A PROJECT REPORT
ON
“INTELLIGENT TRAFFIC LIGHT CONTROL USING IMAGE PROCESSING”

Submitted to

Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur

In partial fulfilment of the requirement for the degree of

BACHELOR OF ENGINEERING

in

Electronics & Telecommunication Engineering

Submitted By

Prateek Joshi **Pushkar Paranjpe**

Akshay Bahe **Sumit Manmode**

Under the Guidance of

Asst.Prof. Pradnya Maturkar

Department of Electronics and Telecommunication Engineering

Rajiv Gandhi College of Engineering & Research

Nagpur - 441110

2019-2020

CERTIFICATE OF APPROVAL

Certified that the project report entitled "**INTELLIGENT TRAFFIC LIGHT CONTROL USING IMAGE PROCESSING**" has been successfully completed by **PRATEEK JOSHI, PUSHKAR PARANJPE, AKSHAY BAHE, SUMIT MANMODE** under the guidance of **Mrs. PRADNYA MATURKAR** in recognition to the partial fulfilment for the award of the degree of Bachelor of Engineering in Electronics & Telecommunication Engineering, Session 2019-2020, **Rajiv Gandhi College of Engineering & Research, Nagpur*(An Institution Affiliated to RashtrasantTukdoji Maharaj Nagpur University)***

Mrs. Pradnya A. Maturkar
(Guide)
[Asst. Prof. E.T.C Dept,
R.G.C.E.R]

Dr.Manisha Khorgade
[HOD, E.T.C Dept,
R.G.C.E.R]

Dr.Manali Kshrisagar
[Principal, R.G.C.E.R]

Department of Electronics and Telecommunication Engineering

Rajiv Gandhi College of Engineering & Research
Nagpur - 441110
2019-2020

ACKNOWLEDGEMENT

We are extremely thankful to our guide Prof. Pradnya Maturkar under whom our project took the shape of reality from mere idea. We are thankful to our guide for enlightening us with her precious guidance and constant encouragement. We thank our guide for providing us with ample support and valuable time. We are indebted to our guide who, constantly provided us stimulus to reach our goals.

We are grateful to Dr.Manisha Khorgade, Head of Department, Electronics and Telecommunication Engineering, RGCER, for his kind cooperation timely help.

We express our gratitude towards Dr. Manali Kshirssagar, Principal, RGCER, for his never ending support and motivation.

Lastly we would like to thank all those who were directly or indirectly related to our project and extended their support to make the project successful.

Prateek Joshi

Pushkar Paranjpe

Akshay Bahe

Sumit Manmode

Electronics and Telecommunication Engineering

Rajiv Gandhi College of Engineering & Research, Nagpur

ABSTRACT

Traffic is the major problem which the country faces today this is because of the increase in the number of vehicles. The increase in the number of vehicles resulting in the need for a smart system that could efficiently handle traffic congestion based on the density of traffic. Here also discussing Canny edge detection and their advantages and disadvantage. Today, the major problems that cities face are that of traffic. Empty roads are becoming a luxury nowadays. Heavy traffic eventually means high waiting time at the traffic signal. The traffic light timers at squares are set according to the average amount of traffic flow at these squares. This results in traffic congestion at certain junctions due to sudden changes in traffic density. These sudden changes in traffic densities resulting in the need for a smart system that could efficiently handle traffic congestion based on the density of traffic. This paper discusses some of the existing traffic light control systems, their drawbacks as well as our proposed system where we are trying to avoid the drawbacks of existing systems. The main motto of our project is to provide a smooth flow of traffic and to reduce the time duration for which one has to stand at the traffic signal. We have achieved this by using raspberry pi processors and pi cameras corresponding to the four traffic lanes. The cameras will monitor the density of vehicles of standing at the signal using image processing. In addition to that, we will be providing automated number plate recognition as well as speed monitoring in the night time.

Keywords: Image Processing, Raspberry pi 3, python, RTC Module, Traffic Signals.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

As the population of modern cities is increasing day by day due to which vehicular travel is increasing which leads to the congestion problem. Traffic congestion has been causing many critical problems and challenges in the major and most populated cities. The increased traffic has to lead to more waiting times and fuel wastages. Due to these congestion problems, people lose time, miss opportunities, and get frustrated. Fast transportation systems and rapid transit systems are some of the important aspects of economic developments for any nation. Mismanagement and traffic congestion occurs due to long waiting times, loss of fuel and money.

Thus it is necessary to have a fast, economical and efficient traffic control system for national development. The traffic monitoring authority needs to find new methods to overcome these issues because the number of vehicles is rapidly increasing day by day recently to improve traffic flow and safety, intelligent control methods are being implemented. Intelligent traffic control will become an important issue in the future as the number of road users constantly increases, and resources provided by current infrastructures are limited. Traffic congestion may result due to heavy traffic at a junction.

But no technique is perfect by itself as the real-time situations are generally continuously changing and the system has to adapt itself to change in the continuously changing circumstances. We tried to provide some traffic management strategy to fit in to continuously changing real-time traffic scenarios. In this system, time is assigned to traffic light of a particular lane according to the traffic density on the road. Limitations of Existing system

A. Heavy Traffic Jams With the increasing number of vehicles on the road, heavy traffic congestion has substantially increased in major cities. This happened usually at the main junctions commonly in the morning, before office hours and in the evening, after office hours. The main effect of this matter is the increased time-wasting of the people on the road.

B. No Traffic, But Still Need To Wait At certain junctions, sometimes even if there is no traffic, people have to wait. Because the traffic light remains red for the preset period, the road users should wait until the light turns green. This happens because the timer values are already set based on average traffic on that lane.

1.2 PROBLEM STATEMENT

In a survey, we found that there is no provision for ANPR and speed monitoring in the night time. To solve this issue the system prototype is designed which provides the optimum traffic control in real-time by using Image Processing as well as night time speed monitoring.

1.3 THESIS OBJECTIVE

The image processing eliminates errors because image processing modules can be trained to detect certain objects with optimum perfection. The system also presents real-time traffic control based on the count of vehicles which will result in traffic fluency. The Automatic Number-plate Recognition system is used to detect traffic rules violation.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

One of the primary objectives of the traffic control system is to provide fluent traffic flow irrespective of the variation in traffic density. When surveyed the existing systems to know more about the drawbacks so that we can overcome them in our system.

The first system1 uses an ultrasonic sensor controlled by the microcontroller. This system places the ultrasonic sensor at all the intersections of the square and after detecting the vehicle it checks for the timer as well as traffic signal status of that lane but, in certain places the ultrasonic sensors may be affected by the weather conditions.

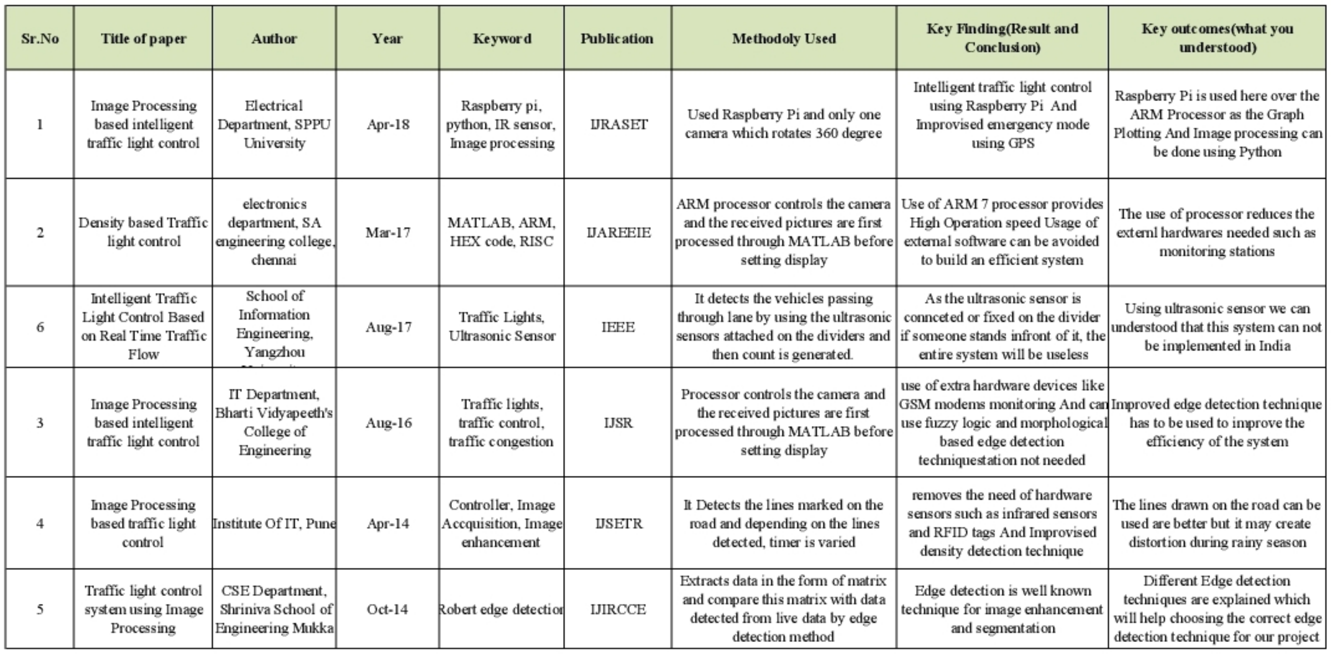
The second system2 uses the lines drawn on each lane at several distances from the center of the intersection. Here a camera detects the number of lines visible to detect the traffic density. The number of lines visible is inversely proportional to the density of the traffic, the system may have an increase in distortion as the lines fade away and in some cities, the system may be affected in rainy seasons.

The third system3 uses the ARM processor to control the cameras and the received pictures are first processed through MATLAB before setting the traffic signals. Use of ARM 7 processor provides High Operation speed Usage of external software can be avoided to build an efficient system, use of MATLAB may require the dedicated desktops at each intersection.

The fourth system4 uses the raspberry pi and the pi-camera located at the center of the intersection to read the traffic densities by using image processing. Raspberry pi is used here over the ARM processor as the graph plotting and image processing can be done using Python and it requires no external dedicated device, as the single pi-camera is rotated the system may affect the speed of operation at certain intersections with low traffic densities.

Our system provides regular traffic flow control as well as the vehicle speed monitoring system. Also, the system will monitor for the traffic rules violation and the number plate of the vehicles violating the traffic rule will be recorded in the database.

Table showing overview of Literature survey is given below:



Sr.No	Title of paper	Author	Year	Keyword	Publication	Methodoly Used	Key Finding(Result and Conclusion)	Key outcomes(what you understood)
1	Image Processing based intelligent traffic light control	Electrical Department, SPPU University	Apr-18	Raspberry pi, python, IR sensor, Image processing	IJRASET	Used Raspberry Pi and only one camera which rotates 360 degree	Intelligent traffic light control using Raspberry Pi And Improvised emergency mode using GPS	Raspberry Pi is used here over the ARM Processor as the Graph Plotting And Image processing can be done using Python
2	Density based Traffic light control	electronics department, SA engineering college, chennai	Mar-17	MATLAB, ARM, HEX code, RISC	IJAREEIE	ARM processor controls the camera and the received pictures are first processed through MATLAB before setting display	Use of ARM 7 processor provides High Operation speed Usage of external software can be avoided to build an efficient system	The use of processor reduces the exterml hardwares needed such as monitoring stations
6	Intelligent Traffic Light Control Based on Real Time Traffic Flow	School of Information Engineering, Yangzhou	Aug-17	Traffic Lights, Ultrasonic Sensor	IEEE	It detects the vehicles passing through lane by using the ultrasonic sensors attached on the dividers and then count is generated.	As the ultrasonic sensor is connceted or fixed on the divider if someone stands infront of it, the entire system will be useless	Using ultrasonic sensor we can understood that this system can not be implemented in India
3	Image Processing based intelligent traffic light control	IT Department, Bharti Vidyapeeth's College of Engineering	Aug-16	Traffic lights, traffic control, traffic congestion	IJSR	Processor controls the camera and the received pictures are first processed through MATLAB before setting display	use of extra hardware devices like GSM modems monitoring And can use fuzzy logic and morphological based edge detection techniquestation not needed	Improved edge detection technique has to be used to improve the efficiency of the system
4	Image Processing based traffic light control	Institute Of IT, Pune	Apr-14	Controller, Image Acquisition, Image enhancement	IJSETR	It Detects the lines marked on the road and depending on the lines detected, timer is varied	removes the need of hardware sensors such as infrared sensors and RFID tags And Improvised density detection technique	The lines drawn on the road can be used are better but it may create distortion during rainy season
5	Traffic light control system using Image Processing	CSE Department, Shriniva School of Engineering Mukka	Oct-14	Robert edge detectior	IJIRCCE	Extracts data in the form of matrix and compare this matrix with data detected from live data by edge detection method	Edge detection is well known technique for image enhancement and segmentation	Different Edge detection techniques are explained which will help choosing the correct edge detection technique for our project

Table 1: Literature Susrvey

CHAPTER 3

WORK DONE

3.1 METHODOLOGY

In the Flow chart given below, the comparison with given data refers to comparing the captured image with the custom classifiers made to identify vehicles in the image and deciding the density count accordingly. For image detection, we have used the canny edge detection method.

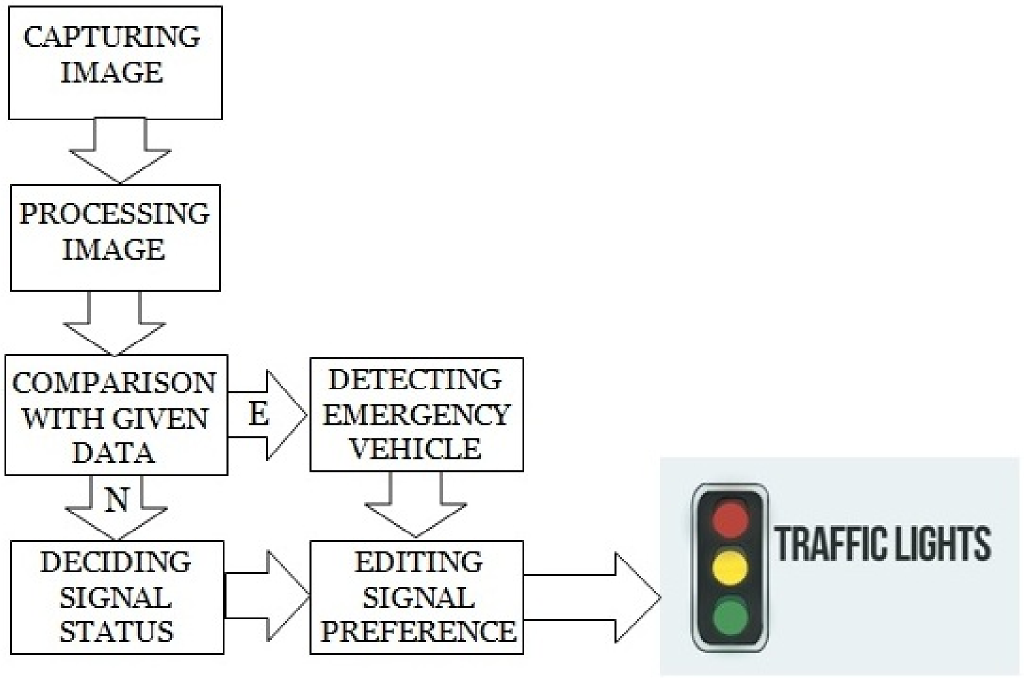

CAPTURING IMAGE
PROCESSING IMAGE
COMPARISON WITH GIVEN DATA
E
DETECTING EMERGENCY VEHICLE
N
DECIDING SIGNAL STATUS
EDITING SIGNAL PREFERENCE
TRAFFIC LIGHTS

Fig 1: Flow chart of Project

1) Image Acquisition: Generally an image is a two-dimensional function f(x,y) (where x and y are plane coordinates).The amplitude of the image at any point say f, is called intensity of the image. It is also called the gray level of image at that point. We need to convert these x and y values to finite discrete values to form a digital image. Each digital image composed of finite elements and each finite element is called a pixel.
2) Image Resizing/Scaling: Image scaling is used here to provide a constant number of pixels for each image. The resolution of the image received may be different for an instance so, to make the algorithm quite simpler and to ease the processes that are to be performed on the received image.
3) RGB to GRAY Conversion: color images are often stored as three separate image matrices; one storing the amount of red (R) in each pixel, one the amount of green (G) and one the amount of blue (B). We call such color images as stored in an RGB format. In grayscale images, however, we do not differentiate how much we emit different colors; we emit the same amount in every channel. We will be able to differentiate the total amount of emitted light for each pixel. In simple terms, it is used to reduce the complexity in the image

and ease the edge detection as far as possible because it is simpler to scan a single matrix than three different matrices.
4) Image Enhancement: Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further analysis. Here we are blurring the image using the bilateral filter because for edge detection, we only need edge values and thus removing details while preserving edges reduces the noise formed by unwanted false contours.
5) Edge Detection: Edge detection is the name for a set of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply or, more technically, has discontinuities or noise. The points at which image brightness alters sharply are typically organized into a set of curved line segments termed edges. There are different edge detection techniques but, we have used the canny edge detection. Canny edge detection in python is done by giving the threshold values to the defined function cv2.canny and selecting the image on which it is to be executed.
6) Image Matching: We have used a totally different approach to image matching. Comparing a reference image with the real-time image pixel by pixel Some of the procedures mentioned above are used for automatic number-plate detection and some are used to find out the densities of the lane. The first three processes are used with the custom classifiers to detect the count of the vehicle whereas the edge detection is performed on the image received from the camera for ANPR detection.

3.1.1WORKING PRINCIPAL

Here is the working principle of the project that for the first iteration of the day when the system is initiated the all four timers will click the pictures and the priorities will be decided according to the density of the traffic. Then when the lane with the highest priorities is green the priorities of the other three lanes will be re-decided. And when the second lane(with respect to priorities) is green the priorities for the remaining two lanes will be re-decided.

When the one signal is green the cameras pointing to the remaining three lanes will check and detect if someone violated the traffic rules and the info. About that vehicle is send for the E-challan purpose.

By effectively using the E-Challan system, the owner of the vehicle who violated the traffic rules will instantly receive the challan. And thus it will be beneficial to reduce the percentage of the traffic rules violation and indirectly reducing the accident count.

The Real Time Clock Module is introduced for following the RTO traffic light timelines of 9 am. To 9 pm. It is also important after 9 pm the night monitoring system will be activated which will monitor the vehicle speed at night, and if the speed of vehicle exceeds the standard limits, that will be noted.

3.1.3 Block Diagram

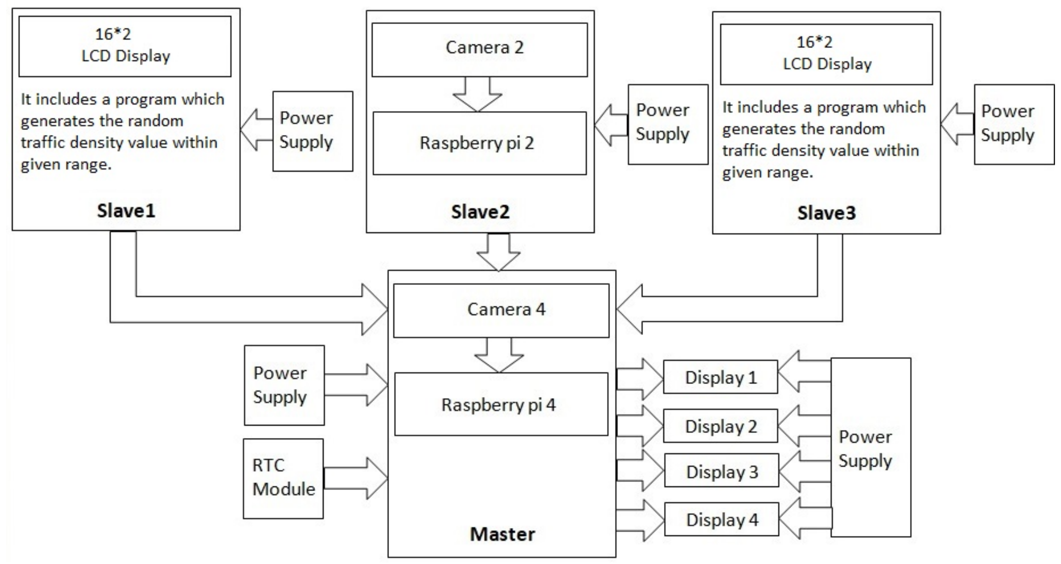


Fig.2: Block Diagram of Proposed System

In this system, the two raspberry pi with a camera mounted on it is placed in two corners of the intersection. The cameras are located such that they face the opposite lane from the lane they are located on. One of the two raspberry pis will be the master raspberry pi and it will control every process of the system, the remaining one raspberry pi are slaves which are used only to detect the traffic density by image processing. All the raspberry pi are connected to each other by using the inbuilt wifi module and are interfaced by connecting them on the same wifi router or hotspot terminal.
The system will restart itself every day at 9 am and for the first iteration all the two cameras and two lcd will capture the current traffic densities by using image processing and the count is sent to the master raspberry pi. Then it will set the traffic signals accordingly. When one of the lanes is green, the cameras facing the remaining lanes will monitor both the traffic rules violation as well as current traffic density to make it real-time and if the traffic rules are violated then the number plate will be recorded. For night time the same system will monitor for the over-speeding vehicles.

3.2 CIRCUIT DIAGRAM

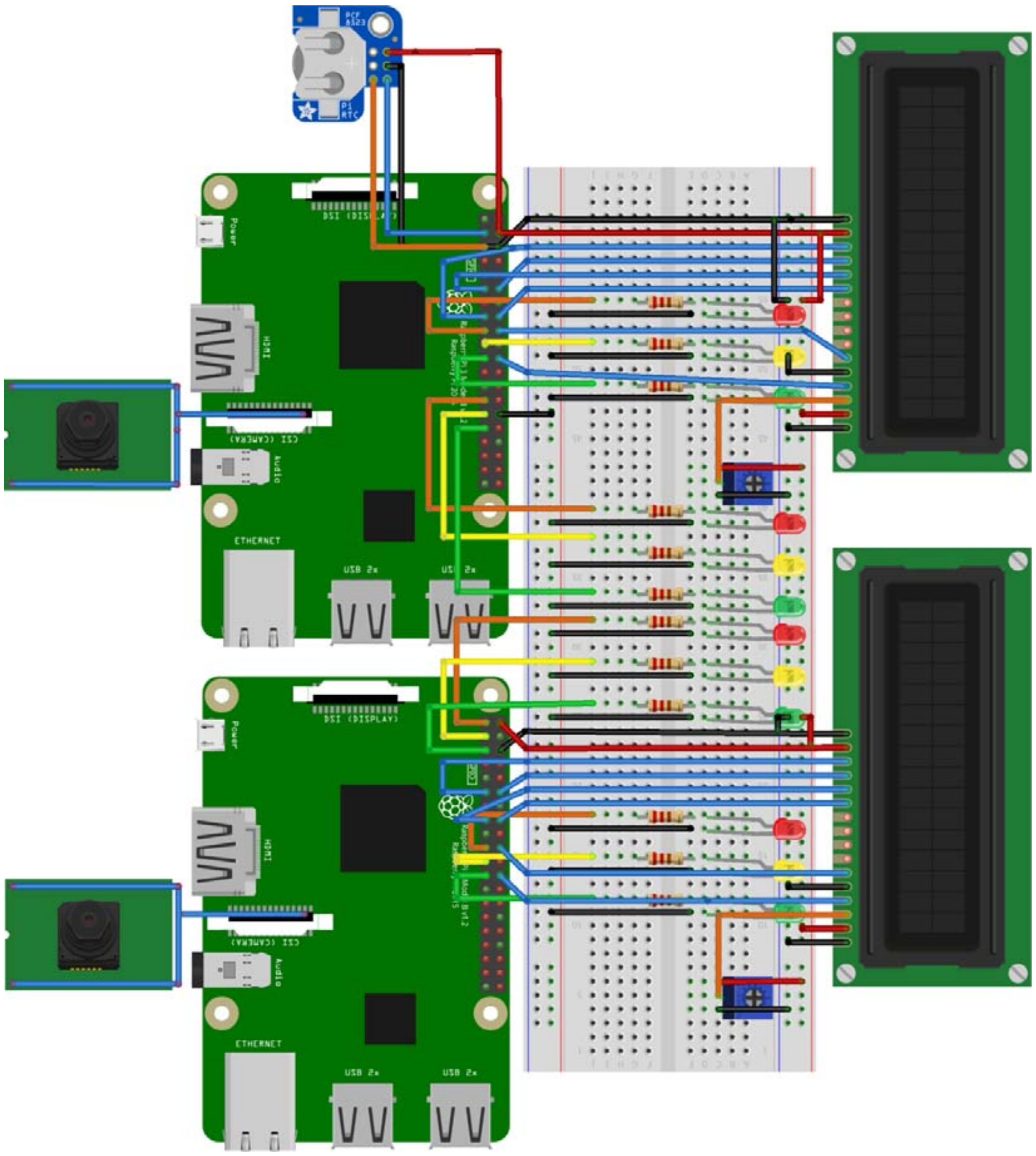


Fig 3: Circuit Diagram of Proposed System

Figure above shows the circuit diagram of our proposed system. The Real Time Clock Model is connected as: pin SCL on pin1 of raspberry pi3, pin SDA on pin3 of raspberry pi3, pin VCC on pin5 of raspberry pi3 and pin GND on pin9 of raspberry pi3.The Cameras, camera1 is connected on the first raspberry pi through camera connector similarly, camera2 is connected on the second raspberry pi.

The two liquid crystal displays are used to display the random counts. The lcds are used for the 8 bit data thus D4 to D7 (i.e., pin 7 to pin 10) are not used. Pins of lcd1, pin1(Vss) and pin16(Gnd) is connected to pin6 of raspberry pi. Pins of lcd1, pin2(Vdd) and pin15(Anode) is connected to pin4 of raspberry pi. Pin3 of lcd1 is connected to potentiometer which controls the contrast. Pin5 of lcd1 is connected to pin6 of raspberry pi. Pin4 of lcd1 is connected to pin22 of raspberry pi. Pin6 is connected to pin18 of raspberry pi. Pin11 of lcd1 is connected to pin16 of raspberry pi. Pin12 of lcd1 is connected to pin 11 of raspberry pi. Pin13 of lcd1 is connected to pin12 of raspberry pi. Pin14 of lcd1 is connected to pin15 of raspberry pi.

Pins of lcd2, pin1(Vss) and pin16(Gnd) is connected to pin6 of raspberry pi. Pins of lcd2, pin2(Vdd) and pin15(Anode) is connected to pin2 of raspberry pi. Pin3 of lcd2 is connected to potentiometer which controls the contrast. Pin5 of lcd2 is connected to pin6 of raspberry pi. Pin4 of lcd2 is connected to pin24 of raspberry pi. Pin6 is connected to pin18 of raspberry pi. Pin11 of lcd2 is connected to pin16 of raspberry pi. Pin12 of lcd2 is connected to pin 11 of raspberry pi. Pin13 of lcd2 is connected to pin12 of raspberry pi. Pin14 of lcd2 is connected to pin15 of raspberry pi.

3.3 Requirement Analysis (Hardware and Software)

3.3.1 Hardware Used

A. RASPBERRY PI 3:

RASPBERRY PI 3 is a development board in PI series. It can be considered as a single board computer that works on LINUX operating system. The board not only has tons of features it also has terrific processing speed making it suitable for advanced applications. PI board is specifically designed for hobbyist and engineers who are interested in LINUX systems and IOT (Internet of Things).

Raspberry Pi-3 Pin Configuration

PIN GROUP	PIN NAME	DESCRIPTION
POWER SOURCE	+5V, +3.3V, GND and Vin	+5V -power output +3.3V -power output GND – GROUND pin
COMMUNICATION INTERFACE	UART Interface(RXD,TXD)[(GPIO15,GPIO14)]	UART (Universal Asynchronous Receiver Transmitter) used for interfacing sensors and other devices.

	SPI Interface(MOSI, MISO, CLK,CE) x 2 [SPI0-(GPIO10, GPIO9, GPIO11, GPIO8)] [SPI1--(GPIO20, GPIO19, GPIO21, GPIO7)]	SPI (Serial Peripheral Interface) used for communicating with other boards or peripherals.
	TWI Interface(SDA, SCL) x 2 [(GPIO2, GPIO3)] [(ID_SD,ID_SC)]	TWI (Two Wire Interface) Interface can be used to connect peripherals.
INPUT OUTPUT PINS	26 I/O	Although these some pins have multiple functionsthey can be considered as I/O pins.
PWM	Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19	These 4 channels can provide PWM (Pulse Width Modulation) outputs. *Software PWM available on all pins
EXTERNAL INTERRUPTS	All I/O	In the board all I/O pins can be used as Interrupts.

Raspberry Pi 3 Technical Specifications:

Microprocessor	Broadcom BCM2837 64bit Quad Core Processor
Processor Operating Voltage	3.3V
Raw Voltage input	5V, 2A power source
Maximum current through each I/O pin	16mA
Maximum total current drawn from all I/O pins	54mA
Flash Memory	16Gbytes SSD memory card

(Operating System)	
Internal RAM	1Gbytes DDR2
Clock Frequency	1.2GHz
GPU	Dual Core Video Core IV® Multimedia Co-Processor. Provide Open GLES 2.0, hardware-accelerated Open VG, and 1080p3 H.264 high- profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs With texture filtering and DMA infrastructure.
Ethernet	10/100 Ethernet
Wireless Connectivity	BCM43143 (802.11 b/g/n Wireless LAN and Bluetooth 4.1)
Operating Temperature	-40ºC to +85ºC

Board Connectors

Name	**Description**
Ethernet	Base T Ethernet Socket
USB	2.0 (Four sockets)
Audio Output	3.5mm Jack and HDMI
Video output	HDMI
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connecto with two data lanes and a clock lane.
Memory Card Slot	Push/Pull Micro SDIO

RASPBERRY Patform is most used after **ADRUINO**. Although overall applications of PI are less it is most preferred when developing advanced applications. Also the RASPBERRY PI is an open source platform where one can get a lot of related information so you can customize the system depending on the need.

Here are few examples where RASPBERRY PI 3 is chosen over other microcontrollers and development boards:

1. Where the system processing is huge. Most ARDUINO boards all have clock speed of less than 100MHz, so they can perform functions limited to their capabilities. They cannot process high end programs for applications like Weather Station, Cloud server, gaming console etc. With **1.2GHz clock speed** and **1 GB RAM RASPBERRY PI** can perform all those advanced functions.

2. Where wireless connectivity is needed. RASPBERRY PI 3 has wireless LAN and Bluetooth facility by which you can setup WIFI HOTSPOT for internet connectivity. For **Internet of Things** this feature is best suited.

3. RASPBERRY PI had dedicated port for connecting touch LCD display which is a feature that completely omits the need of monitor.

4. RASPBERRY PI also has dedicated camera port so one can connect camera without any hassle to the PI board.

5. RASPBERRY PI also has PWM outputs for application use.

6. Other boards of the range are RASPBERRY PI – 4, RASPBERRY PI – 2, RASPBERRY PI – 1, RASPBERRY PI – ZERO, RASPBERRY PI – 2 B+

Applications

- Hobby projects.
- Low cost PC/tablet/laptop
- IoT applications
- Media center
- Robotics
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Web camera
- Gaming
- Wireless access point
- Environmental sensing/monitoring (e.g. WEATHER STATION)

B. Real Time Clock:

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

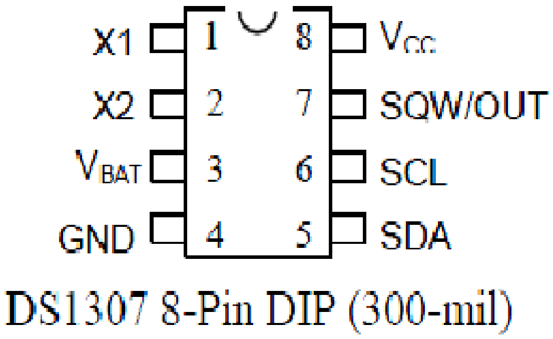


Fig.4: DS1307 pin out

Features:

1. Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
2. 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
3. Two-wire serial interface
4. Programmable square wave output signal
5. Automatic power-fail detect and switch circuitry
6. Consumes less than 500nA in battery backup mode with oscillator running
7. Optional industrial temperature range: -40°C to +85°C
8. Available in 8-pin DIP or SOIC
9. Underwriters Laboratory (UL) recognized

Signal Descriptions:

V_{CC}, GND – DC power is provided to the device on these pins. V_{CC} is the +5V input. When 5V is applied within normal limits, the device is fully accessible and data can be written and read. When a 3V battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited

V_{BAT} – Battery input for any standard 3V lithium cell or other energy source. Battery voltage must be held between 2.0V and 3.5V for proper operation. The nominal write protect trip point voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal.

SCL (Serial Clock Input) – SCL is used to synchronize data movement on the serial interface.

SDA (Serial Data Input/Output) – SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pull-up resistor.

SQW/OUT (Square Wave/Output Driver) – When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pull-up resistor. SQW/OUT will operate with either Vcc or Vbat applied.

X1, X2 – Connections for a standard 32.768khz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5pf. For more information on crystal selection and crystal layout considerations, please consult

Working of IC DS1307:

In the simple circuit the two inputs X1 and X2 are connected to a 32.768 kHz crystal oscillator as the source for the chip. VBAT is connected to positive culture of a 3V battery chip. Vcc power to the I2C interface is 5V and can be given using microcontrollers. If the power supply Vcc is not granted read and writes are inhibited.

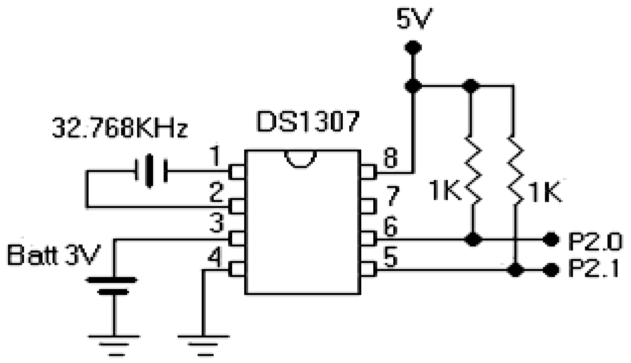


Fig.5: Circuit Details of RTC Module using DS1307

START and STOP conditions are required when a device wants to establish communication with a device in the I2C network. By providing a device identification code and a register address, we can implement the START condition to access the device. The registers can be accessed in serial order until a STOP condition is implemented. The START condition and STOP condition when the DS1307 I2C communication with the microcontroller is shown in the figure below.

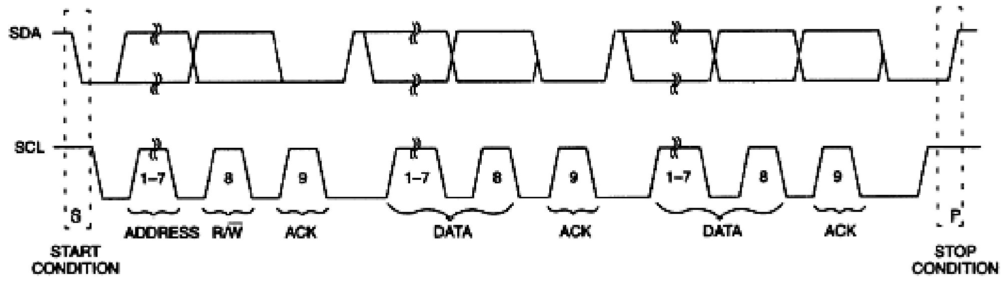
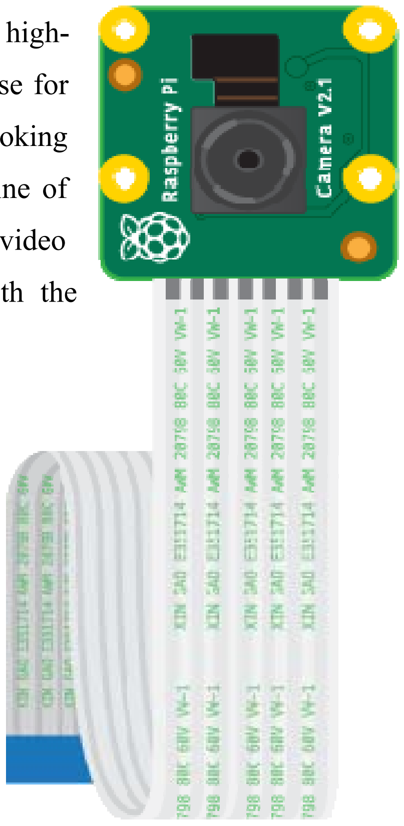


Fig.6: Waveforms showing Working of SDA & SCL pins

C. Pi-Camera Module:

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion and other video cleverness. You can also use the libraries we bundle with the camera to create effects.



If you're interested in the nitty-gritty, you'll want to know that the module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Pi-camera Python library. The camera module is very popular in home security applications, and in wildlife camera traps. You can also use it to take snapshots.

Fig.5: PiCamera Module

Features

- 5MP sensor
- Wider image, capable of 2592x1944 stills, 1080p30 video
- 1080p video supported
- CSI
- Size: 25 x 20 x 9 mm

Camera Details

The camera consists of a small (25mm by 20mm by 9mm) circuit board, which connects to the Raspberry Pi's Camera Serial Interface (CSI) bus connector via a flexible ribbon cable. The camera's image sensor has a native resolution of five megapixels and has a fixed focus lens. The software for the camera supports full resolution still images up to 2592x1944 and video resolutions of 1080p30, 720p60 and 640x480p60/90.

Basic Operation:

The basic commands are `raspivid` and `raspistill`, which (as the names suggest) are for capturing video and still images, respectively. Each of these commands can accept a number of parameters after them. Several parameters are available, but the most basic are:

- `-o` or `-output`, for setting the output filename
- `-t` or `-timeout`, for setting the time in milliseconds for which a preview will display (the default is 5 seconds, and when using `raspistill`, it will capture the last frame of the preview and save it to the specified filename. When using `raspivid`, the `-t` parameter defines the capture time.)

Therefore, to display a 5 second preview and then save a jpeg picture called `test.jpeg`, the command would be:

```
$ raspistill -o test.jpeg
```

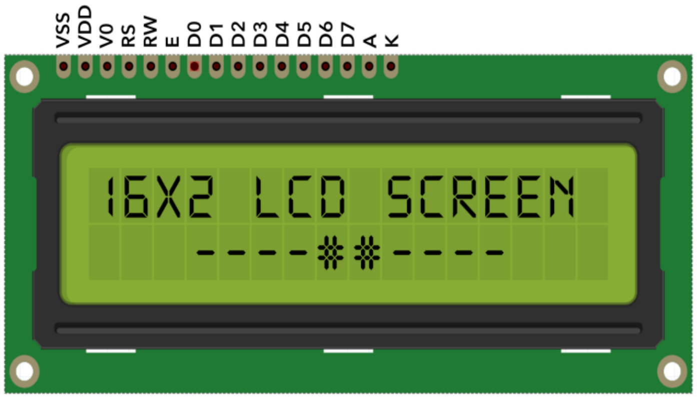
To take a 30-second video and save it in h264 format, use the command:

```
$ raspivid -o test.h264 -t 30000
```

These are the simple commands used in the terminal of Raspberry Pi to check if the the camera module is fully functioning or not

D. Liquid Crystal Display:

A 1602 liquid crystal display (LCD) module consists of 16 columns and two rows. This means that the LCD can display 16 characters per line (16 Characters x 2 Line). The LCD has found a wide range of applications because of its cheap price and the ease to program. The LCD can display the alphabet and number characters making it an alphanumerical display module. Each character is built on a 5×8 pixels dots. It is also possible to display special and custom generated characters on this type of display.



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

Fig.6: Liquid Crystal Display (16*2)

The VSS and the VDD pins are used to power the LCD and are connected to the ground and +5 volts respectively. VO pin is for adjusting the contrast of the LCD and is normally connected to a potentiometer (variable resistor). RS pin is the Register Select pin. This will select data register when high, and command register when low. Read/Write (R/W) pin is used to toggle the LCD between the read from and write to the register operations.

Features:

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit
- These are obtainable in Blue & Green Backlight
- It displays a few custom generated characters

When set low, R/W pin will write to the register. While on the other hand, when set high, it will read from the register. This pin is normally grounded for the data writing operations to the LCD as in our example below. Enable (E) pin is the data read/write operations enable signal. Pins D0 – D7 are for data transfer. D0 – D3 is restricted to 8-bit data. For our case, we will be using pins D4 – D7 for both the 4 bits and 8 bits data transfer. BLA and BLK are for powering the LCD backlight with +5volts and ground respectively.

Modes of Operation:

A. To initialize character lcd in 4 bit mode we send value hex 0x20 to command register of lcd. 0x20 tells the lcd controller that we want to communicate in 4-bit mode. Lcd is 1 line (has 1 row) and we want character shape displayed in 5×7 matrix. If our character lcd has 2 lines (rows) we will send 0x28 instead of 0x20. It tells the lcd controller that we want 4 bit communication and character size is between 5×7 dot matrix.

B. 4-bit mode make use of only just four data pins D4-D5-D6-D7.In 4-bit mode character is displayed on lcd in two pulse signals. First the higher four nibbles of a character are sent to the lcd with an enable stroke. Than the lower four nibbles are send with enable stroke. Since two pulse (enable) signals are required to display a single character so 4-bit mode latency time is high.

3.4 SOFTWARE:

1) PYTHON: Python is a powerful modern computer programming language. It bears some similarities to Fortran, one of the earliest programming languages, but it is much more powerful than Fortran. Python allows you to use variables without declaring them (i.e., it determines types implicitly), and it relies on indentation as a control structure. Think of game programming, rendering graphics, GUI interfaces, web frameworks, and scientific computing. Many (but not all) of the things you can do in c can be done in python. Python is

generally slower at computations than C, but its ease of use makes Python an ideal language for prototyping programs and designing applications that aren't computationally intensive.
2) OPEN CV2: Open CV Stands for Open computer vision it is a source library of functions is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, Usage ranges from interactive art to mines inspection, stitching maps on the web or through advanced robotics.

a) Convert Color: cv2.cvtColor function is used to convert the color of the image from color to grayscale. This can also be done by adding parameter while reading the image, using this option image is read in grayscale instead of its original format.
b) The Image Read: cv2.imread is used to read the image from the given directory here we have to select the image to be read as well as its optional to select the format (original or grayscale) while reading the image.
c) Image Show: cv2.imshow is used to display the image; here the new name for the displayed image as well as the image to be displayed has to be selected.
d) Bilateral Filter: cv2.bilateralFilter is used to reduce the noise from the grayscale image. Here the images whose edges are to be detected as well as various threshold values are selected.
e) Canny Edge Detection5: The OpenCV library has an inbuilt function for the Canny edge detection named cv2.canny and here we need to provide the threshold values which are used to decide and draw contours on the edge detected image.
f) Pytesseract: The pytesseract is an application that is used for the object to text conversion. This software is used through the program by installing it in our Raspberry Pi and calling the app by giving the app location in the program.
g) YOLO V36: We have used YOLO (You Only Look Once) model of version 3 to customize the classifiers as the system may vary depending on the local vehicles, and the design of the number–plate of that local area. For that purpose we have used the trainer GUIs for training the customized classifiers.

Python Code to Generate Density Count and Display on LCD:

First of all as Liquid Crystal Display is used, here we are using predefined LCD library which is created by Adafruit called `Adafruit_CharLCD` and using it as '`LCD`' throughout the code The `random` library provides the use of different expressions used to generate random values as well as here we are importing function sleep from time library which is basically a delay function.

```
import Adafruit_CharLCD as LCD
from time import sleep
import random
```

While using the `CharLCD` library, the raspberry Pi gets the function of every pin of that lcd as it is already declared in the function. Thus we only have to declare on which GPIO pins the pins of LCD are connected. Also there is the `lcd_column` and `lcd_row` variables included which are used by the Raspberry pi to recognize size of the LCD Display

```
lcd_rs        = 25
lcd_en        = 24
lcd_backlight = 4
lcd_columns   = 16
lcd_rows      = 2
```

Here we are using 4 Bit Mode for LCD interfacing. The idea of 4 bit mode is introduced to save pins of microcontroller (serial communication LCD saves more pins). 4 bit mode interfaced lcd will be a bit slower than 8 bit mode. But this speed difference is not significant as LCDs are slow speed devices. Here we have used pins `d4` to `d7` for 4 bit mode.

```
lcd_d4        = 23
lcd_d5        = 17
lcd_d6        = 18
lcd_d7        = 22
```

After declaring all the needed conncetions the lcd is initialized by the following command, this command reads or uses the declarations previously done in this code to fill the variables in the instruction also, `lcd_backlight` is manually controlled according to the need. But for initialization its default values are considered.

```
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
lcd_d7, lcd_columns, lcd_rows, lcd_backlight)
```

For Generating Density count, its necessary to use this code in infinity loop this is done by

```
while True:
```

Before writing anything on the LCD it is needed to clear its contents at the start, this is done by by lcd.clear() command. Then just to check 'Hello World!' is printed with the delay of 2 seconds.

```
lcd.clear()
lcd.message('hello world!')
sleep(2)
```

As this code is to be used for the density count on second lane the count is to be stored in the Lane2 variable which is needed to be stored for its use in the main code. `random.randint()` is the function used to generate random values of given data-type in the given range. The data type is here declared by `randint` i.e., Random integer and the range is declared in round brackets separated by a ','. To print the number on LCD we considered it as a string to do that we converted it into string using `str()` function and then printed on the lcd by using `lcd.message()` command where data to be printed is written inside the round brackets within the Single Quotes. To be able to view the density count properly the delay of 15 seconds is provided at the end.

```
Lane2 = random.randint(5,20)

density = str(Lane2)

lcd.clear()

lcd.message('Count is:'+ (density))

sleep(15)
```

The program above generates the random counts for one of the four lanes and displays that density count on the LCD display as there is no camera for those two lanes. The same code is also used for Lane 4 where the only '`Lane2`'variable is changed with '`Lane4`' as the whole code will be in different functions for final code and we know same variables with limited access can be used in two different functions.

Python code to generate the density from camera:

To generate the density of cars from the image it is necessary to use computer vision. Here cv2 is the computer vision library for python which I had to install before working on the images. Python stores images in the form of matrix as picture is created by pixels, it stores value of each pixel in the matrix, and here numpy is the library used for the same

```
import cv2

import numpy as np
```

For detecting the number of cars from the given image, python should know what is car look like, it should have something (car data) to compare with the given data. The file 'cars.xml' in the command below is the file generated by teaching the python what is a car and correcting it over and over. Thus, accuracy of this program depends entirely on the content of the 'cars.xml' file. Where cascade classifier is the object used to access the file.

```
car_cascade = cv2.CascadeClassifier('cars.xml')
```

As we are going to count the density from 0 the initial count value is zero

```
count=0
```

Before detecting the car from image we have to import image into the code. This is done by cv.imread() command. Where, cv2 stands for computer vision2 and imread stands for image-read also name of image is declared in the braces.

```
image = cv2.imread('train6.jpg')
```

Images can be of different sizes, this may increase complexity of the algorithm. To simplify this we resize the image before using it and that image is scaled to more proportionate one by using cv2.resize() function. Three things are declared in the brackets i.e., image name, shape of height, shape of width.

```
                  resized = cv2.resize(image, (int(image.shape[1]/2),
                  int(image.shape[1]/2)))
```

Here, in this code we just have to detect the number of cars irrespective of the colour and features of the car. Thus to simplify the algorithm we are using grayscale image. This is done by cv2.cvtColor() function in which image name and the conversion of two colors. Here BGR2GRAY refers to colour to black and white conversion.

```
                  gray_img = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)
```

After gray-scaling the image the cars in the images are to be detected by using the detectMultiscale() function as it is tracing the data from 'cars.xml' on the grayscale image. scaleFactor and minNeighbors are the factors by which the image tracer shrinks from the edges to center of the image while detecting the car.

```
cars = car_cascade.detectMultiScale(gray_img,
                                   scaleFactor = 1.05, minNeighbors =5)
```

A there is the possibility of multiple cars in the image, the for loop is used and on every detection of car the count is increased by one. The rectangle is traced on the image on the detected portion by using cv2.rectangle() function where the coordinates of the rectangle and color of the rectangle is given in the parenthesis. This whole loop is executed until the total numbers of cars are detected.

```
            for x,y,w,h in cars:
              resized = cv2.rectangle(resized, (x,y), (x+w, y+h),
                                    (0,255,0), 3)
              count += 1
```

After the rectangles are traced on the image, the count of cars is printed on the screen and also final traced image highlighting the cars is also displayed followed by the waitKey() function which refers to pausing the code on that instruction until any key is pressed by the programmer. destryAllWindows() function destroys all the displayed images and resets the kernel.

```
                  print ("no of cars")
                  print (count)
                  cv2.imshow("Test_Out", resized)
                  cv2.waitKey(0)
                  cv2.destroyAllWindows()
```

Above code generates the density count by detecting vehicles from the captured image by the use of Open-CV. The “cars.xml” file includes the values which are used by the opencv to detect the cars in the image.

Python code for Automatic Number Plate Detection:

To generate the density of cars from the image it is necessary to use computer vision. Here cv2 is the computer vision library for python which I had to install before working on the images. Python stores images in the form of matrix as picture is created by pixels, it stores value of each pixel in the matrix, and imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

```
import cv2

import imutils
```

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images. Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

```
import pytesseract

pytesseract.pytesseract.tesseract_cmd = r"C:\Program
                        Files\Tesseract-OCR\tesseract.exe"
```

Before detecting the car from image we have to import image into the code. This is done by cv.imread() command. Where, cv2 stands for computer vision2 and imread stands for image-read also name of image is declared in the braces.

```
image = cv2.imread('Untitled.png')
```

Images can be of different sizes, this may increase complexity of the algorithm. To simplify this we resize the image before using it and that image is scaled to more proportionate one by using imutil.resize() function. Three things are declared in the brackets i.e., image name, shape of height, shape of width.

```
image = imutils.resize(image, width = 500)
```

After resizing the image, we are making sure the size is correct this is done by displaying resized image on screen by using cv2.imshow() followed by waitKey() function which is used to pause the code until the next key is pressed.

```
cv2.imshow("original Image", image)

cv2.waitKey(0)
```

Here, in this code we just have to detect the number of cars irrespective of the colour and features of the car. Thus to simplify the algorithm we are using grayscale image. This is done by cv2.cvtColor() function in which image name and the conversion of two colors. Here BGR2GRAY refers to colour to black and white conversion.

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("1-Grayscale Conversion", gray)
cv2.waitKey(0)
```

As we have to focus on the number-plate only other details can be lowered which will also help in increasing overall processing speed. This is done by using bilateral filter. This bilateral filter removes noise and excludes details by detecting the thresholds within the image. Here, opencv has its own function cv2.bilateralFilter() for this purpose. This process is followed by commands displaying this image on screen. Because, initially you have to chose thresholds by yourself, its bit of trial and error process.

```
gray = cv2.bilateralFilter(gray, 11, 17, 17)
cv2.imshow("2 - Bilateral Filter",gray)
cv2.waitKey(0)
```

Then after reducing noise while preserving edges in the image, it is time to perform edge detection for more accuracy. Here we have used canny edge detection amongst various edge detection methods. The function cv2.Canny() performs the canny edge detection and improves the edges in the image.

```
edged = cv2.Canny(gray, 170,200)
cv2.imshow("3 - Canny Edges", edged)
cv2.waitKey(0)
```

This image containing number of edges is then used to find contours in the image. Here contours refer to the edges having close path. Reason behind doing this is we know that if we edge out the number-plate that will result in a contour so, removing the non-contour edges will increase simplicity in the detection. This is done by CV2.findContours() function where the copy of the image whose contours are to found is provided.

```
cnts, new = cv2.findContours(edged.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

After finding these contours are drawn on the copy of original image created using image.copy() function. To draw contours on the image, cv2.drawContours() function of opencv is used also followed by displaying contours on the screen.

```
img1 = image.copy()
```

```
                    cv2.drawContours(img1, cnts, -1, (0,255,0), 3)
                    cv2.imshow("4 - All Contours", img1)
                    cv2.waitKey(0)
```

In the contoured image there may be some contours which are irrelevant e.g., small contours representing the grass as well as net from the heat outlet of the bonnet of the vehicle etc. Thus here we are removing all the contours by setting minimum contour area as 30, therefore all the contours with area below 30 are removed. This is done by using sorted() function.

```
      cnts = sorted(cnts, key = cv2.contourArea, reverse = True) [:30]
      NumberPlateCnt = None #we currently have no number plate contour
```

After detecting the top 30 contours these contours are stored in the variable named cnts. Here all these top 30 contours are displayed on the screen using drawContours() function. Also the drawn image is then displayed on the screen

```
                    img2 = image.copy()
                    cv2.drawContours(img2, cnts, -1, (0,255,0), 3)
                    cv2.imshow("5 - Top 30 contours", img2)
                    cv2.waitKey(0)
```

```
#loop over our number plate to find best possible approximate contour of number plate
count = 0
idx = 7
for c in cnts:
      peri = cv2.arcLength(c,True)
      approx = cv2.approxPolyDP(c, 0.02 * peri, True)
      #print ("approx" = "approx")
      if len(approx) == 4:#select contour with 4 corners
      NumberPlateCnt = approx #This is our Approximate Numberplate Contour

#crop these contours and store it in cropped image folder
x, y, w, h = cv2.boundingRect(c) #find co-ordinates for the plate
new_img = image[y: y+h, x: x+w] # create new image
cv2.imwrite('Cropped Images-Text/' + str(idx) + '.png', new_img) # store new image
idx+=1
break
```

```
#Drawing the selected contour on the original Image
#print (NumberPlateCnt)
cv2.drawContours(image, [NumberPlateCnt], -1, (0,255,0), 3)
cv2.imshow(" Final image with Number Plate Detected", image)
cv2.waitKey(0)

Cropped_img_loc = "Cropped Images-Text/7.png"
cv2.imshow("Cropped Image", cv2.imread(Cropped_img_loc))

# Use tesseract to convert the image into string
text = pytesseract.image_to_string(Cropped_img_loc, lang='eng')
print("Number is :", text)
cv2.waitKey(0) #Wait for user input before closing the image displayed
```

Above code uses the Canny edge detection to detect contours from the grayscale image and later it detects the entire numberplate form the image. The entire number-plate is then fed to py-tesseract which is basically an object to text converter and at its output it prints the entire number-Plate.

CHAPTER 4
RESULT

4.1. RESULTS:

4.1. A. Output showing density count:

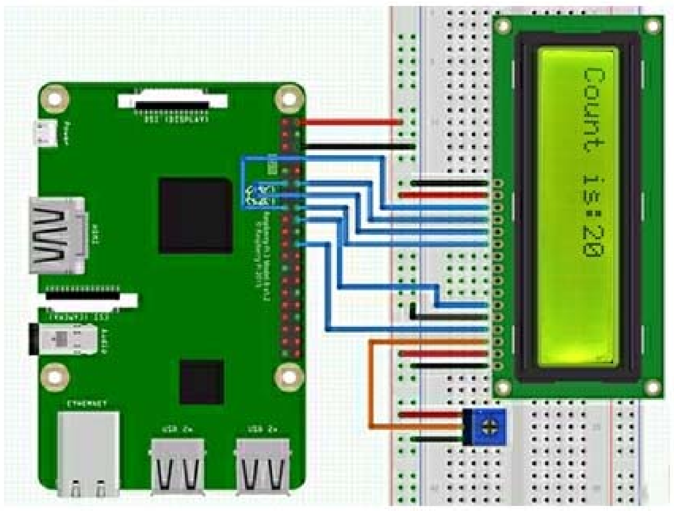
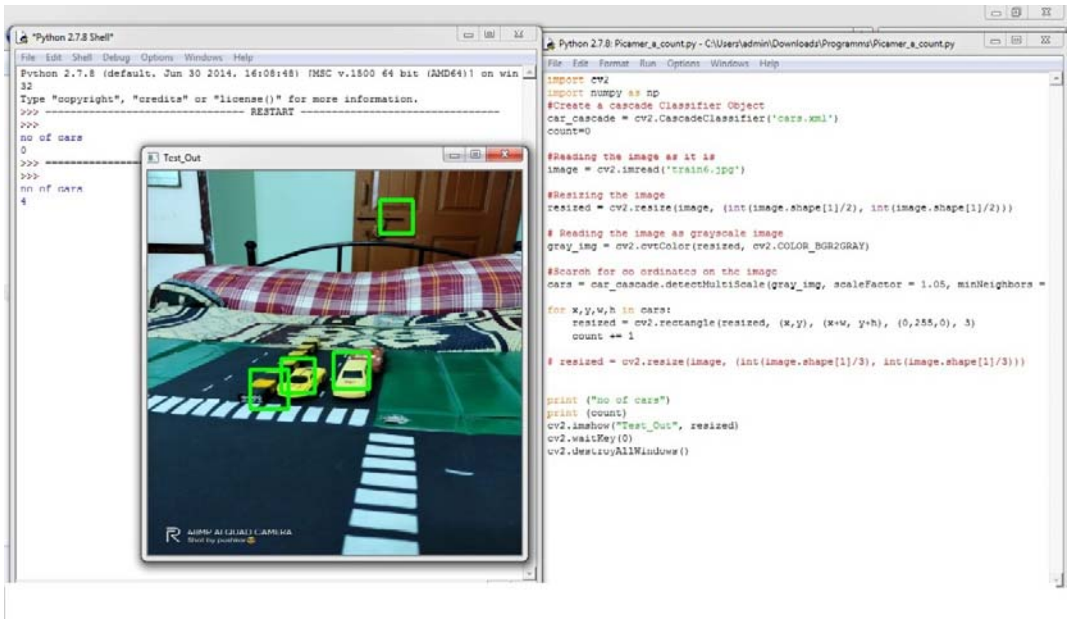


Fig.7: 16*2 LCD Interfacing with Raspberry Pi 3 B+

The Liquid Crystal Display is used to display the count of the vehicles on that lane. We can see that in the figure above shows the count on that lane i.e., 20 vehicles on that lane. For these we used random function to generate random value in the given range. This function is used in the main program according to its need.


"Python 2.7.8 Shell"
File Edit Shell Debug Options Windows Help
Test_Out
Python 2.7.8: Picamer_a_count.py - C:\Users\admin\Downloads\Programms\Picamer_a_count.py
File Edit Format Run Options Windows Help

```
Python 2.7.8 (default, Jun 30 2014, 16:08:48) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
no of cars
0
>>> ================
>>>
no of cars
4
```

```
import cv2
import numpy as np
#Create a cascade Classifier Object
car_cascade = cv2.CascadeClassifier('cars.xml')
count=0

#Reading the image as it is
image = cv2.imread('train6.jpg')

#Resizing the image
resized = cv2.resize(image, (int(image.shape[1]/2), int(image.shape[1]/2)))

# Reading the image as grayscale image
gray_img = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)

#Search for co ordinates on the image
cars = car_cascade.detectMultiScale(gray_img, scaleFactor = 1.05, minNeighbors =

for x,y,w,h in cars:
    resized = cv2.rectangle(resized, (x,y), (x+w, y+h), (0,255,0), 3)
    count += 1

# resized = cv2.resize(image, (int(image.shape[1]/3), int(image.shape[1]/3)))


print ("no of cars")
print (count)
cv2.imshow("Test_Out", resized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig.8: Counting cars from an image

These arcs were then modulated with audio frequency to give audio output. It was found that when the drive to the MOSFET driver circuit was increased i.e. when the frequency was increased above 1.25 MHz the MOSFETs were not able to take the high load and burned. Similarly, the primary was run at only 24V of supply. Increasing the primary voltage led to excessive heating of MOSFETs

4.1. C. Output of Automatic Number-Plate Recognition:

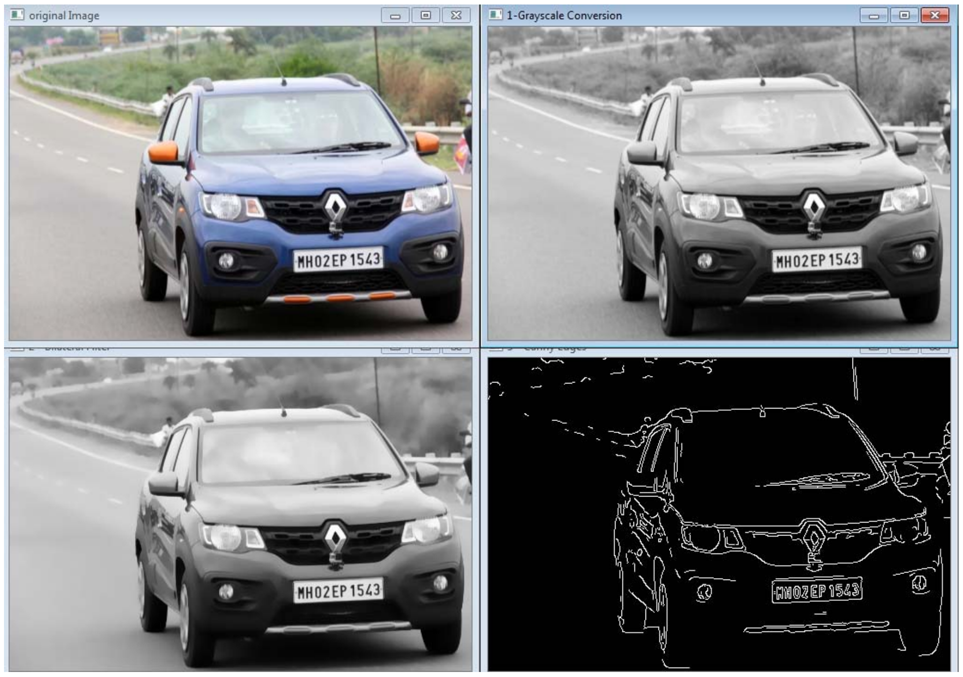
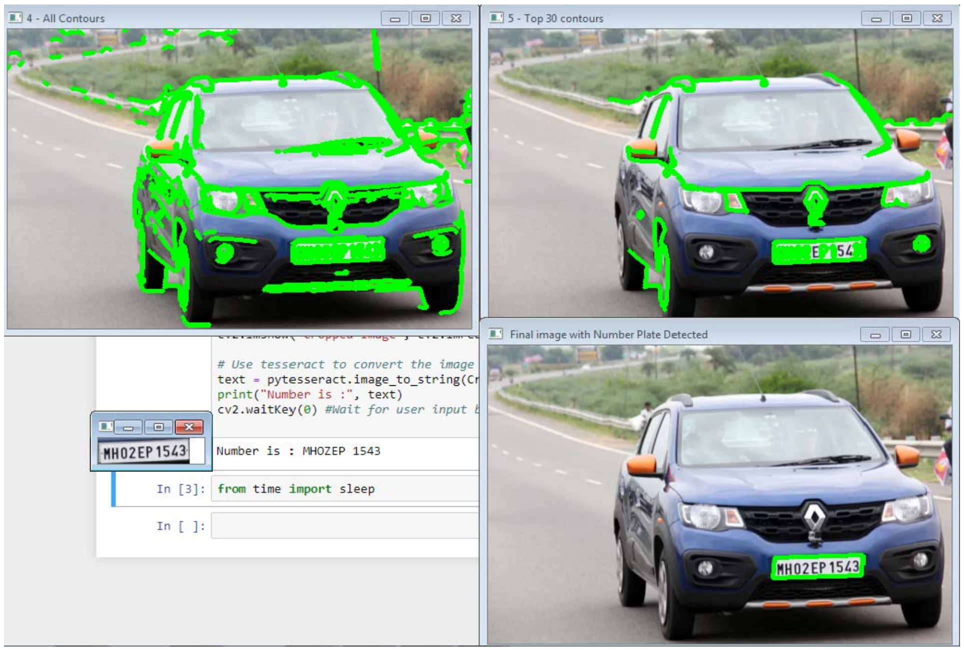


Figure 9(a): Automatic Number Plate Recognition system acquiring the number plate



```
# Use tesseract to convert the image
text = pytesseract.image_to_string(Cr
print("Number is :", text)
cv2.waitKey(0) #Wait for user input b

Number is : MHOZEP 1543

In [3]: from time import sleep

In [ ]:
```

Figure 9(b): Automatic Number Plate Recognition system acquiring the number plate

Figure 9(a), 9(b) shows the result of automatic number plate recognition (ANPR). i.e., how the number plate is extracted from the image. The first slide is the original image; the second one is the grayscale conversion of the original image. The third image is the result of the bilateral
Filter used to remove details, the fourth one is the output result of canny edge detection. The fifth image overlaps the detected contours on the original image whereas the sixth image includes the only top thirty contours and the seventh image is that detecting the number-plate. The eighth image includes cropped number-plate which is gained by cropping the sixth image with the coordinates of the contour and then the pytesseract is used to obtain the number-plate.

CHAPTER 5
SUMMARY & CONCLUSION

5.1 SUMMARY

The intelligent traffic light control was able to successfully monitor the change in traffic density if the lanes and control the traffic lights accordingly. When traffic rule violation is detected this proposed system is successfully noting the number plate of the vehicle. During the starting at the time 9:00 am the system automatically switches to automatic traffic control system and also the proposed system successfully checks the traffic density at the starting of each traffic cycle. Among the emerging technologies intelligent traffic system is the one that provides the traffic fluency as well as traffic rule violation at some point.

5.2 CONCLUSION

Traffic is the major problem which country faces today this is because of the increase in number of vehicles. The increase in number of vehicles resulting to the need of a smart system that could efficiently handle traffic congestion based on the density of traffic. Thus our system efficiently provides the smooth traffic flow as well as records the number plate of a vehicle violating the traffic rules, also it provides speed monitoring in the night time. The traffic flow is smooth because the densities of the lanes are checked in each iteration. The road density data of four roads was sent to raspberry pi and corresponding road signal with highest road density signals green color. The accuracy of vehicle detection depends on the weather conditions.

5.3 FUTURE SCOPE

With proper guidance from the local traffic department, the system can be successfully implemented also, as a future scope the ANPR system can be used to automatically delivering fine messages to the vehicle owner by the use of GSM module and legal local vehicle owner database. As the system can be used anywhere with the same modifications considering updating a local database, number-plate structure. At this point all the recorded data of vehicles is hard stored, in future the cloud storage can be used for this to improve data safety.

CHAPTER 6
APPENDIX

6.1 SNAPSHOTS

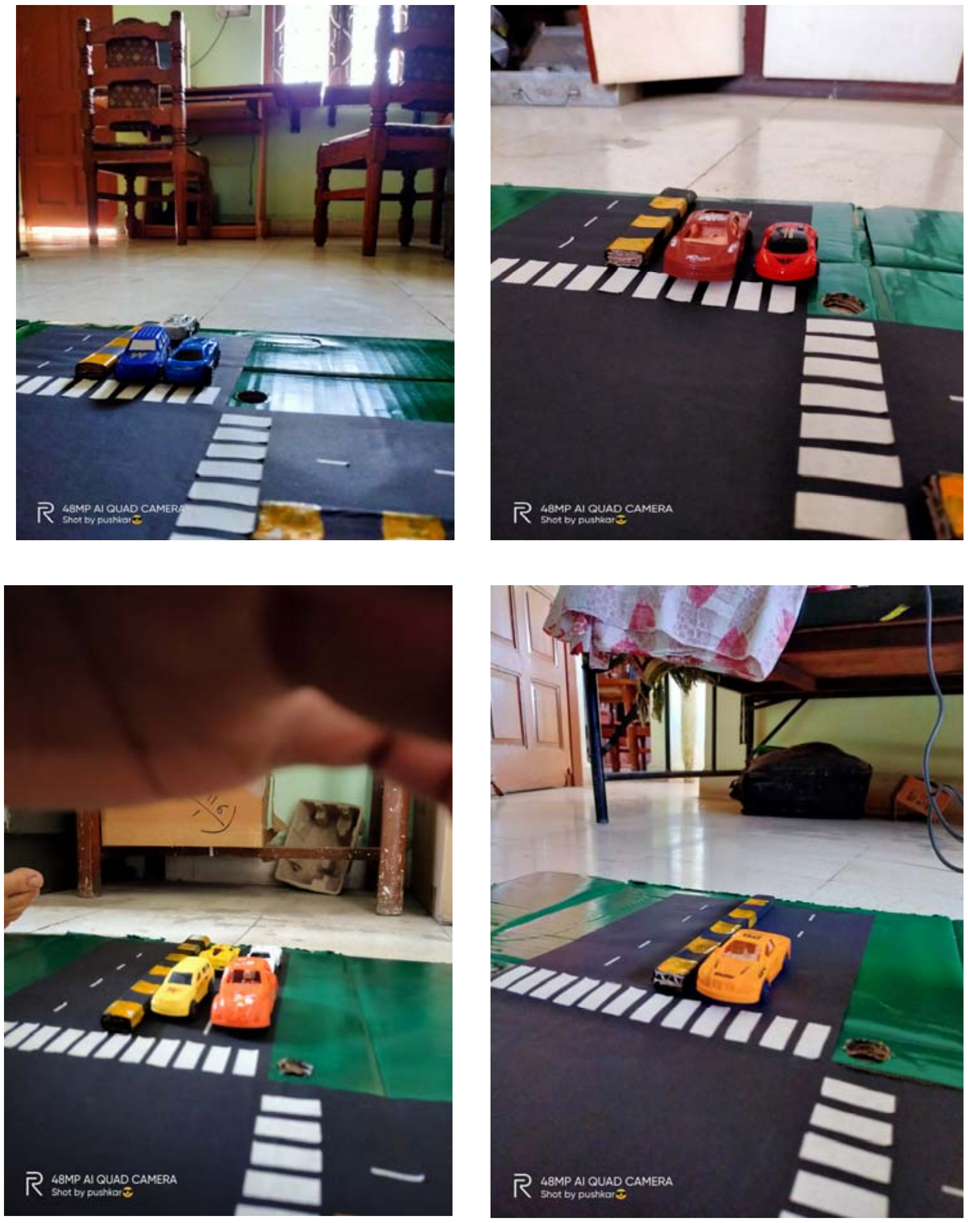


Fig.10: Prototype Design

6.2 LIST OF COMPONENTS

Sr. No.	Components	Quantity
1	RESISTORS	12
2	HDMI TO VGA CABLE	2
3	RASPBERRY PI ADAPTER	2
4	RASPBERRY PI 3B+	2
5	LED GREEN	4
6	LED RED	4
7	LED YELLOW	4
8	PICAMERA MODULE	2
9	REAL TIME CLOCK MODULE	1
10	JUMPRE WIRES M-M	20
11	JUMPRE WIRES M-F	20
12	JUMPRE WIRES F-F	20
13	LIQUID CRYSTAL DISPLAY 16*2	2

Table 5: List of Components

6.3 COSTING:

Sr. No.	Components	Quantity	Per Unit Cost	Total
1	RESISTORS	12	1.2/-	15/-
2	HDMI TO VGA CABLE	2	285/-	570/-
3	RASPBERRY PI ADAPTER	2	304/-	608/-
4	RASPBERRY PI 3B+	2	2600/-	5200/-
5	LIQUID CRYSTAL DISPLAY 16*2	2	130/-	260/-
6	REAL TIME CLOCK MODULE	1	76/-	76/-
7	PICAMERA MODULE	2	450/-	900/-
8	LED GREEN	4	1.5/-	6/-
9	LED RED	4	1.5/-	6/-

10	LED YELLOW	4	1.5/-	6/-
11	JUMPER WIRE M-M	25	1.2/-	30/-
12	JUMPER WIRE M-F	25	1.2/-	30/-
13	JUMPER WIRE F-F	25	1.2/-	30/-
		Total		7737/-

Table 6: Costing

CHAPTER 7
REFERENCES

7.1 REFRENCES

[1] “Intelligent Traffic Light Control System Based on Real-Time Traffic Flows” Zhijun Li, Chunxiao Li, Yanan Zhang and Xuelong Hu School of Information Engineering, Yangzhou University, Yangzhou, 225127, China 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)

[2] “Image Processing Based Traffic Light Control” Omkar Ramdas Gaikwad, Anil Vishwasrao, Prof. Kanchan Pujari, Tejas Talathi International Journal of Science, Engineering and Technology Research (IJSETR) Volume 3, Issue 4, April 2014

[3] “Density Based Traffic Light Control System Using Image Processing” D.Prakash1, B.Sandhya Devi2, R.Naveen Kumar3, S.Thiyagarajan4, P.Shabarinath5 Assistant Professor, Department of Electrical and Electronic Engineering, S.A. Engineering College, Chennai, India1 UG Scholar, Department of Electrical and Electronic Engineering, S.A. Engineering College, Chennai, India 2,3,4 IJAREEIE Vol. 6, Issue 3, March 2017

[4] “Image Processing Based Intelligent Traffic Control System by Using Raspberry Pi” Santoshi Gupta1, Utkarsha Raikar2, Bhavna Mohite3, Priyanka Patil4, Renuka Molavade5 Electrical Department1, 2, 3, 4, 5, SPPU University. Volume 6 Issue IV, April 2018

[5] “Traffic Light Control System Using Image Processing” Kavya P Walad, Jyothi Shetty M.Tech 3rdSemester, Department of Computer Science and Engineering, Srinivas School of Engineering, Mukka, India. Vol.2, Special Issue 5, October 2014

[6] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon University of Washington pjreddie Santosh Divvala Allen Institute for Artificial Intelligence Ross Girshick Facebook AI Research Ali Farhadi University of Washington.

[7] "Image Processing Based Intelligent Traffic Controller", Vikramaditya Dangi, Amol Parab, Kshitij Pawar & S.S Rathod Undergraduate Academic Research
Journal (UARJ), ISSN : 2278 – 1129, Volume-1, Issue-1, 2012.

7.2 LIST OF PAPER PUBLICATIONS / PARTICIPATIONS

Sr. No.	Authors	Title of Paper	Name of International Journals/ International conference	Place and Date of Publication with Citation Index	Status
1	Prateek Joshi Pushkar Paranjpe Akshay Bahe Sumit Manmode	Intelligent Traffic Light Control Using Image Processing	IJRASET Volume 8 Issue I, Jan 2020	www.ijraset.com ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177	Presented On 25th January 2019
2	Prateek Joshi Pushkar Paranjpe Akshay Bahe Sumit Manmode	Intelligent Traffic Light Control Using Image Processing	Luminescence	Dept. of ETC, Electrical, RGCER, Wanadongri, Nagpur	Presented On 25th March 2017

Table 7: List of Publications

CHAPTER 8
AUTHOR'S NOTE

AUTHOR'S NOTE

Name: Prateek Joshi

Email Id: pkjoshi1997@gmail.com@gmail.com

Permanent Address: Renuka Apartment, Gorle Layout, Gopal Nagar, Nagpur-10

Contact No: 8788517756

Name: Pushkar Paranjpe

Email Id: Paranjpepushkar@gmail.com

Permanent Address: 52, Income tax colony,Pratap nagar,Nagpur 440022

Contact No: 9860693368

Name: Akshay Bahe

Email Id: Akshay251828@gmail.com

Permanent Address: Plot no.5 ram nagar khat road Bhandara

Contact No: 9146865741

Name: Sumit Manmode

Email Id: Sumit.manmode@gmail.com

Permanent Address: 12/a express mill colony,near new ajni police station,Nagpur-27

Contact No: 9673055633