**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER ENGINEERING**

**LƯƠNG VĂN ĐẠI**

**CAPSTONE PROJECT**

# PARKING LOT ACCESS CONTROL SYSTEM

# HỆ THỐNG KIỂM SOÁT TRUY CẬP BÃI ĐẬU XE

**MIDTERM CAPSTONE PROJECT REPORT**

**HO CHI MINH CITY, 2023**

**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER ENGINEERING**

**LƯƠNG VĂN ĐẠI – 21521913**

**CAPSTONE PROJECT**

# PARKING LOT ACCESS CONTROL SYSTEM

# HỆ THỐNG KIỂM SOÁT TRUY CẬP BÃI ĐẬU XE

**MIDTERM CAPSTONE PROJECT REPORT**

**MENTOR**

**PhD. TRI NHUT DO**

**HO CHI MINH CITY, 2023**

VIETNAM NATIONAL UNIVERSITY **SOCIALIST REPUBLIC OF VIETNAM**

HO CHI MINH CITY **Independence – Freedom - Happiness**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

## DETAILED TOPICS

| |
|---|
| **VIETNAMESE PROJECT NAME:** |
| **ENGLISH PROJECT NAME:** |
| **Instructor** PhD. Tri Nhut Do, Faculty of Computer Engineering |
| **Implementation time:** From: 02/10/2023 To: 20/11/2023 |
| **Student Perform: Lương Văn Đại – 21521913** |
| **Overview of the topic:** The "Parking Lot Access Control System" is designed to efficiently manage vehicle entry and exit in a parking lot by scanning license plates and storing vehicle information until the vehicle leaves the lot. This system enhances security and ensures accurate tracking of vehicles within the parking facility.<br>**The goal of the subject:**<br>The primary goal of this system is to provide an automated and secure method for managing access to a parking lot. Key objectives include: |

**Methods of implementation:**

Hardware Components:

- Raspberry Pi 4: Use an Raspberry Pi 4 for processing and controlling the system.
- Pi Camera: Attach a camera or image sensor to capture license plate images.
- Led RGB: The color of the LED light is the operating signal of the system.
- - RFID: Use RFID for active the system

License Plate Recognition (LPR):

- Utilize image processing techniques and libraries (e.g., OpenCV) for LPR.
- Develop or integrate a pre-trained deep learning model for accurate license plate recognition.

Database Management:

- Set up a database (e.g., SQLite or MySQL) to store vehicle information, entry/exit timestamps, and access control data.
- Implement data encryption and security measures to protect sensitive information.

Access Control Logic:

- Design algorithms to validate incoming vehicles based on their license plates against a whitelist of authorized vehicles.
- Control gate/barrier mechanisms to allow entry to authorized vehicles and deny access to unauthorized ones.

<table>
<tr><td colspan="2">Main contents of the topic:<br><br>- Learn, design circuits and simulate results on software<br>- Programming Python language on Raspberry Pi<br>- Use the PiCamera (sensor) to capture the license plate, then process the image and put the license plate into the processor and save the license plate into the database, then the barrier will be turned on (actuator).<br>- Software used: Circuit design and simulation: Proteus - IDE code in Python language before creating the complete circuit</td></tr>
<tr><td>Certification of Instructor<br><br>(Sign and clearly state full name)</td><td>HCM city, 2023 Month DayDay<br><br>Student<br><br>(Sign and clearly state full name)</td></tr>
</table>

Table of Contents

## Chapter 1. **Introduction**

### **1.1.** Problem:

In the era of technological advancements, the need for efficient vehicle management has become more pronounced than ever. Managing a fleet of vehicles, whether for personal use or within a business context, poses numerous challenges, including ensuring safety, optimizing performance, and maintaining records. Embedded systems have emerged as a powerful solution to address these challenges, offering real-time monitoring, data processing, and automation. In this context, we will delve into the world of embedded systems and explore the "Vehicle Management System," a comprehensive solution that leverages the capabilities of sensors, processors, actuators, and databases, all harnessed through the Python programming language with the OpenCV library, to provide an integrated and sophisticated approach to vehicle management.

The "Parking Lot Access Control System" is designed to efficiently manage vehicle entry and exit in a parking lot by scanning license plates and storing vehicle information until the vehicle leaves the lot. This system enhances security and ensures accurate tracking of vehicles within the parking facility.

**1.2.** Target:

The target audience for this discussion includes individuals, businesses, and organizations that manage or own a fleet of vehicles. This could encompass transportation companies, logistics firms, rental car agencies, and even individual car owners seeking to optimize their vehicle's performance and safety.

**1.3.** Content of the topic:

The " Parking Lot Access Control System " discussed here is a sophisticated embedded system designed to address the challenges associated with vehicle management. It comprises several essential components:

1. RFID Reader: An RFID reader and corresponding RFID tags for vehicle identification.
2. Sensors (PiCamera): To capture real-time data, the system employs a high-quality camera sensor (PiCamera) that continuously records images and, if needed, videos from the vehicles. This data serves various purposes, such as tracking vehicle location, identifying drivers, and monitoring road conditions.
3. Processor (Raspberry Pi): The Raspberry Pi, a versatile and powerful single-board computer, acts as the brain of the system. It processes the data captured by the sensors, performs image recognition tasks using Python and OpenCV, and coordinates the system's various functions.
4. Actuators (Lever System): To respond to specific events or commands, actuators, such as a lever system, are employed. These actuators can

remotely control vehicle functions like locking and unlocking doors, disabling ignition, or triggering alarms in response to certain conditions. This is a small project so we will use RGB leds as an alternative method.

5. Python Language with OpenCV Library: Python, with its extensive libraries and community support, is the programming language of choice for developing the system. The OpenCV library, in particular, plays a crucial role in image processing, enabling features like object detection, facial recognition, and license plate recognition.
6. MySQL Database: To store and manage the vast amount of data generated by the system, a MySQL database is integrated. This database stores information on vehicle locations, driver identification, maintenance records, and more. This structured data enables efficient record-keeping and data analysis.

In this discussion, we will delve deeper into each of these components, exploring how they work together to create a powerful, integrated system for vehicle management. We will also discuss the benefits of such a system, including enhanced safety, improved performance, and cost savings. Furthermore, we will examine the challenges and considerations associated with implementing and maintaining a "Vehicle Management System" to provide a comprehensive understanding of this innovative solution.

## **1.4.** Function

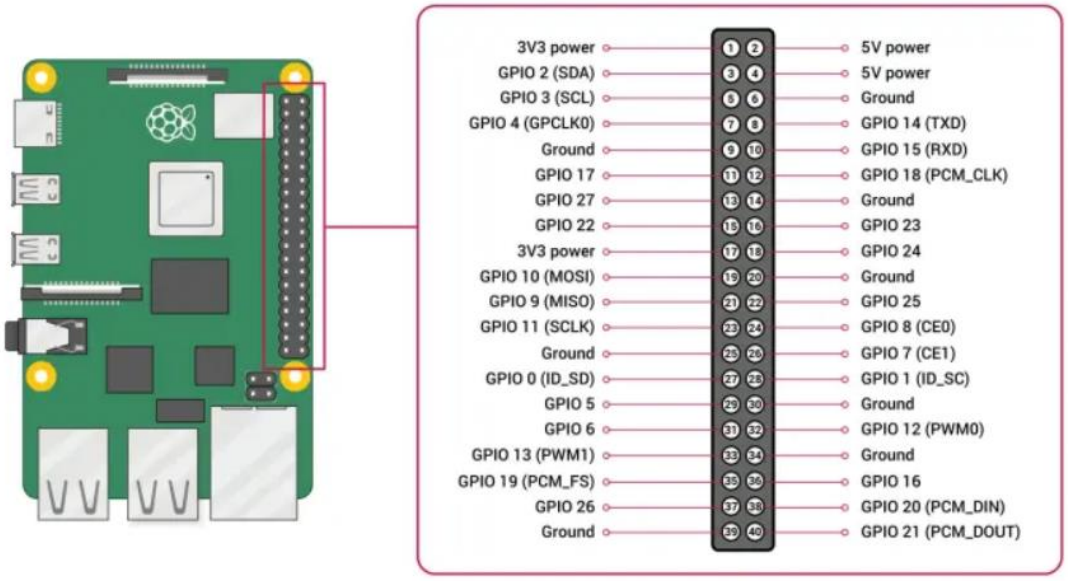
Scan license plate with Picamera then put it into Raspberry Pi for processing, use Opencv library with Python language to scan license plate from camera photo. Then upload to MySQL database. The lever system will open when the RFID tag is valid and the license plate is identifiable.

Chapter 2. **System design**

**1.** Hardware design

1.1. Block diagram

1.1.1. Processor: Raspberry Pi 4B




- **GPIO**:
  - 2 pins have a voltage level of 3.3 V (pin 1 and pin number 17).
  - 2 pins have a voltage of 5V (pin 2, pin 4).
  - 8 Ground pins (0V) (pins 6, 9, 14, 20, 25, 30, 34 and 39).
  - 28 GPIO pins (pin numbers 3, 5, 7, 8, 10, 11, 12, 13, 15, 16, 18, 19, 21, 22, 23, 24, 26, 27, 28, 29, 31, 32, 33, 35, 36, 37, 38, 40).
- **I2C**: Raspberry Pi uses the I2C communication standard to communicate with devices compatible with Inter-Integrated Circuit (a low-speed two-wire serial communication protocol). This communication standard requires a master-slave role between both devices. I2C has two connections: SDA (Serial Data) and SCL

(Serial Clock). They work by sending data to and using the SDA connection, and the data transfer rate is controlled via the SCL pin.

  - Data: (GPIO 2), Clock (GPIO 3)
  - EEPROM Data: (GPIO 0), EEPROM Clock (GPIO 1)

- **UART**: Universal Asynchronous Receiver/Transmitter (UART) or UART (Universal Asynchronous Receiver/Transmitter) pins provide a method of contact between two microcontrollers and controllers or computers. The TX pin is used to transmit serial data and the RX pin is used to receive serial data coming from another serial device. There are 2 pins related to the UART contact.
  - TX (GPIO14)
  - RX (GPIO15)
- **SPI**: SPI (Serial Peripheral Interface) is a protocol used for master-slave communication. Raspberry Pi uses this protocol for rapid communication between one or more peripheral devices. Data is synchronized using a clock (SCLK at GPIO pin 11) from the master device (Raspberry Pi) and data is sent from the Raspberry Pi to the SPI device using the MOSI (Master Out Slave In) pin. If the SPI device needs to communicate with the Raspberry Pi again, it sends data back using the MISO (Master In Slave Out) pin. There are 5 pins involved in SPI communication:
  - GND: Connect all GND pins of all slave components and the Raspberry Pi 4 board together.
  - SCLK: SPI clock signal. Connect all SCLK pins together.
  - MOSI (Master Out Slave In): This pin is used to send data from master to slave.
  - MISO (Master In Slave Out): This pin is used to receive data from slave to master.

- CE (Chip Enable): We need to connect a CE pin to each slave device (or peripherals) in our circuit. By default we have two CE pins but we can configure more CE pins from other available GPIO pins.
- SPI pins on the Raspberry Pi 4 board:
    + MOSI (Master Out Slave In) - GPIO10 (Pin 19)
    + MISO (Master In Slave Out) - GPIO9 (Pin 21)
    + SCLK (Serial Clock) - GPIO11 (Pin 23)
    + CE0 (Chip Enable 0) - GPIO8 (Pin 24)
    + CE1 (Chip Enable 1) - GPIO7 (Pin 26)

**PWM**: PWM (Pulse Width Modulation) is a common technique used to vary the width of pulses in a pulse train. PWM has many applications like controlling the brightness of LEDs, controlling the speed of DC motors, controlling servo motors or where you have to get analog output using digital devices.

- Software PWM is available on all pins
- Hardware PWM is only available on these pins: GPIO 12, GPIO 13, GPIO 18, GPIO 19

#### 1.1.2. Camera: Picamera

- Compatible with all Raspberry Pi versions 1, 2 and 3; and of course the indispensable Zero series (just change the FFC cable)
- 5MP Camera Module - OV5647 from OmniVision Technologies
- Capture still images at 2592 x 1944 resolution
- Supports video recording 1080p at 30fps, 720p at 60fps and 640x480p 60/90
- Connect the MIPI camera to the 15-pin serial interface, plug it directly into the Raspberry Pi board
- Dimensions 20mm x 25mm x 9mm

- Weighs 3g

1.1.3. RFID: Use RFID RC522, it comes with an RFID tag and key fob that includes 1KB of memory.

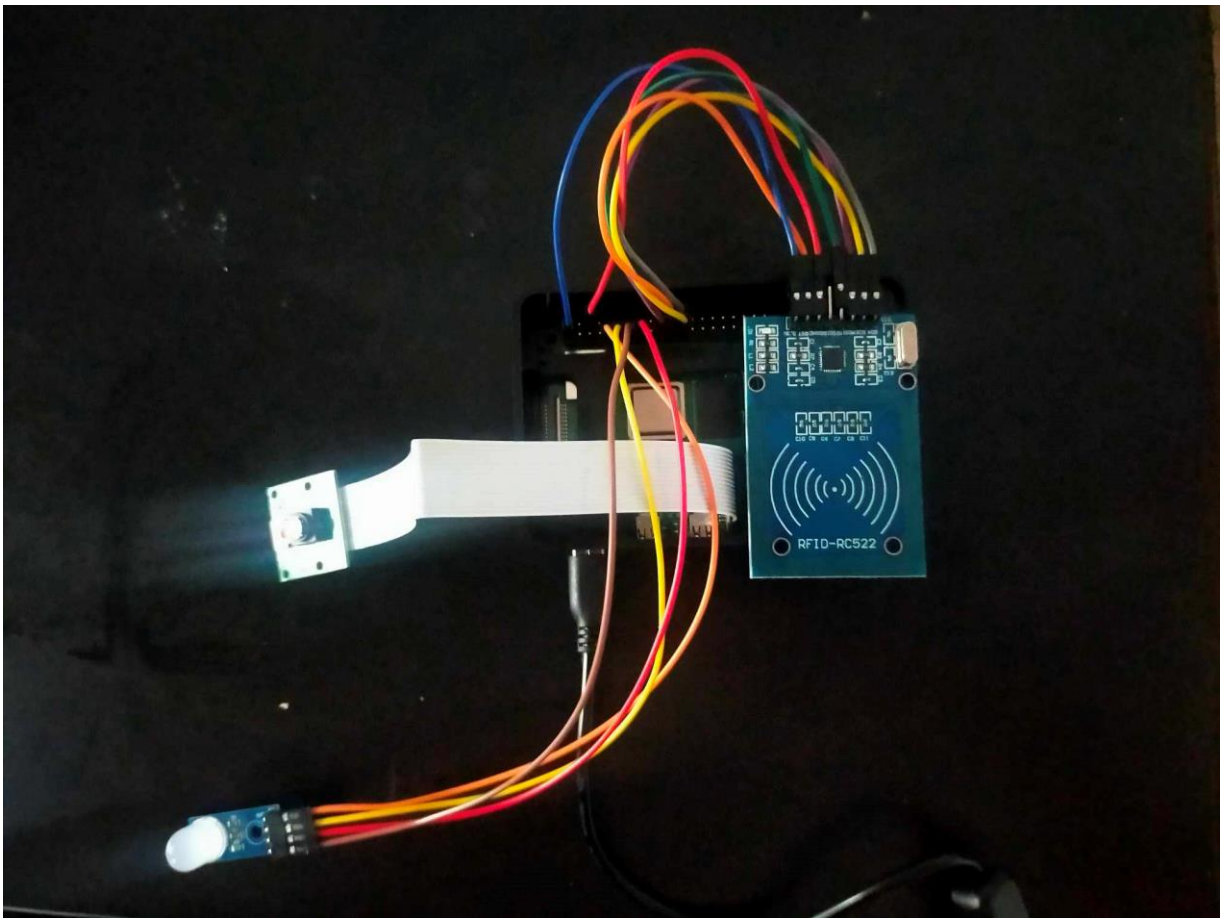
- Features and characteristics:
  - RFID RC522 uses mutual induction for tag activation and 13.56MHz for data transmission.
  - RFID tags can be used from both sides of the module at a maximum distance of 5cm.
  - A single 3.3V is required to activate the device.
  - Its automatic sleep mode makes it less power consuming module.
  - The module has three types of communication (UART, SPI, I2C). Therefore, it can be used with almost any microcontroller or device on the market.
  - The RFID tag and reader (RC522) can transmit data up to 10Mb/s.
- Pins: There are 8 pins:
  - VCC: The source pin is VCC. In some versions of the RC522, this pin is denoted as 3V3 on the module instead of VCC.
  - RST: This is the reset pin for the module. Therefore, it resets the device in case of an error when the device does not give any response.
  - GND: Ground helps create common ground with any external device, for example Power Supply or microcontroller.
  - IRQ: The device can go into sleep mode to save power. So IRQ helps to wake it up.
  - MISO: This pin connects to the microcontroller for SPI communication. However, it transfers data from the module to the microcontroller.
  - The MISO pin can also be used for other functions instead of SPI.
  - It can also communicate with I2C for clock pulses and UART Serial for data transmission from the module.

- MOSI: MOSI is the data input pin for the RFID module in SPI communication
- SCK: SCK pins help send clock pulses in SPI communication.
- SS: SS pin is a chip enable pin in SPI communication. Therefore, it receives a signal when the Master (Raspberry Pi Pico) has to do SPI communication. The SS pin in RFID can be used as a second pin (SDA) for I2C communication. It also receives data during UART communication.

- RFID tag:
  - The RFID tag is a memory storage device with 1KB worth of memory. This memory is divided into 16 sectors (0-15), where each sector is divided into 4 blocks (0,1,2,3). Each block is 16 bytes each. So 4 blocks x 16 bytes x 16 sectors = 1024 bytes is 1KB
- Led RGB:

- 10mm RGB 3-color led module used to make indicator lights and warning lights. The module uses 1 10mm LED with 3 basic colors: Green, Red and Blue (RGB). The above 3 colors can be used in combination to create other colors in the color palette.
- Operating voltage: 5VDC
- RGB LED 10mm round opaque
- Led common cathode pin (GND).
- Dimensions: 25 x 12mm.

## 1.2. Circuit:




1.2.1. RFID connect:

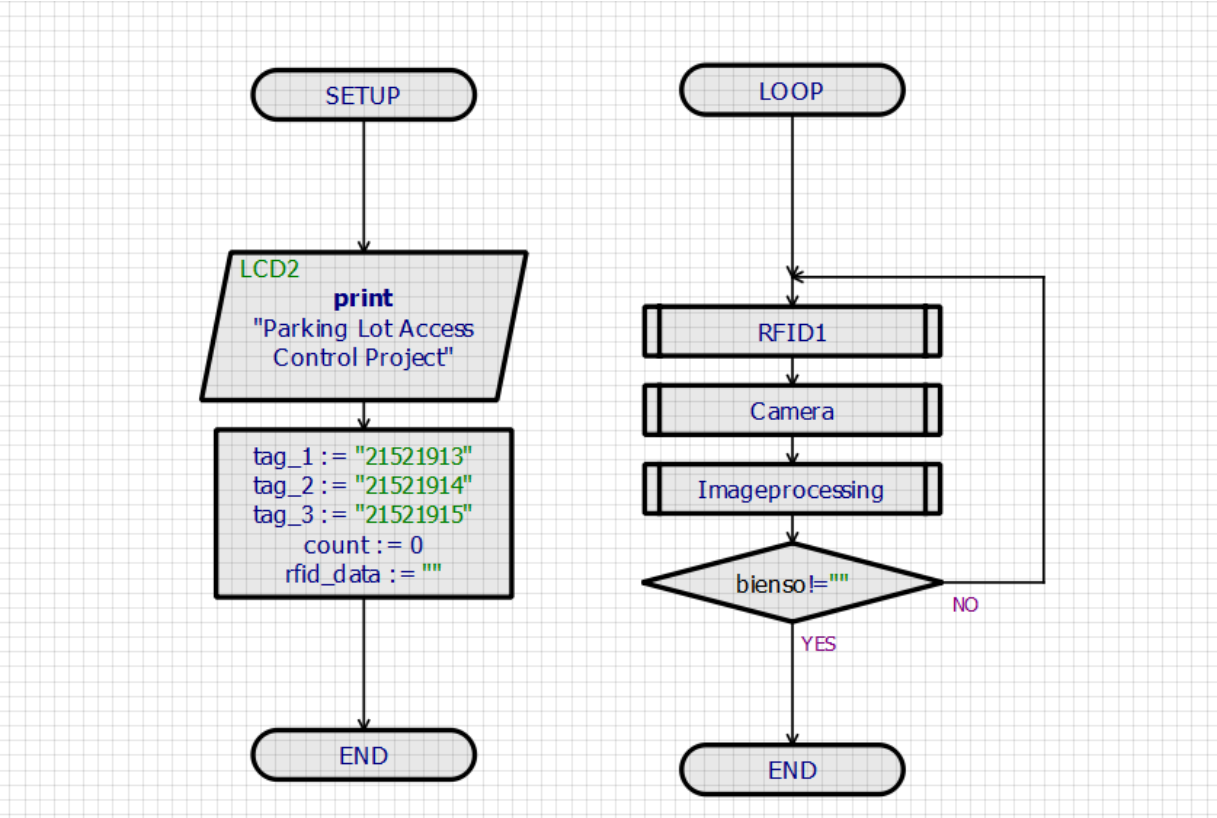
- SDA connects to Pin 24 (GPIO 8)
- SCK connects to Pin 23 (GPIO 11 – SCLK)
- MOSI connects to Pin 19 (GPIO 10 – MOSI)
- MISO connects to Pin 21 (GPIO 9 - MISO)
- GND connects to Pin 6 (GND)
- RST connects to Pin 22 (GPIO 25)
- 3.3v connects to Pin 1 (3.3V)

1.2.2. Led RGB connect:

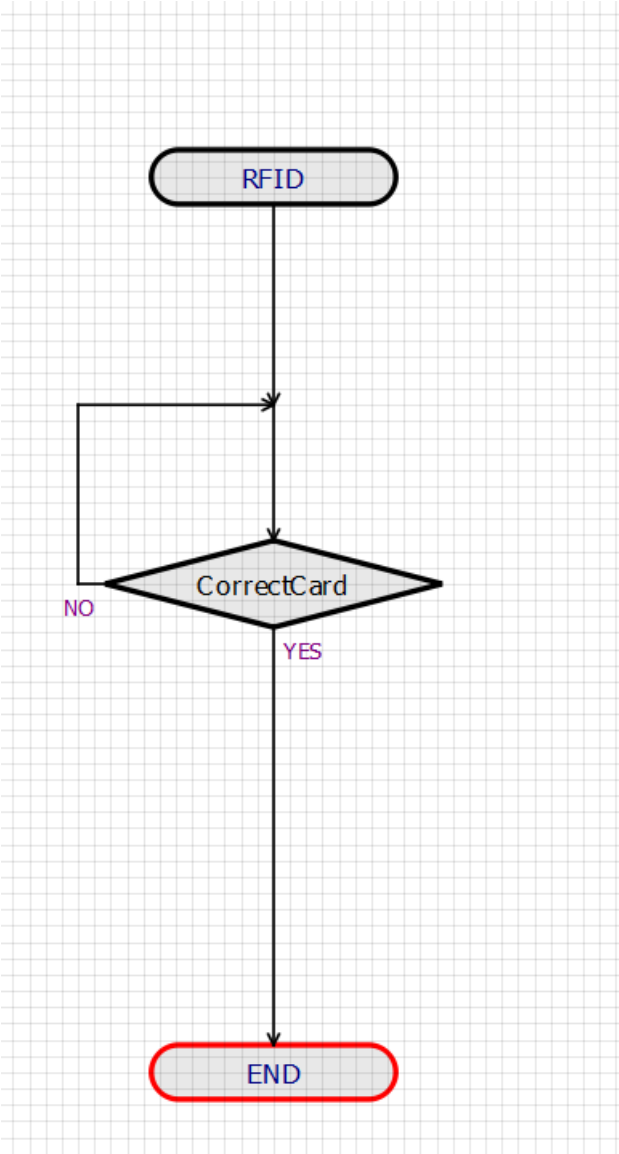
- GND connects to Pin 9(GND)
- R connects to Pin 11 (GPIO 17)
- G connects to Pin 13 (GPIO 27)
- B connects to Pin 15 (GPIO 22)

**2.** Software design

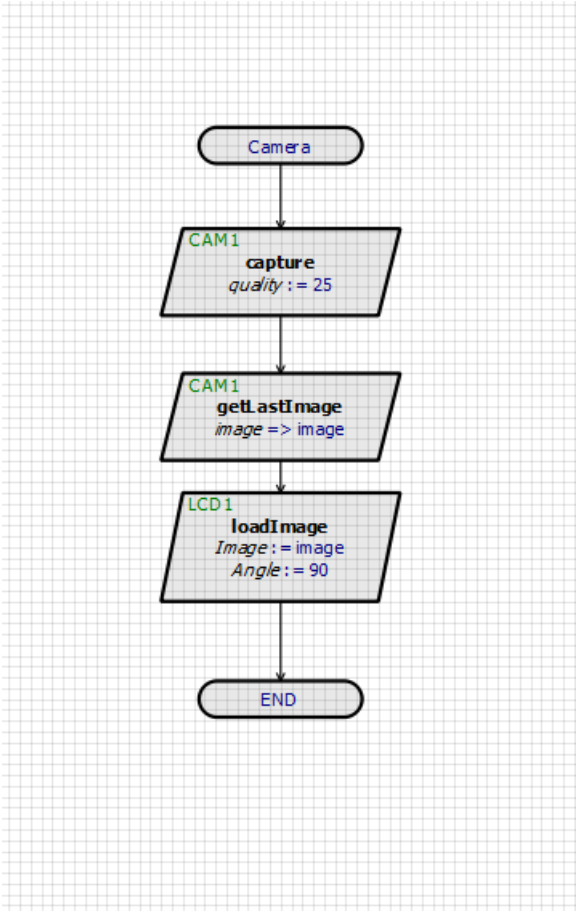
1. Flowchart:



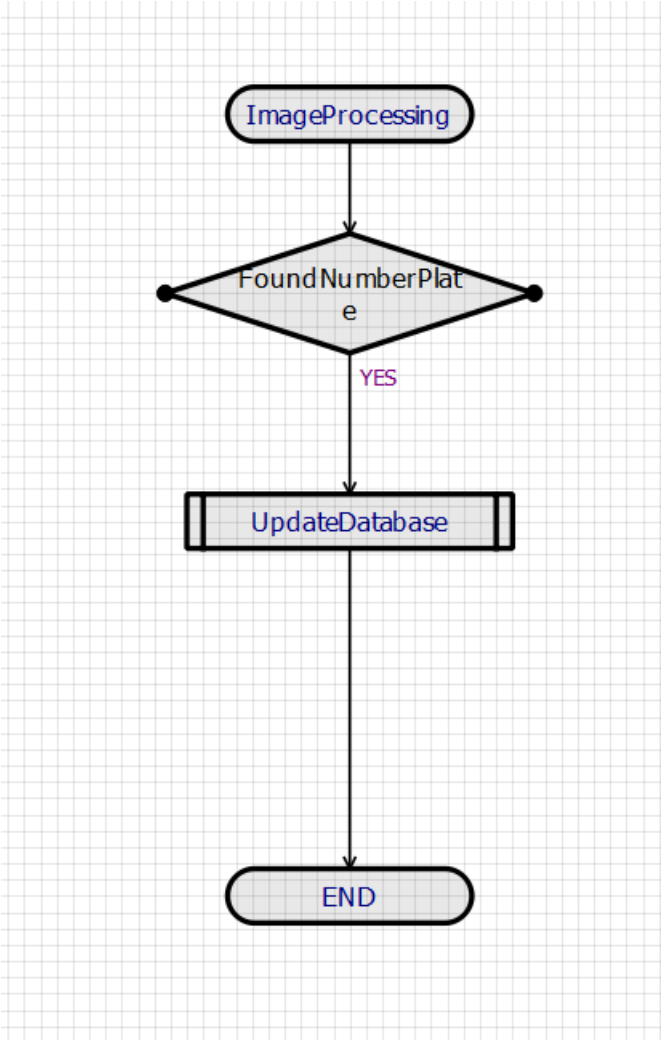

*Project's flowchart*




*RFID flowchart*




*Camera flowchart*




*ImageProcessing flowchart*

2. Full Code:

```
import cv2
from PIL import Image
import mysql.connector
import datetime
import numpy as np
import time
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
import board
```

```
   import neopixel
from gpiozero import LED
green = LED(27)
red = LED(17)
GAUSSIAN_SMOOTH_FILTER_SIZE = (5, 5)
ADAPTIVE_THRESH_BLOCK_SIZE = 19
ADAPTIVE_THRESH_WEIGHT = 9
ADAPTIVE_THRESH_BLOCK_SIZE = 19
ADAPTIVE_THRESH_WEIGHT = 9

n = 1

Min_char = 0.01
Max_char = 0.09

RESIZED_IMAGE_WIDTH = 20
RESIZED_IMAGE_HEIGHT = 30
#Connect to Mysql Database
def connectDB():
    con = mysql.connector.connect(
        host = 'localhost',
        user = 'root',
        password = '21521913',
        database = 'inoutcontrol'
    )
    return con
def checkNp(number_plate):
    con = connectDB()
    cursor = con.cursor()
    sql = "SELECT * FROM licenseplate WHERE number_plate = %s"
    cursor.execute(sql, (number_plate,))
    cursor.fetchall()
    result = cursor._rowcount
    con.close()
    cursor.close()
    return result
# Check tên biển số và trạng thái của bản ghi gần nhất đọc từ
hình ảnh lưu vào thư mục images
```

```
def checkNpStatus(number_plate):
    con = connectDB()
    cursor = con.cursor()
    sql = "SELECT * FROM licenseplate WHERE number_plate = %s
ORDER BY date_in DESC LIMIT 1"
    cursor.execute(sql, (number_plate,))
    result = cursor.fetchone()
    con.close()
    cursor.close()
    return result

#Tạo bản ghi dành cho xe vào bãi giữ xe (Cho xe vào bãi)
#TH1: Tên biển số xe đọc từ ảnh chưa tồn tại trong database
#TH2: Tên biển số xe đọc từ ảnh đã tồn tại trong database

def insertNp(number_plate, key):
    con = connectDB()
    cursor = con.cursor()
    sql = "SELECT COUNT(*) FROM licenseplate"
    cursor.execute(sql)
    result = cursor.fetchone()
    num_result = ''
    char_result = str(result)
    for c in char_result:
        if c.isnumeric():
            num_result = num_result + c;
    char_result = str(int(num_result) + 1)
    sql = "INSERT INTO licenseplate(ID, number_plate, status,
date_in, RFID) VALUES(%s,%s,%s,%s, %s)"
    now = datetime.datetime.now()
    date_in = now.strftime("%Y/%m/%d %H:%M:%S")
    key_str = str(key)
    cursor.execute(sql, (char_result,number_plate, '1',
date_in, key_str))
    con.commit()
    cursor.close()
    con.close()
    print('VAO BAI GUI XE')
```

```
    print('Ngay gio vao: ' +
datetime.datetime.strftime(datetime.datetime.now(), "%Y/%m/%d
%H:%M:%S"))

#Cập nhật bản ghi (Cho xe ra khỏi bãi)
def updateNp(Id):
    con = connectDB()
    cursor = con.cursor()
    sql = "UPDATE licenseplate SET status = 0, date_out = %s
WHERE Id = %s"
    now = datetime.datetime.now()
    date_out = now.strftime("%Y/%m/%d %H:%M:%S")
    cursor.execute(sql, (date_out, Id))
    con.commit()
    cursor.close()
    con.close()
    print('RA KHOI BAI GUI XE')
    print('Ngay gio ra: ' +
datetime.datetime.strftime(datetime.datetime.now(), "%Y/%m/%d
%H:%M:%S"))

def datain(number_plate, key):
    if(number_plate != ''):
        #Gọi hàm kiểm tra xem biển số đã tồn tại trong database
chưa
        check = checkNp(number_plate)
        if(check == 0):
            #Nếu xe chưa từng đến gửi tại bãi thì gọi hàm
insertNp để cho xe vào gửi
            insertNp(number_plate, key)
            led_accepted()
        else:
            #Gọi hàm kiểm tra trạng thái của xe
            check2 = checkNpStatus(number_plate)
            #Nếu trạng thái của xe = 1(xe vào gửi và chưa lấy
ra)
            if(check2[2] == 1):
                #Gọi hàm updateNp lấy xe ra và cập nhật trạng
thái cho xe = 0
```

```
                if(check2[5] == str(key)):
                    updateNp(check2[0])
                    led_accepted()
                else:
                    print("Sai the")
                    led_denied()

            #Nếu trạng thái của xe  = 0 (xe vào gửi và lấy ra
rồi)
            else:
                #Gọi hàm insertNp để cho xe vào gửi
                insertNp(number_plate, key)
                led_accepted()
    else:
        print('Bien so khong xac dinh')
        led_denied()

def preprocess(imgOriginal):

    imgGrayscale = extractValue(imgOriginal)
# Trả về giá trị cường độ sáng, hàm extractValue chỉ trả về giá
trị điểm ảnh, không có giá trị màu ==> ảnh gray
    imgMaxContrastGrayscale = maximizeContrast(imgGrayscale)
#để làm nổi bật biển số hơn, dễ tách khỏi nền
    height, width = imgGrayscale.shape # Lấy ra giá trị height
và width của hình ảnh

    imgBlurred = np.zeros((height, width, 1), np.uint8)
    #tạo một hình ảnh trắng hoặc đen có kích thước height x
width pixels với một kênh màu duy nhất (grayscale) và kiểu dữ
liệu (dtype) là uint8.
    imgBlurred = cv2.GaussianBlur(imgMaxContrastGrayscale,
GAUSSIAN_SMOOTH_FILTER_SIZE, 0)
    #Làm mịn ảnh bằng bộ lọc Gauss 5x5, sigma = 0

    imgThresh = cv2.adaptiveThreshold(imgBlurred, 255.0,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,
ADAPTIVE_THRESH_BLOCK_SIZE, ADAPTIVE_THRESH_WEIGHT)
    #Tạo ảnh nhị phân
```

```
    return imgGrayscale, imgThresh
    #Trả về ảnh xám và ảnh nhị phân
# end function

#####################################################################
###################################
def extractValue(imgOriginal):
    height, width, numChannels = imgOriginal.shape
    #Lấy ra chiều cao, chiều rộng và số kênh màu
    imgHSV = np.zeros((height, width, 3), np.uint8)
    #tạo một mảng trắng (tất cả giá trị là 0) với kích thước
(height, width, 3) và kiểu dữ liệu là uint8 (unsigned 8-bit
integer)
    #được sử dụng để lưu trữ hình ảnh trong không gian màu HSV

    imgHSV = cv2.cvtColor(imgOriginal, cv2.COLOR_BGR2HSV)
    #chuyển đổi hình ảnh ban đầu từ không gian màu BGR (Blue,
Green, Red) sang không gian màu HSV (Hue, Saturation, Value)

    imgHue, imgSaturation, imgValue = cv2.split(imgHSV)
    #màu sắc, độ bão hòa, giá trị cường độ sáng
    #Không chọn màu RBG vì vd ảnh màu đỏ sẽ còn lẫn các màu
khác nữa nên khó xác định ra "một màu"

    return imgValue
# end function


def check_firstline(first_line):
    if(len(first_line) != 4):
            return False
    if len(first_line) == 4:
        if(first_line[2].isalpha() == False or
first_line[0].isnumeric() == False or first_line[1].isnumeric()
== False or first_line[3].isnumeric() == False):
            return False
    return True
def check_secondline(second_line):
    if len(second_line) != 5:
```

```
        return False
    if second_line.isnumeric():
        return True;
    return False
#####################################################################
###################################
def maximizeContrast(imgGrayscale):
    #Làm cho độ tương phản lớn nhất
    height, width = imgGrayscale.shape

    imgTopHat = np.zeros((height, width, 1), np.uint8)
    imgBlackHat = np.zeros((height, width, 1), np.uint8)
    structuringElement =
cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) #tạo bộ lọc
kernel

    imgTopHat = cv2.morphologyEx(imgGrayscale,
cv2.MORPH_TOPHAT, structuringElement, iterations = 10) #nổi bật
chi tiết sáng trong nền tối
    imgBlackHat = cv2.morphologyEx(imgGrayscale,
cv2.MORPH_BLACKHAT, structuringElement, iterations = 10) #Nổi
bật chi tiết tối trong nền sáng
    imgGrayscalePlusTopHat = cv2.add(imgGrayscale, imgTopHat)
    imgGrayscalePlusTopHatMinusBlackHat =
cv2.subtract(imgGrayscalePlusTopHat, imgBlackHat)

    #Kết quả cuối là ảnh đã tăng độ tương phản
    return imgGrayscalePlusTopHatMinusBlackHat

# end function



#Đọc biển số xe lưu ở "images"
def readnumberplate(key):
    img =
cv2.imread("/home/duc3503/LuongDai/images/numberplate.jpg")
    number_plate = ''
```

```
    npaClassifications =
np.loadtxt("/home/duc3503/LuongDai/classifications.txt",
np.float32)
    npaFlattenedImages =
np.loadtxt("/home/duc3503/LuongDai/flattened_images.txt",
np.float32)
    npaClassifications = npaClassifications.reshape(
        (npaClassifications.size, 1))  # reshape numpy array to
1d, necessary to pass to call to train
    kNearest = cv2.ml.KNearest_create()  # instantiate KNN
object
    kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE,
npaClassifications)
    ##########################

    ################ Image Preprocessing #################
    imgGrayscaleplate, imgThreshplate = preprocess(img)
    canny_image = cv2.Canny(imgThreshplate, 250, 255)  # Canny
Edge
    kernel = np.ones((3, 3), np.uint8)
    dilated_image = cv2.dilate(canny_image, kernel,
iterations=1)  # Dilation

    ##########################################

    ###### Draw contour and filter out the license
plate  #############
    contours, hierarchy = cv2.findContours(dilated_image,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea,
reverse=True)[:10]  # Lấy 10 contours có diện tích lớn nhất

    listNp = {}
    screenCnt = []
    for c in contours:
        peri = cv2.arcLength(c, True)  # Tính chu vi
        approx = cv2.approxPolyDP(c, 0.06 * peri, True)  # làm
xấp xỉ đa giác, chỉ giữ contour có 4 cạnh
        [x, y, w, h] = cv2.boundingRect(approx.copy())
```

```
        ratio = w / h

        if (len(approx) == 4):
            screenCnt.append(approx)

            cv2.putText(img, str(len(approx.copy())), (x, y),
cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255, 0), 3)

    if screenCnt is None:
        detected = 0
        print("No plate detected")
    else:
        detected = 1


            cv2.drawContours(img, [screenCnt], -1, (0, 255, 0),
3)  # Khoanh vùng biển số xe

            ############## Find the angle of the license plate
#####################
            (x1, y1) = screenCnt[0, 0]
            (x2, y2) = screenCnt[1, 0]
            (x3, y3) = screenCnt[2, 0]
            (x4, y4) = screenCnt[3, 0]
            array = [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]
            sorted_array = array.sort(reverse=True, key=lambda
x: x[1])
            (x1, y1) = array[0]
            (x2, y2) = array[1]
            doi = abs(y1 - y2)
            ke = abs(x1 - x2)
            angle = 0
            if ke != 0:
                angle = math.atan(doi / ke) * (180.0 / math.pi)
            ######################################

            ########## Crop out the license plate and align it
to the right angle ################
```

```
            mask = np.zeros(imgGrayscaleplate.shape, np.uint8)
            new_image = cv2.drawContours(mask, [screenCnt], 0,
255, -1, )

            # Cropping
            (x, y) = np.where(mask == 255)
            (topx, topy) = (np.min(x), np.min(y))
            (bottomx, bottomy) = (np.max(x), np.max(y))

            roi = img[topx:bottomx, topy:bottomy]
            imgThresh = imgThreshplate[topx:bottomx,
topy:bottomy]
            ptPlateCenter = (bottomx - topx) / 2, (bottomy -
topy) / 2

            if x1 < x2:
                rotationMatrix =
cv2.getRotationMatrix2D(ptPlateCenter, -angle, 1.0)
            else:
                rotationMatrix =
cv2.getRotationMatrix2D(ptPlateCenter, angle, 1.0)

            roi = cv2.warpAffine(roi, rotationMatrix, (bottomy
- topy, bottomx - topx))
            imgThresh = cv2.warpAffine(imgThresh,
rotationMatrix, (bottomy - topy, bottomx - topx))
            roi = cv2.resize(roi, (0, 0), fx=3, fy=3)
            imgThresh = cv2.resize(imgThresh, (0, 0), fx=3,
fy=3)

            ########################################

            ##################### Prepocessing and Character
segmentation ####################
            kerel3 = cv2.getStructuringElement(cv2.MORPH_RECT,
(3, 3))
            thre_mor = cv2.morphologyEx(imgThresh,
cv2.MORPH_DILATE, kerel3)
```

```
            cont, hier = cv2.findContours(thre_mor,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
            cv2.drawContours(roi, cont, -1, (100, 255, 255),
2)  # Vẽ contour các kí tự trong biển số

            ##################### Filter out characters
#################
            char_x_ind = {}
            char_x = []
            height, width, _ = roi.shape
            roiarea = height * width

            for ind, cnt in enumerate(cont):
                (x, y, w, h) = cv2.boundingRect(cont[ind])
                ratiochar = w / h
                char_area = w * h

                if (Min_char * roiarea < char_area < Max_char *
roiarea) and (0.25 < ratiochar < 0.7):
                    if x in char_x:  # Sử dụng để dù cho trùng
x vẫn vẽ được
                        x = x + 1
                    char_x.append(x)
                    char_x_ind[x] = ind
            ############ Character recognition
##########################
            char_x = sorted(char_x)
            first_line = ""
            second_line = ""

            for i in char_x:
                (x, y, w, h) =
cv2.boundingRect(cont[char_x_ind[i]])
                cv2.rectangle(roi, (x, y), (x + w, y + h), (0,
255, 0), 2)
                imgROI = thre_mor[y:y + h, x:x + w]  # Crop the
characters
                imgROIResized = cv2.resize(imgROI,
(RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))  # resize image
```

```
                npaROIResized = imgROIResized.reshape(
                    (1, RESIZED_IMAGE_WIDTH *
RESIZED_IMAGE_HEIGHT))

                npaROIResized = np.float32(npaROIResized)
                _, npaResults, neigh_resp, dists =
kNearest.findNearest(npaROIResized,k=3)  # call KNN function
find_nearest;
                strCurrentChar =
str(chr(int(npaResults[0][0])))  # ASCII of characters
                cv2.putText(roi, strCurrentChar, (x, y + 50),
cv2.FONT_HERSHEY_DUPLEX, 2, (255, 255, 0), 3)
                if (y < height / 3):  # decide 1 or 2-line
license plate
                    first_line = first_line + strCurrentChar
                else:
                    second_line = second_line + strCurrentChar
            listNp[first_line] = second_line
            roi = cv2.resize(roi, None, fx=0.75, fy=0.75)

    for x in listNp:
        first_line = x
        second_line = listNp[x]
        if check_firstline(first_line) and
check_secondline(second_line):
            print("\n License Plate is: " + first_line + " - "
+ second_line + "\n")
            number_plate = first_line + " - " + second_line
            datain(number_plate,key)
            return
    print("Khong tim thay bien so xe\n")
    led_denied()

def led_accepted():
    green.on()
    time.sleep(2)
    green.off()
```

```
def led_denied():

    red.on()
    time.sleep()
    red.off()


reader = SimpleMFRC522()
cam = cv2.VideoCapture(0)

while True:
    #Đọc từng frame
    ret,frame=cam.read()
    #Thiết lập màu sắc cho ảnh
    framegray = cv2.cvtColor(frame, cv2.COLOR_BGR2BGRA)

    key, text = reader.read()

    if key == 157502735209 or key == 40551782013 or key ==
764270878022 or key == 762480893260:
        #Lưu ảnh vào thư mục images
        cv2.imwrite('/home/duc3503/LuongDai/images/numberplate.
jpg', framegray)
        #Gọi hàm xử lý "readnumberplate"
        readnumberplate(key)
    else:
        led_denied()
    time.sleep(1)

cam.release()
cv2.destroyAllWindows()
```

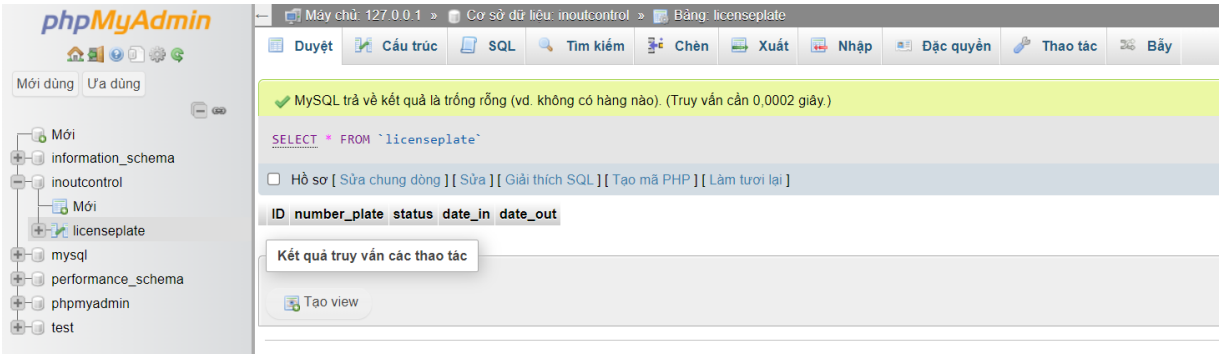
3. Code Explanation:

3.1. Working with databases

- Connect to the database for management (here the code has not been installed on RaspberryPi, so temporarily use MySQL on "xampp" via localhost

```python
#Hàm mở kết nối Mysql với database tên "inoutcontrol"
def connectDB():
    con = mysql.connector.connect(
        host = 'localhost',
        user = 'root',
        password = '',
        database = 'inoutcontrol'
    )
    return con
```

- Create a database named "inoutcontrol" and a table inside named "licenseplate" with 5 columns:
    + ID
    + number_plate: stores the license plate read from the image
    + status: saves the vehicle's status, status = 0 means the vehicle has entered and has not left the parking lot, status = 1 means the vehicle has left the parking lot
    + date_in: save the date and time the vehicle entered the parking lot
    + date_out: save the date and time the vehicle left the parking lot



- Function checkNp(number_plate): checks if the license plate you just read (number_plate) exists in the database by executing the sql query = "SELECT * FROM licenseplate WHERE number_plate = %s" then using the execute function to execute The query with the corresponding number plate data to be found from the number_plate variable is performed on the database.

```
#Check tên biển số đọc từ hình ảnh lưu vào thue mục "images" đã tồn tại trong database chưa
def checkNp(number_plate):
    con = connectDB()
    cursor = con.cursor()
    sql = "SELECT * FROM licenseplate WHERE number_plate = %s"
    cursor.execute(sql, (number_plate,))
    cursor.fetchall()
    result = cursor._rowcount
    con.close()
    cursor.close()
    return result
```

- Similarly, the function checkNpstatus(number_plate) executes the query "SELECT * FROM licenseplate WHERE number_plate = %s ORDER BY date_in DESC LIMIT 1" to return the status 1 or 0 of the entered license plate.

```
def checkNpStatus(number_plate):
    con = connectDB()
    cursor = con.cursor()
    sql = "SELECT * FROM licenseplate WHERE number_plate = %s ORDER BY date_in DESC LIMIT 1"
    cursor.execute(sql, (number_plate,))
    result = cursor.fetchone()
    con.close()
    cursor.close()
    return result
```

- The insertNp(number_plate) function is used to insert the license plate stored in the number_plate variable into the database with the value status = 1 and date_in taken from the datetime.datetime.now() function through the query "sql = "INSERT INTO licenseplate(number_plate, status, date_in, RFID) VALUES(%s,%s,%s,%s)"

```
def insertNp(number_plate, key):
    con = connectDB()
    cursor = con.cursor()
    sql = "SELECT COUNT(*) FROM licenseplate"
    cursor.execute(sql)
    result = cursor.fetchone()
    num_result = ''
    char_result = str(result)
    for c in char_result:
        if c.isnumeric():
            num_result = num_result + c;
    char_result = str(int(num_result) + 1)
    sql = "INSERT INTO licenseplate(ID, number_plate, status, date_in, RFID) VALUES(%s,%s,%s,%s, %s)"
    now = datetime.datetime.now()
    date_in = now.strftime("%Y/%m/%d %H:%M:%S")
    key_str = str(key)
    cursor.execute(sql, (char_result,number_plate, '1', date_in, key_str))
    con.commit()
    cursor.close()
    con.close()
    print('VAO BAI GUI XE')
    print('Ngay gio vao: ' + datetime.datetime.strftime(datetime.datetime.now(), "%Y/%m/%d %H:%M:%S"))
```

- The updateNp(Id) function is the function to update the license plate record (for vehicles to leave the parking lot) using the query "UPDATE licenseplate SET status = 0, date_out = %s WHERE Id = %s" to update status = 0 and date_out for the record

```
def updateNp(Id):
    con = connectDB()
    cursor = con.cursor()
    sql = "UPDATE licenseplate SET status = 0, date_out = %s WHERE Id = %s"
    now = datetime.datetime.now()
    date_out = now.strftime("%Y/%m/%d %H:%M:%S")
    cursor.execute(sql, (date_out, Id))
    con.commit()
    cursor.close()
    con.close()
    print('RA KHOI BAI GUI XE')
    print('Ngay gio ra: ' + datetime.datetime.strftime(datetime.datetime.now(), "%Y/%m/%d %H:%M:%S"))
```

- The datain(number_plate) function is the function that determines how the scanned license plate will be entered into the database. Use the checkNp function to check whether the scanned vehicle's license plate number has been placed in the parking lot or not. If checkNp = 0 means this vehicle is not in the parking lot, then perform the insertNp function to put the vehicle's license plate number into the database and turn on LED with green color to

ensure license plate signs and tags are valid. On the contrary, if the license plate already exists in the database, check its status. If status = 1, it means the vehicle is in the parking lot, if the key(ID of the RFID read in) is equal to the RFID field of the number_plate in question, it proves that the user is using the correct card then call the updateNp function to update status = 0, date_out for the vehicle and turn on LED with green color in 1 second  for the vehicle get out of the beach, on the contrary, the card id read is different from the card id stored in the database, meaning the user has swiped the wrong card, at which point the red light is turned on to signal invalidity. If the license plate already exists in the database and status = 0, it means that this vehicle has previously entered the parking lot and left the parking lot. Now we call the insertNp function to set up a new record for the vehicle and turn on Led with green color.

```python
def datain(number_plate, key):
    if(number_plate != ''):
        #Gọi hàm kiểm tra xem biển số đã tồn tại trong database chưa
        check = checkNp(number_plate)
        if(check == 0):
            #Nếu xe chưa từng đến gửi tại bãi thì gọi hàm insertNp để cho xe vào gửi
            insertNp(number_plate, key)
            led_accepted()
        else:
            #Gọi hàm kiểm tra trạng thái của xe
            check2 = checkNpStatus(number_plate)
            #Nếu trạng thái của xe = 1(xe vào gửi và chưa lấy ra)
            if(check2[2] == 1):
                #Gọi hàm updateNp lấy xe ra và cập nhật trạng thái cho xe = 0
                if(check2[5] == str(key)):
                    updateNp(check2[0])
                    led_accepted()
                else:
                    print("Sai the")
                    led_denied()

            #Nếu trạng thái của xe  = 0 (xe vào gửi và lấy ra rồi)
            else:
                #Gọi hàm insertNp để cho xe vào gửi
                insertNp(number_plate, key)
                led_accepted()
    else:
        print('Bien so khong xac dinh')
        led_denied()
```

## 3.2. Image processing

a. Training a character recognition system using images:

```python
imgTrainingNumbers = cv2.imread("training_chars.png")            # read in training numbers image

imgGray = cv2.cvtColor(imgTrainingNumbers, cv2.COLOR_BGR2GRAY)          # get grayscale image
imgBlurred = cv2.GaussianBlur(imgGray, (5,5), 0)                        # blur

                                            # filter image from grayscale to black and white
imgThresh = cv2.adaptiveThreshold(imgBlurred,                           # input image
                                  255,                                  # make pixels that pass the threshold full white
                                  cv2.ADAPTIVE_THRESH_GAUSSIAN_C,       # use gaussian rather than mean, seems to give better results
                                  cv2.THRESH_BINARY_INV,                # invert so foreground will be white, background will be black
                                  11,                                   # size of a pixel neighborhood used to calculate threshold value
                                  2)                                    # constant subtracted from the mean or weighted mean
```

- Read and preprocess training images:


*training_chars.png*

  - Reads an image containing training characters from the file "training_chars.png".
  - Convert images to grayscale images.
  - Apply the Gaussian Blur filter to blur the image.
- Convert grayscale images to binary images
  - Apply luminance intensity thresholding using adaptive thresholding to create a binary image, where the background is black and the characters are white.
- Find contours on binary images:
  - Use cv2.findContours to find boundaries on a binary image, make sure to use a copy since the function will modify this image in the course of finding contours.
  - Retrieve the outermost contours only, compress horizontal, vertical, and diagonal segments and leave only their end points

```
npaFlattenedImages =  np.empty((0, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT))

intClassifications = []# declare empty classifications list, this will be our list of how we are classifying our chars from user input, we will write to file at the end

                        # possible chars we are interested in are digits 0 through 9, put these in list intValidChars
intValidChars = [ord('0'), ord('1'), ord('2'), ord('3'), ord('4'), ord('5'), ord('6'), ord('7'), ord('8'), ord('9'),
                 ord('A'), ord('B'), ord('C'), ord('D'), ord('E'), ord('F'), ord('G'), ord('H'), ord('I'), ord('J'),
                 ord('K'), ord('L'), ord('M'), ord('N'), ord('O'), ord('P'), ord('Q'), ord('R'), ord('S'), ord('T'),
                 ord('U'), ord('V'), ord('W'), ord('X'), ord('Y'), ord('Z')] #Là mã ascii của mấy chữ này
```

- Declare empty numpy array, we will use this to write to file later ,zero rows, enough cols to hold all image data
- Declare empty classifications list, this will be our list of how we are classifying our chars from user input, we will write to file at the end
- The characters we might be interested in are the digits 0 to 9 and the capital letters 'A' to 'Z', put them in the list intValidChars

```
for npaContour in npaContours:                          # for each contour
    if cv2.contourArea(npaContour) > MIN_CONTOUR_AREA:          # if contour is big enough to consider
        [intX, intY, intW, intH] = cv2.boundingRect(npaContour)         # get and break out bounding rect
                                    # draw rectangle around each contour as we ask user for input
        cv2.rectangle(imgTrainingNumbers,           # draw rectangle on original training image
                      (intX, intY),                 # upper left corner
                      (intX+intW,intY+intH),        # lower right corner
                      (0, 0, 255),                  # red
                      2)                            # thickness
        imgROI = imgThresh[intY:intY+intH, intX:intX+intW]                                 # crop char out of threshold image
        imgROIResized = cv2.resize(imgROI, (RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))     # resize image, this will be more consistent for recognition and storage

        cv2.imshow("imgROI", imgROI)                    # show cropped out char for reference
        cv2.imshow("imgROIResized", imgROIResized)      # show resized image for reference

        cv2.imshow("training_numbers.png", imgTrainingNumbers)      # show training numbers image, this will now have red rectangles drawn on it

        intChar = cv2.waitKey(0)                     # get key press

        if intChar == 27:                   # if esc key was pressed
            sys.exit()                      # exit program
        elif intChar in intValidChars:      # else if the char is in the list of chars we are looking for . . .
            intClassifications.append(intChar)       # append classification char to integer list of chars (we will convert to float later before writing to file)
            #Là file chứa label của tất cả các ảnh mẫu, tổng cộng có 32 x 5 = 160 mẫu.
            npaFlattenedImage = imgROIResized.reshape((1, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT)) #flatten image to 1d numpy array so we can write to file later
            npaFlattenedImages = np.append(npaFlattenedImages, npaFlattenedImage, 0) #add current flattened impage numpy array to list of flattened image numpy arrays

fltClassifications = np.array(intClassifications, np.float32)                # convert classifications list of ints to numpy array of floats
npaClassifications = fltClassifications.reshape((fltClassifications.size, 1))   # flatten numpy array of floats to 1d so we can write to file later

print ("\n\ntraining complete !!\n")
np.savetxt("classifications.txt", npaClassifications)           # write flattened images to file
np.savetxt("flattened_images.txt", npaFlattenedImages)          #
cv2.destroyAllWindows()             # remove windows from memory

return
```

- Browse through each remaining contour:
  - If contour is big enough to consider,  get and break out bounding rect
  - Draw rectangle around each contour as we ask user for inputdraw rectangle on original training image by cv2.rectangle() function with 4 values: upper left corner, lower right corner, color, thickness.

- Crop characters out of threshold image and resize image, this will be more consistent for recognition and storage.
- If the "esc" key is pressed then close the program, else if the char is in the list of chars we are looking for, append classification char to integer list of chars (we will convert to float later before writing to file), flatten image to 1d numpy array so we can write to file later and add current flattened impage numpy array to list of flattened image numpy arrays.
- Convert classifications list of ints to numpy array of floats, flatten numpy array of floats to 1d so we can write to file later
- Save ASCII float values to classifications.txt file and save flattened image data to flattened_images.txt file.

b. Image preprocessing: The “preprocess(imgOriginal)” function is an image preprocessing function that returns two values: gray image and binary image from the original image(imgOriginal)

```
def preprocess(imgOriginal):

    imgGrayscale = extractValue(imgOriginal)
    # Trả về giá trị cường độ sáng, hàm extractValue chỉ trả về giá trị điểm ảnh, không có giá trị màu ==> ảnh gray
    imgMaxContrastGrayscale = maximizeContrast(imgGrayscale) #để làm nổi bật biên số hơn, dễ tách khỏi nền
    height, width = imgGrayscale.shape # Lấy ra giá trị height và width của hình ảnh

    imgBlurred = np.zeros((height, width, 1), np.uint8)
    #tạo một hình ảnh trắng hoặc đen có kích thước height x width pixels với một kênh màu duy nhất (grayscale) và kiểu dữ liệu (dtype) là uint8.
    imgBlurred = cv2.GaussianBlur(imgMaxContrastGrayscale, GAUSSIAN_SMOOTH_FILTER_SIZE, 0)
    #Làm mịn ảnh bằng bộ lọc Gauss 5x5, sigma = 0

    imgThresh = cv2.adaptiveThreshold(imgBlurred, 255.0, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, ADAPTIVE_THRESH_BLOCK_SIZE, ADAPTIVE_THRESH_WEIGHT)
    #Tạo ảnh nhị phân

    return imgGrayscale, imgThresh
    #Trả về ảnh xám và ảnh nhị phân
# end function
```

c. Extract value: The function "extractValue(imgOriginal)" is used to extract the light intensity value (value) from the original image (imgOriginal) in HSV color space (Hue, Saturation, Value).

```
###################################################################################################
def extractValue(imgOriginal):
    height, width, numChannels = imgOriginal.shape
    #Lấy ra chiều cao, chiều rộng và số kênh màu
    imgHSV = np.zeros((height, width, 3), np.uint8)
    #tạo một mảng trắng (tất cả giá trị là 0) với kích thước (height, width, 3) và kiểu dữ liệu là uint8 (unsigned 8-bit integer)
    #được sử dụng để lưu trữ hình ảnh trong không gian màu HSV

    imgHSV = cv2.cvtColor(imgOriginal, cv2.COLOR_BGR2HSV)
    #chuyển đổi hình ảnh ban đầu từ không gian màu BGR (Blue, Green, Red) sang không gian màu HSV (Hue, Saturation, Value)

    imgHue, imgSaturation, imgValue = cv2.split(imgHSV)
    #màu sắc, độ bão hòa, giá trị cường độ sáng
    #Không chọn màu RBG vì vd ảnh màu đỏ sẽ còn lẫn các màu khác nữa nên khó xác định ra "một màu"

    return imgValue
# end function
```

d. Makes for maximum contrast

```
def maximizeContrast(imgGrayscale):
    #Làm cho độ tương phản lớn nhất
    height, width = imgGrayscale.shape

    imgTopHat = np.zeros((height, width, 1), np.uint8)
    imgBlackHat = np.zeros((height, width, 1), np.uint8)
    structuringElement = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) #tạo bộ lọc kernel

    imgTopHat = cv2.morphologyEx(imgGrayscale, cv2.MORPH_TOPHAT, structuringElement, iterations = 10) #nổi bật chi tiết sáng trong nền tối
    imgBlackHat = cv2.morphologyEx(imgGrayscale, cv2.MORPH_BLACKHAT, structuringElement, iterations = 10) #Nổi bật chi tiết tối trong nền sáng
    imgGrayscalePlusTopHatMinusBlackHat = cv2.subtract(imgGrayscalePlusTopHat, imgBlackHat)

    #Kết quả cuối là ảnh đã tăng độ tương phản
    return imgGrayscalePlusTopHatMinusBlackHat

# end function
```

e. Image processing

- Initialize the readnumberplate() function to read the license plate from the image

```
def readnumberplate():
    img = cv2.imread("D:/XulyAnh/images/numberplate.jpg")
    number_plate = ''
    npaClassifications = np.loadtxt("classificationS.txt", np.float32)
    npaFlattenedImages = np.loadtxt("flattened_images.txt", np.float32)
    npaClassifications = npaClassifications.reshape(
        (npaClassifications.size, 1))  # reshape numpy array to 1d, necessary to pass to call to train
    kNearest = cv2.ml.KNearest_create()  # instantiate KNN object
    kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)
    ##########################
```

- Training:
  - Use reshape() function reshape numpy array to 1d, necessary to pass to call to train.

- Create an object of the K-Nearest Neighbors (KNN) class in the OpenCV library and assign it to the variable kNearest. This is the step to initialize the K-Nearest Neighbors (KNN) model in the OpenCV library to train a character recognition model.
- Perform the KNN model training process by passing training data into the model using the "train()" function. Specifically:
  + "npaFlattenedImages": Is an array containing flattened image data, each line of the array corresponds to a character image.
  + "cv2.ml.ROW_SAMPLE": This argument specifies how the data is organized. In this case, ROW_SAMPLE represents that each row of the npaFlattenedImages array corresponds to a sample of data (a character image).
  + "npaClassifications": Is an array containing labels corresponding to each data sample (character image) in npaFlattenedImages.

⇨ Complete training

- Image Preprocessing

```
imgGrayscaleplate, imgThreshplate = preprocess(img)
canny_image = cv2.Canny(imgThreshplate, 250, 255)  # Canny Edge
kernel = np.ones((3, 3), np.uint8)
dilated_image = cv2.dilate(canny_image, kernel, iterations=1)  # Dilation
```

- Use the imgGrayscaleplate variable to store the grayscale image and imgThreshplate to store the binary image returned by the preprocess function when the captured image is processed by this function.
- Using the Canny algorithm (Canny edge detection) to detect edges, we can use the function available in the opencv library, the function "cv2.Canny()", this function includes 3 parameters:

+ "imgThreshplate": This is a threshold image, a binary image (contains only two pixel values: 0 and 255)

+ 250: This is the lower threshold of the Canny algorithm. Any gradient value lower than this threshold will be discarded and considered background. This creates weak "edge segments".

+ 255: This is the high threshold of the Canny algorithm. Any gradient value higher than this threshold will be considered an intense edge segment.

- Create a 3x3 kernel with data type np.uint8 then perform the Dilation transformation on the canny_image using the kernel created in the previous step. Dilation is performed by placing the kernel over each part of the image and dragging it out to expand the white area in the image.

- Draw contour and filter out the license plate:

```
contours, hierarchy = cv2.findContours(dilated_image, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]  # Lấy 10 contours có diện tích lớn nhất
```

- Use the findContours() function in the opencv library to find the boundaries in the dilated_image image. This function returns 2 values:

+ contours: A list containing all the contours found in the image.

+ hierarchy: A hierarchy tree contains information about the relationships between contours. Hierarchy is often used to identify parent contours, child contours, neighbor contours, etc.

Here we only use contours without hierarchy

- Use the sorted function to sort the contours by area then get the 10 contours with the largest area.

```
listNp = {}
screenCnt = []
for c in contours:
    peri = cv2.arcLength(c, True)  # Tính chu vi
    approx = cv2.approxPolyDP(c, 0.06 * peri, True)  # làm xấp xỉ đa giác, chỉ giữ contour có 4 cạnh
    [x, y, w, h] = cv2.boundingRect(approx.copy())
    ratio = w / h
    # cv2.putText(img, str(len(approx.copy())), (x,y),cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255, 0), 3)
    # cv2.putText(img, str(ratio), (x,y),cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255, 0), 3)
    if (len(approx) == 4):
        screenCnt.append(approx)
```

- Create variable listNp as an empty dictionary
- Create a ScreenCnt array to store approximate polygons
- Browse each contour
- Calculate the perimeter (perimeter) of contour c. The second parameter True represents that the contour is closed, which means that the line segment represented by the contour starts and ends at the same point, so it is a closed polygon. The result is the perimeter value of the contour.
- Use the approxPolyDP() function of the opencv library to approximate contour c with a polygon with fewer sides. This approximation can help reduce contour complexity. The second parameter 0.06 * peri is the approximation intercept, which determines the degree of approximation. The third parameter True represents a closed contour. The result of this line of code is the approximate polygon of the contour.
- Calculate the rectangle bounding the polygon approximately approx using the “cv2.boundingRect()” function. The result is the coordinates (x, y) of the top left corner of the rectangle and the dimensions (w, h) of the rectangle, then calculate "ratio" which is the ratio between width (w) and height (h) of the rectangle surrounding the polygon approximately. This ratio is often used to check whether a rectangle has a ratio close to 1 (a square).
- Check if the approximate polygon "approx" has exactly 4 sides. If so, put "approx" in the array "ScreenCnt"

```
if screenCnt is None:
    detected = 0
    print("No plate detected")
else:
    detected = 1
```

- If the ScreenCnt array is empty, it means that no approximate polygons are found in the image, which means no license plates are found in the image, then the detected flag receives the value 0.
- Conversely, if there are approximate polygons stored in the ScreenCnt array, the detected flag will be set to 1

- Found the approximate polygon in the picture:

```
if detected == 1:

    for screenCnt in screenCnt:
        cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)  # Khoanh vùng biển số xe

        ############## Find the angle of the license plate ####################
        (x1, y1) = screenCnt[0, 0]
        (x2, y2) = screenCnt[1, 0]
        (x3, y3) = screenCnt[2, 0]
        (x4, y4) = screenCnt[3, 0]
        array = [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]
        sorted_array = array.sort(reverse=True, key=lambda x: x[1])
        (x1, y1) = array[0]
        (x2, y2) = array[1]
        doi = abs(y1 - y2)
        ke = abs(x1 - x2)
        angle = math.atan(doi / ke) * (180.0 / math.pi)
```

- Browse each ScreenCnt
- Use the drawContours() function of the opencv library to delineate the license plate area
- Get coordinates of corner points on the image:
  + (x1, y1) = screenCnt[0, 0]: Get the coordinates of the first corner point from the screenCnt list.

+ (x2, y2) = screenCnt[1, 0]: Get the coordinates of the second corner point.
+ (x3, y3) = screenCnt[2, 0]: Get the coordinates of the third corner point.
+ (x4, y4) = screenCnt[3, 0]: Get the coordinates of the fourth corner point.

- Create an "array" list containing the coordinates of the corner points.
- Sort based on y value in descending order (from large to small), so the point with the highest y value will be at the top of the list after sorting, then get the coordinates of the two highest points after sorting arrange.
- Then calculate the angle between the line segment connecting the two points (x1, y1) and (x2, y2) and the horizontal axis according to the formula: $angle = \arctan\left(\frac{opposite}{adjacent}\right) * \frac{180}{\pi}\ (degrees)$
- Crop out the license plate and align it to the right angle:

```
mask = np.zeros(imgGrayscaleplate.shape, np.uint8)
new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1, )

# Cropping
(x, y) = np.where(mask == 255)
(topx, topy) = (np.min(x), np.min(y))
(bottomx, bottomy) = (np.max(x), np.max(y))

roi = img[topx:bottomx, topy:bottomy]
imgThresh = imgThreshplate[topx:bottomx, topy:bottomy]
ptPlateCenter = (bottomx - topx) / 2, (bottomy - topy) / 2

if x1 < x2:
    rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, -angle, 1.0)
else:
    rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, angle, 1.0)

roi = cv2.warpAffine(roi, rotationMatrix, (bottomy - topy, bottomx - topx))
imgThresh = cv2.warpAffine(imgThresh, rotationMatrix, (bottomy - topy, bottomx - topx))
roi = cv2.resize(roi, (0, 0), fx=3, fy=3)
imgThresh = cv2.resize(imgThresh, (0, 0), fx=3, fy=3)
```

- Create a black mask (np.zeros) that is created with the same size as the image imgGrayscaleplate and data type uint8 (8-bit unsigned integer). Initially, this mask is all black (value 0), then use the drawContours() function in the OpenCV library to draw the contours found in the imgGrayscaleplate image

on the mask mask. The contour point used here is screenCnt. A white image (255) will be drawn on the area inside the contour (-1 is this value).

- The white points in the mask are found and stored in variables x and y(np.where(). (x,y) now contains the coordinates of the white points on the mask.
- Find the coordinates (top-left) of the top left corner of the critical region by taking the smallest value of (x,y) and the coordinates (bottom-right) of the bottom right corner of the critical region with How to get the maximum value of (x,y).
- Crop out a portion of the original img image using the coordinates calculated above. "roi" will contain the important area of the image.
- Similarly, cut out the portion of the binary image "imgThreshplate" that corresponds to the critical region.
- Calculate the center coordinates of the critical region by taking the average of the x and y coordinates of the top and bottom corners calculated above. The center coordinates will be saved in "ptPlateCenter".

```
if x1 < x2:
    rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, -angle, 1.0)
else:
    rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, angle, 1.0)

roi = cv2.warpAffine(roi, rotationMatrix, (bottomy - topy, bottomx - topx))
imgThresh = cv2.warpAffine(imgThresh, rotationMatrix, (bottomy - topy, bottomx - topx))
roi = cv2.resize(roi, (0, 0), fx=3, fy=3)
imgThresh = cv2.resize(imgThresh, (0, 0), fx=3, fy=3)
```

- Dựa vào góc "angle" và vị trí của các điểm (x1, y1) và (x2, y2), đoạn mã tính toán ma trận biến đổi "rotationMatrix" để căn chỉnh vùng biển số theo góc "angle". Ma trận này được sử dụng trong hàm "cv2.warpAffine()" để xoay và căn chỉnh vùng biển số.

- Cuối cùng, ảnh của biển số và ảnh nhị phân của biển số (imgThresh) được thay đổi kích thước lên gấp 3 lần để tạo ra ảnh có độ phân giải cao hơn.

```
kerel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
thre_mor = cv2.morphologyEx(imgThresh, cv2.MORPH_DILATE, kerel3)
cont, hier = cv2.findContours(thre_mor, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(roi, cont, -1, (100, 255, 255), 2)  # Vẽ contour các kí tự trong biển số
```

- Similar to the previous steps, create "kerel3" as a 3x3 kernel, then the binary image "imgThresh" is dilated using the "kerel3" kernel created in the previous step.
- Search for borders in the image and draw contours of the characters on the license plate.

```
    ###################### Filter out characters #################
    char_x_ind = {}
    char_x = []
    height, width, _ = roi.shape
    roiarea = height * width

    for ind, cnt in enumerate(cont):
        (x, y, w, h) = cv2.boundingRect(cont[ind])
        ratiochar = w / h
        char_area = w * h

        if (Min_char * roiarea < char_area < Max_char * roiarea) and (0.25 < ratiochar < 0.7):
            if x in char_x:  # Sử dụng để dù cho trùng x vẫn vẽ được
                x = x + 1
            char_x.append(x)
            char_x_ind[x] = ind
```

⇨ The above code performs a series of steps to filter out characters (license plate characters) from the list of previously found contour lines. The processing and filtering steps are as follows:

- "char_x_ind = {}" and "char_x = []": These are two variables used to store information about filtered characters and the corresponding x coordinates of each character. "char_x" is a list containing the x coordinates of the

characters, and "char_x_ind" is a dictionary used to map the x coordinates to the corresponding contour index in the "cont" list.

- "height, width, _ = roi.shape": Get the dimensions of the image, i.e. height and width.
- "roiarea = height * width": Calculate the area of the whip area.
- Iterate through each contour line in the cont list using the loop "for ind, cnt in enumerate(cont):"
- "(x, y, w, h) = cv2.boundingRect(cont[ind])": Calculate the rectangle surrounding the current contour line to determine its coordinates and size. x is the left x coordinate of the rectangle, y is the top y coordinate of the rectangle, w is the width of the rectangle, and h is the height of the rectangle.
- "ratiochar = w / h": Calculate the ratio between the width (w) and height (h) of the rectangle surrounding the contour. This ratio can be used to determine if the character has the right shape.
- "char_area = w * h": Calculate the area of the rectangle surrounding the contour line.
- "Min_char * roiarea < char_area < Max_char * roiarea: This condition checks whether the area of the rectangle surrounding the contour line is within a given range. "Min_char" = 0.01 and "Max_char" = 0.09 are the thresholds for Allows the area of the character within this range.
- "0.25 < ratiochar < 0.7": This condition checks the ratio between the width and height of the rectangle surrounding the boundary. This eliminates borders that are not proportional to the character.
- If the considered boundary satisfies the above conditions, its x coordinate (x) is added to the list "char_x" and the corresponding contour line is saved into the dictionary "char_x_ind" with x coordinate as the key.

⇨ Finally, after the loop ends, the list "char_x" will contain the x coordinates of the filtered characters, and the dictionary "char_x_ind" will contain information about the contour corresponding to each character based on the coordinates. degree x.

```
        ############ Character recognition ##########################

        char_x = sorted(char_x)
        strFinalString = ""
        first_line = ""
        second_line = ""

        for i in char_x:
            (x, y, w, h) = cv2.boundingRect(cont[char_x_ind[i]])
            cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 2)
            imgROI = thre_mor[y:y + h, x:x + w]  # Crop the characters
            imgROIResized = cv2.resize(imgROI, (RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))  # resize image
            npaROIResized = imgROIResized.reshape(
                (1, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT))

            npaROIResized = np.float32(npaROIResized)
            _, npaResults, neigh_resp, dists = kNearest.findNearest(npaROIResized,k=3)  # call KNN function find_nearest;
            strCurrentChar = str(chr(int(npaResults[0][0])))  # ASCII of characters
            cv2.putText(roi, strCurrentChar, (x, y + 50), cv2.FONT_HERSHEY_DUPLEX, 2, (255, 255, 0), 3)
            if (y < height / 3):  # decide 1 or 2-line license plate
                first_line = first_line + strCurrentChar
            else:
                second_line = second_line + strCurrentChar
        listNp[first_line] = second_line
        roi = cv2.resize(roi, None, fx=0.75, fy=0.75)

for x in listNp:
    first_line = x
    second_line = listNp[x]
    if check_firstline(first_line) and check_secondline(second_line):
        print("\n License Plate is: " + first_line + " - " + second_line + "\n")
        number_plate = first_line + " - " + second_line
        datain(number_plate)
        return
print("Khong tim thay bien so xe\n")
```

- The list char_x contains the x coordinates of the characters sorted in ascending order.
- first_line = "" and second_line = "": Two variables to store characters on two lines of the license plate (if any)

⇨ The “for i in char_x” loop iterates over each character:

- "(x, y, w, h) = cv2.boundingRect(cont[char_x_ind[i]])": Get information about the rectangle surrounding the current character.

- "cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 2)": Draw a rectangle surrounding the character on the whip image.
- "imgROI = thre_mor[y:y + h, x:x + w]": Cut character container from binary image thre_mor.
- "imgROIResized = cv2.resize(imgROI, (RESIZED_IMAGE_WIDTH, RESIZED_IMAGE_HEIGHT))": Resize the character container to standardize the size for recognition.
- "npaROIResized = imgROIResized.reshape((1, RESIZED_IMAGE_WIDTH * RESIZED_IMAGE_HEIGHT))": Transforms a character container into a one-dimensional array.
- "npaROIResized = np.float32(npaROIResized)": Converts character container array to float32 data type.
- "_, npaResults, neigh_resp, dists = kNearest.findNearest(npaROIResized, k=3)": Uses a pre-trained KNN (K-Nearest Neighbors) model to recognize characters in the container. The recognition results (npaResults) contain the ASCII code of the character.
- strCurrentChar = str(chr(int(npaResults[0][0])): Converts the ASCII identifier to characters and saves it to strCurrentChar.
- "cv2.putText(roi, strCurrentChar, (x, y + 50), cv2.FONT_HERSHEY_DUPLEX, 2, (255, 255, 0), 3)": Draws the recognized character on the whip image.
- "if (y < height / 3"): Checks whether this character belongs to the upper or lower line of the license plate. Based on the line position, the character is added to "first_line" or "second_line".
- "listNp[first_line] = second_line": Store the lines of the license plate into a dictionary with the upper line as the key and the lower line as the value.
-

- "roi = cv2.resize(roi, None, fx=0.75, fy=0.75)": Resize the roi image to reduce the original image size.
- Finally, after all the characters have been processed, the loop ends. The code checks whether the license plate has been successfully recognized by calling the two functions check_firstline and check_secondline. If both lines are valid, the license plate recognition result will be printed to the screen and can be saved or further processed. Otherwise, the message "No license plate found" will be printed to the screen.

3.3. Scan the card and take a photo:

- Create two objects "cam = cv2.VideoCapture(0)" which is the camera and "reader" which is the tag to be read.
- Swipe the card and get the key which is the card's id field with the command "key, text = reder.read()", this id is unique. If "key" is equal to the id of the saved cards, take a photo and save it in the "images" folder with the name "numberplate.jpg", then call the function to process the newly captured image.
- On the contrary, if the card is not in the list of identified cards, the red LED will turn on, not confirming the card.

Chapter 3. **Proposed System setting up:**

1. Raspberry Pi 4B:
    - A 1.2GHz 64-bit quad-core ARMv8 CPU
    - 802.11n Wireless LAN
    - Bluetooth 4.1
    - Bluetooth Low Energy (BLE)
    - 4GB RAM
    - 4 USB ports

- 40 GPIO pins
- 2 Micro HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core

2. PiCamera:
    - Capture still images at 2592 x 1944 resolution
    - Lens: 1/4 5M
    - Aperture: 2.9
    - Focal length: 3.29
    - Viewing angle: 72.4 degrees
3. RFID: Use RFID RC522, it comes with an RFID tag and key fob that includes 1KB of memory.
4. Led RGB:
    - Operating voltage: 5VDC
    - RGB LED 10mm round opaque
    - Led common cathode pin (GND).
    - Dimensions: 25 x 12mm.

Chapter 4: Experimental results

| **RFID Input** | **Actual number plate** | **Led** | **Number_plate** | **Status** | **Date_in** | **Date_out** | **RFID** |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 157502735209 | 35B2-54953 | Green | 35B2 - 54953 | 1 | 2023/11/19 11:43:29 | | 157502735209 |
| 40551782013 | 35B2-54953 | RED | 35B2 - 54953 | 1 | 2023/11/19 11:43:29 | | 157502735209 |
| 157502735209 | 35B2-54953 | Green | 35B2 - 54953 | 0 | 2023/11/19 11:43:29 | 2023/11/19 11:47:02 | 157502735209 |
| 331267483273 (Invalid card) | 35B2-54953 | RED | | | | | |
| 40551782013 | 63B9 - 99999 | Green | 63B9 - 99999 | 1 | 2023/11/19 11:49:17 | | 40551782013 |
| 40551782013 | 63B9 - 99999 | Green | 63B9 - 99999 | 0 | 2023/11/19 11:49:17 | | 40551782013 |
| 40551782013 | 63B9 - 99999 | Green | 63B9 - 99999 | 0 | 2023/11/19 11:49:17 | 2023/11/19 11:49:17 | 40551782013 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 157502735209 | 74L1 - 27762 | Green | 74L1 - 27762 | 1 | 2023/11/19 11:53:02 | | 157502735209 |
| 157502735209 | 74L1 - 27762 | Green | 74L1 - 27762 | 0 | 2023/11/19 11:53:02 | 2023/11/19 11:53:04 | 157502735209 |
| 157502735209 | No license plate | Red | | | | | |

Chapter 5: Evoluation and development

1. Evoluation:
   - Reading license plates quite accurately when the camera is placed at the right angle is a positive point.
   - Database can store large data which is a strong point.
   - No user interface yet
   - LED lights can be replaced with lever systems to help ensure more security
2. Development:
   - Upgrade the camera if necessary, integrating advanced techniques such as noise image processing and improving license plate recognition.
   - Add a camera to record the user's face to avoid losing the car
   - Test and improve database structure, optimize queries, and integrate backup and restore mechanisms.
   - Build a friendly, easy-to-use user interface to monitor system status and manage user information, develop an interactive graphical interface, add features such as viewing access history and booking Alerts via user interface.

Build a price list and calculate money according to delivery time and available price list.

- Replace LEDs with a lever system to provide visual notifications with greater flexibility.
- Enhance security to prevent information security risks.