Traffic Violation Detection and Notification System

By

Muhammad Hamza Cheema

Samar Ali Khan

Ahmad Usman

Submitted to the Faculty of Electrical Engineering, Military College of Signals, National University of Sciences and Technology in fulfillment for the requirements of a B.E Degree in Telecommunication Engineering

JUNE 2015

ABSTRACT

Traffic violations are a serious problem now days; they make our roads a danger to not only the violators themselves, but also to anyone around them. Recent surveys show a rapid increase in road accidents leading to loss of hundreds of valuable lives.

We have tried to solve this problem by developing a Traffic Violation Detection and Notification System which will monitor the traffic 24/7. The system will measure the speed of the moving vehicles, in case of Over-speeding, image of the violating vehicle will be captured, and violator will then be notified using Short-Service Message (SMS) and record of the driver will be maintained at the database until the fine has been paid.

This report contains description about the project along with design details; both software and hardware, project analysis and evaluation, results of rigorous testing and their analysis, problems faced while development of the project, a demonstration outline and recommendations for future work.

CERTIFICATE

It is certified that work contained in this thesis **“Traffic Violation Detection and Notification”** was carried out by **Samar Ali Khan**, **Ahmad Usman** and **Muhammad Hamza Cheema** under the supervision of **Lt Col Dr. Adil Masood Siddique** for partial fulfillment of Degree of Bachelors of Telecommunication Engineering, is correct and approved.

Supervisor:
Lt Col Dr. Adil Masood Siddiqui
HOD, CS Dept.
MCS, NUST

____

DECLARATION

We hereby declare that no content of work presented in this thesis has been submitted in support of another award of qualification or degree either in this institution or anywhere else.

COPYRIGTHS

DEDICATED TO

Almighty Allah,

Faculty for their help

And our parents for their support

ACKNOWLEDGEMENT

Nothing happens without the will of Allah Almighty. We thank Allah Almighty for giving us knowledge and strength to accomplish this task successfully.

We would also like to thank our project supervisor Lt Col Dr. Adil Masood Siddiqui whose constant support and guidance made it possible for us to complete this project.

We would like to express our special gratitude and thanks to Dr. Muqeem Sherry, Asst Prof. Attiya Obaid and Lec. Aimen Aakif for extending help to us during our hour of need.

We also thanks and appreciate to our colleague for helping in developing the project and people who have willingly helped us with their abilities.

Last but not the least, we are very thankful to our parents, who bore with us in times of difficulty and hardship. Without their consistent support and encouragement, we could not have accomplished our targets successfully.

PREFACE

This dissertation is an original intellectual product of the authors Samar Ali Khan, Ahmad Usman and Muhammad Hamza Cheema. None of the text of the dissertation is taken directly from previously published or collaborative articles.

Samar Ali Khan was responsible for defining and finalizing the algorithms and approach mentioned in Chapter 3. Hamza Cheema completed the hardware design and integration which is explained in Chapter 3, while Ahmad Usman completed the Database and Notification part.

Defining feature of any new technology is how well it holds up against the existing technology which it tries to replace. After the completion of this project is was rigorously tested within MCS as well as outside of MCS. This project was tested with the Motorways Police, where the results were compared with the equipment they are currently using. The results were more than satisfactory. An appreciation letter the motorway police has been attached in appendix.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SMS:	Short Message Service
MATLAB:	Matrix Laboratory
RGB:	Red Green Blue
PN:	Part Number
RAM:	Random Access Memory
USB:	Universal Serial Bus
HD:	High Definition
CRT:	Cathode Ray Tube
eMMC:	Embedded Multimedia Card
HDMI:	High-Definition Multimedia Interface
I/O:	Input/output
API:	Application Programming Interface
RF:	Radio Frequency
TBD:	To be Defined
PMIC:	Power Management Integrated Circuit

Chapter 1

INTRODUCTION

Traffic violation is a serious problem that we are facing now days; our roads are becoming more dangerous not only to the violators themselves, but also to anyone around them. Recent surveys show a rapid increase in road accidents leading to loss of thousands of valuable lives worldwide. We have tried to address this problem by developing a Traffic Violation Detection and Notification System which will monitor traffic 24/7.

Transportation is one of the vital components of today's society. It is the key to economic growth and development. Due to recent increase in traffic volume, unfortunately rate of road accidents has increased; therefore research and studies are being carried out worldwide to enforce traffic regulation so that roads can be made safer for everyone.

1.1 Background/Motivation

Road accidents have become a serious problem of modern times. Over 40% of the accidents are caused due to driving over the speed limit. In this project we focus on developing a system that will not only detect the violation but also capture an image of the violating car, notify the violator and maintain record at the database until fine is paid.

1.2 Problem Statement

To control this problem several technologies have been developed and implemented. Radar technology is most commonly used for this purpose. This technology has been used for a long time however there are a lot of loop holes in this system that needs to be addressed. Due to development of jammers accuracy of Radars has decreased. Since Radar can focus on a single vehicle at a time, therefore when traffic is too much the system is not very effective. Another major problem that this system has is that it always need a person to operate it. Accuracy of Radar speed traps deteriorates highly with bad weather condition.

To overcome these problems we have used a camera to detect speed. This system is able to detect, track and calculate speed of multiple objects simultaneously to cope up with the increased traffic. This system can monitor traffic without constant supervision.

1.3 Project Description

To detect traffic violation a camera is used. Live stream from the camera is fed to the processing unit where each moving vehicle will be detected, tracked and speed of each vehicle will be calculated. Once the speed is calculated it is compared against the set speed limit and if the speed is above the speed limit; then that vehicle is over-speeding. From that captured image of violating car, number plate is detected and that registration number of the vehicle is extracted. That registration number is used to get information of the owner. On that contact information a message through SMS will be send notifying the user about details of violation and fine.

This project is most suitable for motorways and highways where roads are straight and chances of over-speeding are more. Although this system can be deployed within cities however to achieve best possible results it should be used over straight stretches of road.

1.4 Salient Features

This is an application based project that can be developed to make a system that can work in real world scenario.

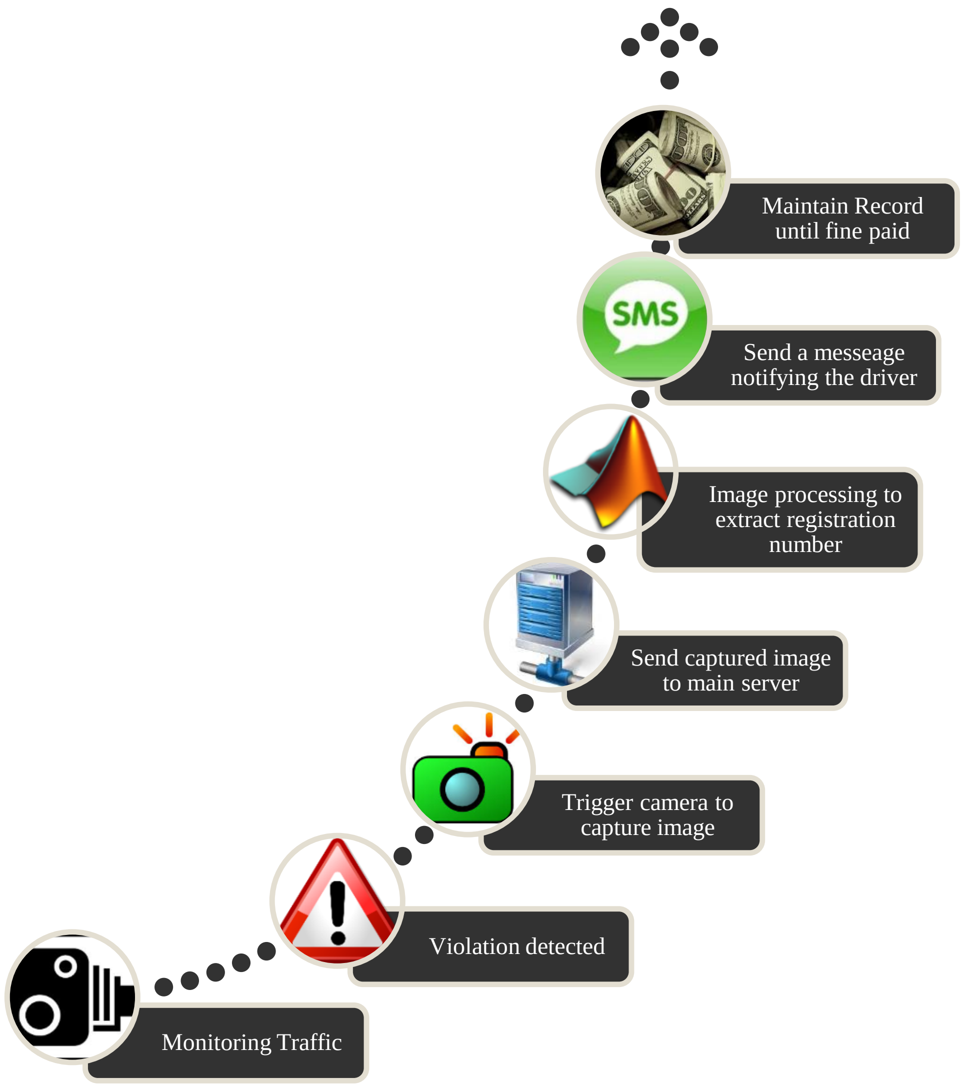
- This project is set to work for real time scenario, therefore real-time processing is really important. This project capture frames, detect and track moving vehicle and calculate speed all in real-time

- Selecting the resolution of the images is also very important. Pixel density of the frames is neither too high due to which efficiency is compromised, nor it is so low that accuracy is affected.

- Algorithm working to detect, track and calculate speed of the moving vehicle is efficient, precise and accurate so that any error in calculation of speed are minimized.

Database needs to maintain a constant record of the violators until dues are paid.

1.5 System Model


Monitoring Traffic
Violation detected
Trigger camera to capture image
Send captured image to main server
Image processing to extract registration number
SMS
Send a messeage notifying the driver
Maintain Record until fine paid

1.6 Scope, Objectives and Specifications

- This project is an application based project. Efficiency, accuracy and precision are the standards on bases of which these project was developed. These three parameters hold great importance when it comes to calculating speed if a moving vehicle.
- Scope of this project is to be as efficient as possible, so that no vehicle is missed. For this the system works in real time scenario. For the system to work in real time first live video feed is fed to the processor, secondly processing speed is enough to meet the demand of incoming data.
- Calibration of camera will has a major impact on the value of speed; since value of distance is estimated with the help of calibration. Calibration affects accuracy of the system, therefore algorithm for calibration is made to be as accurate as possible.
- For this project to be precise the actual algorithm for detection, tracking and calculating speed of vehicle is very important.
- Keeping these specifications in mind the hardware is designed. The processing board is efficient enough to handle the incoming data. While deciding camera we needed to find the best case scenario. Camera needed to have good enough resolution and frame rate that the video feed is clear and that there is minimum noise, but we also kept in mind that higher resolution and frame rate means more data size which means more processing which will cause delay, hence slowing down the system.

1.7 Deliverables

This project will try to meet the specifications mentioned above. The final form of project consists of a camera interfaced with the Beagle-bone Black board. System works in real time with minimum time lag. This project will detect a moving vehicle, track it and use the tracking information to calculate speed of the moving car. Each car will be assigned a tag in order to identify it, and then image of the violating car will be taken to extract registration number of the violating car. Once the registration number has been identified it will be used to get the contact information of the owner of that car and a message will be send on the contact number of the owner, informing him about his violation. Until the fine is paid his record will be maintained at the database.

The final form of the project consists of a camera by Logitech (B525) interfaced with a processing board; Beagle-bone Black, a PC to perform the function of the main server and a wireless communication device to dispatch messages.

Chapter 2

LITERATURE REVIEW

Since road accidents are increasing day by day, a lot of work is being carried out in various universities around the world, try to address this problem. Quite a lot of systems are under development, while some have already been completed and have already been deployed.

A paper by Suzhou Vocational University was published by IEEE [1]. The paper focused on mapping of image domain to real world domain and a comprehensive algorithm for automatic calibration was also provided. These techniques were used to monitor speed of vehicles.

University of California, Berkeley issued a paper titled: "A real-time computer vision system for vehicle tracking and traffic surveillance" [2]. The main focus of this paper was to address problems that occur in already existing systems. The document provides detailed algorithms for detection of moving object and short-comings of those algorithms and then comparison is done amongst the detection algorithms and the best one is suggested. Details of tracking algorithm and speed detection are also mentioned. In this document feature based tracking is done which improves accuracy and the camera is able to distinguish between two vehicles that are side by side; however this also increases computational complexity.

A research paper published in International Journal of Computer and Electrical Engineering, Vol. 3, No. 6, puts forward a new system for detecting vehicle speed which can also be used as an alternative to radar gun [3]. This research paper tells us about the draw backs of existing radar related speed detection systems. Keeping in view the underperformance of current speed detection systems a new more robust algorithm is put forward which uses the technique of image processing to detect speed of moving vehicles from a video sequence. This paper discusses different algorithms and there drawbacks. Only after detailed comparison and rigorous testing a new algorithm is presented in this paper which is a hybrid of adaptive background subtraction and three phase differencing. This combination helps to ratify one major drawback that is generation of false alarms.

Another research paper published by V R Siddhartha Engineering College introduces a method which detects speed of violating vehicles using image processing technique [4]. Digital signal processing chip is used to implement image processing technique over the video sequence netted from a video camera. The vehicles are detected by examining the binary image sequence that is constructed from captured frames by employing interface difference or the background subtraction algorithm. The system also detects the position of the moving vehicle in the scene and the position of the reference points and calculates the speed of each static image frame from the detected positions.

An article published by the name of "Image Processing in Road Traffic Analysis", authored by E. Atkociunas, R. Blake, A. Juozapavicius, M. Kazimianec put forward a new system whose capabilities include vehicle tracking, speed measurement without the use of any sensors, and recognition of license plate number of moving vehicle [5]. The system that we aim to develop as our final year project will also include these capabilities. This article uses computer vision method to monitor the traffic. The algorithmic processing of the system is in the following order:

a) Video Stream input to the computer.
b) Video conversion to a sequence of single frames.
c) Lane masking.
d) Background removal, noise and blob filtering, linking and labeling, contour parameter estimation.
e) Vehicle Tracking.

f) Vehicle speed calculation.

g) Number plate recognition.

A research paper by the name of "Vehicle speed detection in video image sequences using CVS method" and written by Arash Gholami Rad, Abbas Dehghani and Mohamed Rehan Karimshows us a new algorithm that makes use of digital video, image processing and computer vision to detect vehicle speed [6]. In view of our final year project this research paper proved invaluable to us since the final aim of this project was converging with ours that is to calculate speed of moving vehicle through a video camera. The project is divided in to the following parts:

1. **Camera Calibration:**

   Camera placed directly above the surface of the road with its optical axis inclined downward towards the road way. Camera calibration is an important aspect in our project as it will help us in determining the actual distance on the ground that appears in an image. Results obtained from camera calibration will then be used for mapping and speed calculation.

2. **Background update and removal unit:**

   - There is 2 type of data in a video sequence:

     a) Background data: Static objects like buildings, road surface, parked vehicles.

     b) Foreground data: Moving objects.

In order to detect speed of vehicles we have to extract and remove the background.

3. **Vehicle detection unit**

   - Background extraction and removal is one of the important part of vehicle extraction. Different algorithms are used for this purpose such as:
     a) Adaptive background estimation.
     b) Foreground detection using Kalman filter.
     c) Background estimation with Gaussian distribution.
     d) Adaptive mean filter for background estimation.
   - In our case we are using background estimation using Gaussian mixtures and object tracking using Kalman filter.

4. **Speed Measurement Unit:**

   - Speed of the vehicle is calculated using the position of the vehicle in each frame. For this purpose blob bounding box and centroid is found. Blob centroid helps in determining the distance vehicle has moved in the consecutive frames.

5. **Result analysis**

Chapter 3

SYSTEM DESIGN AND DEVELOPMENT

3.1 Approaches

One way to detect violations is by the use of RADAR technology. It works on Doppler Effect that calculates the time of transmitted pulse and received pulse and from that time distance of vehicles is calculated. Pulses of Radio frequency are sent at regular intervals to calculate speed of car. This method is currently being used however this method has many short falls. Radar can only focus on one vehicle at a time so in rush hours the systems' reliability decreases. In different weather conditions such as rain and humidity results not as accurate. Radar can also be jammed. Radar also needs to be used on the line of sight; otherwise results are not as accurate.

Another way is by the help of LiDAR. LiDAR suffers from the same problems as RADAR technology.

Electro-strips are embedded on road surface at a fixed distance when vehicles cross each strip pulse are sent to the processing unit which calculates time. Since distance is fixed speed is calculated. In case of too much traffic results are not accurate.

Signal processing and image processing is an evolving field. Due to increased processing power of processors this technology can not only produce more accurate and precise results, but this approach can be used to monitor multiple lanes and these systems can further be modified for traffic surveillance and security.

The basic design of this project can be broken down into three major parts:

- Detection
- Speed Calculations
- Notification

Each three of these parts consist of a Hardware and Software part.

3.2 Software Design

Software part will contain details of different algorithms implemented in this project.

3.2.1 Calibration

The input to tracking algorithm is an image acquired from the camera. In order to use the image pixels to compute real world distances and speeds, we must calibrate the camera. Following assumptions are made for calibration [1]:

- The road is flat (all points lie along a single plane)
- The road is straight (traffic motion is parallel to some axis)
- Occlusions are minimized (background images separate individual cars)

The transformation from screen space to road space is a projective transformation. This is performed by representing points in screen space as 2d homogenous coordinates (3 individual components), where (u, v) maps to (u', v', 1) in homogenous representation, and (u', v', w) maps to (u'/w v'/w) in non-homogenous representation. A 3x3 matrix is used for both translation and projection. Results of the calibration can be seen in the figure below.

Once the camera has been calibrated, location of lanes in road space and screen space is identified by clicking on center of each lane. For each point (Xs, Ys), the system maps this to the road space points (Xr and Yr), and represents the lane in road space as the line X=Xr, this is shown in the figure below. Then beginning and ending regions are defined. This is to ensure that cars in far distance do not distort statistics. Now screen-space points are transformed into road space.

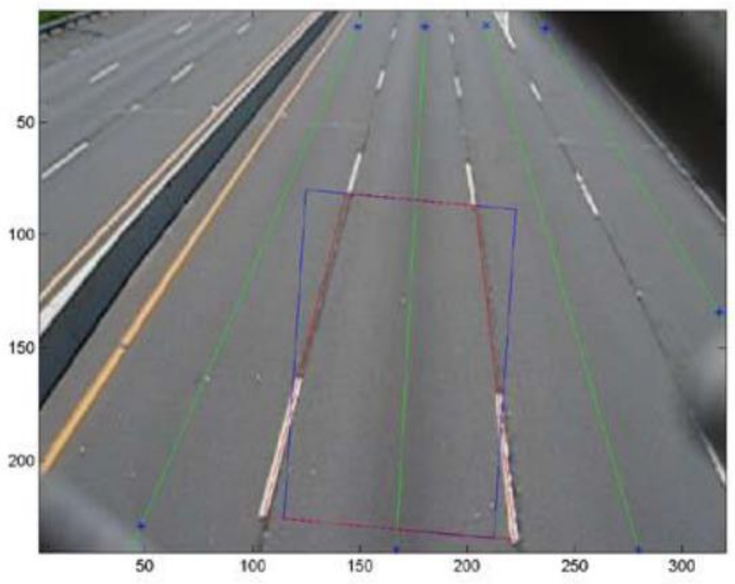


Figure 1 Calibration of the Camera

3.2.2 Background Detection

Car is located in a frame by looking for pixels that differ by some minimum amount from a standard background image.

History-based adaptive background detection method is used. This maintains a mean for each pixel. However, the mean over a fixed-size history is used instead of all the previously seen frames. This makes the algorithm more efficient. If the variance of the pixel over that history is below a conservative threshold, the pixel is considered stable. Only stable pixels are added to the background image. [2]

This method converges to an accurate background image rapidly. At the same time, it is much more stable than the RGB statistical learning method with a high learning rate; new cars do not affect the existing background as they rarely remain in the same place long enough to be considered part of the background. We expect that more global, longer term changes such as changes in lighting would eventually reflect in the background. [3]

3.2.3 Vehicle Tracking and Calculating Speed

The goal of the tracking system will be to generate per-lane traffic statistics at real-time speeds, as would be used on a major highway to report traffic flow information. As such, it was reasonable to assume a couple things about the input footage. First, we require that the camera be fixed in location. A camera strictly dedicated to analyzing traffic of a road would probably be fixed on a pole or overpass in one direction, making this a reasonable requirement. Second, we required an overhead view of the traffic. This second requirement was made in order to prevent the need for tracking multiple layers of traffic. With an overhead view, a system can simply analyze one area of the frame and know that this area belongs to only one lane of traffic. With side views, cars may be occluded by other cars or foreign objects [5].

To analyze the traffic subtract the background from an input frame and create a mask based on some error threshold. This mask needs to be analyzed to track moving objects along the road. The key observation is made that almost all of the movement along the road is done in the same direction. With this in mind, we can reduce the two-dimensional motion problem to the one-dimensional problem of tracking a vehicle within its lane. All of this is done to improve efficiency and reduce complexity.

In order to make this simplification, we define a region of interest in our input frames, one region for each traffic lane. The region of interest defines the center line of a lane as well as the beginning and end of the observation window. Once we subtract the background and generate our mask, we are left with a line of 0's and 1's, where 0's indicate background and 1's indicate the presence of a car. Then, from one frame to the next we only need to follow patches of 1's along the line in a consistent manner [7].

By tracking the bottom of these patches and calculating the differences between them, we can determine how many pixels each car moved along the line between each frame. Using the calibration data, we can map these pixels to a virtual road space and determine the distance in feet. Simple calculation leads to a speed for a car in miles per hour, and in the aggregate case, an average speed for the lane.

The implementation of the above system is straightforward. For each region of interest (each traffic lane being analyzed), we store its coordinates in pixel space, the number of cars it has seen, the average speed of the lane, and an array of car structures with an entry for each car currently in the lane. In an initialization step, we extract the background image of the line by pulling corresponding entries out of the input background image.

For each lane in each frame, we extract the pixels on the line of interest for the lane and subtract our background information. We then compare the sum of squared differences with a threshold, and generate a mask based on this comparison. The threshold is a parameter of the system and depends on the noise sensitivity of the camera. We perform a median filter on this vector to remove noise in the form of small chunks of black or white that might disrupt the algorithm.

The images below show the tracking process for 3 frames. The first 2 frames are consecutive. Adjacent to the frames are images of the lines of interest for the middle lane before thresholding, before using the median filter, and after the median filter. White pixels indicate the presence of a car, and black pixels indicate the background.

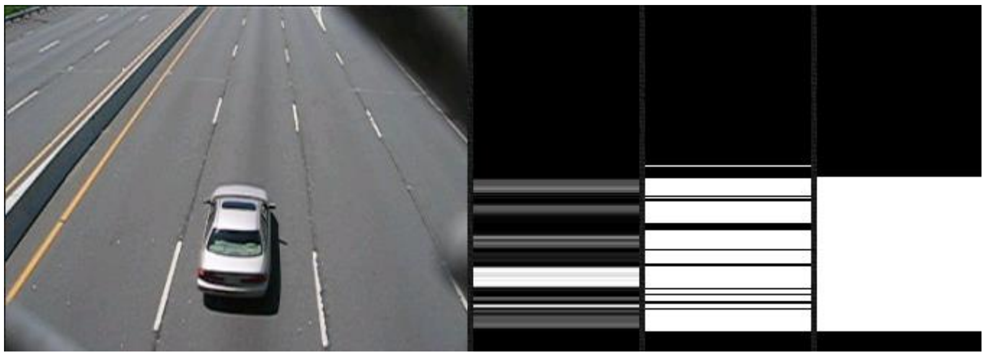


Figure 2 Background Subtraction as Car Enters

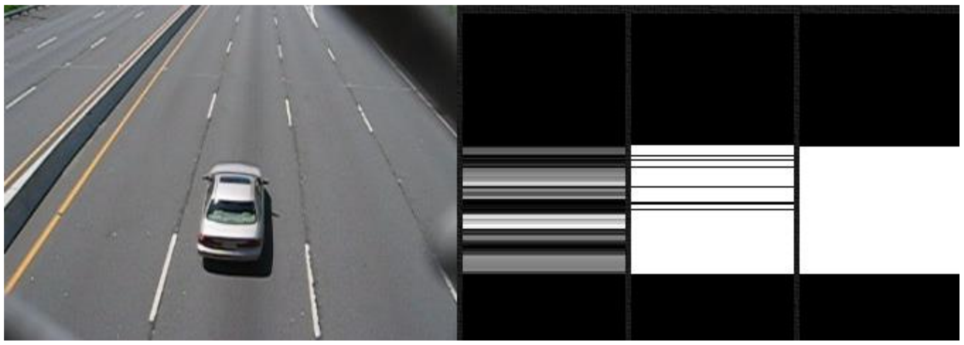


Figure 3 Background Subtraction as Car Travels

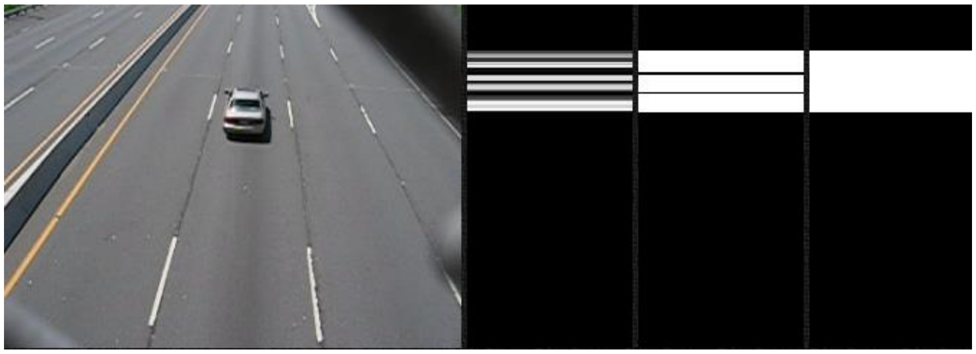


Figure 4 Background Subtraction as Car Leaves

We then examine the beginning of our line of interest and if we check for the case where we saw a positive error in the previous frame and see no error now. This suggests that a new car may have just completely entered the region of interest. The algorithm proceeds up from the bottom to find the next positive error and thus locate the bottom of the new car. A check is made to find the top of the positive error chunk to determine the length of the car. Cars that do not meet a minimum length requirement are rejected as noise, and are usually due to visual aberrations in the input image near the beginning of the lane. If a new car is indeed found, it is added to the cars array of the lane.

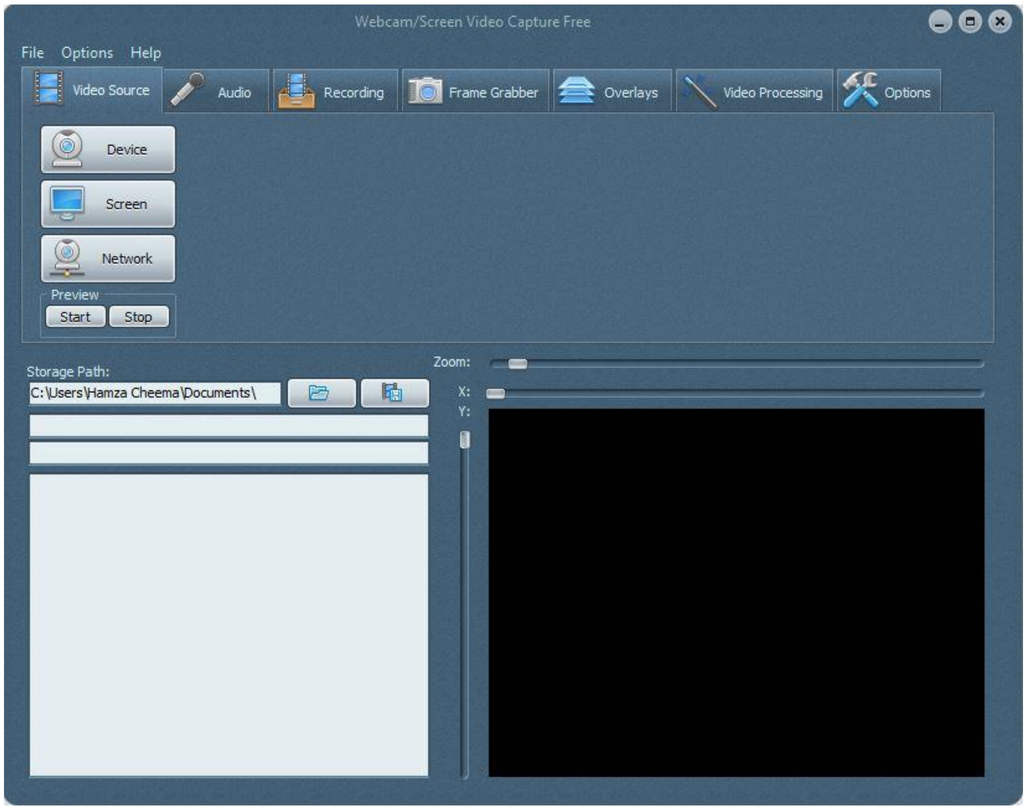
For each existing car in the lane's car array, we start at the last known position of the bottom of the car and look forward along the line to find the next positive error chunk in the current frame. We take this as the new bottom of the car and calculate the distance between the two cars in road space, using the transformation from the calibration procedure. This distance in feet is converted to miles per hour and appended to the car's existing speed information.

If a search for a new bottom of a car reaches the end of the line, the car has exited the region of interest. In this case, we increment the lane statistics and calculate the average speed of the car as the average of the entries in its speed array. This average speed is used to update the average traffic speed of the lane.

3.2.4 Live Stream

There is no such thing as a continuous video stream in computers and embedded systems just like there is no such thing as continuous data. If we get continuous data we need to sample it at specific time intervals and quantize it at certain levels, similarly when we get a constant video stream we sample it after some time intervals. This is done by selecting individual frames after a fixed period of time, then processing is done on these individual frames according to our requirement.

We are using a free firmware webcam grabber developed by SoftPerk to grab frames from our webcam.



Storage Path:
C:\Users\Hamza Cheema\Documents\

Figure 5 Working Environment of Frame Grabber

3.3 Hardware Design

Hardware consists of a camera and beagle board which will be attached to the camera.

3.3.1 Camera

Figure 6 Logitech Camera B525

3.3.1.1 Specifications [8]

What you need:	Windows Vista®, Windows® 7 (32-bit or 64-bit) or Windows® 8
Basic requirements:	▪ 1 GHz ▪ 512 MB RAM or more ▪ 200 MB hard drive space ▪ Internet connection ▪ USB 1.1 port (2.0 recommended)
Part Number	▪ PN 960-000715

Technical Specifications	▪ HD video calling (1280 x 720 pixels) with recommended system ▪ Video capture: Up to 1280 x 720 pixels ▪ Logitech Fluid Crystal™ Technology ▪ Photos: Up to 8 megapixels (software enhanced) ▪ Hi-Speed USB 2.0 certified (recommended) ▪ Universal clip fits laptops, LCD or CRT monitors ▪ Autofocus ▪ Built-in microphone with Logitech Right Sound™ technology
Logitech webcam software:	▪ Pan, tilt, and zoom controls ▪ Video and photo capture ▪ Face tracking ▪ Motion detection

Table 1 Specifications of Camera

3.3.2 BeagleBone Black

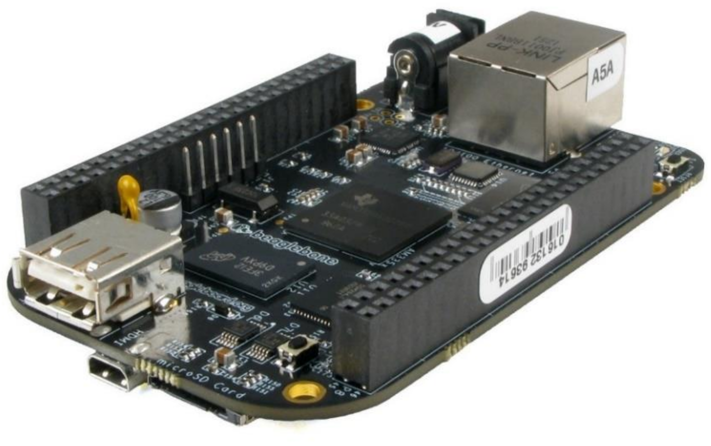


Figure 7 Beagle Board Black Isometric View

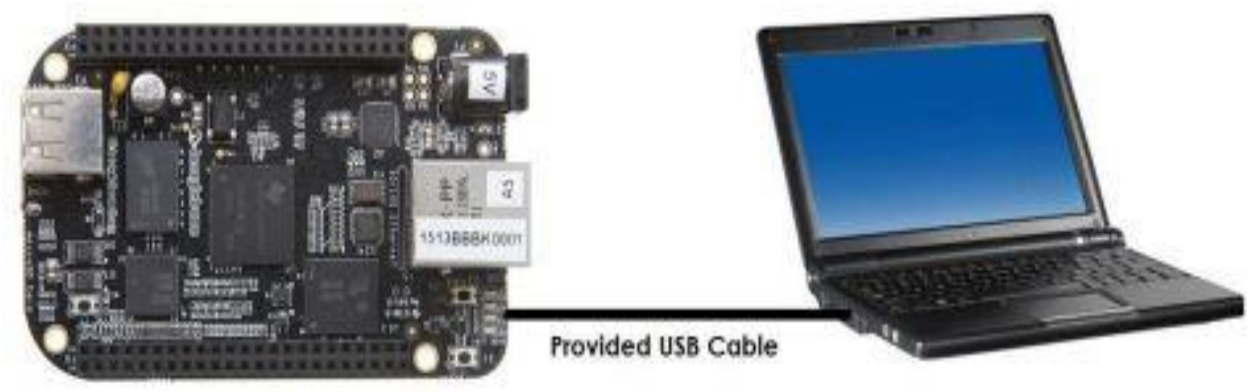


Figure 8 Tethering with PC

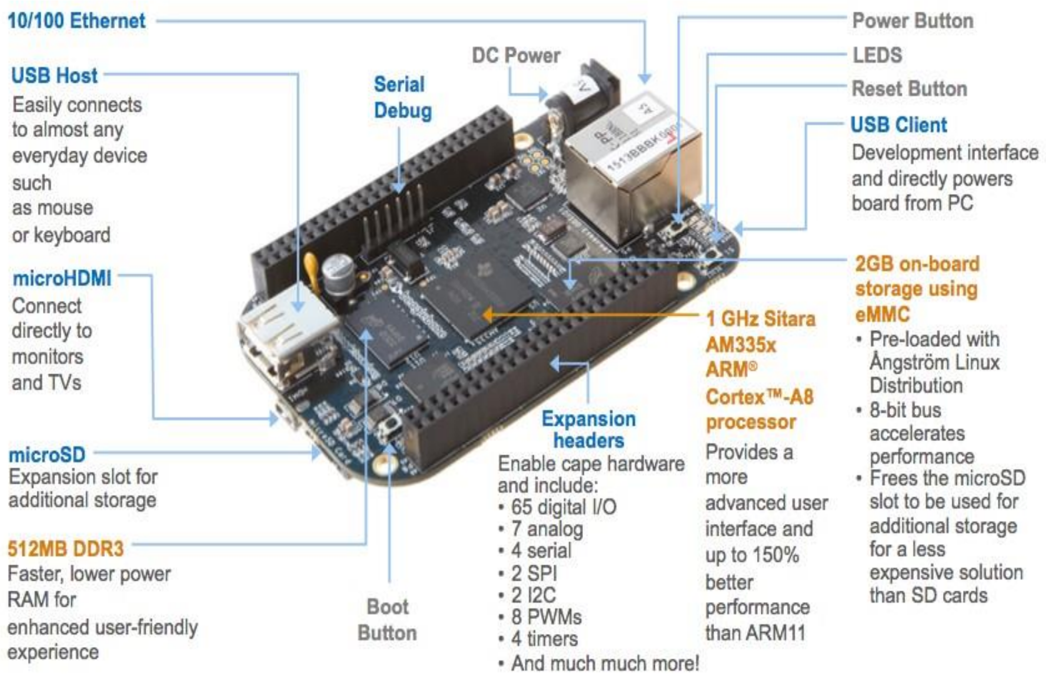
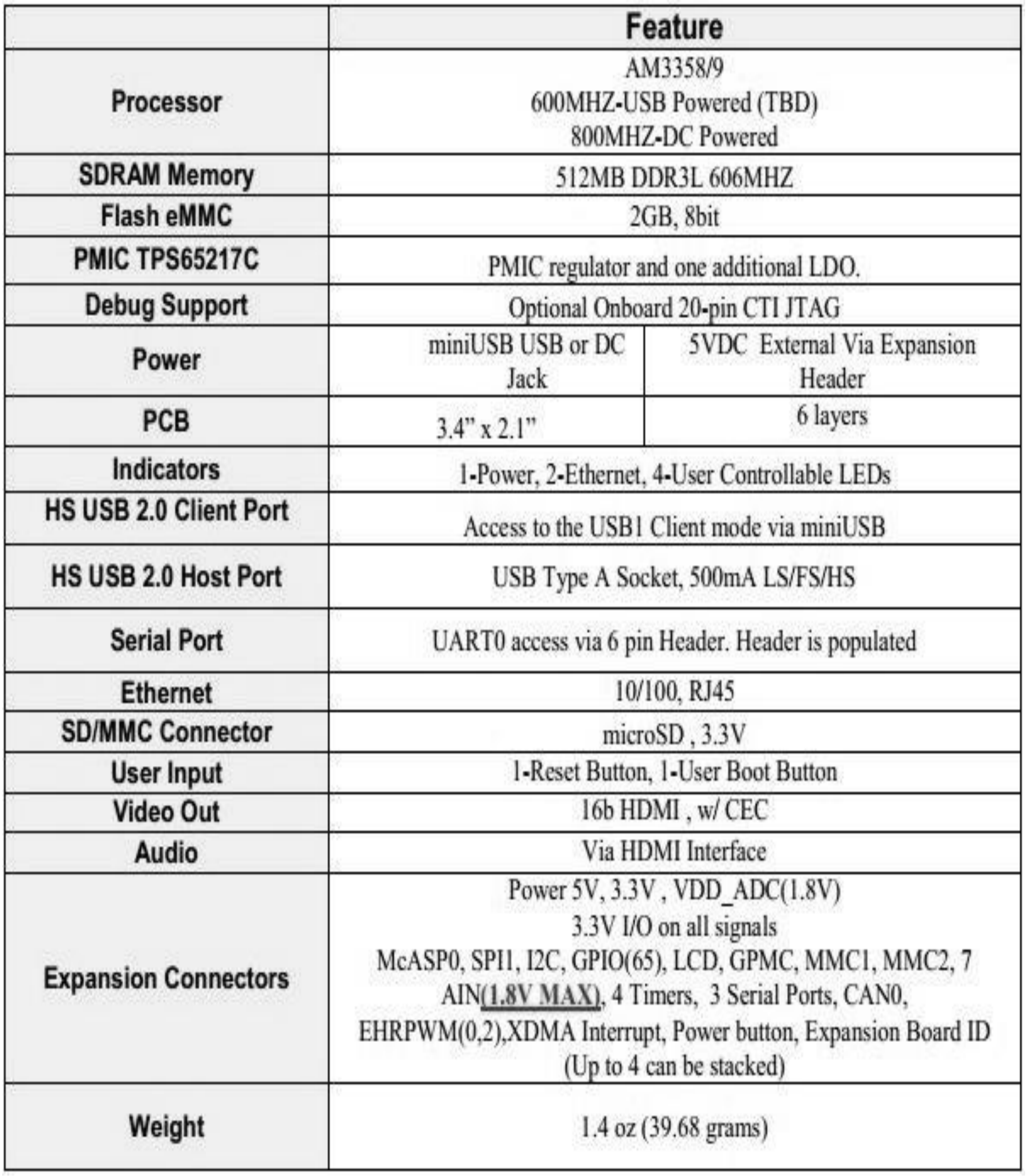

10/100 Ethernet
USB Host
Easily connects to almost any everyday device such as mouse or keyboard
microHDMI
Connect directly to monitors and TVs
microSD
Expansion slot for additional storage
512MB DDR3
Faster, lower power RAM for enhanced user-friendly experience
Serial Debug
DC Power
Boot Button
Expansion headers
Enable cape hardware and include:
• 65 digital I/O
• 7 analog
• 4 serial
• 2 SPI
• 2 I2C
• 8 PWMs
• 4 timers
• And much much more!
1 GHz Sitara AM335x ARM® Cortex™-A8 processor
Provides a more advanced user interface and up to 150% better performance than ARM11
Power Button
LEDS
Reset Button
USB Client
Development interface and directly powers board from PC
2GB on-board storage using eMMC
• Pre-loaded with Ångström Linux Distribution
• 8-bit bus accelerates performance
• Frees the microSD slot to be used for additional storage for a less expensive solution than SD cards

Figure 9 Beagle Board Black Port Labels

3.3.2.1 Specifications [9]



	Feature	
Processor	AM3358/9 600MHZ-USB Powered (TBD) 800MHZ-DC Powered	
SDRAM Memory	512MB DDR3L 606MHZ	
Flash eMMC	2GB, 8bit	
PMIC TPS65217C	PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG	
Power	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to the USB1 Client mode via miniUSB	
HS USB 2.0 Host Port	USB Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	1-Reset Button, 1-User Boot Button	
Video Out	16b HDMI , w/ CEC	
Audio	Via HDMI Interface	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(65), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 3 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	

Table 2 Specifications of BEAGLEBONE BLACK

3.4 Hardware Interface

Beagle-bone black is a development board that harbors an Arm Cortex A8 processor with a Linux kernel. Firstly the operating system of the board is updated by getting the latest image of Linux Debian. Image of the latest operating system was burned into a micro-SD card and when the card was plugged into the board, board itself updated its operating system.

Once the operating system was updated, a network was created between the Beagle-bone Black and PC. Since Beagle-bone Black has a Linux kernel; that is the operating system used in this board is Linux so whatever process is to be performed at the Beagle-bone it is performed using Linux. With the help of newly established network between the board and PC, new drivers can now be installed over the internet.

Once appropriate drivers were installed to the board Camera was connected using the on-board USB port. USB port provide serial communication between the I/O devices. Just by connecting the device it cannot be ensured whether the device has interfaced properly or not, so using a simple command of “lsusb” all the input and output devices can be viewed. When the camera appeared as an I/O device on the board it was checked if video input is enable using the command “ls -al”. When it was discovered that the video input is enable we used the appropriated code to capture frames at a fixed rate and time interval.

These frames are stored in eMMC, which is an on-board memory storage they can be easily accessed for further processing.

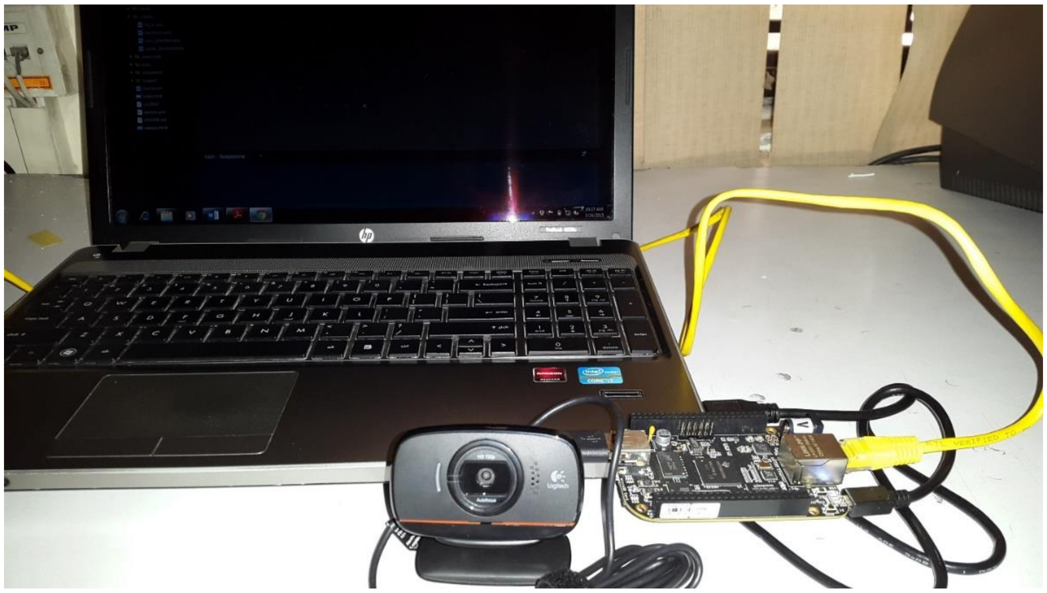


Figure 10 Hardware Interface

3.5 Database and Notification System

Database and Notification is the part that is used to inform the owner of the car that a violation has been made and a fine need to be paid.

3.5.1 Database

Databases are very common these days. Almost every website, Blog, E-mail services, E-commerce sites, and Cloud storage system needs a database to store data. We are also using a database in this project to store the data of violating vehicles and their respective users. We are using PhpMyAdmin to create a MySQL database on a WAMPserver. WAMPserver is a windows web development environment .It allows you to create web applications with Apache2, PHP, and a MySQL database. Alongside, PhpMyAdim is used to manage database.

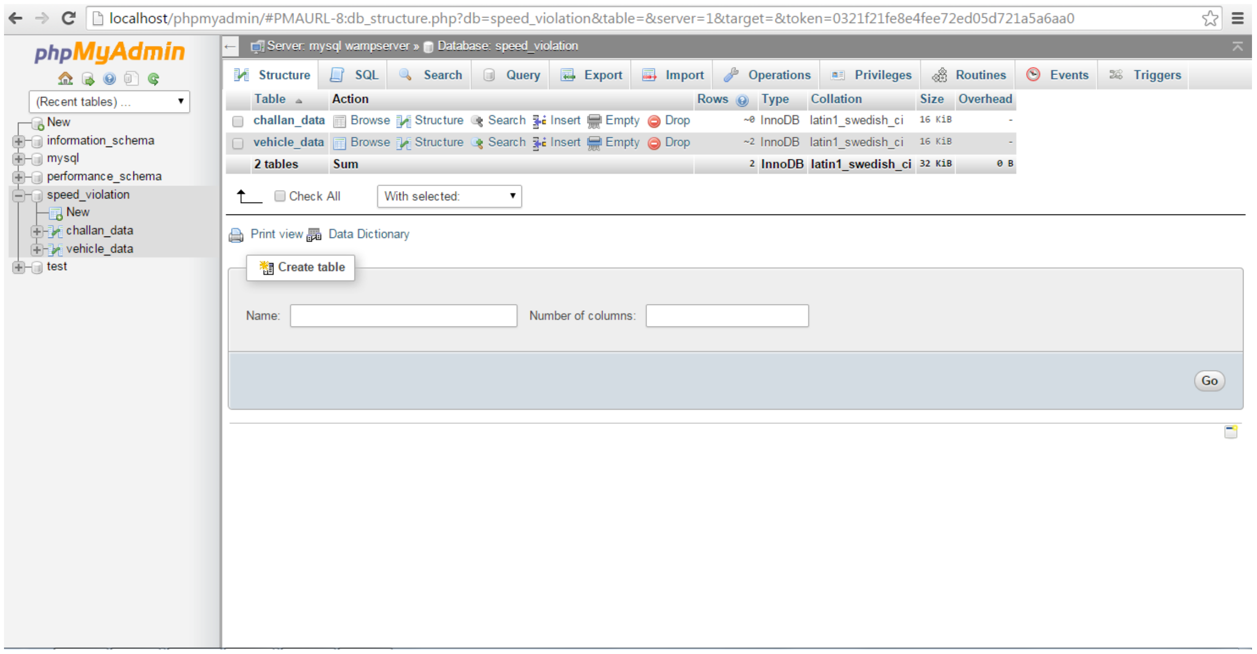


Table	Action	Rows	Type	Collation	Size	Overhead
challan_data	Browse Structure Search Insert Empty Drop	~0	InnoDB	latin1_swedish_ci	16 KiB	-
vehicle_data	Browse Structure Search Insert Empty Drop	~2	InnoDB	latin1_swedish_ci	16 KiB	-
2 tables	Sum	2	InnoDB	latin1_swedish_ci	32 KiB	0 B

Figure 11 Working Environment of php Server

3.5.2 Notification

The following three methods can be used to notify the violator:

- GSM module
- API (Application Programming Interface)
- E-mail

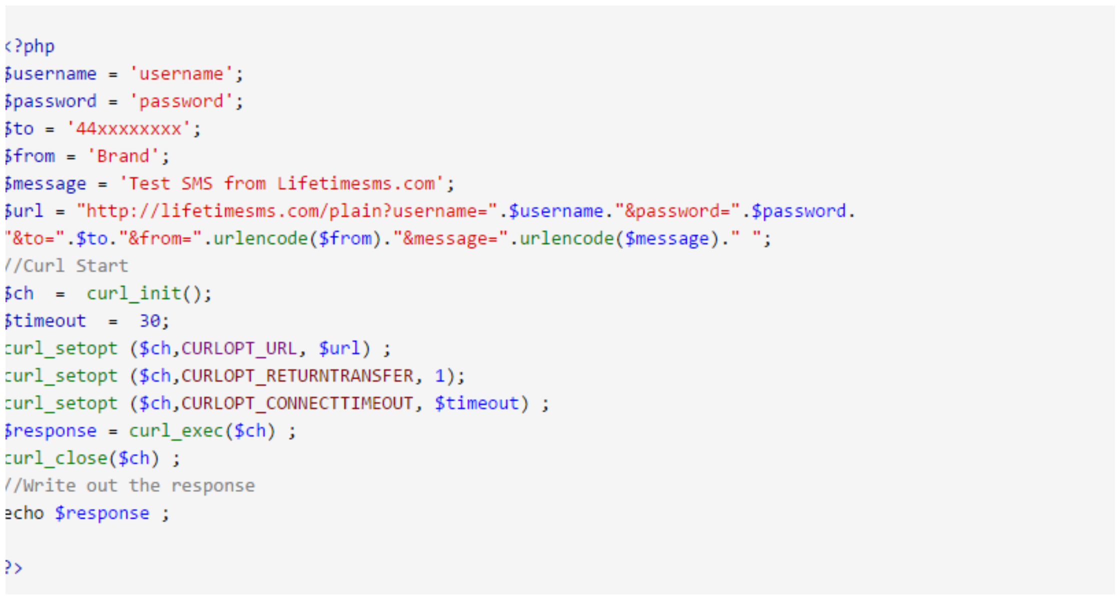
3.5.2.1 GSM Module

A GSM modem is a specialized type of modem which accepts a SIM card and operates over a subscription to a mobile operator, just like a mobile phone. When connected to a computer, this allows the computer to communicate over the mobile network. These modems can be used for sending and receiving SMS and MMS messages.

3.5.2.2 API (Application Program Interface)

API, an abbreviation of application program interface, is a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact and are used when programming graphical user interface (GUI) components.

Since our projects require sending of bulk SMS we are using the PHP rest API for sending SMS to the violator.



```
<?php
$username = 'username';
$password = 'password';
$to = '44xxxxxxxx';
$from = 'Brand';
$message = 'Test SMS from Lifetimesms.com';
$url = "http://lifetimesms.com/plain?username=".$username."&password=".$password.
"&to=".$to."&from=".urlencode($from)."&message=".urlencode($message)." ";
//Curl Start
$ch  =  curl_init();
$timeout  =  30;
curl_setopt ($ch,CURLOPT_URL, $url) ;
curl_setopt ($ch,CURLOPT_RETURNTRANSFER, 1);
curl_setopt ($ch,CURLOPT_CONNECTTIMEOUT, $timeout) ;
$response = curl_exec($ch) ;
curl_close($ch) ;
//Write out the response
echo $response ;

?>
```

Figure 12 The SMS Script Code in PHP

Chapter 4

PROJECT ANALYSIS AND EVALUATION

Last 10 years has seen an uncontrolled population growth worldwide which has given rise to urban congestion. One of the fallout has been increased traffic congestion on existing road networks. This project is relatively new and in this project image processing is even more recent and underdeveloped. Research is still being done throughout the world therefore not a lot of material is available over the internet. However given the resources available and considering that this is the first time this technology is being implemented in Pakistan we tried our best to fulfill our objectives and the results have been quite satisfactory.

4.1 Speed Testing

This project has been tested rigorously within MCS as well as outside MCS. During development of this project it was tested in DHA and MCS. Once it was completed, on 15th May 2015 Friday; we tested this project with Motorway Police on Motorway M2, location 343 Thallian Camp (FWO), and compared the results against already existing technology that is the radar gun. The results came out to be satisfactory and also appreciated by the Motorway Police officials.

4.1.1 Testing in DSP LAB

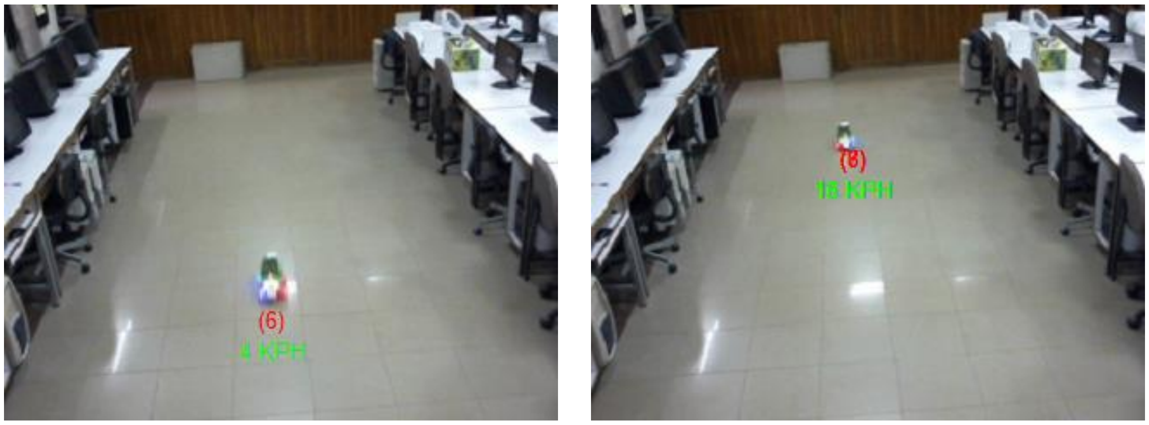


Figure 13 Testing on Toy Car

4.1.2 Testing in MCS

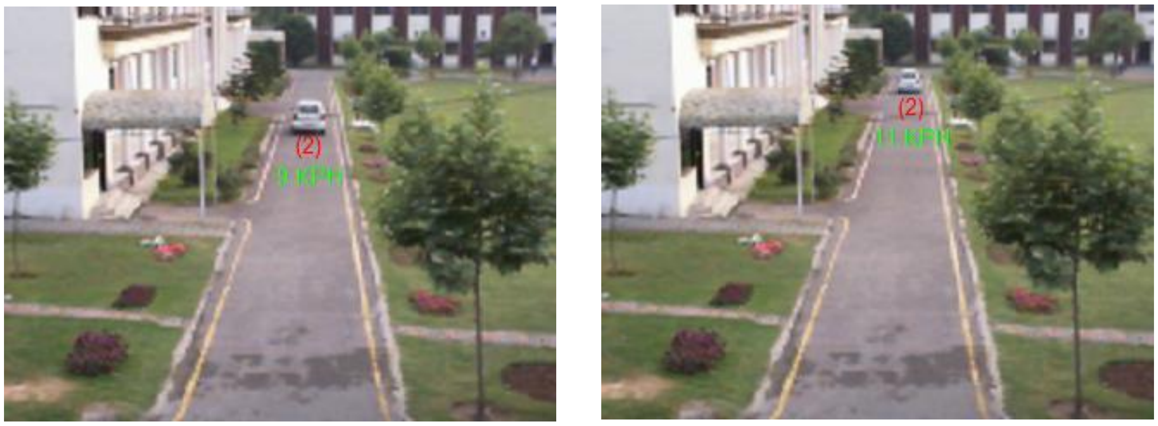


Figure 14 Testing in front of Trg bn MCS

4.1.3 Testing in DHA

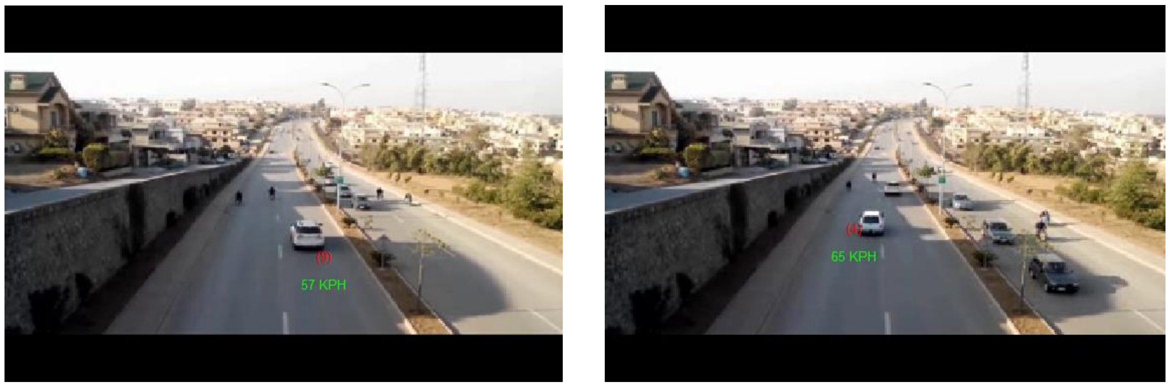


Figure 15 Testing in DHA-I

4.1.4 Testing on Motorway M2 (Location 343 – Thallian Camp)

Figure 16 Installation of Camera on Motorway

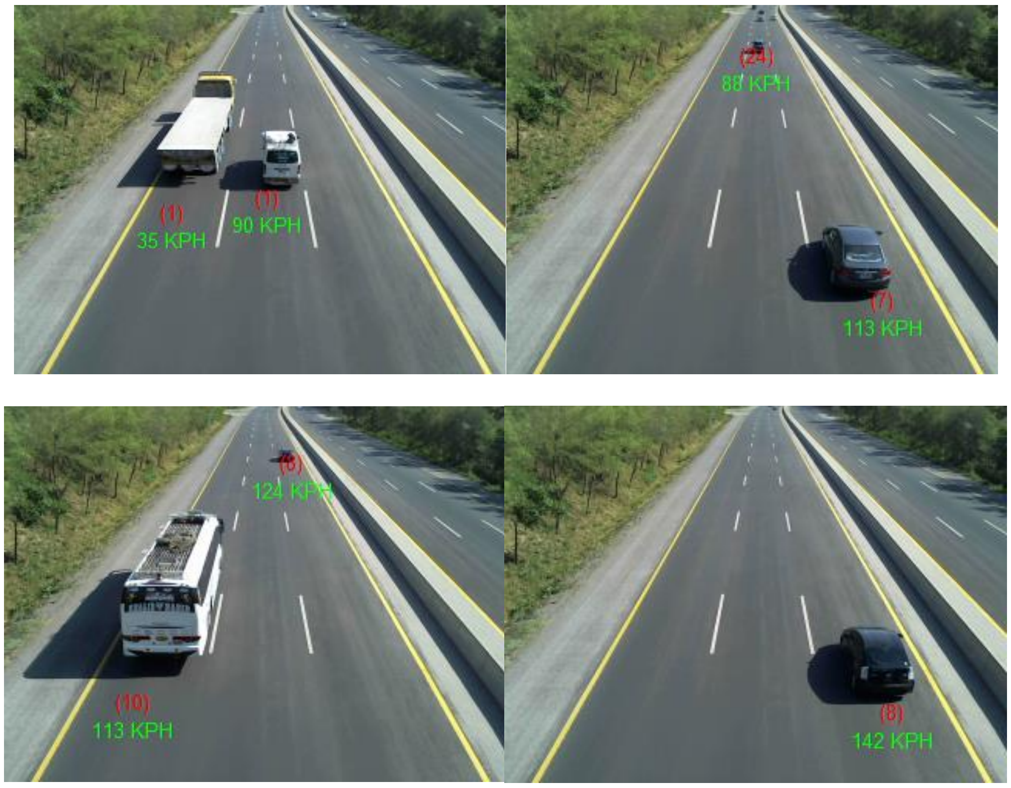


Figure 17 Results Obtained on Motorway

4.2 Final Outlook

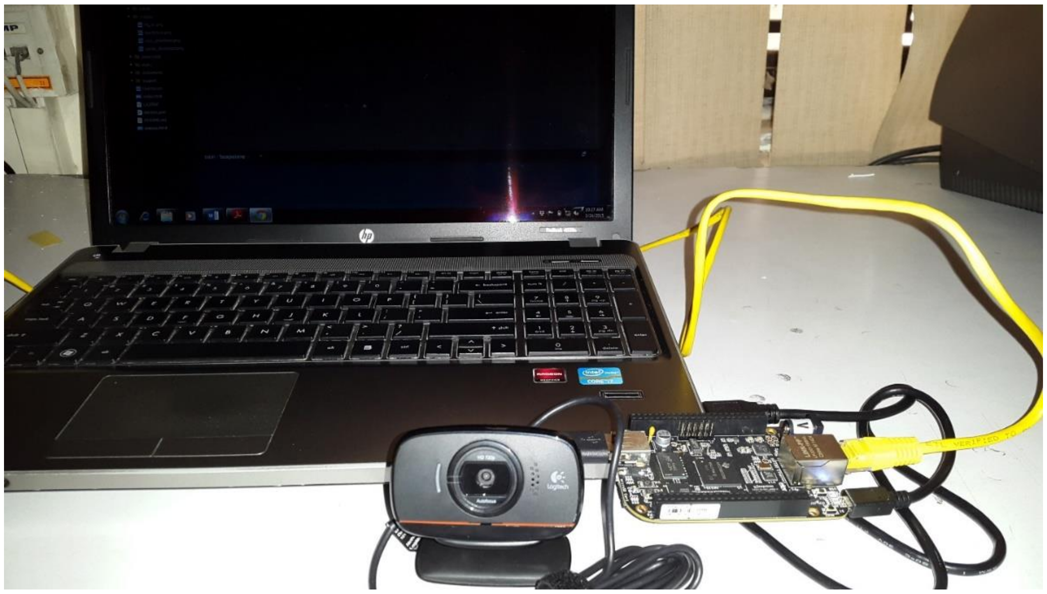


Figure 18 Final Outlook of Project

4.3 Registration Number Extraction

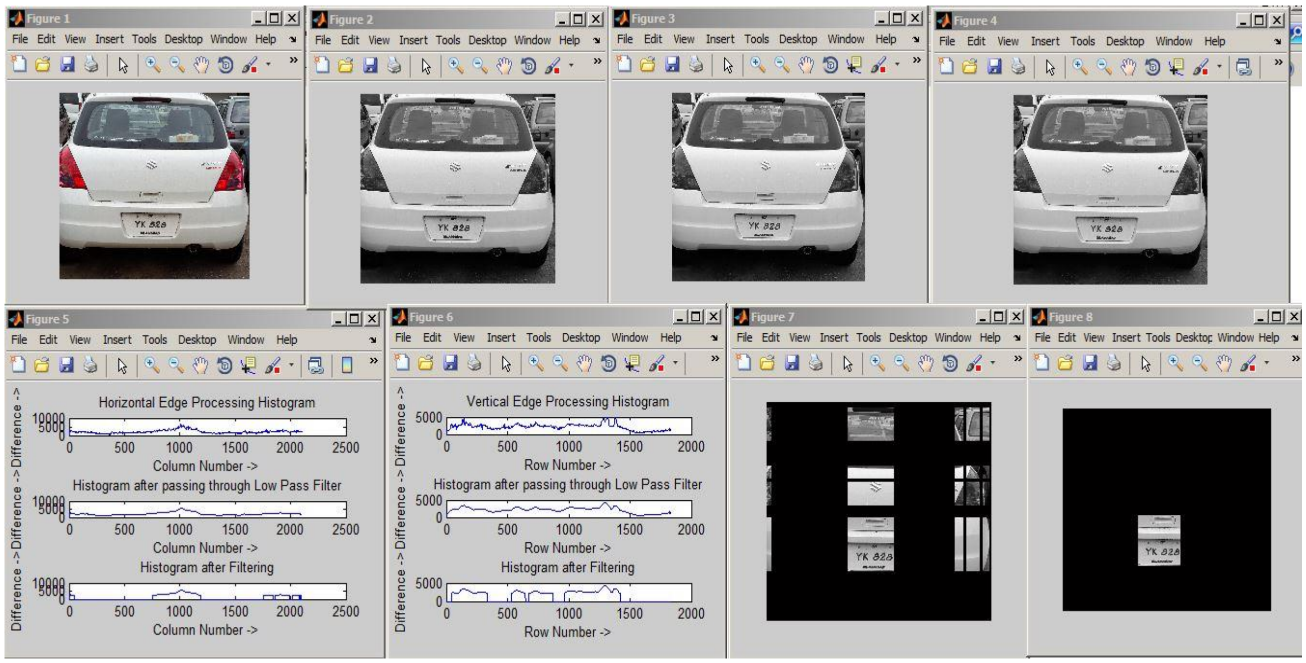
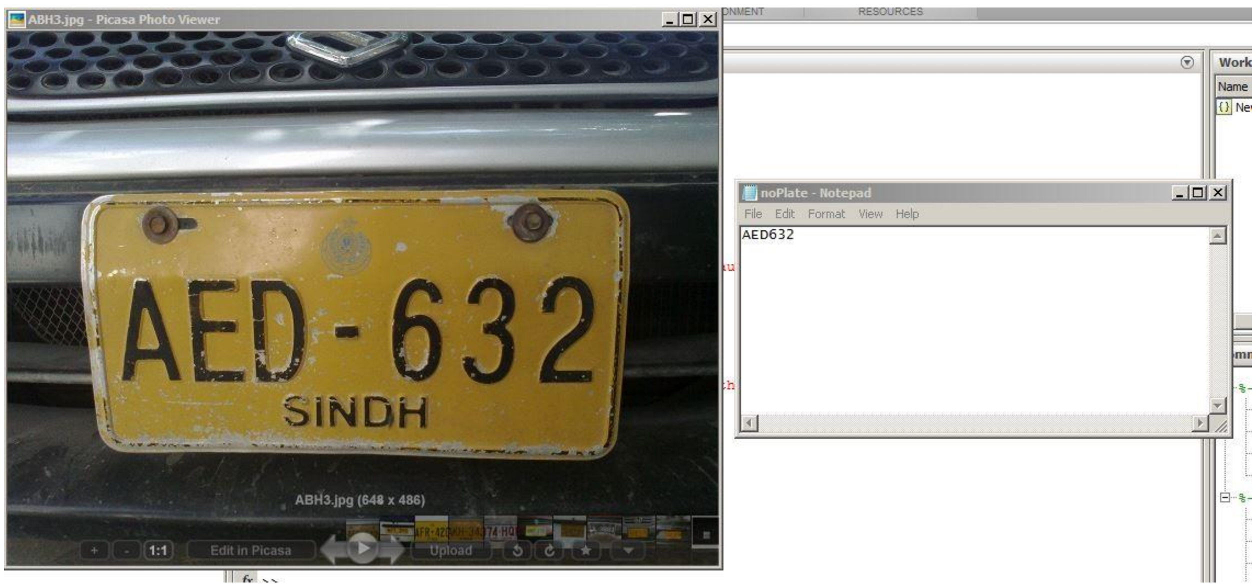


Figure 1
Figure 2
Figure 3
Figure 4
Figure 5
Figure 6
Figure 7
Figure 8
File Edit View Insert Tools Desktop Window Help
Horizontal Edge Processing Histogram
Histogram after passing through Low Pass Filter
Histogram after Filtering
Column Number ->
Difference ->
Vertical Edge Processing Histogram
Row Number ->
YK 828

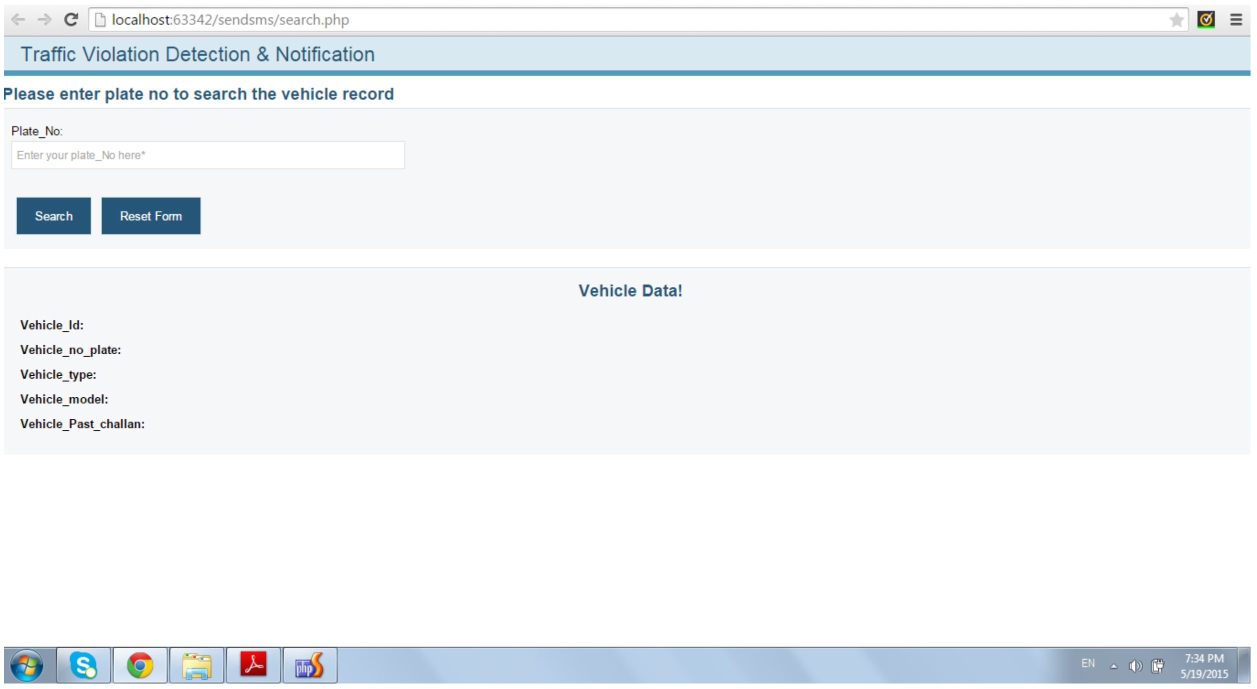
Figure 19 Number Plate Extraction from Vehicle



AED632

Figure 20 Number Plate Reading

4.4 Database and Notification



Traffic Violation Detection & Notification

Please enter plate no to search the vehicle record

Plate_No:

Enter your plate_No here*

Vehicle Data!

Vehicle_Id:

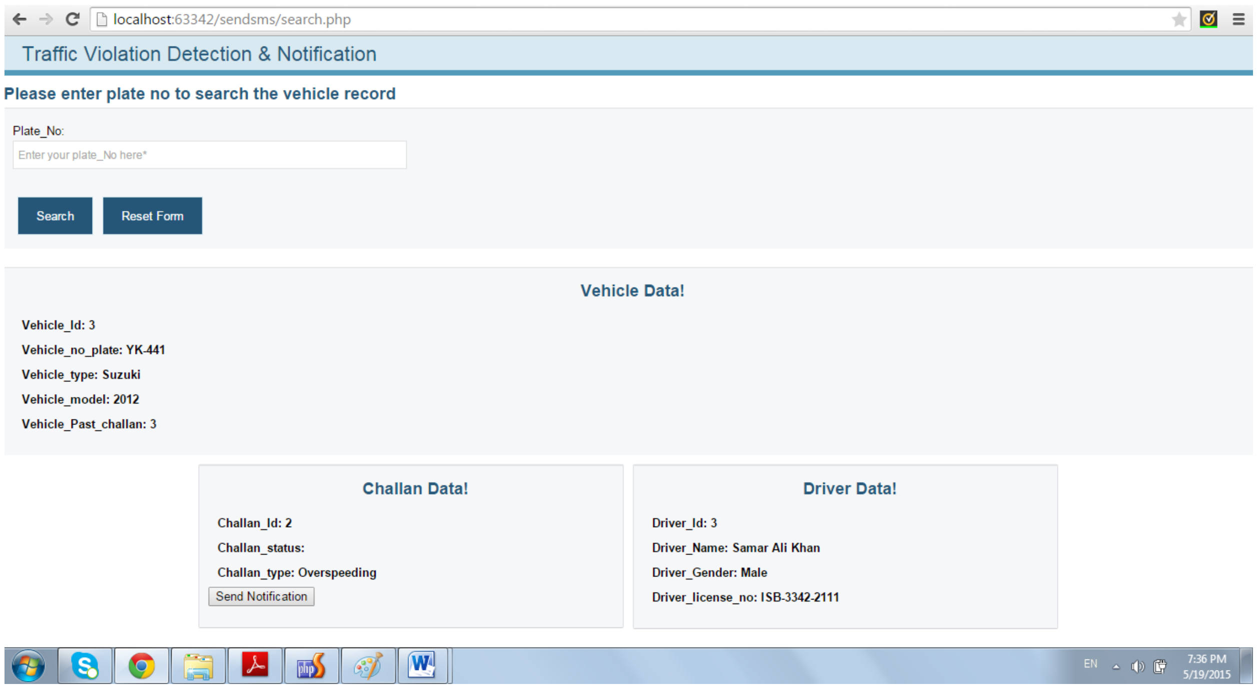
Vehicle_no_plate:

Vehicle_type:

Vehicle_model:

Vehicle_Past_challan:

Figure 21 Database Front End


localhost:63342/sendsms/search.php

Traffic Violation Detection & Notification

Please enter plate no to search the vehicle record

Plate_No:

Enter your plate_No here*

Search | Reset Form

Vehicle Data!

Vehicle_Id: 3
Vehicle_no_plate: YK-441
Vehicle_type: Suzuki
Vehicle_model: 2012
Vehicle_Past_challan: 3

Challan Data!

Challan_Id: 2
Challan_status:
Challan_type: Overspeeding

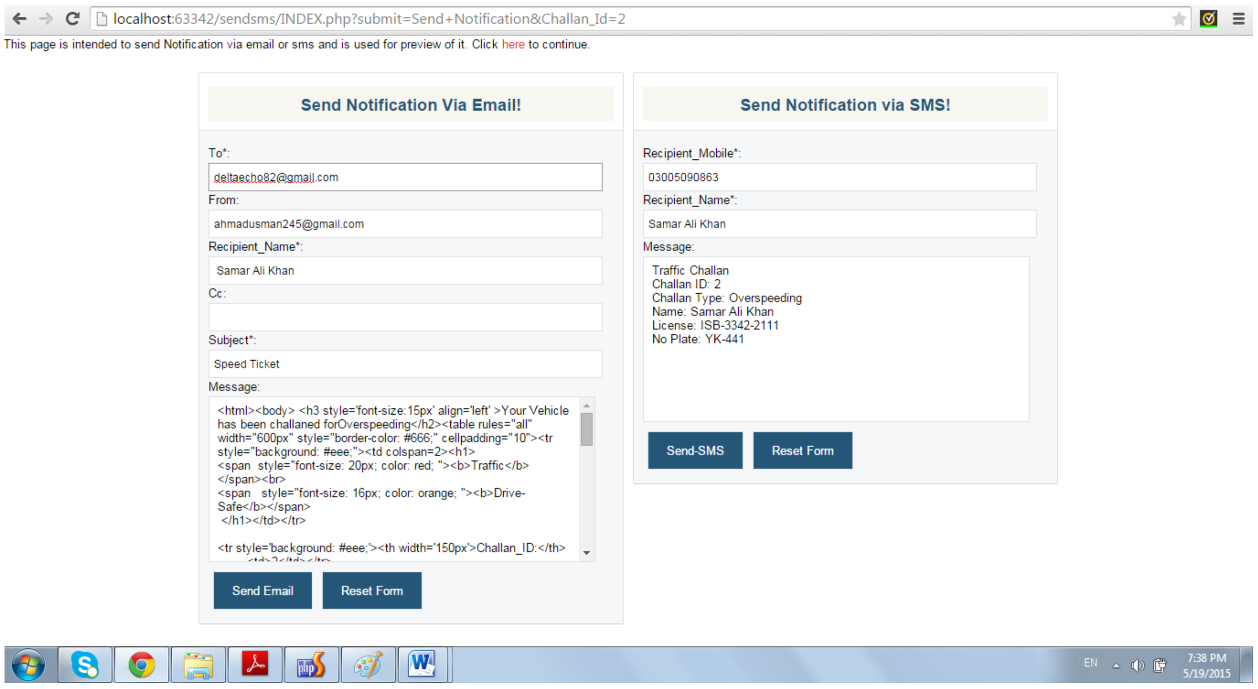
Send Notification

Driver Data!

Driver_Id: 3
Driver_Name: Samar Ali Khan
Driver_Gender: Male
Driver_license_no: ISB-3342-2111

7:36 PM 5/19/2015

Figure 22 After Searching for Required Registration No.


localhost:63342/sendsms/INDEX.php?submit=Send+Notification&Challan_Id=2

This page is intended to send Notification via email or sms and is used for preview of it. Click here to continue.

Send Notification Via Email!

To*: deltaecho82@gmail.com
From: ahmadusman245@gmail.com
Recipient_Name*: Samar Ali Khan
Cc:
Subject*: Speed Ticket
Message:

```
<html><body> <h3 style='font-size:15px' align='left' >Your Vehicle
has been challaned forOverspeeding</h2><table rules="all"
width="600px" style="border-color: #666;" cellpadding="10"><tr
style="background: #eee;"><td colspan=2><h1>
<span  style="font-size: 20px; color: red; "><b>Traffic</b>
</span><br>
<span   style="font-size: 16px; color: orange; "><b>Drive-
Safe</b></span>
 </h1></td></tr>

<tr style='background: #eee;'><th width='150px'>Challan_ID:</th>
```

Send Email | Reset Form

Send Notification via SMS!

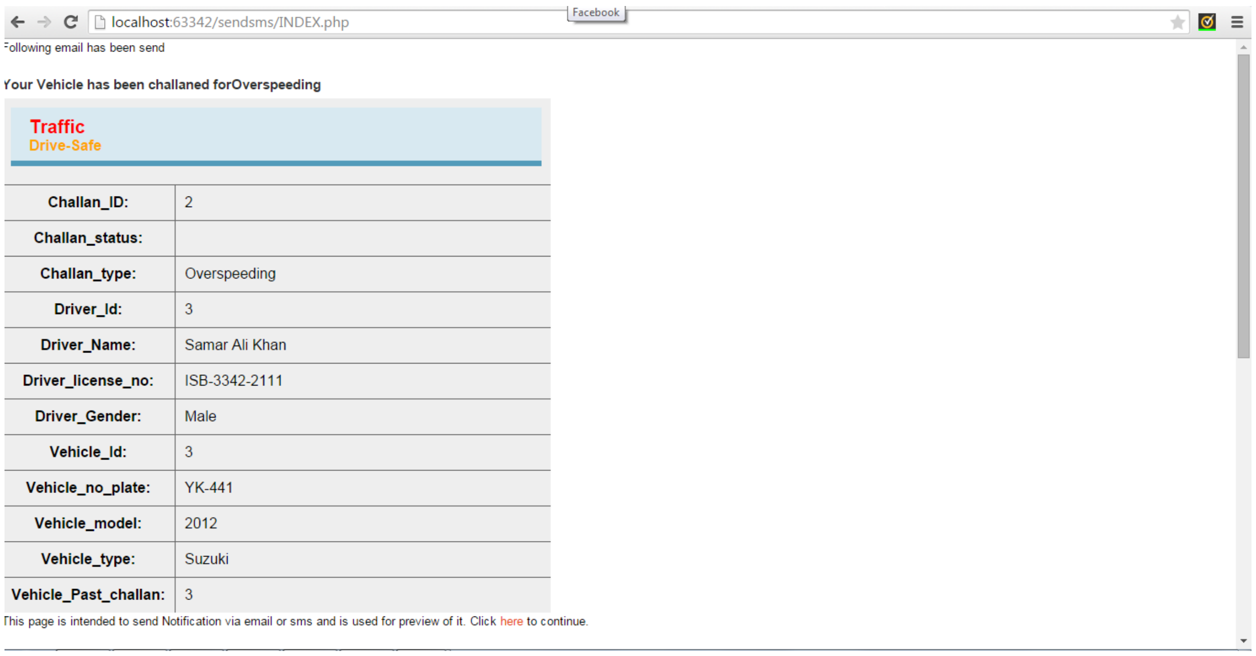
Recipient_Mobile*: 03005090863
Recipient_Name*: Samar Ali Khan
Message:

```
Traffic Challan
Challan ID: 2
Challan Type: Overspeeding
Name: Samar Ali Khan
License: ISB-3342-2111
No Plate: YK-441
```

Send-SMS | Reset Form

7:38 PM 5/19/2015

Figure 23 Notification Panel



Following email has been send

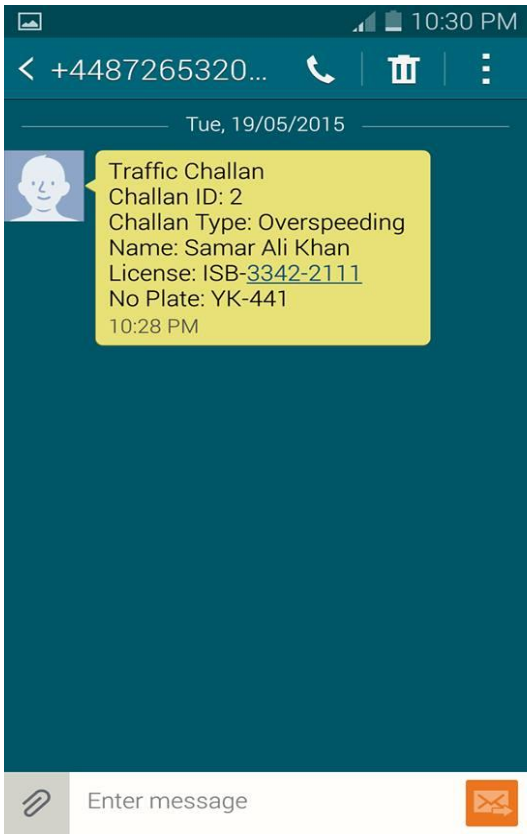
Your Vehicle has been challaned forOverspeeding

Traffic
Drive-Safe

Challan_ID:	2
Challan_status:	
Challan_type:	Overspeeding
Driver_Id:	3
Driver_Name:	Samar Ali Khan
Driver_license_no:	ISB-3342-2111
Driver_Gender:	Male
Vehicle_Id:	3
Vehicle_no_plate:	YK-441
Vehicle_model:	2012
Vehicle_type:	Suzuki
Vehicle_Past_challan:	3

This page is intended to send Notification via email or sms and is used for preview of it. Click here to continue.

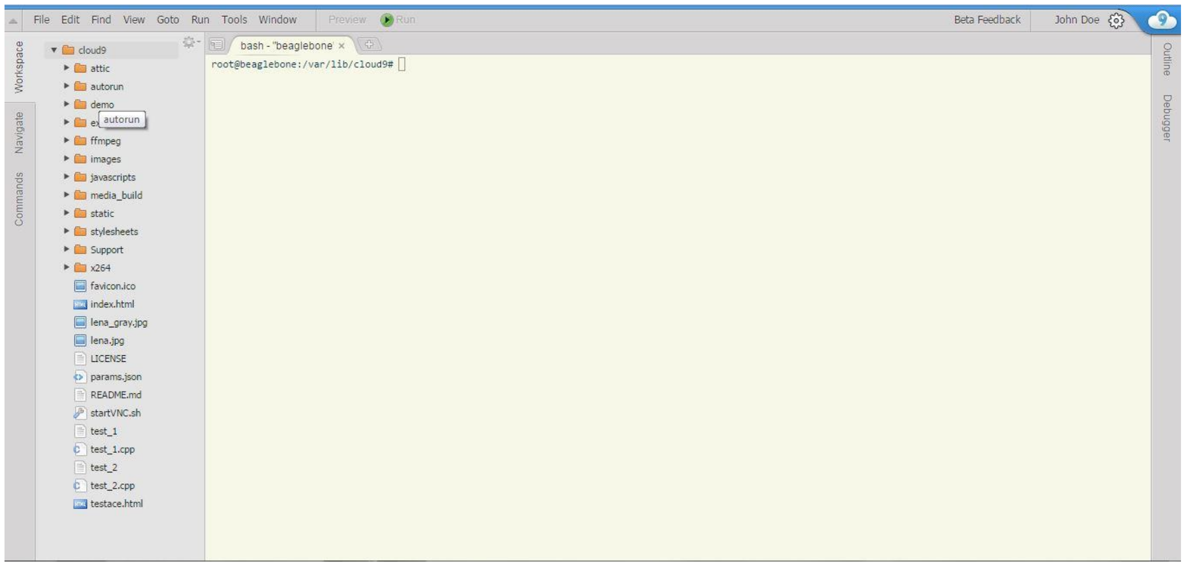
Figure 24 Notification via Email



Traffic Challan
Challan ID: 2
Challan Type: Overspeeding
Name: Samar Ali Khan
License: ISB-3342-2111
No Plate: YK-441
10:28 PM

Figure 25 Notification via SMS

4.5 Beaglebone Black



```
root@beaglebone:/var/lib/cloud9#
```

Figure 26 BeagleBone Cloud9 Interface

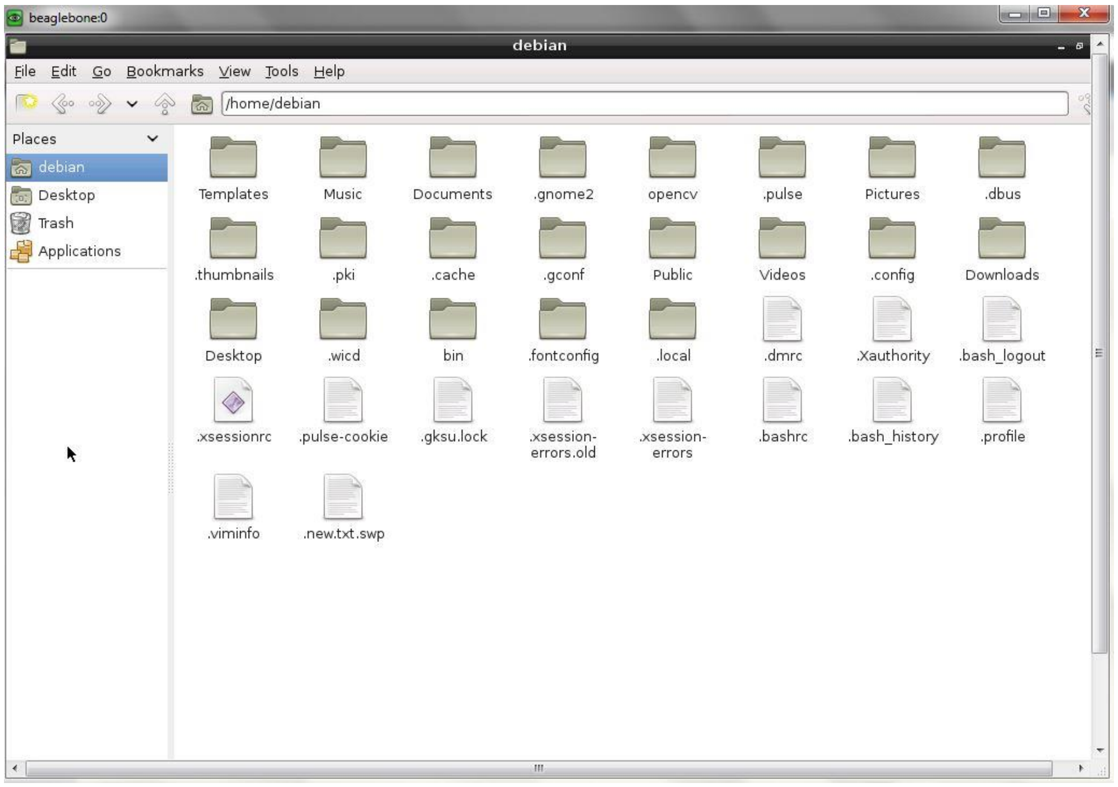


Figure 27 BeagleBone VNC Server Interface

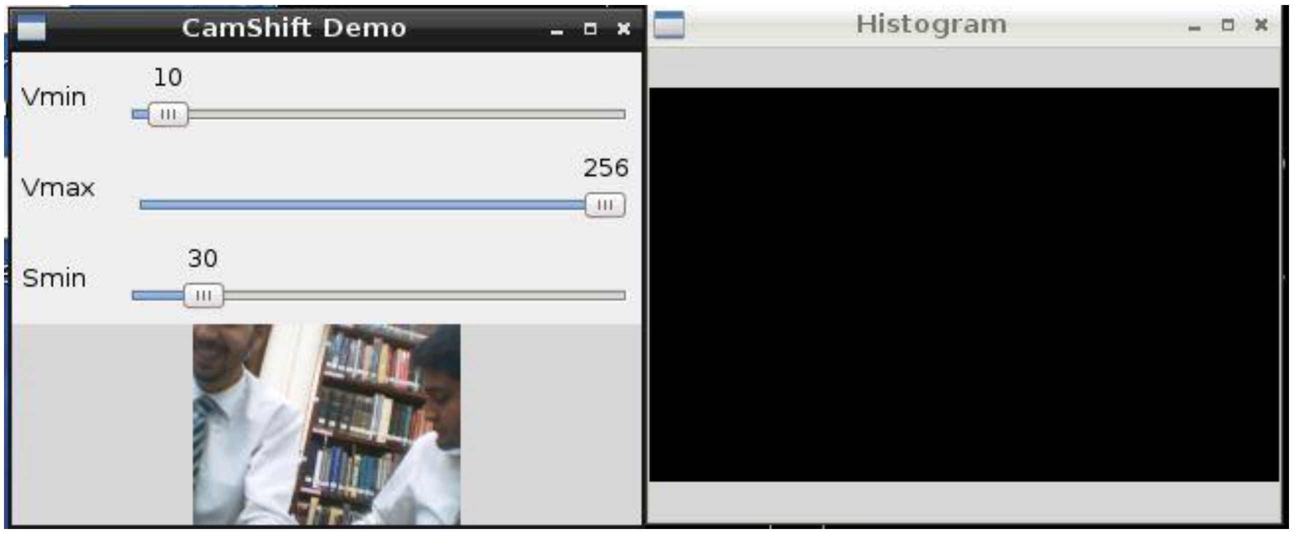


Figure 28 Interface of CamShift

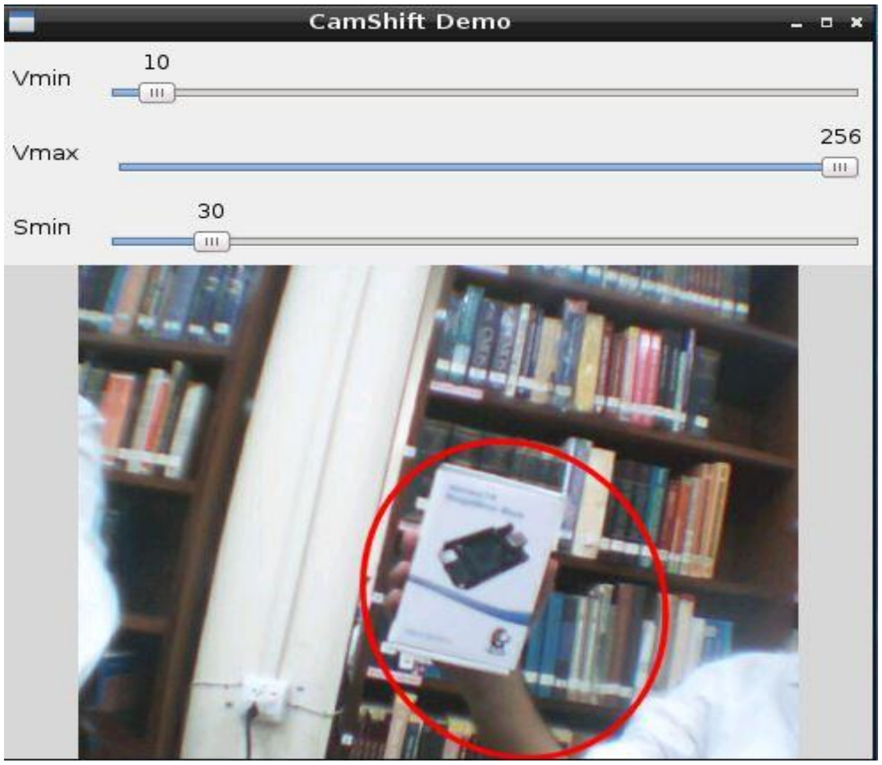


Figure 29 Tracking of Specified Object on BeagleBone

4.6 Problems Faced

While testing of this project following were the major problems faced:

- The first and the most serious problem faced is that camera needs to stay still. Since calculation of speed mainly depends on background subtraction, whenever there is a slight movement of camera the whole background changes which results in various object to be specified as moving. To solve this problem it must be ensured that camera is secured tightly to one place and it does not move at all.
- Second problem faced is that if car jumps lanes (i.e. it changes lane in that case) or the car is in between two different lanes then it is not detected. Since lanes are defined and if any car does not fall within a specified region then it is taken as noise. This is because of an assumption that all cars move within lanes.
- Results of this project are better if the road is straight. If the road has bends then an assumption made earlier about the one dimensional motion of car becomes wrong due to which results are not very accurate.

One major problem we faced in using a GSM module was that in light of the current security situation of Pakistan, government of Pakistan has placed a ban on purchase of GSM modules therefore we are using a different approach. Now instead of using a GSM or 3G module we are using SMS API (sending SMS via internet).

Chapter 5

FUTURE WORK AND CONCLUSION

5.1 Recommendations for Future Work

Since this technology is still under the process of development there are a lot of prospects for future work.

- This system can be modified for security as well. This way cost of installing both security cameras and speed cameras will reduce, only one equipment will be required.
- A complete network can be made between different system points with a main server. This way there will be a centralized point for checking security footage as well as all the data will be in one place.
- This system can be made more compact, more robust and more efficient.
- At this moment this project gives best results over straight stretches of road, algorithm can be modified to work on bends as well.
- This system can also be modified to include other traffic violations as well.
- Another modification that can be done is implement is using IR camera which can give better results during night and rainy weather.

5.2 Conclusion

Traffic Violation Detection and Notification is an application based project, which works in real time to capture, analyze and process real life scenarios. This technology is the way forward in Traffic Management and Control. In UK during 2014 British Government issued order to replace all the previous speed cameras with this technology.

5.2.1 Overview

The basic purpose behind undertaking this project was that over speeding has become a real problem that is actually costing thousands of lives each year. This, very real, problem needed immediate addressing and a system was required that is efficient, accurate and robust.

5.2.2 Objectives Achieved

We achieved most of the objectives we set out to achieve. The system is able to detect, track and tag a fast moving vehicle accurately and efficiently. It is also able to calculate the speed of that vehicle precisely and accurately and violator is notified immediately.

5.2.3 Achievements

This project was tested rigorously and repeatedly in different environment. The results were compared against a Radar gun, which is a slightly outdated, but still highly accurate device. The results were more than satisfactory as there was minimum difference between the two readings. Our achievements were acknowledged by Motorway police.

Chapter 6

REFERENCES

REFERENCES

[1] Z. Zhang, "A Flexible New Technique for Camera Calibration," Microsoft Research, [Online]. Available: http://robots.stanford.edu/cs223b04/JeanYvesCalib/htmls/links.html.

[2] D. B. P. M. Benjamin Coifmana, "A real-time computer vision system for vehicle tracking and traffic surveillance," University of California, Berkeley, 1998.

[3] J. Zhang and Like Zhang and Heng-Ming Tai, "Efficient Video Object Segmentation Using Adaptive Background Registration," 2004.

[4] H. E. A. M. E. Osman Ibrahim, "Speed Detection Camera System using Image Processing," International Journal of Computer and Electrical Engineering, Vol. 3, No. 6, 2011.

[5] P. Getreuer, "Image Processing with MATLAB," [Online]. Available: http://www.getreuer.info/tutorials/matlabimaging.

[6] logitech, "logitech Webcams," [Online]. Available: http://www.logitech.com/en-us/webcam-communications/webcams.

[7] BeagleBoard.org, "BeagleBoard Black Reference Manual," 2013.

APPENDIX A

MATLAB CODES

Code of Background Detection

```
%function bgimage = BackgroundDetect(dataset, start_frame, end_frame,
%history, thresh, asap, display)
%
% dataset - name of the directory with sequential image files
% start_frame - number of first frame to use in directory
% end_frame
% history - number of frames to use, default: 10
% thresh - default: .01
% asap - stops background estimation once background model is stable
% display - shows a side-by-side of the evolution of bg model
%
% bgimage = computed background image
%
% Example: bgimage = BackgroundDetect('speedtrap_data', 0, 1000, 10, .1,
true, true);
%
function bgimage = BackgroundDetect(dataset, start_frame, end_frame,
history, thresh, asap, display)

if nargin < 7
    display = true;
end

if nargin < 6
    asap = true;
end

if nargin < 5
    thresh = .01;
end

if nargin < 4
    history = 10;
end


num_frames = end_frame - start_frame + 1;

history_index = 1;

init_color = [1 0 0];

thresh = thresh * history;

for i = 1 : history,
    data(:, :, :, i) = double(imread(sprintf('%s/%06d.bmp', dataset,
start_frame + i - 1))) / 255;
end

for i = 1 : history - 1,
    error(:, :, :, i) = data(:, :, :, i + 1) - data(:, :, :, i);
    error(:, :, :, i) = error(:, :, :, i) .* error(:, :, :, i);
```

```
end

error_sum = sum(sum(error, 4), 3);
error_mask = error_sum < 0.01;

stable = false;
stable_mask = error_mask;

mean_image = mean(data, 4);

for k = 1 : 3,
    bgimage(:, :, k) = init_color(k) * ~error_mask + mean_image(:, :, k)
.* error_mask;
end

if display
    figure(1);
    imtable([2 1], 1, data(:, :, :, history));
    imtable([2 1], 2, bgimage);
    truesize;
    drawnow;
end

for i = history + 1 : num_frames,
    history_index = mod(i - 1, history) + 1;
    history_prev = mod(i - 2, history) + 1;

    data(:, :, :, history_index) = double(imread(sprintf('%s/%06d.bmp',
dataset, start_frame + i - 1))) / 255;
    error(:, :, :, history_prev) = data(:, :, :, history_index) - data(:,
:, :, history_prev);
    error(:, :, :, history_prev) = error(:, :, :, history_prev) .*
error(:, :, :, history_prev);

    error_sum = sum(sum(error, 4), 3);
    error_mask = error_sum < 0.01;

    mean_image = mean(data, 4);

    for k = 1 : 3,
        if stable
            bgimage(:, :, k) = error_mask .* (bgimage(:, :, k) +
mean_image(:, :, k)) / 2 + ~error_mask .* bgimage(:, :, k);
        else
            bgimage(:, :, k) = (error_mask & stable_mask) .* (bgimage(:,
:, k) + mean_image(:, :, k)) / 2 + (error_mask & ~stable_mask) .*
mean_image(:, :, k) + ~error_mask .* bgimage(:, :, k);
        end
    end

    stable_mask = stable_mask | error_mask;
    stable = ~sum(~stable_mask(:));

    if display
```

```
        imtable([2 1], 1, data(:, :, :, history_index));
        imtable([2 1], 2, bgimage);
        truesize;
        drawnow;
    end

    if asap && stable
        disp(sprintf('Used %d frames to find stable background', i));
        break;
    end
end

return;
```

Calibration Function

```
%Calibrate.m - calibrates the image, prompting user for input
%
% M - background image from BackgroundDetect.m
% w - world space width in feet of the quad
% h - world space height of the quad
%
% returns calibrated parameters that map screen to world coordinates
%
% Example [TM, lane_x, lane_start, lane_end] = CalibrateFunction(bgimage,
% 5, 13);
%
% It will prompt you to click on the four points of that you know a
%priori
% correspond to a world-space quad, for which you know the width and
% height. Start with the lower-left side, then the lower-right, then
% upper-right, and finally upper left. (By Lower, I mean the side of the
% quad that the vehicles will enter first).
%
% Then you click on the center of each lane, and press enter when done.
%
% Finally, for each lane, click once on the start of the lane to track,
%and
% once on the end.
%
function [A, lane, lane_start, lane_end] = CalibrateFunction(M, w, h)

hold off;
figure()
image(M);
fprintf('Click on four points of quad, counter-cw from lower left\n');
[x y] = ginput(4);  %REMOVE THIS LINE TO USE THE LAST POINTS

u = [0; w; w; 0];
v = [0; 0; h; h];

%x0 = eye(3);
x0 = [u, v, [1 1 1 1]']' / [x, y, [1 1 1 1]']';
%x0(3,1) = 0.001;
```

```
hold on;
UV1 = inv(x0) * [u, v, [1 1 1 1]']';
plot([UV1(1,:) UV1(1,1)], [UV1(2,:) UV1(2,1)]);

x0 = x0(1:8);

options = optimset('LargeScale', 'off', 'MaxFunEvals', 800000, 'TolFun',
1e-20, 'TolX', 1e-20, 'display', 'iter');

%[A, fval, exitflag, output] = fminunc(@calibobjfun, x0, options, x, y,
u, v);
%[A, fval, exitflag, output] = fminsearch(@calibobjfun, x0, options, x,
y, u, v);
%x0
%A = reshape([A(:); 1], 3, 3);
A = homography2d([x, y, [1 1 1 1]']', [u, v, [1 1 1 1]']' );

hold on;
UV2 = inv(A) * [u, v, [1 1 1 1]']';
for i=1:4, UV2(:,i) = UV2(:,i) ./ UV2(3,i); end;
plot([UV2(1,:) UV2(1,1)], [UV2(2,:) UV2(2,1)], 'r')

%fval
%exitflag
%output

%Get the lane dividers
fprintf('Click once on each lane from left to right.  Press enter when
finished\n');

lane = [];
[lx ly] = ginput;
for i =1:length(lx),
    S = A*[lx(i); ly(i); 1];
    [minX, minY, maxX, maxY] = ScreenPixelRange(A, S(1)/S(3), size(M,2),
size(M,1));
    plot([minX maxX], [minY maxY], 'g');
    lane = [lane; S(1)/S(3)];
end;

%Get the start and the end of the road
fprintf('For each lane, click on the beginning and the end of the region
to be tracked\n');

lane_start = [];
lane_end = [];
for i = 1:length(lx),
    [lx ly] = ginput(1);
    road_coord = ScreenToRoad(A, [lx; ly]);
    lane_start = [lane_start; road_coord(2)];
    screen_coord = RoadToScreen(A, [lane(i); road_coord(2)]);
    plot( screen_coord(1), screen_coord(2), '*' );
    [lx ly] = ginput(1);
    road_coord = ScreenToRoad(A, [lx; ly]);
    lane_end = [lane_end; road_coord(2)];
    screen_coord = RoadToScreen(A, [lane(i); road_coord(2)]);
    plot( screen_coord(1), screen_coord(2), '*' );
end;
```

Tracking and Speed Calculation Code

```
function TrackCars(dataset, start_frame, end_frame, bgimage, camera_fps,
min_car_length, threshold, median, delay, lane_x, lane_start, lane_end,
TM)
%TrackCars(dataset, start_frame, end_frame, bgimage, camera_fps,
min_car_length, threshold, median, delay, lane_x, lane_start, lane_end,
TM)
%dataset = 'speedtrap_data1';
%start_frame = 0;
%end_frame = 2565;
%camera_fps = 15;
%min_car_length = 5;
%threshold = 0.0015;
%median = 15;

lane = struct('points', [], 'length', [], 'line', [], 'tracked_line', [],
'bg', [], 'num_cars', [], 'done_cars', [], 'flow', [], 'new_car', [],
'cars', [], 'text', []);

num_lanes = length(lane_x);

for i = 1 : num_lanes
    lane(i).points = round(RoadToScreen(TM, [lane_x(i) lane_x(i);
lane_start(i) lane_end(i)]));
    lane(i).length = max(abs(lane(i).points(:, 1) - lane(i).points(:,
2))) + 1;
    lane(i).line(1, :) = round(linspace(lane(i).points(1, 1),
lane(i).points(1, 2), lane(i).length));
    lane(i).line(2, :) = round(linspace(lane(i).points(2, 1),
lane(i).points(2, 2), lane(i).length));
    for k = 1 : 3,
        lane(i).bg(:, k) = bgimage(sub2ind(size(bgimage), lane(i).line(2,
:), lane(i).line(1, :), repmat(k, [1 lane(i).length])))';
    end
    lane(i).num_cars = 0;
    lane(i).done_cars = 0;
    lane(i).flow = 0;
    lane(i).saw_car = true;
    lane(i).cars = struct('id', [], 'bottom', [], 'speed', []);
end

update_stats = false;

fig_im = figure;
set(fig_im, 'DoubleBuffer', 'on');
subplot('position', [0 0 1 1]);
image(bgimage);
```

```
axis off;
truesize;

fig_stats = figure;
axis off;
for j = 1 : num_lanes
    lane(j).text = text((2 * j - 1) / (2 * num_lanes), 0.5, '');
    set(lane(j).text, 'HorizontalAlignment', 'center', 'FontSize', 24);
    set(text((2 * j - 1) / (2 * num_lanes), 0.9, sprintf('Lane %d', j)),
'HorizontalAlignment', 'center', 'FontSize', 24, 'FontWeight', 'bold');
end

for i = start_frame:end_frame,
   data = double(imread(sprintf('%s/%06d.bmp', dataset, i))) / 255;

    figure(fig_im);
%The following two lines will capture output to files
    capture = getframe(fig_im);
    imwrite(capture.cdata, sprintf('%s/out%06d.bmp', dataset, i));
    hold off;
    subplot('position', [0 0 1 1]);
    image(data);
    axis off;
    hold on;

    for j = 1 : num_lanes,
        cars = struct('id', [], 'bottom', [], 'speed', []);

        clear error_line;
        for k = 1 : 3,
            error_line(:, k) = data(sub2ind(size(data), lane(j).line(2,
:), lane(j).line(1, :), repmat(k, [1 lane(j).length])))';
        end
        error_line = error_line - lane(j).bg;
        error_line = error_line .* error_line;
        error_line = sum(error_line, 2);
        lane(j).tracked_line = error_line < threshold;
        lane(j).tracked_line = medfilt1(im2double(lane(j).tracked_line),
median);

        cars_index = 1;

        if lane(j).tracked_line(3)
            if ~lane(j).saw_car
%                disp(sprintf('Found new car in lane %d: [%02d, %02d,
%02d]', j, lane(1).num_cars, lane(2).num_cars, lane(3).num_cars));
                lane(j).saw_car = true;

                tracked_offset = 3 + 1;
                car_indices = find(~lane(j).tracked_line(tracked_offset :
end));

                if car_indices
                    lane(j).num_cars = lane(j).num_cars + 1;
                    car_bottom = tracked_offset + car_indices(1) - 1;
                    tracked_offset = tracked_offset + car_indices(1) + 1;
                    car_indices =
find(lane(j).tracked_line(tracked_offset : end));

                    if car_indices
                        car_top = tracked_offset + car_indices(1) - 2;
                    else
                        car_top = length(lane(j).tracked_line);
                    end

                    road_coords = ScreenToRoad(TM, [lane(j).line(:,
car_top) lane(j).line(:, car_bottom)]);
                    road_distance = road_coords(:, 1) - road_coords(:,
2);
                    road_distance = sqrt(sum(road_distance .*
road_distance));

                    if road_distance > min_car_length
                        cars(1).id = lane(j).num_cars;
                        cars(1).bottom = car_bottom;
                        line(lane(j).line(1, [car_bottom car_top]),
lane(j).line(2, [car_bottom car_top]));
                        cars_index = 2;
                    end
                end
            end
        else
            if lane(j).saw_car
                lane(j).saw_car = false;
            end
        end

        if lane(j).cars(1).id
            for k = 1 : length(lane(j).cars),
                tracked_offset = lane(j).cars(k).bottom;
                car_indices = find(~lane(j).tracked_line(tracked_offset :
end));

                if (car_indices)
                    cars(cars_index).id = lane(j).cars(k).id;
                    cars(cars_index).bottom = tracked_offset +
car_indices(1) - 1;

                    road_coords = ScreenToRoad(TM, [lane(j).line(:,
cars(cars_index).bottom) lane(j).line(:, lane(j).cars(k).bottom)]);
                    road_distance = road_coords(:, 1) - road_coords(:,
2);
                    road_distance = sqrt(sum(road_distance .*
road_distance));

                    speed = road_distance * camera_fps / 3280.84 * 3600;
                    cars(cars_index).speed = [speed
lane(j).cars(k).speed];
```

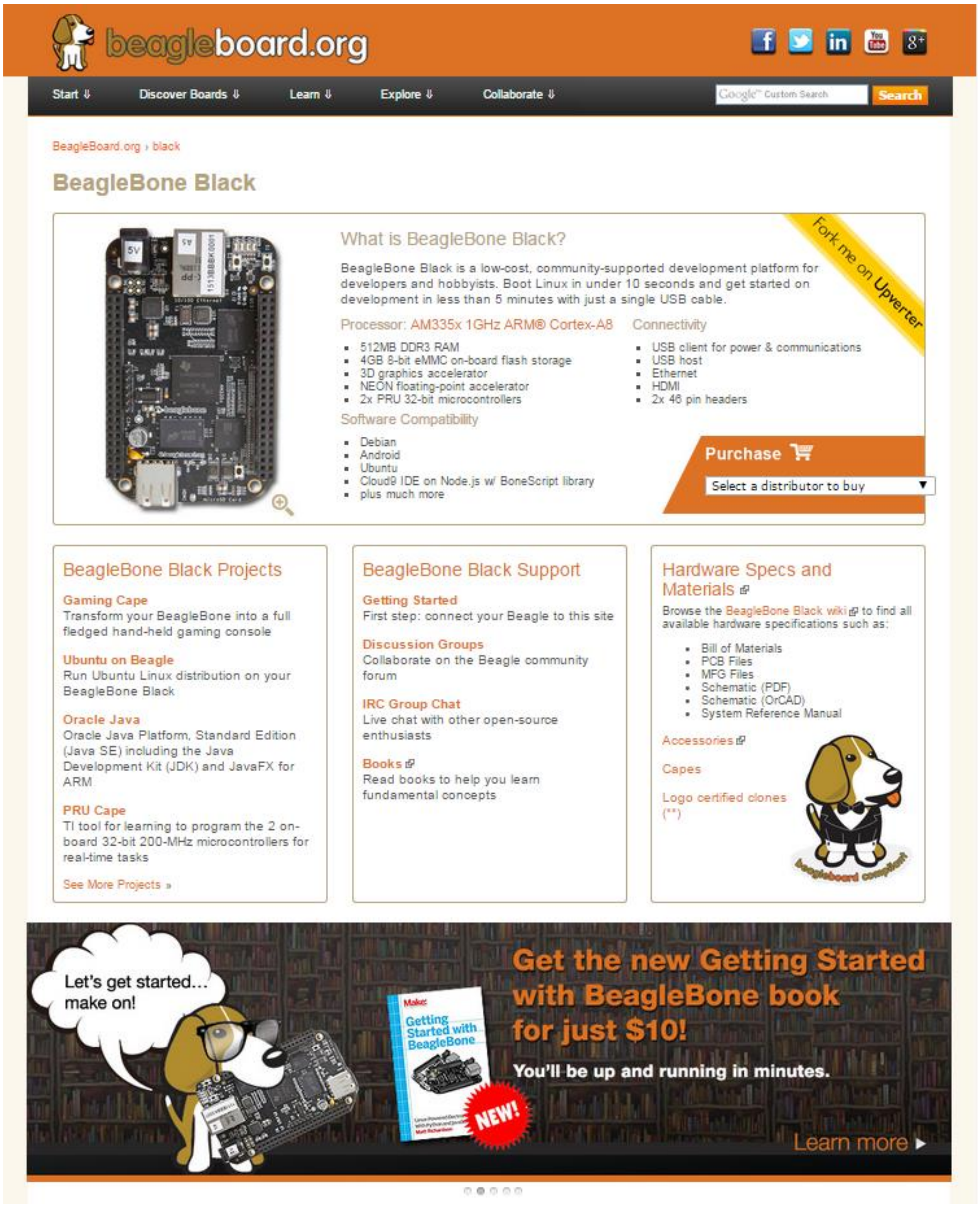
```
                    set(text(lane(j).line(1, cars(cars_index).bottom), 
lane(j).line(2, cars(cars_index).bottom), sprintf('(%d)', 
cars(cars_index).id)), 'Color', [1 0 0], 'HorizontalAlignment', 
'center');
                    set(text(lane(j).line(1, cars(cars_index).bottom), 
lane(j).line(2, cars(cars_index).bottom) + 18, sprintf('%.0f KPH', 
mean(cars(cars_index).speed))), 'Color', [0 1 0], 'HorizontalAlignment', 
'center');
                    cars_index = cars_index + 1;
                else
%                    disp('Car disappeared');
                    lane(j).flow = lane(j).flow + 
mean(lane(j).cars(k).speed);
                    lane(j).done_cars = lane(j).done_cars + 1;
                    update_stats = true;
                end
            end
        end

        lane(j).cars = cars;
    end

    if update_stats
        for j = 1 : num_lanes,
            if lane(j).done_cars > 0
                set(lane(j).text, 'String', {sprintf('%d cars', 
lane(j).done_cars), '', sprintf('%.1f', lane(j).flow / 
lane(j).done_cars), 'KPH'});
            end
        end
    end

    pause(delay);
end
```

Function Calls

```
bgimage = BackgroundDetect('data', 0, 1000, 10, .1, true, true);

world_quad_width=15;
world_quad_height=40;

[TM, lane_x, lane_start, lane_end] = CalibrateFunction(bgimage, 15, 40);

TrackCars('ready3', 0, 1000, bgimage, 10, 15, .001, 20, 0, lane_x, 
lane_start, lane_end, TM)
```

APPENDIX B

BEAGLEBONE WEBSITE



beagleboard.org

Start ⇓ Discover Boards ⇓ Learn ⇓ Explore ⇓ Collaborate ⇓ Google™ Custom Search Search

BeagleBoard.org › black

BeagleBone Black

What is BeagleBone Black?

BeagleBone Black is a low-cost, community-supported development platform for developers and hobbyists. Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.

Fork me on Upverter

Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 46 pin headers

Software Compatibility

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

Purchase

Select a distributor to buy

BeagleBone Black Projects

Gaming Cape
Transform your BeagleBone into a full fledged hand-held gaming console

Ubuntu on Beagle
Run Ubuntu Linux distribution on your BeagleBone Black

Oracle Java
Oracle Java Platform, Standard Edition (Java SE) including the Java Development Kit (JDK) and JavaFX for ARM

PRU Cape
TI tool for learning to program the 2 on-board 32-bit 200-MHz microcontrollers for real-time tasks

See More Projects »

BeagleBone Black Support

Getting Started
First step: connect your Beagle to this site

Discussion Groups
Collaborate on the Beagle community forum

IRC Group Chat
Live chat with other open-source enthusiasts

Books
Read books to help you learn fundamental concepts

Hardware Specs and Materials

Browse the BeagleBone Black wiki to find all available hardware specifications such as:

- Bill of Materials
- PCB Files
- MFG Files
- Schematic (PDF)
- Schematic (OrCAD)
- System Reference Manual

Accessories

Capes

Logo certified clones (**)

beagleboard compliant

Let's get started... make on!

Get the new Getting Started with BeagleBone book for just $10!

You'll be up and running in minutes.

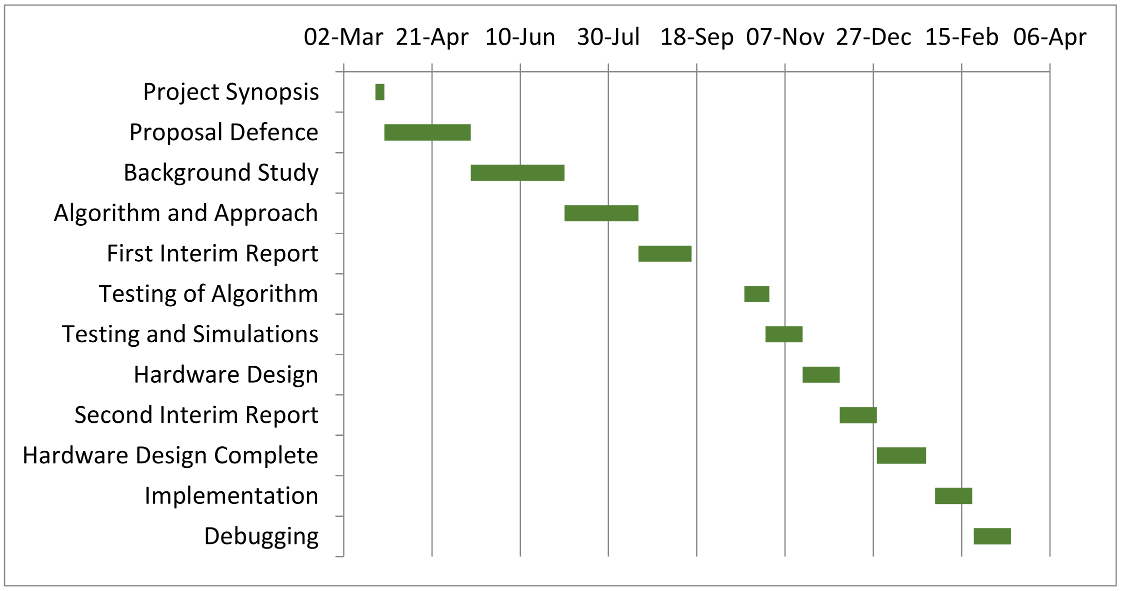
Learn more ▸

APPENDIX C

TIMELINE

Timeline

Task			
Start Date	End Date	Description	Duration (Days)
20-Mar	25-Mar	Project Synopsis	5
25-Mar	13-May	Proposal Defense	49
13-May	5-Jul	Background Study	53
5-Jul	16-Aug	Algorithm and Approach	42
16-Aug	15-Oct	First Interim Report	30
15-Oct	27-Oct	Testing of Algorithm	14
27-Oct	17-Nov	Testing and Simulations	21
17-Nov	8-Dec	Hardware Design	21
8-Dec	29-Dec	Second Interim Report	21
29-Dec	31-Jan	Hardware Design Complete	28
31-Jan	22-Feb	Implementation	21
22-Feb	15-Mar	Debugging	21



APPENDIX D

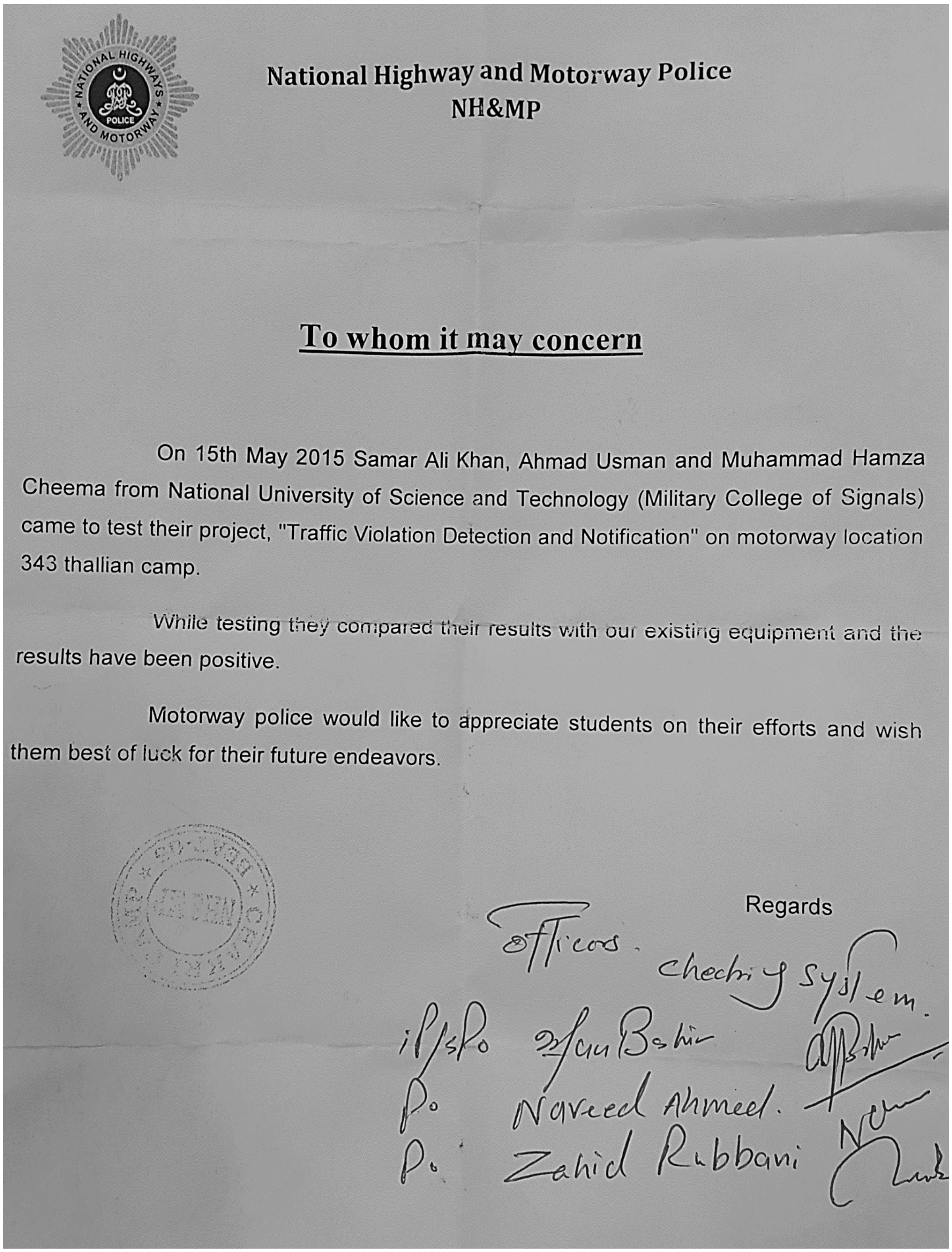
COST BREAKDOWN

Cost Breakdown

Items	Cost
Beagle-bone Black	10000
Logitech B525	5000
HDMI cable	800
Printouts	4000
USB Hub	500
USB Extension	500
Miscellaneous	5000
Total	**25800**

APPENDIX E

LETTER OF APPRECIATION FROM MOTORWAY POLICE



National Highway and Motorway Police
NH&MP

<u>To whom it may concern</u>

On 15th May 2015 Samar Ali Khan, Ahmad Usman and Muhammad Hamza Cheema from National University of Science and Technology (Military College of Signals) came to test their project, "Traffic Violation Detection and Notification" on motorway location 343 thallian camp.

While testing they compared their results with our existing equipment and the results have been positive.

Motorway police would like to appreciate students on their efforts and wish them best of luck for their future endeavors.

Regards

Officers. checking system.

i/sPo Irfan Bashir [illegible]

Po Naveed Ahmed. [illegible]

Po Zahid Rabbani [illegible]