

# **Automatic Toll Collection System using Number Plate Detection**

Internet Of Things - EVD520

**School Of Engineering and Applied Science - Ahmedabad University**



## **Group Details :**

- 1) Het Jagani (1641007)
- 2) Jainam Patel (1641006)
- 3) Jay Patel (1643038)

## Motivation :

As we all know that transportation is the backbone of any country's economy. Improvement in transportation systems result into the good lifestyle, ultimately a better society in which we achieve extraordinary freedom for movement, immense trade in manufactured goods and services, as well as higher rate of employment levels and social mobility. In fact, the economic condition of a nation has been closely related to efficient ways of transportation. Increasing number of vehicles on the road, result into number of problems such as congestion, accident rate, air pollution and many other. All economic activities for different tasks use different methods of transportation. For this reason, increasing transportation is an immediate impact on productivity of nation and the economy.

Considering the present toll collection system where each vehicle has to stop and pay taxes. Suppose the manual toll collection system is very efficient for one vehicle to stop and pay taxes total time taken is 60 seconds. And suppose 100 vehicles cross the toll plaza. Then, time taken by 1 vehicle with 60 second average stop in a month is:  $60 \times 30 = 1800$  seconds Yearly total time taken =  $1800 \times 12 = 216200$ secs = 6.0 hours. If on an average we take 100 vehicles pass through the toll plaza each day, then yearly 36000 vehicles pass through the toll plaza. And each year 36000 vehicles just stand still for 6.0 hours in engine start condition thereby aiding pollution and wasting fuel and money. This figure can be drastically decreased by deploying some kind of automation system in which car does not have to stop to pay toll tax.

## Description :

The system comprises of the cameras connected to RaspberryPi which are kept at the toll gate. The camera captures the car number plate by running number plate detection algorithms on the raspberry pi. The raspberry pi will be connected to the database which stores the information about the toll tax to be paid on that road. The user will have to pay the toll tax afterwards on a website which fetches the details about the toll tax to be paid by user.

When the car enters the toll gate, the speed of the car should be reduced so that proper number plate detection can be done. Ultrasonic sensor will be kept, so when car is near to the camera the flash lights will turn on for the better visibility of the number plate. Speed breakers are used at the entry of the toll gate to reduce the speed of car. The toll payment system will be similar to the challan payment system, which is currently deployed in the city. If the user does not pay the toll tax in some specific amount of time then the registration of car will be cancelled until the user pays the decided fine.

## Current System :

Toll collection system initiated with an approach that might be conventional today, which was through toll booths, where a vehicle stops at the toll booths and the person in charge of the toll booth, registers the vehicle information, collects the payment in the form of cash or cards, at last allowing the vehicle to pass. This system is time consuming and inefficient compared to the later proposed systems for the Toll collection. This system was not only prone to errors like error entries made by the person in charge and mistakes in collecting the amount of toll, but also is very arduous in keeping track of all the transactions and the total amount of toll, which can be later used to analyse for the costs or other factors.

The newly proposed alternative of the conventional system was the use of automation and the electronics through a transponder device carrying vehicle's unique identification number defined as the Electronic Toll Collection System using RFID[Radio Frequency Identification]. In this system, the vehicles are to carry a transponder device. When a vehicle passes through the toll reader which is continuously emitting signal, this radio signal will trigger a response from the transponder device which helps toll reader identify the vehicle, update the information on the online portal available to both user and the toll collector, making the process fast, efficient and automated. If the transponder device or the reader are sometimes not reliable and not available, then the system fails to qualify as robust as for any automated systems, robustness of the system is one of the most important factors.

With the advent of the fields like Machine Learning, and making the process of detection of a vehicle's registration number and vehicle classification easy, a Machine Learning approach was proposed through the use of camera to capture the feed, detect the license plate and perform the transaction on an online portal. Comparing with the Electronic based system where both the transponder and reader has to work properly, here we only have one factor to consider which is proper functionality of the camera's installed and the processing of the feed. Similar to the Electronic system, reliability will play an important role in this approach.

These integrated approaches makes the process easy, efficient and fast for both the toll-collecting company and the user compared to the conventional approach. On the same time. we rely on machines, if not maintained properly, then it will lead to system failure. Even if this phenomenon is rare, it is the most integral part of the newly proposed system.

## Block Diagram :

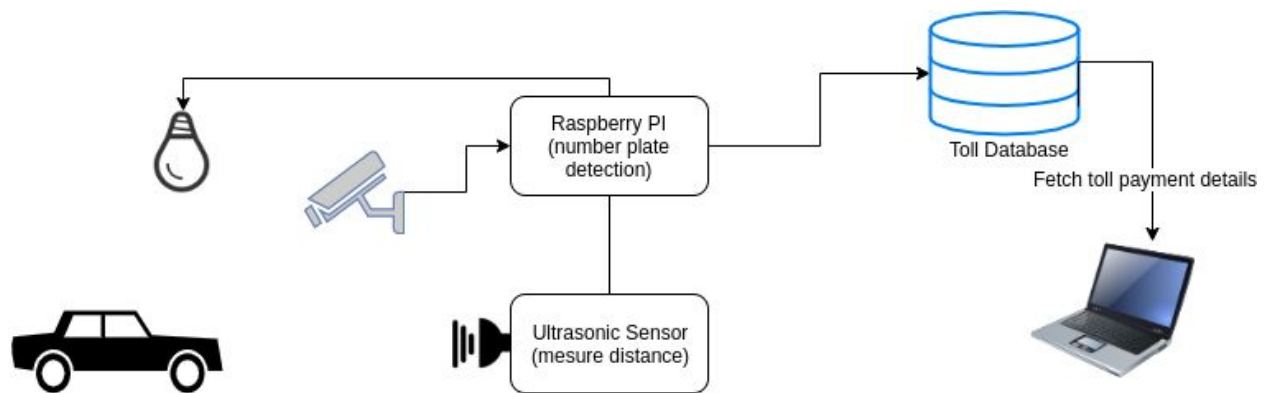


Fig : Block Diagram of system

As seen in the figure, the system comprises of the raspberry pi connected to camera module. Also an ultrasonic sensor is connected, so when car comes to certain distance, the light will turn on to provide better visibility. And then few frames are captured by the camera connected to raspberry pi. From these pictures, the number plate of the vehicle is extracted and then based on that an entry of toll tax is created in the toll database. The user will be able to see all the unpaid and paid toll tax on a website portal. On the website option to pay the tax will be provided.

## Raspberry Pi Features :

The following are the features that are needed to deploy the system like this in real life. The raspberry pi 3 covers most of the features that are needed to deploy such a system.

- Python programming environment, this helps in implementing the number plate detection algorithm. By having efficient libraries such as Tensorflow lite for IoT platform.
- More RAM memory that is needed in loading the pre-trained Machine Learning Models for number plate detection.
- Very fast and easy to use camera module to capture and process live camera feeds.
- A faster 1.2GHz processor that is needed for image processing and number plate detection algorithms.

## Circuit Diagram :

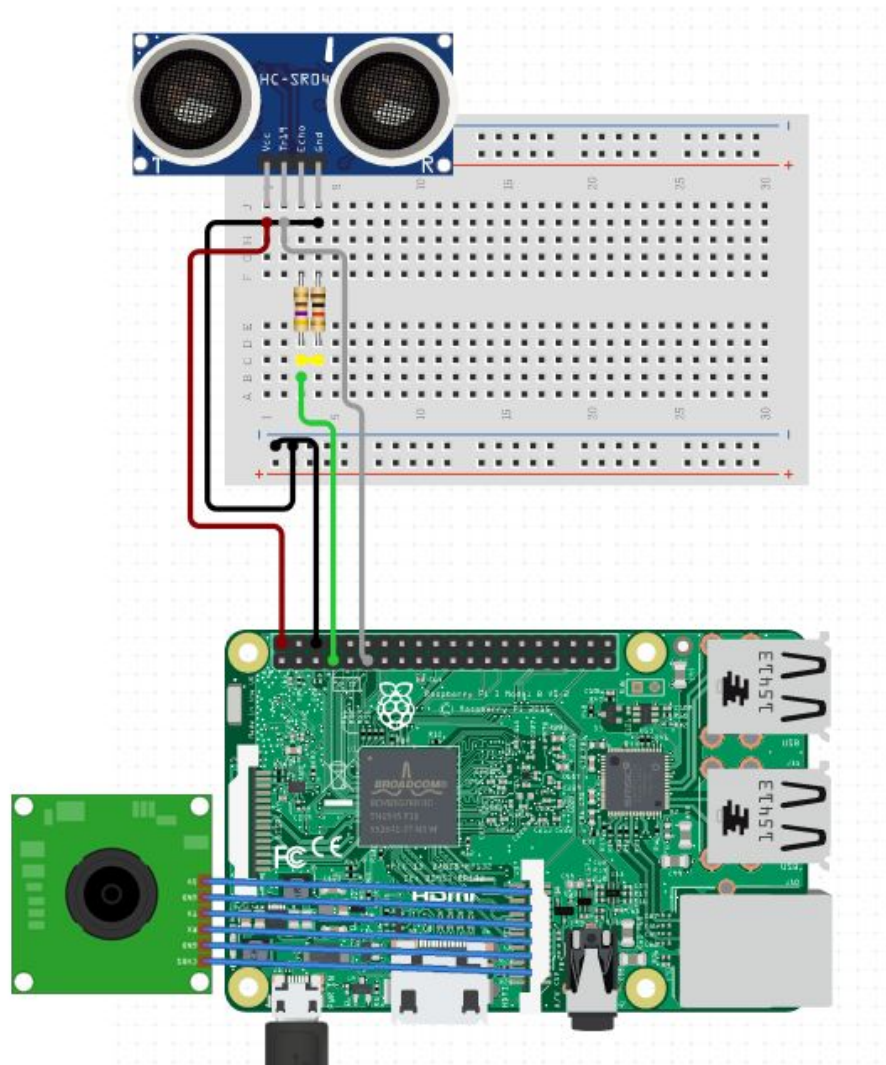


Fig : Circuit Diagram

## Flow Chart :

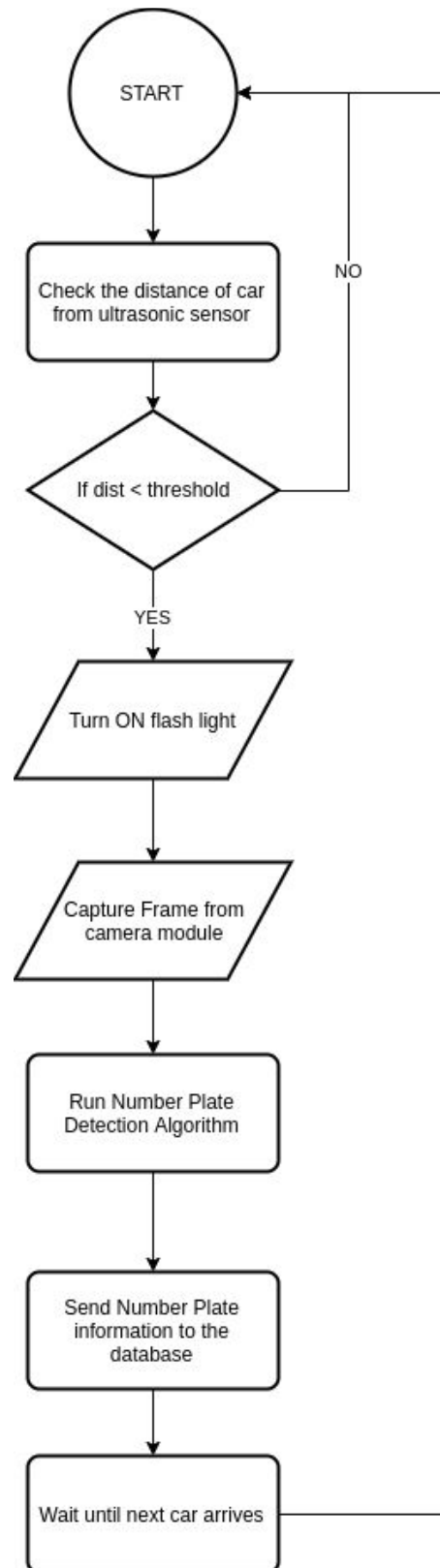


Fig : Flowchart

# Sensors :

## Ultrasonic Sensor (HC-SR04) :

Similar to the working of the radar, the Ultrasonic sensor module emits sound waves with very high frequency which humans cannot hear to measure time, then calculate distance. It basically consists of three parts: [1] Ultrasonic Transmitter [2] Control Circuit [3] Ultrasonic Receiver. The basic principle behind the module works by sending a sound wave through transmitter generating 40kHz sound, as soon as it hits an obstacle it is reflected back and received through the receiver, and based on the time measured, the distance of the obstacle can be calculated given that we know the speed of sound in the medium used.

**Formula:** distance =  $\{1/2\} * T * C$

T : time read through the sensor

C : speed of sound in a given medium where ultrasonic sensor is used

{speed of sound in air : 340m/s ; speed of sound in water : 1460 m/s}

### Specifications:

- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10μS TTL pulse
- Echo Output Signal Input TTL level signal and the range in proportion
- Dimension 45 \* 20 \* 15mm

### Raspberry Pi Interfacing:

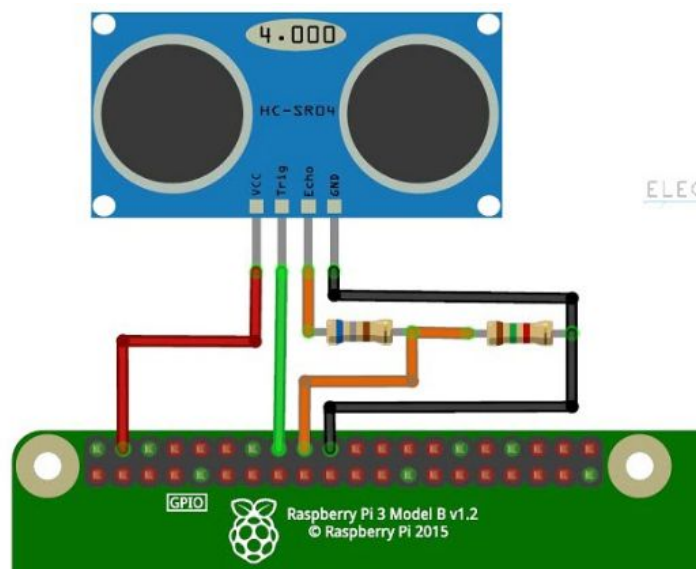


Fig : Connection with ultrasonic sensor

### Python Code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

TRIG = 23
ECHO = 24

GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

GPIO.output(TRIG, False)
print "Waiting For Sensor To Settle"
time.sleep(2)

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:
    pulse_start = time.time()
while GPIO.input(ECHO)==1:
    pulse_end = time.time()

pulse_duration = pulse_end - pulse_start
distance = pulse_duration * 17150
distance = round(distance, 2)
print "Distance:", distance, "cm"
GPIO.cleanup()
```



## Raspberry Pi Camera Module :

This miniature module not only allows pictures, but also allows real time video feature that too can be done programmatically, becoming a solution for a lot of problems on its own.

### Pin Diagram:

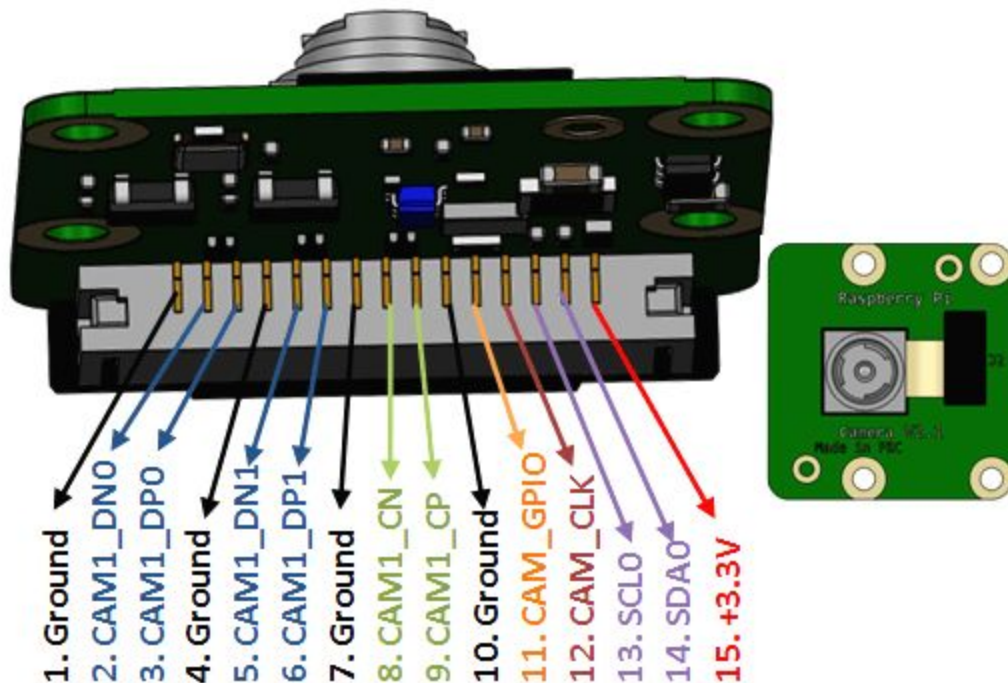


Fig: Pin diagram of camera module

### Pin Description:

Pin	Pin Name	Description
1	Ground	System Ground
2,3	CAM1_DN0,CAM1_DP0	Positive and Negative MIPI Data Pins for lane 0
4	Ground	System Ground
5,6	CAM1_DN1,CAM1_DP1	Positive and Negative MIPI Data Pins for lane 1
7	Ground	System Ground
8,9	CAM1_CN,CAM1_DP	Clock Pulses for MIPI Data Lanes
10	Ground	System Ground
11	CAM_GPIO	Optional GPIO Pin
12	CAM_CLK	Optional Clock Pin

13,14	SCL0,SDA0	For I2C communication
15	+3.3V	Power Pin

### Specifications:

Weight	3g	3g
Still resolution	5 Megapixels	8 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90	1080p30, 720p60 and 640 × 480p60/90
Linux integration	V4L2 driver available	V4L2 driver available
C programming API	OpenMAX IL and others available	OpenMAX IL and others available
Sensor	OmniVision OV5647	<a href="#">Sony IMX219</a>
Sensor resolution	2592 × 1944 pixels	3280 × 2464 pixels
Sensor image area	3.76 × 2.74 mm	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel size	1.4 μm × 1.4 μm	1.12 μm x 1.12 μm
Optical size	1/4"	1/4"

### Raspberry Pi Interfacing:

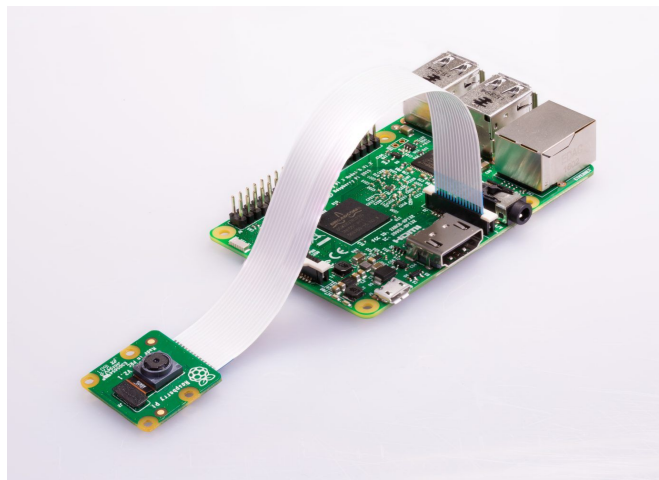


Fig : Connection of camera module

## Python Code:

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5)
camera.stop_preview()
```

## Actuators and Display :

### 1) LED :

The working principle of the Light-emitting diode is based on the quantum theory. The quantum theory says that when the electron comes down from the higher energy level to lower energy level then, the energy emits from the photon. The photon energy is equal to the energy gap between these two energy levels. If the PN-junction diode is forward biased, then the current flows through the diode.

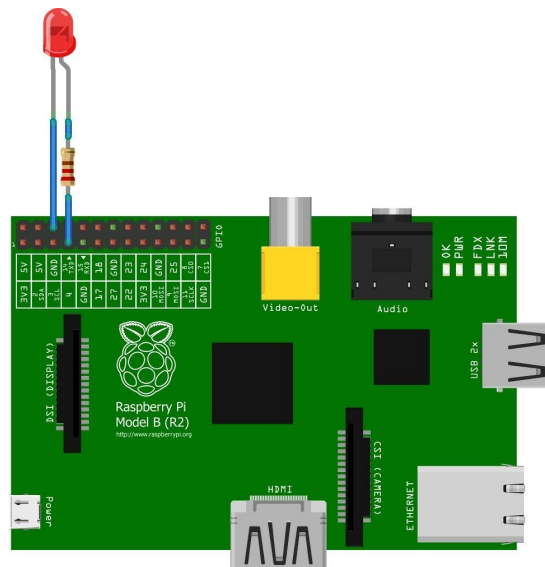
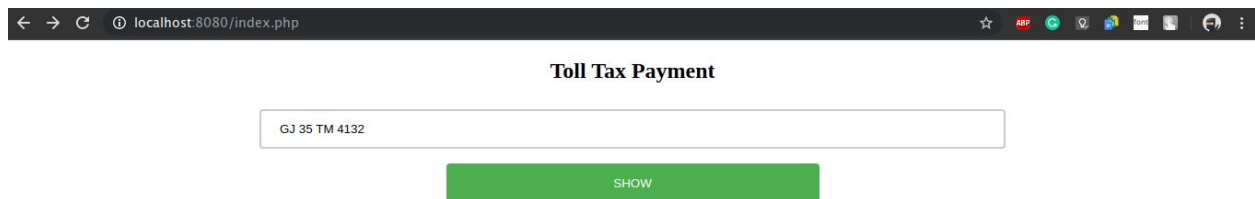


Fig : Connection with LED

# Details of Website :

A simple website is built for the payment of the toll tax. Any user who wants to pay the pending toll dues needs to enter his/her number plate on the search box on the website. Then, the website will fetch the data corresponding to the number-plate from the database. And the user will be able to see the details of his/her due and paid payments. The website is built in PHP and will be hosted on a server so that any user can access it.

## Page 1:



**Toll Tax Payment**

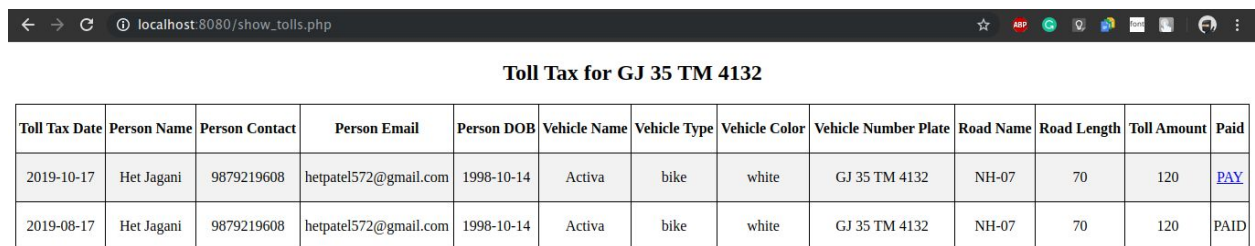
GJ 35 TM 4132

SHOW

Fig : Search page of the website

On this page user needs to enter his/her number plate in the box provided. Then click on the show button, which shows all the toll taxes till date of the vehicle corresponding to the number plate.

## Page 2:



**Toll Tax for GJ 35 TM 4132**

Toll Tax Date	Person Name	Person Contact	Person Email	Person DOB	Vehicle Name	Vehicle Type	Vehicle Color	Vehicle Number Plate	Road Name	Road Length	Toll Amount	Paid
2019-10-17	Het Jagani	9879219608	hetpatel572@gmail.com	1998-10-14	Activa	bike	white	GJ 35 TM 4132	NH-07	70	120	<a href="#">PAY</a>
2019-08-17	Het Jagani	9879219608	hetpatel572@gmail.com	1998-10-14	Activa	bike	white	GJ 35 TM 4132	NH-07	70	120	PAID

Fig : All the toll entries of vehicle with no plate GJ 35 TM 4132

This page shows all the entries of the toll tax which the user has incurred till date. The rightmost column specified if the user has paid the toll tax or not. If the user has paid the toll tax then the column displays "PAID" otherwise the link for the payment is provided. In the real world system different modes of payment may be provided when user click on the PAY link. Currently we have just marked the entry as PAID on clicking on the PAY link.

## Details of Supporting Tools :

For the database, the MySQL database is used. Phpmyadmin is used to access the database for the website (below figure). The database server is run on our local machine, which is accessed by the raspberry pi. The raspberry pi captures the number plate and stores the toll details in the database.

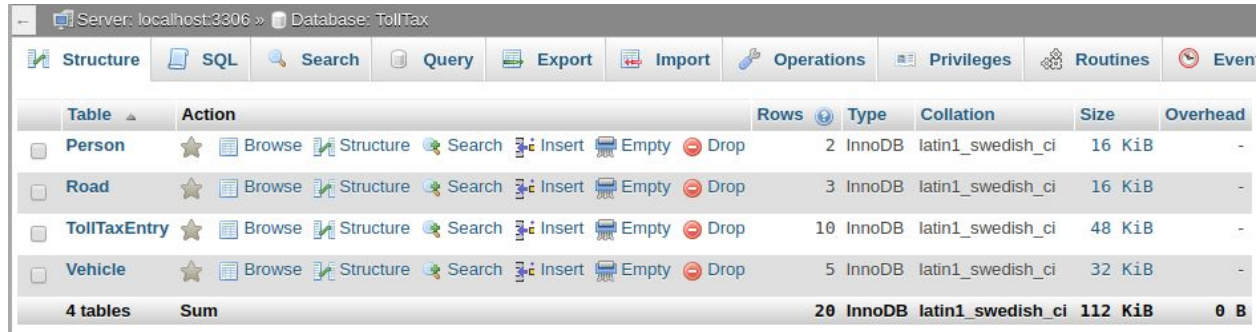


Table	Action	Rows	Type	Collation	Size	Overhead
Person		2	InnoDB	latin1_swedish_ci	16 KiB	-
Road		3	InnoDB	latin1_swedish_ci	16 KiB	-
TollTaxEntry		10	InnoDB	latin1_swedish_ci	48 KiB	-
Vehicle		5	InnoDB	latin1_swedish_ci	32 KiB	-
4 tables	Sum	20	InnoDB	latin1_swedish_ci	112 KiB	0 B

Fig : Structure of the TollTax database

For the number plate detection, OpenALPR cloud service is used. The OpenALPR Cloud API is a web service running in the cloud that analyzes images of vehicles and responds with license plate data, as well as vehicle color, make, model, and body type. We only require the detected number plate so we fetch the number plate from the response that we get. Below is the image of OpenALPR api website.

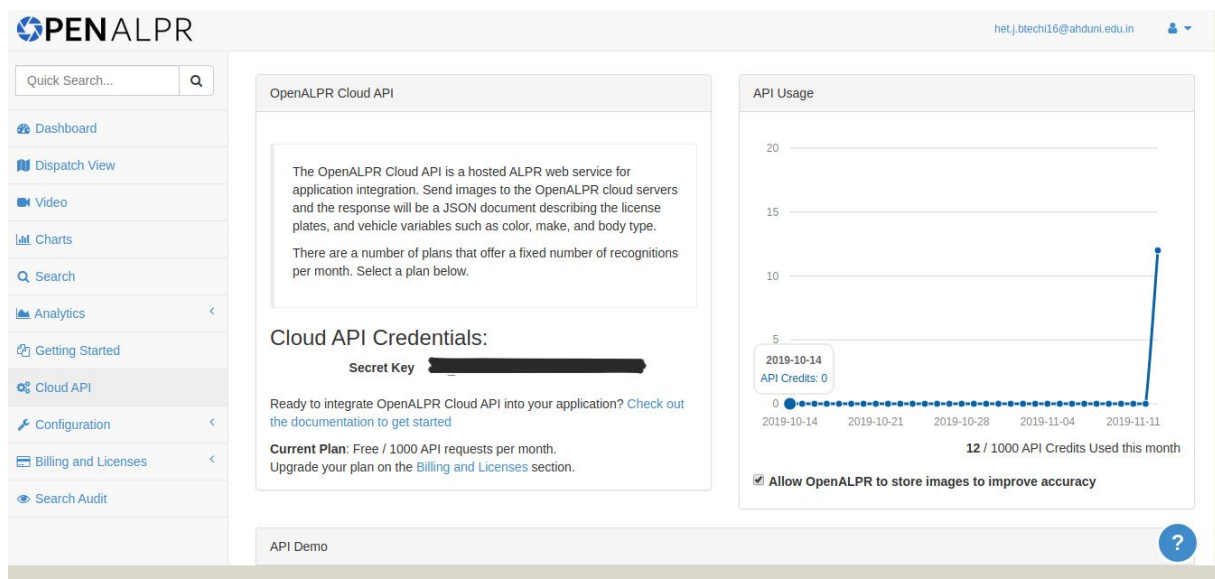


Fig : OpenALPR api webpage

## Code :

The code of the project is divided into various files as follows :

### ultrasonic.py :

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time

GPIO_TRIGGER = 18
GPIO_ECHO = 24
LED = 23

GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)

ledState = 0

def distance():
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.0001)
    GPIO.output(GPIO_TRIGGER, False)

    pul_sta = time.time()
    pul_end = time.time()

    while GPIO.input(GPIO_ECHO) == 0:
        pul_sta = time.time()
    while GPIO.input(GPIO_ECHO) == 1:
        pul_end = time.time()

    durat = pul_end - pul_sta

    dist = durat * 17150
    dist = round(dist, 2)

    return dist

def toggleLED():
    if ledState == 0:
        GPIO.output(LED, GPIO.HIGH)
        ledState = 1
    if ledState == 1:
        GPIO.output(LED, GPIO.LOW)
        ledState = 0
```

### **database\_handel.py :**

```
#!/usr/bin/python3
import MySQLdb
from datetime import datetime

myDB =
MySQLdb.connect(host="192.168.43.235",port=3306,user="root",passwd="password",d
b="TollTax")
cHandler = myDB.cursor()

def add_toll_entry(roadId, vehiclePlate):
    cHandler.execute("SELECT * FROM TollTaxEntry")
    entries = cHandler.fetchall()

    cHandler.execute("SELECT * FROM Vehicle WHERE vehicleNumPlate =
'{}'.format(vehiclePlate))
    num_entries = cHandler.fetchall()

    new_entid = "ent0"+str(int(entries[-1][0][-2:]) + 1)
    vehicleId = num_entries[0][0]
    curr_date = datetime.now()
    formatted_date = curr_date.strftime('%Y-%m-%d')

    ins_query = "INSERT INTO `TollTaxEntry`(`entryId`, `roadId`, `vehicleId`,
`paid`, `tollDate`) VALUES ('{}','{}','{}',0, '{}')".format(new_entid, roadId,
vehicleId, formatted_date)

    cHandler.execute(ins_query)
    myDB.commit()
```

### **openalpr.py :**

```
#!/usr/bin/python3
import requests
import base64
import json
import sys

SECRET_KEY = 'sk_279b338de6f0bc654ea7d370'

def detect_num_plate(image_path):
    with open(image_path, 'rb') as image_file:
        img_base64 = base64.b64encode(image_file.read())

    url =
'https://api.openalpr.com/v2/recognize_bytes?recognize_plate=1&country=in&secre
t_key=%s' % (SECRET_KEY)
    r = requests.post(url, data = img_base64)

    r = r.json()

    try:
```

```

        p = r["results"][0]
        plate = p["plate"]
        confidence = p["confidence"]
        return plate
    except IndexError as e:
        print(r)
        return None

```

### **auto\_toll.py :**

```

#!/usr/bin/python3
from database_handel import *
from openalpr import *
from ultrasonic import *
import picamera
import sys

if sys.argv != 2:
    print("usage ./auto_toll.py <road id>")
    exit(1)

IMAGE_DIR = '/home/pi/Project/pics/'
ROAD_ID = sys.argv[1]
image_path = IMAGE_DIR + 'pic.jpg'

while True:
    if distance() < 150:

        toggleLED()
        camera = picamera.PiCamera()
        camera.capture(image_path)
        number_plate = detect_num_plate(image_path)
        toggleLED()

        if number_plate:
            add_toll_entry(ROAD_ID, number_plate)

```



## Summary :

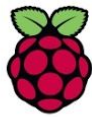
Fully automated postpaid system for toll collection which is inspired from current traffic challan collection system. The car does not have to stop on the toll booth at all. Only by slowing down of car the toll collection process is carried out. And the user has to pay the toll tax from a portal website.

## References :

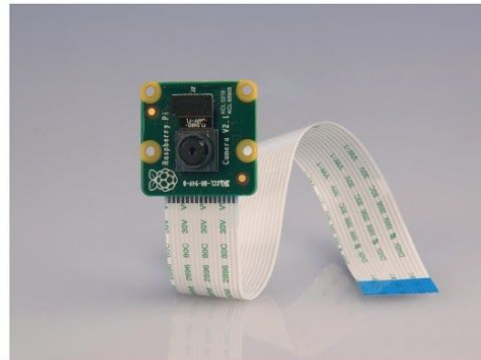
- [1] "Automated Toll Booth System Using Rfid," International Journal of Modern Trends in Engineering & Research, vol. 3, no. 11, pp. 61–65, 2016.
- [2] M. H. Hayward, "Electronic toll collection systems in Europe," Clean Mobility and Intelligent Transport Systems, pp. 135–156.
- [3] "Evaluating The Improvements In Traffic Operations At A Real-Life Toll Plaza With Electronic Toll Collection," ITS Journal - Intelligent Transportation Systems Journal, vol. 3, no. 3, pp. 249–252, 1996.
- [4] F. Aghdasi and H. Ndungo, "Automatic licence plate recognition system," 2004 IEEE Africon. 7th Africon Conference in Africa (IEEE Cat. No.04CH37590).
- [5] D. Tindall, "Deployment of automatic licence plate recognition systems in multinational environments," European Conference on Security and Detection - ECOS97 Incorporating the One Day Symposium on Technology Used for Combatting Fraud, 1997.
- [6] Q. Tian, T. Guo, S. Qiao, Y. Wei, and W.-W. Fei, "Design of Intelligent Parking Management System Based on License Plate Recognition," Journal of Multimedia, vol. 9, no. 6, 2014.\
- [7] "OpenALPR Login," OpenALPR. [Online]. Available: <https://cloud.openalpr.com/>.

# Appendix A :

## Camera Module Datasheet:



Raspberry Pi



## Camera Module

<b>Product Name</b>	<b>Raspberry Pi Camera Module</b>
<b>Product Description</b>	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
<b>RS Part Numer</b>	<b>913-2664</b>
<b>Specifications</b>	
<b>Image Sensor</b>	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
<b>Resolution</b>	8-megapixel
<b>Still picture resolution</b>	3280 x 2464
<b>Max image transfer rate</b>	1080p: 30fps (encode and decode) 720p: 60fps
<b>Connection to Raspberry Pi</b>	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
<b>Image control functions</b>	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
<b>Temp range</b>	Operating: -20° to 60° Stable image: -20° to 60°
<b>Lens size</b>	1/4"
<b>Dimensions</b>	23.86 x 25 x 9mm
<b>Weight</b>	3g



Tech Support: [services@elecfreaks.com](mailto:services@elecfreaks.com)

## Ultrasonic Ranging Module HC - SR04

### Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

- (1) Using IO trigger for at least 10us high level signal,
- (2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- (3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340M/S) / 2,

### Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

### Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

## Appendix B :

Python3	Arduino C
Rich in libraries for every functionality.	Comparatively limited libraries.
Object-Oriented Programming	Procedural Programming
Interpreted language, reduces the time of development.	C programs are first compiled using compilers and executable is created.
Python allows you to split your program into modules that can be reused in other Python programs.	Most C programs are monolithic and less integrative.
Memory management is automatic and implicitly done by the interpreter.	In C explicit declaration for allocation of memory is to be done.
Comparatively slower programs due to high abstraction for user.	Faster programs as compiler converts the code in assembly code.

## **Appendix C :**

### **Troubleshooting problem 1 :**

- For number plate detection, we tried various algorithms and implementations.
- But none of them worked, because the raspberry pi has very less computing power, and the image processing using neural networks is very computation heavy task.
- For better number plate detection, neural networks always give better performance, but every algorithm was computationally heavy.
- As a solution, we used the OpenALPR cloud platform, in which we used its API calls to detect the number plates.

### **Troubleshooting problem 2 :**

- While connecting the ultrasonic sensor, the values provided by the ultrasonic sensor were completely random.
- We looked online for connections, and tried codes, but nothing worked. All the efforts were in vain.
- Later after simplifying the connections, we realized that we have not used resistor, where it was required. After connecting that it worked properly.

### **Troubleshooting problem 3 :**

- While developing the database for the toll tax collection, At first we thought of storing the data in the files and retrieving the data when needed.
- But as all the files cannot be synced and shared between the raspberry pi and the server, we need to use a single database that can be accessed from both.
- As a solution, we have created a mySql database on the computer, which acts as server for the website. And started MySql server on the computer, so that raspberry pi can access the database remotely from the server.

## **Appendix D :**

### **Real Life Scenario 1:**

- As the system is postpaid the user cannot pay the toll tax if he/she does not want to.
- This scenario causes injustice for the people who regularly pay the toll tax, which should be avoided.
- As the whole system is autonomous the problem of trusting the user to pay the toll tax arises.
- As a solution, the rules which are applied for the eChallan system can be applied for the same. The registration of the car must be revoked if the user does not pay the incurred toll tax within a certain period of time.

### **Real Life Scenario 2:**

- In the remote areas and roads, the connectivity with the internet becomes the issue.
- As the raspberry pi continuously captures the photo and has to send it to cloud for number plate recognition, the raspberry pi has to have reliable and robust network connection.
- As a solution, if the toll booth is in the remote area than multiple access points or an ethernet cable should be employed for reliable connection.

### **Real Life Scenario 3:**

- The new data for vehicles and the toll booths needs to be updated in the database time to time.
- Because as the data about the vehicles and toll booths is required to create the toll entry for a vehicle which passes through particular toll booth on a road.
- To tackle this problem, some kind of management console or website can be created in which authorities can add this information from time to time.