SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Upravljanje rasvjetom uz pomoć rukovnih gesti i prepoznavanje lica

Meko računarstvo

Projektni zadatak

Ivan Gudelj

Diplomski studij računarstva, DRB

Osijek, 2022

SADRŽAJ

1. UVOD

Cilj projektnog zadatka jest primjeniti naučene vještine iz područja strojnog učenja i umjetne inteligencije kao i optimiranja i izdvajanja značajki u raspoznavanju uzoraka (prepoznavanje značajki lica i gesti ruku).

Tehnologija prepoznavanja osoba i njihovih pokreta se razvila već početkom ovog stoljeća u vidu korištenja rukavica sa senzorima koji su pružali informaciju o položaju i pozicij ruke. Stalnim razvojem i unaprijeđenjem tehnologije, danas je moguće prepoznati poziciju ruku koristeći svega nekoliko biblioteka u Pythonu, što je nadalje otvorilo puno mogućnosti u vidu “olakšanog” izdavanja komandi za neku određenu operaciju, analizu pozicije osobe itd.

Izrada projektnog zadatka se odvijala u 4 glavna koraka; prikupljanje podataka (slike osobe kojoj je omogućena kontrola), analiza lica osobe (uspoređivanje značajki lica sa slike u stvarnom vremenu i značajki lica koje su unaprijed obrađene), analiza položaja i pozicije ruke te najbitnij korak - slanje željene komande (GET zahtjeva) na uređaj .

2. OPIS ZADATKA I KORIŠTENA TEHNOLOGIJA

Cilj zadatka jest; ovlaštenom korisniku (korisnik koji je u bazi ovlaštenih osoba) omogućiti upravljanje NeoPixel 12B segmentima, postavljajući ruku u unaprijed definirane položaje.

Određivanje ovlaštenih korisnika odvija se dodavanjem fotografije korisnika u određenu mapu u projektu. Sljedeći korak jest realizacija prepoznavanja lica s videa u stvarnom vremenu. Prepoznavanje lica se odvija uz pomoć Python biblioteke *face_recognition* [1], za koju se smatra da je jedna od najjednostavnijih biblioteka za prepoznavanje lica. Stvorena je koristeći već postojeću biblioteku *dlib* koja ima točnost prepoznavanja lica do 99% prema mjerilu *Labeled Faces in the Wild* standarda [2].

Dlib

Dlib je biblioteka napisana u C++ programskom jeziku te od 2016. godine sadrži programsku podršku za rješavanje zadataka iz područja; mreža, rada s nitima, struktura podataka, linearne algebre, strojnog učenja, obrade slika te obrade velikih količina podataka. Pruža dvije funkcije za prepoznavanje lica. Prva funkcija je *HOG + Linear SVM face detector*, dok je druga opcija korištenje MMOD CNN uz duboko učenje. Za realizaciju ovog projekta korištena je opcija HOG + Linear SVM face detector, budući da koristi GPU-ubrzan *face detector* što rezultira boljem vremenu obrade..

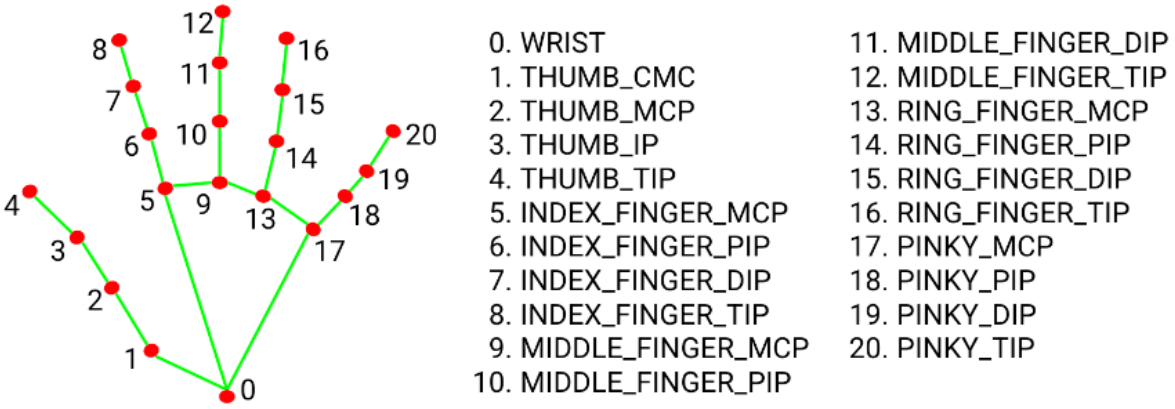
HoG (Histogram of Oriented Gradients) je deskriptor značajki što znači da služi za pojednostavljeni prikaz slike ili djela slike izvlačeći korisne informacije i odbacujući manje korisne. Uobičajeno deskriptori pretvaraju sliku veličine Visina x Širina x 3 (kanali) u vektor značajki odnosno polje veličine n. Dobijeni vektor je moguće koristiti za klasifikacijske algoritme poput SVM (eng. *Support Vector Machine*) te će dati dobre rezultate. Kod HoG deskriptora, distribucija smjerova gradijenata, odnosno distribucija promjene svjetline se koriste kao značajke. Gradijenti fotografije mogu biti veoma korisni zbog toga što je veličina gradijenata velika oko rubova i uglova, što i daje najviše informacija o fotografiji (više informacija o oblicima objekata nego o ravninama).

MediaPipe

Nadalje prepoznavanje položaja i gesti ruku je izvedeno koristeći postojeću *MediaPipe* biblioteku [3] koja je rješenje za praćenje ruku i prstiju visoke pouzdanosti. Koristi strojno učenje (postoje unaprijed definirani modeli koji su i korišteni) kako bi locirao položaj ruke iz 21 3D obilježja ruke iz samo jedne slike (*frame-a*).

MediaPipe Hands koristi višestruke modele za obradu podataka; Model za detekciju dlana koji se izvodi nad cijelom fotografijom i kao rezultat daje orijentiranu opisnu kutiju (*bounding box*) dlana, te model koji obrađuje značajke dlana (izvodi se nad isječenom fotografijom koja je definirana prethodnim modelom) i kao rezultat daje 3D ključne točke (*keypoints*). Položaji ruku korišteni u izradi su vidljivi na Slici 2.1., kao i ključne točke dlana, zajedno s odgovarajućim oznakama za pojedinu točku, na Slici 2.2.

Slika 2.1. Položaji ruku korišteni u izradi



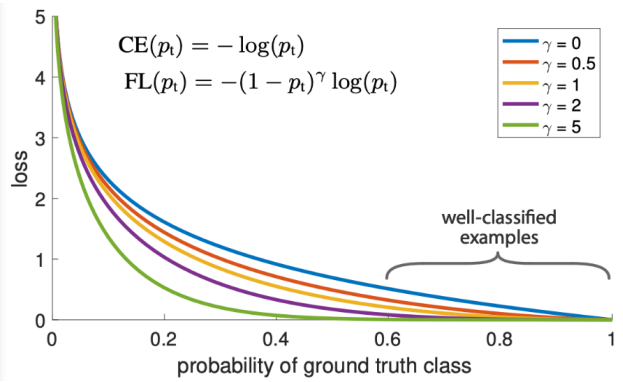
Slika 2.2. Ključne točke dlana

Značajke dlana koje su pronađene na jednoj slici stvarnog videa se uspoređuju s položajima značajki s već postojećim modelom koji je treniran koristeći oko 30 000 fotografija s već spomenutim značajkama. Sam dizajn modela odnosno raspored slojeva je moguće vidjeti naredbom *model.summary(),* nakon što se obavi učitavanje postojećeg MediaPipe modela koristeći *tensorflow* [4] biblioteku (Slika 2.3).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 42)                0
_________________________________________________________________
dense_11 (Dense)             (None, 64)                2752
_________________________________________________________________
dense_12 (Dense)             (None, 128)               8320
_________________________________________________________________
dense_13 (Dense)             (None, 512)               66048
_________________________________________________________________
dense_14 (Dense)             (None, 64)                32832
_________________________________________________________________
dense_15 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_16 (Dense)             (None, 10)                330
=================================================================
Total params: 112,362
Trainable params: 112,362
Non-trainable params: 0
```

Slika 2.3 Raspored slojeva postojećeg MediaPipe modela

Loss funkcija (matematičke jednadžbe koje služe za optimiranje modela neuronske mreže računanjem odstupanja između stvarne i predviđene vrijednosti) našeg modela je *Focal Loss* [5] (nadalje FL). FL funkcija koja se bavi rješavanjem neravnoteže klasa tijekom treninga u zadacima kao što je prepoznavanje objekata. FL primjenjuje modulirajući izraz na gubitke unakrsne entropije (*Cross - Entropy Loss*) kako bi se model fokusirao na učenje ekstremno loše klasificirane primjere. To je dinamički skaliran unakrsni gubitak entropije, gdje faktor skaliranja opada prema nuli kako se povjerenju u ispravnu klasu povećava. Također pomenuti faktor skaliranja može automatski umanjiti težinu doprinosa lakih primjera tijekom treninga kako bi se posvetio težima. Cross - Entropy Loss prestavlja prosjek unakrsnog entropijskog klasifikacijskog gubitka za svaki piksel na slici.



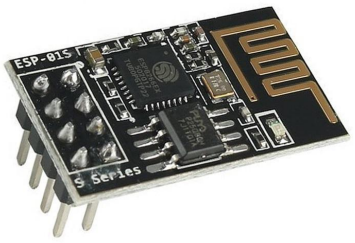
Za izradu projektnog zadatka korištene su sljedeće komponente:

- 2x NeoPixel Ring 12B
- 2x ESP-01S
- 2x ESP-01S RGB LED Controller Module WS2812
- BreadBoard i ožičenje
- 5V eksterno napajanje
- Pogramska podrška

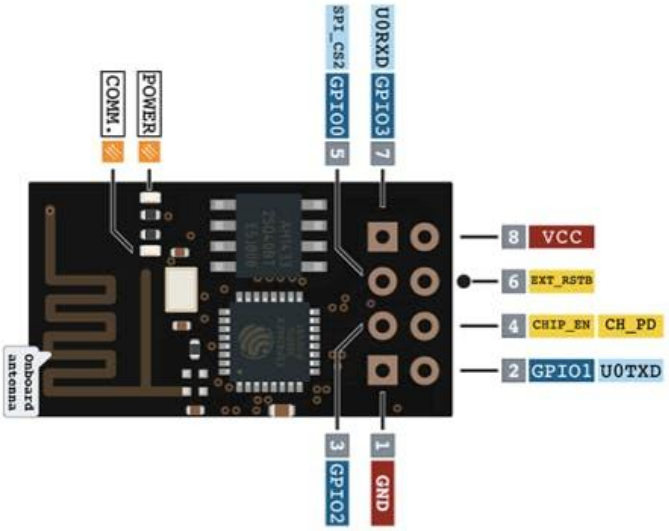
ESP - 01S i RGB LED Controller Module WS2812

ESP - 01S je SOC (eng. System On Chip) WiFi modul s integriranim TCP/IP protokolom koji može svakom mikrokontroleru omogućiti pristup vašoj bežičnoj mreži. Svaki ESP - 01S modul dolazi unaprijed programiran firmware-om koji ima omogućene AT kontrole (eng. *Attention*) što znači da se modul omogućava vrlo jednostavnu zaštitu (WiFi Shield). Ekstremno je učinkovita i isplativ sustav sa sve većim brojem korisnika.

Modul ESP - 01S ima dobre mogućnosti obrade i pohrane podataka što mu omogućava da ga se integira zajedno s senzorima i ostalim ASD (eng. *Application Specific Devices*) kroz GPIO uz minimalnmo učitavanje u stvarnom vremenu. Dizajn ESP - 01S modula je moguće vidjeti na Sl. 2.4 kao i *Pinout* na Sl.2.5.

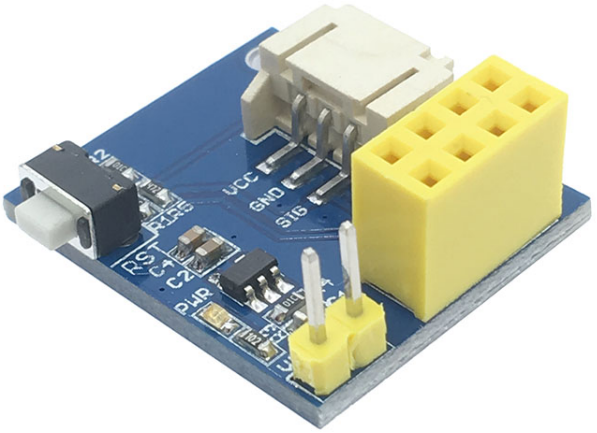


Slika 2.4. Dizajn ESP - 01S modula



Slika 2.5. *Pinout* ESP - 01S modula

WS2812 RGB LED modul služi za ostvarivanje komunikacije između ESP - 01 / 01S sa WS2812 (model mikroprocesora za upravljanje LED modulom), svjetlosnim elementima (LED prsten ili LED traka). Za uspješnu komunikaciju potrebno je spojiti ESP - 01S putem 8 pinova. Dizajn WS2812 modula je moguće vidjeti na Sl 2.6.

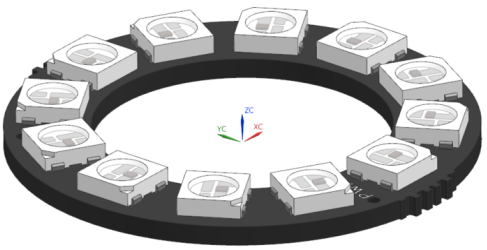


Slika 2.6. Dizajn ESP -01S WS2812 modula

NeoPixel 12B LED prsten

LED prsten koji se sastoji od 12 LED-ica u prstenastom rasporedu dijametra 44.5 mm. Velika prednost uređaja proizvedenih od strane NeoPixel tvrtke je mogućnost ulančavanja svjetlosnih elemenata, odnosno spajanje izlaza jednog modula na ulaz drugog (sinkroniziran rad više prstenova). Svaka LED je programabilna kao zasebni element zahvaljujući driveru koji je integriran u sklopovlje prstena. Napaja se pomoću 5V te putuju kroz ulaz "*Data*" uz veoma precizan *timing* protokol. Budući je protokol osjetljiv na male pomake u vremenu, zahtjeva podatke putem mikrokontolera kao što je Arduino, ESP, ili STM. Dizajn 12B LED prstena se nalazi na slici 2.7 kao i 3D model (Sl 2.8.)

Slika 2.7. Dizajn NeoPixel 12B LED prstena



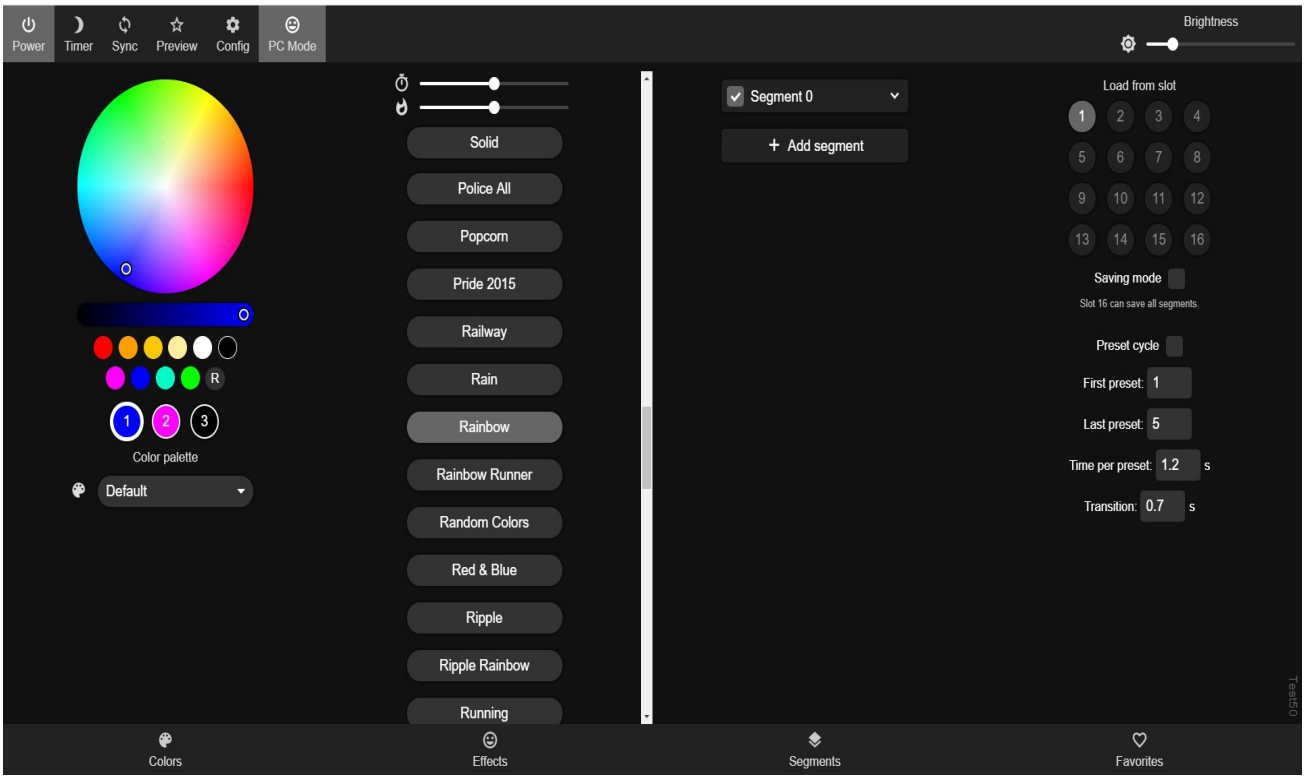
Slika 2.8. 3D model NeoPixel 12B LED prstena

Programska podrška

WLED Firmware

WLED [6] predstavlja brzu i bogatu implementaciju mikrokontrolera kao webservera za kontrolu NeoPixel proizvoda (WS2812B, WS2811, SK6812) ili nekih drugih SPI kontrolera (eng. *Serial Peripheral Interface*). Omogućava preko 100 specijalnih efekata, *user-friendly* API i GUI ukoliko mu se pristupa putem preglednika. Također omogućava ulančavanje NeoPixel proizvoda ukoliko se nalaze u istoj mreži.

Za potrebe projekta, WLED je implementiran na dva ESP - 01S mikrokontrolera s pripadajućim driverom za upravljanje NeoPixel proizvoda. Nakon instalacije WLED-a, pripadajućim ESP - 01S mikrokontrolerima se pristupa putem preglednika ili koristeći jedan od postojećih API (JSON ili HTTP). Korisničko sučelje (verzija na računalu) izgleda kao na Slici 2.9. Komunikacija s ESP - 01S elementima je ostvarena koristeći se HTTP protokolom. (Potrebno je nadodati željenu komandu na kraj "[ipadress]/win&____"). Neke od komandi se nalaze u u tablici 2.1.



Slika 2.9. Korisničko sučelje - pristup WLED-u preko preglednika

3. RJEŠENJE ZADATKA

Prije svega je potrebno definirati korištene biblioteke (Slika 3.1). Za obradu slika u realnom vremenu su korištene *OpenCV* te *NumPy* biblioteke, nadalje za analizu položaja i detekciju gesti ruku kao i učitavanje postojećeg modela su korištene biblioteke *Tensorflow* te *MediaPipe.* Za komunikaciju sa ESP - 01S mikrokontrolerima su korištene biblioteke *Requests* i *Time* (pauziranje između poslanih komandi). Biblioteke *Os i Glob* su korištene za pristup datotekama projekta u kojima se nalaze; model za prepoznavanje gesti, popis gesti te fotografije ovlaštenih osoba.

```
# Import necessary packages
import cv2
import numpy as np
import mediapipe as mp
from tensorflow.keras.models import load_model
import requests
import time
import face_recognition
import os
import glob
```

Slika 3.1 Biblioteke korištene u realizaciji rješenja

Nadalje je potrebno inicijalizirati *MediaPipe* model te postaviti željene parametre kao što je broj ruku za prepoznati (max. 2). Učitavanje već postojećeg modela se vrši uz pomoć *Tensorflow* metode *load_model()*. (Slika 3.2).

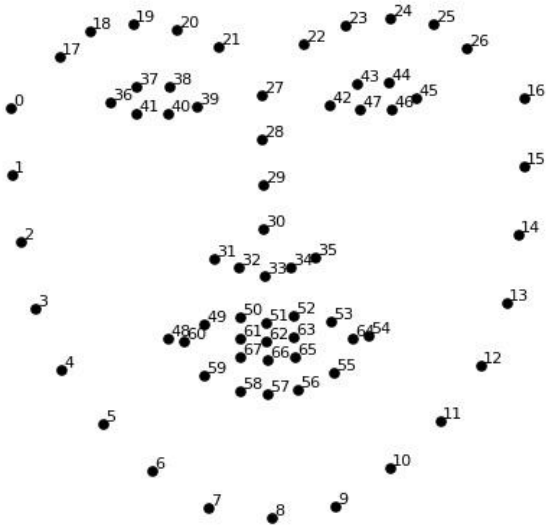
```
# Initialize mediapipe and control variables
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.55)
mpDraw = mp.solutions.drawing_utils
mpDrawingStyles = mp.solutions.drawing_styles

# Load the gesture recognizer model
model = load_model('mp_hand_gesture')
model.summary()

# Load class names
f = open('gesture.names', 'r')
classNames = f.read().split('\n')
f.close()
print(classNames)
```

Slika 3.2. Programska podrška za učitavanje modela za prepoznavanje gesti

Koristeći također postojeću biblioteku *face_recognition* potreno je instancirati varijablu u koju će biti spremljene značajke lica svih osoba koje imaju ovlaštenje za korištenje aplikacije. (Slika 3.3). Taj proces se odvija tako što algoritam uzima svaku sliku iz mape ovlaštenih osoba, te iz nje metodom *face_encodings()* kao rezultat daje polje od 128 elemenata od kojih 68 sačinjavaju ključne točke lica (algoritam zasnovan na *dlib*) (Slika 3.4)

```
# Make array of sample pictures with encodings
known_face_encodings = []
known_face_names = []
dirname = os.path.dirname(__file__)
path = os.path.join(dirname, 'known_people/')

# Make an array of all the saved jpg files' paths
list_of_files = [f for f in glob.glob(path+'*.jpg')]
# Find number of known faces
number_files = len(list_of_files)

names = list_of_files.copy()

for i in range(number_files):
    globals()['image_{}'.format(i)] = face_recognition.load_image_file(list_of_files[i])
    globals()['image_encoding_{}'.format(i)] = face_recognition.face_encodings(
        globals()['image_{}'.format(i)])[0]
    known_face_encodings.append(globals()['image_encoding_{}'.format(i)])

    # Create array of known names
    names[i] = names[i].replace(dirname+"/known_people/", "")
    known_face_names.append(names[i])
```

Slika 3.3. Instanciranje poznatih lica



Slika 3.4. 68 ključnih točki lica iz *dlib* biblioteke

Nakon što su spremne značajke osoba koje su ovlaštene, potrebno je analizirati video u stvarnom vremenu kako bi autorizirali osobu za upravljanje nad NeoPixel elementima. Autorizacija se vrši tako što se uzima svaki drugi *frame* videa te se na slici prvobitno traži lice koristeći metodu *face_location()* koja kao rezultat daje polje koje označava rubove *bounding box-a* lica. Potom se naredbom *compare_faces()* uspoređuju prepoznate značajke lica s trenutnog *frame-a* sa značajkama lica iz baze poznatih ljudi. Rezultat naredbe je popis True/False vrijednosti koje ukazuju na to koje od poznatih lica je prepoznato. (Slika 3.5)

```
# Only process every other frame of video to save time
if process_this_frame:
    # Find all the faces and face encodings in the current frame of video
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        # Use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

        face_names.append(name)

process_this_frame = not process_this_frame
```

Slika 3.5 Uspoređivanje lica s *live* videa i lica iz baze ovlaštenih

Rezultate je moguće prikazati koristeći naredbu *cv2.putText()* koja omogućava pisanje po okviru u kojem se prikazuje video. U ovom trenutku je poznata informacija o tome tko pokušava upravljati NeoPixel elementima te ukoliko je osoba ovlaštena, omogućujemo programu daljnji rad odnosno obradu *frame-a* kako bi pronašli dlan osobe i položaj tog dlana, te kako bismo utvrdili da osoba želi poslati određenu naredbu prema NeoPixel elementima.

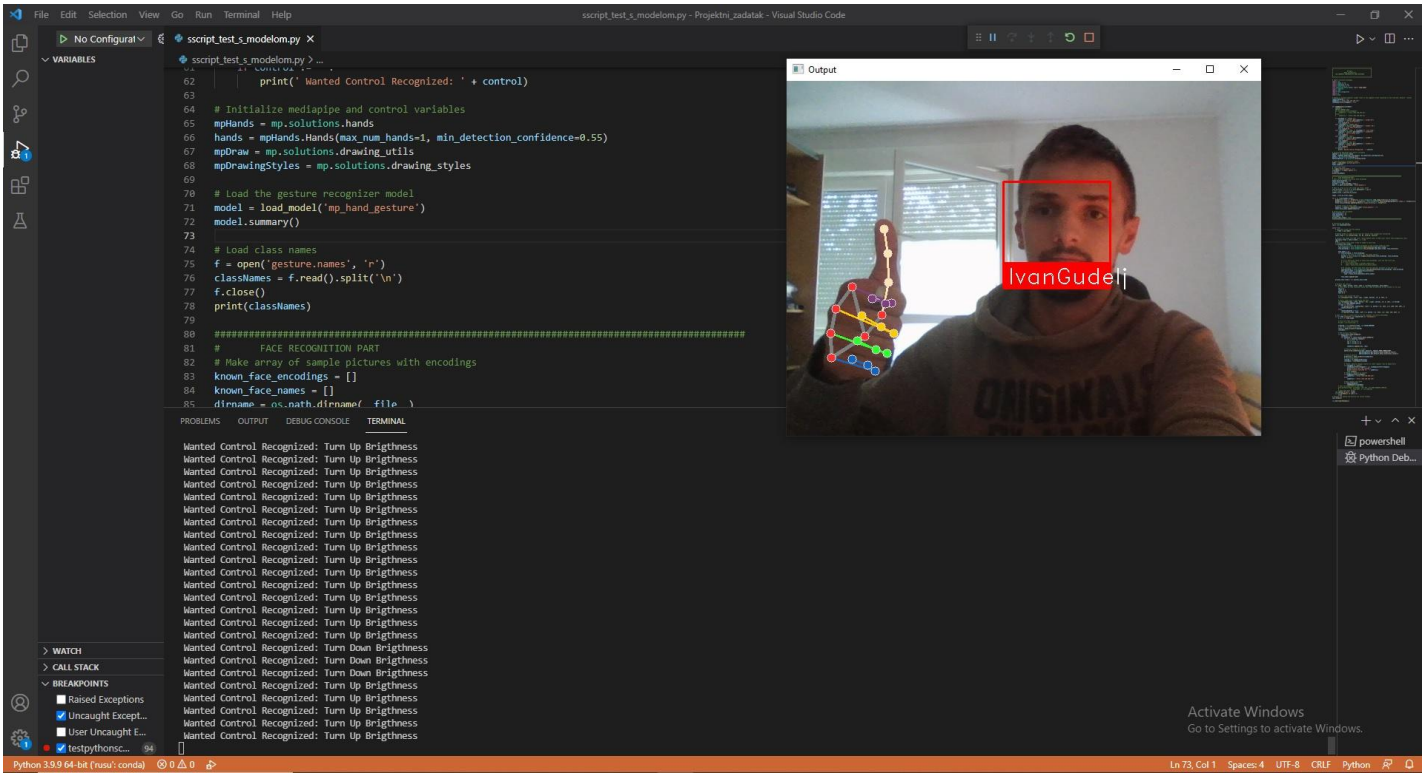
Analiza fotografije koja pronalazi dlan odnosno 21 ključnu točku se odvija koristeći *mp.solutions.hands.Hands.process()* naredbu koja kao rezultat daje skup detektiranih značajki;

> -MULTI_HAND_LANDMARKS - Svaka značajka predstavljena je kao polje veličine 1x3 (X, Y i Z). X i Y su normalizirani na raspon [0.0, 1.0] prema širini i visini, dok Z predstavlja dubinu značajke koja se mjeri u odnosu na zglobnu značajku (nulta značajka).

-MULTI_HAND_WORLD_LANDMARKS - Skup 21 značajke ruke u koordinatama stvarnog svijeta (računa se uz MHT).

-MULTI_HANDEDNESS - Za svaku prepoznatu ruku vraća vrijednosti label-score gdje label poprima vrijednosti “Left” ili “Right”, a score predstavlja procijenjena vjerojatnost predviđene ruke.

Ukoliko je korisnik autoriziran te je pronađen dlan i njegov položaj, metodom *model.predict()* model procjenjuje da li se gesta koju je korisnik napravio, nalazi u već prepoznatim gestama ruku te koja je ona. Ukoliko gesta pripada skupu poznatih gesti, moguće je poslati željenu naredbu prema ESP - 01S mikrokontroleru koristeći GET zahtjev kao što je spomenuto u 2. Poglavlju (Stranica 7.). HTTP GET zahtjev se šalje jednostavno uz pomoć biblioteke *Requests* te njene metode *Requests.get(URL)*. Primjer slanja zahtjeva za prepoznavanje lica i geste prikazan je na slici 3.6., naredbe zajedno s pripadajućim gestama se nalaze u tablici 3.1.



Slika 3.6. Prepoznata ovlaštena osoba, prepoznata gesta i slanje GET zahtjeva

4. LITERATURA

[1] https://github.com/ageitgey/face_recognition

[2] http://vis-www.cs.umass.edu/lfw/

[3] https://github.com/google/mediapipe

[4] https://www.tensorflow.org/

[5] https://arxiv.org/abs/1708.02002v2

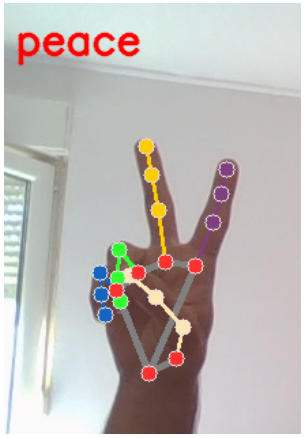
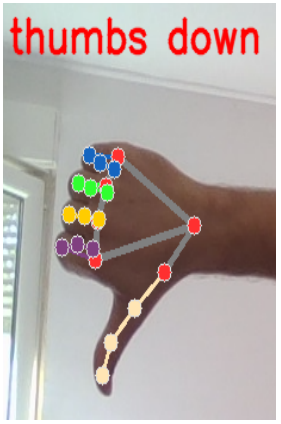
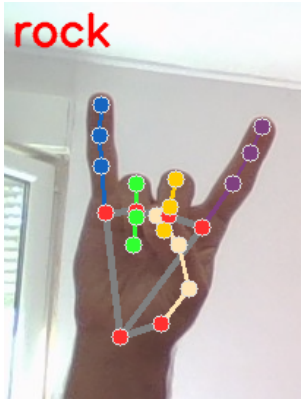
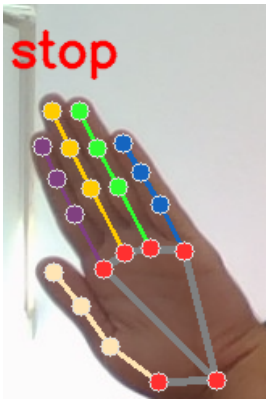
[6] https://kno.wled.ge/

5. POPIS TABLICA

Tablica 2.1 - Neke od komandi koje WLED HTTP sučelje pruža

Parametar	**Raspon vrijednosti**	**Opis funkcije**
&A=	0 - 255	Podešavanje intenziteta svjetlosti
&T=	0, 1 ili 2	Glavni ON/OFF/TOGGLE
&R=	0 - 255	Primarna *Red* vrijednost
&G=	0 - 255	Primarna *Green* vrijednost
&B=	0 - 255	Primarna *Blue* vrijednost
&W=	0 - 255	Primarna *White* vrijednost
&FX=	0 - 101	Indeks LED efekta na popisu efekta
&SX=	0 - 255	Brzina trenutnog efekta

Tablica 3.1 - Korištene geste i pripadajuće komande

		
Promjena segmenta	Povećanje intenziteta svjetlosti	Smanjenje intenziteta svjetlosti
		
Ponovno pokretanje segmenta	Promjena efekta na segmentu	Postavljanje segmenta na jednu boju