSD -----	33 -
FIGURA 24 ARRANQUE DEL SISTEMA OPERATIVO DESDE LA MICROSD -----	33 -
FIGURA 25 INTERFAZ GRÁFICA SISTEMA OPERATIVO DEBIAN7.9 -----	36 -
FIGURA 26 ARREGLO MATRICIAL USANDO NUMPY -----	39 -
FIGURA 27 CÓMO EJECUTAR UN ARCHIVO DE PYTHON -----	41 -
FIGURA 28 IMAGEN CARGADA EN FORMATO RGB DESDE PYTHON -----	41 -
FIGURA 29 IMAGEN CARGADA EN FORMATO ESCALA DE GRISES DESDE PYTHON -----	42 -
FIGURA 30 CAPTURA DE IMAGEN POR MEDIO DE LA CÁMARA LOGITECH C920 CON PYTHON -----	44 -
FIGURA 31 CAPTURA DE IMAGEN POR MEDIO DE LA CÁMARA LOGITECH C920 CON PYTHON EN FORMATO ESCALA DE GRISES -----	45 -

FIGURA 32 DIAGRAMA DE FLUJO DEL FUNCIONAMIENTO GENERAL DEL PROYECTO -----	46 -
FIGURA 33 RECORTE DE LA REGIÓN DE INTERÉS -----	47 -
FIGURA 34 IMAGEN CONVERTIDA A ESCALA DE GRISES (A) Y CANNY (B) -----	48 -
FIGURA 35 IMAGEN DILATADA (A) PLACA DETECTADA (B) -----	49 -
FIGURA 36 (A) RECORTE DE LA PLACA VEHICULAR, (B) CORRECCIÓN DE ÁNGULO DE LA IMAGEN, (C) APLICACIÓN DE OPERACIONES MORFOLÓGICAS Y BINARIZACIÓN DE LA FOTOGRAFÍA. -----	50 -
FIGURA 37 PLACA PROCESADA POR OCR TESSERACT -----	53 -
FIGURA 38 RESULTADO DE MÉTODOS DE VALIDACIÓN DE CARACTERES-----	55 -
FIGURA 39 DIAGRAMA DE FLUJO DEL FUNCIONAMIENTO DE LA VALIDACIÓN DE CARACTERES -----	57 -
FIGURA 40 ACCESO A MODO USUARIO MYSQL -----	59 -
FIGURA 41 CREACIÓN Y LISTADO DE BASES DE DATOS MYSQL -----	59 -
FIGURA 42 SELECCIÓN DE BASE DE DATOS MYSQL -----	60 -
FIGURA 43 CREACIÓN DE TABLAS Y ETIQUETAS DE BASE DE DATOS -----	60 -
FIGURA 44 DESPLIEGUE DE TABLAS CREADAS EN LA BASE DE DATOS-----	61 -
FIGURA 45 ACCESO A BASE DE DATOS A TRAVÉS DE PYTHON-----	62 -
FIGURA 46 REGISTRO DE PLACAS VEHICULARES DETECTADAS-----	63 -
FIGURA 47 DIAGRAMA DE FLUJO DEL FUNCIONAMIENTO DE LA BASE DE DATOS-----	64 -
FIGURA 48 GRAFICA DE INTERFAZ DE USUARIO EN PYTHON -----	65 -
FIGURA 49 50 VENTANA PRINCIPAL GENERADA POR TKINTER -----	68 -
FIGURA 51 GESTIÓN DE GEOMETRÍA TKINTER -----	70 -
FIGURA 52 CREACIÓN DE VENTANA DE TRABAJO TKINTER -----	70 -
FIGURA 53 PRESENTACIÓN Y UBICACIÓN DE UNA VARIABLE EN LA VENTANA TKINTER -----	72 -
FIGURA 54 VARIABLES PRESENTADAS EN TKINTER-----	72 -
FIGURA 55 VISUALIZACIÓN DE IMÁGENES EN VENTANA TKINTER -----	73 -
FIGURA 56 VISUALIZACIÓN DE SLIDERFRAME Y LISTBOX EN VENTANA TKINTER EN PYTHON-----	73 -
FIGURA 57 DESCRIPCIÓN DE LA UBICACIÓN DE CÁMARA RESPECTO AL VEHÍCULO DE PRUEBA-----	75 -
FIGURA 58 RECONOCIMIENTO DE PLACAS VEHICULARES DE 8:00AM – 8:30AM -----	82 -
FIGURA 59 RECONOCIMIENTO DE PLACAS VEHICULARES DE 10:00AM – 10:30AM-----	83 -
FIGURA 60 RECONOCIMIENTO DE PLACAS VEHICULARES DE 12:00PM – 12:30PM-----	83 -

FIGURA 61 RECONOCIMIENTO DE PLACAS VEHICULARES DE 1:00PM – 1:30PM -----	84 -
FIGURA 62 RECONOCIMIENTO DE PLACAS VEHICULARES DE 2:00PM – 2:30PM -----	84 -
FIGURA 63 RECONOCIMIENTO DE PLACAS VEHICULARES DE 4:00PM – 4:30PM -----	85 -
FIGURA 64 RECONOCIMIENTO DE PLACAS VEHICULARES DE 5:00PM – 5:30PM -----	85 -
FIGURA 65 RECONOCIMIENTO DE PLACAS VEHICULARES DE 6:00PM – 6:30PM -----	86 -
FIGURA 66 RESULTADOS DEL RECONOCIMIENTO DE PLACAS VEHICULARES EN LOS DÍAS 1 Y 2 -----	86 -
FIGURA 67 GRÁFICA PORCENTUAL DE RESULTADOS POR HORAS -----	88 -
FIGURA 68 REPRESENTACIÓN DE RESULTADOS OBTENIDOS POR COMPUTADOR -----	90 -
FIGURA 69 RESULTADO PORCENTUAL DE DATOS OBTENIDOS POR COMPUTADOR -----	90 -
FIGURA 70 GRÁFICA COMPARATIVA DE RESULTADOS ENTRE LA BEAGLEBONE Y UN COMPUTADOR -----	91 -
FIGURA 71 VEHÍCULOS INGRESADOS EN LA BASE DE DATOS -----	92 -
FIGURA 72 CONSUMO DE MEMORIA DEL COMPUTADOR ANTES DE EJECUTAR EL PROGRAMA DE RECONOCIMIENTO DE PLACAS VEHICULARES-----	107 -
FIGURA 73 CONSUMO DE MEMORIA DEL COMPUTADOR DESPUÉS DE EJECUTAR EL PROGRAMA DE RECONOCIMIENTO DE PLACAS VEHICULARES---	107 -
FIGURA 74 CONSUMO DE MEMORIA DE LA TARJETA ANTES DE EJECUTAR EL PROGRAMA DE RECONOCIMIENTO DE PLACAS VEHICULARES-----	108 -
FIGURA 75 CONSUMO DE MEMORIA DE LA TARJETA DESPUÉS DE EJECUTAR EL PROGRAMA DE RECONOCIMIENTO DE PLACAS VEHICULARES---	108 -
FIGURA 76 FPS POR RESOLUCIÓN DE IMAGEN CON DIFERENTES TECNOLOGÍAS -----	109 -
FIGURA 77 TIEMPO DE PROCESAMIENTO DE IMAGEN POR SU RESOLUCIÓN CON DIFERENTES TECNOLOGÍAS-----	110 -



## RESUMEN

El proyecto de investigación presenta el diseño e implementación de un sistema de identificación de vehículos robados en movimiento en la calle Quijano Ordoñez junto al parque San Francisco de la ciudad de Latacunga, donde se analizó el funcionamiento de la tarjeta Beaglebone Black RevC es el adecuado, como sistema operativo de procesamiento en conjunto con el lenguaje de programación python bajo la distribución de software libre. El proyecto de investigación se enfoca en el reconocimiento de placas vehiculares en tiempo real, empleando una cámara web c920 logitech, la tarjeta Beaglebone Black RevC como unidad de procesamiento, considerando para ellos el desarrollo de un algoritmo para la identificación y extracción de la placa vehicular de un automotor, donde la misma será comparada con una base de datos pre establecida que contará con un algoritmo para conocer el porcentaje de similitud entre placas vehiculares, generando un mensaje de aviso para tomar las medidas pertinentes.

### **PALABRAS CLAVE:**

- **TARJETA BEAGLEBONE BLACK REVC**
- **PROCESAMIENTO DE IMÁGENES**
- **SOFTWARE LIBRE PYTHON**

## ABSTRACT

The research project presents the design and implementation of a system for identifying stolen moving Quijano Ordoñez street next to San Francisco in the city of Latacunga, where the operation was analyzed vehicles BEAGLEBONE Black RevC card is adequate as an operating system processing in conjunction with the programming language python that uses free software distribution. This research project focuses on the recognition of license plates in real time using a webcam c920 logitech and a BEAGLEBONE Black RevC card as processing unit, this work develops an algorithm to identify and separate the license plate of a vehicle, where in the information of the plate will be compared with a pre-established database which will feature an algorithm to determine the percentage of similarity between the nearest license plates, generating a warning message to take appropriate measure.

### KEYWORDS:

- **CARD BEAGLEBONE BLACK REV C**
- **IMAGE PROCESSING**
- **FREE SOFTWARE PYTHON**

## **CAPÍTULO I**

### **1. INTRODUCCIÓN**

#### **1.1. Prólogo**

El procesamiento digital con énfasis al tratamiento de imágenes actualmente es empleado a gran escala en múltiples aplicaciones industriales, comerciales y de investigación para facilitar el control en las diversas actividades que realiza el ser humano.

En el presente trabajo de investigación se presenta el desarrollo e implementación de un algoritmo programado en software libre (OPENCV) en conjunto con una tarjeta BeagleBone Black RevC y una cámara de video, las mismas que se emplearán para determinar las placas de un vehículo mientras este se encuentra en movimiento dentro de una zona urbana de la ciudad, adicionalmente se comparará los datos obtenidos, producto del procesamiento de imágenes con una base datos previamente diseñada, la misma que constará con el listado de vehículos robados, así como información general del vehículo detectado, generando una señal de alerta en un ordenador ubicado en un determinado centro de control, para ejecutar acciones que permitan controlar dichas operaciones.

#### **1.2. Antecedentes**

El desarrollo de tecnologías para la detección de patrones y procesamiento de imágenes, en los últimos años, ha venido siendo el eje fundamental para la identificación de placas vehículos dentro del sector automotor. El procesamiento digital para el tratamiento de imágenes ostenta sus primeros ejemplos significativos a inicios de la década de los 60 y está ligado con el desarrollo y evolución de las computadoras en conjunto con el constante desarrollo tecnológico del hardware, debido a los altos requerimientos de velocidad y procesamiento para almacenar y procesar las imágenes en tiempo real.

El procesamiento de imágenes empleado específicamente en el área de detección de placas vehiculares se ha empleado en trabajos como “reconocimiento en tiempo real de las matrículas de los vehículos en Sri Lanka” donde mediante algoritmos desarrollados en el software Sri Lanka y empleando técnicas de transformación identifican las placas vehiculares. (D. Wijetunge, 2011) Otra investigación de interés se presente en el tema “Algoritmo de aprendizaje para el reconocimiento de colores de placas de matrícula” en donde el algoritmo de aprendizaje empleado para el reconocimiento de placas vehiculares se define mediante el color de las placa analizando tres componentes del espacio de color como son el tono-saturación-valor (HSV), dando como resultado una mejora la precisión y capacidad de adaptación del color a los algoritmos de reconocimiento. (Wang, Zhang, Man, & Yu, 2010)

Una aplicación de interés se la presenta en el trabajo “Extracción y reconocimiento de la placa de matrícula del vehículo para pasar bajo ambiente exterior”, en donde se resalta el uso de un método basado en la morfología para tratar las imágenes y conseguir aislar la placa vehicular de la imagen digital del coche obtenida por una cámara digital en diferentes circunstancias, tales como la iluminación, decantación, distancia y el ángulo (Kasaei, 2011), como se ha podido apreciar el implemento de sistemas de detección continúan desarrollando de cada vez ofertando nuevas técnicas y sistemas a la par de desarrollo tecnológico del país.

Existen varias técnicas de reconocimiento de placas vehiculares, desde las más simple como identificar el color de las placas hasta sistemas más complejos que detectan el código de identificación de los automotores sin movimiento, es decir, cuando los vehículos se encuentran detenidos, dichos sistemas pueden ser observados en la actualidad en controles de tráfico y seguridad como lo son en parqueaderos, peajes, estacionamientos en centro comerciales, control de velocidad en panamericanas. Estas aplicaciones ya existen en varios países donde procesan la información obtenida de las cámaras y lo comprueban en

una base de datos para poder identificar si algún vehículo reporta algún inconveniente con la ley.

El robo vehicular se ha convertido en una actividad delictiva con un alto nivel de organización que genera un gran impacto en la sociedad en diferentes regiones a nivel mundial, debido a que los robos no solo se realizan para obtener un beneficio económico del mismo, también es un medio que permite financiar otros delitos como secuestros, actividades terroristas, entre otras actividades delictivas.

En la Provincia de Cotopaxi se determina que el robo vehicular también es un problema. En la Tabla 1 se puede visualizar el número de vehículos reportados a la policía Judicial como robada por mes en los últimos 4 años.

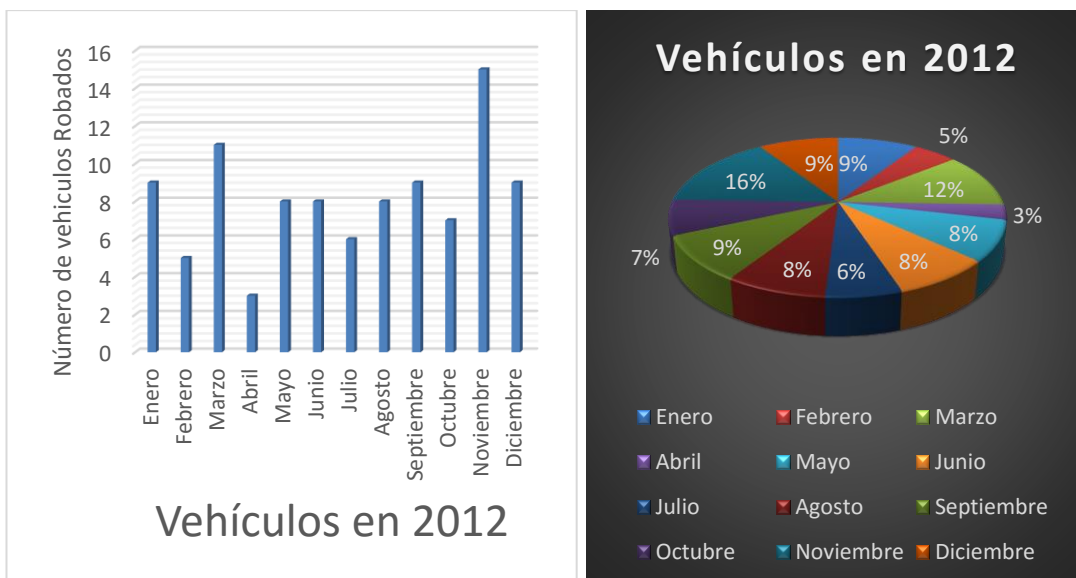
**Tabla 1**

**Vehículos reportados como robados en los últimos 4 años**

<b>Número de vehículos reportados robados en el año de 2012</b>											
Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
9	5	11	3	8	8	6	8	9	7	15	9
<b>Número de vehículos reportados robados en el año de 2013</b>											
Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
7	17	6	7	8	10	15	9	10	5	5	6
<b>Número de vehículos reportados robados en el año de 2014</b>											
Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
5	2	1	5	14	9	5	1	5	6	8	10
<b>Número de vehículos reportados robados en el año de 2015</b>											
Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
9	3	8	9	9	5	2	8	3	4	11	4

Fuente: (Policia Judicial Cotopaxi, 2016)

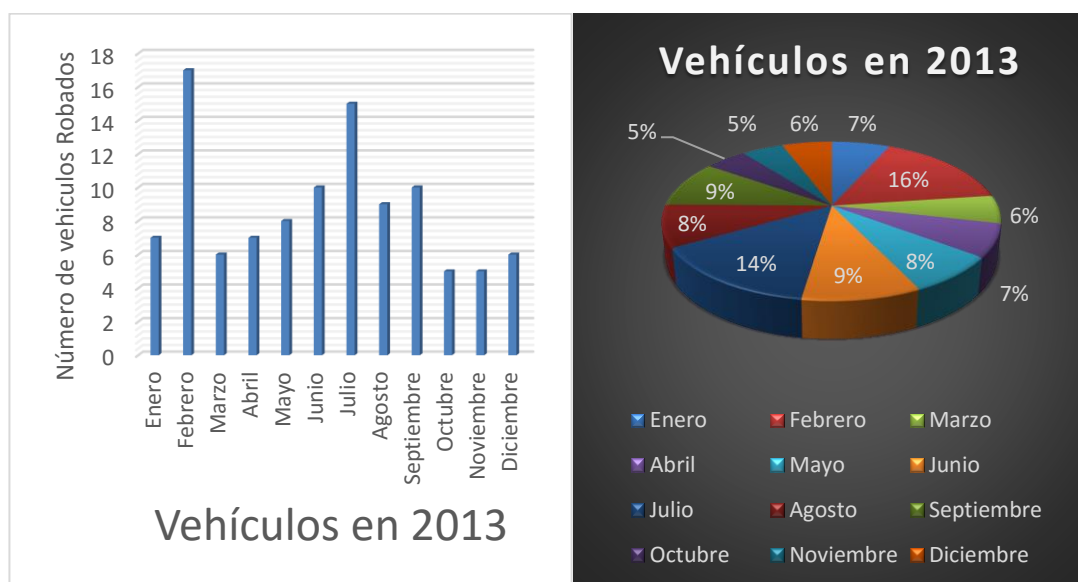
La figura 1 presenta el porcentaje de vehículos robados en el año 2012 semestralmente, siendo marzo el mes con más reportes por este tipo de delitos, para este año el reporte total en la provincia de Cotopaxi fue de 98 de los cuales solo 19 fueron ingresados como recuperados.



**Figura 1 Porcentaje de vehículos reportados como robados en 2012**

Fuente: (Policia Judicial Cotopaxi, 2016)

La figura 2 presenta el porcentaje de vehículos robados en el año 2013 por semestre, siendo febrero con un 16% el mes con más reportes por este tipo de delito, para este año el reporte total en la provincia de Cotopaxi fue de 105, presentando un incremento con respecto al año 2012 de los cuales solo 7 fueron recuperados por la policía.

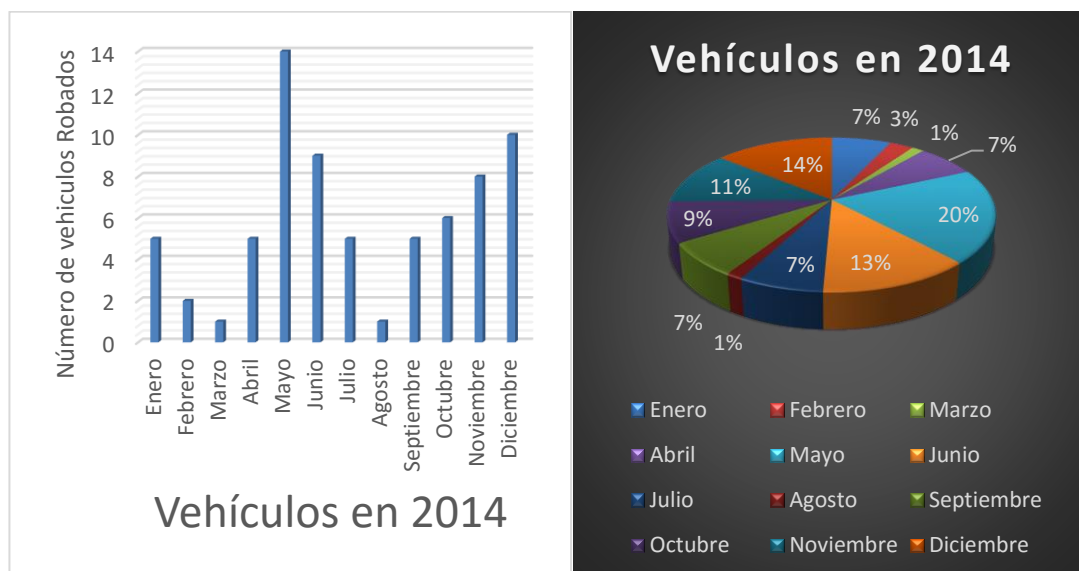


**Figura 2 Porcentaje de vehículos reportados como robados en 2013**

Fuente: (Policia Judicial Cotopaxi, 2016)

La figura 3 presenta el porcentaje de vehículos robados en el año 2014 por semestre, siendo mayor con un 20% el mes con más reportes por

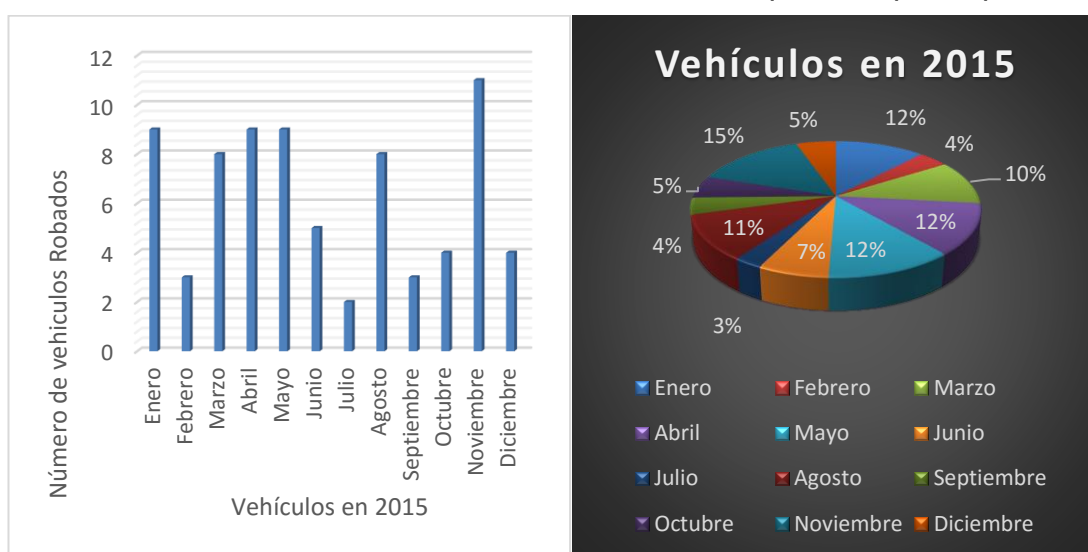
este tipo de delitos, para este año el reporte total en la provincia de Cotopaxi fue de 71 presentando una disminución de delitos a diferencia de años anteriores de los cuales solo 7 fueron recuperados por la policía.



**Figura 3 Porcentaje de vehículos reportados como robados en 2012**

Fuente: (Policia Judicial Cotopaxi, 2016)

La figura 4 presenta el porcentaje de vehículos robados en el año 2015 por semestre, siendo noviembre con un 15% el mes con más reportes por este tipo de delitos, para este año el reporte total en la provincia de Cotopaxi fue de 75, presentando un pequeño incremento en este tipo de delito a diferencia del año 2014. Solo 5 fueron recuperados por la policía.



**Figura 4 Porcentaje de vehículos reportados como robados en 2015**

Fuente: (Policia Judicial Cotopaxi, 2016)

### **1.3. Planteamiento del problema**

A nivel mundial los sistemas y métodos empleados actualmente para identificar vehículos que han sido robados poseen muchas limitantes, como el registro de dichos vehículos mediante agentes de tránsito que derivan en procesos lentos y poco eficientes al momento de rastrear los automotores debido a la lentitud del proceso de la información, de igual manera los sistemas desarrollados para identificación de placas vehiculares hoy en día poseen muchas restricciones, su principal razón es la necesidad de que el vehículo a ser investigado debe encontrarse detenido, es decir, no debe hallarse en movimiento al momento de tomar la imágenes que permitan realizar el procesamiento de la información obtenida, dando como resultado un sistema eficaz en lugares como parqueaderos públicos, estacionamientos en centros comerciales o institucionales, pero presenta grandes deficiencias en zonas urbanas que posean un constante movimiento vehicular, como son las mismas calles de la ciudad.

Los dispositivos empleados para la toma y proceso de información son costosos ya que son manejados por ordenadores que muchas veces realizan múltiples tareas a la vez, generando retardos en reportes y muchas veces pérdidas de información.

### **1.4. Justificación**

La detección de placas vehiculares en determinadas zonas de una ciudad a través del uso de cámaras de vigilancia permite incrementar la protección vehicular en un área específica, optimizando los recursos obtenidos de cámaras para el manejo de información y monitoreo de las actividades cotidianas.

Por tal motivo es necesario el desarrollo de un sistema totalmente enfocado en el diseño de un algoritmo que permita identificar las placas de vehículos de automotores que se encuentren en movimiento por medio de software libre, permitiendo además involucrar las nuevas



tendencias tecnológicas e investigativas. Además los dispositivos de adquisición de información en la actualidad presentan grandes ventajas y enormes facilidades para la toma de datos, como son cámaras de alta velocidad y tarjetas de procesamiento que proporcionarían herramientas en el proceso de detección de placas vehiculares, en conjunto con un código o algoritmo en software libre adecuado para el procesamiento de datos, proporcionando un sistema sujeto a estándares, flexible y fácil de operar a diferencia de los sistemas actuales

## **1.5. Objetivos del proyecto**

### **1.5.1. General**

Diseñar e implementar un sistema para detección de vehículos robados en movimiento, empleando tecnología Beaglebone mediante software libre

### **1.5.2. Específicos**

- Obtener conocimientos acerca de la utilización, configuración y programación de las tarjetas Beaglebone Black RevC.
- Investigar la sintaxis, normas y librerías del software libre Open CV para el procesamiento de imágenes.
- Instalar y calibrar una cámara para la adquisición de las señales de video en tiempo real, e implementar algoritmos para la detección de placas vehiculares en movimiento mediante software libre compatible con tecnología Beaglebone Black RevC.
- Detectar y extraer el código de la placa vehicular desde una señal de video obtenida en tiempo real.
- Normalizar las imágenes de la placa vehicular obtenidas a través de técnicas de corrección geométrica.
- Realizar pruebas para verificar el correcto funcionamiento de la detección.
- Comparar la información adquirida con una base de datos misma que contenga información de vehículos reportados.

- Diseñar un HMI para la presentación de la placa vehicular identificada por la cámara.

## **1.6. Software libre**

De forma común puede mencionarse al software libre como el grado más alto de libertad que posee un desarrollador y un usuario al momento de utilizar un determinado programa, el término libertad se refiere a que un usuario puede copiar, distribuir, cambiar, mejorar y personalizar un programas de acuerdo a sus preferencias sin ningún tipo de recargo, para que un programa sea considerado software libre tiene que cumplir con 4 aspectos que son fundamentales al momento de especificar la libertad del programa. (ARENAS, 2003)

- El programa no debe poseer limitaciones o restricciones de ningún tipo.
- El programa puede ser copiado y distribuido libremente en cuantas computadoras se requiera sin límite alguno.
- Acceso total al código fuente, este apartado es muy importante, pues es donde el usuario puede realizar modificaciones al programa y adecuarlo a sus necesidades.
- La libertad de usar el programa, con cualquier propósito

### **1.6.1. Tipos de licencias de software libre**

Una licencia es un permiso formal con carácter contractual que un desarrollador de software da a un usuario para emplearlo de manera legal. Desde el punto de vista del software libre, existen distintas variantes del concepto o grupos de licencias: (Mendoza, 2012)

**Licencias GPL.** - Licencia Pública General de GNU (GNU GPL). Es una licencia que reserva los derechos de autor y da al usuario permisos para la redistribución y modificación bajo términos diseñados.

**Licencias AGPL.** - La Licencia Pública General de Affero es una licencia procedente de la Licencia Pública General de GNU y está desarrollada específicamente para software que corra en servidores de red.

**Licencias estilo BSD.** - para sistemas operativos BSD, consiste en reservar los derechos sobre un programa y luego añadirle los términos de distribución y modificación.

**Licencias estilo MPL y derivadas.** - esta licencia llamada copyleft débil, se encargaba de algunos puntos que no fueron tenidos en cuenta por las licencias BSD y GNU, estas licencias otorgan privilegios especiales como es la posibilidad de realizar modificaciones hechas por cualquiera al código.

## **1.6.2. Ventajas y desventajas del software libre**

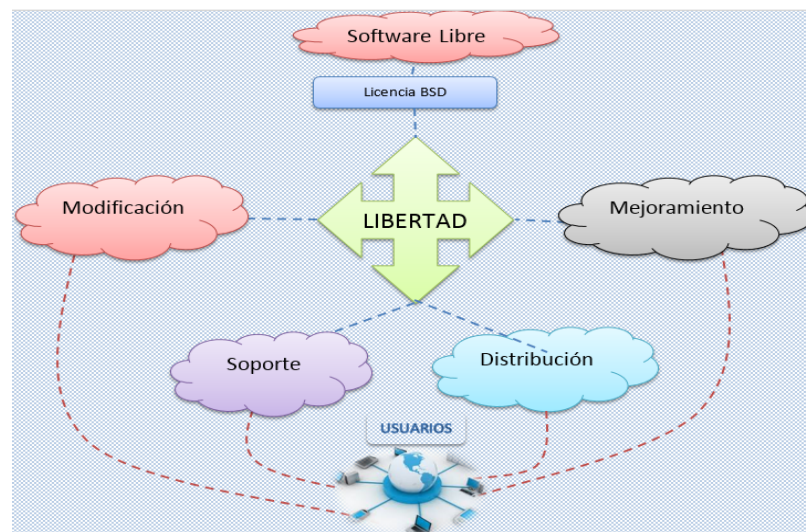
### **Ventajas**

- Cualquier persona puede disponer del software libre bajo las condiciones de la licencia.
- Es gratuito o de bajo costo.
- Puede ser distribuido, modificado, mejorado o personalizado, de acuerdo a los requerimientos del usuario.
- Gran comunidad para intercambio de conocimiento y consultas, facilitando la innovación tecnológica
- Rápida corrección de errores facilitado por el trabajo comunitario a través de Internet y de su libre acceso al código fuente.
- El usuario puede administrar libremente su crecimiento y operación con total autonomía.
- Facilidad para personalizar el software de acuerdo a las necesidades del usuario.
- Posibilidad de traducir el mismo a cualquier idioma, inclusive a una lengua regional o indígena.
- Fácil acceso por parte del sector educativo público y privado.

- Elimina el sistema operativo monousuario. Ya que permite el uso y trabajo de varios usuarios al mismo tiempo. (Mendoza, 2012)

### Desventajas

- Dificultad en el intercambio de archivo, dan errores o se pierden datos.
- Desconocimiento ya que el usuario común está muy familiarizado con los soportes del sistema operativo Windows.
- Ausencia de garantía. El software libre no se hace responsable por los daños.
- Para su configuración se requieren conocimientos previos de funcionamiento del sistema operativo.
- No existe un control de calidad previo.
- Baja expansión de su uso en centros educativos.
- Baja difusión en publicaciones. (Mendoza, 2012)



**Figura 5 Libertades de un Software Libre**

Fuente: (Martínez, 2015)

### 1.7. Visión artificial

Es un campo que, mediante el empleo de diferentes técnicas, permite la detección, procesamiento, análisis e identificación de diversos tipos de información a través de imágenes digitales, la visión artificial está conformada por un conjunto de métodos o técnicas (captación de

imágenes, procesado, correcciones, segmentaciones e interpretación de resultados) con un único fin el análisis de una imagen. Con la visión artificial se puede acceder a diversas tareas tales como:

- Automatizar tareas repetitivas de inspección.
- Implementar controles de calidad de productos.
- Inspeccionar de objetos sin contacto físico.
- Monitoreo y detección de objetos
- Disminuir el tiempo de procesos automatizado, Etc.



**Figura 6 Visión artificial empleada a la detección de objetos**

Fuente: (Muñoz Manso, 2014)

### **1.7.1. Ventajas y Beneficios de la visión artificial**

El empleo de los nuevos sistemas basados en visión artificial ha generado un mundo de beneficios en las aplicaciones tanto a nivel industrial como comercial son:

- Calidad y aumento en la producción obteniendo resultados 100% confiables.
- Erradicación de errores humanos.
- Monitoreo y detección de productos o funcionamientos defectuosos en la línea de producción.
- Toma de lecturas o medidas sin contacto
- Generación de históricos con imágenes

### 1.7.2. Componentes de un sistema de visión artificial

Un sistema de visión artificial (SVA) esta estructura con varios elementos de hardware para realizar una acción específica, los elementos mínimos que debe poseer un sistema SVA son los siguientes:

**Sensor Óptico:** El sensor puede ser una cama la cual se encargue de generar una imagen de una determinada zona, objeto, etc. El sensor a emplearse depende de la aplicación.

**Tarjeta de adquisición de imagen:** Es la encargada de digitalizar la imagen generada por el sensor óptico.

**Computador:** Es la Unidad central encargada de almacenar, procesar y modificar la imagen digitalizada para detectar un patrón o característica específica.

**Monitor de Video:** Es una unidad periférica empleada para visualizar la imagen que se capta a través del sensor previamente realizado el procesamiento necesario.



**Figura 7 Partes de un Sistema de adquisición**

### 1.7.3. Software para tratamiento de imágenes por visión Artificial

El software y herramientas para el tratamiento de imágenes son muy variados, actualmente existen muchas librerías y herramientas

software para facilitar la labor de programación de estos SVA siendo los más destacados:

Matlab que es un entorno de programación matricial muy sencilla y fácil de manipular.

Python, Eclipse, C++ que son entornos de programación de alto nivel y altos recursos para tiramiento y procesamiento de imágenes.

Programas de usuario: PhotoShop, Paint Shop Pro, Gimp, Xv y librerías OpenGL para aceleración por hardware.

### 1.8. Opencv

OpenCV (Open Source Computer Vision Library) es una biblioteca desarrollada en base a código abierto que incluye cerca de 2500 algoritmos para aplicaciones de visión artificial y machine learning. Está publicada bajo una licencia BSD (Berkeley Software Distribution), lo cual permite usar, modificar y personalizar el código además que puede ser usada libremente para propósitos comerciales y académicos.



**Figura 8 Logo Software libre OpenCV**

Fuente: (OpenCV, 13)

Es multiplataforma, existiendo versiones para Windows, GNU/Linux, Mac OSX, iOS y Android. Así como también en interfaces para C, C++, Python y Java. Se enfoca principalmente hacia capturas de video tiempo real para procesamiento de imágenes, en la

actualidad esta librería oferta una serie de herramientas para aplicaciones donde se requiera: (OpenCV, 13)

- Captura en tiempo real
- Archivo de vídeo de importaciones
- Tratamiento de imagen básica (brillo, contraste, umbral, ...)
- Detección de objetos (cara, cuerpo, ...)
- Reconocimiento de regiones

### **1.8.1. Origen y Uso**

Originariamente presenta sus primeros pasos y desarrollo en los laboratorios de Intel en el año de 1990, con el único objetivo de impulsar la investigación, desarrollo y uso de la visión artificial o por computadora como herramienta base para nuevos estudios y aplicaciones. Los usos de OpenCV van desde seguridad y vigilancia en tiempo real, monitoreo de equipo minero, pasando por asistencia en visión de robots autónomos, detección de accidentes en piletas de natación en Europa, el chequeo de escombros en pistas de aterrizaje en Turquía, hasta el chequeo de etiquetas de productos en todo el mundo o el reconocimiento de rostros en Japón. (Auchterberge, 2014)

### **1.8.2. Características de opencv**

Al ser OpenCV una librería compuesta con miles de algoritmos de programación para diferentes aplicaciones, goza de una cantidad ilimitada de rutinas, que en conjunto con diversos lenguajes de programación ofrecen una adecuada asistencia para el tratamiento de código e información (imágenes digitales), ya que suministra varios beneficios como:

- Proporciona herramientas para interpretar una imagen.
- Provee una serie de ejemplo prácticos que facilitan la comprensión de comandos, además de contar con grandes soportes de ayuda en la red.



- Es una librería que funciona para C/C++, por ser una librería completa en funciones para realizar operaciones sobre imágenes.
- Es un conjunto de librerías con algoritmos totalmente gratuitos.
- Permite la realización de estudios de movimiento y seguimiento de objetos, así como reconocimientos faciales.
- La biblioteca está en constante actualización.

### **1.8.3. Aplicaciones de OpenCV.**

OpenCV se muestra como una herramienta sencilla e interactiva de diseño mediante un software libre con grandes prestaciones, que permite a los diferentes usuarios, emplear una serie algoritmos de programación para una extensa variedad de proyectos con fines detección o monitoreo tales como:

- **Detección reconocimiento y rastreo de objetos**  
Empleado para la detección diversos objetos "olvidados" en aeropuertos, centros comerciales como valijas, bolsas, paquetes, etc.
- **Detección de movimiento:**  
Ampliamente utilizados en modernos sistemas de seguridad en oficinas, bancos, etc.
- **Grabación de frames para sistemas de seguridad:**  
Empleado de manera conjunta una PC con el software OpenCV se obtienen imágenes de personas u objetos que son grabadas en base a su movimiento, para posteriormente en conjunto con personal de seguridad revisar detalladamente y analizar las imágenes para detectar anomalías.

- **Reconocimiento de rostros:**

Empleado ampliamente en sistemas de seguridad para detectar posibles personas con antecedentes penales, personas extraviadas etc. Las diversas aplicaciones para el reconocimiento de rostros son diversas y pueden utilizarlas personas que no tiene acceso a la tecnología.

#### 1.8.4. Estructura y contenido de OpenCV

La biblioteca de OpenCV comprende cerca de 2500 algoritmos los cuales pueden clasificarse dentro de 4 grandes grupos dependiendo su aplicación como son:

**CXCORE:** En este apartado se puede localizar las estructuras y algoritmos básicos que usan las demás funciones. Ejemplo: suma, media. Operaciones-binarias. etc.

**CV:** Presenta algoritmos donde para las funciones principales de procesamiento de imágenes. Ejemplo: Erosión, Dilatación, Operaciones de umbralización, etc.

**HighGUI:** Muestra todo lo relacionado a la interfaz gráfica de OpenCV así como comandos para importar imágenes y video.

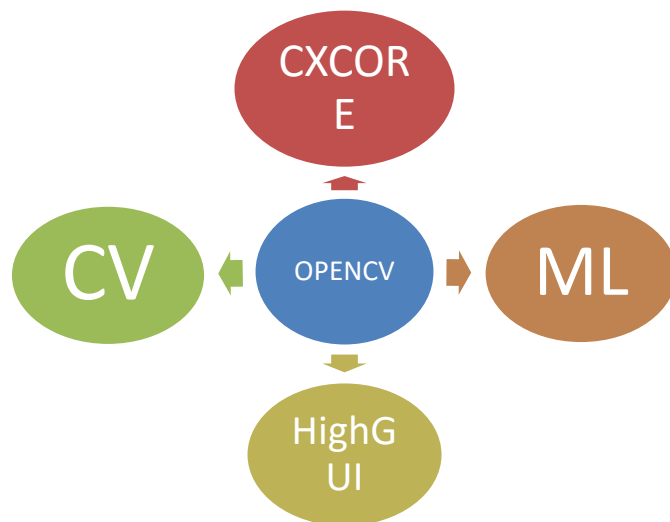
**ML:** Cuenta con algoritmos de aprendizaje y clasificadores. (Martínez, 2015)

OpenCV tiene una estructura modular, lo que significa que el paquete que incluye varias librerías compartidas o estáticas. Entre los módulos disponibles tenemos:

- core: Un módulo compacto que define las estructuras de datos esenciales, como el arreglo denso2 multidimensional Mat y funciones básicas usadas por los demás módulos.

- imgproc: Un módulo de procesamiento de imágenes que incluye filtrado lineal y no lineal de imágenes, transformaciones geométricas (redimensionar, transformación afín y de perspectiva, re-mapeo genérico basado en tablas), conversión de espacios de color, histogramas, herramientas para, extraer canales individuales, encontrar valores máximos y mínimos de un canal, sumar imágenes, aplicar umbrales, y más.

En OpenCV el concepto de matriz es algo más abstracto del concepto que se utiliza comúnmente en álgebra lineal, ya que sus elementos no son simplemente números, sino que pueden ser usados para almacenar vectores de valores reales o complejos, matrices, imágenes color y en escala de grises, volúmenes voxel, vectores de campo, nubes de puntos, tensores, histogramas. (Auchterberge, 2014)



**Figura 9 Estructura de OpenCV**

Fuente: (Martínez, 2015)

### 1.9. Selección de región de interés

La región de interés es aquella que el usuario pueda modificar a su conveniencia, ya que existen regiones en las cuales pueden tomar datos erróneos para el análisis respectivo de las imágenes provocando problemas al momento de brindar un resultado final.

Para este proyecto se utilizará OpenCV para poder seleccionar nuestra área de interés que será la placa vehicular, para lo cual se realizará el procesamiento de la imagen y la extracción de la información.



**Figura 10 Ejemplo selección de región de interés**

Fuente: (Andrade Miranda Gustavo, 2011)

#### **1.10. Cámara logitech c920**

Para el presente proyecto se utilizará la cámara logitech C920 que cuenta con las siguientes características:



**Figura 11 Cámara Logitech C920**

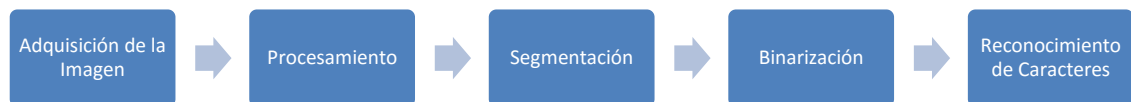
Fuente: (logitech, 2016)

- Videoconferencias Full HD 1080p (hasta 1920 x 1080 píxeles) con la versión más reciente de Skype para Windows
- Videoconferencias HD 720p (1280 x 720 píxeles) con clientes compatibles
- Grabaciones de video Full HD (hasta 1920 x 1080 píxeles)

- Tecnología Logitech Fluid Crystal
- Compresión de video H.264
- Micrófonos estéreo integrados con reducción de ruido automática
- Corrección automática de iluminación escasa
- Clip universal compatible con trípodes para monitores LCD, CRT o laptops
- Captación de hasta 30 fps en Full HD 1080p.

### 1.11. Procesamiento digital de imágenes

El procesamiento digital de imágenes consiste en modificar ciertos aspectos de las imágenes y hacer más evidentes en ellas ciertos detalles que se deseen hacer notar, el análisis consiste en la extracción de ciertas características de las imágenes para lo cual debe pasar por las siguientes etapas: Adquisición de la Imagen, Procesamiento, Segmentación, Binarización y Reconocimiento de caracteres.



**Figura 12 Etapas para el procesamiento de imágenes**

#### 1.11.1. Adquisición de la Imagen

La formación de una imagen digital es el primer paso para cualquier procesamiento de imágenes digitales, ya que consiste básicamente en un sistema óptico en conjunto con un digitalizador, mediante el cual la imagen óptica se transforma en una señal digital es decir esta información es representada en sistema binario (0 y 1), que permitirá el procesamiento.

Se utilizará una cámara web o digital de alta definición para la adquisición de las imágenes frontales del vehículo ya que se encontrará en movimiento. Para poder realizar este tipo de procesamiento la

cámara debe encontrarse con una buena inclinación y la intensidad de la luz no debe perjudicar la fotografía entre otros factores. (Astudillo, 2015)



**Figura 13 Adquisición de la Imagen Placa Vehicular**

Fuente: (Kradac, 2015)

### 1.11.2. Procesamiento

La imagen pasa por una serie de algoritmos para poder mejorar su calidad, identificando el vehículo redimensionando el tamaño de la imagen para que el sistema no se haga lento y pueda ser más veloz al momento de realizar el procesamiento.

### 1.11.3. Segmentación

Para este proyecto es muy importante la segmentación de la imagen ya que se requiere dividir la parte del vehículo con la placa vehicular, este proceso permite encontrar objetos presentes en la imagen que sean útiles para el usuario y pueda clasificarlos entre sí, la segmentación se basa en tres conceptos:

**Similitud:** Los píxeles pueden contener valores similares con respecto alguna propiedad determinada.

**Discontinuidad:** Los objetos dentro de una imagen se diferencian del mismo por que contienen puntos aislados, líneas y aristas.

**Conectividad:** Los píxeles se conectan para formar una región homogénea. (Carla, 2015)



**Figura 14 Segmentación de una Imagen**

Fuente: (Kradac, 2015)

#### **1.11.4. Binarización**

La Binarización no es más que la reducción de información de una imagen basándose en un umbral para poder decidir a qué pixel darle el valor de 0 o 1, mediante esta técnica se pueden separar objetos o regiones que se han de importancia para el usuario. (Cortez, 2014)

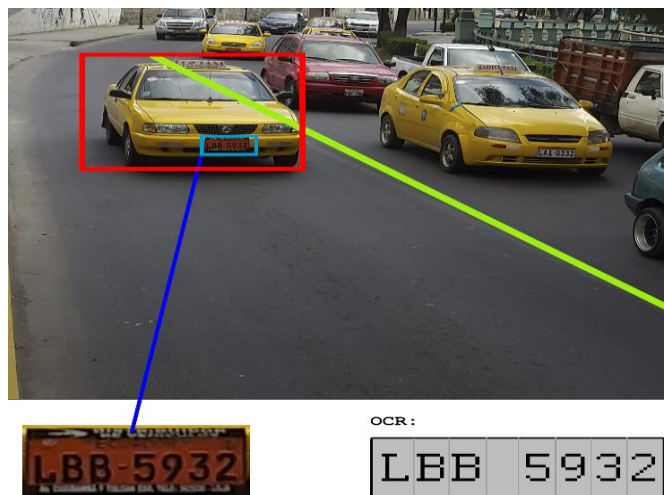


**Figura 15 Binarización de una Imagen**

Fuente: (Kradac, 2015)

#### **1.11.5. Reconocimiento Óptico de Caracteres**

El Reconocimiento Óptico de Caracteres (ROC), es un proceso que tiene como objetivo la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlo en forma de datos, de esta manera se podrá interactuar con los mismos en la siguiente figura 16 podemos apreciar la cualidades específicas en cuanto a rendimiento de los motores OCR más utilizados. (RODRÍGUEZ YAGUAL CRISTHIAN ANTONIO, 2013)



**Figura 16 Reconocimiento de Caracteres**

Fuente: (Kradac, 2015)

### 1.11.6. Tesseract

Tesseract es un motor OCR libre. Fue desarrollado originalmente por Hewlett Packard como software propietario entre 1985 y 1995. Los desarrolladores pueden entrenar Tesseract con sus propias fuentes y asignación de caracteres para obtener eficientes resultados. Tesseract se introdujo en el 1995 y se desarrolla actualmente por Google publicado bajo la Licencia de Apache. (Astudillo, 2015)

Una vez convertida la imagen Binaria y Segmentada se comparan con una base motora (OCR) Reconocedor Óptico de Caracteres que devuelve el resultado en código ASCII. (Astudillo, 2015)

	Ocrad	GOCR	Tesseract
Precisión en el reconocimiento	Buena	Buena	Excelente
Tiempo de Respuesta	Excelente	Mala	Buena
Sistemas operativos de desarrollo	Unix	Windows Linux	Windows Linux Mac OS
Lenguaje de desarrollo	C++	C	C y C++
Licencia	GNU Gneral Public License	GNU Gneral Public License	Licencia Apache
Formato de imagen de entrada	bmp, pgm, ppm	pnm, pbm, pgm y otros	TIFF
Calidad de imagen para análisis	Buena	Excelente	Buena

**Figura 17 Comparación de motores OCR**

Fuente: (Astudillo, 2015)

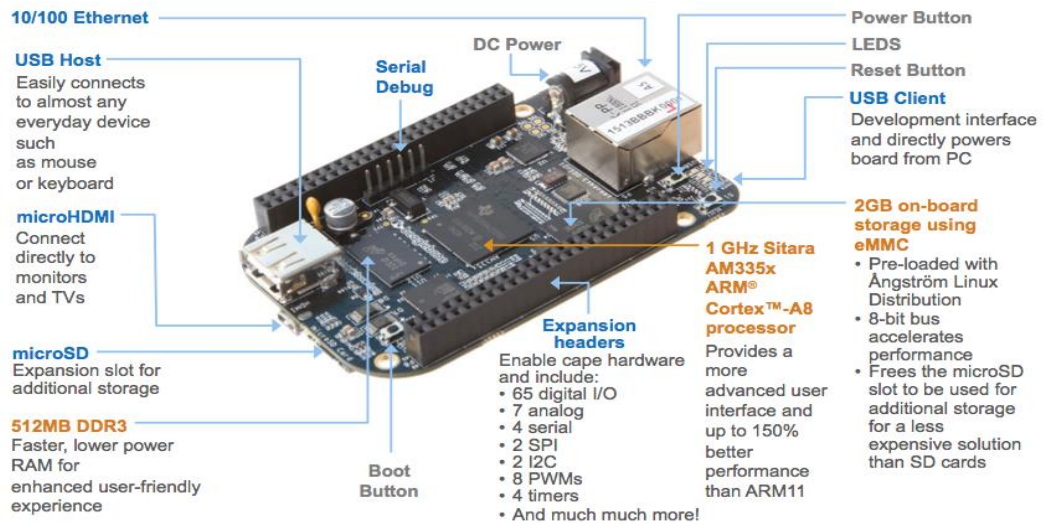


### **1.12. Beaglebone black rev c**

La BeagleBone Black es una tarjeta de desarrollo de bajo costo y del tamaño de una tarjeta de crédito con muy buen soporte de una comunidad de rápido crecimiento. La BeagleBone Black difiere ligeramente de la versión regular, ya que ésta proporciona un puerto micro HDMI, 512 MB de DRAM DDR3L, 4 GB de memoria interna flash, un procesador AM3358 a 1GHz, y haciendo JTAG (Joint Test Action Group), es una norma empleada para pruebas de circuitos impresos empleando escaneo de límites, su nombre estandarizado es IEEE1149.1) opcional con una cabecera proporcionada por el usuario. Además, la BeagleBone Black sigue siendo perfecta para la informática física y pequeñas aplicaciones embebidas, en la figura 18 podemos observar la tarjeta beaglebone black. (Electronic)

BeagleBone Black no sólo es capaz de realizar una interconexión de todos los controladores de motor de robótica, de ubicación o de presión sensores y cámaras en 2D o 3D, sino también correr OpenCV, OpenNI y otros programas de recopilación de imagen y análisis para reconocer los objetos alrededor de un robot y los gestos que puede hacer para controlarla.

A través de las tarjetas de expansión micro HDMI o VGA y LCD, es capaz de decodificar y visualizar múltiples formatos de vídeo que utiliza una pila de software de código completamente abierto y la sincronización de la reproducción a través de Ethernet o USB con otros BeagleBoards para crear paredes de video masivas. (Electronic)



**Figura 18 Tarjeta BEAGLEBONE BLACK**

Fuente: (Normal, 2014)

### 1.12.1. Características Generales

Este tipo de sistemas ya se encuentran en muchos lugares y sirven para medir la temperatura del ambiente, realizar procesamiento de imágenes, para llevar el control de la seguridad de una fábrica, por ejemplo, o incluso como sistemas que se pueden mandar información inalámbricamente. Los usos sólo están limitados por la imaginación de los desarrolladores.

A continuación les presentamos las características generales que nos ofrece la tarjeta Beaglebone Black.

- AM3358 1GHz ARM® Cortex-A8 Processor
- 4GB 8-bit eMMC Onboard Flash
- 3D Graphics Accelerator
- NEON Floating-Point Accelerator
- 2x PRU 32-bit microcontrollers
- USB Client Port
- USB Host Port
- Ethernet
- Micro HDMI
- 2x 46-pin Headers

### 1.12.2. Softwares Compatibles

Beaglebone Black trae incluido el sistema operativo Angstrom, pero si se utiliza el sistema operativo por defecto Angstrom se puede encontrar mayor información de soporte, a continuación indicamos los sistemas operativos que soporta la tarjeta Beaglebone Black.

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ BoneScript library

### 1.13. Python

Existen diferentes lenguajes de programación como por ejemplo C, C++, Java, Visual Basic, Perl, matlab, etc., que pueden emplearse para el diseño y desarrollo de múltiples aplicaciones de control y monitoreo, en base a un código previamente establecido que se encargó de ejecutar una acción específica.

Python es un lenguaje de programación de propósito general, interpretado o de script que es empleado a menudo en roles de scripting. Es un lenguaje de programación orientado a objetos interesante, muy expresivo y con una sintaxis sencilla, aplicable a proyectos de campo y propósito general. Python pertenece a los lenguajes interpretados por tal razón, es muy flexible, potable y de muy alto nivel, lo que lo convierte una de las herramientas más potentes de software libre en la actualidad. (Terán, 2009)



**Figura 19 Logo Python**

Fuente: (Python, s.f.)

### **1.13.1. Antecedentes**

Python tiene su origen hace más de una década creado por Guido Van Roussum, tomando como base otro lenguaje (ABC), Guido se propuso diseñar un software sencillo de propósito general que sirviera para diversas tareas dentro de la programación habitual usando C. El desarrollo de Python duró varios años, hasta que en el 2000 se consolidó una estructura y sintaxis para un producto bastante completo y un equipo de desarrollo y diseño de alto nivel. Actualmente se trabaja en Zope, una plataforma de gestión de contenidos y servidor de aplicaciones para el web, por supuesto, programada por completo en Python. (Alvarez, 2003)

### **1.13.2. Características**

Las principales características citadas por usuarios de Python son las siguientes:

- **Calidad del software**

El código de Python tiene la característica de ser completamente legible y coherente y de calidad, es decir, que se puede reutilizar y conservar mucho más que cualquier otro lenguaje de script tradicional. (Terán, 2009)

- **Propósito general**

Permite el diseño y desarrollo de múltiples programas para diversas aplicaciones del mundo real.

- **Productividad del desarrollador**

Un código creado a través de Python es mucho más potente y requiere de menos espacio de memoria en relación C++ o Java, además que su sintaxis es mucho más interactiva reduciendo la cantidad de código por escribir y depurar.

- **Multiplataforma**

Hay muchas versiones disponibles de Python por tal motivo casi cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

- **Interpretado**

No requiere de un compilador de código para ejecutar la creación de un programa diseñado en su entorno.

- **Soporte de bibliotecas.**

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red.

- **Integración de componentes.**

Los scripts creados en Python presentan un gran poder de integración con diversos dispositivos externos, es decir, que tiene una gran capacidad de adaptación y compatibilidad con otras partes de una aplicación, utilizando distintos mecanismos de integración. (Terán, 2009).

### **1.13.3. Principales Librerías en Python**

A la hora de diseñar un código en Python se dispone de una extensa variedad de algoritmos dispuestos a través de librerías, que facilitan la creación de un programa con múltiples tareas a través de un algoritmo simple, entre las más empleadas tenemos:

- **NumPy** Es un paquete fundamental que consta de funciones y herramientas de arrays que permiten la integración de código entre C++ y Fortran, además NumPy se puede emplear como un contenedor multi-dimensional para datos, lo que facilita la integración rápida y de calidad con una amplia gama de bases de datos, por otro lado, se puede resaltar que NumPy trabaja bajo licencia BSD, por lo que el código puede ser distribuido, modificado y mejorado por el usuario.

- **Scipy** Es una librería específicamente de cálculo numérico estructurada y diseñada para el manejo de todo tipo de cálculo numérico y para implantar datos de todo tipo matemático.
- **Matplotlib** Esta librería tiene la característica de trazado 2D para la generación de figuras de calidad de generalización, posee una amplia gama de formatos impresos y entornos interactivos, que en conjunto con los scripts para Python facilitan en manejo de gráficos, histogramas, barras, etc. (Malaver, 2014)

#### 1.14. Base de datos mysql

MYSQL es un sistema gestor de base de datos (SGBD, DBMS por sus siglas en inglés) muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Es una opción atractiva tanto para aplicaciones comerciales, como de entretenimiento precisamente por su facilidad de uso y tiempo reducido de puesta en marcha. Esto y su libre distribución en internet bajo licencia GPL (Licencia Pública General) le otorgan como beneficios adicionales contar con un alto grado de estabilidad y un rápido desarrollo.



**Figura 20 Logo MYSQL**

Fuente (MYSQL, 2016)

##### 1.14.1. Características de MYSL

El software de bases de datos MYSQL consiste de un sistema cliente/servidor que se compone de un servidor SQL (Lenguaje de Consulta Estructurada) multihilo, a continuación se muestra las características de esta base de datos. (Mora, 2013)

- Esta desarrollado en C/C++.

- Se distribuyen ejecutables para cerca de diecinueve plataformas diferentes.
- La API se encuentra disponible en C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y TCL.
- Está optimizado para equipos de múltiples procesadores.
- Es muy destacable su velocidad de respuesta.
- Se puede utilizar como cliente-servidor o incrustado en aplicaciones.
- Cuenta con un rico conjunto de tipos de datos.
- Soporta múltiples métodos de almacenamiento de las tablas, con prestaciones y rendimiento diferentes para poder optimizar el SGBD a cada caso concreto.
- Su administración se basa en usuarios y privilegios.
- Se tiene constancia de casos en los que maneja cincuenta millones de registros, sesenta mil tablas y cinco millones de columnas.
- Sus opciones de conectividad abarcan TCP/IP, sockets UNIX y sockets NT, además de soportar completamente ODBC.
- Los mensajes de error pueden estar en español y hacer ordenaciones correctas con palabras acentuadas o con la letra 'ñ'.
- Es altamente confiable en cuanto a estabilidad se refiere.

### **1.15. Placas Vehiculares**

Las placas vehiculares son el registro que usan los vehículos automotores para su identificación y circulación legal en todo el territorio nacional. Todas las placas de identificación vehicular deben ser de una lámina metálica rectangular de 30 cm x 15 cm y deben cumplir con las normas de seguridad, recubrimiento y reflectancia determinada por el Consejo Nacional de Tránsito. (TERRESTRE, 2012)



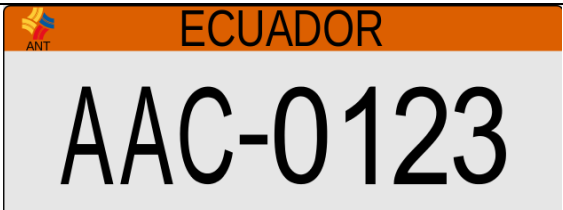
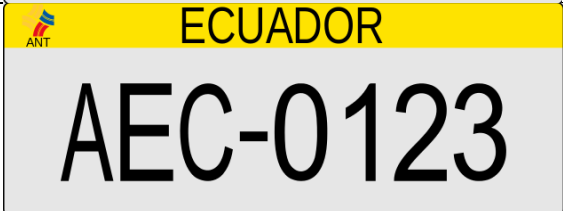
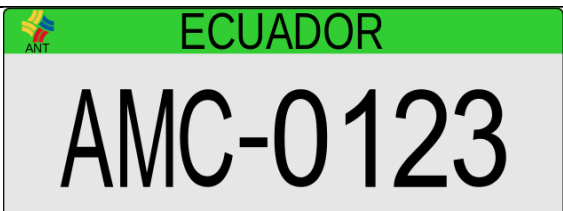
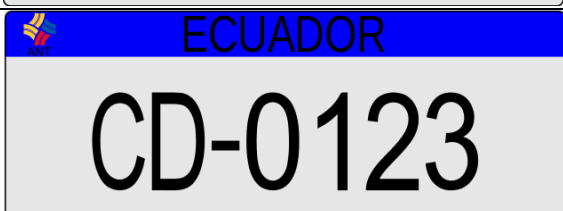
**Figura 21 Placa Vehicular Pertenciente a Ecuador**

Fuente: (Ecuador, 2013)

Las placas de identificación vehicular indicarán el tipo de servicio con una franja de color en la parte superior de 33mm de acuerdo al siguiente cuadro.

**Tabla 2**

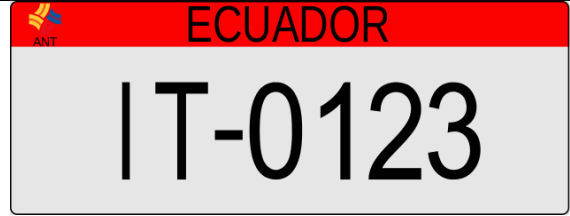
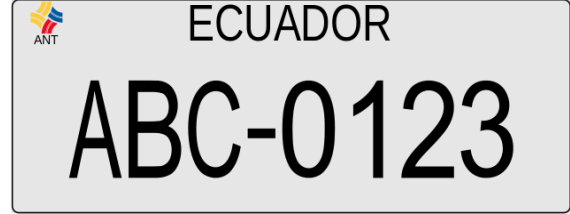
**Tipos de Placas por Servicios**

SERVICIO	COLOR	PLACA
PÚBLICO O COMERCIAL	FRANJA NARANJA	
ORGANISMOS DEL ESTADO	FRANJA ORO	
GAD'S REGIONALES, PROVINCIALES, MUNICIPALES Y PARROQUIALES.	FRANJA VERDE LIMÓN	
DIPLOMÁTICOS Y ORGANISMOS INTERNACIONALES	FRANJA AZUL	

Continua





INTERNACIÓN TEMPORAL	FRANJA ROJA	 The image shows a red license plate with the word "ECUADOR" at the top and "IT-0123" in the center. A small logo with the letters "ANT" is visible in the top left corner of the plate.
PARTICULAR	COLOR BLANCO	 The image shows a white license plate with the word "ECUADOR" at the top and "ABC-0123" in the center. A small logo with the letters "ANT" is visible in the top left corner of the plate.

**Fuente:** (TERRESTRE, 2012)

## CAPÍTULO II

### 2. DESARROLLO

Este capítulo explica de manera detallada el proceso para el reconocimiento de placas vehiculares en movimiento y la creación de una base de datos usando Mysql, partiendo desde la captura de imágenes, el procesamiento, posterior extracción de caracteres por medio del OCR (reconocimiento óptico de caracteres) empleando el algoritmo *tesseract* y la comparación de la placa detectada con nuestra base de datos *Mysql*.

#### 2.1. Instalación del sistema operativo en beaglebone black rev c

Para el proyecto se seleccionó el sistema operativo Debian 7.9, siendo el sistema que más se acopla con mayor eficiencia a las características de la tarjeta en cuanto a rendimiento y compatibilidad entre software y hardware.

A continuación se detallan los pasos necesarios para la instalación del sistema operativo en una tarjeta microSD, se sugiere que sea de una capacidad mayor a 4 Gb y tipo clase 10 (10MB/S).

Primeramente se debe descargar la imagen Debian 7.9 que se encuentra en el link <https://beagleboard.org/latest-images>.

Después de haber descargado el archivo *.img.xz*, se utiliza el programa *7zip* para extraer la imagen como se puede observar en la figura 22.

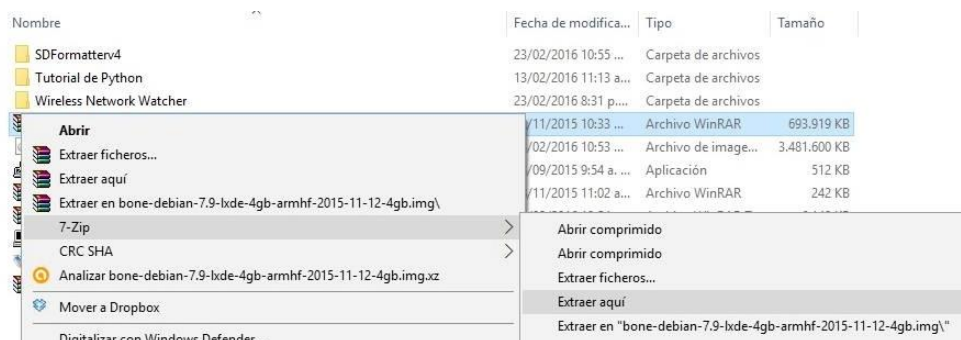
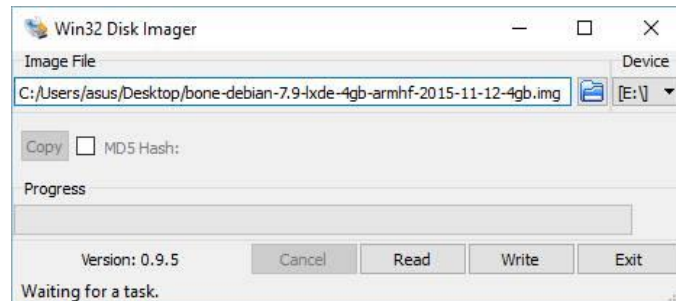


Figura 22 Extracción del Sistema Operativo Debian 7.9 con 7 Zip

Para escribir la imagen en la tarjeta microSD se utiliza el Software **Win32 Disk** el cual cumple con la función de cargar el sistema operativo en dicha tarjeta, como se detalla en la figura 23.



**Figura 23 Instalación del Sistema Operativo en la MicroSD**

Una vez instalado el sistema operativo se procede a insertar la tarjeta microSD en la Beaglebone Black.

Finalmente para encender el equipo se mantiene pulsado el botón **“Inicio de usuario”** de 5 a 7 seg, mientras se conecta la fuente de alimentación a la tarjeta, como se puede apreciar en la figura 24.



**Figura 24 Arranque del Sistema Operativo desde la MicroSD**

### **2.1.1. Expansión de la Memoria en una MicroSD**

Existen dos maneras para realizar una expansión de la imagen Debian en una microSD desde la Beaglebone Black. A continuación se explican los pasos a seguir.

El primer procedimiento es bastante sencillo, en la dirección **“/opt/scripts/tolos”** existe un archivo llamado **“grow\_partition.sh”**

que se encargará de hacerlo de forma automática, cuando termine el proceso se debe reiniciar el dispositivo y estará expandida la partición al tamaño de capacidad de almacenamiento de la memoria microSD.

El segundo procedimiento es un poco más elaborado pero igual de fácil, una vez encendido el sistema se procede a ingresar a la consola LXTerminal como súper usuario (root), e ingresar la instrucción: “**fdisk /dev/mmcblk0**” para cambiar la tabla de particiones, seleccionar la letra **d**, la cual cumple con la función de borrado, y se selecciona la segunda partición con el número **2**.

El siguiente paso es crear la partición para el sistema de archivos, seleccionando la letra **n** para una nueva partición, la letra **p** para partición primaria, el número **2** para especificar que es la segunda partición, la letra **p** para verificar que las particiones se han creado, la letra **w** para escribir la tabla de particiones y por último se realiza un **reboot** que reiniciará el equipo.

Después de haber reiniciado el equipo, se ingresa a la terminal como súper usuario (root) y se escribe “**resize2fs /devmmcblk0p2**” expandiendo la partición 2 a su máxima capacidad de almacenamiento.

### **2.1.2. Instalación de OpenCV 2.4.9**

Antes de realizar la instalación de OpenCV es necesario tener actualizado el sistema operativo Debian 7.9, para lo cual se ingresó las siguientes líneas de código en la consola LXTerminal.

***sudo apt-get update***

***sudo apt-get upgrade***

Entre los requerimientos de instalación de OpenCV se pueden destacar varias librerías como: build-essential, cmake y pkg-config, librerías que se encargan de la generación de meta-paquetes, la compilación e instalación de programas y la configuración de

banderas en Debian. Además en conjunto se instala el compilador GCC y C/C++, mediante las líneas de código:

```
sudo apt-get install build-essential cmake pkg-config  
sudo apt-get install libtiff4 libjpeg-dev-dev-dev libjasper libpng12-  
dev  
sudo apt-get install libavcodec-dev-dev libavformat libswscale-  
dev libv41-dev
```

A continuación se descarga la última versión de OpenCV de GitHub, desde el mismo terminal.

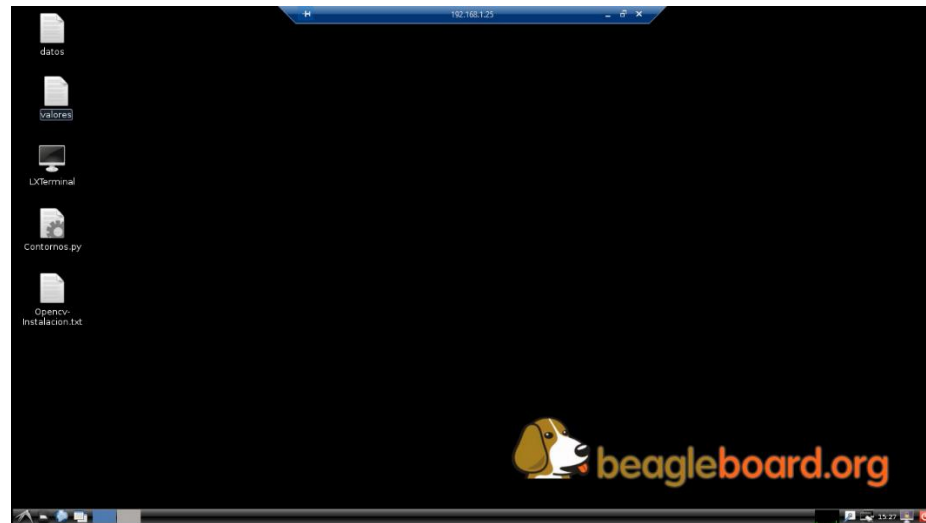
```
git clone https://github.com/Itseez/opencv.git
```

Una vez obtenido los paquetes de instalación, se crea una carpeta mkdir dentro de la dirección de OpenCV la cual contendrá las funciones y algoritmos necesarios para el manejo y procesamiento de imágenes, videos, etc.

```
cd opencv  
mkdir build && cd build  
cmake -D CMAKE_BUILD_TYPE=RELEASE -D  
CMAKE_INSTALL_PREFIX=/usr/local -D WITH_CUDA=OFF -D  
WITH_CUFFT=OFF -D WITH_CUBLAS=OFF -D  
WITH_NVCUVID=OFF -D WITH_OPENCL=OFF -D  
WITH_OPENCLAMDFFT=OFF -D WITH_OPENCLAMDBLAS=OFF -  
D BUILD_opencv_apps=OFF -D BUILD_DOCS=OFF -D  
BUILD_PERF_TESTS=OFF -D BUILD_TESTS=OFF -D  
ENABLE_NEON=on ..  
make  
sudo make install  
sudo ldconfig
```

Finalmente se procede a reiniciar el equipo y OpenCV está listo para trabajar. En la figura 25 se puede visualizar la pantalla de la

tarjeta Beaglebone Black Rev c instalado el sistema operativo Debian 7.9.



**Figura 25 Interfaz Gráfica Sistema Operativo Debian7.9**

## **2.2. Lectura de imágenes con opencv y python**

La lectura de imágenes en el reconocimiento de objetos es la parte fundamental del proyecto, para facilitar este proceso se decidió utilizar las librerías de OpenCV y Python en la tarjeta Beaglebone Black, siendo un instrumento adecuado para los problemas de visión por ordenador. Para poder realizar la lectura de imágenes en OpenCV se debe tomar en cuenta el formato de cada imagen con su respectiva librería ya que no todas son compatibles.

Antes de realizar la lectura y presentación de una imagen es importante no confundir las funciones con los tipos de datos propios de OpenCV. Para lo cual la propia librería utiliza una sintaxis distinta para cada caso, se la declara al inicio de cada programa con **“import (nombre de la librería)”** para poder acceder a sus recursos su sintaxis es la siguiente **cv2.”Nombre de la función”** en el caso de que se requiera utilizar únicamente una de las funciones que abarca una librería, su sintaxis sería la siguiente, **from”nombre de la librería” import”nombre de la función”**.

### 2.2.1. Librería para la Adquisición de Imágenes

Para cargar o leer una imagen desde la tarjeta Beaglebone Black en Python y OpenCV, se necesita de la librería PIL (Python Image Library) o a su vez la librería cv2, que pueden leer diferentes tipos de imágenes, realizar el procesamiento de las mismas o la creación de imágenes desde cero, las librerías se definen de la siguiente manera.

***import PIL***

***import cv2***

En comparación con diferentes lenguajes de programación, en Python no es necesario especificar la ubicación exacta de cada librería ya que esta se guarda por defecto en una localidad determinada por Python.

### 2.2.2. Funciones para Lecturas de Imágenes

Para trabajar con imágenes, OpenCV dispone de diferentes funciones para el procesamiento que permiten modificar sus características según las necesidades del usuario, entre estas se puede destacar Imread, numpy.

#### **Imread**

La función imread lee una imagen desde una cámara o un archivo especificado, su sintaxis es la siguiente.

***cv2.imread(nombre del archivo,bandera)***

Donde el término bandera posee diferentes características que son:

- CV\_LOAD\_IMAGE\_ANYDEPTH – Permite retornar imágenes de 32 bits, 16 bits y 8 bits.
- CV\_LOAD\_IMAGE\_COLOR – Convierte la imagen a un solo color.
- CV\_LOAD\_IMAGE\_GRAYSCALE - Convierte una imagen a escala de grises.
- >0 Devuelve una imagen en color de 3 canales.

- = 0 Devuelve una imagen en escala de grises.
- <0 Devuelve la imagen cargada.

Si la imagen no se puede leer (puede ser debido a la falta de archivos, permisos incorrectos, formato no compatible o no válido), si esto llegara a suceder, la función devuelve una matriz vacía. En la actualidad los formatos compatibles para el manejo de la función `Imread` son: (OpenCV, OpenCV, 2014)

- Mapas de bits de Windows BMP, DIB, JPEG, JPG, JPE, PNG
- Formato de imagen Portable PBM, PGM, PPM
- Raster Sun SR, RAS
- Archivos TIFF, TIF

### **Numpy**

Es el encargado de añadir toda la capacidad matemática y vectorial a Python haciendo posible el manejo de datos numéricos o arrays. Incorpora operaciones básicas como la suma, la multiplicación u otras mucho más complejas como la transformada de Fourier o de álgebra lineal. Además contiene herramientas que permiten incorporar código fuente de otros lenguajes de programación como son C/C++ o Fortran lo que incrementa notablemente su compatibilidad e implementación. (José María Herrera Fernández, 2015)

El desarrollo del módulo Numpy es la creación y modificación de arrays multidimensionales. En Python cada clase puede tener atributos que se pueden llamar simplemente escribiendo a continuación de cada clase seguido de un punto y el atributo, un ejemplo de sus sintaxis se puede ver en la figura 26.



```
>>> np.ones((6,6),np.uint8)
array([[1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1]], dtype=uint8)
```

### Figura 26 Arreglo Matricial Usando Numpy

Ndarray es una matriz de objetos multidimensionales que se utiliza para almacenar grandes conjuntos de datos, permitiendo realizar operaciones matemáticas, los principales atributos son los siguientes:

- **ndarray.ndim** Proporciona el número de dimensiones del array. El array identidad es un array cuadrado con una diagonal principal unitaria.
- **ndarray.shape** Devuelve la dimensión del array, es decir, una tupla de enteros indicando el tamaño del array en cada dimensión. Para una matriz de n filas y m columnas se obtiene (n, m).
- **ndarray.size** Es el número total de elementos del array.
- **ndarray.dtype** Es un objeto que describe el tipo de elementos del array.
- **ndarray.itemsize** Devuelve el tamaño del array en bytes.
- **ndarray.data** El buffer contiene los elementos actuales del array.

Al momento de trabajar con numpy es necesario llamar a la librería “**import numpy as np**”. Para poder desarrollar los ejemplos que se analizarán en los siguientes puntos, es importante ya que al definirlo como “**as np**” se facilita el manejo dentro de la programación.

#### 2.2.3. Presentación de Imágenes

Para presentar una imagen con OpenCV, se lo realiza por medio de la función “**imshow**”, y se define de la siguiente manera:

```
imshow("Etiqueta", variable);
```

En el primer argumento, “etiqueta” se define el nombre de la ventana en la que se va a ejecutar. El segundo argumento “variable” es donde se encuentra la imagen que se va a visualizar. Se pueden crear tantas ventanas como desee, pero con diferentes nombres.

Para poder visualizar una imagen en la ventana que fue definida o generada de manera automática se necesita de la función “**waitKey**”. Esta función lo que hace es esperar por un evento generado por teclado, en caso de poner cero a este parámetro el programa se ejecutará hasta esa línea y no saldrá del ciclo, se define de la siguiente manera. (OpenCV, OpenCV, 2014)

### ***cv2.waitKey (0)***

Nota: Cuando se trabaje con varias ventanas al mismo tiempo es aconsejable utilizar la función `cv2.destroyAllWindows ()`, dicha función cierra todas las ventanas una vez terminado el programa.

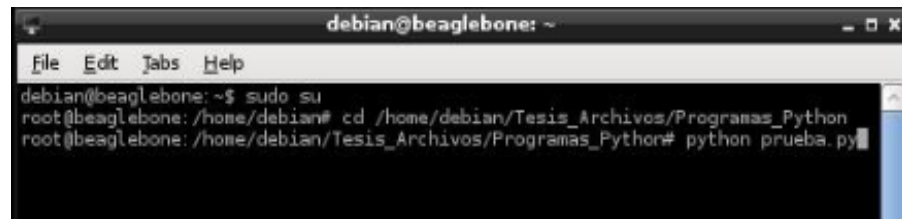
### **2.2.4. Ejemplos**

A continuación se muestran ejemplos básicos con imágenes realizados con OpenCV y Python.

```
import numpy as np  
import cv2  
img=cv2.imread("/home/debian/Pictures/mitsubishi.jpg",1) #Se abre el archivo en la dirección indicada, el número uno indica que las propiedades de la imagen no serán modificadas, se abrirá tal como está  
cv2.imshow("Mitsubishi",img) #Muestra la imagen en la ventana  
cv2.waitKey(0) # Espera que se presione una tecla para salir del ciclo  
cv2.destroyAllWindows () #Destruye todas las ventanas abiertas
```

Para poder ejecutar el programa, en LXTerminal se ingresa como súper usuario (`sudo su`) y se ubica la dirección donde se encuentra el

archivo, anteponer el nombre python seguido del nombre y la extensión del archivo como se puede ver en la figura 27.



**Figura 27 Cómo Ejecutar un Archivo de Python**

En la figura 28 se puede visualizar que se ha cargado una imagen en formato RGB.



**Figura 28 Imagen Cargada en Formato RGB desde Python**

Para poder visualizar la imagen en escala de grises basta con colocar dentro de la función cv2.imread la bandera en 0.

```
import numpy as np  
import cv2  
img=cv2.imread("/home/debian/Pictures/mitsubishi.jpg",0) #Se  
abre el archivo en la dirección indicada, el número cero indica  
que las propiedades de la imagen serán modificadas, se abrirá en  
escala de grises  
cv2.imshow("Mitsubishi",img) #Muestra la imagen en la ventana  
cv2.waitKey(0) # Espera que se presione una tecla  
cv2.destroyAllWindows () #Destruye todas las ventanas abiertas
```

En la figura 29 se puede visualizar que se ha cargado una imagen en formato escala de grises.



**Figura 29 Imagen Cargada en Formato Escala de Grises desde Python**

Los programas están basados en los ejemplos de la documentación de OpenCV2.4.9.0 (documentation O. 2., 2014)

### **2.3. Adquisición de imágenes en tiempo real con opencv y python**

Para realizar la adquisición de imágenes mediante videos, capturas en tiempo real se emplea una cámara digital y una tarjeta Beaglebone Black, al momento de realizar la captura, lo que se hace es tener una señal analógica por medio de la cámara, lo que la hace digital es el software de la misma, teniendo así un arreglo de matrices con información de cada imagen por pixeles para poder proceder con el procesamiento requerido por el usuario.

El procedimiento es similar al de cargar una imagen directamente desde el computador o la tarjeta con la diferencia que ahora se va a utilizar una cámara de video para lo cual en las siguientes secciones se explicará cómo inicializar la cámara y la lectura de la misma.

#### **2.3.1. Inicialización de la Cámara**

Para inicializar la cámara y leer las capturas de la misma, se lo realiza por medio de la función “**cv2.VideoCapture(variable)**”, la cual permite trabajar con objetos de tipo Numpy para almacenamiento y presentación de las capturas, para trabajar con una cámara web en tiempo real es necesario reemplazar el argumento “variable” por un número cero correspondiente al puerto de acceso de la cámara de

video. En el caso de que se requiera trabajar con videos existentes, se debe especificar la dirección del mismo seguido del nombre con su extensión, los formatos de video admitidos son: AVI, MP4, WMV. Para iniciar y capturar desde una cámara se lo hace de la siguiente manera.

### ***cv2.VideoCapture(0)***

Es muy importante fijar algunos parámetros como la altura, ancho, formato de pixeles, para poder trabajar en adquisición de imágenes según el requerimiento del usuario y así ahorrar memoria de la tarjeta. (documentation O. , 2014)

#### **2.3.2. Lectura de la Cámara**

Para poder obtener la lectura de un video utilizando VideoCapture se requiere el uso de la siguiente función.

### ***captura.read()***

Esta función devuelve un booleano (Verdadero\Falso) cuando la captura es leída correctamente, el valor retornado es Verdadero y almacena una captura del video en una variable asignada, si no encuentra ninguna captura retorna Falso, esto quiere decir que la cámara ha sido desconectada o no hay más capturas en el archivo de vídeo. (documentation O. , 2014)

#### **2.4. Captura de imágenes con opencv y python**

Las siguientes instrucciones presentan la captura de imágenes en tiempo real por medio de una cámara web.

```
import cv2 #Se importa las librerías requeridas  
import numpy as np  
captura = cv2.VideoCapture(0) #Se Inicializa la cámara  
while(1):  
_, imag = captura.read()#Captura una imagen y la guarda en la  
variable imagen
```

```
cv2.imshow('VideoOriginal', imag) #Muestra la imagen en la ventana  
tecla = cv2.waitKey(5) & 0xFF #Espera que se presione una tecla  
para salir del bucle  
if tecla == 27:  
break  
cv2.destroyAllWindows() #Destruye las ventanas abiertas
```

En la figura 30 se puede visualizar una imagen capturada en tiempo real.



**Figura 30 Captura de Imagen por Medio de la Cámara Logitech C920 con Python**

Para poder obtener capturas en escala de grises, se necesita de la función `CvtColor` que permite la conversión a formato escala de grises, en el siguiente ejemplo se capturan imágenes en tiempo real y posteriormente se convierten a escala de grises.

```
import cv2 #Importa las librerías necesarias  
import numpy as np  
captura = cv2.VideoCapture(0) #Inicializa la cámara  
while(1):  
#Captura una imagen y la guarda en la variable imagen  
_, imagen = captura.read()  
#Convierte la imagen en escala de grises  
grises=cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
```

```
cv2.imshow('VideoOriginal', grises) #Muestra la imagen en la  
ventana  
#Espera que se presione una tecla para salir del bucle  
tecla = cv2.waitKey(5) & 0xFF  
if tecla == 27:  
break  
cv2.destroyAllWindows() #Destruye las ventanas abiertas
```

En la figura 31 se puede visualizar la imagen capturada convertida en escala de grises en tiempo real.



**Figura 31 Captura de Imagen por Medio de la Cámara Logitech C920 con Python en Formato Escala de Grises**

Los programas están basados en los ejemplos de la documentación de OpenCV2.4.9.0 (documentation O. , 2014)

## **2.5. Desarrollo del algoritmo para el reconocimiento de placas vehiculares**

En este apartado se explicará el proceso del algoritmo para el reconocimiento de placas vehiculares por medio del diagrama de flujo que se muestra en la figura 32, donde se presenta el funcionamiento del programa desarrollado en Python con OpenCV en una tarjeta Beaglebone Black. El algoritmo se divide en cuatro partes fundamentales, ubicación de la placa en una región de interés, extracción y corrección del ángulo de la placa, reconocimiento óptico de caracteres (OCR) con Tesseract, y finalmente la búsqueda de la placa dentro de una base de datos.

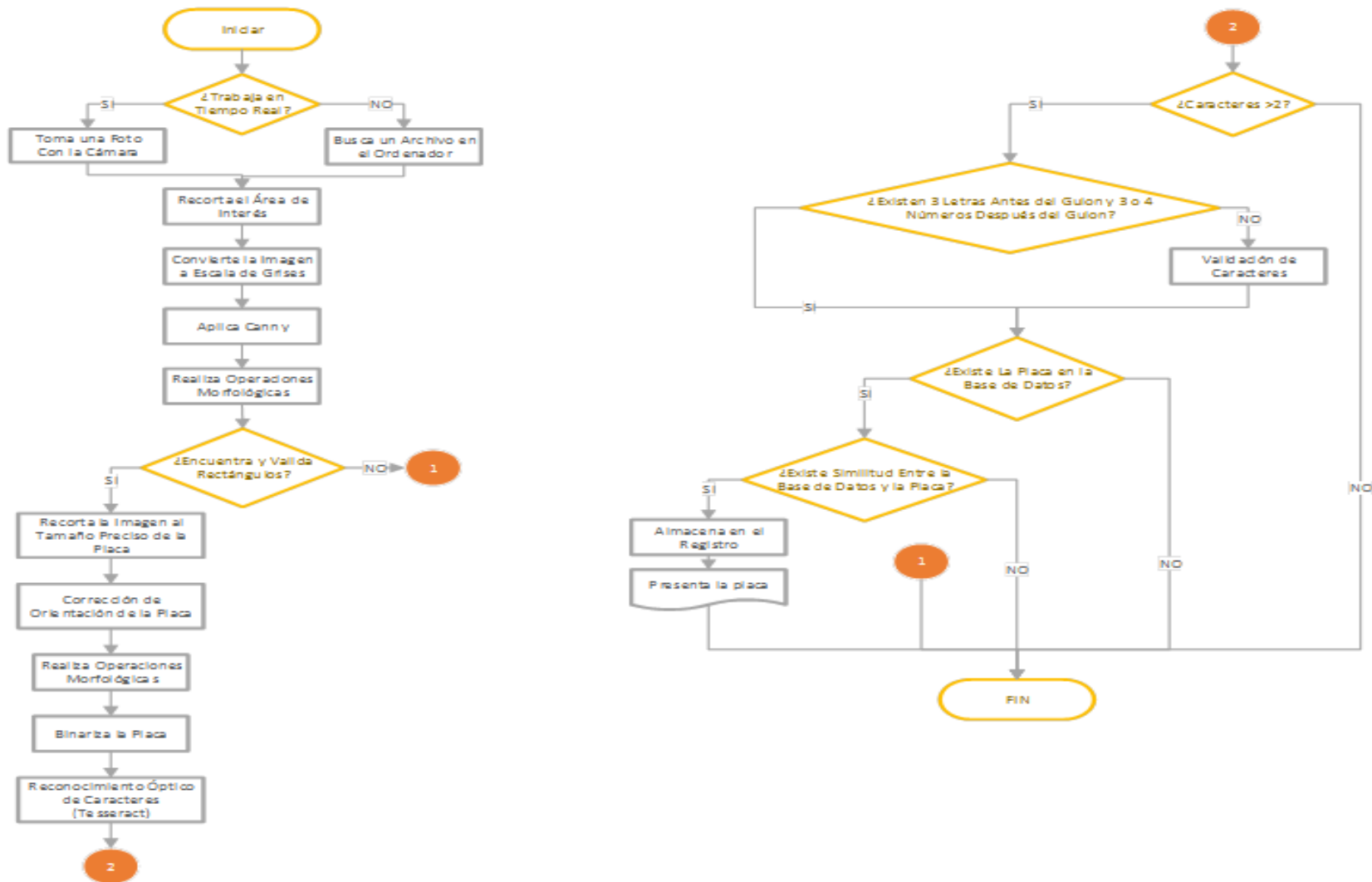


Figura 32 Diagrama de Flujo del Funcionamiento General del Proyecto



### 2.5.1. Ubicación de la Placa en una Región de Interés

El programa inicia dando a escoger al usuario si desea trabajar con videos o una cámara en tiempo real, para este caso se ha escogido la cámara Logitech C920 y se deberá modificar dentro de las líneas de código la función `captura=cv2.VideoCapture(0)` que especifica que se trabaja con la cámara, en el caso de que se desee trabajar con videos dentro de la misma función se debe especificar la dirección de la misma así:

**`captura=cv2.VideoCapture('/home/debian/Tesis_Archivos/Videos/Logitech Webcam/Video 36.wmv')`,**

Para poder leer las imágenes capturadas se utilizó la función `captura.read()` la cual permite almacenar dichas imágenes en la variable `imglectura`.

**`_, imglectura = captura.read()`**

El siguiente paso por el diagrama de flujo que se puede visualizar en la figura 32, es el recorte de la sección de interés, para lo cual se cuenta con una imagen inicial de 480p alto x 864p ancho que para el procesamiento de la tarjeta es demasiado alto. Se procede a recortar la imagen a una dimensión de 107p alto x 432p de ancho que será la región de interés para el desarrollo de este proyecto, ayudando así a disminuir el trabajo de procesamiento que debe realizar la tarjeta, como se puede apreciar en la figura 33.



**Figura 33 Recorte de la Región de Interés**

A continuación se procede a convertir la imagen recortada a Escala de Grises, es decir, una imagen de 8 bits y poder facilitar el manejo de la imagen en una sola matriz, no se ha realizado la conversión a escala de grises de la imagen capturada porque el número de píxeles a convertir es mayor, ahorrando 0.01 segundos en el procesamiento de la tarjeta. Una vez realizada la conversión a Escala de Grises, se procede a utilizar la función Canny que ayuda a buscar bordes dentro de una imagen, la cual será de utilidad para localizar la placa vehicular dentro de la región de interés como será explicado en la siguiente sección. Para definir los valores del Canny se han realizado pruebas en diferentes ambientes calculando un promedio aproximado entre valores que resalten los bordes de la placa, a continuación su función.

***edges = cv2.Canny(imgGRIS,400,102)***

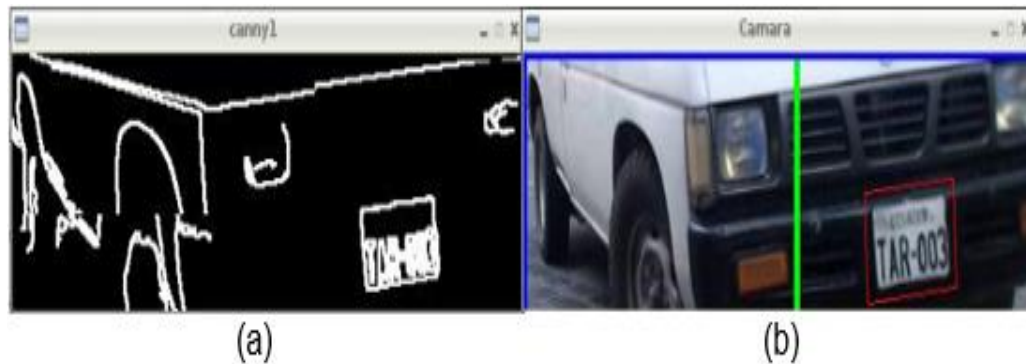
En la figura 34(a) se puede visualizar la imagen en formato escala de grises, en la figura 34 (b) se puede apreciar la imagen aplicado el algoritmo Canny.



**Figura 34 Imagen convertida a Escala de Grises (a) y Canny (b)**

Posteriormente se procede a realizar una operación morfológica (dilatación) que ayudará a dar mayor detalle a los bordes de la placa vehicular, que se puede observar en la figura 35(a), para poder encontrar un rectángulo dentro de la imagen se procede a realizar una búsqueda de contornos donde la función “cv2.approxPolyDP” busca figuras dentro de la región de interés que contengan 4 lados como pueden ser cuadrados trapecios rectángulos, etc. Adicionalmente preguntará si el ancho es mayor que el alto y su área se encuentra

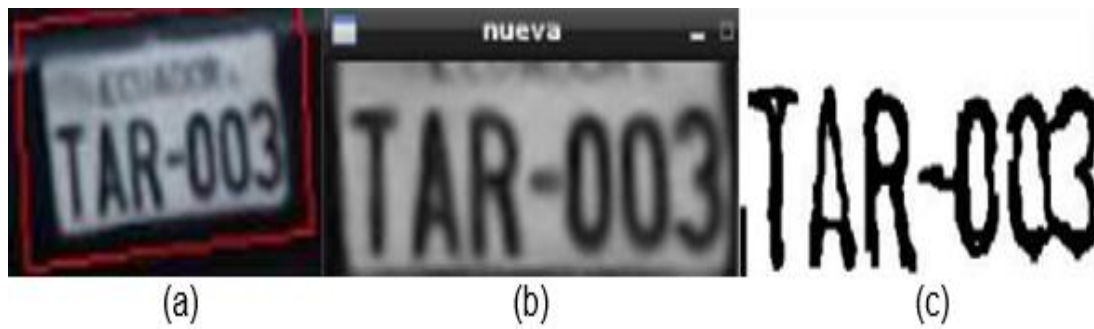
entre 800p y 4000p, solo si se cumple estas 3 condiciones esenciales se puede decir que es una placa vehicular en la cual se dibuja un rectángulo, como se puede apreciar en la figura 35 (b).



**Figura 35 Imagen dilatada (a) Placa Detectada (b)**

### 2.5.2. Extracción y Corrección del Ángulo de la Placa

La placa vehicular pasa por varios procesos antes de ser extraída, para lograr este objetivo se procede a desarrollar un algoritmo que recorta la sección previamente dibujada del rectángulo anterior, obteniendo así la imagen completa de la placa vehicular como se puede visualizar en la figura 36(a), para lo cual se corrige su ángulo con el método de la pendiente y su ángulo de inclinación, dando como resultado una placa totalmente corregida que se puede procesar de mejor manera y permita obtener los caracteres de dicha imagen como se muestra en la figura 36 (b). Se realiza una ampliación del recorte de la nueva imagen de la placa vehicular además se aplica el algoritmo Canny ejecutando un cálculo de pixel promedio aplicando diferentes valores a la imagen para poder emplear operaciones morfológicas obteniendo una imagen binarizada que pueda ser utilizada con el motor de búsqueda OCR (Reconocimiento Óptico de Caracteres) que se explicará en la siguiente sección. En la figura 36(c) se aprecian lo mencionado con anterioridad.



**Figura 36 (a) Recorte de la Placa Vehicular, (b) Corrección de Ángulo de la Imagen, (c) Aplicación de Operaciones Morfológicas y Binarización de la Fotografía.**

### **2.5.3. Algoritmo de Reconocimiento Óptico de Caracteres Tesseract**

Finalizado el procesamiento de detección y normalización de la imagen de la placa vehicular, se procede a obtener los caracteres correspondientes a la información o código particular de cada vehículo mediante un algoritmo, para lo cual se emplean motores de búsqueda de reconocimiento óptico de caracteres, de igual manera se desarrollan funciones de validación alfanumérica para determinar la validez de las diferentes letras y números de las placas vehiculares analizadas.

El motor de búsqueda Tesseract OCR permite el análisis de diversos tipos de formatos de documento de imágenes, examinando los espacios en blanco, descomponiendo el texto en palabras y caracteres, de forma que el sistema puede formular diferentes suposiciones o proximidades y compararlas con los diccionarios de idiomas predefinidos contenidos por el mismo, para crear y aproximar de forma efectiva las letras y números.

Tesseract tiene un diseño básico de funcionamiento, el cual se lo puede dividir en cuatro etapas esenciales para el proceso de adquisición de los caracteres de interés, culminando el proceso de

detección con la validación de la información obtenida. A continuación se describen las etapas del motor de búsqueda Tesseract:

#### **2.5.3.1. Binarización**

Inicialmente el algoritmo carga una imagen a color o una en escala de grises al sistema, para posteriormente efectuar una conversión (binarización) de los atributos de la imagen, obteniendo como resultado una nueva imagen con solo dos colores blanco y negro, en la cual se puedan resaltar con mayor fuerza los cambios de intensidad de un color a otro en la imagen, formando relieves que facilitan la detección de los bordes y símbolos que presentan una determinada imagen.

#### **2.5.3.2. Segmentación de la Imagen**

Segmentar la imagen es el procedimiento más complejo dentro del funcionamiento de OCR Tesseract, aquí el algoritmo realiza una división de la imagen en varias partes, es decir, separa diferentes grupos de píxeles unos de otros, obteniendo solo las áreas de utilidad para el procesamiento, respaldándose de dos propiedades basadas en torno al valor de los niveles de gris de la imagen. En primera lugar se analiza la discontinuidad que permite notar cambios bruscos en los niveles de escala de grises, la segunda propiedad es la similitud que enfatiza la detección de las zonas que presentan similitud entre píxeles, es decir, relaciona los grupos de píxeles con características parecidas.

#### **2.5.3.3. Adelgazamiento de las Componentes**

Realizada la segmentación de las áreas de interés de la imagen, se ejecutan diferentes procesos morfológicos para obtener una imagen más clara, nítida y manejable para el reconocimiento de los caracteres. Principalmente se utiliza la erosión para disminuir y suavizar levemente los bordes o cambios de intensidad entre los colores blancos y negros intentando

siempre mantener sus rasgos característicos para facilitar su representación.

#### **2.5.3.4. Comparación con Patrones**

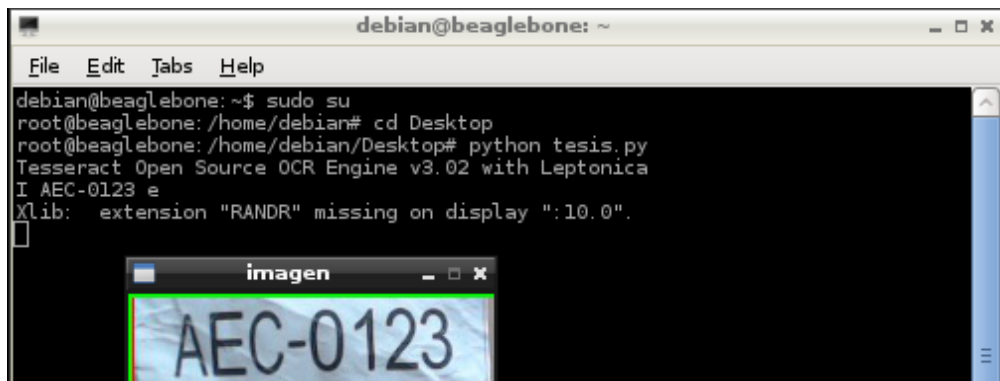
Finalmente, obtenidas las áreas de interés, Tesseract mediante el uso de plantillas predefinidas, es decir, caracteres precargados contenidos en el motor de búsqueda, compara el carácter obtenido en los procesos anteriores, con uno patrón establecido que será la guía para determinar el valor máximo de similitud o coincidencia entre las imágenes analizadas, dando como resultado un valor string en cual representará el carácter más aproximado que detecte.

#### **2.5.4. Manejo de OCR Tesseract a través de Python**

Para emplear el motor de búsqueda OCR mediante el intérprete de python se importa la librería *"from pytesseract import \*"*, la cual dará acceso a las funciones y comandos para el manejo de caracteres. A continuación, para adquirir el dato string correspondiente, a la información de la imagen analizada se procede a implementar el código *"image\_to\_string"* con la variable que contenga la imagen cargada previamente como se presenta a continuación:

```
imgTesse=Image.open('/home/debian/Tesis_Archivos/PruebasTesseract/17052016/Tesseract/Tessefoto0'+str(a)+'.tif')  
placa = image_to_string(imgTesse).strip()
```

El resultado del OCR Tesseract es un dato string conformado por los símbolos y caracteres obtenidos del proceso de reconocimiento óptico de caracteres anteriormente expuesto. Como se aprecia en la figura 37, el string obtenido es el dato *"I AEC-0123 e"* de la imagen de placa vehicular *"AEC-0123"*, mostrando valores adicionales erróneos que posteriormente serán eliminados mediante el algoritmo de validación de caracteres.



**Figura 37 Placa Procesada por OCR Tesseract**

## 2.6. Métodos de validación de caracteres

El código generado por Tesseract OCR, en variadas ocasiones tiende a mostrar ciertas variaciones o semejanzas entre símbolos y caracteres alfanuméricos dependiendo de la calidad de la imagen examinada, para validar los caracteres de tipo String, el manejo de la información se lo realiza a nivel de vectores mediante el comando "list " que es un tipo de dato muy versátil que permite escribir la información como una lista de valores separados por comas entre corchetes. Una vez creada la lista que contiene los caracteres de la posible placa vehicular, se procede a validar la información por medio de tres métodos de mando de datos String disponibles en Python como son: isalnum(), isalpha() y isdigit()).

### 2.6.1. El Método isalnum()

La función de isalnum() es empleada específicamente para verificar si una cadena se compone solamente de caracteres alfanuméricos por ejemplo:

```
str = "beagle2016";
print str.isalnum()
>>> True

str = "Ejemplo beagle..!!!";
print str.isalnum()
>>> False
```

La variable `str` contiene el string “beagle2016” y al ser todos sus caracteres alfanuméricos este devuelve una condición de verdad “True”, mientras en el segundo caso el string “Ejemplo beagle..!!!”, la condición de retorno será falsa “False”, puesto que el string posee símbolos además de caracteres alfanuméricos.

### 2.6.2. El Método `isalpha()`

Este método comprueba si la cadena se encuentra conformada de sólo caracteres alfabéticos ejemplo.

```
str = "beagle";  
print str.isalpha()  
>>> True
```

```
str = "Ejemplo beagle2016.....!!!";  
print str.isalpha()  
>>> False
```

La variable `str` contiene el string “beagle” y al ser todos sus caracteres alfanuméricos, este devuelve una condición de verdad “True”, mientras en el segundo caso el string “Ejemplo beagle2016..!!!”, la condición de retorno será falsa “False”, puesto que el string posee símbolos y caracteres numéricos.

### 2.6.3. El Método `isdigit()`

Este método se encarga de validar que solamente dígitos conformen una determinada cadena ejemplo.

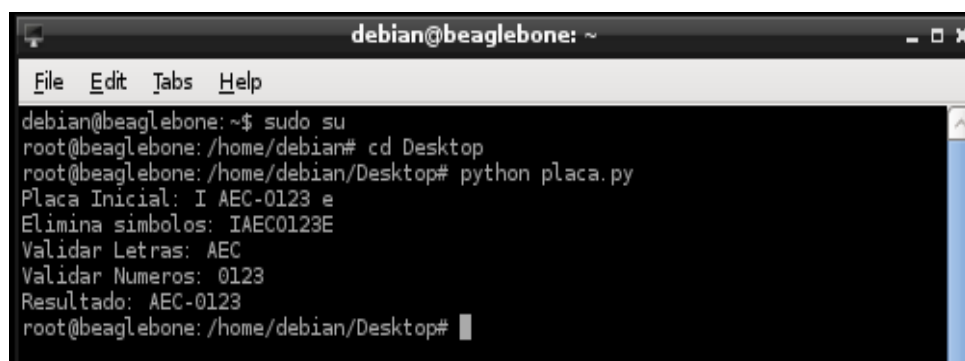
```
str = "123456";  
print str.isdigit()  
>>> True
```

```
str = "Ejemplo beagle2016..!!";  
print str.isdigit()  
>>> False
```



La variable `str` contiene el string "123456" y al ser todos sus caracteres numéricos este devuelve una condición de verdad "True", mientras en el segundo caso el string "Ejemplo beagle2016..!!!", la condición de retorno será falsa "False", puesto que el string posee símbolos y caracteres alfabéticos.

En la figura 37 se aprecia el dato string obtenido anteriormente correspondiente a la placa analizada, la cual presenta varias anomalías incluidas, el proceso de validación se puede visualizar detalladamente en la figura 38 donde la función "placa inicial" hace referencia a los datos obtenidos por el motor de búsqueda Tesseract. A continuación la función "elimina símbolos", realiza la acción de buscar símbolos diferentes a caracteres alfanuméricos eliminándolos y dejando un dato string con valores alfanuméricos, la tercera fase "Validar Letras" toma los tres primeros caracteres del string y verifica que todas sean letras y de existir caracteres numéricos los reemplaza con su dato alfabético más parecido, de manera similar "Validar Números" verifica que los últimos tres o cuatro caracteres del string sean números, la etapa final del proceso de validación muestra la función "Resultado" que se encarga de unir tanto el dato de la función validar letras, como el dato de validar números en nuevo valor string, que contendrá la placa vehicular detectada por el algoritmo.



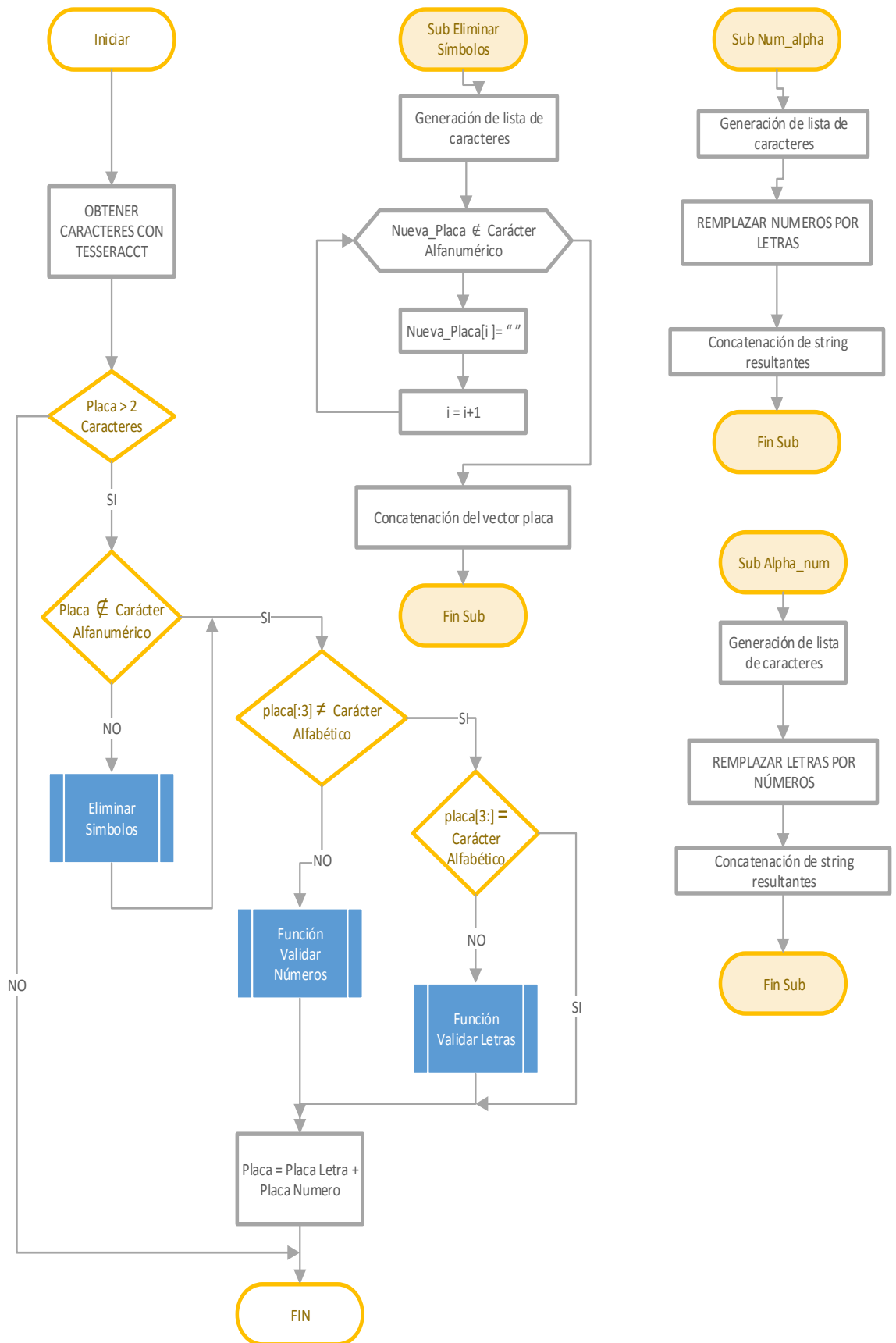
```
debian@beaglebone: ~  
File Edit Tabs Help  
debian@beaglebone:~$ sudo su  
root@beaglebone:/home/debian# cd Desktop  
root@beaglebone:/home/debian/Desktop# python placa.py  
Placa Inicial: I AEC-0123 e  
Elimina simbolos: IAEC0123E  
Validar Letras: AEC  
Validar Numeros: 0123  
Resultado: AEC-0123  
root@beaglebone:/home/debian/Desktop#
```

**Figura 38 Resultado de Métodos de Validación de Caracteres**

Cada método de validación expuesto con anterioridad se maneja mediante funciones con parámetros para él envío de información del

programa principal a cada función, retornando la información validada al final de cada proceso. Para concluir con la validación de la información, los caracteres retornados tanto de la función “alpa\_num” (caracteres alfabéticos) como de la “num\_alpa” (caracteres numéricos) son concatenados, es decir, se unen para formar un solo string.

El procedimiento general de adquisición, corrección y validación de los caracteres obtenidos del OCR tesseract se presenta de manera gráfica mediante un flujograma general de funcionamiento en la figura 39.



**Figura 39 Diagrama de Flujo del Funcionamiento de la Validación de Caracteres**

## **2.7. Base de datos python-mysql**

SQL es un lenguaje estructurado diseñado para realizar consultas, destinado para el acceso a bases de datos relacionales, al ser un lenguaje de programación ostenta ciertos estándares que cambian dependiendo del tipo de base de dato a emplearse. Para poder vincular Python con el servidor MySQL, es necesaria la instalación de librerías que permitan actuar de intermediario entre ambos programas, para el sistema operativo Debian el fichero de instalación se lo puede realizar con el comando.

**sudo apt-get install python-mysql.connector**

Una vez Instalada la librería, la base de datos se encuentra lista para ser programada a través del Intérprete de Python mediante código, para ello se importa la librería.

**import MySQLdb**

Importando las funciones necesarias para la configuración y manejo de una base de datos previamente creada.

### **2.7.1. Crear Base de Datos**

El diseño de la base de datos MySQL se lo efectúa a través de la consola (LXTerminal) en Debian ingresando por medio del cliente mysql, empleando el comando de acceso *“mysql -u root -p”* que se representa en la figura 40, donde “mysql” hace referencia a la base de datos, la variable “-u” hace alusión al nombre de usuario en este caso root para referirse a súper usuario, finalmente la sigla “-p” permite resaltar el ingreso de la contraseña de acceso a la base de datos cada vez que se requiera ingresar a la misma.

```
root@beaglebone:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 39
Server version: 5.5.49-0+deb7u1 (Debian)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

**Figura 40 Acceso a Modo Usuario Mysql**

Para crear la base de datos se emplea el comando “*create database mybase;*” donde el término *mybase*” tomará el nombre de la base de datos a crear, la base del proyecto se denominará *matriculas*, seguidamente se verifica que la base de datos se ha creado exitosamente y no presente ningún tipo de error, escribiendo en la consola el comando “*show databases;*” tal como se observa en la figura 41.

```
COM7 - PuTTY
mysql> create database matricula
-> ;
Query OK, 1 row affected (0.00 sec)

mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| MATRICULAS |
| matricula |
| matriculas |
| mysql |
| performance_schema |
+-----+
6 rows in set (0.00 sec)

mysql>
```

**Figura 41 Creación y Listado de Bases de Datos Mysql**

Una vez creada la base de datos se selecciona el nombre de base en la cual se va a trabajar, dentro de una lista de nombres de bases de datos disponibles que han sido desplegadas anteriormente, para el proyecto se selecciona la base *matriculas* como se presenta en la figura 42.



```
COM7 - PuTTY
-> ;
Query OK, 1 row affected (0.00 sec)

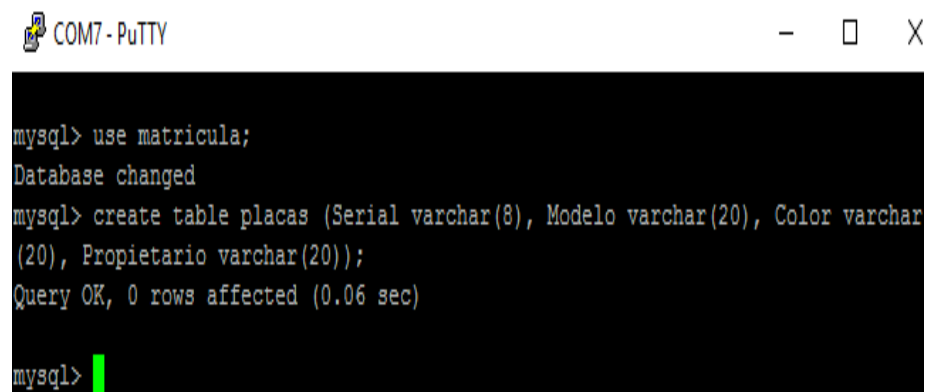
mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| MATRICULAS |
| matricula |
| matriculas |
| mysql |
| performance_schema |
+-----+
6 rows in set (0.00 sec)

mysql> use matricula;
Database changed
mysql>
```

**Figura 42 Selección de Base de Datos Mysql**

### 2.7.2. Ingreso de Información en Base Mysql

Creada la base de datos se procede a insertar los atributos que contendrán la información referente a la placa vehicular por medio de tablas, el presente proyecto requiere de una tabla llamada “placas” compuesta por 4 columnas correspondientes a los tipos de datos que serán admitidos dentro de la base como son “serial varchar(8)”, “modelo varchar(20)”, “color varchar(20)” y “propietario varchar(20)” datos particulares de cada vehículo detectado que será ingresado como se presenta en la figura 43.



```
COM7 - PuTTY
mysql> use matricula;
Database changed
mysql> create table placas (Serial varchar(8), Modelo varchar(20), Color varchar(20), Propietario varchar(20));
Query OK, 0 rows affected (0.06 sec)

mysql>
```

**Figura 43 Creación de Tablas y Etiquetas de Base de Datos**

### 2.7.3. Desplegar Base de Datos

Finalizado el proceso de creación se emplea el comando “*show tables;*”, para exhibir las tablas disponibles en la base de datos y de requerir la visualización del contenido de cada tabla se emplea el

código “*describe placas;*” que genera la tabla creada y dentro de esta las 4 columnas insertadas con sus respectivas etiquetas, figura 44.

```
mysql> show tables;
+-----+
| Tables_in_matricula |
+-----+
| placas |
+-----+
1 row in set (0.01 sec)

mysql> describe placas;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Serial     | varchar(8) | YES  |     | NULL    |      |
| Modelo     | varchar(20) | YES  |     | NULL    |      |
| Color      | varchar(20) | YES  |     | NULL    |      |
| Propietario | varchar(20) | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

**Figura 44** Despliegue de Tablas Creadas en la Base de Datos

## 2.8. Manejo de datos en mysql

El manejo de la información contenida en la base de datos de MySQL se lo realiza a través de código, a razón de que Python no cuenta con una interfaz para relacionarse interactivamente con MySQL, por lo que la información solamente puede ser manipulada mediante código.

### 2.8.1. Acceso a Base de Datos

El acceso de la base de datos de *mysql*, se lo puede realizar de dos maneras, la primera mediante la consola (LXTerminal) siendo esta la más cómoda para el acceso especificando el nombre de usuario y contraseña para ingresar. La segunda forma es a través del intérprete de Python estableciendo varios parámetros de acceso, concurriendo en un proceso más elaborado para ingresar a la base. Cabe mencionar que ambas técnicas de acceso se realizan mediante código, puesto que Python no posee un módulo para interactuar directamente con *mysql*.

El acceso a la base de datos del proyecto se lo desarrolló con el segundo método, es decir, empleando el intérprete de Python. Una vez inicializado el intérprete se importa la librería “*MySQLdb*” que es un módulo necesario para establecer la conexión con bases *mysql*,

posteriormente se realiza ingreso del código de acceso como se aprecia en la figura 45 y se verifica que la conexión de la base sea la correcta.

```
bd = MySQLdb.connect("localhost","root","12345","matriculas" )
cursor = bd.cursor()
sql = "SELECT * FROM placas"
try:

    cursor.execute(sql)

    resultados = cursor.fetchall()
except:
    print "Error: No se pudo obtener los Datos"

bd.close()
```

#### **Figura 45 Acceso a Base de Datos a través de Python**

Como se detalla en la figura 45, se especifican varias parámetros para el acceso, el primero es generalmente el “*localhost*” que no es más que el host necesario para conectarse a una base de datos, ya que se encarga de proveer datos como *ip*, *url* o un puerto de acceso, el segundo parámetro es el de “*usuario*” donde se especifica el nombre o alias del propietario de la base creada, el tercer parámetro contenido es de “*password*” donde se define la contraseña que posee la base *mysql* como nivel de seguridad de ingreso a la misma, finalmente se define el parámetro “*nombre*” que hará referencia al nombre de la base de datos desarrollada previamente.

Una vez creado el código de acceso, se crea un objeto cursor (un objeto cursos un elemento que representa a un conjunto de datos determinado por una consulta SQL, el cursor permite recorrer fila a fila, leer y eventualmente modificar un conjunto de resultados) “*bd.cursor*”, que permitirá ejecutar todos los parámetros anteriormente definidos con la línea de código “*cursor.execute (sql)*” dando como resultado la conexión de la base de batos *mysql* con Python, creado el vínculo, se puede cargar la información en una lista para facilitar el



manejo de la información contenida en la base de datos. Finalmente se cierra el objeto cursor para evitar problemas en el funcionamiento de la misma mediante el comando “*bd.close*”.

Adicional a la base de datos *mysql*, se desarrolla un algoritmo para almacenar la información de cada vehículo que es detectado en un archivo de texto, generando un registro al cual se puede acceder desde la interfaz de usuario del programa o desde la interfaz del sistema operativo, seleccionando el archivo en su ubicación de origen en una dirección establecida, la ubicación predefinida del proyecto es: */home/Debian/archivos\_tesis/base\_de\_datos*, como se observa en la figura 46, ubicación donde se encontrara un archivo de texto con la información del vehículo detectado.



**Figura 46 Registro de Placas Vehiculares Detectadas**

El procedimiento general de acceso, funcionamiento y comparación de la base de datos con la información de la placa vehicular se presenta de manera gráfica mediante un flujograma de funcionamiento en la figura 47.

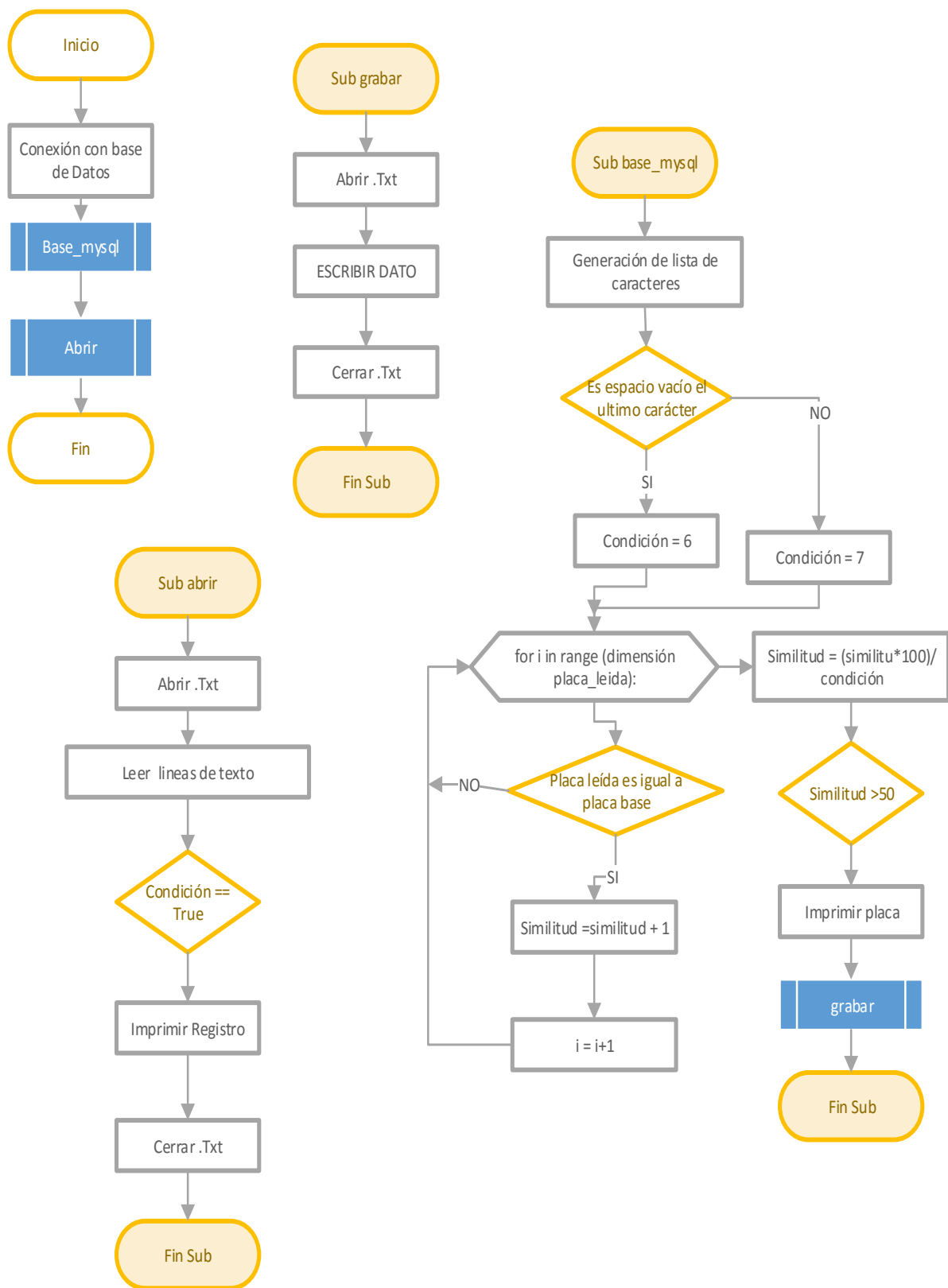


Figura 47 Diagrama de Flujo del Funcionamiento de la Base de Datos

## 2.9. Interfaz de usuario python

La interface gráfica en python es una de las herramientas más importantes al momento de crear un medio directo e interactivo entre el usuario y un lenguaje de programación, permitiendo al administrador el manejo y ejecución rápida de diferentes funciones, comandos, entre otros. La presentación de información a través de ventanas se maneja con los widgest (cajas de textos, botones, barras de direcciones, etc.) que permiten procesar datos dependiendo de un algoritmo previamente diseñado como se presenta en la figura 48.

Actualmente Python posee diferentes alternativas disponibles como Tkinter, wxPython, PyQt, PySide, QtCreator y PyGTK que cumplen con todos los estándares de componente visual, siendo compatible con varias plataformas de Unix y sistemas operativos como Windows y Macintosh.



Figura 48 Grafica de Interfaz de Usuario en Python

El módulo Tkinter (“Tk”) es una de varias interfaces gráficas estándar que posee Python, esta interfaz se suministra a través de una serie de módulos de Python. Para utilizar Tkinter, se requiere de un módulo que

necesariamente debe ser importado desde el intérprete de Python de la siguiente manera:

***import Tkinter o tambien from Tkinter import \****

Al importar la librería Tkinter Python puede hacer uso de varias funciones y clases de widgets, que proporcionan un medio de presentación de datos para diversos tipos de las aplicaciones. Para facilitar la sintaxis del algoritmo es recomendable asignar una etiqueta que permita minimizar el algoritmo de diseño ahorrando algo de mecanografía, efectuando la importación de la librería como:

***import Tkinter as Tk***

Entre las funciones que permite utilizar el módulo Tkinter se encuentran cuadros de textos, etiquetas, botones, etc, que se hallan detallados de mejor manera en la tabla 3.

**Tabla 3**

**Funciones disponible para manejo de Tkinter**

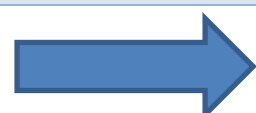
<b>Operador</b>	<b>Descripción</b>
<b>Button</b>	El widget de button se utiliza para mostrar los botones de la aplicación.
<b>Canvas</b>	El widget de canvas se utiliza para dibujar formas, tales como líneas, óvalos, polígonos y rectángulos, en su aplicación.
<b>CheckButton</b>	El widget CheckButton se utiliza para mostrar una serie de opciones como casillas de verificación. El usuario puede seleccionar varias opciones a la vez.
<b>Entry</b>	El control Entry se utiliza para mostrar un campo de texto de una sola línea para la aceptación de los valores de un usuario.

**Continúa**



<b>Frame</b>	El control Frame se utiliza como un widget contenedor para organizar otros widgets.
<b>Label</b>	El control Label se utiliza para proporcionar un subtítulo de una sola línea para otros widgets. También puede contener imágenes.
<b>Listbox</b>	El widget de Listbox se utiliza para proporcionar una lista de opciones a un usuario.
<b>Menu Button</b>	El widget Menu Button se utiliza para visualizar los menús de la aplicación.
<b>Menu</b>	El widget de Menu se utiliza para proporcionar varios comandos de un usuario.
<b>Mensaje</b>	El widget de mensajes se utiliza para mostrar los campos de texto de varias líneas para la aceptación de los valores de un usuario.
<b>Radio Button</b>	El widget de Radio Button se utiliza para mostrar una serie de opciones como botones de radio. El usuario puede seleccionar sólo una opción a la vez.
<b>Scale</b>	El control de Scale se utiliza para proporcionar un control deslizante.
<b>Scrollbar</b>	El widget de Scrollbar se utiliza para agregar la capacidad de desplazamiento de varios widgets, tales como cuadros de lista.
<b>Text</b>	El widget de Text se utiliza para mostrar texto en varias líneas.
<b>Upper Level</b>	Los widget de Upper Level se utilizan para proporcionar un recipiente de ventana separada.
<b>Spinbox</b>	Los widget Spinbox son una variante del widget estándar Entrada Tkinter, que puede ser utilizado para seleccionar a partir de un número fijo de valores.
<b>PanedWindow</b>	Un PanedWindow es un widget que contiene cualquier número de paneles, dispuestos en horizontal o vertical.

Continua



<b>LabelFrame</b>	Un LabelFrame es un simple widget contenedor. Su objetivo principal es actuar como un separador o recipiente para diseños de ventana complejas.
<b>tkMessageBox</b>	Este módulo se utiliza para mostrar cuadros de mensaje en sus aplicaciones.

### 2.9.1. Crear Ventanas con Tkinter

Para generar ventanas python normalmente importa toda la librería para facilitar su ejecución. Una vez importado el módulo se inicializa Tkinter, para lo cual es necesario crear un widget raíz Tk, que es un tipo de ventana normal como se aprecia en la figura 49, con una barra de título y otra decoración proporcionada por el gestor de ventanas. Sólo se debe crear un widget raíz para cada programa y debe ser creado antes que los otros widgets así.

```
window= tk.Tk().
```

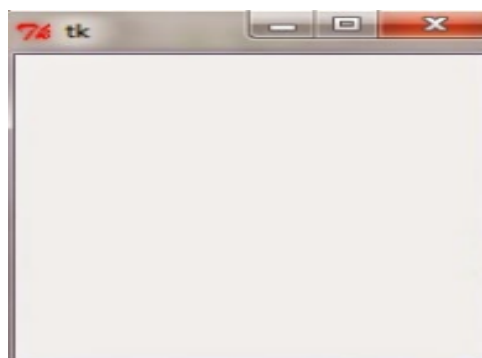
A continuación, se inserta un título en la ventana generada para otorgar un nombre a la ventana que se ha creado:

```
window.wm_title("Interfaz tk")
```

Finalizando siempre con la línea de código:

```
window.mainloop ()
```

Dicho código hará que el programa permanezca dentro de un bucle de eventos hasta que se cierre la ventana generada. El *main loop* no sólo controla eventos de usuario (como clics del ratón y las pulsaciones de tecla) o los sistemas de ventanas, además se encarga de las operaciones en cola por sí mismo, tales como son la gestión de la geometría y actualizaciones de pantalla.



**Figura 49 50 Ventana Principal Generada por Tkinter**

### **2.9.2. Manejo de Etiquetas**

El uso de etiquetas en la interfaz gráfica facilita la interacción de los widgets con el usuario mediante el comando Label (ventana,text), donde el primer parámetro es la instancia de ventana, donde se especifica el nombre del objeto Tk(), creado, y a la variable "text" permite asignar un valor tipo cadena entre 2 comillas.

### **2.9.3. Gestión de la geometría**

En Tkinter todos los widgets poseen un acceso a las técnicas de gestión de la geometría, es decir, posee métodos de ubicación y posicionamiento con el único objetivo de posicionar un determinado widget (figura 49), dentro de una zona específica de la ventana Tkinter, permitiendo así organizar los widgets de una mejor manera y posicionar todos los componentes a emplearse en la interfaz correctamente. Tkinter ostenta las siguientes clases:

#### **2.9.3.1. El método pack ()**

Este gestor de la geometría organiza los widgets en bloques antes de colocarlos en el widget padre como se puede apreciar en figura 50 (a).

#### **2.9.3.2. El método grid ()**

La figura 50 (b) presenta el gestor de widgets de la geometría grilla que organiza en una estructura de tabla los widget dentro la ventana de tk.

#### **2.9.3.3. El método place ()**

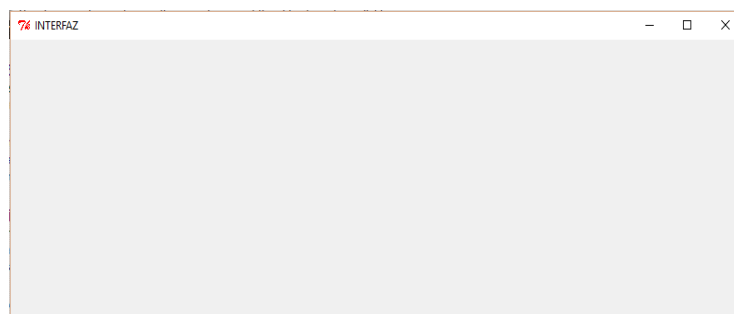
Finalmente el gestor de widgets de la geometría place organiza los objetos colocándolos en una posición específica en la ventana detallando coordenadas en x, y para ubicar un objeto como se observa en la figura 50 (c).



**Figura 51 Gestión de Geometría Tkinter**

### 2.9.4. Desarrollo de la Interfaz

Para el diseño de la interfaz se emplea el intérprete de python, iniciando por la importación de los paquetes o librerías requeridas como se menciona con anterioridad, posteriormente se inicializa la ventana de trabajo empleando el comando `“window= tk.Tk()”`, además se agrega el nombre de la ventana con el código `“window.wm_title(“Interfaz”)”` y se especifican las dimensiones de ancho y alto del área de trabajo mediante la sentencia `“imageFrame=tk.Frame(window ,width=880, height=300)”`, obteniendo una ventana como se presenta en la figura 51



**Figura 52 Creación de Ventana de Trabajo TKinter**

Creado el entorno de trabajo, se procede a inicializar las variables, que contendrán la información que será presentada en la ventana Tk, el módulo Tkinter emplea una función denominada “rastreo”, la cual permite conocer cuando el sistema realiza un cambio en las variables, es decir, permite actualizar determinados widgets cada que se modificada una variable asociada a la función rastreo.

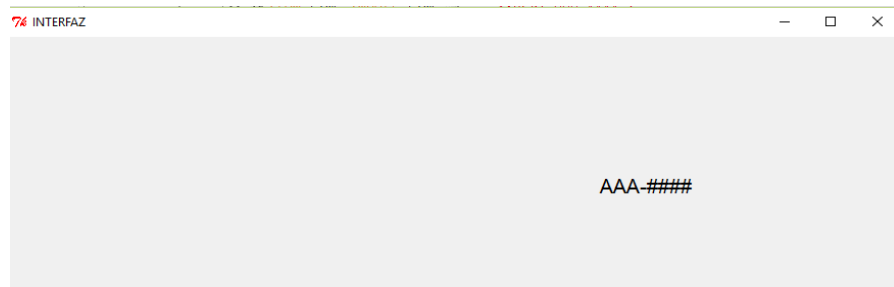


Las variables se inicializan a través de expresión “v=tk.StringVar()”, que es un constructor ( subrutina cuya finalidad es inicializar un objeto de una clase) de tipo string, donde el argumento “v” es el nombre de la variable que almacena la información, mientras tk.StringVar() especifica el tipo de constructor pudiendo ser este de tipo entero, flotante, cadena, etc,. Para el proyecto se ha optado por los constructores de tipo cadena, para facilitar el manejo de los datos string generados por el reconocimiento óptico de caracteres de las palcas vehiculares.

Finalizada la inicialización de las variables se procede a especificar ciertos parámetros correspondientes al tipo de fuente de texto en la ventana Tk, como son el tipo de letra, tamaño, etc., mediante el manejo de etiquetas (Label), además se asigna una ubicación determinada en la ventana empleando el método de gestión de la geometría place, definiendo sus coordenadas tan en el eje X como en el eje Y, anteponiendo el nombre de la variable que contiene la etiqueta seguida de termino place.

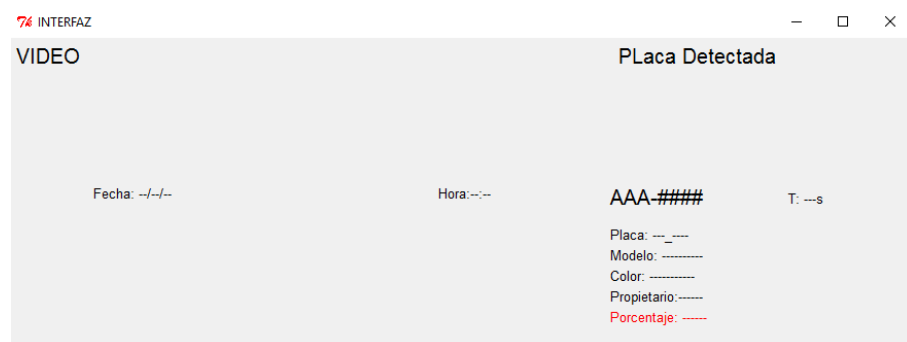
```
placatext=tk.Label(window,textvariable=v,font=('Agency FB:',15))  
placatext.place(x=432+150,y=140)
```

Finalmente para visualizar las variables previamente creadas y configuradas se recurre a la sentencia de control “set”, que se encarga de exportar los datos de las variables hacia la ventana de trabajo, a través de las etiquetas anteriormente modificadas, la sentencia set se presenta anteponiendo el nombre de la variable de la siguiente manera “v.set”.



**Figura 53 Presentación y ubicación de una variable en la Ventana Tkinter**

De igual manera para visualizar más variables en la venta solamente se reemplaza el nombre de la variable, su zona de ubicación obteniendo una ventana como se presenta en la figura (53)



**Figura 54 Variables presentadas en Tkinter**

El módulo Tkinter además del manejo de texto estático en ventanas Tk a través de etiquetas, permite visualizar imágenes o capturas de video, para esto, se requiere importar el paquete o librería PIL (Librería de Imágenes de Python) para manejo de imágenes y se procede implementar el algoritmo:

```
"img = PhotoImage(file="tutor de programacion.gif")",  
"display = tk.Label(imageFrame)"  
"display.place(x=440+150,y=30)"
```

Especificando el nombre de la imagen a ser presentada, de igual manera para ubicar la imagen de una determinada zona de la pantalla se emplea la gestión de geometría place como se puede visualizar en la figura (54).



**Figura 55 Visualización de Imágenes en Ventana Tkinter**

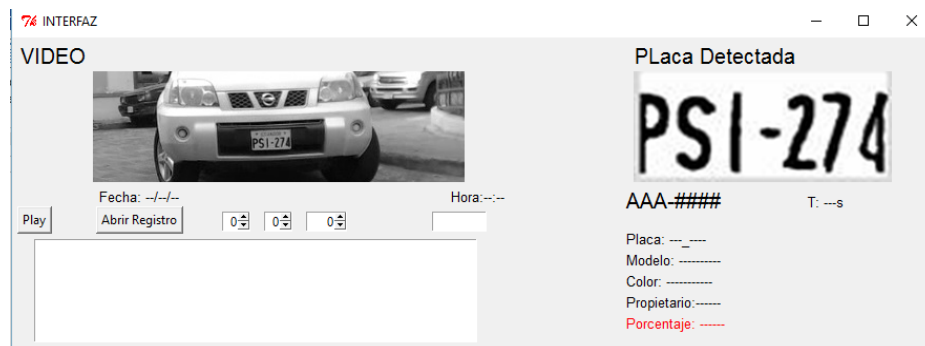
Para culminar el diseño de la interfaz se implementa un sliderFrame y un listBox, figura 55 para presentar los datos guardados en el registro de datos creado anteriormente en el apartado 2.8.1, para ello se emplea el siguiente algoritmo, especificando sus parámetros tanto de ancho y alto de la lista así como el valor de la columna y fila de inicio.

```
"sliderFrame = tk.Frame(window, width=0, height=0)"
```

```
"sliderFrame.grid(row = 1, column=0, padx=0, pady=0)"
```

```
"listbox = tk.Listbox(window,width=70,height=6)"
```

```
"listbox.place(x=20, y=190)"
```



**Figura 56 Visualización de sliderFrame y Listbox en Ventana Tkinter en Python**

## CAPÍTULO III

### 3. PRUEBAS Y RESULTADOS

En este capítulo se presentan las pruebas y resultados obtenidos en las diferentes etapas del proyecto de manera práctica, en el que se incluye el funcionamiento del sistema para obtención de placas vehiculares, el proceso de reconocimiento óptico de caracteres y finalmente la comparación de las placas obtenidas con placas preestablecidas (patrón) en un base de datos.

#### 3.1. Descripción de pruebas realizadas

A continuación se describe el proceso realizado para verificar el correcto funcionamiento del sistema de reconocimiento de placas vehiculares, usando una tarjeta Beaglebone Black RevC, además de los resultados obtenidos.

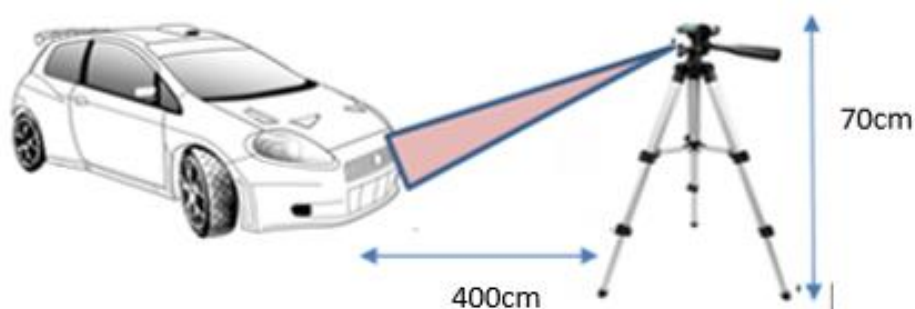
El sistema presenta un HMI (interfaz humano maquina) denominado “*Interfaz*”, la cual consta de diversos campos visuales de datos como la extracción de la información perteneciente a la placa vehicular de un automotor que será procesada por medio del algoritmo de reconocimiento óptico de caracteres Tesseract, donde se puede constatar que los datos extraídos por el sistema coinciden con la placa vehicular.

Una vez obtenida la información resultante del sistema de reconociendo óptico de caracteres, el dato es visualizado por medio de la interfaz gráfica, posteriormente es comparado con la información previamente establecida en una base datos, si dicha información coincide se visualizará “*vehículo reportado*” en la interfaz gráfica, lo que significa que el vehículo presenta antecedentes.

### 3.2. Descripción de pruebas realizadas

Las pruebas de funcionamiento del software de reconcomiendo de placas vehiculares se han realizado en la calle Quijano Ordoñez junto al parque San Francisco de la ciudad de Latacunga, las pruebas se realizaron durante tres días a diferentes horas para varios vehículos de prueba, en variadas ocasiones.

Para la detección de las placas vehiculares la cámara se encuentra ubicada a una altura de 70 cm, con una distancia para la medición de máximo 400 cm entre la cámara y el vehículo de prueba, rango en el cual se puede apreciar con mayor detalle la placa vehicular para su análisis como se puede apreciar en la figura 57, dependiendo además del estado de la placa vehicular, es decir, que la placa debe ser legible, deberá encontrarse en perfectas condiciones, ni poseer un gran deterioro físico, además no debe poseer accesorios adicionales a los establecido por la Agencia Nacional de Tránsito especificados en los artículos 1, 3, 4 de la resolución N° 056-DIR-2013-ANT.



**Figura 57 Descripción de la Ubicación de Cámara Respecto al Vehículo de Prueba**

#### 3.2.1. Pruebas Realizadas a Diferentes Vehículos con la Tarjeta Beaglebone Black

Para el análisis de resultados se presentan las tablas 4 y 6 que contiene varias imágenes de vehículos sometidos a pruebas en diferentes horas del día entre las 8:30am y las 18:30pm, pruebas que se realizaron durante 2 días en la calle Quijano Ordoñez junto al parque San Francisco de la ciudad de Latacunga, a diferentes vehículos que transitan diariamente por el sector.

En la tabla 4 se presentan 45 placas vehiculares identificadas correctamente con un porcentaje del 100%, en relación al número de caracteres acertados. Dicho porcentaje es calculado dependiendo del número de caracteres que posea la placa vehicular, es decir, si la placa contiene 6 o 7 letras equivaldrán al 100% para cada caso.

**Tabla 4**  
**Placas Vehiculares Detectadas por el Sistema con un Porcentaje de 100%**

Placa Vehicular Inspeccionada Visualmente	Placa Detectada por el Sistema	Porcentaje de Similitud entre las Placas (%)
PBY-1226	PBY-1226	100
PJC-262	PJC-262	100
PBU-9347	PBU-9347	100
CBA-1322	CBA-1322	100
PTN-685	PTN-685	100
XBA-8235	XBA-8235	100
PYB-162	PYB-162	100
PBL-2213	PBL-2213	100
TAO-0504	TAO-0504	100
PTN-685	PTN-685	100
XBA-3464	XBA-3464	100
PCJ-1603	PCJ-1603	100
XBB-2978	XBB-2978	100
PBK-4325	PBK-4325	100
PJU-350	PJU-350	100
PBJ-5807	PBJ-5807	100

Continua



IBA-5281	IBA-5281	100
PKR-154	PKR-154	100
XBA-3809	XBA-3809	100
XBB-2309	XBB-2309	100
PSO-297	PSO-297	100
XBA-4550	XBA-4550	100
XBU-326	XBU-326	100
XBA-3120	XBA-3120	100
XBB-2116	XBB-2116	100
PVU-437	PVU-437	100
PBH-7720	PBH-7720	100
PCF-2939	PCF-2939	100
PBH-8347	PBH-8347	100
XBA-1240	XBA-1240	100
XBB-2867	XBB-2867	100
XBB-1404	XBB-1404	100
TDH-254	TDH-254	100
XBB-4234	XBB-4234	100
XBA-7332	XBA-7332	100
TCP-627	TCP-627	100
XBA-4271	XBA-4271	100
PBF-1221	PBF-1221	100
XBB-2232	XBB-2232	100
XAI-428	XAI-428	100

Continua



XBB-3172	XBB-3172	100
XBB-2867	XBB-2867	100
PBH-5955	PBH-5955	100
XBA-3888	XBA-3888	100

En la tabla 5 se puede apreciar ejemplos de placas vehiculares detectadas con 100% de efectividad, durante el desarrollo de las pruebas del software de detección.

**Tabla 5**







**Ejemplos de Vehículos Detectados con un Porcentaje de 100%**

PLACA VEHICULAR	IMAGEN VEHÍCULO	Placa Detectada por el sistema
XBS-0385		
XAH-0196		
PBS-404		
POH-538		

Continua





<p>PYT-679</p>		<p>PLaca Detectada <b>PYT-679</b> PYT-679</p>
<p>TBA-8998</p>		<p>PLaca Detectada <b>TBA-8998</b> TBA-8998</p>
<p>PBD-2054</p>		<p>PLaca Detectada <b>PBD-2054</b> PBD-2054</p>
<p>PTG-940</p>		<p>PLaca Detectada <b>PTG-940</b> PTG-940</p>
<p>XBB-1973</p>		<p>PLaca Detectada <b>XBB-1973</b> XBB-1973</p>
<p>XBA-8936</p>		<p>PLaca Detectada <b>XBA-8936</b> XBA-8936</p>

De igual manera en la tabla 6 se puede apreciar placas identificadas por el sistema con un reconocimiento óptico de caracteres deficiente, es decir, que las letras obtenidas no corresponden en un 100% a su placa original.








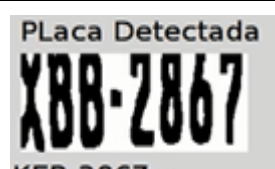

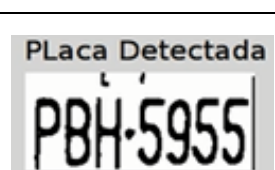

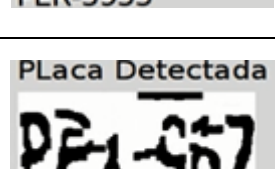

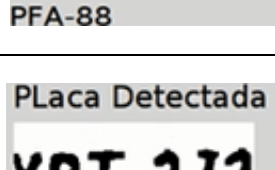
**Tabla 6****Placas Vehiculares Detectadas con un Porcentaje Menor al 100%**

<b>Placa Vehicular Inspeccionada Visualmente</b>	<b>Placa Detectada por el Sistema</b>	<b>Porcentaje Similitud entre las Placas (%)</b>
XAI-1199	XMH-95	16,67
PXK-774	PYM-774	66,67
PCZ-249	FCZ-249	83,33
TBC-8693	TBO-8693	85,71
PBA-9318	PEA-9318	85,71
PLP-248	WIS-11	0
PBU-9347	PBU-9317	85,71
PZM-011	BIN-0111	50
PXQ-938	PXN-1544	33,33
IBB-5440	BBS-110	14,29
PLD-806	DBD-8	33,33
PBD-3303	FED-3303	71,43
XBB-2344	KBB-2311	57,14
PKR-154	PIE-154	66,67
PCC-7435	PCC-7635	85,71
PBO-8844	EII-1181	0
PBO-8844	PBO-8811	71,43
XBA-3809	XBA-3803	85,71
XBB-2009	BLL-11	0
TDO-590	TBG-590	66,67

En la tabla 7 se puede apreciar ejemplos de placas vehiculares detectadas con una efectividad menor al 100%, es decir, más de un carácter resultó erróneo, durante el desarrollo de las pruebas del software de detección.

Tabla 7

Ejemplos de Vehículos Detectados con un Porcentaje menor al 100%

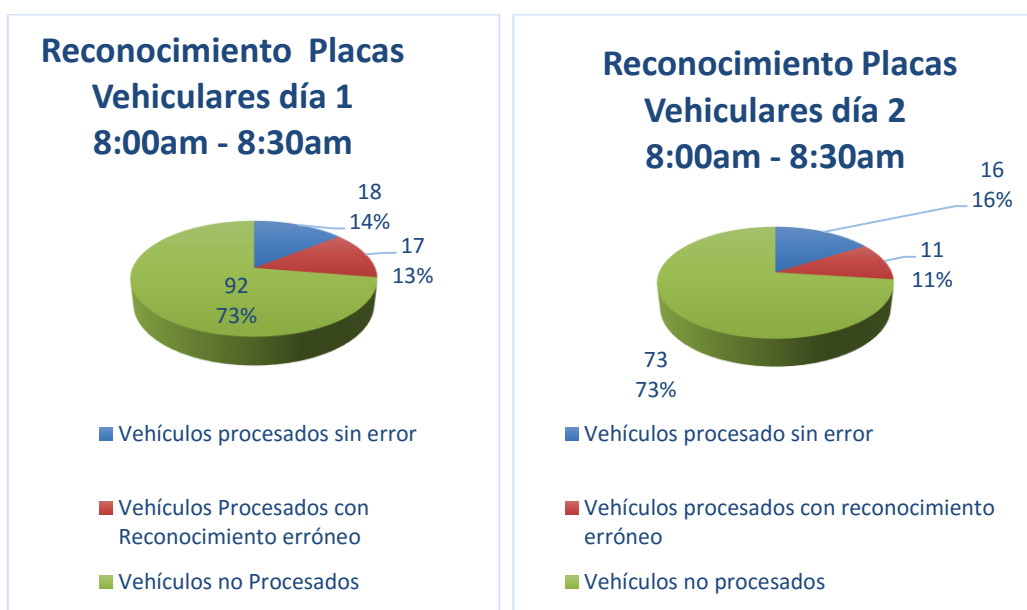
Placa Vehicular	Imagen del Vehículo	Placa Detectada por el sistema
PBG-1306		<p>PLaca Detectada</p>  <p>FBG-1308</p>
TBD-2734		<p>PLaca Detectada</p>  <p>TBD-273</p>
XAI-420		<p>PLaca Detectada</p>  <p>XAI-118</p>
XBB-2867		<p>PLaca Detectada</p>  <p>KER-2867</p>
PBH-5955		<p>PLaca Detectada</p>  <p>PER-5955</p>
PPA-967		<p>PLaca Detectada</p>  <p>PFA-88</p>
XBT-272		<p>PLaca Detectada</p>  <p>XBT-212</p>

## RESULTADOS

Los siguientes resultados fueron obtenidos en el transcurso de dos días, empleando una tarjeta Beaglebone Black para el reconocimiento de placas vehiculares.

- **De 8:00am a 8:30am**

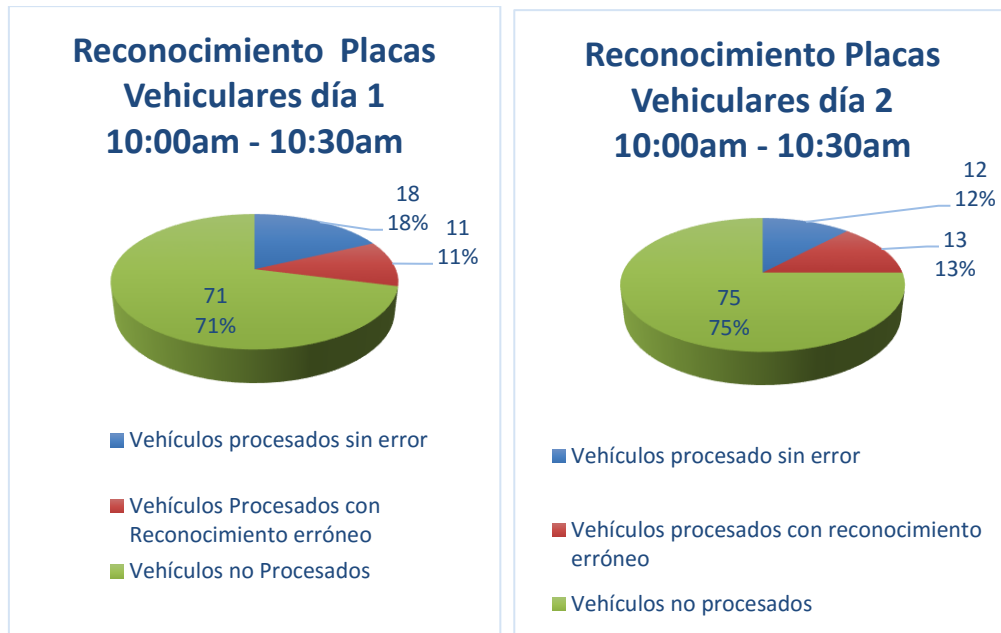
Para las pruebas realizadas se tomaron en consideración 120 vehículos para el día uno y 100 vehículos para el segundo día para los análisis, obteniendo los resultados que se presentan en la figura 58.



**Figura 58 Reconocimiento de Placas Vehiculares de 8:00am – 8:30am**

- **De 10:00am a 10:30am**

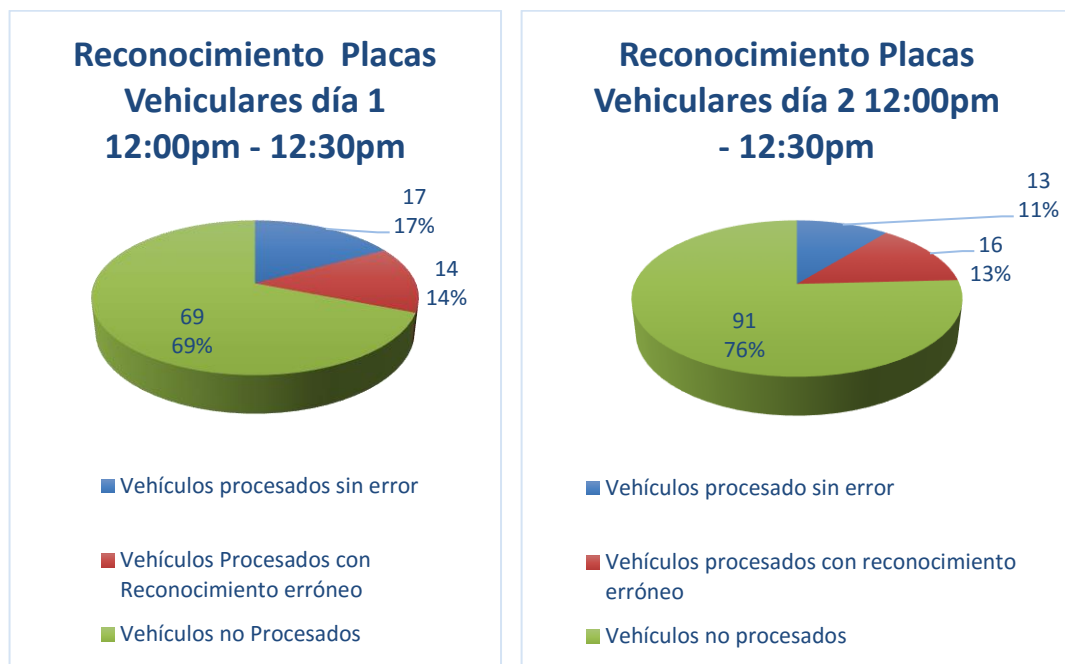
La figura 59 presenta las pruebas realizadas a 100 vehículos tanto para el día de prueba uno como para el día de prueba dos, obteniendo los siguientes resultados.



**Figura 59 Reconocimiento de Placas Vehiculares de 10:00am – 10:30am**

- **De 12:00pm a 12:30pm**

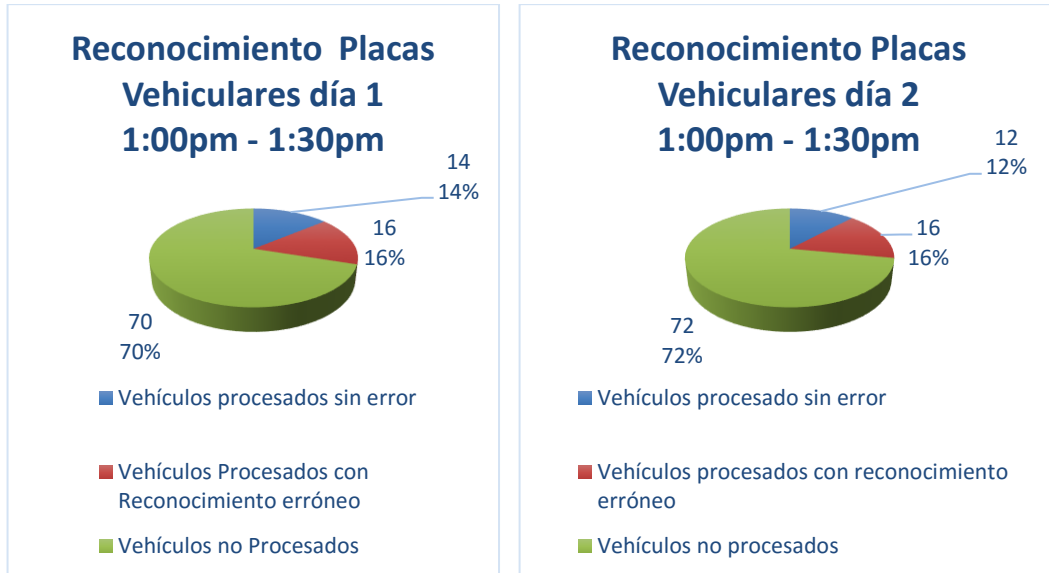
La figura 60 presenta las pruebas realizadas a 100 vehículos el primer día y 120 vehículos para el segundo día alcanzando los siguientes resultados.



**Figura 60 Reconocimiento de Placas Vehiculares de 12:00pm – 12:30pm**

- **De 1:00pm a 1:30pm**

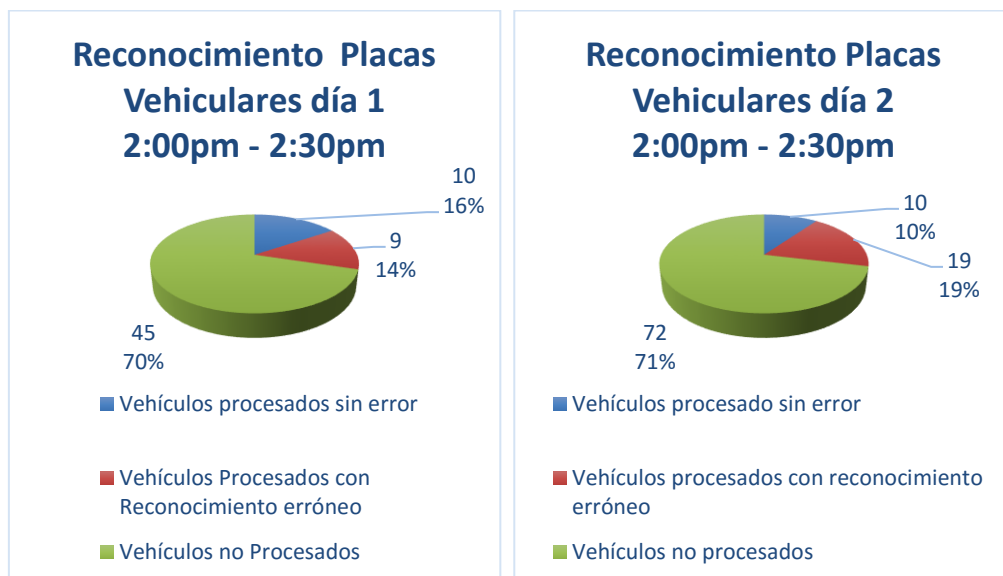
Para las pruebas realizadas se tomaron en consideración 100 vehículos para el día uno de prueba, al igual que el segundo día de prueba, consiguiendo los resultados siguientes de la figura 61.



**Figura 61 Reconocimiento de Placas Vehiculares de 1:00pm – 1:30pm**

- **De 2:00pm a 2:30pm**

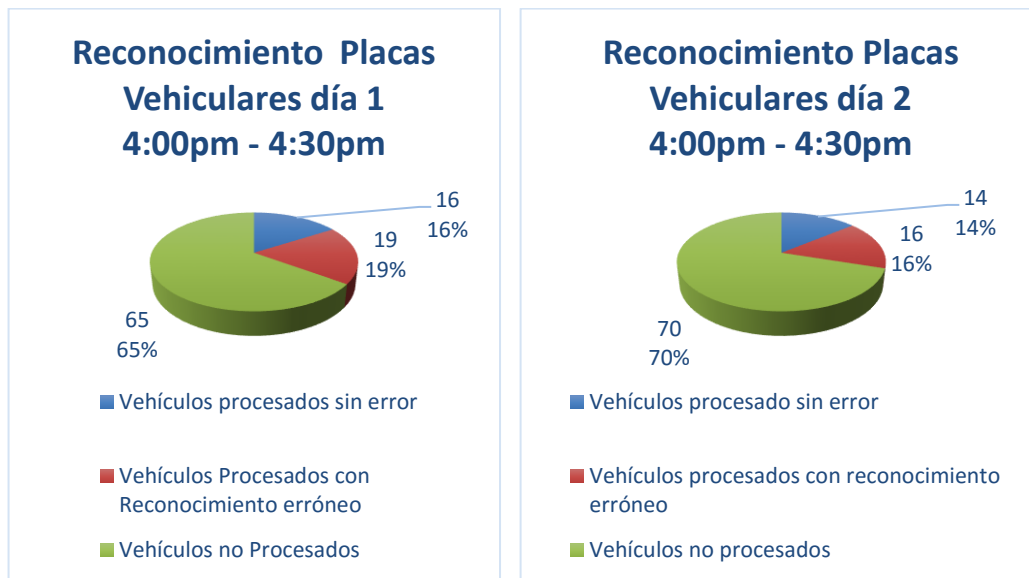
Para las pruebas realizadas se tomaron en consideración 64 vehículos para el día uno y 101 vehículos para el día dos para los análisis, obteniendo los siguientes resultados figura 62.



**Figura 62 Reconocimiento de Placas Vehiculares de 2:00pm – 2:30pm**

- **De 4:00pm a 4:30pm**

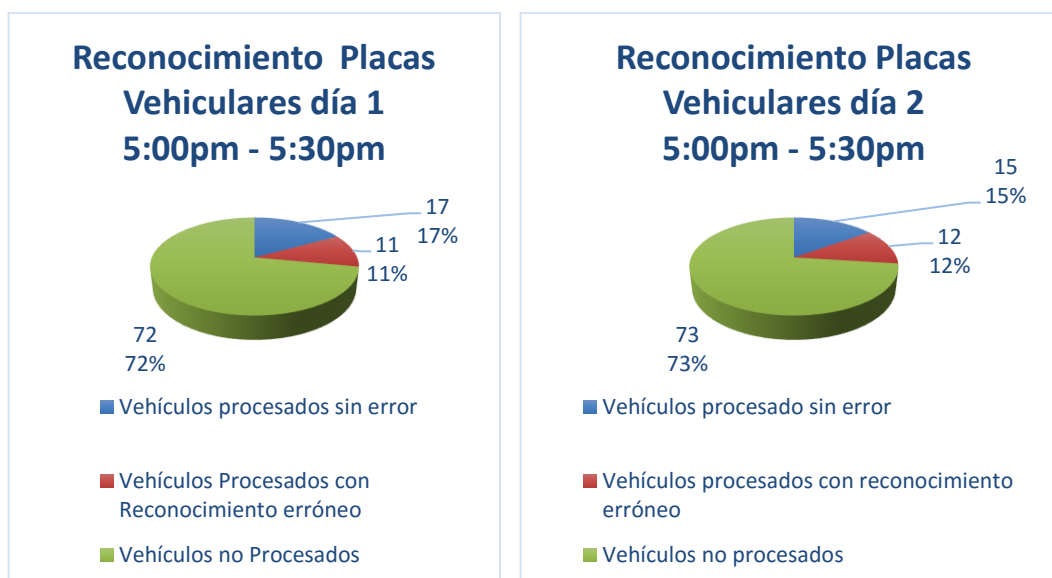
A continuación se realizó pruebas a 100 vehículos para el primer caso, al igual que el segundo día de prueba para verificar el funcionamiento del sistema consiguiendo la información de la figura 63.



**Figura 63 Reconocimiento de Placas Vehiculares de 4:00pm – 4:30pm**

- **De 5:00pm a 5:30pm**

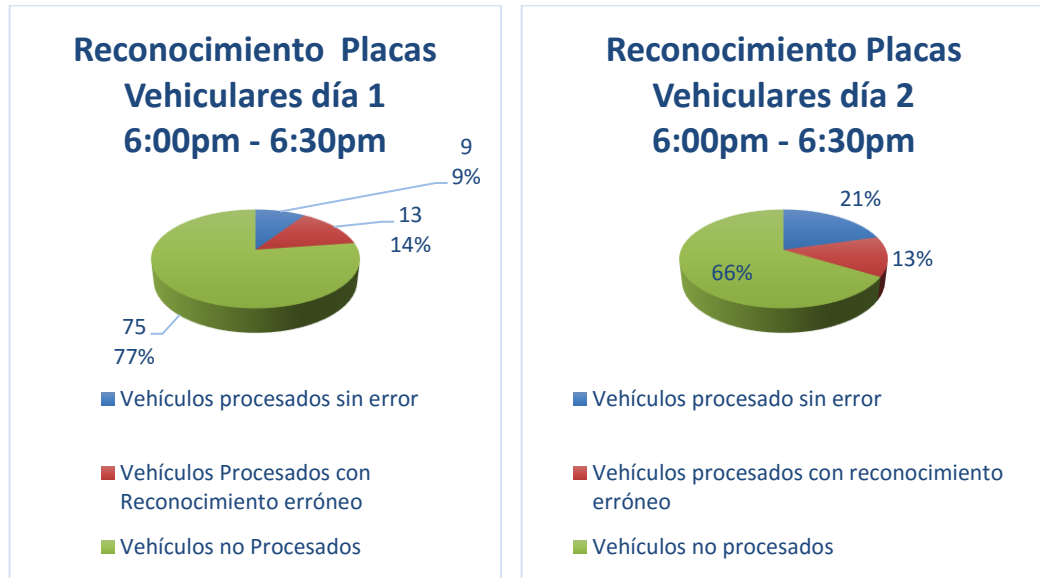
La figura 64 presenta las pruebas realizadas a 100 vehículos tanto para el primer día de prueba como para el segundo día de prueba alcanzando la información siguiente.



**Figura 64 Reconocimiento de Placas Vehiculares de 5:00pm – 5:30pm**

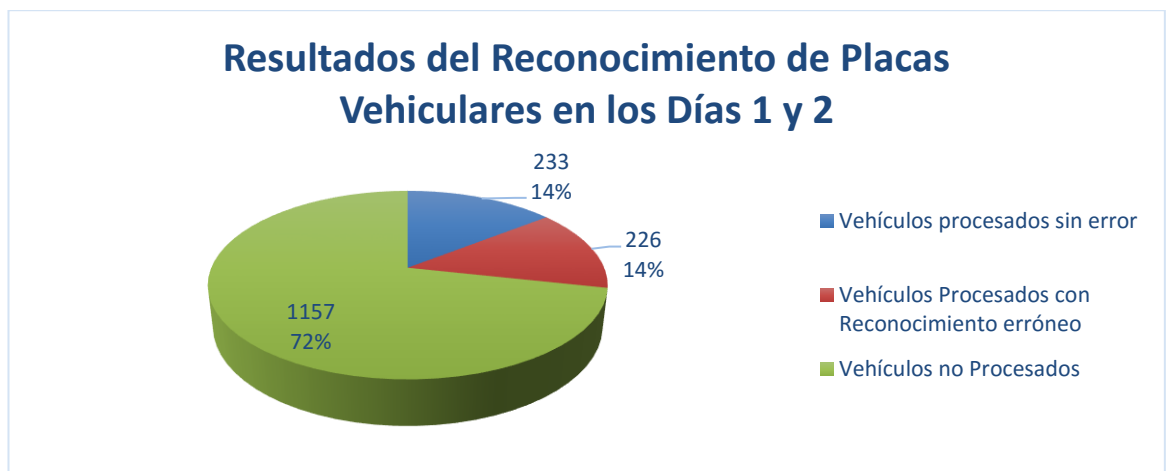
- **De 6:00pm a 6:30pm**

Finalmente se examinó a 97 vehículos para el primer caso mientras para el segundo caso se examinó 107 vehículos para verificar el funcionamiento del sistema consiguiendo la información de la figura 65.



**Figura 65 Reconocimiento de Placas Vehiculares de 6:00pm – 6:30pm**

Una vez finalizado el análisis de los días 1 y 2 se procedió a reunir la información para realizar un estudio general del sistema de reconocimiento de placas vehiculares, para conocer el porcentaje y cantidad de vehículos identificados correctamente, así como el número de vehículos no detectados por el software como se demuestra en la figura 66.



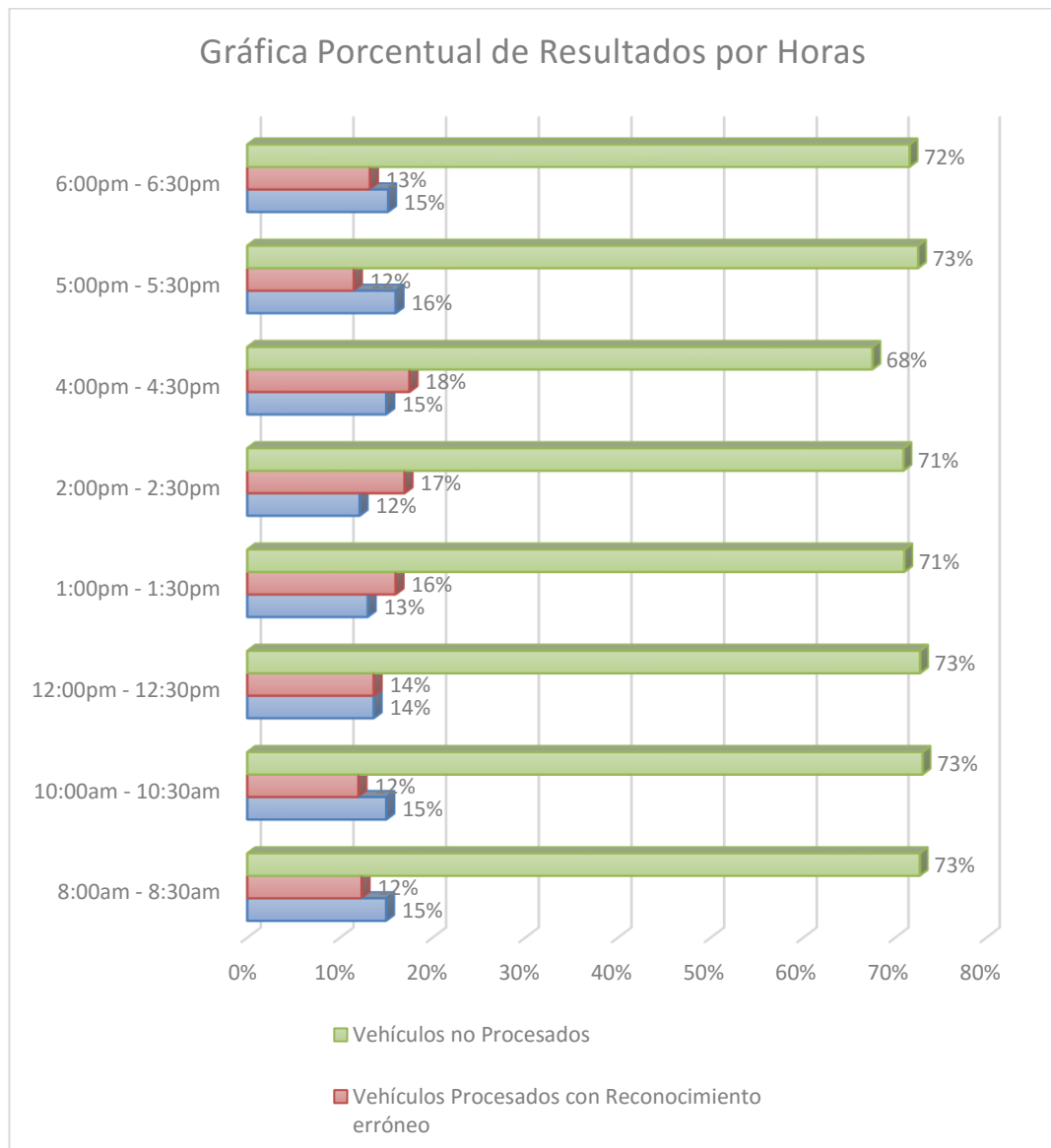
**Figura 66 Resultados del Reconocimiento de Placas Vehiculares en los Días 1 y 2**



En el cual se obtuvo un porcentaje del 14%, correspondiente a 233 vehículos detectados correctamente, ya que las condiciones de las placas vehiculares cumplían adecuadamente con las especificaciones técnicas de los artículos establecidos por la Agencia Nacional de Tránsito en la resolución N° 056-DIR-2013-ANT. Además el resultado correspondiente al 14% que equivale a 226 vehículos, que fueron identificados por el sistema obteniendo una extracción de caracteres no idónea. Finalmente, el 72% de resultados obtenidos corresponden a 1157 vehículos que no fueron detectados por el sistema de reconocimiento de placas vehiculares, por los motivos que se indican a continuación: procesamiento deficiente de la tarjeta Beaglebone Black, placas vehiculares deterioradas, ubicación de la placa vehicular inadecuada, deficiencia del algoritmo, resolución de la cámara, entre otros, dando un total de 1616 vehículos considerados para el estudio manejando una tarjeta Beaglebone Black.

En la figura 67 se muestran los resultados obtenidos durante los dos días de prueba utilizando un gráfico porcentual de barras, en donde se particulariza los datos obtenidos en cada intervalo de tiempo empleando la tarjeta Beaglebone Black, dejando en claro que el porcentaje de vehículos no procesados por el sistema a lo largo del día en la mayoría de los casos se mantiene por arriba del 70% de los vehículos que transitaron, además el porcentaje de vehículos procesados sin error ostentan un mayor nivel de acierto entre 2 intervalos de tiempos grandes, siendo el primero entre las 8:00am y las 12:30pm con un porcentaje casi constante de aciertos del 15%, también en el horario comprendido entre las 4:00pm y las 6:00pm de igual manera manteniendo un porcentaje del 15%, en el resto de tiempos el porcentaje de acierto no superó el 14% de todos los vehículos analizados para cada lapso de tiempo. En cuanto al análisis de los vehículos procesados con reconocimiento erróneo, la información muestra que durante la 1:00pm y las 4:30pm existió un

mayor porcentaje de autos procesados con errores en su identificación de caracteres con un valor promedio correspondiente al 17%, de todos los automóviles analizados en el mencionado intervalo de tiempo, mientras que en el resto de pruebas realizadas el porcentaje no superó el 15% de los vehículos en cada horario de estudio.



**Figura 67 Gráfica Porcentual de Resultados por Horas**

### 3.2.2. Pruebas Realizadas en el Computador

En este apartado se presentan pruebas y resultados del funcionamiento del algoritmo desarrollado para la detección de placas vehiculares empleando como sistema de procesamiento una

computadora personal, incrementando los recursos de procesamiento como son la memoria, capacidad de procesamiento de video y velocidad del procesador como se indica en la tabla 8.

**Tabla 8**  
**Especificaciones Técnicas del Computador**

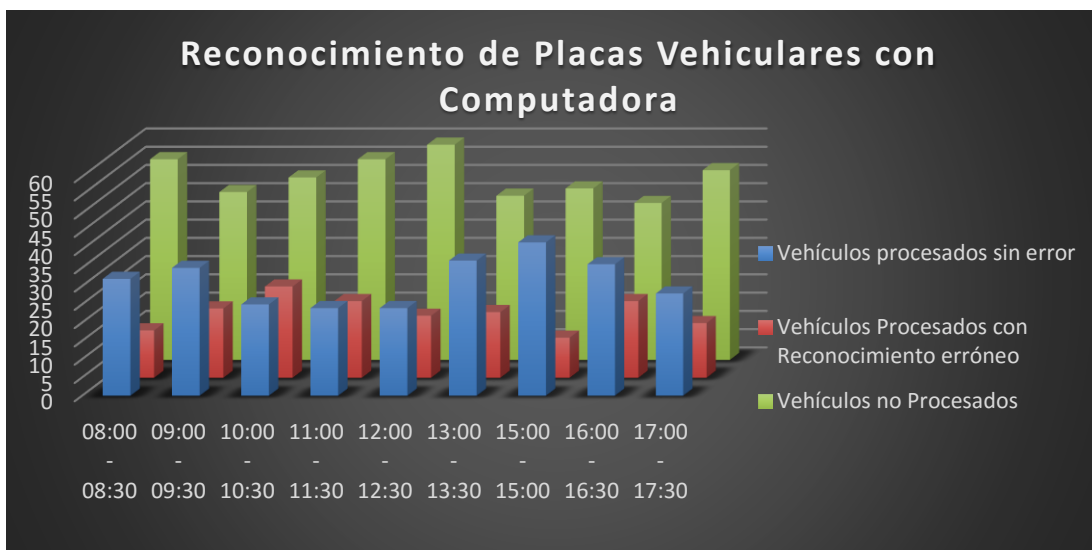
Procesador	Intel Core i5 - 2.2Ghz
Memoria RAM	6GB
Disco Duro	1 TB
Puertos	HDMI, USB 3.0, USB2.0
Sistema operativo	Debian 8.2
Memoria Caché	3 MB de caché interna

Las pruebas realizadas se efectuaron durante un día a diferentes horas entre las 8:00 am y las 17:30, para alrededor de 900 vehículos puestos a prueba para demostrar el funcionamiento del sistema, se empleó una pc portátil Acer serie E5-471-54v8-core i5, arrojando al finalizar la actividad los resultados expuestos en la tabla 9.

**Tabla 9**  
**Resultados Obtenidos realizando el procesamiento mediante el uso de un Computador**

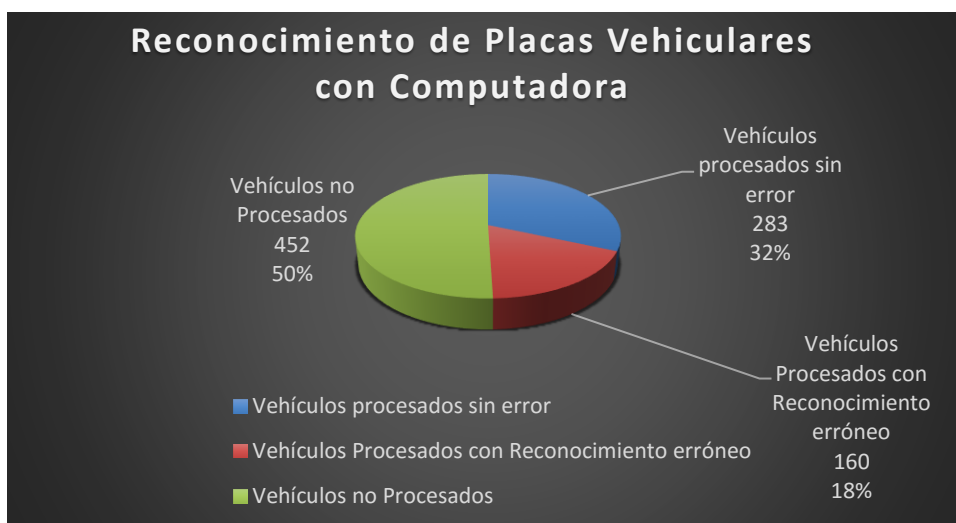
Horas	Vehículos Procesados sin error	Vehículos Procesados con Reconocimiento erróneo	Vehículos no Procesados
8:00 - 8:30	32	13	55
9:00 - 9:30	35	19	46
10:00 - 10:30	25	25	50
11:00 - 11:30	24	21	55
12:00 - 12:30	24	17	59
13:00 - 13:30	37	18	45
15:00 - 15:00	42	11	47
16:00 - 16:30	36	21	43
17:00 - 17:30	28	15	52

Los datos expuestos anteriormente se pueden apreciar de mejor manera en la figura 68 mediante el diagrama de barras para cada ítem evaluado.



**Figura 68 Representación de Resultados Obtenidos por Computador**

En la figura 69 se puede apreciar en forma porcentual el número de vehículos con una respuesta correcta, es decir, con un reconocimiento de caracteres del 100%, además el porcentaje de vehículos que transitaron y fueron detectados por el sistema pero no obtuvieron un reconocimiento idóneo. Finalmente se muestra la cantidad de autos que no fueron procesados el sistema.

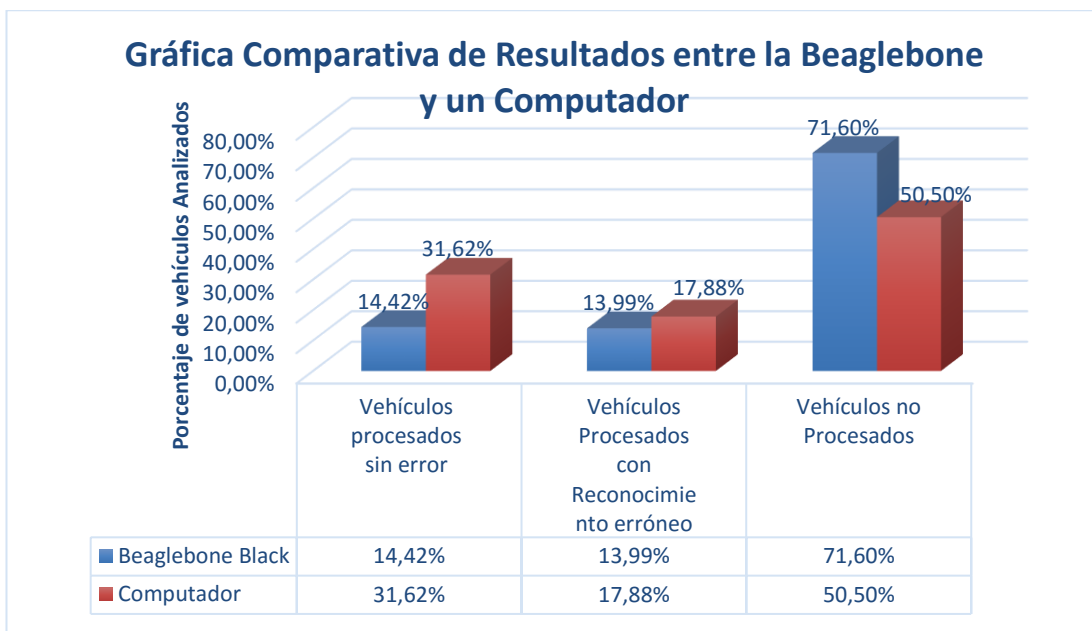


**Figura 69 Resultado Porcentual de Datos Obtenidos por Computador**

Los resultados expuestos anteriormente permiten reconocer que de 895 vehículos analizados, 283 presentan una efectividad del 100% en relación a la detección de sus caracteres que equivale al 32% de la cantidad de vehículos comprobados, por otro parte 160 vehículos que conforman el 18% del total de vehículos estudiados fueron detectados, sin embargo la obtención de los caracteres presentaba más de una letra fallida imposibilitando la identificación de un determinado vehículo, Finalmente los 452 vehículos que representa el 50% de los vehículos que transitaron no fueron reconocidos por el sistema, es decir, que software no adquirió la placa vehicular para efectuar el respectivo procesamiento de la imagen.

### 3.2.3. Análisis Comparativos entre Beaglebone y el Computador

En la figura 70 se muestran los resultados obtenidos mediante dos tipos de tecnologías de procesamiento de información, los cuales fueron obtenidos durante 3 días de pruebas, en los 2 primeros días se adquirieron los datos empleando la tarjeta Beaglebone Black, mientras que el tercer día se incorporó una computadora como unidad de procesamiento para realizar la ejecución del algoritmo.



**Figura 70 Gráfica Comparativa de Resultados entre la Beaglebone y un Computador**

Al observar la figura 70 se puede comprobar que el funcionamiento del sistema de reconocimiento de placas vehiculares en el computador tiene un mayor desempeño, al examinar el promedio general de vehículos procesados sin error, superando el rendimiento de la tarjeta Beaglebone Black con una diferencia del 17,2%. Por otro lado el porcentaje de vehículos procesados con reconocimiento erróneo presenta un mayor desempeño al emplear un computador incrementando el número de vehículos procesados en un 3,89% a diferencia de la tarjeta estudiada, evidenciando mejores resultados en la actividad de prueba realizada.

Para culminar el análisis se realizó la comparación del porcentaje de vehículos que no fueron procesados entre la tarjeta y el computador, obteniendo mayor cantidad de vehículos procesados con el ordenador alcanzando una diferencia del 21,10% en eficiencia, debido principalmente a su superioridad en recursos tanto de memoria como de procesamiento comparado con la tarjeta Beaglebone Black.

### 3.2.4. Pruebas Realizadas a Vehículos Específicos con una Tarjeta Beaglebone Black y un Computador

En esta sección se realizaron pruebas a 14 vehículos que fueron ingresados en una base de datos del sistema de reconocimiento de placas vehiculares que se puede visualizar en la figura 71 donde se comparó el rendimiento del algoritmo en la tarjeta Beaglebone Black Rev C y un computador Acer serie E5-471-54V8 con procesador core i5 a diferentes horas y lugares de la ciudad de Latacunga. En forma adicional se realizaron pruebas en la ciudad de Salcedo.



```
betorc@debian: ~
Archivo Editar Pestañas Ayuda
Query OK, 1 row affected (0.00 sec)

mysql> select * from placas;
+-----+-----+-----+-----+
| placa | marca | color | propietario |
+-----+-----+-----+-----+
| TBD-7443 | MAZDA | BLANCO | JUAN VEGA |
| POG-209 | CHEVROLET | VINO | PATRICIO VARGAS |
| PVI-907 | CHEVROLET | PLOMO | GUSTAVO GAVILANEZ |
| PXX-372 | VOLKSWAGEN | PLOMO | DAVID PACHECO |
| PPG-219 | NISSAN | AZUL | OSWALDO PACHECO |
| TDP-131 | KIA | VINO | KEVIN BEDON |
| PBO-6430 | NISSAN | ROJO | MARIO PACHECO |
| GOK-545 | FORD | DORADO | JORGE GARZON |
| PYN-526 | CHEVROLET | PLOMO | DANNY HERRERA |
| POI-214 | SKODA | BLANCO | SEBASTIAN PANCHI |
| XBW-289 | TOYOTA | ROJO | FRANCISCO PANCHI |
| POJ-626 | NISSAN | PLOMO | ITALO ROSERO |
| XBB-1060 | CHEVROLET | NEGRO | JONATHAN PACHECO |
| XBU-326 | CHEVROLET | VINO | GALO CHACON |
+-----+-----+-----+-----+
```

Figura 71 Vehículos Ingresados en la base de datos

**Tabla 10**


**Vehículo de Prueba N° 1**

<b>VEHÍCULO N° 1</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: JUAN VEGA		
PLACA: TBD-7443		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	4.163s	0.313s
Placa reconocida	SI	SI
Caracteres reconocidos	7	7
Porcentaje de reconocimiento de caracteres	100%	100%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Ninguna	Ninguna

En la tabla 10 se puede apreciar que el vehículo de placas TBD-7443 es expuesto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 4.163s y en el computador un tiempo de 0.313s, con un porcentaje de reconocimiento del 100% en ambos casos.

**Tabla 11**

**Vehículo de Prueba N° 2**

<b>VEHÍCULO N° 2</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: PATRICIO VARGAS		
PLACA: POG-209		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	2.623s	0.062s
Placa reconocida	SI	SI
Caracteres reconocidos	5	5
Porcentaje de reconocimiento de caracteres	83.33%	83.33%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible, Error una letra	Placa legible, Error una letra

En la tabla 11 se puede apreciar que el vehículo N°2 de placas POG-209 es sometido a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 2.623s y en el computador un tiempo de 0.062s, con un porcentaje de reconocimiento del 83,33% en ambos casos.



Tabla 12

Vehículo de Prueba N° 3

VEHÍCULO N° 3		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: GUSTAVO GAVILÁNEZ		
PLACA: PVI-907		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE</b>	<b>COMPUTADOR</b>
	<b>BLACK</b>	<b>PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	2.942s	0.495s
Placa reconocida	SI	SI
Caracteres reconocidos	6	6
Porcentaje de reconocimiento de caracteres	100%	100%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Ninguna	Ninguna

En la tabla 12 se puede apreciar que el vehículo N°3 de placas PVI-907 es sujeto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 2.942s y en el computador un tiempo de 0.495s, con un porcentaje de reconocimiento del 100% en ambos casos.

**Tabla 13**


**Vehículo de Prueba N° 4**

<b>VEHÍCULO N° 4</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: DAVID PACHECO		
PLACA: PXX-372		
 <p>The screenshot shows a video player with a car and its license plate. To the right, a 'Placa Detectada' box displays 'PXX-372' and 'PIX-372 T:0.0218s'. Below the video, there are controls for 'Fecha: 21/7/2016', 'Hora: 13:16:20', and 'Abrir Registro'. On the right side of the interface, there is a list of vehicle details: 'Placa: PXX-372', 'Modelo: VOLKSWAGEN', 'Color: PLOMO', 'Propietario: DAVID PACHECO', 'Similitud: 83.33%', and 'Vehiculo Reportado'.</p>		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	3.749s	0.021s
Placa reconocida	SI	SI
Caracteres reconocidos	5	5
Porcentaje de reconocimiento de caracteres	83.33%	83.33%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible, Error una letra	Placa legible, Error una letra

En la tabla 13 se puede apreciar que el vehículo N°4 de placas PXX-372 es expuesto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 3.749s y en el computador un tiempo de 0.021s, con un porcentaje de reconocimiento del 83,33% en ambos casos.

**Tabla 14**


**Vehículo de Prueba N° 5**

<b>VEHÍCULO N° 5</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: OSWALDO PACHECO		
PLACA: PPG-219		
 <p>The screenshot shows a video player window titled 'Interfaz'. On the left, a video frame shows a vehicle with license plate PPG-219. On the right, a panel titled 'Placa Detectada' displays the text 'AAA-####T: ---s'. Below the video, there are controls for 'Fecha: 21/7/2016', 'Hora: 13:18:36', and a 'Play' button. A dropdown menu shows 'Abrir Registro' with '21', '7', and '2016' selected. On the right side of the panel, there are fields for 'Placa: ___-___', 'Modelo: -----', 'Color: -----', 'Propietario: -----', and 'Porcentaje: -----'.</p>		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	----	----
Placa reconocida	NO	NO
Caracteres reconocidos	0	0
Porcentaje de reconocimiento de caracteres	0%	0%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	No se reconoció la placa vehicular	No se reconoció la placa vehicular

En la tabla 14 se puede apreciar que el vehículo N°5 de placas PPG-219 es sujeto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en ambos casos la placa vehicular no fue identificada por el sistema debido a los accesorios que lleva el vehículo puesto a prueba.

**Tabla 15**

**Vehículo de Prueba N° 6**

<b>VEHÍCULO N° 6</b>		
<b>PRUEBA DE FUNCIONAMIENTO DEL SISTEMA</b>		
<b>PROPIETARIO: KEVIN BEDÓN</b>		
<b>PLACA: TDP-131</b>		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	----	0.025s
Placa reconocida	NO	SI
Caracteres reconocidos	0	1
Porcentaje de reconocimiento de caracteres	0%	16.67%
Velocidad máxima del vehículo	10 km/h	15 km/h
Observaciones	No se reconoció la placa vehicular	Mala extracción de caracteres de la imagen

En la tabla 15 se puede apreciar que el vehículo N°6 de placas TDP-131 es sometido a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el caso de la tarjeta Beaglebone Black no se pudo identificar la placa vehicular, por otro parte el computador identificó la imagen de la placa pero la extracción del algoritmo OCR Tesseract no fue idóneo con un tiempo de procesamiento de 0.025s.

**Tabla 16**

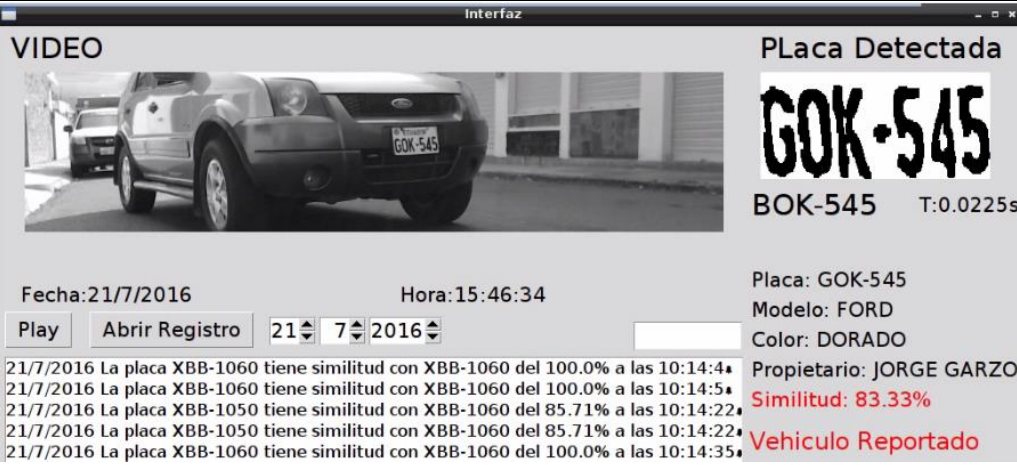
**Vehículo de Prueba N° 7**

<b>VEHÍCULO N° 7</b>		
<b>PRUEBA DE FUNCIONAMIENTO DEL SISTEMA</b>		
<b>PROPIETARIO: MARIO PACHECO</b>		
<b>PLACA: PBO-6430</b>		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	3.693s	0.078s
Placa reconocida	SI	SI
Caracteres reconocidos	6	6
Porcentaje de reconocimiento de caracteres	85.71%	85.71%
Velocidad máxima del vehículo	0 km/h	0 km/h
Observaciones	Placa legible, Error una letra	Placa legible, Error una letra

En la tabla 16 se puede apreciar que el vehículo N°7 de placas PBO-6430 es expuesto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 3.693s y en el computador un tiempo de 0.078s, con un porcentaje de reconocimiento del 85,71% en ambos casos.

Tabla 17

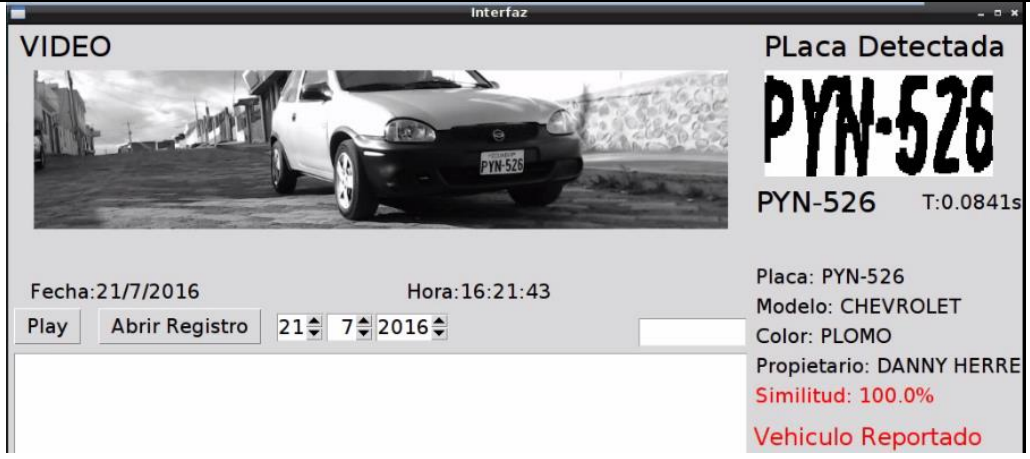
Vehículo de Prueba N° 8

VEHÍCULO N° 8		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: JORGE GARZÓN		
PLACA: GOK-545		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	2.834s	0.022s
Placa reconocida	SI	SI
Caracteres reconocidos	5	5
Porcentaje de reconocimiento de caracteres	83.33%	83.33%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible, Error una letra	Placa legible, Error una letra

En la tabla 17 se puede apreciar que el vehículo N°8 de placas GOK-545 es sujeto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 2,834s y en el computador un tiempo de 0.022s, con un porcentaje de reconocimiento del 83,33% en ambos casos.

**Tabla 18**

**Vehículo de Prueba N° 9**

<b>VEHÍCULO N° 9</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: DANNY HERRERA		
PLACA: PYN-526		
 <p>The screenshot shows a video player with a car and its license plate 'PYN-526' detected. The interface includes a 'VIDEO' label, a 'Placa Detectada' section with the license plate number, and a list of vehicle details: Placa: PYN-526, Modelo: CHEVROLET, Color: PLOMO, Propietario: DANNY HERRE, Similitud: 100.0%, and Vehiculo Reportado. The date is 21/7/2016 and the time is 16:21:43.</p>		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	4.273s	0.084s
Placa reconocida	SI	SI
Caracteres reconocidos	6	6
Porcentaje de reconocimiento de caracteres	100%	100%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible	Placa legible

En la tabla 18 se puede apreciar que el vehículo N°9 de placas PYN-526 es sometido a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 4.273s y en el computador un tiempo de 0.084s, con un porcentaje de reconocimiento del 100% en ambos casos.



**Tabla 19**

**Vehículo de Prueba N° 10**

<b>VEHÍCULO N° 10</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: SEBASTIAN PANCHI		
PLACA: PYN-526		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	----	0.017s
Placa reconocida	NO	SI
Caracteres reconocidos	0	2
Porcentaje de reconocimiento de caracteres	0%	33.34%
Velocidad máxima del vehículo	10 km/h	15 km/h
Observaciones	No se reconoció la placa vehicular	Mala extracción de caracteres de la imagen

En la tabla 19 se puede apreciar que el vehículo N°10 de placas POI-214 es expuesto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el caso de la tarjeta Beaglebone Black no se pudo identificar la placa vehicular, por otra parte el computador identificó la imagen de la placa vehicular pero la extracción del algoritmo OCR Tesseract no fue idóneo con un tiempo de procesamiento de 0.017s.



**Tabla 20**


**Vehículo de Prueba N° 11**

<b>VEHÍCULO N° 11</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: FRANCISCO PANCHI		
PLACA: XBW-289		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	----	0.442s
Placa reconocida	NO	SI
Caracteres reconocidos	0	2
Porcentaje de reconocimiento de caracteres	0%	33.34%
Velocidad máxima del vehículo	10 km/h	15 km/h
Observaciones	Demasiada intensidad de luz	Demasiada intensidad de luz

En la tabla 20 se puede apreciar que el vehículo N°11 de placas XBW-289 es sujeto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el caso de la tarjeta Beaglebone Black no se pudo identificar la placa vehicular, por otra parte el computador identificó la imagen de la placa vehicular pero la extracción por parte del algoritmo OCR Tesseract no fue idóneo con un tiempo de procesamiento de 0.442s.

**Tabla 21**

**Vehículo de Prueba N° 12**

<b>VEHÍCULO N° 12</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: ITALO ROSERO		
PLACA: POJ-626		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	2.174s	0.39s
Placa reconocida	SI	SI
Caracteres reconocidos	6	6
Porcentaje de reconocimiento de caracteres	100%	100%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible	Placa legible

En la tabla 21 se puede apreciar que el vehículo N°12 de placas POJ-626 es sometido a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 2.174s y en el computador un tiempo de 0.39s, con un porcentaje de reconocimiento del 100% en ambos casos.

**Tabla 22**

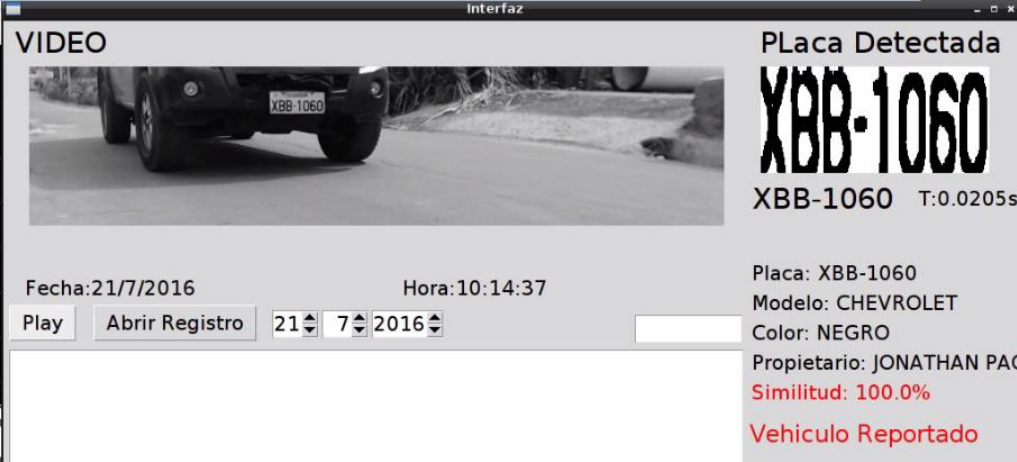
**Vehículo de Prueba N° 13**

<b>VEHÍCULO N° 13</b>		
<b>PRUEBA DE FUNCIONAMIENTO DEL SISTEMA</b>		
<b>PROPIETARIO: GALO CHACÓN</b>		
<b>PLACA: XBU-326</b>		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	4.239s	0.019s
Placa reconocida	SI	SI
Caracteres reconocidos	6	6
Porcentaje de reconocimiento de caracteres	100%	100%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible	Placa legible

En la tabla 22 se puede apreciar que el vehículo N°13 de placas XBU-326 es expuesto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 4.239s y en el computador un tiempo de 0.019s, con un porcentaje de reconocimiento del 100% en ambos casos.

**Tabla 23**

**Vehículo de Prueba N° 14**

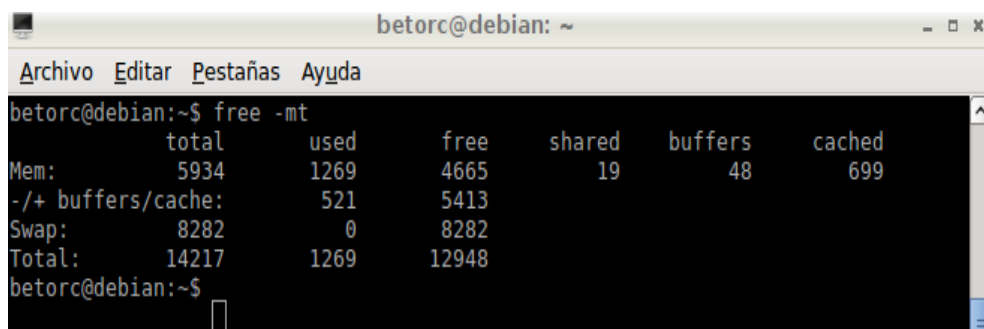
<b>VEHÍCULO N° 14</b>		
PRUEBA DE FUNCIONAMIENTO DEL SISTEMA		
PROPIETARIO: JONATHAN PACHECO		
PLACA: XBB-1060		
		
<b>ESPECIFICACIONES</b>	<b>BEAGLEBONE BLACK</b>	<b>COMPUTADOR PORTÁTIL</b>
Altura de la cámara	70cm	70cm
Tiempo de procesamiento	5.139s	0.020s
Placa reconocida	SI	SI
Caracteres reconocidos	7	7
Porcentaje de reconocimiento de caracteres	100%	100%
Velocidad máxima del vehículo	10 km/h	30 km/h
Observaciones	Placa legible	Placa legible

En la tabla 23 se puede apreciar que el vehículo N°13 de placas XBB-1060 es sujeto a pruebas del sistema de reconocimiento de placas vehiculares empleando diferentes tecnologías con el mismo algoritmo, obteniendo los resultados en el tiempo de procesamiento de la tarjeta Beaglebone Black de 5.139 s y en el computador un tiempo de 0.020s, con un porcentaje de reconocimiento del 100% en ambos casos.

### 3.2.5. Análisis de los Resultados Obtenidos en las Diferente Pruebas

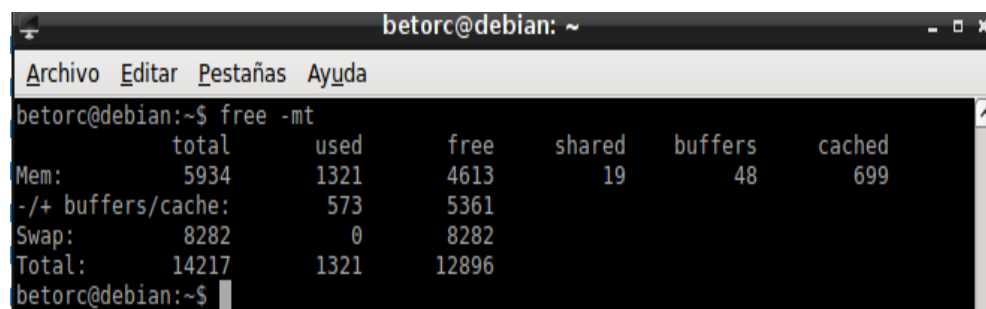
#### 3.2.5.1. Consumo de Memoria

El consumo de los recursos de memoria al momento de ejecutar el algoritmo en la computadora no presenta mayores cambios al realizar el procesamiento respecto a la cantidad de memoria utilizada, en la figura 72 se presenta el comportamiento de la memoria sin efectuar ninguna operación, mientras en la figura 73 se puede notar claramente un pequeño incremento en el uso de los recursos de memoria que pasaron de 1270 MB a más de 1320 MB, es decir, para la ejecución del algoritmo el computador empleó alrededor de 52 MB, que representa cerca del 1,2% de la memoria total con la que cuenta el sistema, por ende la ejecución del algoritmo en el computador se desarrolla sin ningún tipo problema en cuanto a procesamiento de la información.



```
betorc@debian: ~  
Archivo Editar Pestañas Ayuda  
betorc@debian:~$ free -mt  
              total        used         free       shared    buffers     cached  
Mem:           5934         1269        4665           19          48         699  
-/+ buffers/cache:         521        5413  
Swap:          8282           0         8282  
Total:        14217         1269       12948  
betorc@debian:~$
```

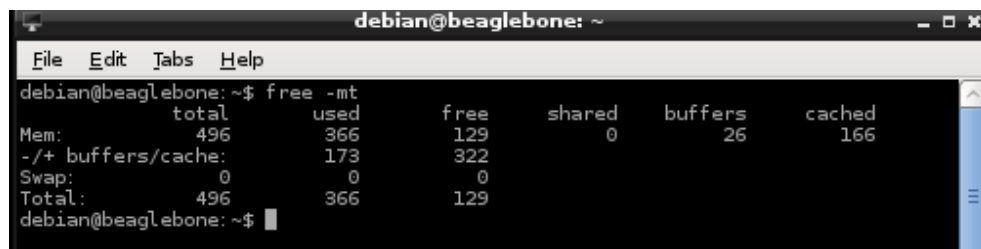
Figura 72 Consumo de Memoria del Computador Antes de Ejecutar el Programa de Reconocimiento de Placas Vehiculares



```
betorc@debian: ~  
Archivo Editar Pestañas Ayuda  
betorc@debian:~$ free -mt  
              total        used         free       shared    buffers     cached  
Mem:           5934         1321        4613           19          48         699  
-/+ buffers/cache:         573        5361  
Swap:          8282           0         8282  
Total:        14217         1321       12896  
betorc@debian:~$
```

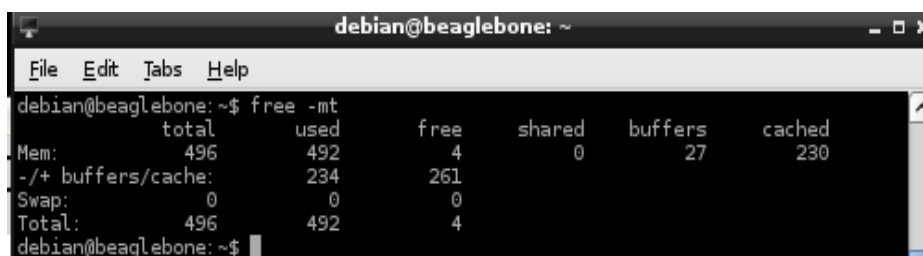
Figura 73 Consumo de Memoria del Computador Después de Ejecutar el Programa de Reconocimiento de Placas Vehiculares

Por otro lado el procesamiento de la información en la tarjeta Beaglebone Black, presenta limitados recursos en cuanto a memoria y procesamiento, como se presenta en la figura 74 al iniciar la tarjeta la memoria emplea 366 MB de los 496 disponibles, es decir, que solo en el arranque del sistema operativo se consume casi el 73.7% de la memoria RAM disponible, dejando 129 MB libres para ser utilizados, por otro lado la figura 75 presenta el consumo de memoria cuando el algoritmo se encuentra en ejecución y claramente se visualiza que la memoria usada es ahora de 492 MB, es decir que se incrementó en 126 MB, dejando solamente 4 MB libre, lo que quiere decir que la memoria se encuentra casi en su totalidad saturada al momento de ejecutar el algoritmo de reconocimiento, llevando así al calentamiento de la tarjeta y obligando a suspender su funcionamiento, con un tiempo de operación continuo alrededor de 2 horas a máxima capacidad, una vez superado este tiempo se deberá esperar un lapso de tiempo para poner en marcha la tarjeta.



```
debian@beaglebone: ~$ free -mt
total          used         free   shared  buffers   cached
Mem:           496          366         129        0         26        166
-/+ buffers/cache: 173          322
Swap:           0           0           0
Total:          496          366         129
debian@beaglebone: ~$
```

**Figura 74 Consumo de Memoria de la Tarjeta Antes de Ejecutar el Programa de Reconocimiento de Placas Vehiculares**



```
debian@beaglebone: ~$ free -mt
total          used         free   shared  buffers   cached
Mem:           496          492           4        0         27        230
-/+ buffers/cache: 234          261
Swap:           0           0           0
Total:          496          492           4
debian@beaglebone: ~$
```

**Figura 75 Consumo de Memoria de la tarjeta Después de Ejecutar el Programa de Reconocimiento de Placas Vehiculares**

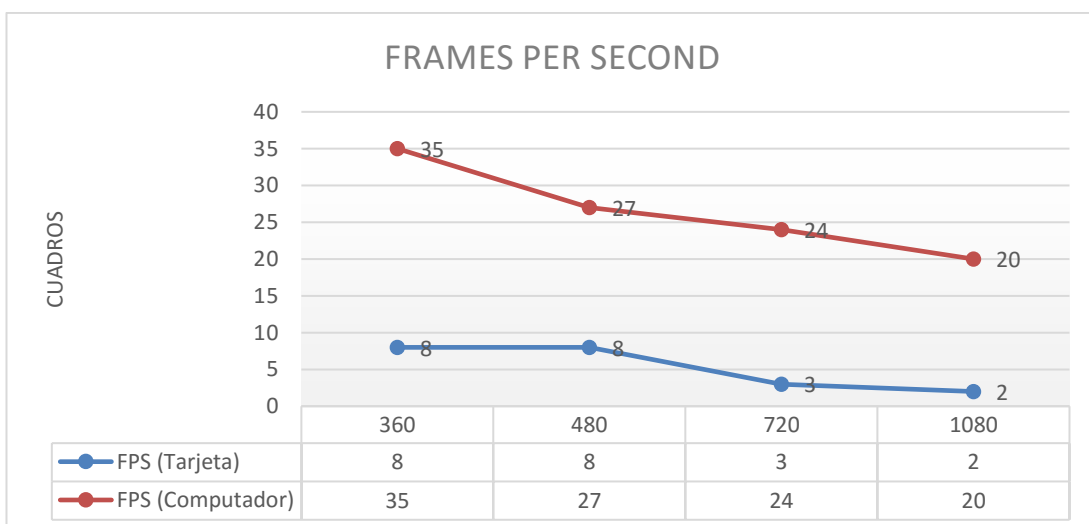
En cuanto a los tiempos de procesamiento de la tarjeta Beaglebone Black con respecto al computador, son muy superiores como se puede apreciar en la tabla 24, obligando a la tarjeta a no procesar todos los vehículos que se encuentran de manera consecutiva ya que se pierde información con cada FPS (Frames per second) que no procesa la tarjeta.

**Tabla 24**

**Tiempos de Procesamiento entre Sistemas**

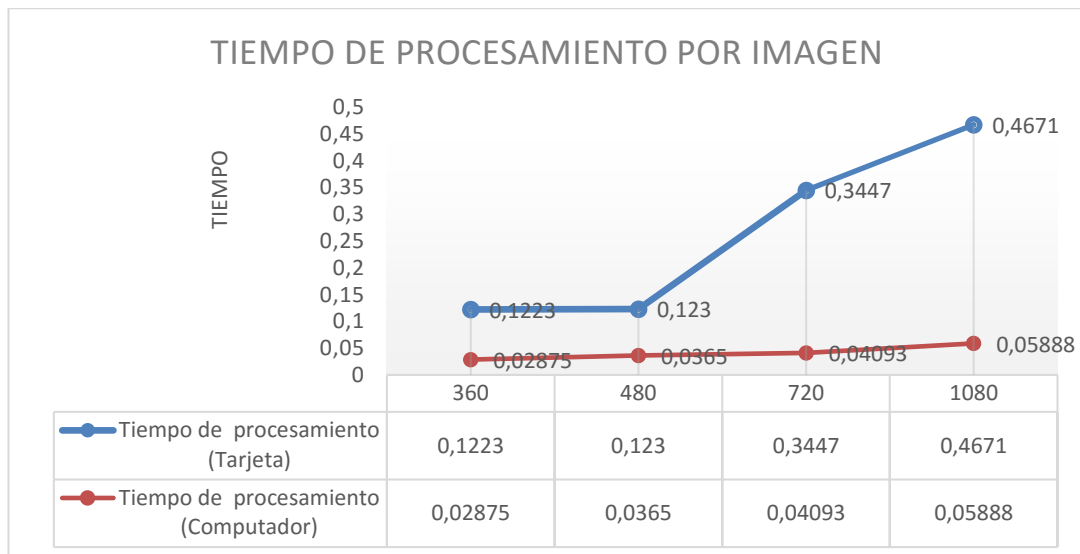
Resolución de la imagen	Beaglebone Black		Computador	
	FPS	Tiempo de procesamiento	FPS	Tiempo de procesamiento
<b>360</b>	8	0,1223	35	0,02875
<b>480</b>	8	0,1230	27	0,03650
<b>720</b>	3	0,3447	24	0,04093
<b>1080</b>	2	0,4671	20	0,05888

La figura 76 presenta el número de cuadros (capturas de imagen) por segundo que permiten realizar tanto la tarjeta Beaglebone Black como el computador dependiendo específicamente de la resolución de la imagen a procesar.



**Figura 76 FPS por Resolución de Imagen con Diferentes Tecnologías**

El tiempo de procesamiento de una imagen en la tarjeta Beaglebone Black es relativamente alto en relación al tiempo que emplea un computador en realizar la misma acción como se puede apreciar en la figura 77, para cada resolución evaluada.



**Figura 77 Tiempo de Procesamiento de Imagen por su Resolución con Diferentes Tecnologías**

Para el desarrollo del algoritmo en la tarjeta Beaglebone se optó por trabajar con la capturas de 480x864 pixeles ya que posee mayor cantidad de información, dando a la imagen mayor realce, manteniendo un nivel de pixel legible para algoritmos de reconocimiento y manteniendo un nivel de procesamiento aceptable para la operación de la tarjeta.

Con capturas de imagen superiores a 480 pixeles la velocidad de procesamiento disminuye provocando gran pérdida de información en cuanto a videos en tiempo real, por otro lado los recursos del computador son suficientes para cumplir con los requerimientos del sistema para trabajar con imágenes de alta resolución y por consiguiente la tarjeta Beaglebone no es adecuada para la aplicación en proyectos de visión artificial con una alta demanda de información, es decir, procesos que manejen imágenes con una alta resolución y velocidad de respuesta



### 3.3. Verificación de la hipótesis

La hipótesis planteada para esta tesis de grado fue la siguiente:

“La implementación del sistema para detección de vehículos robados en movimiento, empleando tecnología Beaglebone por medio de software libre (OpenCV), permitirá identificar la placa vehicular de un automotor”.

Al concluir el presente trabajo de tesis se ha alcanzado los siguientes resultados que permiten verificar la hipótesis planteada empleando el método estadístico Chi-Cuadrado:

**Tabla 25**

**Datos obtenidos de las Pruebas Realizadas con la Tarjeta Beaglebone Black RevC**

Beaglebone Black Rev C				
Pruebas	Vehículos procesados sin error	Vehículos procesados con reconocimiento erróneo	Vehículos no procesados	Total
Día 1	119	109	560	788
Día 2	114	117	597	828
<b>Total</b>	233	226	1157	1616

Para la comprobación de la hipótesis se utilizó el método Chi-Cuadrado con un margen de error de 0.05, el cual requiere generar una Hipótesis Nula H0 y una Hipótesis Alternativa H1 que son mostradas a continuación.

H0= ¿No se puede identificar la placa vehicular de un automotor empleando tecnología Beaglebone?

H1= ¿Si se puede identificar la placa vehicular de un automotor empleando tecnología Beaglebone?

El cálculo de la frecuencia esperada se lo realizó de la siguiente manera

Donde:

A= Vehículos procesados sin error

B= Vehículos procesados con reconocimiento erróneo

C= Vehículos no procesados

**Tabla 26**

**Modelo de tabla de Frecuencias Esperadas**

Tabla de Frecuencias Esperadas				
Pruebas	A	B	C	Total
Día 1	Dato1	Dato3	Dato5	
Día 2	Dato2	Dato4	Dato6	
Total				Total

Donde:

$$\text{Dato1} = (\text{Total A} \times \text{Total Día1}) / (\text{Total})$$

$$\text{Dato2} = (\text{Total A} \times \text{Total Día2}) / (\text{Total})$$

$$\text{Dato3} = (\text{Total B} \times \text{Total Día1}) / (\text{Total})$$

$$\text{Dato4} = (\text{Total B} \times \text{Total Día2}) / (\text{Total})$$

$$\text{Dato5} = (\text{Total C} \times \text{Total Día1}) / (\text{Total})$$

$$\text{Dato6} = (\text{Total C} \times \text{Total Día2}) / (\text{Total})$$

Obteniendo la tabla 27 de frecuencias esperadas

**Tabla 27**

**Valores Calculados de Frecuencias Esperadas**

Tabla de Frecuencias Esperadas			
Pruebas	Vehículos procesados sin error	Vehículos procesados con reconocimiento erróneo	Vehículos no procesados
Día 1	113,6163366	110,20297	564,180693
Día 2	119,3836634	115,79703	592,819307

Una vez obtenidos los valores de frecuencia esperada se procedió a utilizar el programa Excel el cual cuenta con la función "Prueba.chicquad" permitiendo obtener el valor de la distribución Chi-Cuadrado que posibilita determinar si un experimento se ajusta a los resultados teóricos.

$$p=\text{PRUEBA.CHICUAD}(\text{rango\_real}; \text{rango\_esperado})$$
$$p= 0,746778521$$

Posteriormente se procedió a calcular los grados de libertad y el valor Chi-Cuadrado-Inverso con la función "prueba.chi.inv".

$$V=\text{grados de libertad.}$$
$$V=(\text{nfilas} - 1) \times (\text{ncolumnas} - 1)$$
$$V=(2 - 1) \times (3 - 1).$$
$$V=2.$$
$$x^2_p=\text{PRUEBA.CHI.INV}(\text{probabilidad}; \text{grados\_de\_libertad})$$
$$x^2_p=0,583973258$$

Finalmente se comparó el valor Chi-Cuadrado calculado  $x^2_p = 0,583973258$ , con el valor obtenido de la tabla de distribución Chi-Cuadrado con dos grados de libertad y un margen de error de 0.05 se obtuvo un valor Chi-Cuadrado tabla  $x^2_t = 0.59915$ , permitiendo determinar si la hipótesis planteada es o no verdadera.

- Si el valor de Chi-Cuadrado calculado  $x^2_p > \text{Chi-Cuadrado tabla } x^2_t$  se puede declarar que la  $H_0$  es inválida, obteniendo como resultado  $H_1$
- Si el valor de Chi-Cuadrado calculado  $x^2_p < \text{Chi-Cuadrado tabla } x^2_t$  se puede declarar que la  $H_1$  es inválida, obteniendo como resultado la  $H_0$

$$x^2_p = 0,583973258 < x^2_t = 0.59915,$$

Al analizar los datos obtenidos y compararlos con los criterios anteriormente mencionados se puede concluir que no se puede identificar la placa vehicular de un automotor empleando tecnología Beaglebone.

## **CAPÍTULO IV**

### **4. CONCLUSIONES Y RECOMENDACIONES**

#### **4.1. Conclusiones**

Debian es el sistema operativo más eficiente al momento de manejar una Tarjeta Beaglebone Black, debido a su ligereza y gran compatibilidad entre arquitecturas de PC, incrementando el rendimiento del software y hardware.

El algoritmo desarrollado para la detección de vehículos robados en movimiento cumple con el requerimiento del dispositivo, siempre y cuando no exista gran afluencia vehicular debido al tiempo de procesamiento que le toma al sistema en procesar la información, de no ser así, el sistema pierde información cuando esta es procesada.

El consumo de memoria al ejecutar una aplicación en la tarjeta Beaglebone Black, es muy elevado en relación a la capacidad total de memoria RAM disponible en el dispositivo, tal es el caso que solo en el arranque del sistema operativo de la tarjeta utiliza cerca del 70% (366MB) de la memoria, dejando un valor no superior a los 130MB a disposición del usuario, cabe mencionar que al emplear algoritmos que procesen una alta tasa de información la tarjeta tiende a saturar su memoria rápidamente ocasionado pérdidas de información y retrasos en su tiempo de ejecución.

El tiempo de procesamiento de una imagen en una tarjeta Beaglebone, depende específicamente de la resolución a la cual se trabaje, siendo la resolución de 480x864 pixeles la más adecuada, pues permite el manejo de 8 frames por segundo al igual que la resolución de 360p con la diferencia que en 480p existe mayor cantidad de información, por otro lado al emplear resoluciones superiores a los 480p el tiempo de procesamiento es mucho mayor disminuyendo el número de frames a manejar llegando a procesar entre 2 y 3 frames con un tiempo de procesado por imagen de

0,3447s y 0.4671s respectivamente, obteniendo como resultados imágenes con una alta tasa de detalle con un elevado tiempo de procesamiento, perdiendo información de vehículos en tiempo real mientras se procesa cada frame.

El uso prolongado de la tarjeta Beaglebone Black en conjunto con el algoritmo de detección de vehículos, produce un sobrecalentamiento en el hardware, ya que el sistema emplea todos sus recursos de memoria y procesamiento, provocando de esta manera un colapso de su sistema operativo obligándolo a suspender su funcionamiento en su totalidad.

El motor de busque OCR Tesseract permite convertir imágenes preseleccionadas por el sistemas de detección de vehículos captadas por la cámara web a texto, obteniendo como resultado una cadena tipo string con los caracteres presentes en la imagen de la placa vehicular.

La tarjeta Beaglebone Black no es adecuada para aplicaciones que empleen visión artificial, en donde los procesos requieran una gran cantidad de procesamiento de información como se pudo apreciar en el proyecto desarrollado, en el cual el tiempo de respuesta es muy grande para los objetivos planteados, además existen pocas cámaras que son compatibles con la tarjeta, debido a sus limitados recursos de memoria y procesamiento

## **4.2. Recomendaciones**

Se recomienda el uso de un computador, pero de no ser posible y requerirse de una tarjeta como la Beaglebone Black, se deberá utilizar una tarjeta micro SD mayor a 4Gb de almacenamiento, con una velocidad de 10MB/S para que la tarjeta Beaglebone Black pueda aprovechar la máxima velocidad de transferencia de datos mejorando su rendimiento.

Se recomienda verificar la compatibilidad de la cámara con el sistema operativo Debian, pues no toda cámara presenta una adecuada relación con el sistema operativo impidiendo una conexión exitosa entre la cámara y la tarjeta Beaglebone Black.

Utilizar dispositivos periféricos de entrada (mouse y teclado inalámbrico) para evitar la aglomeración de varios puertos de entrada en la tarjeta, ya que solo dispone de una entrada USB, por tal motivo para emplear varios dispositivos se requiere emplear un HUB-USB para dicha conexión, presentado de esta manera un mayor consumo de energía que en algunos casos no alcanza a abastecer a todos los dispositivos.

Para evitar daños en la tarjeta Beaglebone Black no es recomendable desconectar la fuente de alimentación directamente, sino más bien pulsar el botón de reseteo y esperar a que las luces led del dispositivo se apaguen siendo este el momento idóneo para desconectar la fuente de alimentación.

Es recomendable el uso de OpenCV para aplicaciones relacionadas a visión artificial, ya que cuenta con miles de algoritmos y funciones empaquetadas en librerías para el manejo, modificación, creación y visualización de imágenes disponibles para el uso del usuario, además de contar con un gran poder de compatibilidad con varios lenguajes de programación dentro de diversos sistemas operativos.

Para mejorar el rendimiento de la tarjeta Beaglebone Black RevC es aconsejable instalar únicamente los paquetes y librerías con las que se va a trabajar para evitar el uso innecesario de recursos de la tarjeta.

Para ejecutar el algoritmo desarrollado en Python, se lo realiza por medio de la consola LXTerminal especificando la dirección donde se

encuentra ubicado el programa, posteriormente se ingresa la palabra python seguido del nombre del archivo con su respectiva extensión “.py”, que contiene el algoritmo desarrollado

Es recomendable no poner la tarjeta Beaglebone Black RevC en superficies metálicas ya que esto puede generar un cortocircuito en sus terminales dañando totalmente la tarjeta.



## BIBLIOGRAFÍA

- Alvarez, M. A. (10 de Noviembre de 2003). *Desarrolloweb*. Obtenido de Desarrolloweb: <http://www.desarrolloweb.com/articulos/1325.php>
- Andrade Miranda Gustavo, L. E. (2011). Sistema de control vehicular utilizando reconocimiento óptico de caracteres. Guayaquil: Facultad de Ingeniería Eléctrica y Computación, Escuela Politécnica de Litoral.
- ARENAS, H. F. (2003). *LA BIBLIA DE LINUX*. BUENOS AIRES RGENTINA: MP EDICIONES.
- Astudillo, H. (2015). *EXTRACCIÓN Y EVALUACIÓN DE CARACTERÍSTICAS DE SECUENCIAS DE VIDEO UTILIZANDO MÉTODOS ORIENTADOS A DISPOSITIVOS DE BAJO PODER COMPUTACIONAL*. Guayaquil: Escuela Superior Politécnica del Litoral.
- Auchterberge, S. E. (2014). *Implementación y análisis de un algoritmo*. Universidad Nacional de Córdoba.
- Carla, C. (2015). *Sistema de semaforización inteligente para el control de flujo vehicular mediante el procesamiento digital de imágenes*. Ambato: Universidad Técnica de Amabato, Facultad de Ingeniería en sistemas electrónica e industrial.
- Cortez, F. (17 de 05 de 2014). *Programación para Estudiante*. Recuperado el 19 de 01 de 2016, de <http://cortesfernando.blogspot.com/2014/05/binarizacion-imagen.html>
- D. Wijetunge, C. R. (16 de 08 de 2011). *IEEEXPLORER*. Obtenido de Dept. of Stat. & Comput. Sci., Univ. of Peradeniya, Peradeniya, Sri Lanka : <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6038045&queryText=real-time%20recognition%20of%20license%20plates%20of%20vehicles%20in%20Sri%20Lanka&newsearch=true>
- documentation, O. 2. (21 de 04 de 2014). *OpenCV*. Obtenido de <http://docs.opencv.org/2.4.9/doc/tutorials/tutorials.html>
- documentation, O. (04 de 21 de 2014). *OpenCV*. Obtenido de OpenCV: [http://docs.opencv.org/2.4.9/modules/highgui/doc/reading\\_and\\_writing\\_images\\_and\\_video.html?highlight=videocapture#VideoCapture](http://docs.opencv.org/2.4.9/modules/highgui/doc/reading_and_writing_images_and_video.html?highlight=videocapture#VideoCapture)
- Ecuador, F. (07 de Mayo de 2013). *Foros Ecuador*. (Placas de Ecuador por Provincias) Recuperado el 18 de Enero de 2016, de <http://www.forosecuador.ec/forum/aficiones/autos-y-motos/212-placas-de-ecuador-por-provincias>
- Electronic, T. M. (s.f.). *Tech Make Electronic*. Recuperado el 19 de 01 de 2016, de <http://www.techmake.com/00120.html>
- IEEE Xplore. (11 de 02 de 2016). *IEEE Xplore Digital Library*. Obtenido de IEEE Xplore Digital Library:

<http://ieeexplore.ieee.org/search/searchresult.jsp?queryText=recognition%20of%20license%20plates&newsearch=true>

José María Herrera Fernández, L. M. (23 de 03 de 2015). *Computación científica con Python para módulos de evaluación continua en asignaturas de ciencias aplicadas*.

Obtenido de

[http://pendientedemigracion.ucm.es/info/aocg/python/modulos\\_cientificos/numpy/index.html](http://pendientedemigracion.ucm.es/info/aocg/python/modulos_cientificos/numpy/index.html)

Kasaei, S. H. (14 de Septiembre de 2011). *IEEEXPLORER*. Obtenido de Intelligence and Security Informatics Conference (EISIC),:

<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6061241&newsearch=true&queryText=%20Extraction%20and%20recognition%20of%20the%20license%20plate%20of%20the%20vehicle%20to%20pass%20under%20outdoor%20environment>

Kradac. (3 de 12 de 2015). *Kradac*. Recuperado el 19 de 01 de 2016, de

<http://www.kradac.com/fotomulta.html>

logitech. (16 de 02 de 2016). *logitech cámara c920*. Obtenido de logitech Web site:

<http://www.logitech.com/es-roam/product/hd-pro-webcam-c920>

Malaver, C. G. (2014). *Análisis Estructural por el Método de Elementos Finitos Asistido*. Universidad Nacional de Cajamarca.

Martínez, M. E. (2015). *Programación en C++ Utilizando OpenCV*. Huapalcalco, Tulancingo, Hgo., México: universidad politecnica de tulancingo.

Mendoza, I. G. (6 de Abril de 2012). *EL SOFTWARE LIBRE EN EL ECUADOR Lu 8-12*. Obtenido de EL SOFTWARE LIBRE EN EL ECUADOR Lu 8-12:

<http://andreitamedina.blogspot.com/>

Mora, L. A.-M.-Ó. (2013). *Bases de datos en MYSQL*. Catalunya: FIJOC.

Muñoz Manso, R. (2014). *Sistema de visión artificial para la*. Valladolid: UNIVERSIDAD DE VALLADOLID. Obtenido de Sistema de visión artificial para la.

MYSQL. (01 de 07 de 2016). *MYSQL*. Obtenido de MYSQL: <https://www.mysql.com/>

Normal, N. (2014). *BeagleBone Black*. Make.

OpenCV. (2015 de 12 de 13). *OpenCV*. (OpenCV) Recuperado el 2015 de 12 de 22, de <http://opencv.org/>

OpenCV. (10 de 10 de 2014). *OpenCV*. Obtenido de [http://docs.opencv.org/3.0-beta/modules/imgcodecs/doc/reading\\_and\\_writing\\_images.html](http://docs.opencv.org/3.0-beta/modules/imgcodecs/doc/reading_and_writing_images.html)

OpenCV. (10 de 10 de 2014). *OpenCV*. Obtenido de [http://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_gui/py\\_image\\_display/py\\_image\\_display.html](http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html)

Policia Judicial Cotopaxi. (2016). *Vehículos Reportados*. Latacunga: Destacamento Policial.

Python. (s.f.). *Python*. Obtenido de Python: <https://www.python.org/>

RODRÍGUEZ YAGUAL CRISTHIAN ANTONIO, T. G. (03 de 11 de 2013).

<http://repositorio.upse.edu.ec/>. Obtenido de UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA:

<http://repositorio.upse.edu.ec/xmlui/bitstream/handle/46000/1604/IMPLEMENTACION%20DEL%20SISTEMA%20DE%20REGISTRO%20AUTOMATICO%20DE%20LAS%20PLACAS%20VEHICULARES%20UTILIZANDO%20RECONOCIMIENTO%20COMPUTACIONAL%20DE%20CARACTERES%20Y%20VISION%20ARTIFICIAL>

Terán, L. A. (2009). *Análisis de Opciones Reales utilizando Python*. Cumbaya: UNIVERSIDAD SAN FRANCISCO DE QUITO.

TERRESTRE, L. O. (22 de 09 de 2012). *ecuador-vial.com*. (LEY ORGÁNICA DE TRANSPORTE TERRESTRE) Recuperado el 18 de 01 de 2016, de <http://www.ecuador-vial.com/wp-content/uploads/2012/09/ECUADOR-CON-NUEVAS-PLACAS-PARA-LOS-VEHICULOS.pdf>

Vélez, P. X. (septiembre de 2014). Desarrollo de un sistema Didactico Aplicado a Inversores trifásicos, empleano el sistema embebido raspberry PI, mediante la técnic de modulación vectorial espacial. Cuenca, Azuay, Ecuador.

Wang, F., Zhang, D., Man, L., & Yu, J. (16 de Noviembre de 2010). *IEEEXPLORER*. Obtenido de Intelligent Systems and Knowledge Engineering (ISKE): <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5680872&queryText=Learning%20algorithm%20for%20color%20recognition%20of%20license%20plates&newsearch=true>

## Anexo



**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA


**DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA**  
**CARRERA DE INGENIERÍA ELECTRÓNICA E**  
**INSTRUMENTACIÓN.**

**CERTIFICACIÓN**

Se certifica que el presente trabajo fue desarrollado por el señor **ROBERTO CARLOS GARZÓN CANCHIGNIA** y el señor **JONATHAN ALEXANDER PACHECO GAVILÁNEZ**.

En la ciudad de Latacunga a los 17 días del mes de agosto del 2016


Aprobado por

  
Ing. Eddie Galarza Zambrano

**DIRECTOR DEL PROYECTO**

  
Ing. Franklin Silva

**DIRECTOR DE LA CARRERA**

  
Dr. Rodrigo Vaca Corrales  
**SECRETARIO ACADÉMICO**