



Bachelor of Technology (B. Tech.)
Computer and Communication Engineering (CCE)
Amrita School of Engineering
Coimbatore Campus (India)

Academic Year – 2022 - 23

Team 7

S. No	Name	Roll Number
1	Malavika Menon T	CB.EN.U4CCE20031
2	Manoj Parthiban	CB.EN.U4CCE20032
3	PG Mukund	CB.EN.U4CCE20034
4	V Srihari Moorthy	CB.EN.U4CCE20060

Fifth Semester

19CCE301 Internet of Things (IoT) Project

LICENSE PLATE DETECTION USING RASPBERRY PI

Faculty In-charge – Peeyush K P Sir

19CCE301 INTERNET OF THINGS

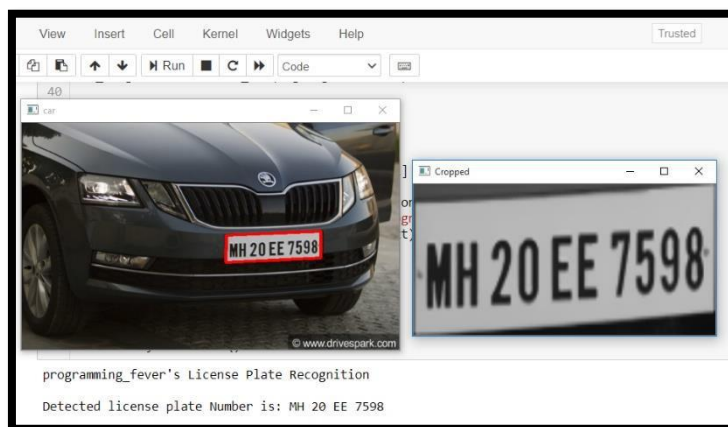
TERM WORK PROJECT

LICENSE PLATE DETECTION USING RASPBERRY PI

INTRODUCTION:

Humanity has always placed a high priority on security. Today, to help people feel comfortable, video surveillance cameras are installed in public places like schools, hospitals, and other buildings. According to an HIS survey, there were an estimated 245 million installed and operational security cameras in 2014, or one security camera for every 30 persons on the earth. These cameras can be made smarter by teaching them to process data from the video feed thanks to technological advancements, particularly in image processing and machine learning.

Vehicle-related security and safety issues are growing in importance as more automobiles are present on the roadways in our metropolitan centres. Since there are so many vehicles on the road, it is practically difficult to personally check and verify each one. In order to identify unregistered and suspect automobiles, some sort of automated technology is required for vehicle number plate identification. Using a Raspberry Pi for vehicle number plate recognition can be an excellent answer to these issues.



DESCRIPTION:

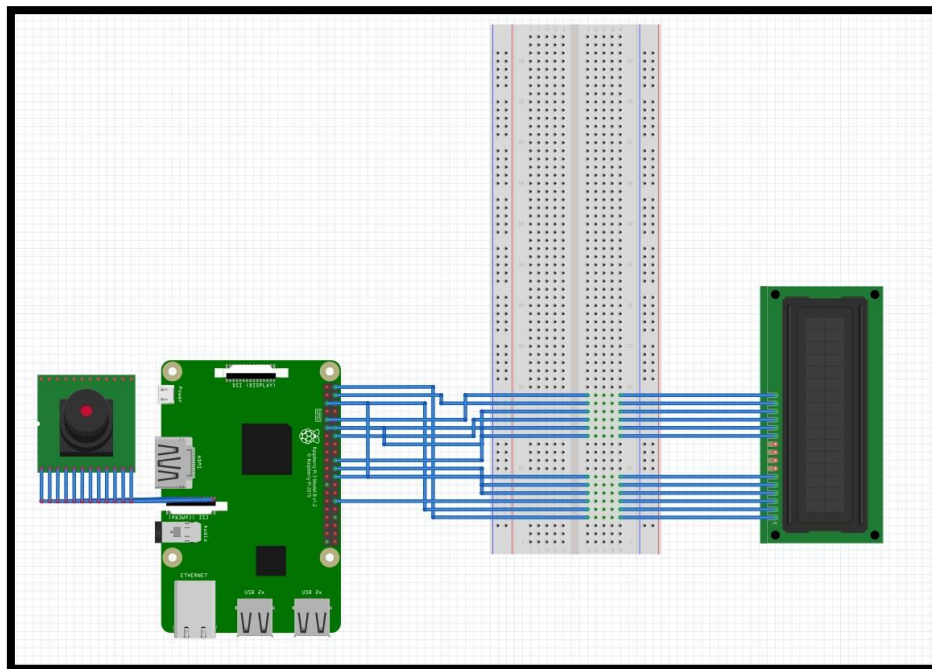
In fields including toll collecting, parking management, traffic policing, and crime investigation, automatic vehicle number plate detection is crucial. A vehicle number plate

recognition project based on the Raspberry Pi is suggested for the effective identification and detection of cars. The project includes a Raspberry Pi 4, Pi camera, and LCD 16x2 display.

HARDWARE COMPONENTS:

- 1) Raspberry Pi 4 - **Broadcom BCM2711 SoC**
- 2) Pi Camera module – 5 Mega Pixels
- 3) 16x2 LCD Panel
- 4) Connecting wires
- 5) Smartphone with sample number plate images

CIRCUIT DIAGRAM:



STEPS INVOLVED:

1. License Plate Detection:

The initial stage is to identify the vehicle's license plate. In order to locate the number plate, we will use OpenCV's contour feature to detect for rectangular objects. If we are aware of the exact dimensions, colour, and general location of the number plate, the accuracy can be increased. Typically, the location of the camera and the kind of licence plate used in that nation are utilized to train the detection algorithm. If there isn't even a

car in the photograph, things grow more difficult; in this instance, we'll take an extra step to find the automobile and then the licence plate.

2. Character Segmentation:

Once the licence plate has been located, we must clip it out and save the image as a new one. Again, OpenCV makes this simple to accomplish.

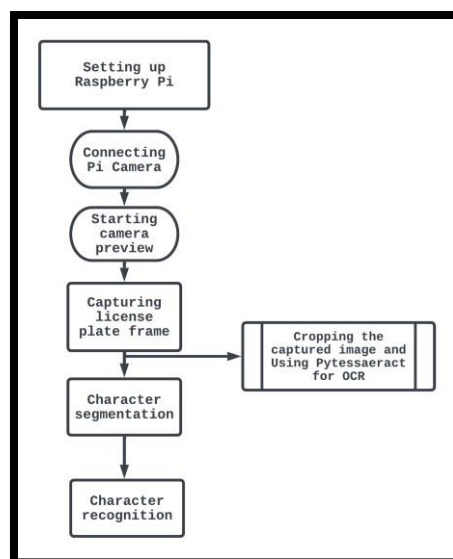
3. Character Recognition:

The newly acquired image from the previous phase is almost certainly written with some characters (numbers or alphabets). OCR (Optical Character Recognition) can be used on it to find the number. Using a Raspberry Pi, we previously discussed optical character recognition (OCR).

ALGORITHM:

- 1) License plate capture using PiCamera.
- 2) Image Gray scaling using OpenCV
- 3) Contour Detection using Canny Edge Algorithm
- 4) Masking and cropping the desired contour.
- 5) Perform image segmentation using PyTesseract
- 6) If string matches with database, impose penalty on related number plate
- 7) Else, License Plate is not registered with concerned authorities.
- 8) Then, Email is sent.

FLOWCHART:



WORKING:

Step 1: Detection of licence plates

The detection of the licence plate is the first stage in this Raspberry Pi licence plate reader. Take a look at a sample photograph of an automobile and begin by locating the licence plate on it. Then, the same image will be used for both character segmentation and character recognition. The whole code is supplied at the bottom of this page.



1) Resize the image to the required size and then grayscale it

When resizing, ensure that the licence plate is still visible in the frame to avoid any issues with higher resolution photographs. In every stage of picture processing, grey scaling is used. As we are no longer dealing with the colour details when processing an image, other subsequent processes move more quickly. After completing this stage, the image would resemble this.

2) Applying bilateral filters

Each image will contain both helpful and useless information; in this case, only the licence plate is useful for our software; the other information is largely irrelevant. Noise is the term for this unimportant information. A bilateral filter (Blurring) will typically eliminate the extraneous elements from an image.

To blur out more background information, you can increase the sigma colour and sigma space from 17 to higher numbers. Just be careful not to blur the useful area. The final image is displayed below; as you can see, it has background details (a tree and a building) that are blurry. By doing this, we may prevent the programme from focusing on these areas in the future.

3) Edge detection using Canny

The next step is interesting where we perform edge detection. There are many ways to do it, the most easy and popular way is to use the canny edge method from OpenCV.



- 4) Now we can start looking for contours on our image, we have already learned about how to find contour using OpenCV, The counters are then detected, sorted from large to tiny, and the first 10 results are solely taken into consideration. The value 0.018 is an experimental value; you can play around it to check which works best for you.

Once we have found the right contour, we save it in a variable called (screenCnt) and then draw a rectangle box around it to make sure we have detected the license plate correctly.

- 5) So, we can **proceed with masking the entire picture except for the place where the number plate is.**

Step 2: Character Segmentation

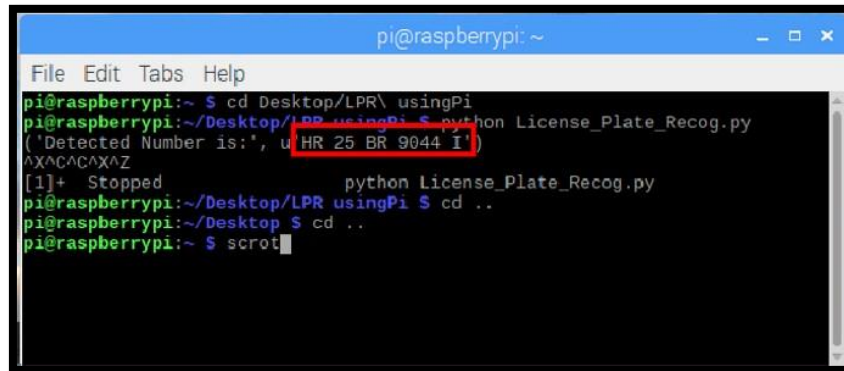
The next step in Raspberry Pi Number Plate Recognition is to segment the license plate out of the image by cropping it and saving it as a new image. We can then use this image to detect the character in it. The code to crop ROI (Region of interest) image form the main image is shown below

The resulting image is shown below. Normally added to cropping the image, we can also gray it and edge it if required. This is done to improve the character recognition in next step. However, I found that it works fine even with the original image.



Step 3: Character Recognition

The Final step in this Raspberry Pi Number Plate Recognition is to actually read the number plate information from the segmented image. We will use the *pytesseract* package to read characters from image, just like we did in previous tutorial. The code for the same is given below.



The screenshot shows a terminal window titled 'pi@raspberrypi: ~'. The user has navigated to the directory ~/Desktop/LPR usingPi and executed the command 'python License_Plate_Recog.py'. The output of the script is 'Detected Number is: u'HR 25 BR 9044 I'', where the detected text is highlighted with a red box. The terminal also shows the user pressing Ctrl+C to stop the script and then running 'scrot' to capture the screen.

SOURCE CODE:

```
import cv2
import imutils
import numpy as np
import pytesseract
from PIL import Image
from picamera import PiCamera
import time
import smtplib
import pandas as pd
from time import sleep
import RPi.GPIO as GPIO
import board
import digitalio
import random
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

import adafruit_character_lcd.character_lcd as characterlcd
lcd_rs = digitalio.DigitalInOut(board.D4)
lcd_en = digitalio.DigitalInOut(board.D15)
lcd_d7 = digitalio.DigitalInOut(board.D22)
lcd_d6 = digitalio.DigitalInOut(board.D9)
lcd_d5 = digitalio.DigitalInOut(board.D10)
lcd_d4 = digitalio.DigitalInOut(board.D27)
lcd_columns=16
lcd_rows=2
lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
lcd_d7, lcd_columns, lcd_rows)
```



```

p=PiCamera()

from difflib import SequenceMatcher

def similar(plate_text,word):
    a=SequenceMatcher(None,plate_text,word)
    return a.ratio()

def send_msg(t, plate_text):
    fine = random.randrange(100, 20000,50)
    mailmsg="""
    Name: %s
    License Number: %s
    Fine: Rs. %d""%(t, plate_text, fine)
    server=smtplib.SMTP('smtp.office365.com',587)
    server.starttls()
    server.login("sender's_email_id","password")
    server.sendmail("sender's_email_id ", t, mailmsg)
    server.quit()

p.start_preview()
time.sleep(4)
p.capture('4.jpg')
p.stop_preview()

img = cv2.imread('4.jpg',cv2.IMREAD_COLOR)
img = cv2.resize(img, (620,480) )

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert to grey scale
gray = cv2.bilateralFilter(gray, 11, 17, 17) #src dst dia sigma color
sigma space bordertype
edged = cv2.Canny(gray, 30, 200) #Perform Edge detection

# find contours in the edged image, keep only the largest
# ones, and initialize our screen contour
cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]
screenCnt = None
# loop over our contours
for c in cnts:
    # approximate the contour
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.018 * peri, True)
    # if our approximated contour has four points, then
    # we can assume that we have found our screen
    if len(approx) == 4:
        screenCnt = approx
        break

if screenCnt is None:
    detected = 0
    print("No contour detected")

```



```

else:
    detected = 1
if detected == 1:
    cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)

# Masking the part other than the number plate
mask = np.zeros(gray.shape,np.uint8)
new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
new_image = cv2.bitwise_and(img,img,mask=mask)

# Now crop
(x, y) = np.where(mask == 255)
(topx, topy) = (np.min(x), np.min(y))
(bottomx, bottomy) = (np.max(x), np.max(y))
Cropped = gray[topx:bottomx+1, topy:bottomy+1]
cv2.imshow('cropped',Cropped)
im1=cv2.imwrite('Cropped.png',Cropped)
#Read the number plate6
#text = pytesseract.image_to_string(Cropped, config='--psm 9')
text=pytesseract.image_to_string(Image.open('Cropped.png'),lang='eng',
config='--oem 3 --psm 11 ')

plate_text=[]
for i in text:
    if i.isalnum():
        plate_text.append(i)
plate_text="".join(plate_text)
cv2.imshow('image',img)
print("Detected Number is:",plate_text)
plate_text=str(plate_text)

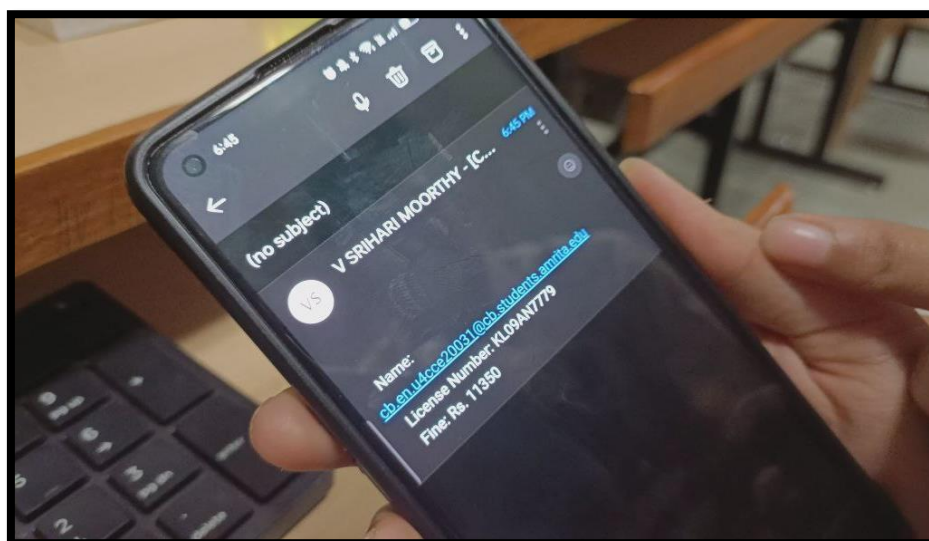
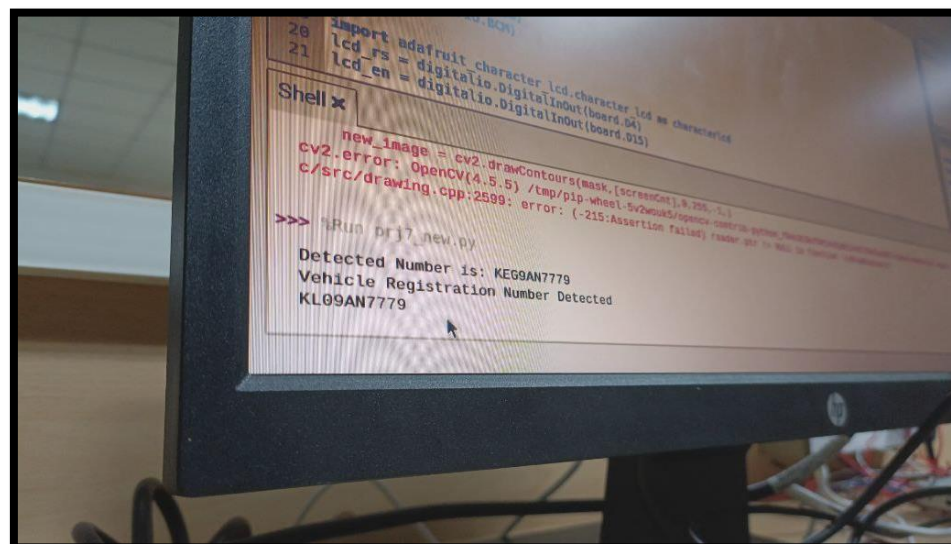
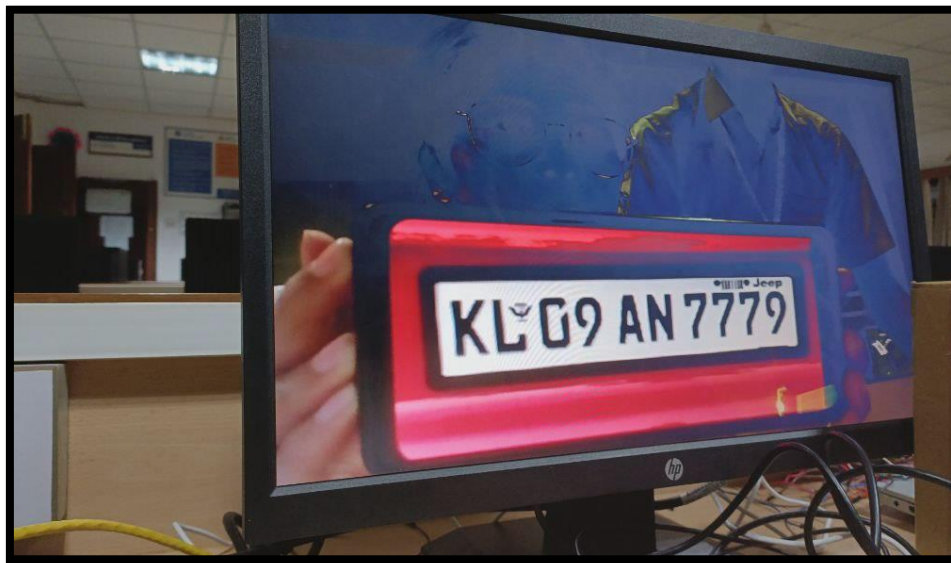
cv2.waitKey(0)
cv2.destroyAllWindows()
num=""

ob=pd.read_csv('/home/pi/Desktop/data.csv')
for i in ob['Number']:
    if similar(plate_text,i)>0.5:
        print("Vehicle Registration Number Detected")
        num=i
        print(i)
        msg = "Reg No:" + str(num)
        t=ob.loc[ob['Number']==i,'emailid'].values[0]
        lcd.message= msg
        send_msg(t, num)
        sleep(30)
        lcd.clear()

if num=="":
    msg="Not Regd"
    lcd.message= msg

```

OUTPUT:



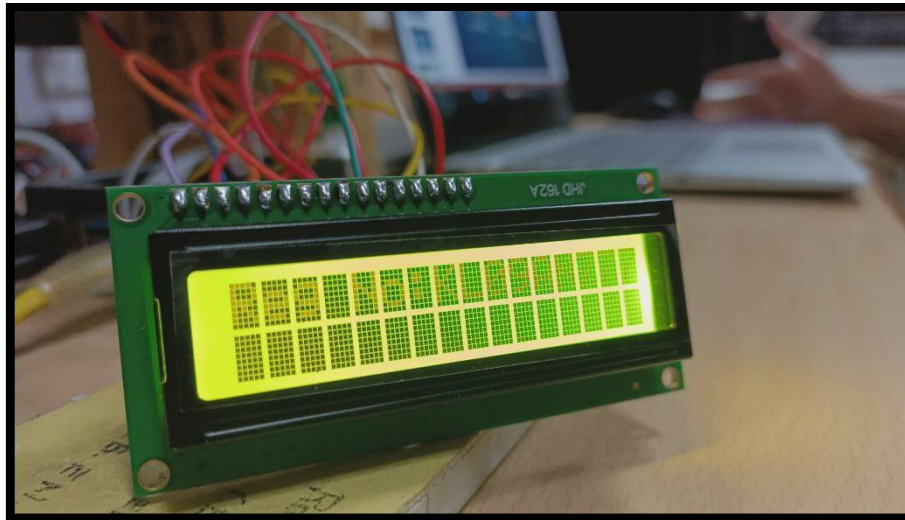


Fig: Printing the Regd. No on LCD

CONCLUSION:

The project is mainly based on license plate detection and recognition method. Firstly, the license plate is processed for obtaining a better image and removing noise. We used Python libraries including OpenCV and Tesseract OCR for License Plate Recognition. Horizontal and vertical projection, Optical character recognition, Convolutional network methods are used for character segmentation and character recognition.

We achieved more than 98% total accuracy.

THANK YOU