PROJECT

License Plate Recognition System

(IOT base)

Problem description :-

Develop a real-time License Plate Recognition System using Raspberry Pi and Camera for accurate plate capture in various conditions, suitable for applications like parking management and security. Prioritize real-time processing, recognition accuracy, and compatibility with Raspberry Pi hardware. (based on AIIOT).

Follow these contents :-

Problem statement

Scope pf the solution

Required component to develop solutions (include IDE name, software, and Hardware)

Simulated Circuit (Tinker Cad/Fritzing)

Video of the demo

Gerber file

Code for the solution

Batch(48)

Roll no.	Name
21781A04L8-	ADITYA KUMAR
21781A04L9-	AMAN KUMAR
21718A04M1-	ANKIT SHUKLA
22785A0416-	M. RAMAIAH

BRANCH- ECE (THIRD YEAR)

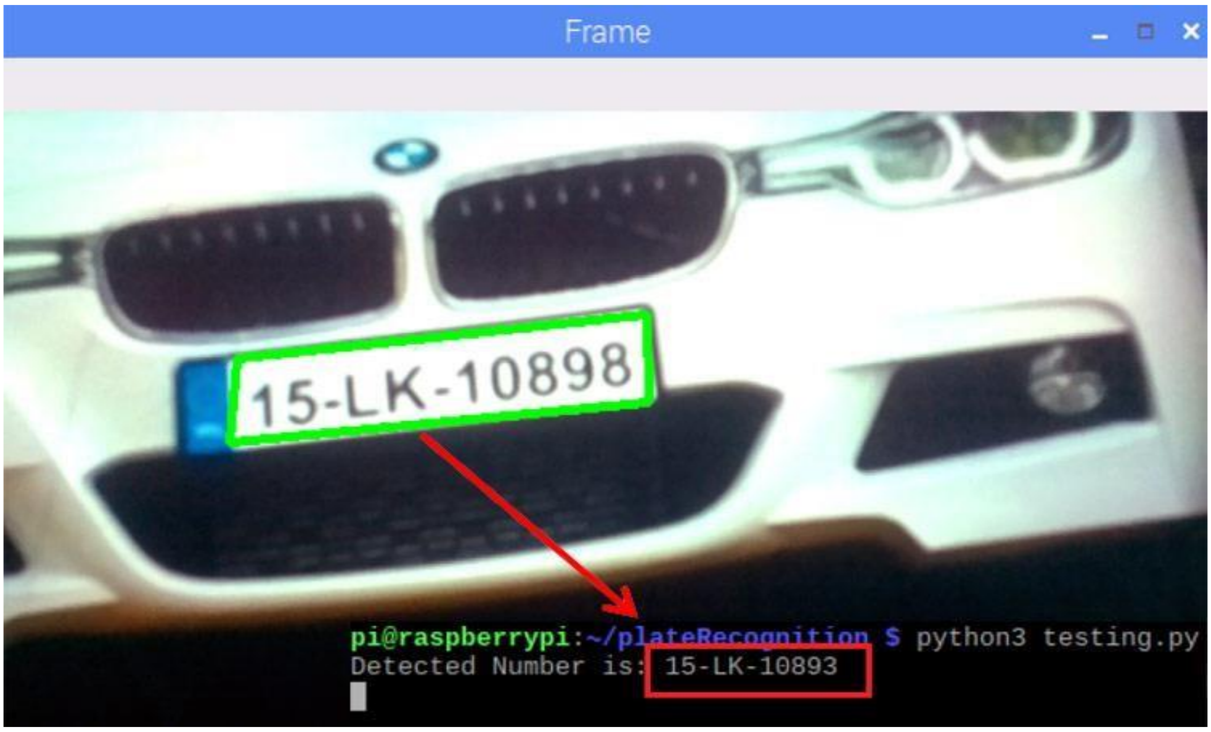
COLLEGE NAME-
SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY(A)

Problem statement:-

Develop a real-time License Plate Recognition System using Raspberry Pi and Camera for accurate plate capture in various conditions, suitable for applications like parking management and security. Prioritize real-time processing, recognition accuracy, and compatibility with Raspberry Pi hardware.

Scope of solutions:-

LPR, sometimes referred to as *automated license plate recognition* (ALPR), which cross-references the license plate number to national and/or international databases, captures a video image of a license plate and then uses an optical character recognition (OCR) algorithm to convert the plate number to the ASCII character set (see Chapter 3 for more details). For this to be a successful representation of the LPR, there must be a minimum of 25 pixels on the vertical side of the license plate characters. That would mean that a total of 5,000 pixels represent the entire license plate.



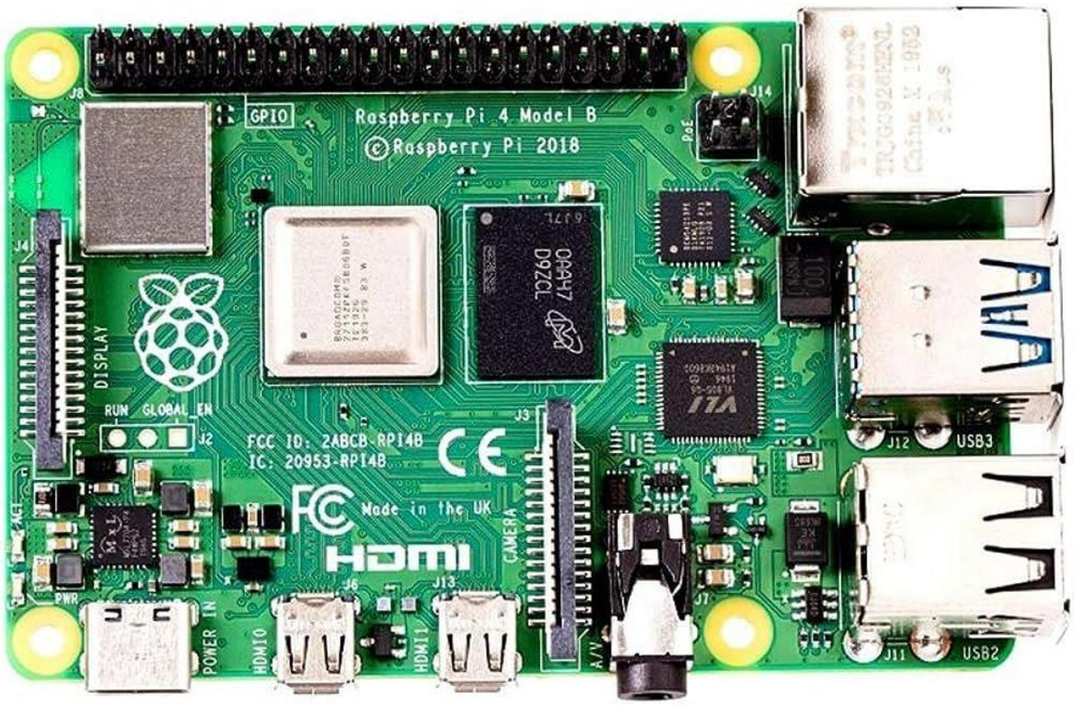
```
pi@raspberrypi:~/plateRecognition $ python3 testing.py
Detected Number is: 15-LK-10893
```

Security has always been a major concern for mankind. Today we have video surveillance cameras in schools, hospitals and every other public place to make us feel secured. According to a survey by HIS it is estimated that there were around 245 million security cameras installed and functioning back on 2014, which is like having one security camera for every 30 people on this planet. With the advancement in technology especially in Image processing and Machine Learning, it is possible to make these cameras smarter by training them to process information from the Video feed.

Requirement component:-

1. Raspberry pi 2. Pi camera 3. Basic tools 4. Power supply
5. Bread bord 6. Esp module
7. Monitor 8. Connecting wires

1.**Raspberry pi**



RASPBERRY PI

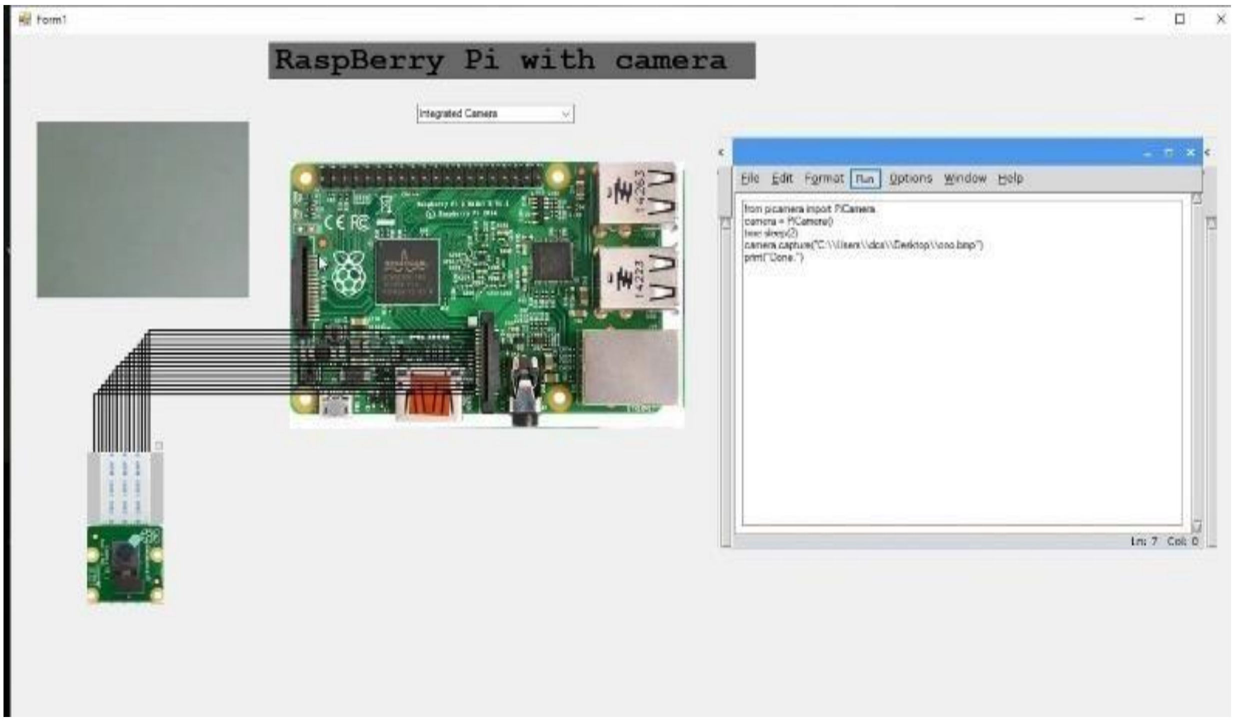
- Raspberry Pi is used all over the world as an easy, affordable way to teach programming and physical computing to people of all ages and all backgrounds.
- The Raspberry Pi was designed to be a low-cost, low-power device that can be used for a variety of purposes, including. Teaching basic computer science: The Raspberry Pi is often used in schools and educational settings to introduce students to the basics of computer science and programming.

2. Pi camera

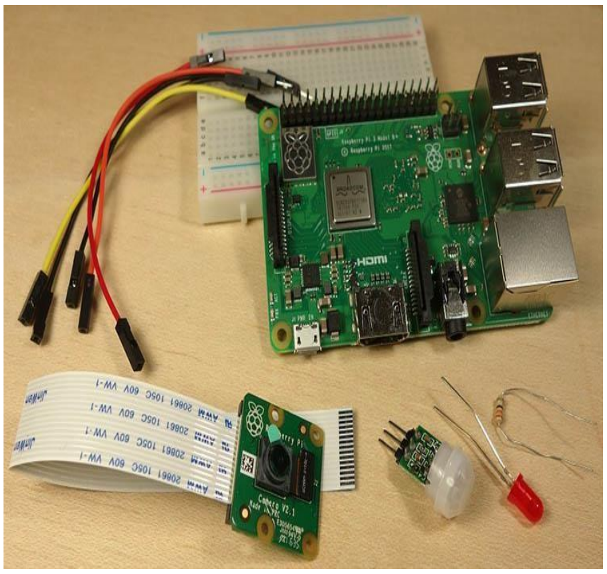
- The Raspberry Pi Camera Board is a custom designed add-on module for Raspberry Pi hardware. It attaches to Raspberry Pi hardware through a custom CSI interface. The sensor has 5 mega pixel native resolution in still capture mode. In video mode it supports capture resolutions up to 1080p at 30 frames per second. The camera module is light weight and small making it an ideal choice for mobile projects.

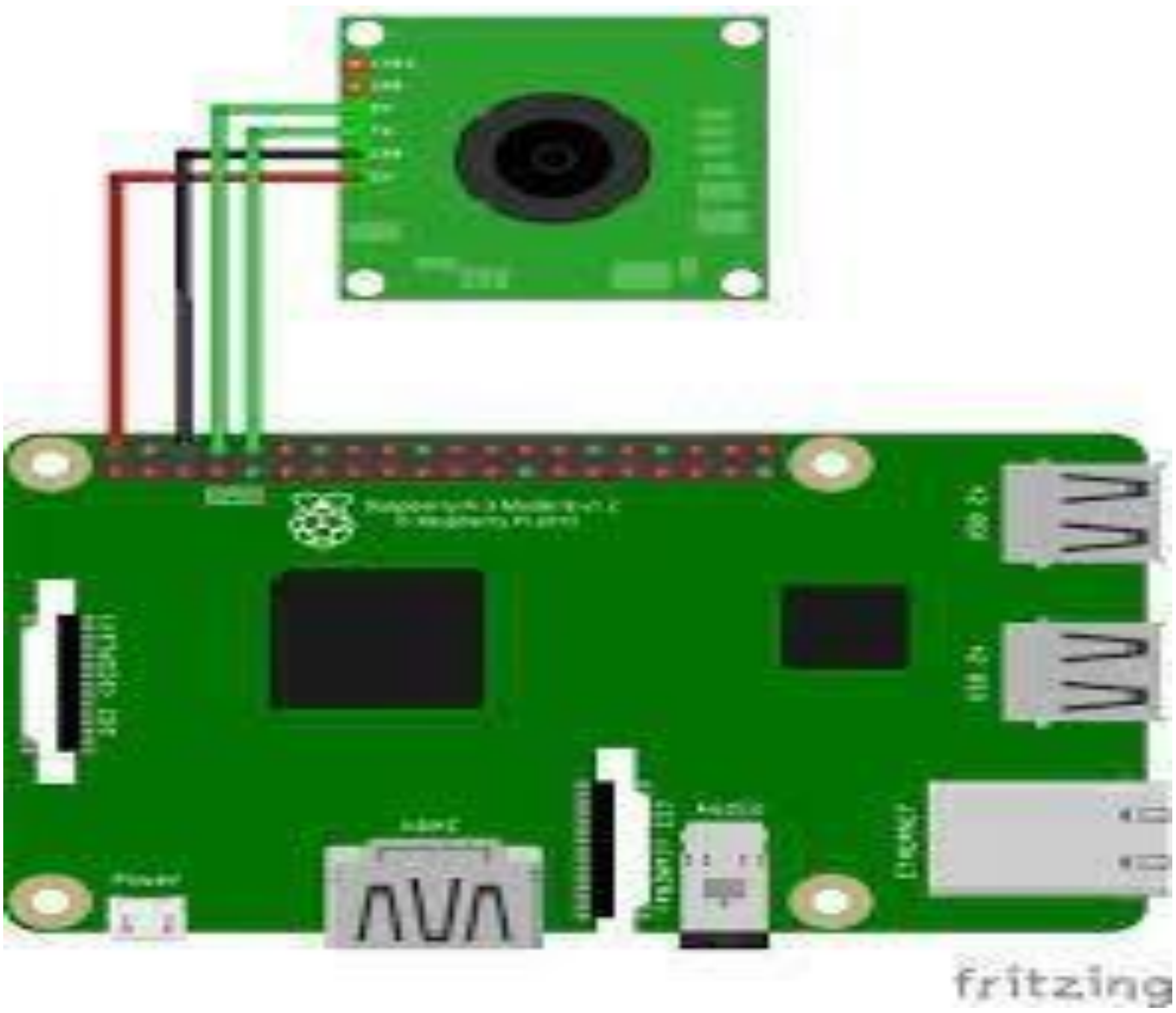
- VS CODE(SOFTWARE)-PROGRAMMING LANGUAGE (PYTHON)
- WORKN ON TERMINAL IN LINUX
- USING RASSBERY SYSTEM

Simulated circuit:-



```
from picamera import PiCamera
camera = PiCamera()
time.sleep(2)
camera.capture("C:\\Users\\dcs\\Desktop\\ooo.bmp")
print("Done.")
```



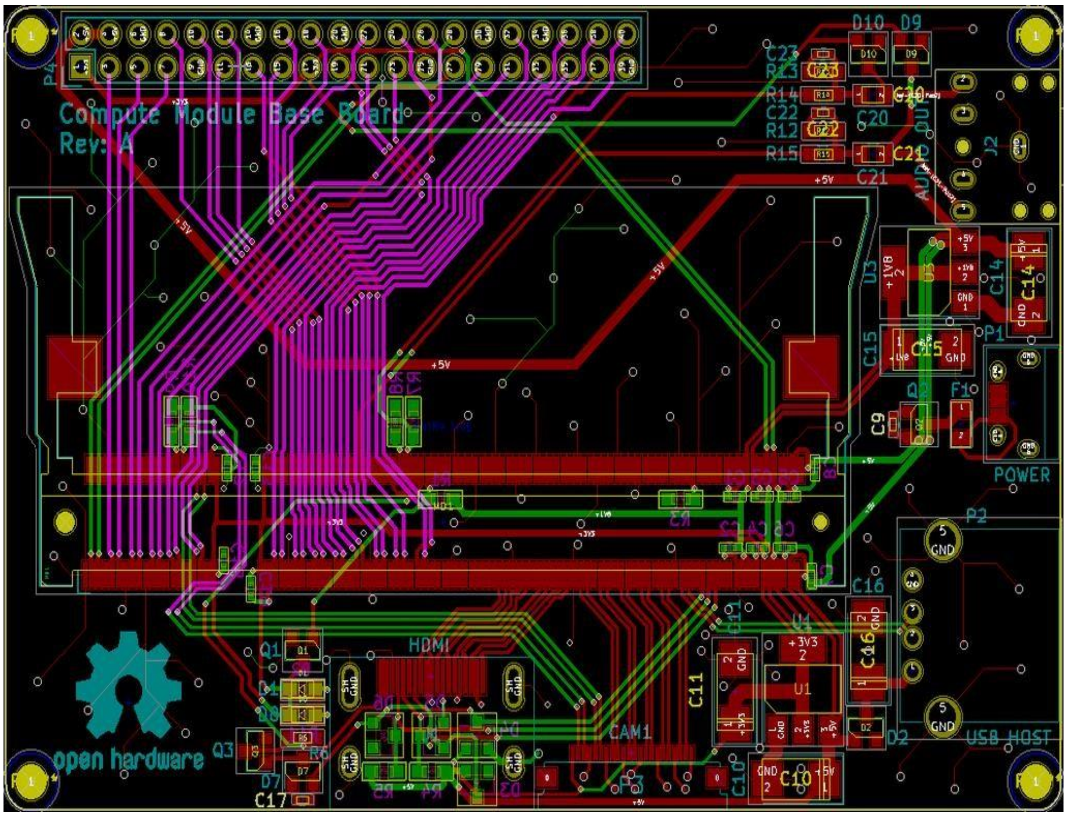


Connecting the Camera

The flex cable inserts into the connector labelled CAMERA on the Raspberry Pi, which is located between the Ethernet and HDMI ports. The cable must be inserted with the silver contacts facing the HDMI port. To open the connector, pull the tabs on the top of the connector upwards, then towards the Ethernet port. The flex cable should be inserted firmly into the connector, with care taken not to bend the flex at too acute an angle. To close the connector, push the top part of the connector towards the HDMI port and down, while holding the flex cable in place.

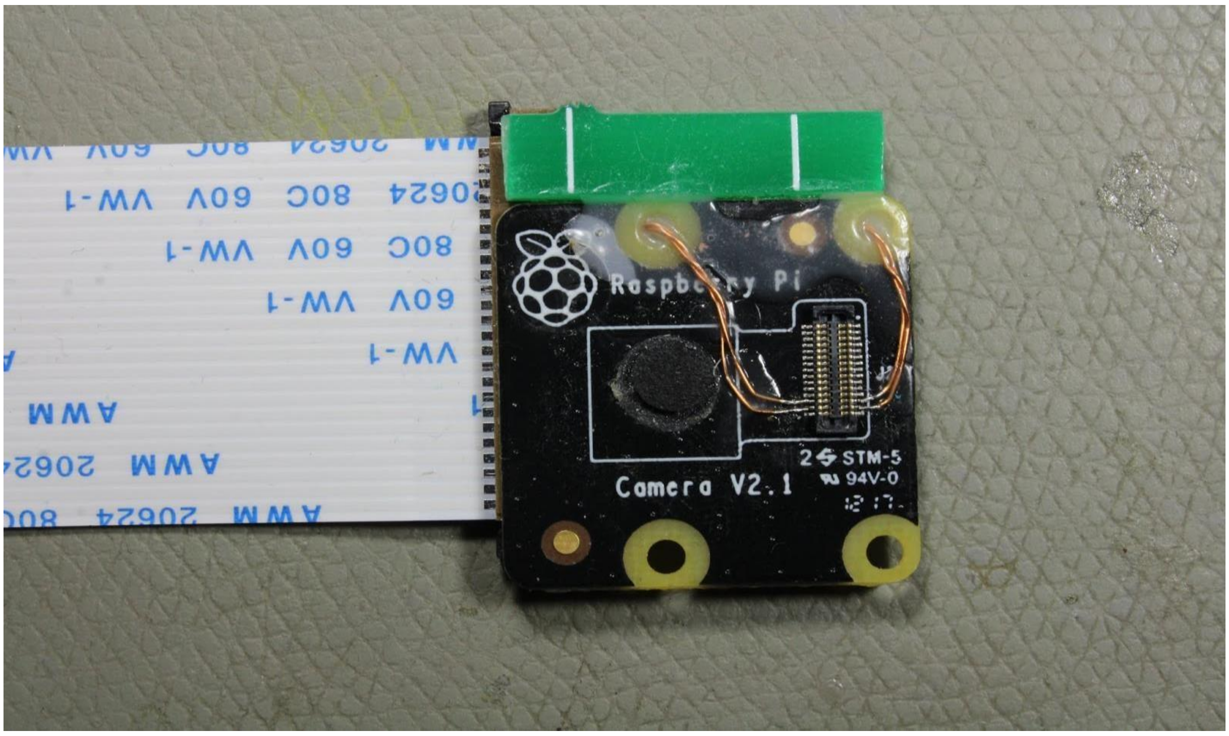
We have created a video to illustrate the process of connecting the camera. Although the video shows the original camera on the original Raspberry Pi 1, the principle is the same for all camera boards:

Gerber file



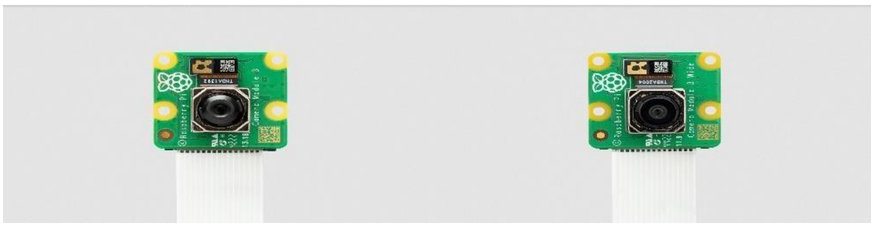
Now that you are done with all the basic configuration you can move to designing your own custom board based on the Compute Module. Since this is going to be your first project, I highly encourage you to grab my design and extend it to include any additional hardware you like The back of the board has plenty of space for adding your own components and for relatively small projects you likely don’t even have to increase the dimensions of the board.

Also, in case this is a standalone project and you don’t need a physical GPIO header on your board, you can easily get rid of it and save some space on the top side of the PCB. The GPIO header is also the only component that is routed through the second inner layer and removing it frees it up completely. I should point out that I have successfully assembled and tested one of the boards myself, and I have verified that everything including the camera and the HDMI output appears to be working as expected. So, as long as you don't make any huge changes to the way I've routed everything you shouldn't have any issues.



There are now several official raspberry pi camera modules. The original 5 mega fixel model was released in 2013 it was followed by an 8 mega fixel camera module 2 which was released in 2016, the latest camera is the 12 megapixel camera module 3 which was released in 2023. The original 5mp device is no longer available from raspberry pi.

All of these cameras come in visible light and infrared version while the camera module 3 also comes as a started or wide for model for a total of four different variants.



Code :-

```
import cv2 import imutils import numpy as
np import pytesseract from PIL import
Image from picamera.array import
PiRGBArray from picamera import
PiCamera import smtplib
server=smtplib.SMTP('smtp.gmail.com',587)
server.starttls() server.login("Your Email ID", "Email ID Password") camera = PiCamera()
camera.resolution = (640, 480) camera.framerate = 30 rawCapture = PiRGBArray(camera,
size=(640, 480)) for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):        image = frame.array        cv2.imshow("Frame", image)
key = cv2.waitKey(1) & 0xFF        rawCapture.truncate(0)        if key == ord("s"):
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #convert to grey scale
gray = cv2.bilateralFilter(gray, 11, 17, 17) #Blur to reduce noise            edged =
cv2.Canny(gray, 30, 200) #Perform Edge detection            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
for c in cnts:
```

```
            peri = cv2.arcLength(c, True)                approx =
cv2.approxPolyDP(c, 0.018 * peri, True)                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(imag
e, [screenCnt], -1, (0,
255, 0), 3)             mask
=
np.zeros(gray.shape,np.
uint8)
new_image =
cv2.drawContours(mask
,[screenCnt],0,255,-1,)
new_image =
cv2.bitwise_and(image,i
mage,mask=mask)
          (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]             text =
pytesseract.image_to_string(Cropped, config='--psm 11')

print("Detected Number is:",text)            server.sendmail("Sender's

Email ID@gmail.com","Sender's Email ID@gmail.com",text)

cv2.imshow("Frame", image)            cv2.imshow('Cropped',Cropped)

cv2.waitKey(0)            break

cv2.destroyAllWindows()
```

Conclusion:-

In order to gain a high frequency rate if license plate recognition, the images must be with good quality. Several factors can effect the quality of an image. These include the type of camera us3d, camera resolution, light and orientation of the camera used for acquiring the input images.

from the above results, we can conclude that number plate recognition will perform better as the quality of the camera used for scanning the plate will be excellent. Using low quality camera will degrade the performance and may misclassify the characters.