UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE
Školska 2023. / 2024. godina

Satelitske Telekomunikacije

SmartGate License Plate Recognition (**SGLPR**)

Mentor:

Dr. sc. Alma Šećerbegović

Studenti:

Amar Mehmedović
Mahir Terzić
Damir Muminović
Belma Hadžiefendić

Tuzla, juni 2024. godine

CILJ:

Cilj projekta je izgraditi sistem za automatsko prepoznavanje registarskih tablica (ANPR - Automatic Number Plate Recognition) i kontrolu pristupa vozila.

Detaljno, projekat ima sledeće ciljeve:

Detekcija vozila i mjerenje udaljenosti:

Koristeći ultrazvučni senzor, sistem kontinuirano mjeri udaljenost od vozila.

Kada vozilo dođe na određenu udaljenost (10-40 cm), sistem započinje proces detekcije registarske tablice.

Prepoznavanje registarske tablice:

Slika vozila se snima pomoću Pi kamere.

Slika se preprocesira primjenom različitih filtera i algoritama za detekciju ivica.

Identifikuju se konture na slici kako bi se pronašli potencijalni regioni koji sadrže registarske tablice.

Iz tih regiona se izvlači tekst pomoću OCR tehnologije (EasyOCR biblioteka).

Provjera u bazi podataka:

Ekstraktovani tekst (registarski broj) se upoređuje sa podacima u bazi podataka.

Ako je registarski broj pronađen u bazi, sistem prikazuje poruku "OTVORI RAMPU" na LCD ekranu i omogućava pristup.

Ako registarski broj nije pronađen, sistem prikazuje poruku "ZABRANJEN PROLAZ".

Prikaz informacija:

Koristeći LCD ekran, sistem prikazuje odgovarajuće poruke u vezi sa statusom pristupa vozilu.

Ako se prepozna tablica, na ekranu se prikazuje njen broj i odgovarajuća poruka (dozvoljen ili zabranjen pristup).

Kontrola i čišćenje resursa:

Sistem koristi try-except blok za upravljanje glavnim procesom, omogućavajući ručno zaustavljanje sistema pomoću tastature (KeyboardInterrupt).

Po završetku rada, sistem čisti GPIO pinove da bi se osiguralo pravilno zatvaranje resursa.

Detaljan Opis Svake Funkcije:

1. measure_distance: Mjerenje udaljenosti vozila pomoću ultrazvučnog senzora.
2. cleanup: Čišćenje GPIO pinova nakon završetka rada.
3. imageProcessing: Preprocesiranje slike za poboljšanje detekcije ivica.
4. detect_license_plate: Detekcija registarske tablice, OCR prepoznavanje, provjera u bazi podataka i prikaz informacija na LCD ekranu.
5. lcd_init, lcd_string, lcd_clear: Funkcije za kontrolu LCD ekrana.
6. databaseConnect: Povezivanje na MySQL bazu podataka.

Glavni Tok Programa:

Inicijalizacija LCD ekrana.

Kontinuirano mjerenje udaljenosti vozila.

Kada se detektuje vozilo na odgovarajućoj udaljenosti, pokreće se proces detekcije registarske tablice.

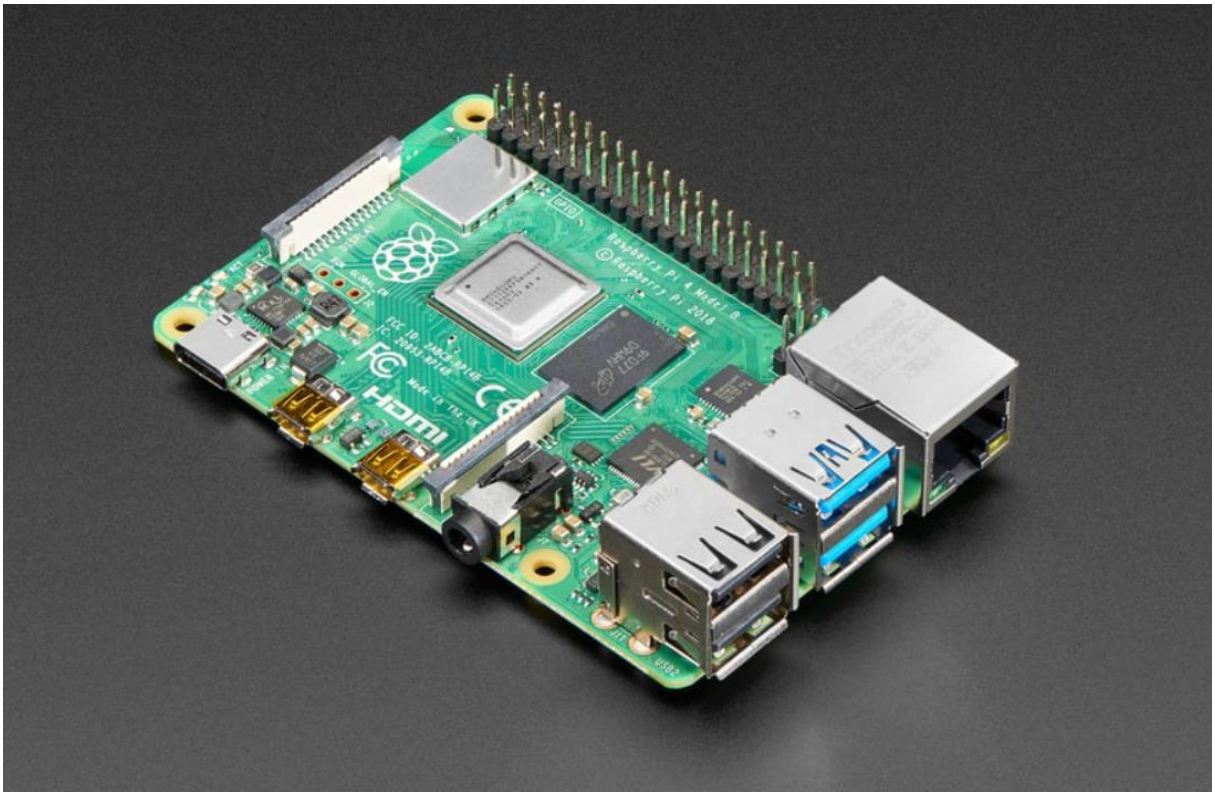
Nakon detekcije, rezultat se prikazuje na LCD ekranu i provjerava se pristupna dozvola u bazi podataka.

Program se ponovo vraća u režim mjerenja udaljenosti dok se ne detektuje novo vozilo.

Ovaj sistem je koristan za automatske sisteme kontrole pristupa u garažama, sigurnosnim kontrolama ili bilo kom drugom scenariju gdje je potrebno automatizovati prepoznavanje vozila i kontrolu pristupa.

OPREMA:

Za kreiranje ovog projekta korištena je sljedeća oprema:



Raspberry PI 4

HC-SR04 ultrazvučni senzor

1602 LCD Display

MAIN.py

```python
import cv2
import numpy as np
import easyocr
import imutils
import mysql.connector
import time
from picamera2 import Picamera2, Preview
from libcamera import Transform
import re
from displej import lcd_init, lcd_string, LCD_LINE_1, LCD_LINE_2 ,lcd_clear
from senzor import measure_distance, cleanup

picam2 = Picamera2()
preview_config = picam2.create_preview_configuration(transform=Transform(vflip=True, hflip=True))
capture_config = picam2.create_still_configuration(main={"size": (640, 480)}, transform=Transform(vflip=True, hflip=True))
reader = easyocr.Reader(['en'])

def piCameraConf():
    picam2.configure(preview_config)
    picam2.start()
    time.sleep(5)

def databaseConnect():
    mydb = mysql.connector.connect(
        host="localhost",
        user="rampa",
        passwd="1234"
    )
    mycursor = mydb.cursor()
    mycursor.execute("USE db")
    return mycursor, mydb
```

```python
def imageProcessing(image):
    blurred = cv2.bilateralFilter(image, 13, 18, 18)
    edges = cv2.Canny(blurred, 30, 200)
    return edges


def detect_license_plate():
    mycursor, mydb = databaseConnect()
    cropped_images = []
    img = cv2.imread("passat.jpg")
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    edges = imageProcessing(img)
    #cv2.imshow('edge', edges)
    #cv2.waitKey(0)

    contours = cv2.findContours(edges.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

```

```
    for contour in contours:
        peri = cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, 0.018 * peri, True)
        if 4 <= len(approx) <= 9:
            cv2.drawContours(img, [approx], -1, (0, 255, 0), 2)
            x, y, w, h = cv2.boundingRect(approx)
            cropped_image = gray[y:y+h, x:x+w]
            cropped_image = cv2.bilateralFilter(cropped_image, 13, 18, 18)
            result = reader.readtext(cropped_image)
            extracted_text = ' '.join([text[1] for text in result]).upper()
            extracted_text = re.sub("[^A-Za-z0-9]", "", extracted_text)
            cv2.imshow('edge', cropped_image)
            cv2.waitKey(0)
            print(extracted_text)

            lcd_clear()  # Izbrisi sadrzaj displeja prije postavljanja novih poruka

            lcd_string("Reg broj:"+extracted_text, LCD_LINE_1)

            if extracted_text and len(extracted_text) in [7, 8]:
                cropped_images.append(cropped_image)

                # Provjeri da li je tablica u bazi
                mycursor.execute("SELECT reg_broj FROM uposlenici")
                resultBP = mycursor.fetchall()
                access_granted = False
                for row in resultBP:
                    if row == (extracted_text,):
                        print("OTVORI RAMPU")
                        lcd_string("OTVORI RAMPU", LCD_LINE_2)
                        access_granted = True
                        break
```

```
                if not access_granted:
                    lcd_string("ZABRANJEN PROLAZ", LCD_LINE_2)

                mycursor.close()
                mydb.close()
                time.sleep(3)
                lcd_clear()  # Izbrisi sadrzaj displeja prije nastavka

                return  # Izlaz iz funkcije ako je tablica registrovana

    # Ako nema detektovane tablice
    lcd_clear()  # Izbrisi sadrzaj displeja ako nema tablice
    time.sleep(5)
    mycursor.close()
    mydb.close()
```

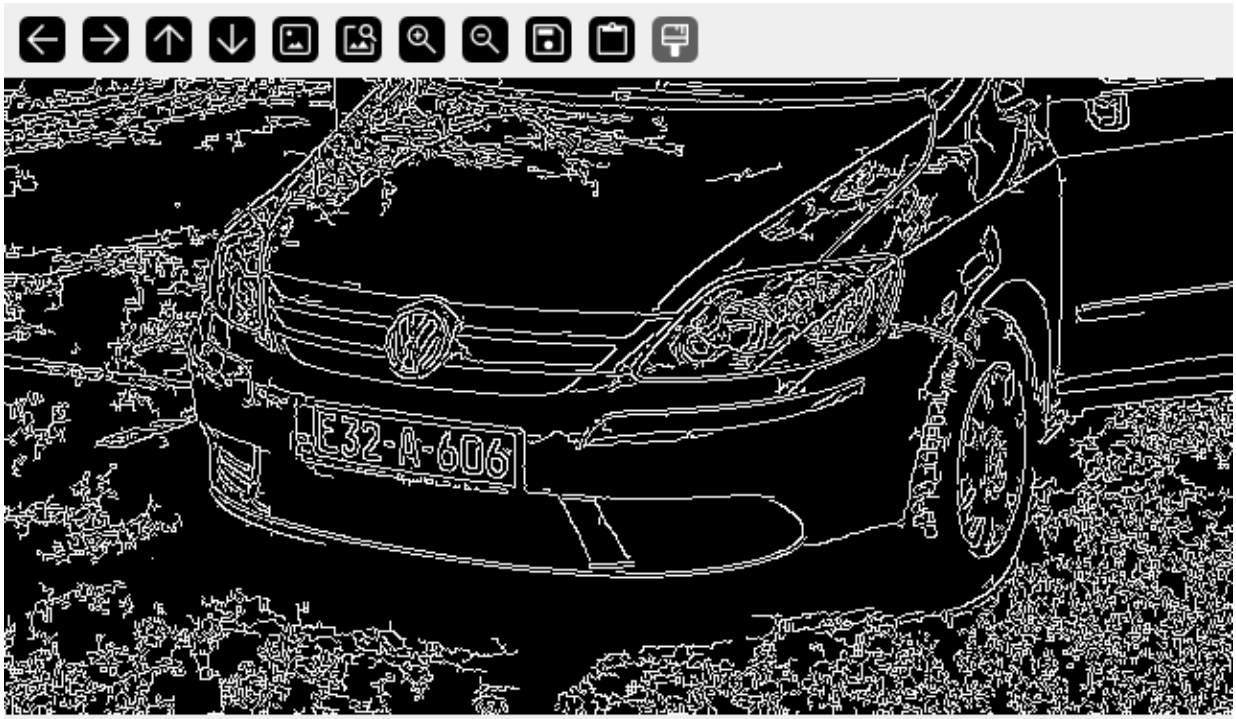
```
104  lcd_init()
105
106  try:
107      while True:
108          consistent_distance_count = 0
109          while consistent_distance_count < 10:  # 3 sekunde * 10 mjerenja u sekundi
110              distance = measure_distance()
111              print(f"Provjeravanje udaljenosti: {distance:.1f} cm")
112              if 10 <= distance <= 40:
113                  consistent_distance_count += 1
114              else:
115                  consistent_distance_count = 0
116              time.sleep(0.5)
117          print("Detekcija tablice počinje...")
118          detect_license_plate()
119          time.sleep(3)  # Pauza prije nego što se ponovo provjeri udaljenost
120  except KeyboardInterrupt:
121      pass
122  finally:
123      cleanup()
124
125
```

FUNKCIONALNOST U main.py

lcd_init se koristi za inicijalizaciju samog LCD displeja. Vrši se try blok odnosno pokušava se navedeni blok koda sve dok se ne napravi prekid. S obzirom da treba biti siguran da će, kada dođe do rampe, uređaj započeti proces detektovanja tablica ali da uređaj ne vrši detektovanje prolaznika, životinja ili slično. Zato postoji varijabla **consistent_distance_count** koja je incijalizirana na 0 i tek kada ona ima vrijednost 10 počinje proces mjerenja distance. Ona će imati vrijednost 10 samo kada se 10x zaredom izmjeri da je distanca objekta koji se nalazi kod senzora u određenom intervalu udaljenosti.

Interval koji smo postavili je između 10 i 40cm što znači da uređaj mora 10x izmjeriti udaljenost objekta od 10-40cm i ako jednom izmjeri da je na primjer objekat na udaljenosti od 50cm, vraća varijablu na 0. Ovako se zapravo osigurava da je auto u mirovanju kada se započne proces detektovanja tablica.

Između svakog mjerenja postoji pauza od 0.5 sekundi što znači da mora biti minimalno 5 sekundi(10 mjerenja po 0.5s) da bi započeo proces detekcije od strane kamere. U funkciji **detect_license_plate** jeste ispisivanje registracijskog broja na displej kada se on detektuje pomoću **lcd_string** funkcije, te također vrši se provjera baze podataka gdje ukoliko se broj tablica koji je detektovan poklapa sa nekim brojem tablica unesenim u bazu podataka, LCD displej ispisuje poruku “OTVORI RAMPU” što znači da je detekcija uspješna . U suprotnom ako je **access_granted**(varijabla koja ima rezultat upoređivanja) vrijednosti false to znači da je prolaz zabranjen pa se ispisuje ta poruka na displej. Nakon pauze od 3 sekunde briše se poruka sa ekrana te se zatvara baza podataka što bi značilo da je vozilo nastavilo dalje.



(x=542, y=315) ~ L:0

(x=464, y=153) ~ L:0

```
MariaDB [db]>
MariaDB [db]> show tables;
+--------------+
| Tables_in_db |
+--------------+
| uposlenici   |
+--------------+
1 row in set (0.001 sec)

MariaDB [db]> select * from uposlenici;
+---------+------------+----------+
| ime     | prezime    | reg_broj |
+---------+------------+----------+
| Asmir   | Gogic      | M18K001  |
| Nermin  | Suljanovic | A00J210  |
| Amar    | Mehmedovic | A51K868  |
| Mahir   | Terzic     | E32A606  |
| Tesa    | Tesanovic  | E32K332  |
+---------+------------+----------+
5 rows in set (0.001 sec)

MariaDB [db]> exit
Bye
amar@raspberrypi:~ $ scrot -s
```

Primjena bilateralnog filtera:
Smanjuje šum na slici uz očuvanje oštrih ivica.
Detekcija ivica:
Primjenjuje Canny algoritam za detekciju ivica na zamućenoj slici.
Vraćanje rezultata:
Vraća sliku sa detektovanim ivicama.

Originalna slika:
Ulazna slika koju treba obraditi.
Zamućena slika:
Slika nakon primjene bilateralnog filtera, gdje su šumovi smanjeni, ali su ivice očuvane.
Detektovane ivice:
Slika koja prikazuje samo detektovane ivice, dobijena primjenom Canny algoritma.
Funkcija je korisna za pripremu slike za dalje procese kao što je detekcija registarskih tablica ili drugi oblici prepoznavanja objekata.

DISPLAY I2C 1602

displej.py

```python
import smbus2
import time

# I2C adresa
I2C_ADDR = 0x3f
# I2C bus
I2C_BUS = 1

# LCD konstante
LCD_CHR = 1  # Karakter mod
LCD_CMD = 0  # Komanda mod

LCD_LINE_1 = 0x80  # 1. linija
LCD_LINE_2 = 0xC0  # 2. linija

LCD_BACKLIGHT = 0x08  # Ukljuci pozadinsko osvjetljenje
ENABLE = 0b00000100  # Omoguci pin

# Postavljanje I2C komunikacije
bus = smbus2.SMBus(I2C_BUS)

def lcd_byte(bits, mode):
    """Posalji byte podataka na LCD."""
    bits_high = mode | (bits & 0xF0) | LCD_BACKLIGHT
    bits_low = mode | ((bits << 4) & 0xF0) | LCD_BACKLIGHT

    bus.write_byte(I2C_ADDR, bits_high)
    lcd_toggle_enable(bits_high)

    bus.write_byte(I2C_ADDR, bits_low)
    lcd_toggle_enable(bits_low)
```

```
33  def lcd_toggle_enable(bits):
34      """Toggle enable pin"""
35      time.sleep(0.0005)
36      bus.write_byte(I2C_ADDR, (bits | ENABLE))
37      time.sleep(0.0005)
38      bus.write_byte(I2C_ADDR, (bits & ~ENABLE))
39      time.sleep(0.0005)
40
41  def lcd_init():
42      """Inicijalizacija LCD ekrana."""
43      lcd_byte(0x33, LCD_CMD)
44      lcd_byte(0x32, LCD_CMD)
45      lcd_byte(0x06, LCD_CMD)
46      lcd_byte(0x0C, LCD_CMD)
47      lcd_byte(0x28, LCD_CMD)
48      lcd_byte(0x01, LCD_CMD)
49      time.sleep(0.0005)
50
51  def lcd_string(message, line):
52      """Posalji string na LCD ekran."""
53      message = message.ljust(16, " ")
54      lcd_byte(line, LCD_CMD)
55      for i in range(16):
56          lcd_byte(ord(message[i]), LCD_CHR)
57
58  def lcd_clear():
59      """Obri?i sadr?aj LCD ekrana."""
60      lcd_byte(0x01, LCD_CMD)
61      time.sleep(0.0005)
```

Prvo je potrebno da nađemo I2C adresu preko terminala sa komandom sudo i2cdetect -y 1. I2C bus je 1 i to je uobičajeno za raspberry PI 4. Inicijalizujemo dvije komande LCD_CHR i CMD sa 1 i 0, komanda CHR je mod za slanje karaktera a CMD je mod za slanje komandi. Zatim inicijaliziramo linije 1 i 2 displeja sa adresama koje ce označavati njihov početak. Pomoću LCD_BACKLIGHT uključujemo pozadinsko osvjetljenje a sa ENABLE omogućavamo pin koji će predstavljati pin za komunikaciju sa LCD-om. Bus kreira SMBus objekat za komunikaciju preko I2C bus-a.
Funkcija lcd_byte šalje byte podataka na LCD ekran i uzima dva parametra a to su bits koji se šalju i mode u kojem modu se podaci šalju(CMD ili CHR).U toj funkciji se vrši slanje 4 visoka bita na LCD i 4 niska bita na LCD te se ponovo aktivira enable pin. Sa lcd_toggle_enable se aktivira enable pin kako bi LCD mogao prepoznavati podatke.
lcd_init vrši inicijalizaciju LCD ekrana gdje se prvo inicijalizira u 8-bitnom modu zatim prelazi u 4-bitni mod. Podešava se kursor da prelazi na desno, uključuje se ekran, isključuje kursor, postavljaju se 2 linije te briše se ekran. lcd_string uzima poruku i liniju gdje se linija podešava na 16 karaktera (dodavanjem praznih mjesta ako je kraća poruka). Prosljeđuje se adresa početka linije te se šalje karakter jedan po jedan preko lcd_byte. lcd_clear briše sadržaj LCD ekrana.

senzor.py

```python
import RPi.GPIO as GPIO
import time

# Definisanje GPIO pinova
GPIO_TRIGGER = 23
GPIO_ECHO = 24

# Postavljanje moda za GPIO pinove
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def measure_distance():
    # Slanje ultrazvučnog impulsa
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    start_time = time.time()
    stop_time = time.time()

    while GPIO.input(GPIO_ECHO) == 0:
        start_time = time.time()

    while GPIO.input(GPIO_ECHO) == 1:
        stop_time = time.time()

    # Izračunavanje trajanja pulsa i udaljenosti
    time_elapsed = stop_time - start_time
    distance = (time_elapsed * 34300) / 2

    return distance

def cleanup():
    GPIO.cleanup()

```

Prvo postavljamo pin broj 23 kao trigger pin koji se koristi za slanje ultrazvučnog impulsa a pin broj 24 kao echo odnosno koristi se za prijem odbijenog ultrazvučnog impulsa. GPIO.setmode(GPIO.BCM) se koristi kao način numeracije pinova gdje se može npr. navesti GPIO_TRIGGER kao naziv pina i postaviti za OUT pin. Funkcija measure_distance je ključna funkcija jer vrši mjerenje udaljenosti.Postavlja GPIO_TRIGGER na visoki nivo(TRUE) kako bi poslao ultrazvučni impuls.Nakon kratke pause postavlja GPIO_TRIGGER na niski nivo(FALSE) kako bi završio impuls.Pomoću start_time i stop_time zabilježi vrijeme za početak i kraj impulse.

Zatim se čeka da GPIO_ECHO postane visok odnosno dok ne primi odraženi signal te se zabilježi trenutno vrijeme kao početak odziva.Čim GPIO_ECHO postane nizak to znači da se signal završio te se zabilježi trenutno vrijeme kao kraj odziva.

Kod proračuna udaljenosti, time_elapsed se određuje kao stop_time – start_time odnosno to je vrijeme koliko je trajao impuls. Distanca se određuje kao (time_elapsed * 34300)/2 gdje je 34300cm/s brzina zvuka te /2 je jer se ide do objekta i nazad.Vrijednost koja se vraća je u centimetrima jer smo i u proračunu koristili centimetre/s . Cleanup funkcija se koristi za vraćanje GPIO pinova koji su korišteni u početno stanje.

Raspodjela zadataka prilikom izrade projekta:

Belma Hadžiefendić → Konfiguracija kamere, prikupljanje podataka, izrada PDF izvještaja i PPT prezentacije

Damir Muminović →Implementacija baze podataka u projekat, pristup podacima i upravljanje istih

Amar Mehmedović → Implementacija algoritma za detekciju registarskih tablica, preprocesiranje slike, identifikovanje kontura, ekstraktovanje teksta

Mahir Terzić → Implementacija senzora i displeja u projekat, detekcija vozila, određivanje udaljenosti, poređenje dobijenih rezultata, ispisivanje poruke o uspješnosti na ekran