Sustav sigurnog otključavanja vrata prepoznavanjem lica korištenjem Raspberry Pi računala

Borši, Tibor

SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET

TIBOR BORŠI

SUSTAV SIGURNOG OTKLJUČAVANJA VRATA PREPOZNAVANJEM LICA KORIŠTENJEM RASPBERRY PI RAČUNALA

ZAVRŠNI RAD

Rijeka, 2024.

SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET

SUSTAV SIGURNOG OTKLJUČAVANJA VRATA PREPOZNAVANJEM LICA KORIŠTENJEM RASPBERRY PI RAČUNALA

SECURE DOOR UNLOCKING SYSTEM USING FACIAL RECOGNITION AND A RASPBERRY PI COMPUTER

ZAVRŠNI RAD

BACHELOR THESIS

Kolegij: Mikro i osobna računala

Mentor: izv. prof. dr. sc. Jasmin Ćelić

Student: Tibor Borši

Studijski smjer: Elektroničke i informatičke tehnologije u pomorstvu

JMBAG: 0112086371

Rijeka, rujan 2024.

Student: Tibor Borši

Studijski program: Elektroničke i informatičke tehnologije u pomorstvu

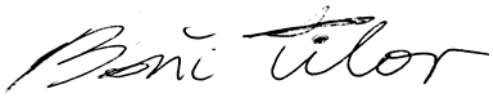
JMBAG: 0112086371

IZJAVA O SAMOSTALNOJ IZRADI ZAVRŠNOG RADA

Kojom izjavljujem da sam završni rad s naslovom SUSTAV SIGURNOG OTKLJUČAVANJA VRATA PREPOZNAVANJEM LICA KORIŠTENJEM RASPBERRY PI RAČUNALA izradio samostalno pod mentorstvom izv. prof. dr. sc. Jasmina Ćelića.

U radu sam primijenio metodologiju izrade stručnog/znanstvenog rada i koristio literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u završnom radu na uobičajen, standardan način citirao sam i povezao s fusnotama i korištenim bibliografskim jedinicama, te nijedan dio rada ne krši bilo čija autorska prava. Rad je pisan u duhu hrvatskoga jezika.

Student



(potpis)

Tibor Borši

Student: Tibor Borši

Studijski program: Elektroničke i informatičke tehnologije u pomorstvu

JMBAG: 0112086371

IZJAVA STUDENTA – AUTORA
O JAVNOJ OBJAVI OBRANJENOG ZAVRŠNOG RADA

Izjavljujem da kao student – autor završnog rada dozvoljavam Pomorskom fakultetu Sveučilišta u Rijeci da ga trajno javno objavi i besplatno učini dostupnim javnosti u cjelovitom tekstu u mrežnom digitalnom repozitoriju Pomorskog fakulteta.

U svrhu podržavanja otvorenog pristupa završnim radovima trajno objavljenim u javno dostupnom digitalnom repozitoriju Pomorskog fakulteta, ovom izjavom dajem neisključivo imovinsko pravo iskorištavanja bez sadržajnog, vremenskog i prostornog ograničenja mog završnog rada kao autorskog djela pod uvjetima *Creative Commons* licencije CC BY Imenovanje, prema opisu dostupnom na http://creativecommons.org/licenses/

Student– autor

(potpis)

Tibor Borši

SAŽETAK

Ovaj završni rad bavi se izradom sustava sigurnog otključavanja vrata temeljenog na prepoznavanju lica koristeći Raspberry Pi 4 računalo. Sustav koristi Python za obradu slike i prepoznavanje lica korištenjem biblioteke poput OpenCV. Sustavi prepoznavanja lica postaju sve popularniji u svijetu zbog svoje praktičnosti, brzine i visokog stupnja sigurnosti koje pružaju. Implementacija sustava prepoznavanja lica na Raspberry Pi-u omogućuje pristupačno i učinkovito rješenje za kućnu sigurnost. Rad uključuje detalje o hardverskoj i softverskoj implementaciji, uključujući modeliranje u programu Fusion360 i 3D ispis vrata. Rezultati pokazuju visoku učinkovitost sustava u stvarnim uvjetima.

Ključne riječi: prepoznavanje lica, Raspberry Pi 4, sigurnost, Python, 3D ispis.

SUMMARY

This final thesis deals with the development of a secure door unlocking system based on facial recognition using Raspberry Pi 4. The system utilizes Python for image processing and facial recognition, employing libraries such as OpenCV and dlib. Facial recognition systems are becoming increasingly popular worldwide due to their convenience, speed, and high level of security they provide. Implementing a facial recognition system on the Raspberry Pi offers an affordable and effective solution for home security. The thesis includes details on hardware and software implementation, including modeling in Fusion360 and 3D printing of the door. The results demonstrate high system efficiency in real-world conditions.

Keywords: facial recognition, Raspberry Pi 4, security, Python, 3D printing.

SADRŽAJ

1. UVOD

U suvremenom svijetu prepoznavanje lica postaje sve popularnije zahvaljujući svojoj jednostavnosti, visokoj razini sigurnosti i mogućnosti izbjegavanja fizičkog kontakta u sigurnosnim sustavima. Ova tehnologija korisnicima omogućuje brz i nesmetan pristup prostorijama ili podacima, eliminirajući potrebu za tradicionalnim ključevima ili karticama. Raspberry Pi, zahvaljujući svojoj povoljnosti i fleksibilnosti, predstavlja idealnu platformu za implementaciju inovativnih rješenja u različitim tehničkim područjima. Upotrebom Pythona i naprednih biblioteka kao što je OpenCV, sustav za prepoznavanje lica postaje dostupan čak i korisnicima s osnovnim programerskim vještinama.

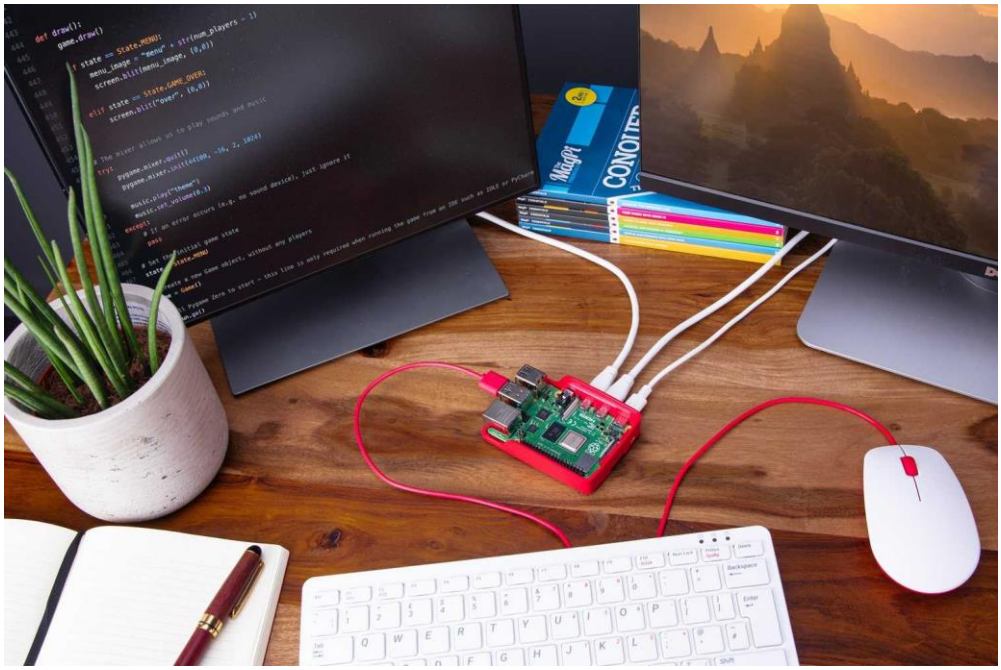
Ovaj rad fokusira se na integraciju Raspberry Pi računala s web kamerom za snimanje i prepoznavanje lica te servo motora koji služi kao aktuator za mehaničko otključavanje vrata. Detaljno će biti obrađena hardverska i softverska implementacija, uključujući odabir komponenti, njihovu konfiguraciju i integraciju u funkcionalan sustav. Softverski dio uključuje detaljan opis algoritama za detekciju i prepoznavanje lica, s posebnim naglaskom na obradu slika i usporedbu uzoraka.

U dijelu posvećenom mehaničkom dizajnu, bit će prikazan proces 3D modeliranja vrata kako bi se omogućila integracija elektroničkih i mehaničkih komponenti u funkcionalnu cjelinu. Kroz sve navedeno, rad će istaknuti sve veću važnost sustava prepoznavanja lica u kontekstu modernih sigurnosnih izazova, naglašavajući njegovu primjenjivost i prednosti u odnosu na tradicionalne metode pristupa.

2. RASPBERRY PI

2.1. Općenito o Raspberry Pi računalu

Raspberry Pi je računalna platforma tipa SBC (Single Board Computer) razvijena u Ujedinjenom Kraljevstvu 2012. godine od tvrtke "Raspberry Pi Foundation" s primarnim ciljem edukacije studenata i učenika u osnovama računalstva. Ova platforma predstavlja pristupačno i funkcionalno rješenje, nudeći korisnicima puni računalni sustav po vrlo niskoj cijeni. Raspberry Pi može koristiti različite operativne sustave među kojima se Linux pokazao kao prihvatljivo rješenje zato jer osigurava fleksibilnost i široku podršku za raznovrsne aplikacije. Opremljen je najčešće korištenim priključcima poput HDMI, USB i Ethernet što omogućava jednostavno povezivanje s različitim perifernim uređajima. Jedna od ključnih značajki Raspberry Pi računala su GPIO (General Purpose Input/Output) pinovi, koji omogućuju povezivanje i kontrolu različitih elektroničkih komponenti poput senzora i aktuatora. Ovi pinovi pružaju korisnicima mogućnost direktne interakcije s hardverom, što je posebno korisno za projekte u području automatizacije i robotike. Danas, Raspberry Pi se široko koristi u kućnim i poslovnim primjenama, od hobističkih projekata elektronike do raznovrsnih osobnih projekata koje ljudi realiziraju. Njegova pristupačnost, zajedno s bogatom dokumentacijom i podrškom zajednice, čini ga idealnim izborom za edukaciju, prototipiranje i DIY (Do It Yourself) projekte.



Slika 1: Radna stanica sa Raspberry Pi računalom

Izvor: https://www.raspberrypi.com/for-home/

2.2. Modeli Raspberry Pi računala

Od svog prvog izlaska 2012. godine, Raspberry Pi je kroz svaku iteraciju donio značajna poboljšanja u specifikacijama i mogućnostima povezivanja s vanjskom periferijom. Ove nadogradnje učinile su Raspberry Pi sve svestranijim i moćnijim uređajem, sposobnim za sve širi spektar primjena. Prvi model Raspberry Pi - Raspberry Pi 1 Model B imao je osnovne specifikacije s 256 MB RAM-a i nekoliko USB portova, dok najnoviji model Raspberry Pi 5 ima četverojezgreni procesor, do 8 GB RAM-a i napredne mogućnosti povezivanja, uključujući USB 3.0 i dvostruki HDMI izlaz. Današnji najmoderniji Raspberry Pi modeli mogu se bez problema usporediti s većinom računala koja ljudi imaju kod kuće, jer su ih specifikacijama dostigli, čime su postali relevantna opcija za razne primjene, od kućnih servera i medijskih centara do obrazovnih alata i razvojnih platformi za profesionalce.

Slika 2: Raspberry Pi 1 Model B

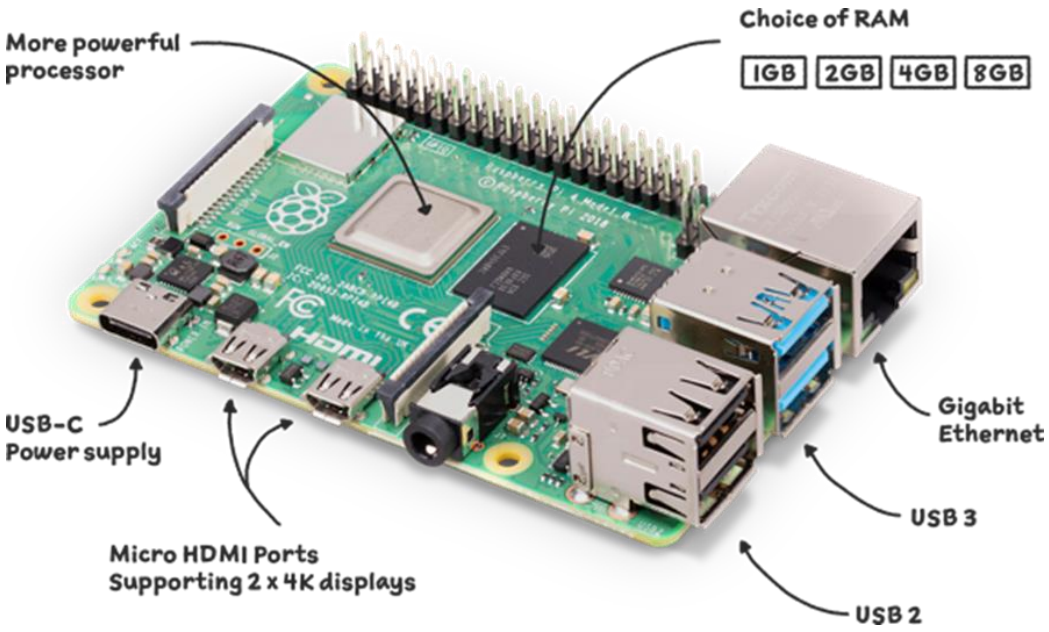
Izvor:
https://cdn.sparkfun.com/assets/parts/7/4/9/7/11546-01.jpg

Slika 3: Raspberry Pi 5

Izvor:
https://thepihut.com/cdn/shop/files/raspberry-pi-5-raspberry-pi-40958498898115_800x.jpg?v=1695819922

2.2.1. Raspberry Pi 4 Model B

Raspberry Pi 4 Model B je verzija Raspberry Pi računala koja se koristi u ovome radu, a predstavlja značajnu prekretnicu u razvoju ove platforme. Ova verzija donijela je mnoga poboljšanja u odnosu na prethodne modele, uključujući unapređenja memorije, procesora, USB portova, priključka za napajanje, HDMI izlaza i mrežnih mogućnosti. Što se tiče procesora, Raspberry Pi 4 Model B donio je prvu veliku promjenu prelaskom s Cortex-A53 na Cortex-A72. Ovaj novi procesor omogućuje znatno bolje performanse, s frekvencijom rada povećanom za čak 100 MHz uz minimalno povećanje potrošnje energije. Pored poboljšanja CPU-a, unaprijeđen je i GPU, prelaskom s Broadcom VideoCore IV na VideoCore VI. Ovaj napredak omogućuje Raspberry Pi 4 Model B podršku za dva 4K ekrana, čime se značajno povećava njegova funkcionalnost u multimedijskim aplikacijama. Uz promjene u procesoru i grafici, Raspberry Pi 4 Model B također je dobio USB 3.0 portove uz postojeće USB 2.0 portove, što omogućava brži prijenos podataka i veću kompatibilnost s modernim perifernim uređajima. Također je prvi model koji je omogućio korisnicima izbor količine memorije prilikom kupnje, pružajući opcije od 1 GB, 2 GB, 4 GB i 8 GB RAM-a. Ove opcije omogućuju prilagodbu uređaja specifičnim potrebama korisnika, bilo da se radi o jednostavnim projektima ili zahtjevnijim aplikacijama. Promjene u priključku za napajanje uključuju prelazak na USB-C, što omogućava stabilnije napajanje i veću kompatibilnost s modernim adapterima.



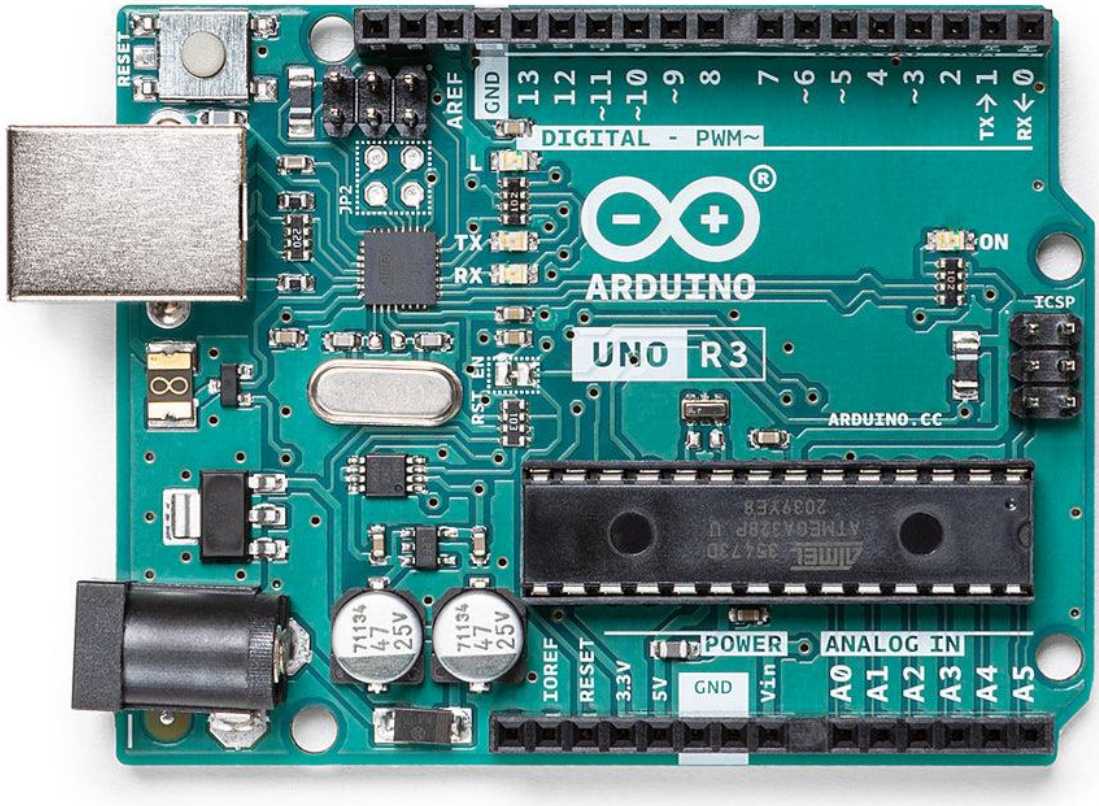
Slika 4: Raspberry Pi 4 Model B

Izvor: https://assets.raspberrypi.com/static/raspberry-pi-4-labelled-f5e5dcdf6a34223235f83261fa42d1e8.png

2.2.1. Konkurentske platforme

U svijetu jednopločnih računala, Raspberry Pi ima nekoliko značajnih konkurenata, poput Arduino, BeagleBone Black, Odroid serije, Jetson Nano i Banana Pi. Svaka od ovih platformi ima svoje jedinstvene značajke, čineći ih pogodnim za različite aplikacije i potrebe korisnika, pružajući alternativu Raspberry Pi-u ovisno o specifičnim zahtjevima.

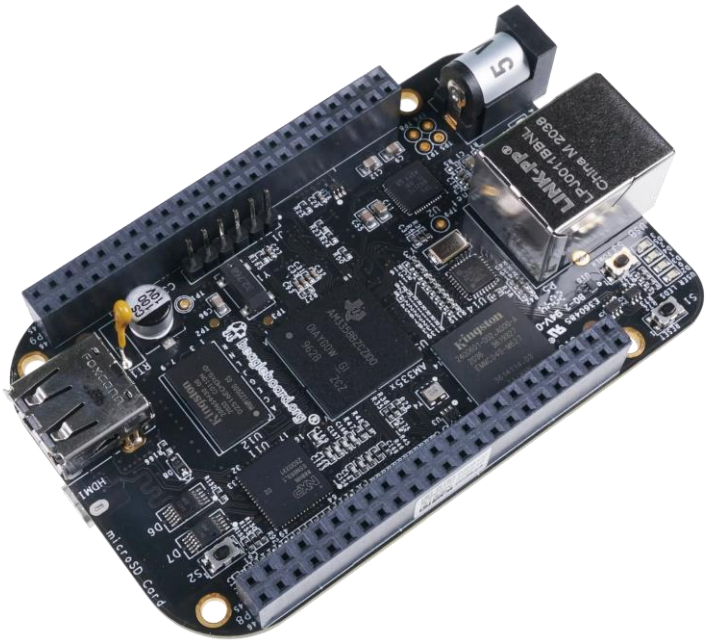
Arduino je poznat kao popularan mikrokontroler idealan za jednostavne zadatke upravljanja hardverom. Za razliku od Raspberry Pi-a, Arduino nije punopravno računalo; nema operativni sustav i temelji se na mikrokontrolerskoj arhitekturi, omogućavajući direktno programiranje hardverskih komponenti putem Arduino IDE-a. Najčešće se koristi u projektima robotike, DIY elektronike i automatizacije, gdje su potrebni niska potrošnja energije i direktan pristup hardveru. Iako nije toliko moćan kao Raspberry Pi, idealan je za aplikacije koje zahtijevaju preciznu kontrolu senzora i aktuatora.



Slika 5: Arduino Uno Rev3

Izvor: https://store.arduino.cc/en-hr/products/arduino-uno-rev3

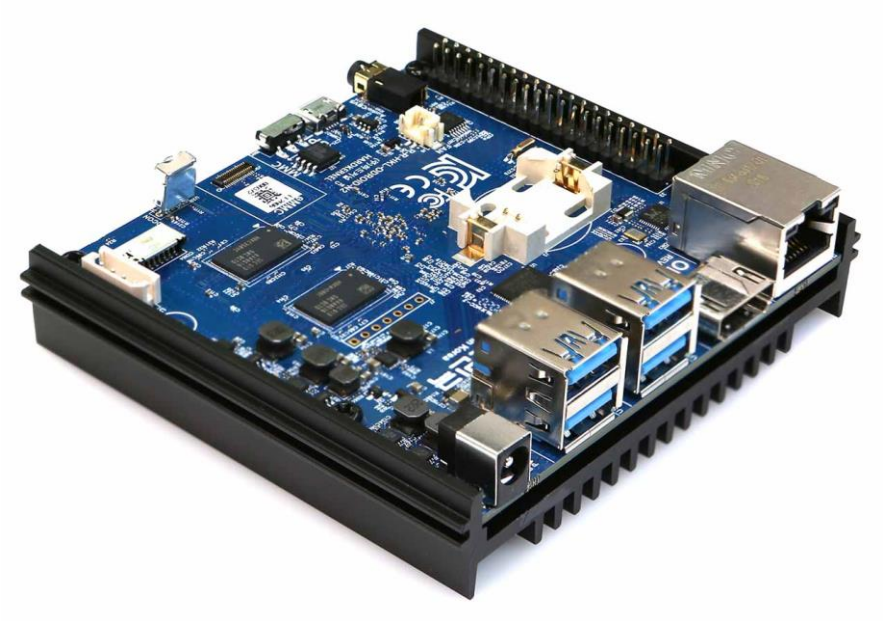
BeagleBone Black je još jedna popularna opcija koja se izdvaja po većem broju I/O priključaka i boljoj podršci za industrijske primjene u usporedbi s Raspberry Pi-em. Opremljen ARM Cortex-A8 procesorom, može pokretati Linux distribucije poput Debian-a, što ga čini pogodnim za razvoj aplikacija koje zahtijevaju stabilnost i pouzdanost. Zbog naprednih mogućnosti I/O priključaka, često se koristi u projektima robotike i industrijske automatizacije gdje je potrebno povezivanje različitih senzora i aktuatora. BeagleBone Black je izbor za one koji trebaju stabilno i dugotrajno rješenje za kritične aplikacije.



Slika 6: BeagleBone Black

Izvor: https://www.beagleboard.org/boards/beaglebone-black

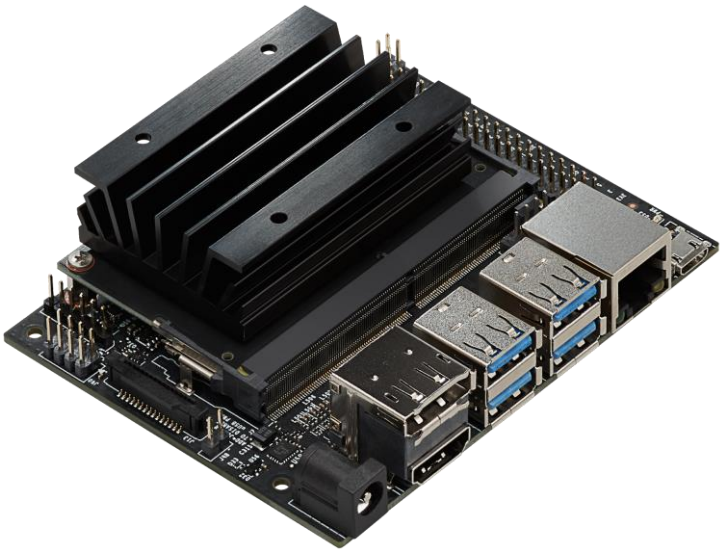
Odroid serija, uključujući modele poput Odroid-C4 i Odroid-N2+, poznata je po snažnim hardverskim specifikacijama, često nadmašujući performanse drugih jednopločnih računala, uključujući Raspberry Pi. Sa snažnim ARM Cortex procesorima i podrškom za različite Linux distribucije i Android, Odroid uređaji su prikladni za aplikacije koje zahtijevaju visoke performanse, kao što su multimedijalni centri, serveri i igre. Ovi uređaji nude veću memoriju i brže sustave pohrane, što ih čini pogodnima za zahtjevne zadatke poput obrade videa i igranja igara.



Slika 7: ODROID-N2+

Izvor: https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram-2/

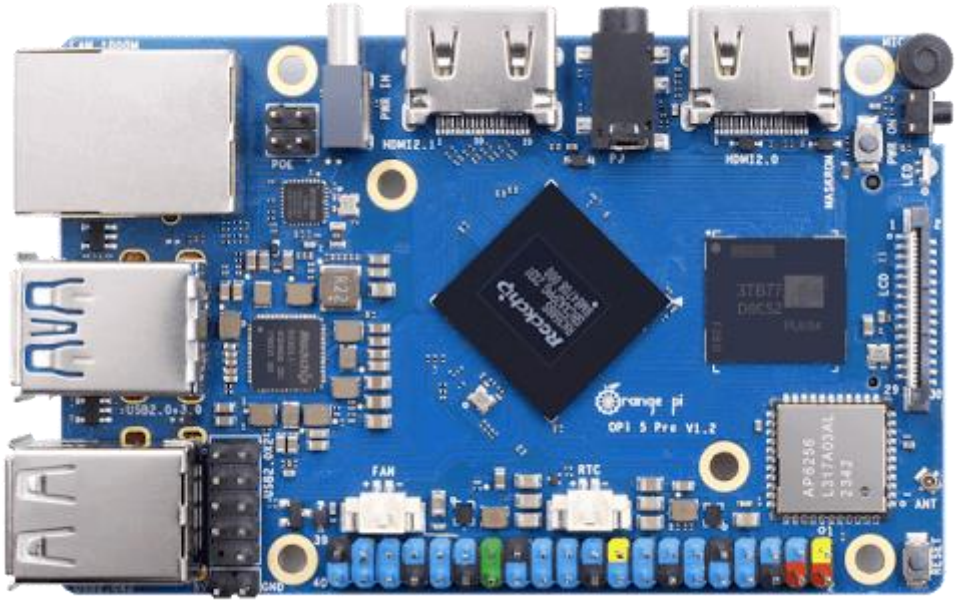
Jetson Nano, razvijen od strane NVIDIA-e, specijaliziran je za aplikacije u području umjetne inteligencije i strojnog učenja. Opremljen je ARM Cortex-A57 CPU-om i GPU-om s 128 CUDA jezgri, što omogućava učinkovitu obradu paralelnih zadataka i složenih AI algoritama. Koristi Linux za Tegra, specijaliziranu verziju Ubuntu-a prilagođenu za NVIDIA hardver, idealnu za razvoj aplikacija računalnog vida, dubokog učenja i robotike. Jetson Nano je stoga specijaliziran za zadatke koji zahtijevaju visoku računalnu snagu i specifičnu podršku za AI aplikacije.



Slika 8: Jetson Nano

Izvor: https://developer.nvidia.com/blog/jetson-nano-ai-computing/

Banana Pi je platforma koja se koristi za slične projekte kao i Raspberry Pi, ali nudi dodatne mogućnosti kao što su bolja mrežna povezivost sa gigabitnim Ethernetom i veća memorija, ovisno o modelu. Podržava različite Linux distribucije i Android, što ga čini fleksibilnim za širok spektar aplikacija, od medijskih centara do obrazovnih platformi.



Slika 9: Orange Pi 5 Pro

Izvor: https://developer.nvidia.com/blog/jetson-nano-ai-computing/

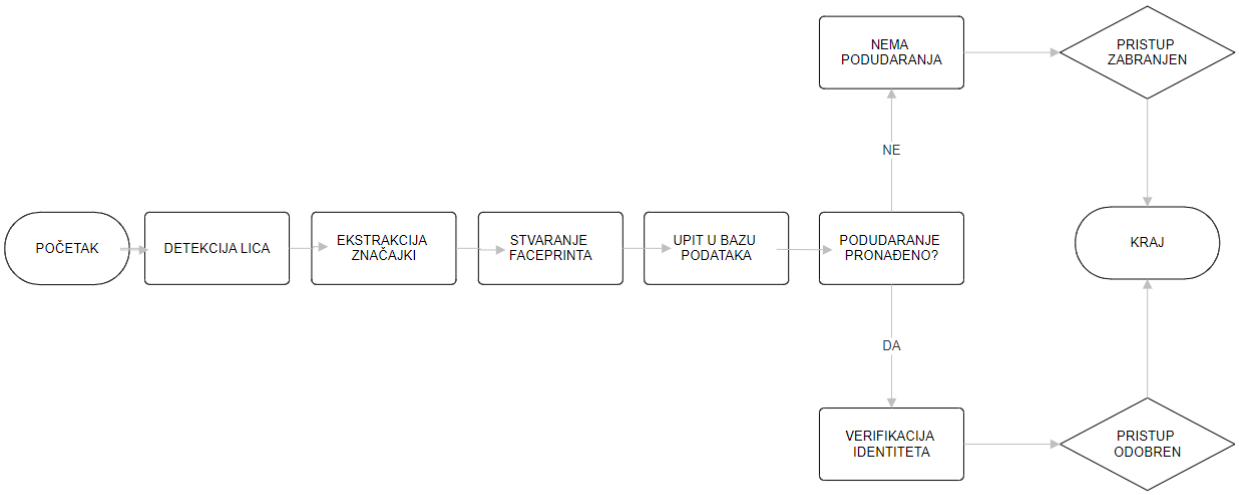
U konačnici, izbor platforme ovisi o specifičnim zahtjevima projekta i preferencijama korisnika. Dok Raspberry Pi nudi široku svestranost i podršku zajednice, druge platforme kao što su Arduino, BeagleBone Black, Odroid, Jetson Nano, Banana Pi i Orange Pi nude specijalizirane karakteristike i prednosti za različite aplikacije i primjene.

3. PREPOZNAVANJE LICA

3.1. ŠTO JE PREPOZNAVANJE LICA?

Prepoznavanje lica spada pod kategoriju biometrijske sigurnosti, uz prepoznavanje glasa, prepoznavanje otiska prsta i prepoznavanje retine oka. Ova tehnologija temelji se na analizi karakteristika lica, poput udaljenosti između očiju, oblika nosa i strukture jagodica. Proces prepoznavanja lica odvija se kroz nekoliko koraka:

1. Detekcija lica: Prvi korak uključuje pronalazak lica na slici ili videu. Ovaj korak koristi algoritme za prepoznavanje oblika lica kako bi identificirao prisutnost lica u vizualnim podacima.
2. Analiza karakteristika lica: Nakon detekcije, sustav analizira specifične karakteristike lica. To uključuje mjerenje udaljenosti između ključnih točaka na licu, poput očiju, nosa i usta, te mapiranje oblika i strukture tih značajki.
3. Pretvorba u digitalni zapis: Identificirane karakteristike lica pretvaraju se u digitalni zapis poznat kao "faceprint". Faceprint je jedinstveni kod koji predstavlja specifične značajke lica pojedinca.
4. Usporedba s bazom podataka: Odabrani algoritam zatim analizira faceprint i uspoređuje ga s bazom podataka poznatih lica. Cilj je pronaći podudaranje koje bi potvrdilo identitet osobe. Ako algoritam pronađe dovoljno visok stupanj podudarnosti, identitet osobe se potvrđuje.



Shema 1: Dijagram toka procesa prepoznavanja

Izvor: napravljeno od strane studenta

3.1.1. PRIMJENA SUSTAVA PREPOZNAVANJA LICA U SVIJETU

Prepoznavanje lica postaje sve važnija tehnologija zbog svojih sigurnosnih i praktičnih prednosti. Ova tehnologija nalazi primjenu u različitim sektorima, uključujući i svakodnevnu upotrebu, sigurnost, trgovinu, bankarstvo, zdravstvo, obrazovanje, automobilsku industriju, kockarnice i potragu za nestalim osobama.

U svakodnevnoj upotrebi, prepoznavanje lica koristi se za otključavanje pametnih telefona, pružajući dodatnu sigurnost osobnim podacima korisnika. U sektoru sigurnosti, policijske službe primjenjuju ovu tehnologiju za identifikaciju osumnjičenih, dok se u zračnim lukama koristi za provjeru identiteta putnika, povećavajući sigurnost i smanjujući vrijeme čekanja.

U trgovinama se prepoznavanje lica koristi za identifikaciju poznatih kradljivaca i poboljšanje korisničkog iskustva kroz personalizirane usluge. U bankarstvu, ova tehnologija omogućuje autorizaciju transakcija bez potrebe za lozinkama, smanjujući rizik od prijevara. U zdravstvu, bolnice koriste prepoznavanje lica za pristup medicinskim kartonima i praćenje emocionalnog stanja pacijenata. U obrazovnim ustanovama i na radnim mjestima koristi se za praćenje prisutnosti učenika i zaposlenika.

Automobilska industrija eksperimentira s prepoznavanjem lica kako bi zamijenila ključeve i prilagođavala postavke vozila, dok kockarnice koriste ovu tehnologiju za identifikaciju ovisnika o kockanju i praćenje njihovih aktivnosti. Prepoznavanje lica također pomaže u pronalaženju nestalih osoba i žrtava trgovine ljudima.

Konkretni primjeri velikih tvrtki koje primjenjuju prepoznavanje lica su sljedeći:

- Apple koristi prepoznavanje lica za otključavanje telefona, prijavu u aplikacije i obavljanje kupovina.
- British Airways omogućava putnicima ukrcaj na letove iz SAD-a bez pokazivanja putovnice ili ukrcajne karte.
- Cigna omogućuje klijentima u Kini podnošenje zahtjeva za zdravstveno osiguranje putem fotografije umjesto pisanog potpisa.
- Coca-Cola koristi prepoznavanje lica za nagrađivanje kupaca koji recikliraju na automatima u Kini te za personalizirane reklame na automatima u Australiji.
- Facebook je uveo automatsko označavanje ljudi na fotografijama pomoću alata za prijedloge oznaka.

- Google koristi ovu tehnologiju u Google Photos za sortiranje slika i automatsko označavanje na temelju prepoznatih osoba.
- Snapchat omogućuje kreiranje filtera koji se prilagođavaju korisnikovom licu.
- McDonald's koristi prepoznavanje lica u restoranima u Japanu za procjenu kvalitete usluge.

Prepoznavanje lica donosi mnoge prednosti, ali suočava se i sa izazovima vezanim uz privatnost i točnost, što zahtijeva daljnja istraživanja i regulaciju kako bi se osigurala etička uporaba. Prednosti i nedostaci prepoznavanja lica bit će detaljno razmotreni u sljedećem poglavlju.

3.1.2. PREDNOSTI I NEDOSTACI SUSTAVA PREPOZNAVANJA LICA

Prepoznavanje lica već je opisano kao tehnologija koja donosi brzinu i učinkovitost sustava te izostanak potrebe za fizičkim kontaktom. Osim tih prednosti, ova tehnologija donosi i dodatne koristi i nedostatke.

3.1.2.1. Prednosti

Prepoznavanje lica omogućuje brzo i precizno identificiranje osoba, čime se povećava sigurnost u različitim okruženjima. Također, smanjuje se rizik od krađe identiteta i prijevara, omogućujući autentifikaciju korisnika bez potrebe za tradicionalnim lozinkama. Tehnologija donosi praktičnost jer omogućuje jednostavno otključavanje uređaja i autorizaciju radnji, a može raditi i offline, čime se dodatno poboljšava brzina i učinkovitost. Prepoznavanje lica omogućuje personalizirane usluge u različitim sektorima, automatski prilagođavajući postavke prema preferencijama korisnika. U industrijama usmjerenim prema korisnicima, eliminira potrebu za pamćenjem lozinki ili nošenjem identifikacijskih kartica, što rezultira višom razinom zadovoljstva i lojalnosti korisnika.

3.1.2.2. Nedostaci

Tehnologija prepoznavanja lica suočava se s nizom izazova. Prikupljanje, pohrana i analiza biometrijskih podataka može izazvati zabrinutost za privatnost i sigurnost podataka. Ako podaci nisu pravilno zaštićeni, mogu biti podložni provalama i zloupotrebi. Sustavi prepoznavanja lica mogu imati poteškoća s varijacijama u osvjetljenju, kutovima i izrazima

lica, što se može riješiti implementacijom multimodalnih biometrijskih sustava koji kombiniraju više tehnologija za poboljšanje točnosti. Algoritmi prepoznavanja lica mogu biti pristrani prema rasama i spolovima, posebno ako su trenirani na neraznovrsnim skupovima podataka. Upotreba raznolikih skupova podataka može smanjiti ovaj problem i poboljšati točnost prepoznavanja. Sustavi prepoznavanja lica mogu biti prevareni fotografijama ili maskama. Napredne značajke poput detekcije živosti mogu razlikovati žive osobe od fotografija ili maski, smanjujući rizik od neovlaštenog pristupa.

3.2. ALGORITMI ZA DETEKCIJU I PREPOZNAVANJE LICA

U svijetu biometrijske sigurnosti, algoritmi za detekciju i prepoznavanje lica igraju ključnu ulogu u identifikaciji i verifikaciji osoba na temelju njihovih karakteristika lica. Razlikovanje između detekcije i prepoznavanja lica je od bitne važnosti za razumijevanje rada ovih algoritama. Detekcija lica odnosi se na proces lociranja lica unutar slike ili videozapisa. Ovo je prvi korak u kojem se identificiraju regije unutar vizualnog inputa koje sadrže lica, bez obzira na identitet osobe. S druge strane, prepoznavanje lica uključuje proces identifikacije ili verifikacije osobe čije je lice već detektirano. Prepoznavanje zahtijeva usporedbu detektiranih lica s bazom podataka poznatih lica radi utvrđivanja identiteta.

Za detekciju lica koriste se različiti algoritmi, među kojima su najpopularniji Haar Cascade, Histogram of Oriented Gradients (HOG) i Convolutional Neural Networks (CNN). Haar Cascade je poznat po svojoj brzini i efikasnosti, dok HOG koristi gradijentne informacije za robusnu detekciju lica u različitim uvjetima. CNN-ovi, kao dio dubokog učenja, pružaju najpreciznije rezultate detekcije koristeći složene neuronske mreže.

Prepoznavanje lica, s druge strane, često se oslanja na napredne tehnike dubokog učenja i vektorizacije lica. Convolutional Neural Networks koriste se i za detekciju i za prepoznavanje lica zbog svoje sposobnosti da uče složene obrasce u slikama. CNN modeli kao što su DeepFace i FaceNet postigli su značajne uspjehe u ovom polju.

U nastavku će biti pobliže analiziran rad Haar Cascade, CNN i HOG algoritama, istražujući njihove mehanizme, prednosti i ograničenja. Ova analiza pružit će dublji uvid u to kako ovi algoritmi doprinose procesu detekcije i prepoznavanja lica te kako ih učinkovito primijeniti u različitim aplikacijama.

3.2.1. HAAR CASCADE ALGORITAM

Haar Cascade algoritam razvili su Paul Viola i Michael Jones 2001. godine, a njegova primjena opisana je u radu iz 2001. godine na Međunarodnoj konferenciji o računalnoj viziji i prepoznavanju uzoraka (ICCVPR). Ovaj algoritam postao je popularan zbog svoje brzine i efikasnosti u detekciji lica u stvarnom vremenu.

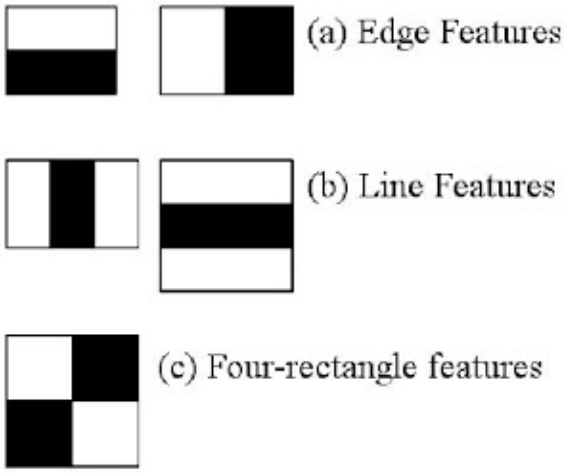
Haar Cascade algoritam koristi se u različitim aplikacijama, uključujući nadzor, sigurnost i interakciju čovjeka i računala. Unatoč tome što su noviji algoritmi poput konvolucijskih neuronskih mreža (CNN) postali dominantni, Haar Cascade se i dalje koristi zbog svoje jednostavnosti i niskih zahtjeva za računalnim resursima.

Proces detekcije lica pomoću Haar Cascade algoritma temelji se na korištenju rektangulskih značajki koje opisuju različite dijelove lica. Algoritam se sastoji od više kaskadnih klasifikatora koji su trenirani na velikom broju pozitivnih (lica) i negativnih (ne-lica) slika. Svaki klasifikator u kaskadi filtrira regije slike, postupno smanjujući broj potencijalnih lica koja se dalje analiziraju.

U sljedećem dijelu bit će detaljno opisani koraci rada Haar Cascade algoritma, uključujući izračunavanje značajki, generiranje integralne slike i primjenu kaskade klasifikatora za detekciju lica.

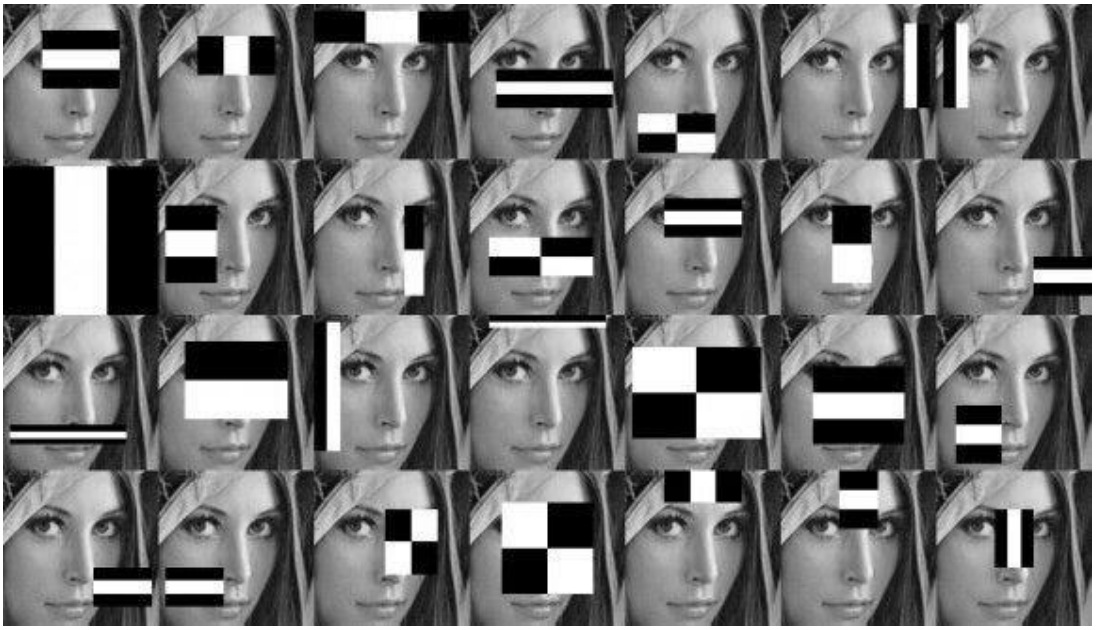
1. Izračunavanje Haarovih značajki:

Haarove značajke temelje se na kontrastu između svijetlih i tamnih područja slike. Najčešće korištene značajke uključuju pravokutne regije u različitim konfiguracijama (dvije, tri ili četiri susjedne pravokutne regije). Izračunava se razlika između suma piksela u svijetlim i tamnim regijama, čime se identificiraju osnovni obrasci kao što su rubovi, linije i promjene teksture.



Slika 10: Tipovi Haar značajki

Izvor: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d



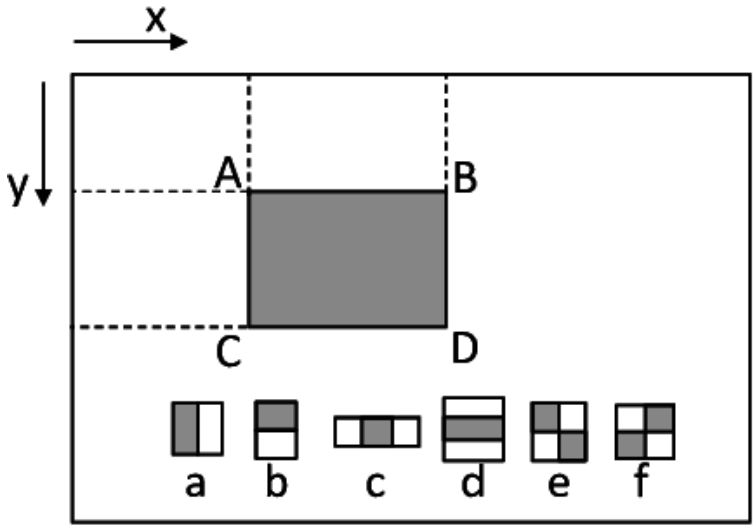
Slika 11: Princip rada izvlačenja Haar značajki

Izvor: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

2. Generiranje integralne slike:

Uvođenje integralne slike značajno ubrzava izračunavanje Haarovih značajki, što je ključno za učinkovito prepoznavanje lica. Integralna slika omogućuje brzo računanje zbroja vrijednosti piksela unutar pravokutnika, čime se smanjuje potreba za izračunom na svakoj pojedinoj točki

slike. Ovaj pristup koristi reference na podpravokutnike, čime se optimizira proces izračunavanja Haarovih značajki.

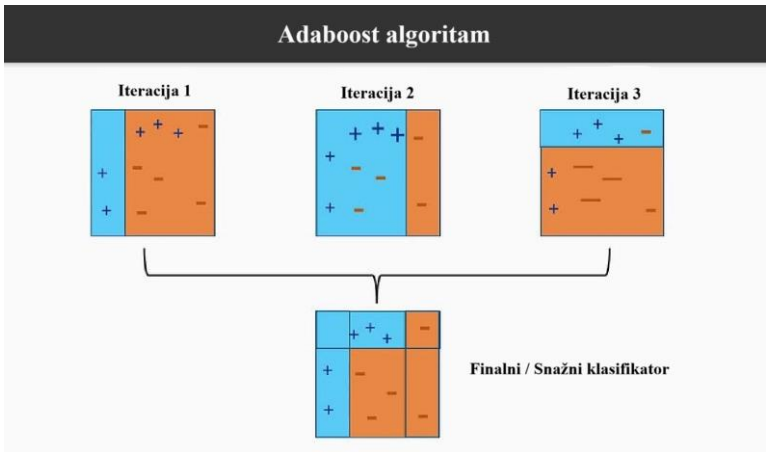


Slika 12: Ilustracija načina rada integralne slike

Izvor: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

3. Adaboost algoritam za selekciju značajki:

Adaboost algoritam koristi se za odabir najznačajnijih značajki među tisućama generiranih značajki. Kreira snažan klasifikator kombiniranjem velikog broja slabih klasifikatora, pri čemu svaki odabire najbolje značajke koje pomažu u razlikovanju lica od ne-lica. Adaboost također daje težinski čimbenik svakom klasifikatoru, poboljšavajući točnost detekcije.



Slika 13: Princip rada Adaboost algoritma

Izvor: autor

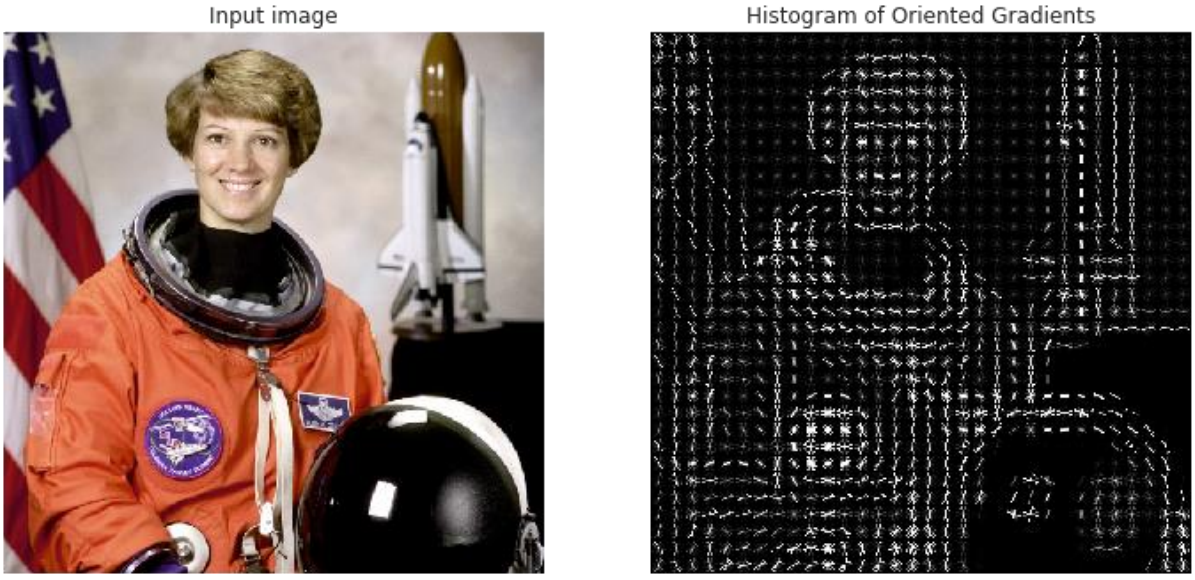
4. Primjena kaskade klasifikatora:

Kaskada klasifikatora sastoji se od serije jednostavnih klasifikatora primijenjenih sekvencijalno na regije slike. Svaki klasifikator odbacuje negativne primjere (ne-lica) i propušta potencijalna lica do sljedećeg klasifikatora u kaskadi. Ovaj pristup značajno smanjuje vrijeme obrade, jer većina negativnih regija bude odbačena u ranijim fazama, omogućujući efikasnu detekciju lica.

3.2.2. HISTOGRAM OF ORIENTED GRADIENTS ALGORITAM

Histogram of Oriented Gradients (HOG) algoritam je metoda za detekciju objekata koja se oslanja na analizu gradijentnih orijentacija u lokalnim dijelovima slike. Razvijen 2005. godine, HOG algoritam postao je popularan zbog svoje temeljitosti i točnosti u prepoznavanju lica i drugih objekata u različitim uvjetima osvjetljenja i pozicioniranja. HOG se temelji na pretpostavci da lokalni objekti i oblici u slici mogu biti učinkovito opisani raspodjelom intenziteta gradijenata ili smjerova rubova.

HOG algoritam koristi se u raznim aplikacijama, uključujući sigurnosne sustave, nadzor i interakciju čovjeka i računala. Njegova učinkovitost i preciznost čine ga relevantnim za uvjete gdje su varijacije osvjetljenja i pozicioniranja značajni faktori. HOG se posebno ističe u scenarijima gdje je potrebno brzo i točno prepoznavanje objekata u stvarnom vremenu.



Slika 14: Princip rada HOG algoritma

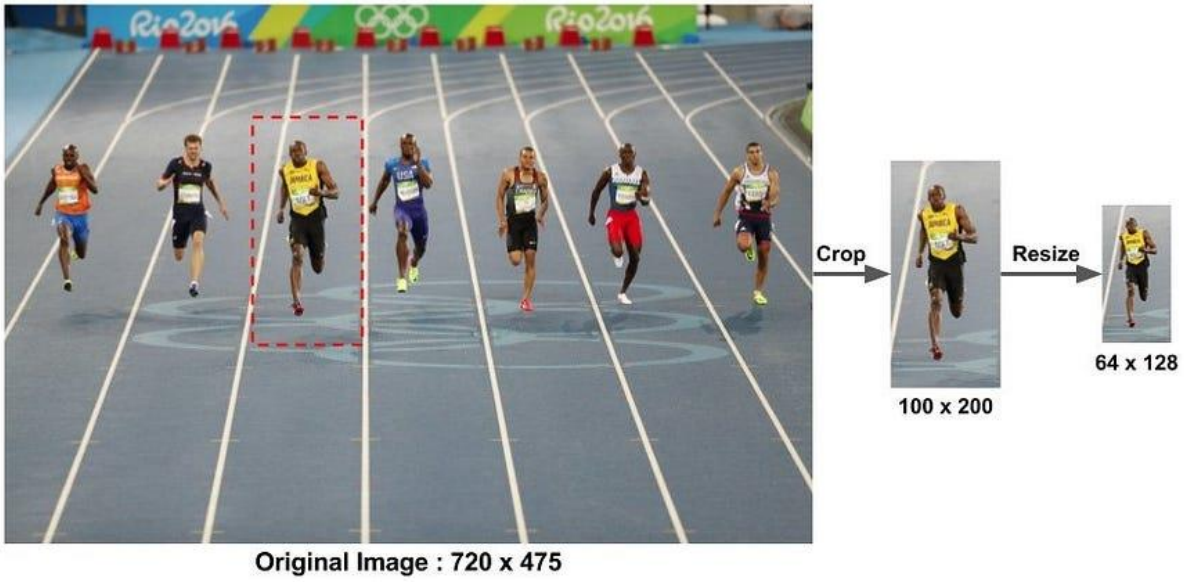
Izvor: https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa

Proces detekcije lica pomoću HOG algoritma uključuje nekoliko ključnih koraka koji omogućuju precizno prepoznavanje obrazaca u slikama.

U nadolazećem dijelu bit će opisani ključni koraci u radu HOG algoritma, uključujući pretprocesiranje slike, izračunavanje gradijenata i generiranje histograma orijentacija za preciznu detekciju lica.

1. Pretprocesiranje slike:

Prvi korak uključuje normalizaciju slike kako bi se smanjile varijacije u osvjetljenju. Ovo se postiže prilagodbom intenziteta piksela kako bi se osiguralo da gradijenti bolje predstavljaju strukturu slike, a ne osvjetljenje.



Slika 15: Primjer pretprocesiranja slika

Izvor: https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa

2. Izračunavanje gradijenata:

Svaki piksel slike prolazi kroz izračun gradijenta, pri čemu se dobivaju informacije o smjeru i veličini promjene intenziteta. Za izračunavanje gradijenata unutar HOG deskriptora, prvo je potrebno dobiti horizontalne i vertikalne gradijente slike. Ovo se postiže korištenjem filtara koji ističu promjene u intenzitetu piksela. Konkretno, horizontalni i vertikalni gradijenti se računaju primjenom Sobelovih operatora.

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Slika 16: Sobelovi operatori

Izvor: https://www.projectrhea.org/rhea/index.php/An_Implementation_of_Sobel_Edge_Detection

Nakon što smo filtrirali sliku kroz Sobelove operatore, možemo izračunati magnitudu i smjer gradijenata koristeći sljedeće formule:

$$\emptyset = \tan^{-1}\frac{g_y}{g_x}$$

$$g = \sqrt{{g_x}^2 + {g_y}^2}$$

Gdje je:

g_x - komponenta gradijenta u x-smjeru (horizontalna komponenta)

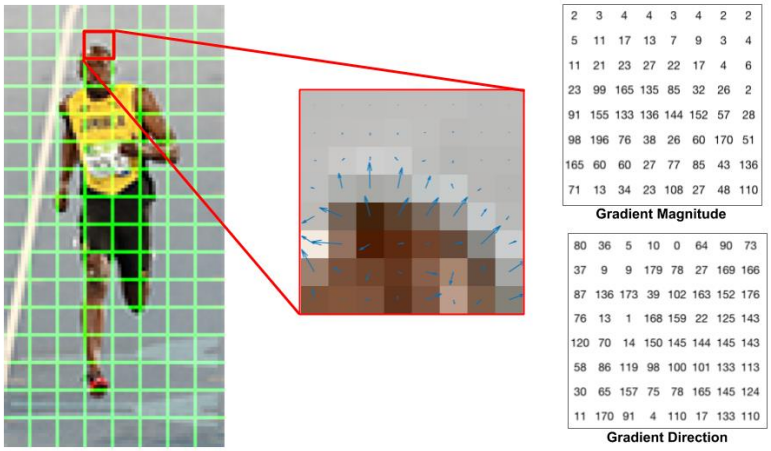
g_y - komponenta gradijenta u y-smjeru (vertikalna komponenta)

g - ukupna veličina gradijenata (magnituda)

$\emptyset$ - smjer gradijenata (kut gradijenta)

3. Generiranje histograma orijentacija:

Slika se dijeli na male povezane regije zvane ćelije. Unutar svake ćelije, gradijenti piksela se grupiraju u orijentacijske binove, stvarajući histogram koji predstavlja distribuciju gradijentnih smjerova. Ovi histogrami sažimaju informacije o lokalnim oblicima i teksturama unutar ćelije



Slika 17: Generacija histograma

Izvor: https://learnopencv.com/histogram-of-oriented-gradients/

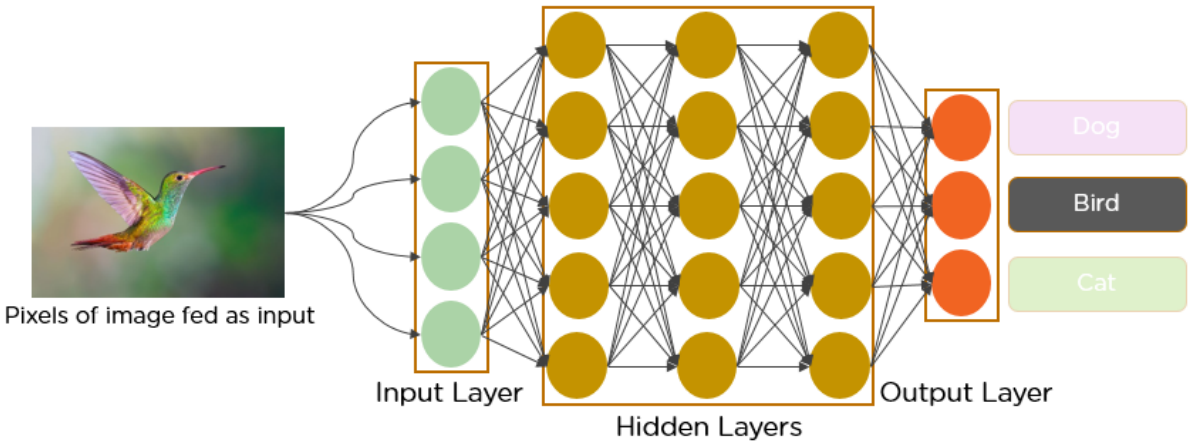
4. Normalizacija blokova:

Višestruke ćelije grupiraju se u blokove koji se preklapaju kako bi se poboljšala robusnost značajki. Histogramski podaci unutar blokova se normaliziraju, smanjujući osjetljivost na promjene osvjetljenja i kontrasta. Ova normalizacija pomaže u stvaranju konzistentnih značajki koje su otporne na varijacije u osvjetljenju.

3.2.2. CONVOLUTIONAL NEURAL NETWORKS ALGORITAM

Convolutional Neural Networks (CNN) algoritam je metoda dubokog učenja koja se koristi za analizu vizualnih podataka i prepoznat je kao jedan od najnaprednijih i najpreciznijih algoritama za detekciju i prepoznavanje lica. CNN se razlikuje od Haar Cascade i HOG algoritama po svojoj sposobnosti da automatski uči značajke iz podataka bez potrebe za ručnim određivanjem značajki. CNN koristi složene neuronske mreže koje omogućuju učenje složenih obrazaca i značajki iz velikih skupova podataka.

CNN započinje s ulaznom slikom koja se transformira u mapu značajki kroz niz konvolucijskih i pooling slojeva. Konvolucijski sloj primjenjuje skup filtera na ulaznu sliku, pri čemu svaki filter stvara mapu značajki koja naglašava specifične aspekte ulazne slike. Pooling sloj zatim smanjuje dimenzionalnost mape značajki, zadržavajući najvažnije informacije. Konačni izlaz mreže je predviđena klasa ili vjerojatnost za svaku klasu, ovisno o zadatku.



Slika 18: Arhitektura Convolutional Neural Network-a

Izvor: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

3.3. REALIZACIJA PREPOZNAVANJA LICA U PROJEKTU

U ovom poglavlju opisani su koraci i tehnologije korištene za realizaciju sustava prepoznavanja lica u projektu. Cijeli projekt realiziran je kroz dvije glavne Python skripte, koristeći biblioteke face_recognition, NumPy, OpenCV (cv2), json, time i RPi.GPIO.

Temelj cijelog sustava je biblioteka face_recognition, koja je izgrađena pomoću dlib biblioteke. Ona nudi snažne algoritme za obradu slike i analizu, čineći ovu kombinaciju idealnom za projekte prepoznavanja lica. Bibilioteka face_recognition je odabrana zbog svoje visoke točnosti i jednostavnosti korištenja. Ona pruža napredne alate za prepoznavanje lica, omogućujući pouzdano i efikasno prepoznavanje. Biblioteka face_recognition koristi algoritme temeljene na dubokom učenju, uključujući Histogram of Oriented Gradients i konvolucijske neuronske mreže za detekciju i prepoznavanje lica.

3.3.1. OPIS RADA KORIŠTENIH SKRIPTI

Prva Skripta – face_encodings.py: Implementira prepoznavanje lica koristeći face_recognition biblioteku, ona koristi HOG i CNN algoritme za detekciju i prepoznavanje lica. Skripta uključuje funkcionalnosti za učitavanje slika, treniranje modela i prepoznavanje lica na novim slikama.

Druga Skripta – face_recognition_system.py: Koristi biblioteku OpenCV za obradu ulaznog video streama sa priključene web kamere i primjenu modela face_recognition za

stvarnovremensko prepoznavanje lica. Također koristi RPi.GPIO za upravljanje servo motorom koji zaključava ili otključava sef.

3.3.1.1. Skripta - face_encodings.py

```python
import cv2
import face_recognition
import json
import numpy as np

# Funkcija za hvatanje slika, obradu i spremanje face encodings (kodiranja lica)
def capture_images(num_images):
    known_faces = {}  # Rječnik za spremanje imena i kodiranja lica

    for i in range(num_images):
        # Otvaranje kamere
        cap = cv2.VideoCapture(0)

        # Hvatanje slike
        ret, frame = cap.read()

        # Spremanje uhvaćene slike
        image_filename = f"captured_image_{i}.jpg"
        cv2.imwrite(image_filename, frame)
        print(f"Slika {i + 1} uspješno uhvaćena!")

        # Upit korisniku za unos imena osobe na slici
        name = input("Unesite ime osobe na slici: ")

        # Obrada slike i generiranje kodiranja lica
        face_encodings = face_recognition.face_encodings(frame)

        if len(face_encodings) > 0:
            # Pretvaranje NumPy niza u Python listu radi serijalizacije
            known_faces[name] = face_encodings[0].tolist()  # Pretpostavljamo da je na slici samo jedno
            print(f"Kodiranje lica generirano za {name}")
        else:
            print("Nijedno lice nije prepoznato na slici")

        # Oslobađanje kamere
        cap.release()

    # Spremanje poznatih lica i kodiranja u datoteku
    with open("known_faces.json", "w") as f:
        json.dump(known_faces, f)

    print("Sve slike su obrađene i kodiranja lica su spremljena u 'known_faces.json'")

# Glavni dio programa
if __name__ == "__main__":
    num_images = int(input("Unesite broj slika koje želite uhvatiti: "))
    capture_images(num_images)
```

Skripta počinje s učitavanjem potrebnih biblioteka: cv2 za rad s kamerom i obradom slika, face_recognition za detekciju i enkodiranje lica, json za serijalizaciju podataka i numpy za rad s numeričkim podacima.

face_encodings.py > capture_images

```
1   import cv2
2   import face_recognition
3   import json
4   import numpy as np
```

Unutar funkcije capture_images, skripta najprije inicijalizira prazan rječnik known_faces koji će pohraniti imena i enkodirane podatke lica. Za svaki od predviđenog broja slika, kamera se otvara pomoću cv2.VideoCapture(0), te se slika snima i pohranjuje pomoću cv2.imwrite funkcije.

```
6   # Funkcija za hvatanje slika, obradu i spremanje face encodings (kodiranja lica)
7   def capture_images(num_images):
8       known_faces = {}  # Rječnik za spremanje imena i kodiranja lica
9
10      for i in range(num_images):
11          # Otvaranje kamere
12          cap = cv2.VideoCapture(0)
13
14          # Hvatanje slike
15          ret, frame = cap.read()
16
17          # Spremanje uhvaćene slike
18          image_filename = f"captured_image_{i}.jpg"
19          cv2.imwrite(image_filename, frame)
20          print(f"Slika {i + 1} uspješno uhvaćena!")
```

Korisnik zatim mora unijeti ime osobe na slici, nakon čega se slika obrađuje korištenjem funkcije face_recognition.face_encodings koja generira enkodirane podatke lica. Ako je lice uspješno detektirano, enkodirani podaci se pohranjuju u rječnik known_faces, pri čemu se NumPy niz pretvara u Python listu radi serijalizacije.

```
    # Upit korisniku za unos imena osobe na slici
    name = input("Unesite ime osobe na slici: ")

    # Obrada slike i generiranje kodiranja lica
    face_encodings = face_recognition.face_encodings(frame)

    if len(face_encodings) > 0:
        # Pretvaranje NumPy niza u Python listu radi serijalizacije
        known_faces[name] = face_encodings[0].tolist()  # Pretpostavljamo da je na slici samo jedno lice
        print(f"Kodiranje lica generirano za {name}")
    else:
        print("Nijedno lice nije prepoznato na slici")

    # Oslobađanje kamere
    cap.release()
```

Na kraju, svi prikupljeni podaci se pohranjuju u JSON datoteku known_faces.json pomoću funkcije json.dump. Ova datoteka omogućuje jednostavnu pohranu i pristup biometrijskim podacima za kasniju upotrebu u sustavu prepoznavanja lica.

```
# Spremanje poznatih lica i kodiranja u datoteku
with open("known_faces.json", "w") as f:
    json.dump(known_faces, f)

print("Sve slike su obrađene i kodiranja lica su spremljena u 'known_faces.json'")
```

3.3.1.2. Skripta - face_recognition_system.py

```python
import cv2
import face_recognition
import json
import numpy as np
import RPi.GPIO as GPIO
import time

# Postavljanje GPIO za servo motor
servo_pin = 11
GPIO.setmode(GPIO.BOARD)
GPIO.setup(servo_pin, GPIO.OUT)
pwm = GPIO.PWM(servo_pin, 50)
pwm.start(0)

# Funkcija za postavljanje kuta servo motora
def set_servo_angle(angle):
    duty = (angle / 18) + 2
    GPIO.output(servo_pin, True)
    pwm.ChangeDutyCycle(duty)
    time.sleep(0.5)
    GPIO.output(servo_pin, False)
    pwm.ChangeDutyCycle(0)

# Funkcija za učitavanje poznatih kodiranja lica iz JSON datoteke
def load_known_faces():
    with open("known_faces.json", "r") as f:
        known_faces = json.load(f)

    # Pretvaranje kodiranja lica iz liste u NumPy niz
    for name, encoding in known_faces.items():
        known_faces[name] = np.array(encoding)

    return known_faces

# Funkcija za prepoznavanje lica
def detect_faces(known_faces):
    # Inicijalizacija web kamere
    cap = cv2.VideoCapture(0)
    last_servo_angle = None  # Praćenje posljednjeg kuta servo motora
    last_known_face_time = None  # Praćenje vremena posljednjeg prepoznavanja lica

    # Početno postavljanje servo motora u zatvoreni položaj (35 stupnjeva)
    set_servo_angle(35)
    last_servo_angle = 35

```

```
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Smanjivanje veličine slike radi brže obrade
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

        # Pretvaranje BGR slike u RGB (potrebno za face_recognition biblioteku)
        rgb_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)

        # Pronalaženje svih lokacija lica i kodiranja lica na trenutnoj slici
        face_locations = face_recognition.face_locations(rgb_frame)
        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

        if face_encodings:
            face_found = False
            for face_encoding, face_location in zip(face_encodings, face_locations):
                # Uspoređivanje kodiranja lica s poznatim licima
                matches = face_recognition.compare_faces(list(known_faces.values()), face_encoding)
                name = "Unknown"
                color = (0, 0, 255)  # Crvena za nepoznata lica
                servo_angle = 35

                # Provjera podudaranja s poznatim licima
                if True in matches:
                    face_found = True
                    match_index = matches.index(True)
                    name = list(known_faces.keys())[match_index]
                    color = (0, 255, 0)  # Zelena za poznata lica
                    servo_angle = 135
                    last_known_face_time = time.time()  # Ažuriranje vremena posljednjeg prepoznavanja

                # Pomicanje servo motora ovisno o rezultatu prepoznavanja lica
                if last_servo_angle != servo_angle:
                    set_servo_angle(servo_angle)
                    last_servo_angle = servo_angle

                # Crtanje pravokutnika i oznake oko lica
                top, right, bottom, left = face_location
                cv2.rectangle(frame, (left*4, top*4), (right*4, bottom*4), color, 2)
                cv2.putText(frame, name, (left*4, bottom*4 + 20), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 2

            # Ako nijedno poznato lice nije prepoznato, pomakni servo na 35 stupnjeva
            if not face_found and last_known_face_time and (time.time() - last_known_face_time > 5):
                if last_servo_angle != 35:
                    set_servo_angle(35)
                    last_servo_angle = 35
        else:
            # Ako nema prepoznatih lica, provjeri je li prošlo više od 5 sekundi od posljednjeg prepozr
            if last_known_face_time and (time.time() - last_known_face_time > 5):
                if last_servo_angle != 35:
                    set_servo_angle(35)
                    last_servo_angle = 35

        # Prikaz slike s prepoznatim licima
        cv2.imshow('Facial Recognition', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
    pwm.stop()
    GPIO.cleanup()

if __name__ == "__main__":
    known_faces = load_known_faces()  # Učitavanje poznatih kodiranja lica
    detect_faces(known_faces)  # Pokretanje prepoznavanja lica
```

Na početku skripte uvoze se potrebne biblioteke: cv2 za rad s kamerom i obradom slika, face_recognition za prepoznavanje lica, json za serijalizaciju podataka, numpy za rad s numeričkim podacima i RPi.GPIO za kontrolu GPIO pinova na Raspberry Pi-u.

facial_recognition_system.py > ...

```
1   import cv2
2   import face_recognition
3   import json
4   import numpy as np
5   import RPi.GPIO as GPIO
6   import time
7
8   # Postavljanje GPIO za servo motor
9   servo_pin = 11
10  GPIO.setmode(GPIO.BOARD)
11  GPIO.setup(servo_pin, GPIO.OUT)
12  pwm = GPIO.PWM(servo_pin, 50)
13  pwm.start(0)
```

Funkcija set_servo_angle koristi se za kontrolu pozicije servo motora pomoću PWM signala. Kut servo motora izračunava se prema formuli (angle / 18) + 2 i postavlja pomoću biblioteke RPi.GPIO. Ova funkcija omogućuje precizno postavljanje kuta servo motora, što je ključno za pouzdanu kontrolu mehaničkog sustava.

```
15  # Funkcija za postavljanje kuta servo motora
16  def set_servo_angle(angle):
17      duty = (angle / 18) + 2
18      GPIO.output(servo_pin, True)
19      pwm.ChangeDutyCycle(duty)
20      time.sleep(0.5)
21      GPIO.output(servo_pin, False)
22      pwm.ChangeDutyCycle(0)
```

Funkcija load_known_faces učitava biometrijske podatke iz JSON datoteke i pretvara ih u NumPy nizove radi lakše obrade. Ova funkcija omogućuje pristup prethodno pohranjenim enkodiranim podacima lica, što je ključno za proces prepoznavanja lica.

```
24  # Funkcija za učitavanje poznatih kodiranja lica iz JSON datoteke
25  def load_known_faces():
26      with open("known_faces.json", "r") as f:
27          known_faces = json.load(f)
28
29      # Pretvaranje kodiranja lica iz liste u NumPy niz
30      for name, encoding in known_faces.items():
31          known_faces[name] = np.array(encoding)
32
33      return known_faces
```

Funkcija detect_faces koristi se za prepoznavanje lica u stvarnom vremenu pomoću web kamere. Skripta kontinuirano snima slike, smanjuje ih radi brže obrade, pretvara u RGB format i pronalazi lokacije lica te generira jedinstvene kodove (enkodiranja) za svako lice. Prepoznata lica uspoređuju se s poznatim licima i odgovarajuće akcije se izvršavaju, uključujući kontrolu servo motora. Ova funkcija omogućuje dinamičko prepoznavanje lica i odgovarajuću reakciju sustava.

```
35  # Funkcija za prepoznavanje lica
36  def detect_faces(known_faces):
37      # Inicijalizacija web kamere
38      cap = cv2.VideoCapture(0)
39      last_servo_angle = None  # Praćenje posljednjeg kuta servo motora
40      last_known_face_time = None  # Praćenje vremena posljednjeg prepoznavanja lica
41
42      # Početno postavljanje servo motora u zatvoreni položaj (35 stupnjeva)
43      set_servo_angle(35)
44      last_servo_angle = 35
45
46      while True:
47          ret, frame = cap.read()
48          if not ret:
49              break
50
51          # Smanjivanje veličine slike radi brže obrade
52          small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
53
54          # Pretvaranje BGR slike u RGB (potrebno za face_recognition biblioteku)
55          rgb_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
56
57          # Pronalaženje svih lokacija lica i kodiranja lica na trenutnoj slici
58          face_locations = face_recognition.face_locations(rgb_frame)
59          face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)
```

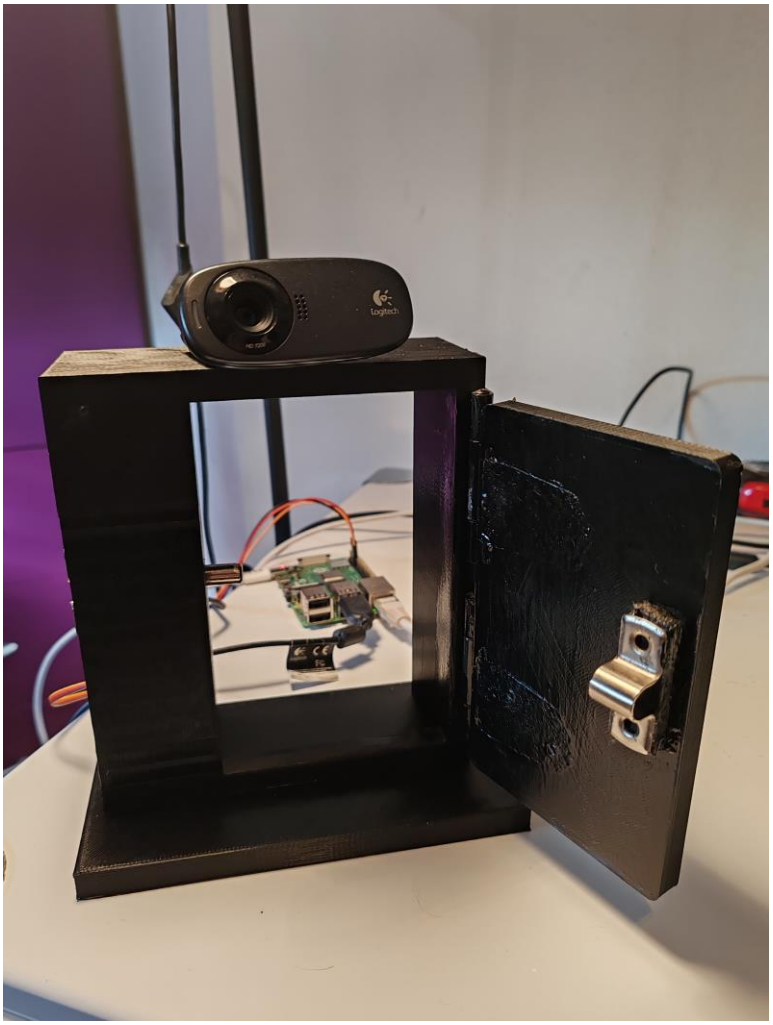
Unutar funkcije detect_faces, prepoznata enkodirana lica uspoređuju se s poznatim licima pomoću funkcije face_recognition.compare_faces. Ako je lice prepoznato, ime osobe se prikazuje na ekranu ispod zelenog pravokutnika koji prikazuje podudaranje, a servo motor se pomiče u položaj za otključavanje. Ako lice nije prepoznato ili nije detektirano dulje vrijeme, servo motor se vraća u početni položaj. Ova funkcionalnost omogućuje dinamičku kontrolu pristupa na temelju prepoznatog lica.

```
        if face_encodings:
            face_found = False
            for face_encoding, face_location in zip(face_encodings, face_locations):
                # Uspoređivanje kodiranja lica s poznatim licima
                matches = face_recognition.compare_faces(list(known_faces.values()), face_encoding)
                name = "Unknown"
                color = (0, 0, 255)  # Crvena za nepoznata lica
                servo_angle = 35

                # Provjera podudaranja s poznatim licima
                if True in matches:
                    face_found = True
                    match_index = matches.index(True)
                    name = list(known_faces.keys())[match_index]
                    color = (0, 255, 0)  # Zelena za poznata lica
                    servo_angle = 135
                    last_known_face_time = time.time()  # Ažuriranje vremena posljednjeg prepoznavanja lica

                # Pomicanje servo motora ovisno o rezultatu prepoznavanja lica
                if last_servo_angle != servo_angle:
                    set_servo_angle(servo_angle)
                    last_servo_angle = servo_angle

                # Crtanje pravokutnika i oznake oko lica
                top, right, bottom, left = face_location
                cv2.rectangle(frame, (left*4, top*4), (right*4, bottom*4), color, 2)
                cv2.putText(frame, name, (left*4, bottom*4 + 20), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 255), 1)

            # Ako nijedno poznato lice nije prepoznato, pomakni servo na 35 stupnjeva
            if not face_found and last_known_face_time and (time.time() - last_known_face_time > 5):
                if last_servo_angle != 35:
                    set_servo_angle(35)
                    last_servo_angle = 35
        else:
            # Ako nema prepoznatih lica, provjeri je li prošlo više od 5 sekundi od posljednjeg prepoznavanja
            if last_known_face_time and (time.time() - last_known_face_time > 5):
                if last_servo_angle != 35:
                    set_servo_angle(35)
                    last_servo_angle = 35
```

4. IZRADA PROJEKTA

Za realizaciju ovog projekta bilo je potrebno znanje iz nekoliko tehničkih područja, uključujući mikroračunala, senzore, aktuatore, 3D modeliranje i ispis te osnovno programiranje. Projekt obuhvaća dizajn, razvoj i implementaciju sustava za prepoznavanje lica za otključavanje vrata, a završeni projekt prikazan je na slici 13 u prilogu.

Dizajn vrata izveden je korištenjem edukacijske verzije Fusion 360 programa, dok je ispis modela realiziran pomoću 3D pisača na fakultetu. Za mehanizam vrata korišteni su šarke i zasun, čime je osigurana stabilnost i funkcionalnost sustava. Raspberry Pi 4 Model B, korišten u ovom projektu, bio je potpuno nov te je zahtijevao inicijalnu konfiguraciju i instalaciju operativnog sustava.



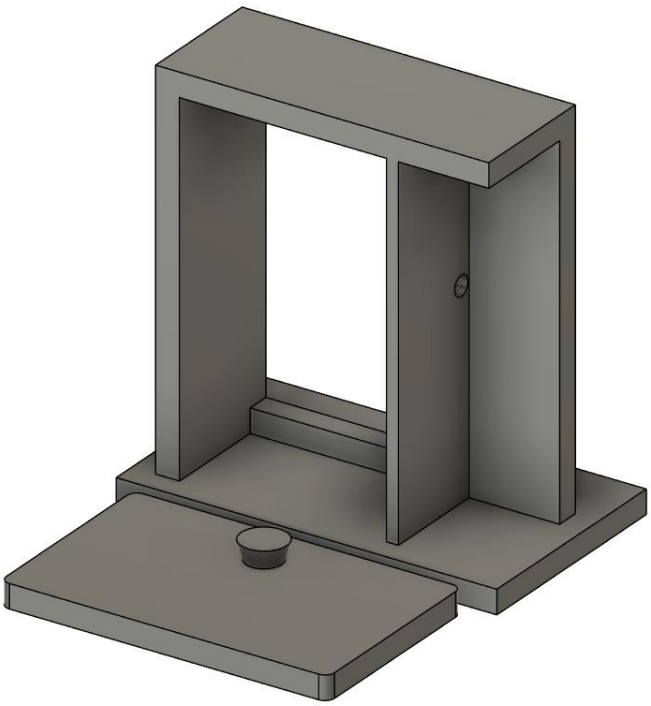
Slika 19: Izgled sustava

Izvor: autor

4.1. 3D MODELIRANJE I PRINTANJE

Za izradu 3D modela ovoga rada korišten je program Fusion 360. Fusion 360 je napredan alat za 3D modeliranje, dizajn i inženjering razvijen od strane Autodeska. Odabran je za ovaj projekt zbog njegove sveobuhvatnosti, preciznosti i fleksibilnosti. Fusion 360 integrira CAD, CAM i CAE funkcionalnosti, što omogućuje jednostavan prijelaz iz faze dizajna u fazu proizvodnje. Također nudi opcije za optimizaciju modela za 3D ispis, podržava suradnju u stvarnom vremenu i verziranje datoteka, što je vrlo korisno za timski rad. Dostupnost edukacijske verzije omogućila je korištenje ovog profesionalnog alata bez dodatnih troškova. Korištenjem Fusion 360, postignuta je visoka kvaliteta i preciznost dizajna vrata, što je bilo ključno za uspješnu implementaciju sustava za prepoznavanje lica.

Za izradu 3D modela bilo je ključno obratiti pažnju na nekoliko važnih faktora. Prvo, dimenzije 3D pisača morale su se uzeti u obzir kako bi se osiguralo da dizajn vrata ne prelazi granice podloge 3D pisača. Također, bilo je potrebno precizno definirati dimenzije samih vrata, kao i prostor za ugradnju zasuna i servo motora. Na vrhu vrata trebalo je osigurati stabilno mjesto za kameru, kako bi sustav za prepoznavanje lica mogao ispravno funkcionirati. Sva ta ograničenja i zahtjevi utjecali su na konačni dizajn vrata, omogućujući da se svi elementi pravilno integriraju i funkcioniraju u cjelini.



Slika 20: Model vrata u Fusion 360-u

Izvor: autor

Za 3D ispis korišten je 3D pisač na fakultetu. Projekt je ispisivan ukupno tri puta: jednom je ispisivan okvir za vrata, a drugi put postolje i sama vrata s ručkom. Ukupno vrijeme ispisa iznosilo je između 15 i 16 sati. Tijekom ispisivanja bilo je potrebno obratiti pažnju na postavke samog pisača, kao što su infil (ispuna), tip ispune, brzina ispisivanja i debljina slojeva. Posebna pažnja posvećena je postavljanju podloge u ravninu kako bi se izbjegle potencijalne greške pri ispisu. Postavljanje podloge u ravninu može se postići ručnim podešavanjem visine podloge pomoću nivelacijskih vijaka i korištenjem kalibracijskog papira za postizanje pravilnog razmaka između mlaznice i podloge. Također je bilo važno održavati odgovarajuću temperaturu mlaznice i podloge kako bi se osigurala dobra adhezija prvog sloja. 3D pisač je imao mogućnost Wi-Fi povezivanja, što je omogućilo jednostavan i brz prijenos 3D modela s prijenosnog računala na pisač. Tijekom ispisivanja, pisač je radio bez nadzora, što je omogućilo učinkovitije korištenje vremena i resursa. Pravilno postavljanje parametara i održavanje pisača bili su ključni za osiguranje kvalitete i preciznosti ispisanih dijelova.

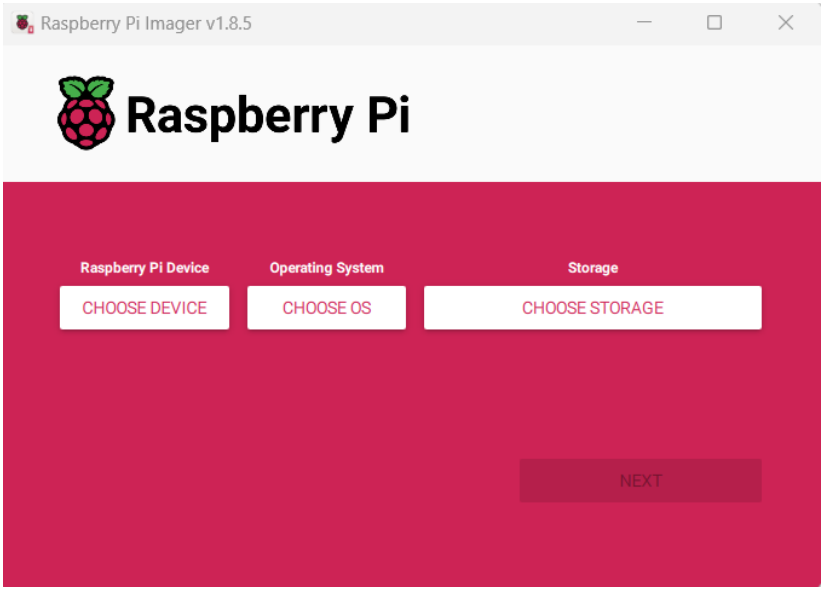
Slika 21: Korišteni 3D pisač na fakultetu

Izvor: autor

4.2. PRIPREMA RASPBERRY PI-A

Mikroračunalo Raspberry Pi nema unaprijed instaliran operativni sustav pa se prije operativne uporabe isti mora ugraditi. Od više ponuđenih inačica operativnih sustava odabran je Raspbian OS stoga je prvi korak uvijek instalacija operativnog sustava na uređaj. Iako se mogu koristiti i drugi operativni sustavi, u ovom projektu korišten je Raspbian OS. Instalacija se provodi korištenjem alata Raspberry Pi Imager koji se instalira na računalo.

Nakon preuzimanja i pokretanja Raspberry Pi Imagera, korisnik ima tri opcije: odabir modela Raspberry Pi-a koji će se koristiti, odabir operativnog sustava kojeg se želi instalirati te odabir SD kartice na koju će sustav biti instaliran. Prije instalacije, SD karticu je potrebno formatirati, što se može obaviti izravno putem Imagera. Nakon što je operativni sustav instaliran na SD karticu, kartica se umeće u Raspberry Pi, koji se zatim povezuje na napajanje čime postaje spreman za korištenje.



Slika 22: Raspberry Pi Imager

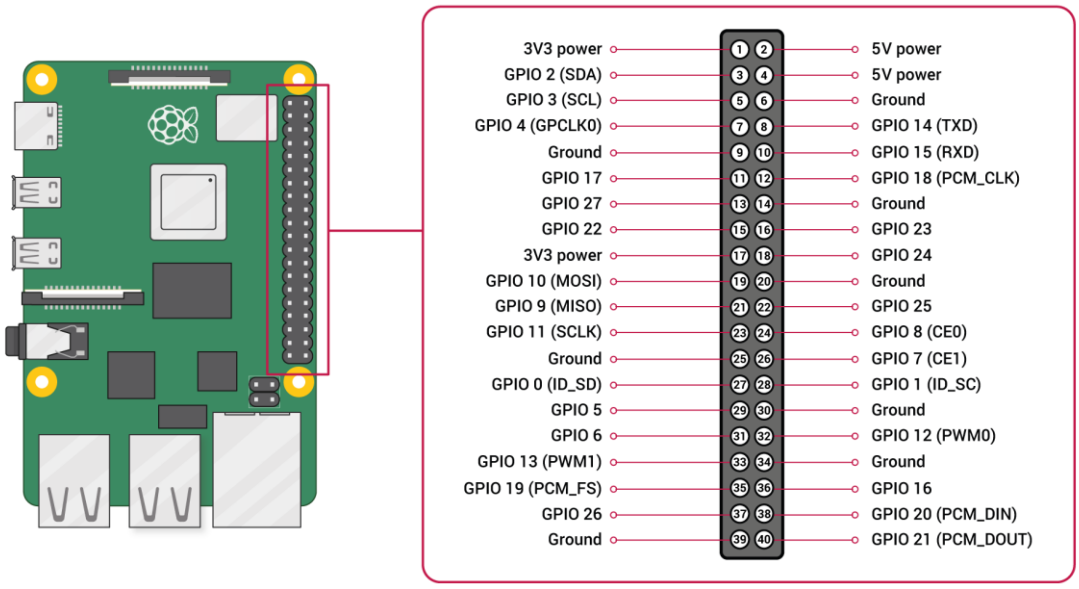
Izvor: autor

Raspberry Pi omogućuje izravno povezivanje periferija poput tipkovnice, miša i web kamere putem USB priključaka. Međutim, u ovom projektu odlučeno je povezati Raspberry Pi s prijenosnim računalom korištenjem Ethernet kabela. Ovaj način povezivanja omogućava korištenje VNC (Virtual Network Computing) programa za daljinsko upravljanje Raspberry Pi-em.

VNC je alat koji omogućava daljinski pristup i kontrolu drugog računala preko mreže. Kako bi se koristio VNC za upravljanje Raspberry Pi-em, potrebno je instalirati VNC server na Raspberry Pi mikroračunalu te VNC viewer na prijenosnom računalu. Nakon instalacije,

korisnik se može spojiti na Raspberry Pi unosom adrese „raspberrypi.local“ u VNC viewer program. Nakon toga, korisnik treba unijeti korisničko ime i lozinku za Raspberry Pi, čime dobiva pristup radnoj površini Raspberry Pi-a. Ovaj pristup omogućava korištenje monitora, miša i tipkovnice prijenosnog računala za kontroliranje Raspberry Pi uređaja, što olakšava rad i upravljanje projektom iz udobnosti vlastitog radnog prostora.

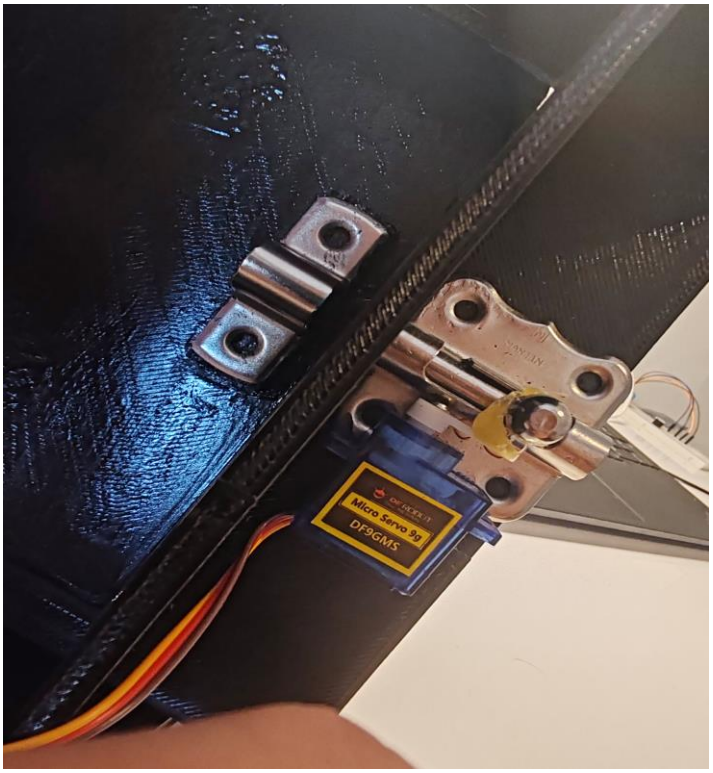
Kada je Raspberry Pi postavljen i spreman za upotrebu, spojeni su kamera i servo motor. Kamera je povezana putem USB priključka, dok je servo motor spojen na GPIO pinove Raspberry Pi-a. Servo motor ima tri kabela za spajanje: napajanje, uzemljenje i signalni kabel. Prema pinoutu Raspberry Pi 4 Model B, napajanje se povezuje na 5V pin, uzemljenje na GND pin, a signalni kabel na GPIO 17 - pin 11.



Slika 23: Raspored GPIO pinova

Izvor: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

Pravilno povezivanje servo motora ključno je za funkcioniranje sustava. Signalni kabel na GPIO 17 - pin 11 koristi se za kontrolu kuta servo motora, omogućujući precizno upravljanje mehaničkim dijelovima vrata. Prilikom programiranja potrebno je pravilno definirati GPIO pinove u Python skriptama, uključujući inicijalizaciju i postavljanje načina rada za svaki pin.



Slika 24 i 25: Spojeni servo motor i sustav u radu

Izvor: autor

5. ZAKLJUČAK

Ovaj završni rad prikazuje razvoj sustava sigurnog otključavanja vrata temeljenog na prepoznavanju lica korištenjem Raspberry Pi 4 računala. Implementacija sustava obuhvaća korištenje Python programskog jezika i relevantnih biblioteka poput OpenCV-a što omogućuje preciznu i pouzdanu obradu slike i prepoznavanje lica. Kroz testiranja, sustav je pokazao visok stupanj učinkovitosti i pouzdanosti, čak i u različitim uvjetima osvjetljenja. Primjenom naprednih algoritama za detekciju i prepoznavanje lica, kao što su Haar Cascade, HOG i CNN, postignuta je visoka razina sigurnosti i brzine prepoznavanja. Ključna komponenta ovog rada je integracija hardverskih i softverskih elemenata. Raspberry Pi 4, kao centralni dio sustava, omogućuje fleksibilnost i pristupačnost, dok je 3D modeliranje i ispisivanje vrata omogućilo stvaranje prilagođenog i funkcionalnog rješenja. Rezultati rada ukazuju na veliki potencijal sustava za široku primjenu u kućnoj sigurnosti. Daljnja istraživanja i razvoj mogu dodatno unaprijediti točnost i sigurnost ovih sustava, omogućujući njihovu primjenu u različitim svakodnevnim situacijama. Zaključno, sustav predstavljen u ovom radu potvrđuje vrijednost i učinkovitost prepoznavanja lica kao modernog sigurnosnog rješenja. Ova tehnologija nudi praktičnost, sigurnost i jednostavnost korištenja, čime doprinosi unapređenju sigurnosnih normi u svakodnevnom životu.

LITERATURA

[1] Innovatrics 2024, Facial recognition technology, Innovatrics, online: https://www.innovatrics.com/facial-recognition-technology/ (27.6.2024.)

[2] Raspberry Pi Foundation 2024, Documentation - Raspberry Pi, Raspberry Pi Foundation, online: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html (27.6.2024.)

[3] OpenCV 2024, Face recognition with OpenCV, OpenCV, online: https://docs.opencv.org/4.x/da/d60/tutorial_face_main.html (27.6.2024.)

[4] Opensource.com 2024, Resources - Raspberry Pi, Opensource.com, online: https://opensource.com/resources/raspberry-pi (27.6.2024.)

[5] Pimylifeup 2024, Raspberry Pi versions, Pimylifeup, online: https://pimylifeup.com/raspberry-pi-versions/ (27.6.2024.)

[6] Thulluri, K.V., Kanchana, C.S. & Vijayalakshmi, M. 2019, Face recognition based door unlocking system using Raspberry Pi, International Journal of Advance Research, Ideas and Innovations in Technology, vol. 5, no. 2, pp. 1320-1324, online: https://www.ijariit.com (27.6.2024.)

[7] Kaspersky 2024, What is facial recognition?, Kaspersky, online: https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition (27.6.2024.)

[8] Visage Technologies 2024, Benefits of face recognition, Visage Technologies, online: https://visagetechnologies.com/benefits-of-face-recognition/ (27.6.2024.)

[9] Medium 2024, Haar Cascades Explained, Medium, online: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d (27.6.2024.)

[10] Learn OpenCV 2024, Histogram of Oriented Gradients, Learn OpenCV, online: https://learnopencv.com/histogram-of-oriented-gradients/ (27.6.2024.)

[11] Medium 2024, A Gentle Introduction into the Histogram of Oriented Gradients, Medium, online: https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa (27.6.2024.)

[12] LinkedIn 2024, What is Convolutional Neural Network (CNN) in Deep Learning?, LinkedIn, online: https://www.linkedin.com/pulse/what-convolutional-neural-network-cnn-deep-learning-nafiz-shahriar/ (27.6.2024.)

KAZALO KRATICA

Kratica	Puni naziv na engleskom jeziku	Tumačenje na hrvatskom jeziku
3D	Three-Dimensional	Trodimenzionalno
4K	4K Resolution	4K Rezolucija
CAD	Computer-Aided Design	Računalno potpomognuto projektiranje
CAM	Computer-Aided Manufacturing	Računalno potpomognuta proizvodnja
CAE	Computer-Aided Engineering	Računalno potpomognuto inženjerstvo
CNN	Convolutional Neural Network	Konvolucijska neuronska mreža
CPU	Central Processing Unit	Središnja procesorska jedinica
DIY	Do It Yourself	Uradi sam
GB	Gigabyte	Gigabajt
GND	Ground	Uzemljenje
GPIO	General Purpose Input/Output	Ulaz/Izlaz opće namjene
GPU	Graphics Processing Unit	Grafička procesorska jedinica
HOG	Histogram of Oriented Gradients	Histogram orijentiranih gradijenata
HDMI	High-Definition Multimedia Interface	Višemedijsko sučelje visoke definicije
JSON	JavaScript Object Notation	Notacija za objekte u JavaScriptu
OS	Operating System	Operacijski sustav
PWM	Pulse Width Modulation	Modulacija širine impulsa
RPi	Raspberry Pi	Raspberry Pi
RAM	Random Access Memory	Memorija s izravnim pristupom
RGB	Red Green Blue	Crvena Zelena Plava (model boja)
SAD	System Application Development	Razvoj sustava aplikacija
SBC	Single Board Computer	Jednokratno računalo
SD	Secure Digital	Sigurna digitalna (kartica)
USB	Universal Serial Bus	Univerzalna serijska sabirnica
VNC	Virtual Network Computing	Virtualno mrežno računanje
ICCVPR	International Conference on Computer Vision and Pattern Recognition	Međunarodna konferencija o računalnom vidu i prepoznavanju uzoraka

POPIS SHEMA