Sigurnosna vrata s antropomorfnim svojstvom prepoznavanja

Herak, Marin

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Marin Herak

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Mladen Crneković

Student:

Marin Herak

Zagreb, 2023.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svojoj obitelji i prijateljima na potpori i razumijevanju, svim profesorima koji su mi svojim trudom i entuzijazmom prenijeli svoje znanje što je veoma pripomoglo pri izradi ovog diplomskog rada te svom mentoru prof. dr.sc. Mladenu Crnekoviću koji mi je dao ideju za sam rad i pomogao savjetima u savladavanju prepreka za ostvarivanje iste u praksi.

Marin Herak

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 23 - 6 / 1**	
Ur.broj: **15 - 23 -**	

DIPLOMSKI ZADATAK

Student: **Marin Herak** JMBAG: 0035220795

Naslov rada na hrvatskom jeziku: **Sigurnosna vrata s antropomorfnim svojstvom prepoznavanja**

Naslov rada na engleskom jeziku: **Security Door with an Anthropomorphic Recognition Feature**

Opis zadatka:

U cilju sprječavanja neovlaštenog ulaza u određene prostore sve više ustanova i tvrtki ugrađuje sigurnosna vrata koja za prolaz traže neku vrstu identifikacije. Do sada najčešći beskontaktni način, pomoću RFID tehnologije i kartica, ima nekoliko nedostataka. To su: mogućnost gubitka ključa-identifikatora, mogućnost kopiranja identifikatora, potreba za vođenjem evidencije o vezama između identifikatora i osobe, čuvanje identifikatora itd.

Svatko od nas uvijek nosi svoje jedinstvene identifikatore, lice i otisak prsta, a oni nemaju prethodno navedene nedostatke.

Potrebno je projektirati sigurnosna vrata koja će omogućiti prolaz osobama prema prepoznatom licu i otiskom prsta. Također je potrebno osigurati glasovnu komunikaciju prema čovjeku.

U radu je potrebno:

- konstruirati mehanizam sigurnosnih vrata,
- odabrati kameru i čitač otisaka prstiju, aktuatore i upravljački kontroler,
- napisati upravljački program,
- predložiti glasovno sučelje prema korisniku i način programiranja,
- procijeniti vrijednost uređaja.

Potrebno je navesti korištenu literaturu i ostale izvore informacija te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
28. rujna 2023.	30. studenoga 2023.	4. – 8 . prosinca 2023.

Zadatak zadao: M. Crneković

Prof.dr.sc. Mladen Crneković

Predsjednik Povjerenstva:

Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

SB-0000-00 Safe box

POPIS OZNAKA

Oznaka	Jedinica	Opis
Smjer gradijenta	θ	Označava smjer gradijenta piksela
Veličina gradijenta	g	Označava veličinu gradijenta piksela
Memorija	GB	Prostor pohrane
Napon	V	Razlika električnog potencijala
Struja	A	Električna struja
Temperatura	°C	Zagrijanost, toplina
Masa	kg	Količina tvari u tijelu
Snaga	W	Brzina obavljanja rada

SAŽETAK

U sklopu diplomskog rada prikazana je važnost implementacije tehnologija prepoznavanja lica, otiska prsta i glasovne komunikacije u kontekstu otvaranja sigurnosnih vrata. Navedene su već postojeće metode i tehnike navedenih tehnologija kao i već postojeći proizvodi. Jedan od tih je i NEC koji je globalni lider u području biometrijske autentifikacije, s šest originalnih tehnologija biometrijske autentifikacije. HOG algoritam koji se koristi za prepoznavanje lica u ovom diplomskom radu je detaljnije opisan jer je važan dio ovog rada. Najvažnija komponenta ovog rada je, u današnje vrijeme, jako popularno računalo Raspberry Pi koje spojeno sa navedenim komponentama čini funkcionalnu cjelinu. Nakon programiranja, teorija je potvrđena eksperimentom i kako sam sklop radi. On radi na način da za otvaranje vrata potrebno ispravno lice i otisak prsta inače ulaz nije dopušten i sve je to popraćeno glasovnom komunikacijom prema korisniku. Ovim radom je stvoren temelj za daljnji razvoj sličnih sustava i pristupa unutar područja kontrole pristupa i sigurnosti prostora.

Ključne riječi: prepoznavanje lica, prepoznavanje otiska prsta, glasovna komunikacija, sigurnosna vrata, NEC, HOG algoritam, Raspberry Pi.

SUMMARY

The importance of implementing facial recognition, fingerprint, and voice communication technologies in the context of opening secure doors is presented within the framework of the master's thesis. Existing methods and techniques of these technologies, as well as already available products, are discussed. One such product is NEC, a global leader in biometric authentication, with six original biometric authentication technologies. The HOG algorithm used for facial recognition in this thesis is detailed as it is a crucial part of the work. The most significant component of this thesis is the widely popular Raspberry Pi computer, which, when connected with the mentioned components, forms a functional unit. After programming, the theory is confirmed through experimentation, demonstrating the functioning of the system. It operates by requiring the correct face and fingerprint for door access; otherwise, entry is not permitted, accompanied by voice communication with the user. This work establishes a foundation for further development of similar systems and approaches in the field of access control and space security.

Key words: facial recognition, fingerprint recognition, voice communication, security doors, NEC, HOG algorithm, Raspberry Pi.

1. UVOD

U današnjem digitalnom dobu, sigurnost i praktičnost su postali ključni elementi svakodnevnog života. Kako bi se udovoljilo ovim potrebama, razvijaju se različite tehnologije i sustavi za kontrolu pristupa i identifikaciju korisnika. Tradicionalni pristupi, poput beskontaktnih tehnologija temeljenih na RFID karticama, iako široko korišteni, nose sa sobom određene nedostatke poput gubitka ključeva-identifikatora, mogućnosti kopiranja, te potrebe za vođenjem preciznih evidencija. U tom kontekstu, oslanjanje na unikatne osobne karakteristike, kao što su lice i otisak prsta, izdvaja se kao potencijalno učinkovito rješenje.

Ovaj diplomski rad fokusira se na projektiranje inovativnih sigurnosnih vrata koja integriraju antropomorfna svojstva prepoznavanja, koristeći lice i otisak prsta kao autentične identifikatore prolaza. Cilj je stvoriti sustav koji nadmašuje tradicionalne metode, eliminirajući probleme vezane uz gubitak, kopiranje i vođenje evidencije identifikatora.

U procesu realizacije ovog projekta, istražit će se ključni elementi, uključujući konstrukciju mehanizma sigurnosnih vrata, odabir odgovarajuće kamere i čitača otisaka prstiju, aktuatora te upravljačkog kontrolera. Dodatno, razvit će se upravljački program koji će omogućiti integraciju navedenih tehnologija u funkcionalan sustav. Važan aspekt ovog rada bit će i predloženo glasovno sučelje prema korisniku, zajedno s metodologijom programiranja istog, kako bi se osigurala potpuna sigurnost i praktičnost u korištenju.

Kroz sve ove korake, razmotrit će se vrijednost predloženog uređaja u kontekstu sigurnosti, praktičnosti i upotrebljivosti. Kroz literaturni pregled i analizu relevantnih izvora informacija, rad će osigurati čvrstu teorijsku osnovu. Eventualna dobivena pomoć i suradnja s drugima bit će transparentno dokumentirane kako bi se osigurala integritet i autentičnost rezultata.

2. PREGLED LITERATURE

Biometrijska autentifikacijska tehnologija postala je ključni faktor u određivanju autentičnosti i zaštiti privatnosti. NEC je globalni lider u području biometrijske autentifikacije, s šest originalnih tehnologija biometrijske autentifikacije: prepoznavanje lica, prepoznavanje šarenice, prepoznavanje otiska prsta/dlanova, prepoznavanje glasa i prepoznavanje akustike uha [3].

Windows Hello tvrtke Microsoft je značajka biometrijske autentifikacije koja omogućuje korisnicima da se prijave na svoje uređaje sa sustavom Windows 11 pomoću PIN-a, prepoznavanja lica ili otiska prsta.

FaceKey Corporation je tvrtka koja proizvodi i instalira sustave za kontrolu pristupa koji koriste prepoznavanje lica i otiska prsta kako bi identificirali osoblje i poboljšali sigurnost [4].

Danas pametni telefoni koriste prepoznavanje otiska prsta i šarenica najčešće, ali novi modeli Samsung, Huawei i iPhone mobitela počeli su sa 3D prepoznavanjem lica. Neki od modela koji već imaju su: Samsung Galaxy Z Fold5, S23, Huawei Mate 20 Pro, iPhone X. Također danas svaki pametni telefon ima ugrađenog glasovnog asistenta sa kojim se bez ikakvih problema može razgovarati o bilo čemu.

Slika 1. NEC autentifikacijske tehnologije [3]

2.1. Pregled postojećih metoda i tehnika prepoznavanja lica

2.1.1. Općenito o prepoznavanju lica

Prepoznavanje lica oduvijek je imalo važnu ulogu u društvenoj interakciji ljudi. Prepoznavanjem lica čovjek može utvrditi nečiji identitet i emocionalno stanje. Istraživanja T. Radford-a [6] su pokazala da čovjek može prepoznati do deset tisuća lica koje je upoznao tijekom života i može identificirati poznata lica u trenu čak i nakon dužeg razdoblja razdvojenosti. Osim toga, prilikom prepoznavanja čovjek može zanemariti otegotne okolnosti poput različitih osvjetljenja, izraza lica, kutova gledanja, promjene frizure ili brade, odnosno, brkova. Ljudska sposobnost vizualnog procesiranja lica je stoljećima fascinirala mnoge filozofe i znanstvenike.

Kroz povijest se pokušalo shvatiti na koji način čovjek prepoznaje osobe kako bi se mogućnost prepoznavanja lica mogla implementirati te kako bi mogla imati još širu primjenu od isključivo društvene interakcije. U starijim člancima koji govore o algoritmima za prepoznavanje lica ponajviše se govori o važnosti prepoznavanja lica u kriminalističkim istragama te o važnosti prepoznavanja nalazi li se lice na slici prilikom razvijanja fotografija (kako bi se obratila pažnja na boje na slici) [5]. Međutim, napretkom tehnologije i algoritama, pronađene su još mnoge primjene prepoznavanja lica.

Prema člancima [7] i [2] danas se prepoznavanje lica koristi u potrazi za nestalim osobama (aplikacija [8]). Ova primjena potaknuta je činjenicom da u Indiji svakoga sata nestane 14 djece te se, na osnovu usporedbe slike koju korisnici aplikacije uslikaju i slika u bazi, pronalaze nestali. Nadalje, široka primjena uočava se i u pitanjima sigurnosti - razni sustavi mogu na osnovu identifikacije osobe prema licu dati, odnosno, osporiti prava na pristup. Tako se u Kini razvija bankomat koji bi trebao raditi na osnovu prepoznavanja lica, te korisnici ne bi sa sobom trebali imati nikakve dokumente kako bi mogli podići novac. Osim toga, implementiraju se rješenja koja bi se trebala koristiti u obrazovnim ustanovama kako bi se lakše i vjerodostojnije pratili dolasci učenika (studenata) na predavanja i kako bi se osiguralo da osobe koje polažu ispite zaista jesu te osobe. Također, primjena se pronalazi i u identifikaciji ljudi kod izlazaka na izbore. Osim toga, društvene mreže također imaju ugrađenu mogućnost prepoznavanja lica i predlaganja osoba za označivanje na fotografijama koje korisnici postavljaju.

Naposlijetku, važno je spomenuti i mogućnost analiziranja video snimki u svrhu prepoznavanja osoba pri čemu računala mogu prepoznati oko 100 osoba u 5-6 sekundi, dok bi čovjeku za isti zadatak moglo trebati i cijeli dan. Prepoznavanje lica, kao što je do sada navedeno, ima tendenciju još šire primjene u budućnosti. Zbog toga se konstantno radi na razvoju novih rješenja koja bi omogućila što precizniju i pouzdaniju identifikaciju osoba na ovaj način.

2.1.2. Algoritmi za prepoznavanje lica

Većina algoritama za prepoznavanje lica napravljena je tako da koristi strojno učenje (eng. machine learning). Proces strojnog učenja svodi se na to da se računalu najprije daju podaci za učenje, u slučaju prepoznavanja lica, radi se o fotografijama lica. Potom se ti podaci procesiraju i pohranjuju u bazu podataka. U procesu testiranja, nastavlja se i proces učenja na način da se novo dobiveni podaci pohranjuju u bazu i samim time se baza proširuje i dobivaju se precizniji podaci.

Kod algoritama za prepoznavanje lica postoji nekoliko osnovnih koraka:

1. Izdvajanje lica:

   Algoritmi počinju proces prepoznavanja tako da izdvajaju lice iz okoline slike ili videozapisa. Ovaj korak uključuje tehnike poput detekcije rubova, segmentacije, ili korištenje predobučenih modela za prepoznavanje lica.

2. Određivanje karakterističnih točaka:

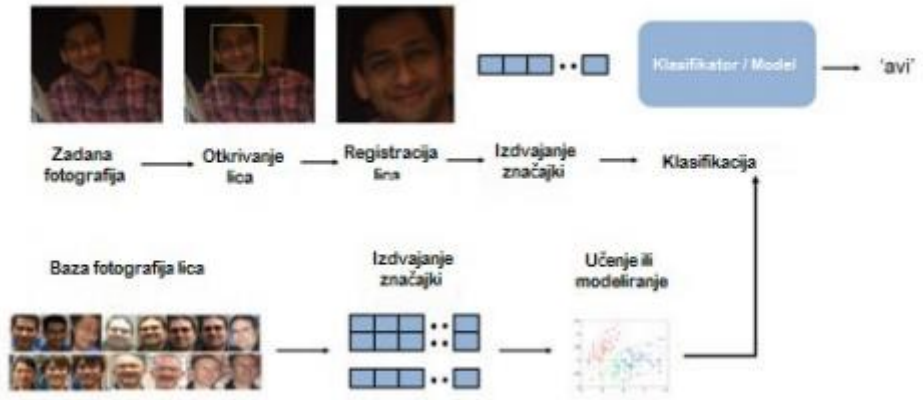
   Identifikacija ključnih točaka na licu, poput vrha nosa, očiju, usta, i obrva, pomaže stvaranju jedinstvenog "biometrijskog otiska" za svaku osobu. Ovaj korak često uključuje korištenje tehnika poput lokalizacije značajki i detekcije ključnih točaka.

3. Predstavljanje lica:

   Nakon izdvajanja lica i određivanja ključnih točaka, slijedi korak predstavljanja lica u obliku matematičkog modela. Ovo može uključivati vektorske reprezentacije, histogram orijentacija gradijenata (HOG), ili druge metode koje omogućavaju jednostavnu usporedbu.

4. Usporedba i Prepoznavanje:

   Konačni korak je usporedba predstavljanja lica s onima pohranjenima u bazi podataka ili modelu. Algoritmi koriste različite metode usporedbe, uključujući Euklidsku udaljenost, kosinusnu sličnost ili korištenje neuronskih mreža.



Slika 2. Koraci u računalnom prepoznavanju lica

Algoritme možemo grubo podijeliti na one koji koriste dvodimenzionalni prikaz te iz njega izdvajaju karakteristične točke i udaljenosti među njima, te one koji koriste trodimenzionalni prikaz, odnosno, trodimenzionalne modele za prepoznavanje. Osim navedenih, postoje još i metode koje se koriste video snimkama kako bi dobile što bolji prikaz, a samim time i veću preciznost i otpornost na najčešće probleme koji se javljaju prilikom računalnog prepoznavanja lica - različito osvjetljenje i različiti izrazi lica.

Postoje različite metode i algoritmi koji se koriste za prepoznavanje lica, a neki od najčešće korištenih su:

1. Eigenfaces:

   Ovaj pristup koristi analizu glavnih komponenti (PCA) za pretvaranje slika lica u skup linearnih nezavisnih varijabli, tzv. "eigenfaces". Usporedbom slika s ovim eigenfacesima omogućava se prepoznavanje lica.

2. Local Binary Patterns (LBP):

   LBP algoritmi fokusiraju se na lokalne teksture lica, identificirajući uzorke promjene intenziteta piksela. Ovi algoritmi su otporni na promjene osvjetljenja.

3. Viola-Jones:

   Ovaj kaskadni klasifikator koristi se za detekciju objekata, uključujući lica. Koristi se integralna slika za brzu evaluaciju velikog broja potencijalnih karakteristika te primjenjuje strojno učenje za klasifikaciju lica.

4. Deep Face i Deep Learning:

   S razvojem dubokog učenja (deep learning), posebice konvolucijskih neuronskih mreža (CNN), postignuti su značajni napretci u prepoznavanju lica. Algoritmi poput DeepFace, razvijenog od strane Facebooka, koriste duboke neuronske mreže za prepoznavanje i analizu lica.

5. Histogram of Oriented Gradients (HOG):

   Ovaj pristup analizira orijentacije gradijenata piksela kako bi identificirao oblik i konture lica. HOG je često korišten u kombinaciji s drugim metodama i kasnije će biti detaljnije objašnjen.

6. LBPH (Local Binary Pattern Histogram):

   Ovaj algoritam kombinira LBP s histogramima kako bi stvorio reprezentaciju lica. Posebno je robustan na promjene osvjetljenja i deformacije lica.

7. Fisherfaces:

   Sličan Eigenfaces algoritmu, Fisherfaces također koristi analizu glavnih komponenti. Razlika je u tome što se Fisherfaces fokusira na razlučivanje između različitih osoba umjesto na opću varijancu.

8. Dlib:

   Dlib je popularna biblioteka za strojno učenje koja sadrži alat za prepoznavanje lica temeljen na histogramu orijentiranih gradijenata (HOG) i strojnom učenju.

9. MTCNN (Multi-task Cascaded Convolutional Networks):

   MTCNN je algoritam za detekciju lica koji koristi kaskadu konvolucijskih mreža za prepoznavanje i lokalizaciju lica te karakteristika poput očiju, nosa i usta.

2.1.3. Senzori za prepoznavanje lica

Senzori za prepoznavanje lica koriste različite tehnologije kako bi snimali i analizirali karakteristike lica radi identifikacije ili verifikacije osoba. Ovdje su neke od glavnih vrsta senzora za prepoznavanje lica:

1. Optički senzori:
   Koriste kamere i svjetlosne senzore za snimanje 2D ili 3D slike lica. Ovisno o tehnologiji, optički senzori mogu detektirati oblike, konture i druge karakteristike lica.
2. Infracrveni senzori:
   Detektiraju infracrveno zračenje koje emitira lice. Infracrveni senzori mogu raditi i pri slabom osvjetljenju, čineći ih pogodnima za noćno prepoznavanje lica.
3. Točkasti senzori:
   Koriste laserske točke ili projekcije svjetlosti na lice kako bi stvorili 3D model lica. Ova tehnologija omogućuje preciznije prepoznavanje karakteristika lica.
4. Termalni senzori:
   Mjere temperaturne razlike na površini lica. Termalni senzori su otporni na promjene osvjetljenja i mogu prepoznati lica bez obzira na boju kože ili okolne uvjete.
5. Ultrazvučni senzori:
   Koriste ultrazvučne valove za stvaranje 3D slike lica. Ova tehnologija omogućuje dobivanje detaljnih informacija o reljefu lica.
6. Kombinirani senzori:
   Kombiniraju više tehnologija, poput optičkih i infracrvenih senzora, kako bi poboljšali preciznost prepoznavanja lica.
7. Senzori za detekciju pokreta očiju i izraza lica:
   Fokusiraju se na praćenje pokreta očiju, izraza lica i gesta kako bi dobili dodatne informacije o korisniku.

Odabir određene vrste senzora ovisi o specifičnim zahtjevima aplikacije, uvjetima okoline i sigurnosnim potrebama. Kombinacija različitih senzora može pružiti sveobuhvatniji pristup prepoznavanju lica u različitim situacijama.

2.2. Pregled postojećih metoda i tehnika prepoznavanja otiska prsta

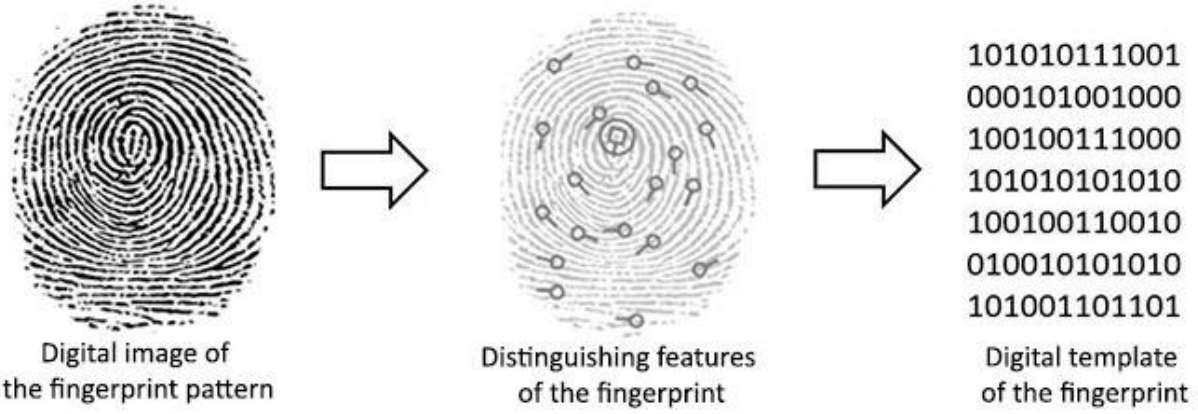
2.2.1. Općenito o prepoznavanju otiska prsta

Prepoznavanje otiska prsta je biometrijski sustav koji koristi jedinstvene karakteristike otiska prsta pojedinca kako bi ga identificirao ili potvrdio njegov identitet. Ova tehnologija temelji se na činjenici da su otisci prstiju jedinstveni za svaku osobu, čak i među bliskim rođacima.

Bilješke o upotrebi otisaka prstiju sežu do davne 2000. godina prije Krista kada su u starom Babilonu otiske prstiju koristili u poslovne svrhe. Radnici bi prstima dodirivali glinene ploče kako bi ih potpisali ili označili. U drevnoj Kini otisci prstiju koristili su se za identifikaciju i autentifikaciju pečata i identifikaciju djece. Još postoje puno sličnih primjera gdje su se otisci prstiju koristili na potpisivanje raznim dokumentima i kriminalističkim svrhama sve dok tijekom 20. stoljeća, tehnologija prepoznavanja otiska prsta znatno se razvila. Digitalizacija i automatizacija procesa omogućile su brže i preciznije usporedbe otisaka prstiju. Otisci prstiju postali su ključna biometrijska metoda u pravosudnim, sigurnosnim i forenzičkim aplikacijama [9]. Danas, prepoznavanje otiska prsta široko se koristi u različitim sektorima, uključujući forenziku, sigurnost, pravosuđe, mobilne uređaje i druge tehnološke sustave.

Postupak prepoznavanja otiska prsta počinje skeniranjem otiska prsta pomoću odgovarajućeg uređaja, poput optičkog senzora otiska prsta ili kapacitivnog senzora otiska prsta. Skener bilježi detalje poput crta, linija, točaka grananja i drugih karakteristika otiska prsta.

Nakon skeniranja, algoritam za prepoznavanje otiska prsta izdvaja ključne karakteristike otiska prsta, kao što su krivulje, točke grananja, udaljenosti između određenih točaka ili druge osobitosti. Te karakteristike koriste se za stvaranje jedinstvenog referentnog modela, koji predstavlja digitalnu sliku otiska prsta u obliku matematičkih parametara.



Slika 3. Stvaranje referentnog modela otiska prsta [9]

U sljedećem koraku, referentni model uspoređuje se s pohranjenim otiscima prsta u bazi podataka. Ako postoji podudaranje, sustav prepoznaje otisak prsta i povezuje ga s odgovarajućim identitetom. Ovisno o potrebi, sustav može provesti verifikaciju, gdje osoba potvrđuje svoj identitet, ili identifikaciju, gdje sustav traži podudaranje otiska prsta u cijeloj bazi podataka kako bi identificirao nepoznatu osobu.

Tehnologija prepoznavanja otiska prsta široko se koristi u sigurnosnim sustavima, mobilnim uređajima, bankarskim transakcijama, pristupu računalima i drugim područjima gdje je potrebna autentifikacija identiteta.

2.2.2. Algoritmi za prepoznavanje otiska prsta

Algoritmi za prepoznavanje otiska prsta igraju ključnu ulogu u procesu identifikacije i verifikacije. Ovdje su neki od klasičnih algoritama korištenih u prepoznavanju otisaka prstiju:

1. Minutiae-Based Algoritmi:

   Osnovna ideja: Identifikacija i usporedba specifičnih točaka na otisku prsta, poznatih kao minutije. Minutije uključuju točke grananja i krajeva linija.

   Primjena: Popularna metoda koja se koristi u forenzici i sustavima kontrole pristupa.

2. Pattern-Based Algoritmi:

   Osnovna ideja: Analiziranje i usporedba ukupnog obrasca otiska prsta, uključujući crte i udaljenosti među njima.

   Primjena: Ovaj pristup koristi se za identifikaciju vrsta otisaka prstiju, kao što su vijugavi, zrnati ili meškoljiti otisci.

3. Poincaré Index-Based Algoritmi:

   Osnovna ideja: Koristi se topologija otisaka prstiju, posebno Poincaré indeks, koji opisuje broj omotaja uzoraka otiska prsta.

   Primjena: Ovaj algoritam fokusira se na globalne karakteristike otiska prsta.

4. Singularity-Based Algoritmi:

   Osnovna ideja: Prepoznavanje singulariteta, kao što su delta, dvostruka delta i ostali točkasti centri u otiscima prstiju.

   Primjena: Ovaj pristup fokusira se na posebne točke koje imaju važnost u analizi otisaka prstiju.

5. Textural-Based Algoritmi:

   Osnovna ideja: Analiza teksture otiska prsta, uključujući mikrostrukture i druge teksturalne karakteristike.

   Primjena: Ovaj pristup naglašava vizualne aspekte otiska prsta kako bi postigao veću preciznost.

6. Wavelet-Based Algoritmi:

   Osnovna ideja: Korištenje transformacija valova za analizu otisaka prstiju na različitim razlučivostima.

   Primjena: Omogućava bolju prilagodljivost u analizi različitih detalja otiska prsta.

7. Neural Networks za prepoznavanje Otisaka Prstiju:

   Osnovna ideja: Korištenje neuronskih mreža za učenje i prepoznavanje obrazaca u otiscima prstiju.

   Primjena: Duboko učenje i neuronske mreže mogu pridonijeti visokoj preciznosti u prepoznavanju otisaka prstiju.

8. Obrada Grafova:

   Osnovna ideja: Modeliranje otisaka prstiju kao grafove i primjena algoritama obrade grafova za identifikaciju ključnih značajki.

   Primjena: Ova metoda omogućava efikasnu analizu i usporedbu otisaka prstiju.

Svaki od ovih algoritama ima svoje prednosti i nedostatke, a izbor ovisi o specifičnim zahtjevima sustava i primjeni prepoznavanja otisaka prstiju. U posljednjim godinama, s razvojem dubokog učenja, neuronske mreže postaju sve značajnije u ovom području, pridonoseći povećanoj preciznosti i skalabilnosti sustava prepoznavanja otisaka prstiju.

2.2.3. Senzori za prepoznavanje otiska prsta

Senzori za prepoznavanje otiska prsta su uređaji koji se koriste za prikupljanje otisaka prstiju i njihovo prepoznavanje. Postoje različite tehnologije senzora za prepoznavanje otiska prsta, uključujući kapacitivne, optičke, senzore osjetljive na pritisak, termo-električne čitače, senzore električnog polja i senzore bez dodira.

Kapacitivni senzori (od jednog ili više poluvodičkih čipova) u dodiru s kožom na zaslonu skenira otisak prstiju generiranjem digitalne slike papilarnih linija koje čine otisak. Senzor koristi kondenzatore (dvije strujne ploče) koji pomoću elektriciteta mjere otisak prsta dok procesor čita izlazni napon i određuje obilježja papilarnih linija te tako sastavlja opću sliku otiska prsta.

Optički senzori su jedna od najpoznatijih i najčešće korištenih tehnologija na ovom. Koriste se CCD kamere odnosno jednostavne diode osjetljive na svjetlost. Prst se postavlja na staklenu površinu te ga osvjetljavaju integrirane LED-ice kako bi se osvijetlile papile, a CCD sustav mjeri svjetlost (generira obrnute slike) koja se odbija od prsta na temelju čega se stvara digitalni otisak. Optički senzori mogu biti linijski, skeniraju liniju po liniju otiska ili površinu koja ima u sebi pokretni dio koji skenira svaku liniju površine otiska prsta. Druga podskupina senzora ove vrste je podešena tako da prati prisustvo ili odsustvo tzv. potpunog unutarnjeg odbijanja (total internal reflectance – TIR), koje predstavlja pojavu kod koje sučelje između stakla i zraka pod određenim uglovima funkcionira kao ogledalo. Problem ove tehnologije je što se ona jednostavno može prevariti korištenjem lažnih prstiju, a tu je i problem nevidljivih otiska to jest otiska koji ostaju na staklu nakon korištenja koji mogu stvarati probleme pri uzimanju uzorka.

Senzor osjetljiv na pritisak ima poluvodički materijal osjetljiv na pritisak koji snima nepravilnosti otiska prstiju. Senzori osjetljivi na pritisak temelje se na činjenici da samo papilarne linije za razliku od udubina dolaze u kontakt sa senzorom. Prednost je uzimanje otisaka bez utjecaja faktora poput mokrih ili suhih prstiju, ali nedostataka su binarne slike koje pohranjuju manje informacija.

Termički čitač otiska prsta mjeri temperaturnu razliku između točki koje su u spoju (izbočine) i onih koje to nisu (udubljenja) i pretvara ju u određeni napon. Taj način upotrebljava se i u infracrvenim kamerama.

Termo-električni senzori iako stvaraju vrlo upotrebljive digitalne slike otisaka prstiju visoke rezolucije bez obzira na eventualna onečišćenja na površini kože jer dolazi do samočišćenja, manje su korišteni. Metoda se sastoji u prelasku prsta preko senzora koji slika prst gdje započinje otisak prsta. Prednost ove tehnologije je što ne mjeri vanjski sloj kože, čime se zaobilaze mogući lažni prsti.

Senzori bazirani na beskontaktnoj tehnologiji rade slično kao i optički senzori. Koriste preciznu leću koja je na udaljenosti nekoliko centimetara od prsta koji se skenira. Za razliku od konvencionalnih sustava, nije potrebno pritisnuti prstom na staklenu površinu. Prednost je neosjetljivost na nečistoće, visoki temperaturni raspon , vrlo precizno snimanjem slika za visoku razinu prepoznavanja.

2.3. Pregled postojećih metoda i tehnika glasovne komunikacije prema čovjeku

2.3.1. Općenito o glasovnoj komunikaciji prema čovjeku

Glasovne naredbe predstavljaju metodu interakcije između ljudi i računala putem govora.

Još u 1950-ima i 1960-ima eksperimentiralo se s konceptom prepoznavanja govora, ali sa ograničenom učinkovitošću i točnošću. U sljedećih trideset godina istraživanje o glasovnoj komunikaciji postalo je sve intenzivnije. Razvijeni su prvi sustavi za prepoznavanje govora. U periodu od 2000. do 2010. godina došlo je do ubrzanog razvoja sustava za prepoznavanje govora(ASR). Tvrtke poput Google-a i Microsoft-a počele su integrirati glasovnu komunikaciju u svoje proizvode i usluge. Od tada do danas dolaskom pametnih telefona i pametnih zvučnika dovelo je do masovnog usvajanja glasovnih asistenata. Siri (Apple), Google Assistant (Google), Cortana (Microsoft), i Alexa (Amazon) postali su popularni. Ovi asistenti kombiniraju prepoznavanje govora, obradu prirodnog jezika i sintezu govora [12].

Glasovne naredbe postale su sastavni dio svakodnevnog života. Pored pametnih asistenata, tehnologija se integrira u automobile, uređaje za kuću, pametne televizore i razne druge uređaje.

Sve veći fokus je na poboljšanju točnosti prepoznavanja, razumijevanju konteksta i prilagodbi korisničkim postavki.

Tehnologije koje podržavaju glasovne naredbe danas:

1. Duboko učenje (Deep Learning),
2. Cloud Computig,
3. Internet of Things (IoT),
4. Prepoznavanje govora (ASR - Automatic Speech Recognition),
5. NLP (Natural Language Processing),
6. Hidden Markov Models (HMM),
7. Gaussovi modeli mješavine (GMM).

2.3.2. Algoritmi glasovne komunikacije prema čovjeku

Postoji nekoliko ključnih algoritama i tehnika koje se koriste u sustavima za glasovne naredbe. Evo nekoliko od njih:

1. Hidden Markov Models:

   HMM je većinom korišten algoritam u automatskom prepoznavanju govora (ASR).

   Kao što naziv govori koristi skrivene („hidden“) stanja koja se povezuju s govorom.

2. Gaussove mješavine (Gaussian Mixture Models - GMM):

   GMM se koristi za modeliranje distribucije zvuka u okviru HMM-a. Koristi se za modeliranje različitih stanja zvuka i pomaže u prepoznavanju riječi.

3. Rekurentne neuronske mreže (Recurrent Neural Networks - RNN):

   To je vrsta neuronskih mreža koja je dobra za rad sa sekvencijalnim podacima, uključujući zvuk govora.

4. Long Short-Term Memory Networks (LSTM) i Gated Recurrent Units (GRU):

   LSTM i GRU su bolje verzije RNN-a koje pomaže u rukovanju dugoročnim zavisnostima u podacima. Oni su posebno korisni u zadatcima gdje je potrebno pamćenje informacija.

5. Duboko učenje za obradu prirodnog jezika (Deep Learning for NLP):

Za razumijevanje naredbi i konteksta, često se koriste duboke neuronske mreže u kombinaciji sa tehnikama obrade prirodnog jezika.

6. Sinteza govora:

Ova tehnologija pretvara tekst u govor. Sinteza govora koristi algoritme za generiranje prirodnog zvuka govora. To omogućava računalima da "govore" korisnicima. Google Text-to-Speech, Amazon Polly i Microsoft Azure Speech Synthesis su primjeri sinteze govora.

7. TTS (Text-To-Speech):

To je algoritam kao što i naziv govori koji pretvara tekst u govor. Text to Speech je jednostavan i cjelovito rješenje za tekst u govor pogotovo na hrvatskom jeziku.

Ovi algoritmi često rade zajedno u okviru cjelokupnog sustava glasovnih naredbi, gdje se kombiniraju tehnike prepoznavanja govora, analize govora i obrade prirodnog jezika kako bi se pružila efikasna i točna usluga korisnicima.

2.3.3. Senzori za glasovnu komunikaciju prema čovjeku

Ovo je zapravo, zvučnik, elektromehanički pretvarač koji pobuđen električnim signalom proizvodi zvuk frekvencijskog opsega od 20 do 20.000 Hz, odnosno zvuk namijenjen ljudskom uhu. Pojam zvučnik se također često koristi i za zvučničku kutiju u kojoj se nalazi jedna ili više zvučničkih jedinica, tj. zvučnika u užem smislu.

Prvi je zvučnik je bila telefonska slušalica koju je 1876. patentirao Alexander Graham Bell. Nedugo zatim uslijedila je poboljšana verzija Ernsta Siemensa u Njemačkoj i Engleskoj 1878. Vjeruje se da je sličnu napravu stvorio 1881. i Nikola Tesla. Modernu konstrukciju zvučnika s pomičnom zavojnicom ostvario je Oliver Lodge u Engleskoj 1898.

Prvi su zvučnici koristili elektromagnete jer je u to doba cijena permanentnih magneta potrebne veličine i snage bila prevelika. Elektromagnet se pobuđivao istosmjernom strujom kroz zavojnicu koja je obično služila i kao prigušnica u ispravljaču pojačala na koje je zvučnik bio priključen.

Vrste zvučnika:

1. Dinamički zvučnik:

Koriste elektromagnetski sustav za pretvaranje električne energije u mehaničke vibracije membrane zvučnika.

2. Elektrostatski zvučnik:

Koriste elektrostatsku silu za generiranje zvuka. Imaju tanku elektrostatsku membranu između dvije elektrode.

3. Piezoelektrični zvučnik:

Generiraju zvuk pomoću piezoelektričnih materijala koji proizvode mehaničke oscilacije pod električnim naponom.

4. Magnetostriktivni zvučnici:

Koriste promjene u duljini materijala pod utjecajem magnetskog polja za generiranje zvuka.

5. Ribbon (Trakasti) zvučnici:

Imaju tanku metalnu vrpcu koja oscilira u magnetskom polju, generirajući zvuk.

6. Subwooferi:

Specijalizirani zvučnici za reprodukciju niskih frekvencija, uključujući basove.

7. Satelitski zvučnici:

Manji zvučnici često korišteni u višekanalnim audio sustavima.

8. Aktivni zvučnici:

Imaju ugrađeni pojačalo, što znači da ne zahtijevaju dodatnu opremu za pojačavanje zvuka.

9. Bežični zvučnici:

Povezani su putem bežičnih tehnologija kao što je npr. Bluetooth.

10. Tanjurni (Flat Panel) zvučnici:

Tanak panel vibrira kako bi stvarao zvuk.

Svaka vrsta zvučnika ima svoje prednosti i ograničenja, a odabir ovisi o specifičnim potrebama i zahtjevima audio sustava.

3. METODOLOGIJA

U ovom poglavlju detaljnije će biti opisani algoritmi koji su korišteni u ovom diplomskom radu.

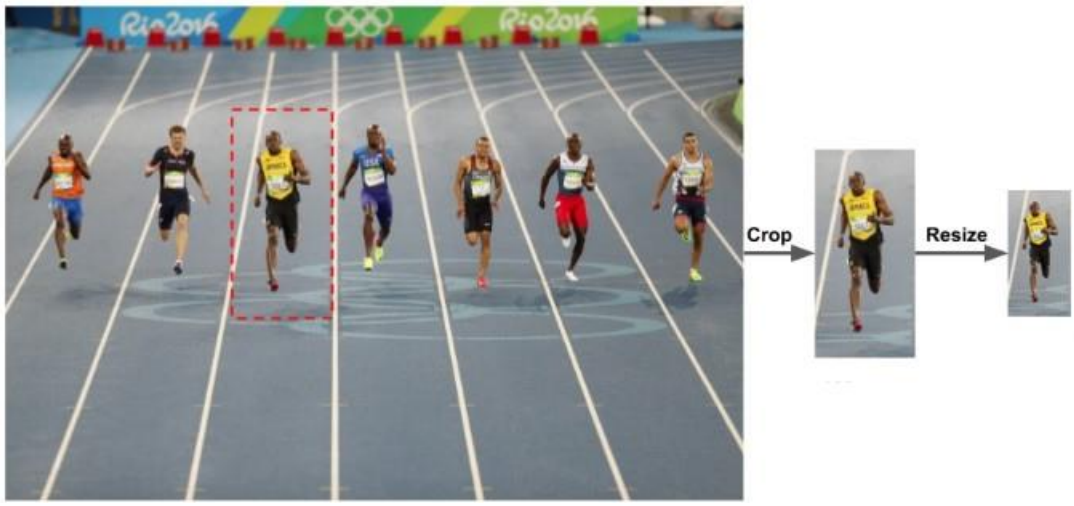
3.1. HOG (Histogram of Oriented Gradients)

To je algoritam koji se koristi ne samo za prepoznavanje lica već i drugih objekata u digitalnim slikama. Ovaj algoritam je prilično popularan i široko korišten u računalnom vidu.

Osnovna ideja iza HOG algoritma je da se karakteristike objekata u slici mogu opisati na osnovu raspodjele orijentacija gradijenata piksela. Algoritam se primarno koristi za prepoznavanje lica, ali se može primijeniti i na druge objekte kao što je već spomenuto.

HOG algoritam se sastoji od nekoliko osnovnih koraka:

1. Predprocesiranje slike: Na slici [Slika 4.] prikazan je primjer predprocesiranja ulazne slike kao što su izrezivanje i promjena veličine. Također, može se primijeniti normalizacija kontrasta i druge tehnike za poboljšanje kvaliteta slike.



Slika 4. Predprocesiranje slike [13]

2. Izračunavanje gradijenata: Zatim se izračunaju horizontalni i vertikalni gradijentni pikseli na slici. To se jednostavno postiže filtriranjem slike primjenom operatora gradijenta poput Sobel operatora prikazanog na slici [Slika 5.].

Slika 5. Sobel operator

Koristeći formule (1) i (2) nalazimo veličinu i smjer gradijenta.

$$g = \sqrt{g_x^2 + g_y^2} \quad \textbf{(1)}$$

$$\theta = \tan^{-1}\frac{g_y}{g_x} \quad \textbf{(2)}$$

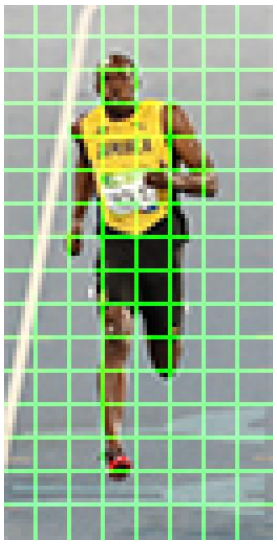
Kao što je vidljivo na slici [Slika 6.], x-gradijent se aktivira na okomitim linijama, a y-gradijent na vodoravnim linijama. Veličina gradijenta aktivira se gdje god postoji oštra promjena intenziteta. Nijedan od njih se ne aktivira kada je područje glatko.

Slika gradijenta uklonila je mnogo nebitnih informacija (npr. konstantno obojena pozadina), ali istaknula obrise. Drugim riječima, možete pogledati sliku gradijenta i još uvijek lako reći da na slici ima osoba.

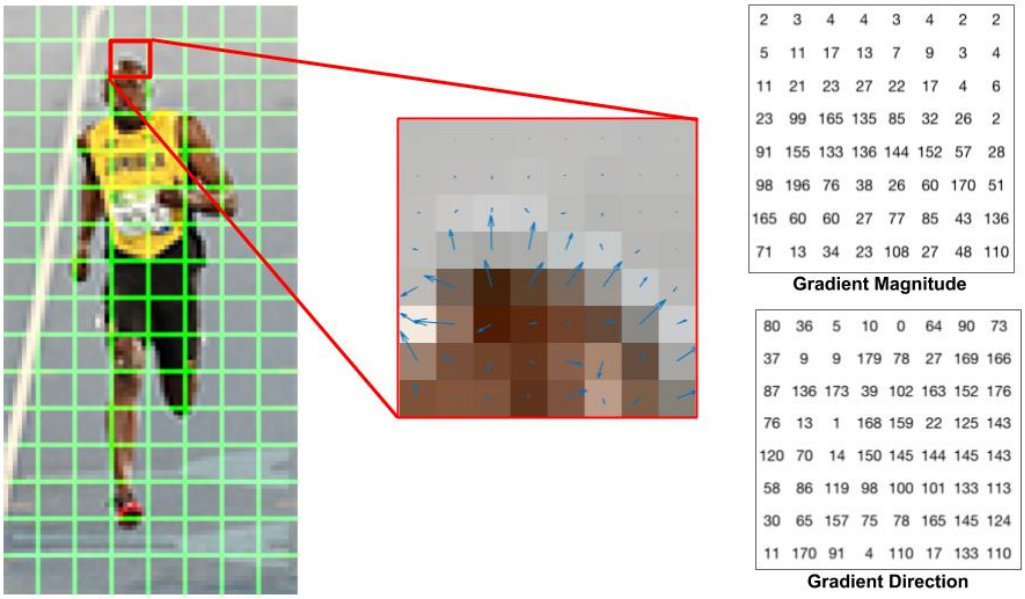
Na svakoj točki, gradijent ima veličinu i smjer. Za slike u boji, gradijenti tri kanala se procjenjuju (kao što je prikazano na slici ispod). Veličina gradijenta na pikselu je maksimum veličine gradijenata tri kanala, a kut je kut koji odgovara maksimalnom gradijentu.

Slika 6. Lijevo: x-gradijent, centar: y-gradijent, desno: veličina gradijenta [13]

3. Izračunavanje histograma orijentacija gradijenata: Slika se dijeli na manje dijelove (ćelije) kao što je prikazano na slici [Slika 7.] i za svaku ćeliju se izračunava histogram orijentacija gradijenata. Histogram predstavlja raspodjelu orijentacija gradijenata u ćeliji.



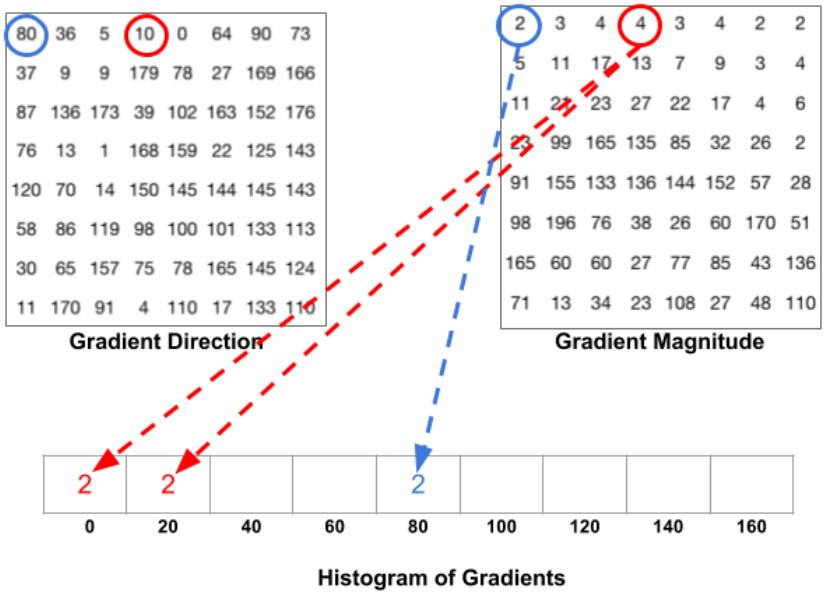
Slika 7. Slika podijeljena na 8x8 ćelije [13]



Slika 8. Centar: gradijenti prikazani strelicama, desno: gradijenti prikazani brojevima [13]

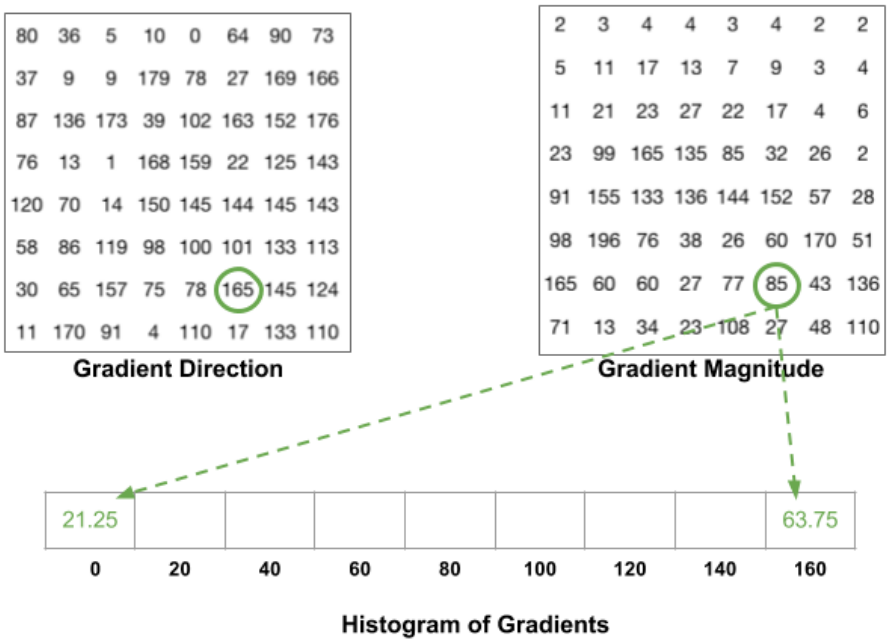
Slika u sredini prikazuje gradijent sa strelicama na način da strelice prikazuju smjer gradijenta, veličina strelica veličinu gradijenta. Na desnoj strani vide se gradijenti prikazani brojevima između 0 i 180.

Sljedeći korak je izrada histograma. Histogram sadrži 9 spremnika povezanih sa kutevima 0, 20, 40,… 160. Na slici [Slika 9.] prikazan je proces izrade histograma za istu ćeliju kao na slici ranije. Jedan spremnik se bira na temelju smjera, a vrijednost se bira na temelju veličine. Prvo gledamo na piksel koji je okružen plavim krugom. Ima kut (smjer) od 80 stupnjeva i veličinu od 2. Dakle, dodaje se 2 u 5. spremnik. Gradijent na pikselu okruženom crvenim krugom ima kut od 10 stupnjeva i veličinu od 4. Budući da su 10 stupnjeva na pola puta između 0 i 20, vrijednost se ravnomjerno dijeli na dva spremnika.



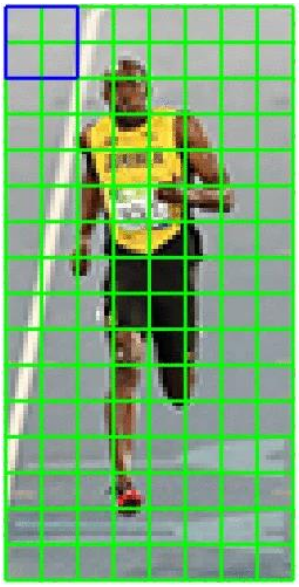
Slika 9. Izrada histograma [13]

Postoji još jedan detalj na koji treba obratiti pažnju. Ako je kut veći od 160 stupnjeva, nalazi se između 160 i 180, a znamo da se kut odvija oko 0 i 180, što znači da su 0 i 180 ekvivalentni. Dakle, na slici u nastavku, piksel s kutom od 165 stupnjeva proporcionalno doprinosi spremniku od 0 stupnjeva i spremniku od 160 stupnjeva.



Slika 10. Izrada histograma poseban primjer [13]

4. Normalizacija blokova: Histogrami se grupiraju u preklapajuće blokove i normaliziraju kako bi se smanjio utjecaj promjene osvjetljenja i kontrasta. Ovo se obično postiže primjenom lokalnog kontrastnog normalizacijskog algoritma.



Slika 11. Normalizacija blokova [13]

5. Klasifikacija: Normalizirani vektori karakteristika se prosljeđuju klasifikatoru, kao što je Support Vector Machine (SVM), koji može da nauči razlikovati lica od drugih objekata.

HOG algoritam ima nekoliko prednosti. On je robustan na promjene osvjetljenja i deformacije objekata. Također, algoritam može raditi na slikama različitih veličina i rotirajućih objektima. Međutim, HOG ima ograničenu sposobnost prepoznavanja objekata s kompleksnijim teksturama i kontekstualnim informacijama.

HOG algoritam je bio prvi u razvoju prepoznavanja objekata u računalnom vidu, a danas se koristi u mnogim aplikacijama, uključujući video nadzor, autonomnu vožnju, biometrijsku identifikaciju i druga područja gdje je prepoznavanje objekata od interesa.

3.2. Minutiae-Based algoritam

Minutiae-Based algoritam za prepoznavanje otisaka prstiju je tehnike koja koristi specifične značajke, nazvane minutije, kako bi se identificirale i razlikovale otiske prstiju pojedinaca. Ovaj pristup temelji se na prepoznavanju ključnih točaka, poput završetaka i dioba grebena, koje su inherentno prisutne na svakom otisku prsta.

Osnovni koraci Minutiae-Based algoritma uključuju prvo akviziciju visokokvalitetne slike otiska prsta. Nakon toga, slijedi faza preprocesiranja koja ima za cilj poboljšati kvalitetu slike, eliminirati šum i normalizirati karakteristike otiska prsta.

Glavni dio algoritma je izdvajanje minutija, što uključuje identifikaciju i ekstrakciju ključnih značajki poput završetaka i dioba grebena. Ove minutije se zatim koriste za stvaranje jedinstvene reprezentacije otiska prsta.

Usporedba minutija obavlja se usporedbom izdvojenih značajki s minutijama pohranjenima u bazi podataka. U ovom koraku algoritam analizira prostorne odnose, orijentacije i druge karakteristike minutija kako bi pronašao podudaranje. Konačno, donosi se odluka o podudaranju ili nepodudaranju otiska prsta.

Minutiae-Based algoritmi su popularni zbog svoje visoke preciznosti, ali su također osjetljivi na faktore poput kvalitete slike otiska prsta. Stoga se često primjenjuju dodatne tehnike, poput poboljšanja algoritama za prepoznavanje ili korištenja više vrsta minutija, kako bi se povećala robusnost i stabilnost sustava.

3.3. TTS (Text-To-Speech)

Pyttsx3 je biblioteka koja omogućuje jednostavno korištenje sinteze govora u Pythonu, bez potrebe za poznavanjem i radom s unaprijed definiranim algoritmima, poput HMM-a, GMM-a ili sličnih tehnika specifičnih za automatsko prepoznavanje govora (ASR) ili sintezu govora.

Kada se koristi pyttsx3, radi se s višim nivoom apstrakcije koja omogućuje jednostavnu sintezu govora na lokalnom uređaju. Ova biblioteka iznenađujuće dobro surađuje s Microsoft SAPI5, što je API sustava za sintezu govora na Windows platformi. Kako točno pyttsx3 radi bit će detaljnije objašnjeno u poglavlju programiranje.

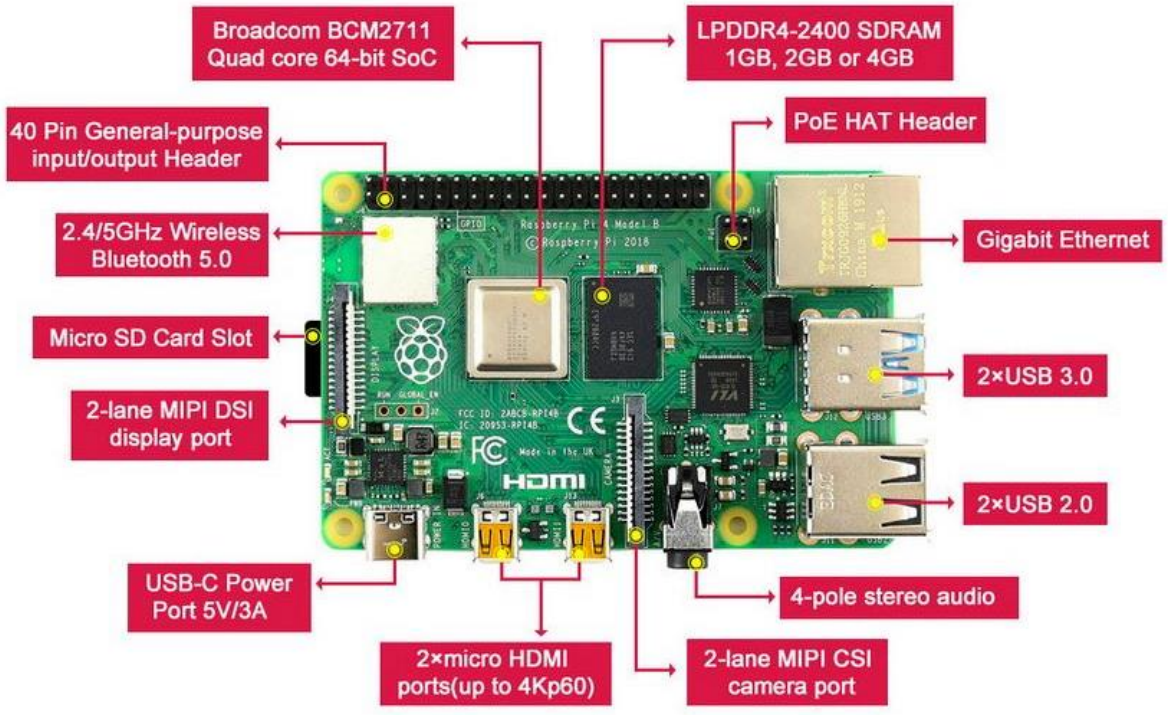
Pyttsx3 apstrahira složenost korištenja specifičnih algoritama za sintezu govora i pruža jednostavno sučelje za rad s lokalnim resursima na platformi na kojoj se izvršava.

4. IMPLEMENTACIJA

4.1. Odabir komponenata

4.1.1. Raspberry Pi 4 Model B

Raspberry Pi 4 Model B je najnoviji proizvod u popularnom rasponu računala Raspberry Pi. Nudi revolucionarne povećanja brzine procesora, multimedijske performanse, memorije i povezivosti u usporedbi s prethodnom generacijom Raspberry Pi 3 Model B+, zadržavajući kompatibilnost unatrag i sličnu potrošnju energije. Za krajnjeg korisnika, Raspberry Pi 4 Model B pruža performanse radne površine usporedive s ulaznim x86 PC sustavima. Glavne značajke ovog proizvoda uključuju visoko performansni 64-bitni četverojezgreni procesor, podršku za dvostruki prikaz na rezolucijama do 4K putem para micro-HDMI priključaka, hardversko dekodiranje videozapisa do 4Kp60, do 8 GB RAM-a, dvostruka 2.4/5.0 GHz bežična LAN mreža, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, i PoE mogućnost (putem zasebnog PoE HAT dodatka). Dvostruka bežična LAN i Bluetooth imaju modularnu certifikaciju sukladnosti, omogućujući dizajniranje ploče u konačne proizvode s znatno smanjenim testiranje sukladnosti, poboljšavajući trošak i vrijeme izlaska na tržište.



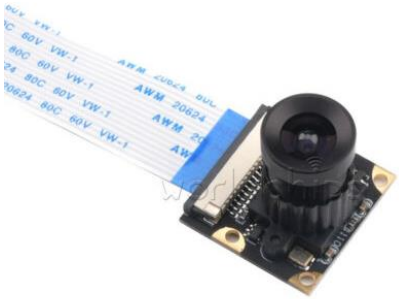
Slika 12. Raspberry Pi 4 Model B [16]

Tablica 1. Karakteristike Raspberry Pi 4 računala

Procesor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memorija	1GB, 2GB, 4GB ili 8GB LPDDR4 zavisno o modelu
Povezanost	2.4 GHz i 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 port 2 × USB 2.0 port
GPIO	Standardni 40-pin GPIO sučelje
Video i zvuk	2 × micro HDMI port (do 4Kp60 podržano) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedija	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD kartica	Micro SD kartica za operacijski sustav i spremanje podataka
Ulazna snaga	5V DC via USB-C connector (minimum 3A) 5V DC via GPIO header (minimum 3A) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Okoliš	Operacijska temperatura 0-50°C

4.1.2. Kamera modul za Raspberry Pi 4 B

Raspberry Pi kamera posebno je dizajnirana kamera kako bi radila sa svim Raspberry Pi modelima. Priključuje se na Raspberry putem 15-pinskog flat kabela u kamera konektor na RPi ploči. Naravno radi, bez instaliranja drivera, na Raspbian OS-u.



Slika 13. Kamera modul za Raspberry Pi 4 B

U tablici [Tablica 2.] prikazane su glavne karakteristike kamere modula za Raspberry Pi 4.

Tablica 2. Karakteristike kamere modula za Raspberry Pi 4 B

CCD veličina	¼ inch
Otvor(F)	1,8
Žarišna duljina	3,6 mm
Vidno polje	72 stupnja
Preferirana razlučivost senzora	1080p
Dimenzije	25x24 mm

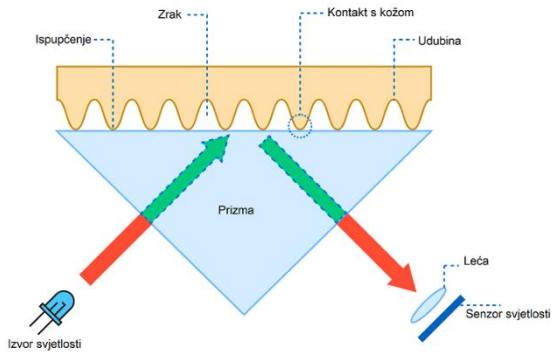
4.1.3. Senzor otiska prsta

Senzor otiska prsta omogućuje čitanja i prepoznavanje otisaka prstiju. Sa mikroupravljačem komunicira preko serijske veze, a u mogućnosti je spremiti do 300 različitih otisaka prstiju. Nakon što se prisloni prst, fotografija otiska uspoređuje se s bazom spremljenih otisaka te ako isti prst postoji, daje se signal za otključavanje.

Ovaj senzor je optički što znači da pomoću svijetla prepoznaje otisak prsta, a osim optičkog ima još i kapacitivni senzor(koristi se na mobilnim uređajima) i drugi kao što je ranije navedeno. Optički senzor otiska prsta je starija tehnologija pa je stoga i jednostavnija od ostalih, ali baš zbog jednostavnosti rada je pristupačan kao što je ranije navedeno.

Slika 14. Senzor otiska prsta [17]

Kao što znamo na prstu imamo ispupčenja i udubine koje su jedinstvene za svaku osobu i to je otisak prsta. Da bi detektirao udubine i ispupčenja senzor otiska prsta koristi led diode kao osvjetljenje i fotodetektor kako bi primio odbijene zrake svjetlosti. Naravno senzor nema samo jednu diodu i fotodetektor nego se unutar njega nalazi puno fotodetektora smještenih u matricu, kao i prizma i leća kako bi točnije mogao primiti odbijenu svjetlost. Na slici [Slika 15.] vidimo princip na koji senzor radi i koji je prethodno objašnjen.



Slika 15. Princip rada senzora otiska prsta

Naravno u senzoru imamo i kontroler koji upravlja diodama i fotodetektorima te analizira primljene signale. Kako bi mogao prepoznati otisak kontroler u memoriji ima bazu spremljenih otisaka te ih uspoređuje s trenutnim otiskom i ako ga ima u bazi vraća nam koliko se podudaraju.

Tablica 3. Karakteristike senzora otiska prsta

Napon	3,3 V
Struja	120 mA
Komunikacija	UART
Dimenzije	50 x 21 x 20 mm

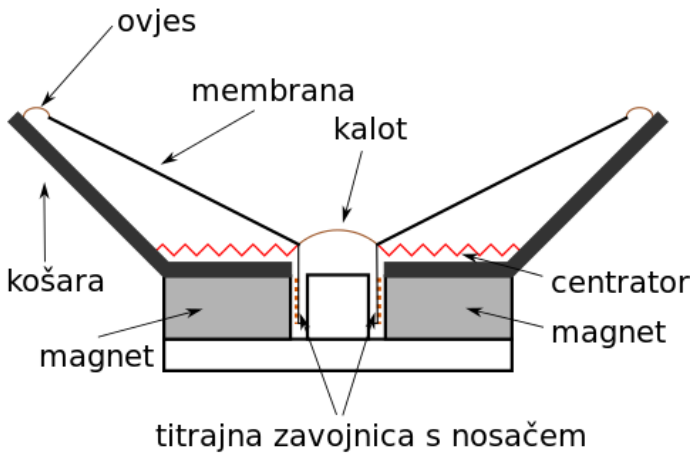
Senzor ima 6 pinski kabel, ali za spajanje se koriste samo 4 pina (2 za napajanje i 2 za komunikaciju). RX pin se spaja na GPIO 14, a TX sa senzora spaja se na GPIO 15 na Raspberry Pi-u, za napajanje se koristi 3,3V kako je naznačeno na senzoru.

4.1.4. Dinamički zvučnik

On koristi elektromagnetski sustav za pretvaranje električne energije u zvučne valove. Njegova osnovna struktura uključuje membranu (zvučnički stožac) povezanu s okvirom unutar magnetskog polja. Kada električna struja prolazi kroz žicu omotanu oko magneta, stvara se magnetsko polje koje integrira s magnetskim poljem membrane, uzrokujući vibracije koje reproduciraju zvuk.

Slika 16. Dinamički zvučnik

Ova vrsta zvučnika često se koristi zbog svoje sposobnosti reprodukcije širokog frekvencijskog raspona, od niskih do visokih tonova. Membrana zvučnika obično je izrađena od laganih, ali čvrstih materijala kao što su papir, plastika ili kevlar. Dinamički zvučnici su rasprostranjeni u raznim uređajima, uključujući kućne audio sustave, automobile, mobilne uređaje i profesionalnu audio opremu. Njihova učinkovitost u pretvaranju električne energije u zvuk čini ih popularnim izborom u mnogim aplikacijama.



Slika 17. Dijelovi dinamičkog zvučnika

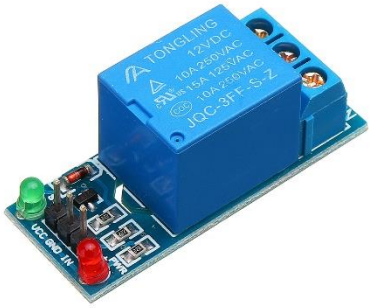
U tablici [Tablica 4.] prikazuje osnovne karakteristike dinamičkog zvučnika.

Tablica 4. Karakteristike zvučnika

Snaga	0.5 W
Promjer	47 mm
Otpor	8 OHM
Raspon frekvencija	0-12 000 Hz
Jakost zvuka	84 dB

4.1.5. Relej

Relej je elektromagnetski prekidač pomoću kojeg možemo s manjim naponom kontrolirati protok struje većeg napona. Modul prikazan na slici [Slika 18.] koristi se za aktiviranje elektromagnetske brave na principu - uključivanjem releja pokrenuti će se brava što će omogućiti ulaz korisniku.

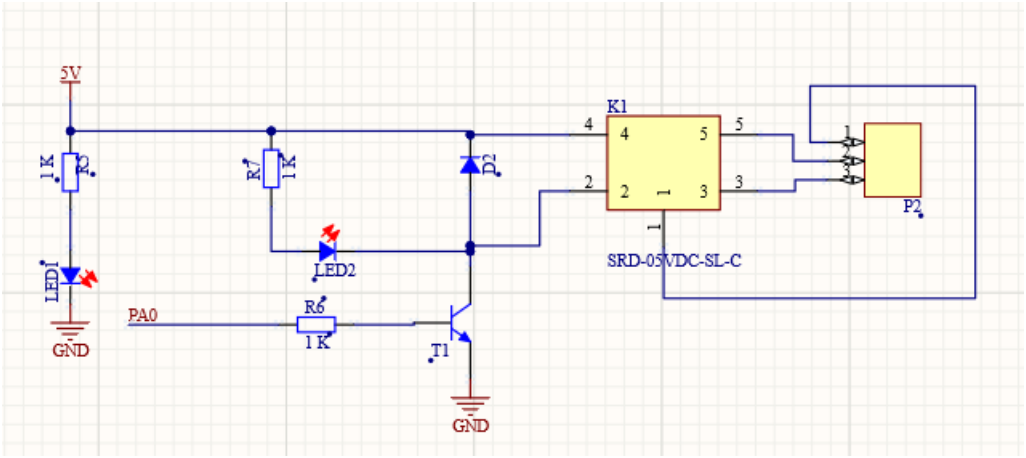


Slika 18. Relej

Elektromagnetski relej se sastoji od zavojnice, željezne kotve i kontakta. Kada struja proteče kroz zavojnicu, stvara se magnetsko polje koje privlači željeznu kotvu. Na kotvi se nalaze kontakti koji ovisno o protoku struje zatvaraju ili otvaraju drugi strujni krug. Unutar tog drugog strujnog kruga moguće je spojiti trošilo koje zahtjeva visoki napon ili izmjenični napon. Za rukovanje relejom potrebno je znati nekoliko karakteristika samog releja. Prvo je napon koji je potrebno dovesti zavojnici da se aktivira. Postoje razne vrijednosti tog napona ovisno o potrebi i namjeni (često 5, 12, 24V DC). Druga stvar na koju je potrebno obratiti pažnju je snaga trošila koje spajamo na relej. Na samom releju je napisana maksimalna količina struje i napona koja može biti spojena s relejom (često 12, 14, 30V DC ili 120, 220V AC). Ako se prekorače granice, relej se može oštetiti ili pregorjeti.

Tablica 5. Karakteristike releja

Struja potrebna za aktivaciju	20 mA
Napon zavojnice	5 V
Max DC	12 V, 10 A
Max AC	250 V, 10 A



Slika 19. Shematski prikaz releja

4.1.6. Elektromagnetska brava za vrata

Elektromagnetska brava prikazan na slici [Slika 20.] omogućava zaključavanje vrata putem mikroupravljača. Specifikacije brave prikazane su u tablici [Tablica 6.]. Navedena brava može držati do 80kg svojim elektromagnetom, što je otprilike na granici za korištenje s pravim vratima. Aktiviran je, tj. zaključava se kada mu se dovede struja. Sastoji se od dva dijela: elektromagneta i komada metala. Ugrađuje se vijcima za ugradnju.

Slika 20. Elektromagnetska brava za vrata

Tablica 6. Karakteristike elektromagnetske brave

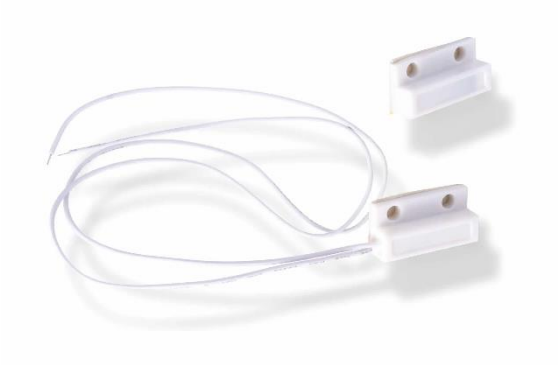
Napon aktivacije	12 V
Stalna struja aktivacije	100 mA
Dimenzije	80 x 40 x 20 mm
Masa	cca 500 g

4.1.7. Magnetni prekidači za vrata/prozore

Magnetni prekidači za vrata i prozore [Slika 21.] su sigurnosni uređaji koji se koriste za detekciju otvaranja ili zatvaranja vrata ili prozora. Ovi prekidači obično sadrže magnetnu komponentu i senzor.

Zapravo se radi o magnetu i reed senzoru od kojih se svaki nalazi u jednom od ova dva dijela. Postavljaju se najčešće na vrata i prozore jedan pored drugoga. Magnetna komponenta se postavlja na pokretni dio vrata ili prozora, dok se senzor postavlja na nepokretni dio.

Kada su blizu jedan drugoga(točnije 13mm ili manje), reed switch se zatvara, stvarajući zatvoren magnetni krug. Spaja se na Raspberry Pi / Arduino kao običan prekidač i moguća je detekciju otvorenih vrata, prozora, ladica, kuhinjskih ormarića, itd.



Slika 21. Magnetni prekidači za vrata/prozore

Tablica 7. Karakteristike magnetskih prekidača

Napon	max. 200V, najbolje koristite 5V od Raspberry Pi-a
Maksimalna struja kroz reed	100mA
Maksimalna udaljenost za detekciju	15mm, 13mm preporučeno
Dimenzije	29 x 14 x 9mm
Duljina kabla	30cm

4.1.8. Monitor, tipkovnica i miš

Raspberry Pi je računalo koje je potrebno spojiti na monitor da se može koristiti odnosno prikazivati sliku. Ovdje je konkretno korišten Samsung P2270HD monitor koji ja inače koristim za svoje računalo i koji je prikazana na slici [Slika 22.]. Monitor se sa Raspberry Pi-em spaja pomoću HDMI kabla.

Slika 22. Monitor za eksperiment

Za tipkovnicu je korištena službena Raspberry Pi tipkovnica koja uključuje tri USB porta za spajanje dodatnih vanjskih uređaja i koja je prikazana na slici [Slika 23.]. Ona se na Raspberry Pi spaja pomoću mikro USB priključka. Ima 79 tipki, automatski prepoznaje jezik, ergonomski je dizajnirana i kompatibilna je sa svim Raspberry Pi modelima.

Slika 23. Raspberry Pi tipkovnica

Za miš je korišten klasični miš koji sam imao kod kuće i spojen je na jedan od portova na tipkovnici.

4.1.9. VNC

VNC (Virtual Network Computing) je tehnologija koja omogućava udaljeni pristup i kontrolu računala putem mreže. To je ovdje posebno korisno jer ako želite upravljati Raspberry Pi uređajem, a nemate fizički pristup zaslonu i tipkovnici. Evo detaljnijeg objašnjenja kako koristiti VNC za ovaj diplomski rad:

1. Uključivanje VNC na Raspberry Pi uređaju
   Potrebno je omogućiti VNC u postavkama na Raspberry Pi-u
2. Instalacija i pokretanje VNC servera
   Ako već nije instaliran, potrebno ga je instalirati
3. Povezivanje s Raspberry Pi putem VNC klijenta
   Na osobnom računalu sa kojeg želimo upravljati Raspberry Pi uređajem potrebno je preuzeti VNC klijent, ovdje je konkretno korišten TigerVNC, ali postoje i drugi poput RealVNC Viewer, TightVNC Viewer, itd.
   Nakon što se pokrene VNC klijent, unese se IP adresa Raspberry Pija ili hostname te odgovarajući port i podaci za prijavu
4. Rad na Raspberry Pi-u putem VNC-a

Sada bi se trebala vidjeti radna površina Raspberry Pija i također se može koristiti tipkovnica i miš na svom računalu kako bi se upravljalo Raspberry Pijem.

5. Dodatne postavke

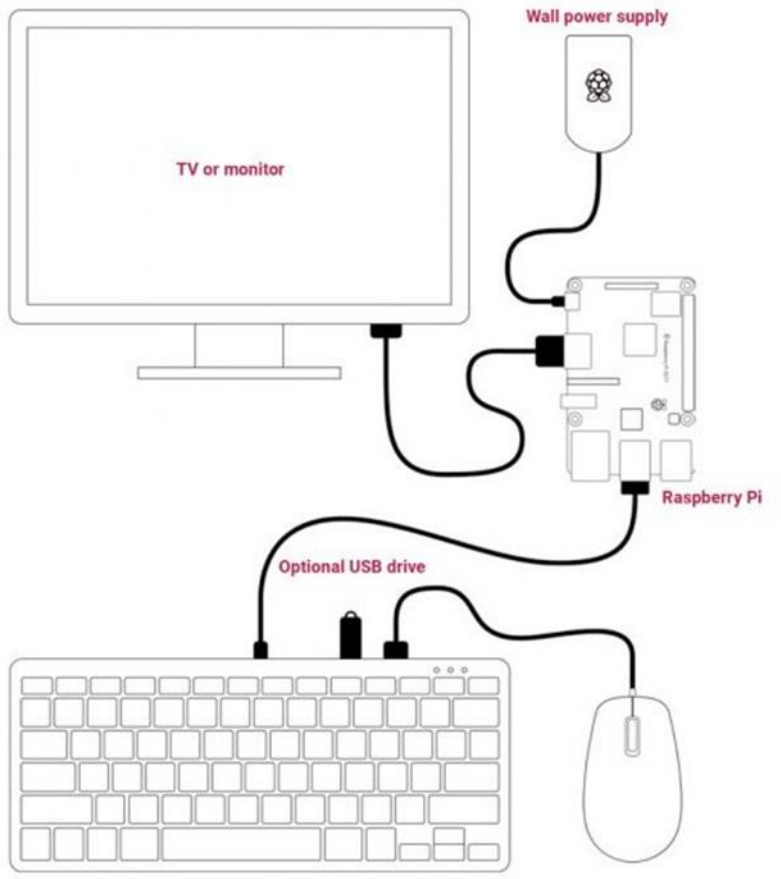
   Mogu se prilagoditi postavke prema vlastitim potrebama, kao što je rezolucija, boje i druge opcije.

Ovaj proces omogućuje udaljeni pristup i upravljanje Raspberry Pi uređajem, što je ovdje jako korisno.

Slika 24. Logo TigerVNC-a

4.2. Spajanje

Nakon odabira gotovih modula koji će se koristiti za izradu sklopa prepoznavanja lica potrebno je međusobno ih spojiti na odgovarajuće priključke svakog od uređaja. Prvi korak je pažljivo proučiti svaki priručnik (engl. datasheet) od pojedinog modula, vidjeti protokol kojim komunicira, te zatim taj protokol i priključke pronaći na Raspberry Pi 4 pločici i spojiti ih. Na sljedećoj slici [Slika 25.] je prikazana opća službena shema spajanja Raspberry Pi. Kao što je već ranije navedeno Raspberry Pi se na monitor spaja pomoću HDMI-a, tipkovnica se spaja preko jednog USB porta, a na samu tipkovnicu se spaja miš. Napajanje se dovodi preko USB Tip C porta. Kamera se priključuje pomoću kamera porta koji je prikazan na slici [Slika 12.]. Ostali moduli se spajaju preko 40 pinskog priključka na odgovarajući način. Raspored pinova je prikazan na slici [Slika 26.].



Slika 25. Opća shema spajanja Raspberry Pi-a

Kao što je već navedeno ranije ostali moduli se spajaju preko 40 pinskog priključka i to na sljedeći način:

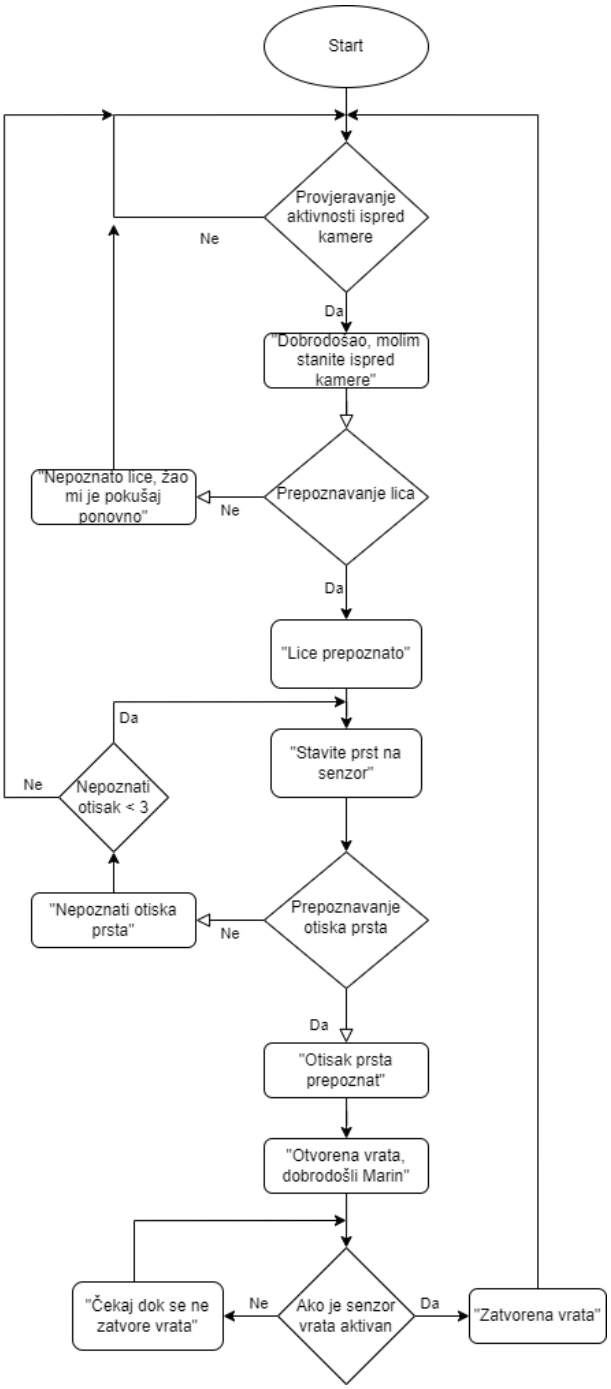
1. Senzor otiska prsta spaja se na 3,3 V i Ground te ostale dvije žice na TxD i RxD na GPIO 14 i 15.
2. Relej se također spaja na 3,3 V i Ground te na GPIO 18.
3. Magnetni prekidač se spaja na Ground jedna žica, a druga na GPIO 17.
4. Dvije Ledice koje se spajaju na Ground i GPIO 22 i 23.

Slika 26. Raspored pinova

4.3. Programiranje

4.3.1. Dijagram toka

Osmišljen je općenit dijagram toka kojim bi program trebao izvoditi naredbe i prikazan je na slici [Slika 27.].



Slika 27. Dijagram toka

Prije samog programiranja potrebno je instalirati potrebne biblioteke OpenCV, face_recognition module, imutils, pillow, dlib, adafruit_fingerprint, pyttsx3.

To se izvodi na način da se unutar naredbenog retka upiše „pip3 install“ i pripadajuće ime biblioteke.

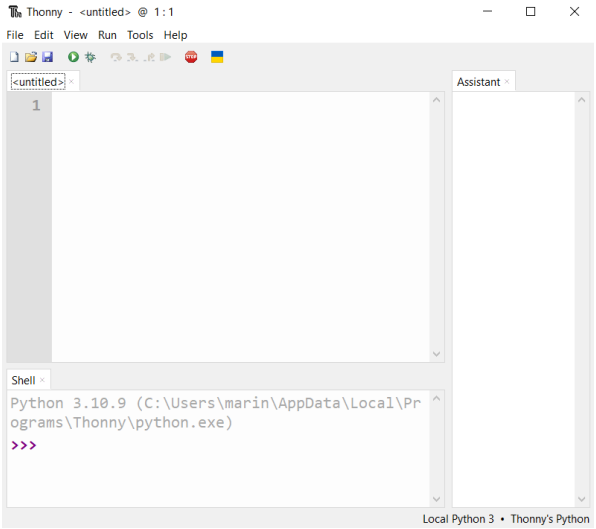
4.3.2. Thonny IDE

Thonny je već instalirani tekstualni uređivač na Raspbian operacijskom sustavu. To je jednostavno integrirano razvojno okruženje (IDE) dizajnirano posebno za programski jezik Python. Razvijen je kako bi bio jednostavan za korištenje, posebno za početnike, ali također pruža dovoljno alata za napredne korisnike.

Kada se Thonny pokrene, dobije se jednostavno i intuitivno sučelje tekstualnog uređivača prikazano na slici [Slika 28.]. Kao i u svakom programu može se stvoriti nova datoteka ili otvoriti postojeća datoteka kako bi se uredila. Thonny pruža značajke poput bojanja sintakse, automatskog završetka koda, pretraživanja i zamjene teksta, integriranog terminala i mnoge druge korisne funkcionalnosti.

Također, Thonny ima mogućnost izvršavanja programa napisanih u različitim programskim jezicima. Može se jednostavno napisati kod, spremiti ga i pokrenuti iz samog Thonny sučelja.

U ovom projektu je korišten za izradu i uređivanje Python skripti jer je Thonny jedan od popularnijih izbora zbog svoje jednostavnosti i funkcionalnosti.



Slika 28. Sučelje Thonny-ja

Slika 29. Logo Thonny-ja

4.3.3. Spremanje slika lica

Prvo je potrebno spremiti slike lica kojeg će program naučiti odnosno koje će se prepoznavati. Na početku je potrebno upisati ime za koje se slika. Moguće je dodati osoba koliko želimo samo se mijenjaju imena. Prilikom pokretanja programa otvara se prozor odnosno upali se kamera i mogu se slikati slike pritiskom na tipku „space“. Poželjno je naravno da se glava okreće odnosno da se slikaju slike iz više različitih pozicija glave da konačni rezultat bude bolji.

```
import cv2

name = 'Marin'  #upisi ime za koje slikas

cam = cv2.VideoCapture(0)

cv2.namedWindow("press space to take a photo ", cv2.WINDOW_NORMAL)
cv2.resizeWindow("press space to take a photo ", 500, 300)

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
```

```
        break
    cv2.imshow("press space to take a photo", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name = "dataset/"+ name +"/image_{}.jpg".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

Za spremanje slika potrebno je u istoj mapi izraditi novu mapu sa svojim imenom.

Na ovakav način moguće je spremati i slikati broj lica po vlastitoj želji.

4.3.4. Treniranje

Sada je potrebno analizirati slike spremljene u mapu. Na početku se instaliraju sve potrebne biblioteke i moduli. Zatim se sve slike koje se nalaze u mapi pohranjuju u listu 'imagePaths'. Kroz tu listu prolazi for petlja i učitane slike konvertira iz BGR formata u RGB format. Ova konverzija je potrebna jer biblioteka 'face_recognition' koristi RGB redoslijed boja. Nakon toga se detektiraju koordinate okvira (x, y) koje odgovaraju svakom licu na slici pomoću funkcije 'face_recognition.face_locations(rgb, model =“hog“)'. Parametar 'model=“hog“' označava da se koristi HOG (Histogram of Oriented Gradients) model za detekciju lica.

Druga for petlja prolazi kroz kodirana lica (encodigs) i dodaje ih, zajedno s odgovarajućim imenima u liste i zatim se sve skupa sprema u rječnik koji se sprema na disk.

```
# import the necessary packages
from imutils import paths
import face_recognition
#import argparse
import pickle
import cv2
import os

# our images are located in the dataset folder
print("[INFO] start processing faces...")
imagePaths = list(paths.list_images("dataset"))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
        # extract the person name from the image path
```

```
        print("[INFO] processing image {}/{}".format(i + 1,
                len(imagePaths)))
        name = imagePath.split(os.path.sep)[-2]

        # load the input image and convert it from RGB (OpenCV ordering)
        # to dlib ordering (RGB)
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # detect the (x, y)-coordinates of the bounding boxes
        # corresponding to each face in the input image
        boxes = face_recognition.face_locations(rgb,
                model="hog")

        # compute the facial embedding for the face
        encodings = face_recognition.face_encodings(rgb, boxes)
        # loop over the encodings
        for encoding in encodings:
                # add each encoding + name to our set of known names and
                # encodings
                knownEncodings.append(encoding)
                knownNames.append(name)

# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

4.3.5. Upisivanje otiska prsta (eng. enroll fingerprint)

Ovaj kod služi za upisivanje, brisanje i potpuno resetiranje otisaka prstiju spremljenih na senzoru. Na početku se instaliraju sve potrebne biblioteke i moduli.

```
# Fingerprint enroll,delete,reset


import time
import serial
import pyttsx3


import adafruit_fingerprint
engine = pyttsx3.init()
engine.setProperty('voice', 'croatian')
```

Ovdje se postavlja serijska komunikacija s uređajem preko UART-a, zatim se inicijalizira objekt 'Adafruit_Fingerprint' pomoću te serijske veze.

```
# import board
# If using with Linux/Raspberry Pi and hardware UART:
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)



finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)

##################################################
```

Ova funkcija služi za upisivanje otiska prsta. Prima 'location', što predstavlja lokaciju (ID) gdje će se otisak prsta pohraniti. Prima dvije slike otiska prsta, templira ih i pohranjuje modele. Zatim stvara model i pohranjuje ga na odabranoj lokaciji.

```
def enroll_finger(location):
    """Take a 2 finger images and template it, then store in 'location'"""
    for fingerimg in range(1, 3):
        if fingerimg == 1:
            print("Place finger on sensor...", end="")
             engine.say("Stavi prst na senzor")
            engine.runAndWait()
        else:
            print("Place same finger again...", end="")
            engine.say("Stavi isti prst na senzor")
            engine.runAndWait()
        while True:
            i = finger.get_image()
            if i == adafruit_fingerprint.OK:
                print("Image taken")
                break
            if i == adafruit_fingerprint.NOFINGER:
                print(".", end="")
            elif i == adafruit_fingerprint.IMAGEFAIL:
                print("Imaging error")
                return False
            else:
                print("Other error")
                return False


        print("Templating...", end="")
```

```
        i = finger.image_2_tz(fingerimg)
        if i == adafruit_fingerprint.OK:
            print("Templated")
        else:
            if i == adafruit_fingerprint.IMAGEMESS:
                print("Image too messy")
            elif i == adafruit_fingerprint.FEATUREFAIL:
                print("Could not identify features")
            elif i == adafruit_fingerprint.INVALIDIMAGE:
                print("Image invalid")
            else:
                print("Other error")
            return False

        if fingerimg == 1:
            print("Remove finger")
            engine.say("Odmakni prst")
            engine.runAndWait()
            time.sleep(1)
            while i != adafruit_fingerprint.NOFINGER:
                i = finger.get_image()

    print("Creating model...", end="")
    i = finger.create_model()
    if i == adafruit_fingerprint.OK:
        print("Created")
    else:
        if i == adafruit_fingerprint.ENROLLMISMATCH:
            print("Prints did not match")
```

```
        else:
            print("Other error")
        return False


    print("Storing model #%d..." % location, end="")
    i = finger.store_model(location)
    if i == adafruit_fingerprint.OK:
        print("Stored")
        engine.say("Spremljeno")
        engine.runAndWait()
    else:
        if i == adafruit_fingerprint.BADLOCATION:
            print("Bad storage location")
        elif i == adafruit_fingerprint.FLASHERR:
            print("Flash storage error")
        else:
            print("Other error")
        return False

    return True



##################################################
```

Funkcija koristi input() kako bi dobila valjani broj unutar raspona od 0 do max_number - 1. Korisnika će tražiti da unese ID otiska prsta koji želi upisati, izbrisati ili resetirati.

```
def get_num(max_number):
    """Use input() to get a valid number from 0 to the maximum size
    of the library. Retry till success!"""
    i = -1
```

```
    while (i > max_number - 1) or (i < 0):
        try:
            i = int(input("Enter ID # from 0-{}: ".format(max_number - 1)))
        except ValueError:
            pass
    return i
```

Ovdje je prikazana glavna petlja gdje korisniku nudi 4 opcije, a to su: „e“(enroll(upisivanje) otiska prsta), „d“(delete (brisanje) otiska prsta), „r“ (resetiranje biblioteke otisaka prsta), „q“ (izlaz iz programa).

Također ovdje se koriste i različite funkcije iz Adafruit biblioteke, poput 'read_templates()', 'count_templates()', 'read_sysparam()', "delete_model(location)', 'empty_library()' i drugih, kako bi se čitali, upisivali, brisali ili resetirali otisci prsta.

```
while True:
    print("----------------")
    if finger.read_templates() != adafruit_fingerprint.OK:
        raise RuntimeError("Failed to read templates")
    print("Fingerprint templates: ", finger.templates)
    if finger.count_templates() != adafruit_fingerprint.OK:
        raise RuntimeError("Failed to read templates")
    print("Number of templates found: ", finger.template_count)
    if finger.read_sysparam() != adafruit_fingerprint.OK:
        raise RuntimeError("Failed to get system parameters")
    print("Size of template library: ", finger.library_size)
    print("e) enroll print")
    print("d) delete print")
    print("r) reset library")
    print("q) quit")
    print("----------------")
    c = input("> ")
```

```
    if c == "e":
        enroll_finger(get_num(finger.library_size))
    if c == "d":
        if finger.delete_model(get_num(finger.library_size)) == adafruit_fingerprint.OK:
            print("Deleted!")
            engine.say("Izbrisano")
            engine.runAndWait()
        else:
            print("Failed to delete")
            engine.say("Nije moguce izbrisati")
            engine.runAndWait()
    if c == "r":
        if finger.empty_library() == adafruit_fingerprint.OK:
            print("Library empty!")
            engine.say("Biblioteka prazna")
            engine.runAndWait()
        else:
            print("Failed to empty library")
    if c == "q":
        print("Exiting fingerprint example program")
        engine.say("Izlaz")
        engine.runAndWait()
        raise SystemExit
```

4.3.6. Testiranje

Kao i na početku svakog koda potrebno je napraviti inicijalizaciju, ovdje su konkretno postavljeni GPIO pinovi za senzor vrata, relej, zelenu i crvenu LED diodu. Također se instaliraju sve potrebne biblioteke i moduli kao i ranije.

```
# Diplomski zadatak
# Sigurnosna vrata s antropomorfnim svojstvom prepoznavanja
# Marin Herak


# Import the necessary packages
import pyttsx3
import time
import serial
import RPi.GPIO as GPIO
import adafruit_fingerprint
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import cv2


DOOR_SENSOR_PIN = 17
RELAY_PIN = 18
GREEN_LED_PIN = 22
RED_LED_PIN = 23

GPIO.setmode(GPIO.BCM)
```

```
GPIO.setwarnings(False)
GPIO.setup(DOOR_SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(RELAY_PIN, GPIO.OUT)
GPIO.setup(GREEN_LED_PIN, GPIO.OUT)
GPIO.setup(RED_LED_PIN, GPIO.OUT)
GPIO.output(RED_LED_PIN, GPIO.HIGH)
GPIO.output(RELAY_PIN, GPIO.LOW)
GPIO.output(GREEN_LED_PIN, GPIO.LOW)

# Initialize the text-to-speech engine
engine = pyttsx3.init()

# Set a specific voice (you can choose one from the list)
# For example, setting the voice to "Microsoft David Desktop - English (United States)"
engine.setProperty('voice', 'croatian')

# import board
# If using with Linux/Raspberry Pi and hardware UART:
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)
finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)

#Initialize 'currentname' to trigger only when a new person is identified.
currentname = "unknown"
#Determine faces from encodings.pickle file model created from train_model.py
encodingsP = "encodings.pickle"

# load the known faces and embeddings along with OpenCV's Haar
# cascade for face detection
print("[INFO] loading encodings + face detector...")
```

```
data = pickle.loads(open(encodingsP, "rb").read())

# initialize the video stream and allow the camera sensor to warm up
# Set the ser to the followng
# src = 0 : for the build in single web cam, could be your laptop webcam
# src = 2 : I had to set it to 2 inorder to use the USB webcam attached to my laptop
#vs = VideoStream(src=2,framerate=10).start()
vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)
# start the FPS counter
fps = FPS().start()
name = "Unknown"
########################################################



print("-----------------")
if finger.read_templates() != adafruit_fingerprint.OK:
   raise RuntimeError("Failed to read templates")
print("Fingerprint templates: ", finger.templates)
if finger.count_templates() != adafruit_fingerprint.OK:
  raise RuntimeError("Failed to read templates")
print("Number of templates found: ", finger.template_count)
if finger.read_sysparam() != adafruit_fingerprint.OK:
  raise RuntimeError("Failed to get system parameters")
print("Size of template library: ", finger.library_size)



def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
```

```
    print("Waiting for image...")
    while finger.get_image() != adafruit_fingerprint.OK:
        pass
    print("Templating...")
    if finger.image_2_tz(1) != adafruit_fingerprint.OK:
        return False
    print("Searching...")
    if finger.finger_search() != adafruit_fingerprint.OK:
        return False
    return True


# pylint: disable=too-many-branches
def get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""
    print("Getting image...", end="")
    i = finger.get_image()
    if i == adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")
        elif i == adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
        return False
```

```
    print("Templating...", end="")
    i = finger.image_2_tz(1)
    if i == adafruit_fingerprint.OK:
        print("Templated")
    else:
        if i == adafruit_fingerprint.IMAGEMESS:
            print("Image too messy")
        elif i == adafruit_fingerprint.FEATUREFAIL:
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
        return False

    print("Searching...", end="")
    i = finger.finger_fast_search()
    # pylint: disable=no-else-return
    # This block needs to be refactored when it can be tested.
    if i == adafruit_fingerprint.OK:
        print("Found fingerprint!")
        return True
    else:
        if i == adafruit_fingerprint.NOTFOUND:
            print("No match found")
        else:
            print("Other error")
        return False
```

```
##################################################


def get_num(max_number):
    """Use input() to get a valid number from 0 to the maximum size
    of the library. Retry till success!"""
    i = -1
    while (i > max_number - 1) or (i < 0):
        try:
            i = int(input("Enter ID # from 0-{}: ".format(max_number - 1)))
        except ValueError:
            pass
    return i


def Open_door():
    GPIO.output(RELAY_PIN, GPIO.HIGH)
    GPIO.output(RED_LED_PIN, GPIO.LOW)
    GPIO.output(GREEN_LED_PIN, GPIO.HIGH)
    print("Door open")
    engine.say("Otvorena vrata, dobrodošli Marin")
    engine.runAndWait()
    time.sleep(1)
    return False


def Close_door():
    GPIO.output(RELAY_PIN, GPIO.LOW)
    GPIO.output(RED_LED_PIN, GPIO.HIGH)
```

```
    GPIO.output(GREEN_LED_PIN, GPIO.LOW)
    print("Door close")
    engine.say("Zatvorena vrata")
    engine.runAndWait()
    time.sleep(1)
    return False
```

Ovdje je sada prikazana glavna petlja koja se cijelo vrijeme izvodi i radi na način da kaže da se stane ispred kamere kada primijeti da se u vidnom polju kamere nešto nalazi i zatim pokušava prepoznati lice. Ako lice ne prepozna kreće ponovno, a ako ga prepozna govori sljedeću naredbu koja glasi da se stavi prst na senzor. Ako je otisak prsta ispravan otvaraju se vrata i želi nam dobrodošlicu, a ako se stavi pogrešan prsta 3 ili više puta ponovno se vraća na početak programa odnosno na prepoznavanje lica. Na vratima se nalaze magnetni prekidači (u programu DOOR_SENSOR_PIN) koji provjeravaju kada se vrata zatvore i onda se vrata zaključavaju ne prije.

```
while True:
    face_detected = False
    fingerprint_attempts = 0
    # Wait until a face s detected
    while not face_detected:

        # grab the frame from the threaded video stream and resize it
        # to 500px (to speedup processing)
        frame = vs.read()
        frame = imutils.resize(frame, width=300)
        # Detect the fce boxes
        boxes = face_recognition.face_locations(frame)

        if boxes:
            # Face detected, break out of the loop
            engine.say("Dobrodošli, molim stanite ispred kamere")
```

```
        engine.runAndWait()
        time.sleep(2)
        face_detected = True

    else:
        # No face detected, display a massage
        engine.say("Čekaj")
        engine.runAndWait()
        print("Waiting for face")
        time.sleep(2)

    # Face Recognition
    # grab the frame from the threaded video stream and resize it
    # to 500px (to speedup processing)
    frame = vs.read()
    frame = imutils.resize(frame, width=300)
    # Detect the fce boxes
    boxes = face_recognition.face_locations(frame)
    # compute the facial embeddings for each face bounding box
    encodings = face_recognition.face_encodings(frame, boxes)
    names = []

    # loop over the facial embeddings
    for encoding in encodings:
        # attempt to match each face in the input image to our known
        # encodings
        matches = face_recognition.compare_faces(data["encodings"], encoding)
        name = "Unknown" #if face is not recognized, then print Unknown
        #engine.say("Nepoznato lice, zao mi je pokusaj ponovno")
```

```
            #engine.runAndWait()
            # check to see if we have found a match
            if True in matches:
                # find the indexes of all matched faces then initialize a
                # dictionary to count the total number of times each face was matched
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}



                # loop over the matched indexes and maintain a count for each recognized face face
                for i in matchedIdxs:
                    name = data["names"][i]
                    counts[name] = counts.get(name, 0) + 1

                # determine the recognized face with the largest number
                # of votes (note: in the event of an unlikely tie Python
                # will select first entry in the dictionary)
                name = max(counts, key=counts.get)

                #If someone in your dataset is identified, print their name on the screen
                if currentname != name:
                    print("Face recognized:", name)
                    engine.say("Lice prepoznato")
                    engine.runAndWait()
                    break
            else:
                engine.say("Nepoznato lice, žao mi je pokušaj ponovno")
                engine.runAndWait()
                face_detected = False
```

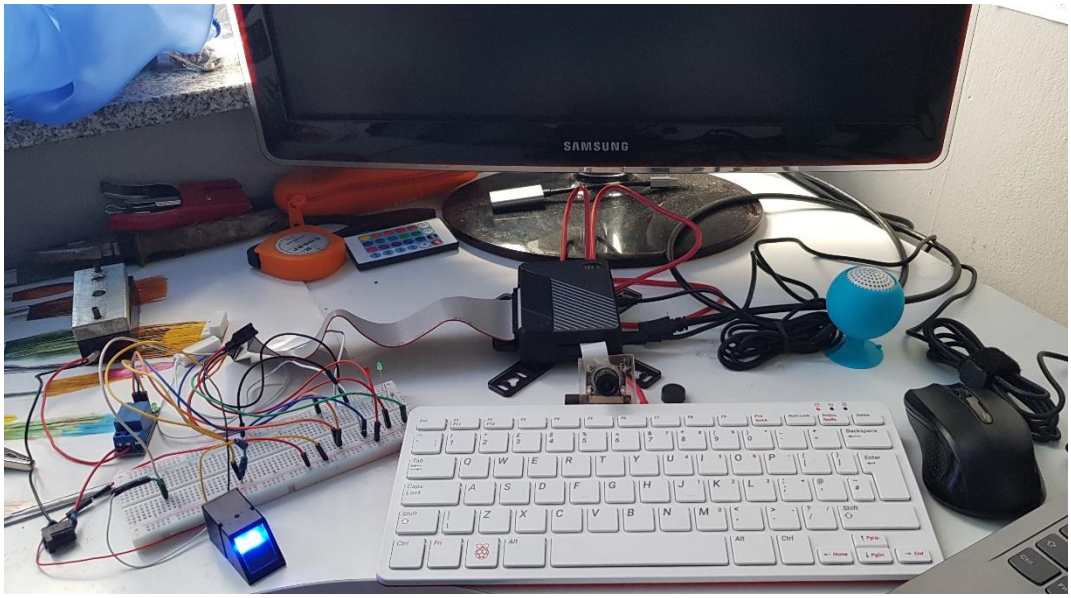
```
            break

    if face_detected:
        # Fingerprint Recognition
        if currentname != name:
            while fingerprint_attempts < 3:
                engine.say("Stavite prst na senzor")
                engine.runAndWait()
                if get_fingerprint():
                    print("Detected #", finger.finger_id, "with confidence", finger.confidence)

                    # If the door is locked, unlock it
                    engine.say("Otisak prsta prepoznat")
                    engine.runAndWait()
                    Open_door()
                    print("Door unlocked")

                    time.sleep(2)
                    if GPIO.input(DOOR_SENSOR_PIN):

                        while GPIO.input(DOOR_SENSOR_PIN):
                            # Wait until the door is closed
                            time.sleep(1)
                    Close_door()
                    print("Door locked")

                    name = "unknown"
                    fingerprint_attemts = 0
                    break
```

```
            else:
                print("Finger not found")
                engine.say("Nepoznati otisak prsta")
                engine.runAndWait()
                Close_door()
                print("Door locked")

                fingerprint_attempts += 1
                if fingerprint_attempts == 3:
                    # If fingerprint not recognized tree times, go back to face recognition
                    engine.say("Previše krivih otisaka")
                    engine.runAndWait()
                    currentname = "unknown"
                    break



            # loop over the recognized faces
        for ((top, right, bottom, left), name) in zip(boxes, names):
            # draw the predicted face name on the image - color is in BGR
            cv2.rectangle(frame, (left, top), (right, bottom),(0, 255, 225), 2)
            y = top - 15 if top - 15 > 15 else top + 15
            cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,.8, (0, 255,
255), 2)

        # display the image to our screen
        cv2.imshow("Facial Recognition is Running", frame)
        key = cv2.waitKey(1) & 0xFF
```

```
            # quit when 'q' key is pressed
            if key == ord("q"):
                break

            # update the FPS counter
            fps.update()

# stop the timer and display FPS informationqqqqqqq
fps.stop()
print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
GPIO.cleanup()
cv2.destroyAllWindows()
vs.stop()

# Add a delay to avoid high CPU usage
time.sleep(1)
```

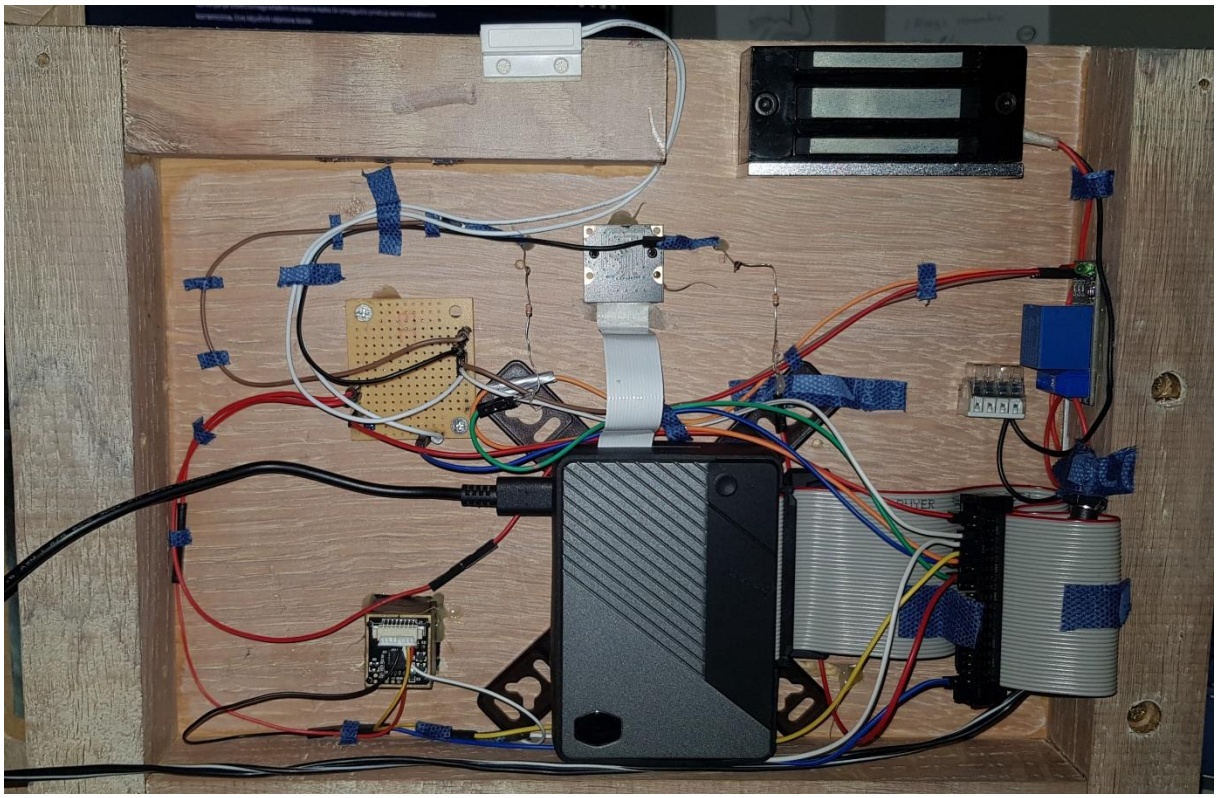
5. EKSPERIMENTALNI REZULTATI

Svaku teoriju potrebno je potvrditi eksperimentom. Potrebno je provjeriti ispravnost komponenata, spojeva i njihova funkcionalnost. Također prije samog spajanja potrebno je programirati Raspberry Pi kako bismo odredili protokol komunikacije prema ostalim modulima i definirali sve potrebne funkcije. Taj dio obavljen je u prethodnom poglavlju. Nakon programiranja u svrhu potvrde teorije eksperimentom koristila se eksperimentalna ploča na kojoj su pomoću žica za spajanje i već ranije spomenutih modula spojeni svi elementi u funkcionalnu cjelinu. Prikazani eksperimentalni spoj može se vidjeti na slici [Slika 30.]. Ovime je potvrđen rad samog sklopa koji je detaljnije bio opisan u poglavlju programiranja.



Slika 30. Eksperimentalni spoj

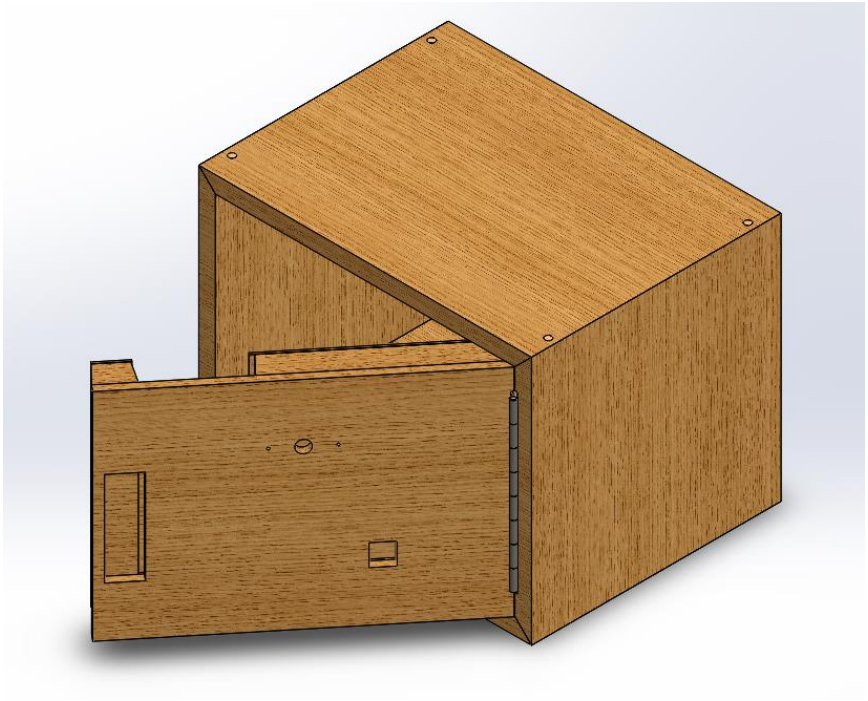
U izradi PCB pločice korištena je pločica s rupicama. Takav pristup ne zahtijeva izradu tiskane pločice čija cijena izrade u malim serijama, u ovom slučaju jednog komada, može biti visoka.



Slika 31. Izgled sklopa unutar kučišta

Nakon izrade prototipne pločice potrebno je za nju izraditi kućište koje je prvo konstruirano u programu SolidWorks (verzija 25) čiji je model prikazan na slici [Slika 32.]. SolidWorks je CAD program za crtanje 3D modela na Microsoft Windows računalu. Omogućuje modeliranje dijelova i spremanje na mnogo različitih načina.

Gotovo kućište prikazano je slikom [Slika 33.]. Izvedeno je drveno kućište koje je cjenovno povoljnije i prikazuje zapravo prototip sefa. Ovakvo gotovo kućište odnosno cijeli sklop je izrađen za cijenu od oko 300 eura. Kako sam sklop radi prikazano je na kratkom videu [20].



Slika 32. Model kučišta

Slika 33. Gotov sklop

6. ZAKLJUČAK

U ovom diplomskom radu, razvijen je sustav sigurnosnih vrata s antropomorfnim svojstvom prepoznavanja, koji integrira prepoznavanje lica i otiska prsta kako bi osigurao siguran pristup određenim prostorima. Cilj ovog rada bio je prevladati nedostatke tradicionalnih beskontaktnih sustava identifikacije, poput gubitka ključa-identifikatora, mogućnosti kopiranja identifikatora te potrebe za vođenjem evidencije o vezama između identifikatora i osobe.

Implementirani sustav koristi kameru za prepoznavanje lica, senzor otiska prsta za identifikaciju jedinstvenih osobnih karakteristika, elektromagnetsku bravu za kontrolu pristupa te dodatne komponente za upravljanje i kontrolu sustava. Kroz različite petlje prepoznavanja, sustav osigurava da samo ovlašteni korisnici mogu proći kroz vrata, čime se povećava razina sigurnosti.

Dodatno, implementirana je glasovna komunikacija prema korisnicima kako bi se pružila dodatna razina interakcije i informiranja. TTS sustav obavještava korisnike o događajima poput prepoznavanja lica i otiska prsta, čime se poboljšava korisničko iskustvo.

U procesu konstrukcije mehanizma sigurnosnih vrata, odabrane su odgovarajuće komponente, uključujući kameru, čitač otisaka prstiju, aktuatore te upravljački kontroler. Razvijen je i upravljački program koji koordinira rad svih dijelova sustava te omogućuje integraciju prepoznavanja lica i otiska prsta.

Osim tehničkih aspekata, diplomski rad uključuje i predloženo glasovno sučelje prema korisniku, prateći moderne smjernice u korisničkom sučelju. Predloženo sučelje doprinosi jednostavnom i intuitivnom korištenju sustava, čime se olakšava interakcija korisnika sa sustavom.

Sve navedene komponente sustava, od konstrukcije mehanizma vrata do programiranja upravljačkog sustava, predstavljaju cjelovito rješenje koje može pridonijeti unapređenju sigurnosti i praktičnosti u različitim okruženjima. Kroz ovaj rad, stvoren je temelj za daljnji razvoj sličnih sustava i pristupa unutar područja kontrole pristupa i sigurnosti prostora.

LITERATURA

[1] Kraut, B.: Strojarski priručnik, Tehnička knjiga Zagreb, 1970.

[2] M. Turk, A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neurosicence, Vol. 3, No. 1, 1991, pp. 71-86

[3] NEC web stranica, (https://www.nec.com/en/global/solutions/biometrics/index.html), posjećeno 20.11.2023.

[4] FaceKey web stranica, (https://www.facekey.com/), posjećeno 20.11.2023.

[5] AINDRA SYSTEMS, World's Top 10 Usage of Face Recognition Technology #2015 (https://aindrasystems.wordpress.com/2015/08/26/worlds-top-10-usage-of-face-recognition-technology-2015/), posjećeno 7. ožujka 2023.

[6] T.Radford, How we recognise faces (https://www.theguardian.com/uk/2004/dec/13/sciencenews.research), posjećeno 7. ožujka 2023.

[7] B. Virdee-Chapman, Inspiring Uses of Facial Recognition in the Real World (https://www.kairos.com/blog/), posjećeno 7. ožujka 2023.

[8] Web stranica organizacije Helping faceless (https://apkcombo.com/helping-faceless/in.amolgupta.helpingfaceless/), posjećeno 10. studeni 2023.

[9] Web stranica howstuffworks (https://science.howstuffworks.com/fingerprinting3.htm), posjećeno 10. studenog 2023.

[10] Web stranica skenera otiska prsta, (https://pctown.co.nz/kako-radi-skener-otiska-prsta/#google_vignette), posjećeno 10. listopada 2023.

[11] Web stranica skenera otiska prsta, (https://www.racunalo.com/skeneri-otisaka-prstiju-u-elektronickim-uredajima-sasvim-jednostavno-objasnjenje-tehnologije/)

[12] Web stranica glasovne komunikacije, (https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction)

[13] Web stranica Hog-a, (https://learnopencv.com/histogram-of-oriented-gradients/), posjećeno 20.11.2023.

[14] Web stranica TTS-a, (https://www.respeecher.com/blog/what-is-text-to-speech-tts-initial-speech-synthesis-explained)

[15] Web stranica zvučnika, (https://hr.wikipedia.org/wiki/Zvu%C4%8Dnik)

[16] Web stranica tvrtke Raspberry Pi (https://www.raspberrypi.com/), posjećeno 21. ožujka 2023.

[17] Senzor otiska prsta, (https://soldered.com/hr/proizvod/senzor-otiska-prsta/), posjećeno 10. studenoga 2023.

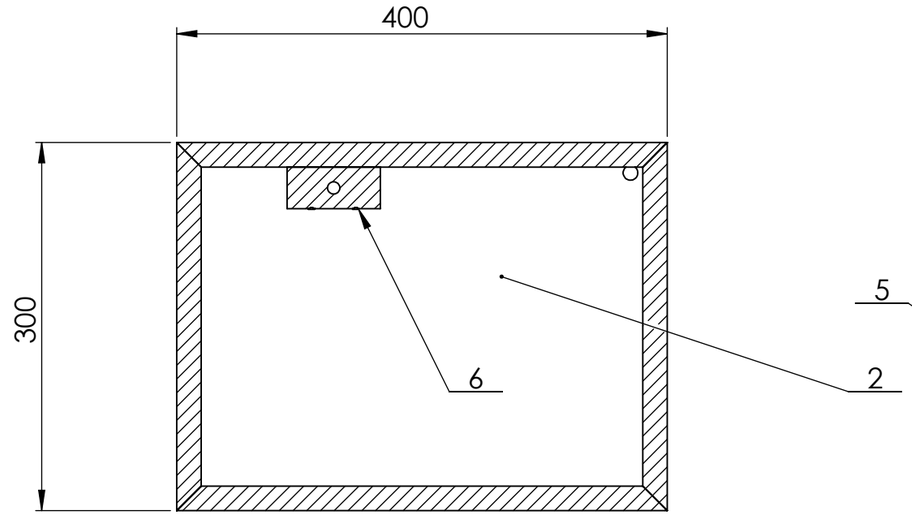
[18] Relej, (https://www.ad-electronic.hr/index.php/elektronika-i-elektrika%20-%2023/modul-relej-5v-10a-1kom8422486-detalji), posjećeno 21. ožujka 2023.

[19] Elektromagnetski lokot (https://soldered.com/hr/proizvod/elektromagnetski-lokot-za-vrata/), posjećeno 21. ožujka 2023.

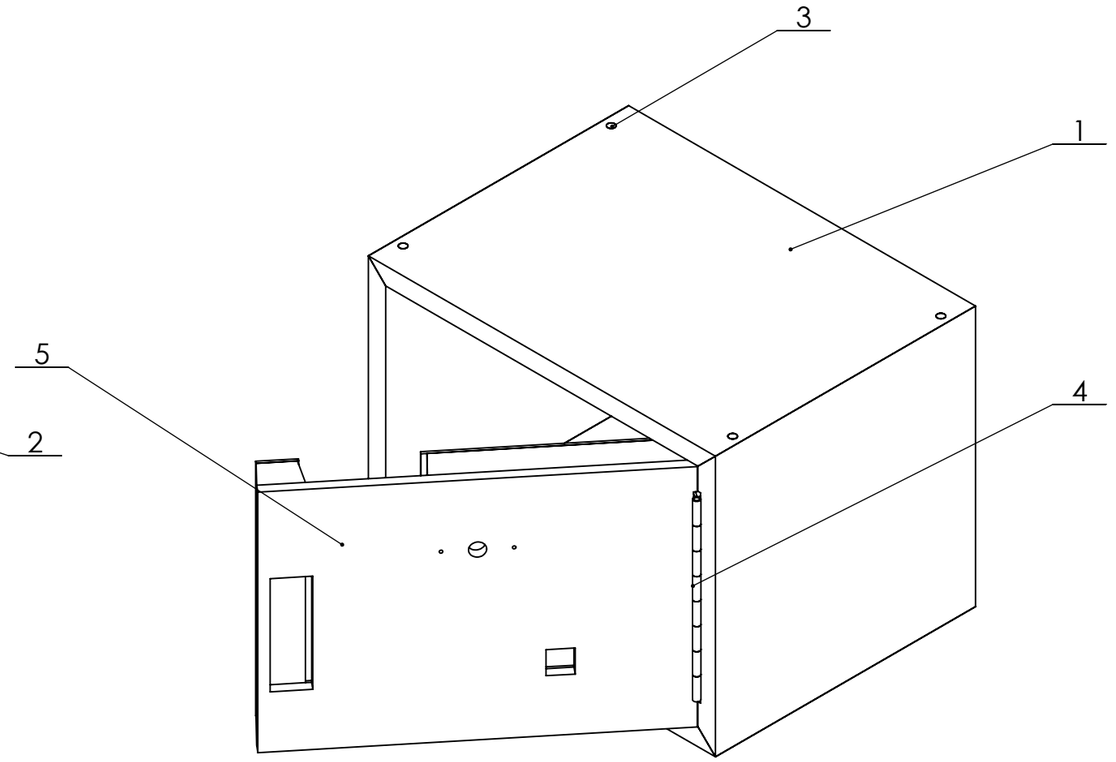
[20] YouTube video rada sklopa, (https://www.youtube.com/shorts/ZSFsuG7ilm4)

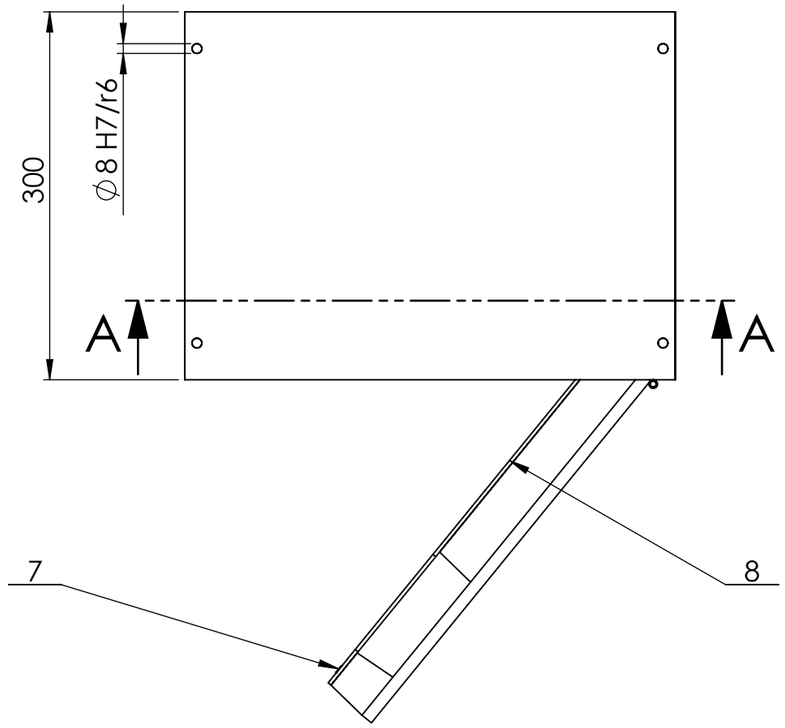
PRILOZI

I. Tehnička dokumentacija



Presjek A-A





Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa
8	Poklopac za vrata	1	SB-0000-04			
7	Samourezni vijak 3,5x25	12	EN 14566			
6	Samourezni vijak 3,5x45	2	EN 14566			
5	Vrata	1	SB-0000-03			
4	Spojna šarka	1	DIN 7954 D			
3	Tipla drvena	8	DIN 68150		8x35	
2	Zadnji poklopac	1	SB-0000-02			
1	Baza	1	SB-0000-01			

Broj naziva - code		Datum	Ime i prezime	Potpis
	Projektirao	09.09.2023.	Marin Herak	
	Razradio	09.09.2023.	Marin Herak	
	Crtao	09.09.2023.	Marin Herak	
	Pregledao			
	Mentor			

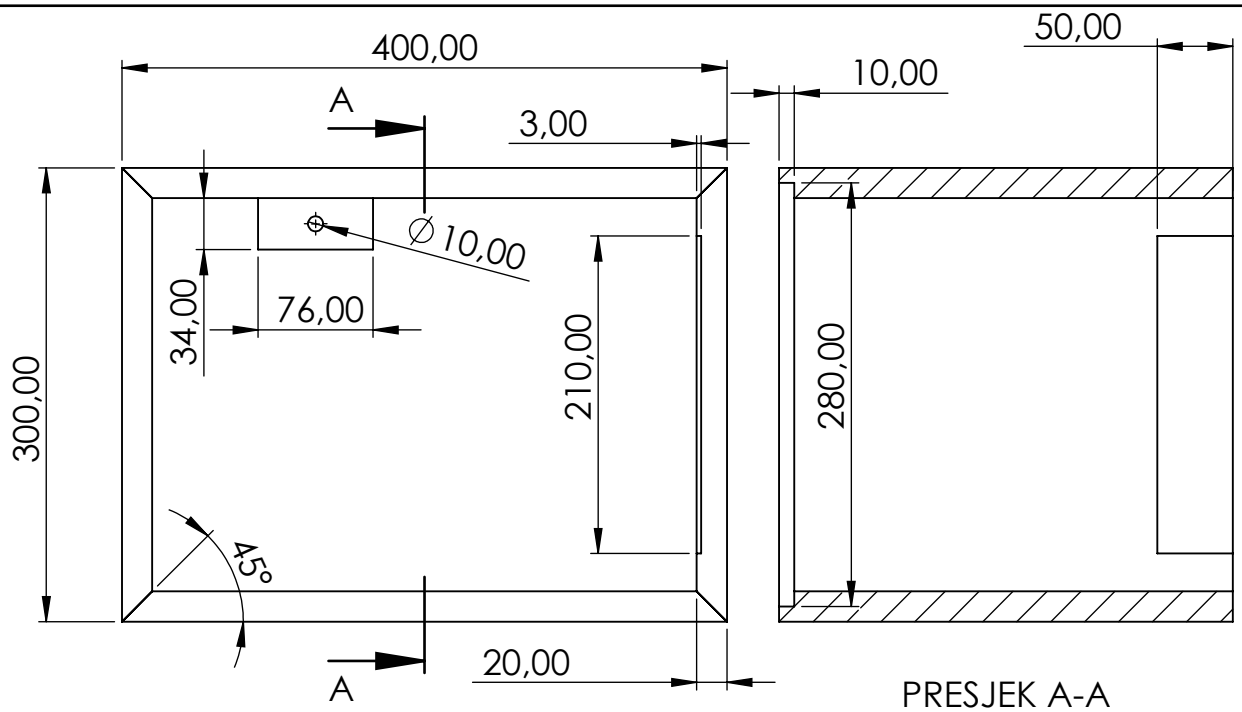
FSB Zagreb

ISO - tolerancije		Objekt:	Objekt broj:
∅8 H7	+0,015 / 0		R. N. broj:
∅8 r6	+0,028 / +0,019	Napomena:	Kopija
∅8 H7/r6	-0,004 / -0,028	Materijal:	Masa: 8 kg

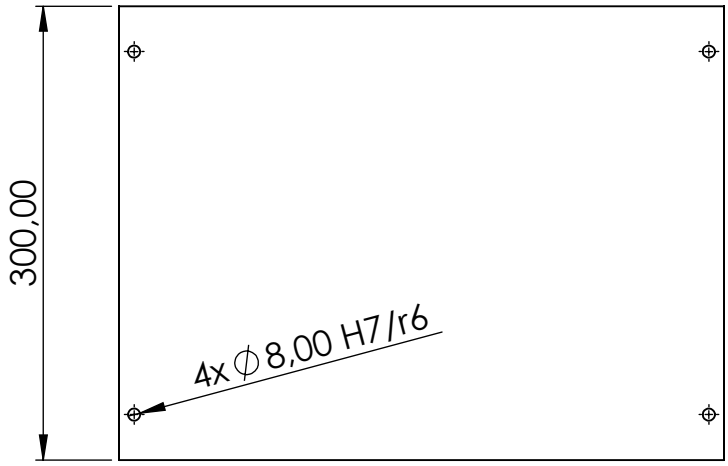
Safe box

Mjerilo originala	Naziv:	Pozicija:	Format: A3
M 1:5	Sef	00	Listova: 1
	Crtež broj: SB-0000-00		List: 1

0 10 20 30 40 50 60 70 80 90 100



PRESJEK A-A
M 1:5



		Datum	Ime i prezime	Potpis
Broj naziva - code	Projektirao	09.09.2023.	Marin Herak	
	Razradio	09.09.2023.	Marin Herak	
	Crtao	09.09.2023.	Marin Herak	
	Pregledao			

FSB Zagreb

ISO - tolerancije			
∅8 H7	+0,015	0	
∅8 r6	+0,028	+0,019	
∅8 H7/r6	-0,004	-0,028	

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Materijal:

Masa:

Naziv: Baza

Pozicija: 1

Format: A4

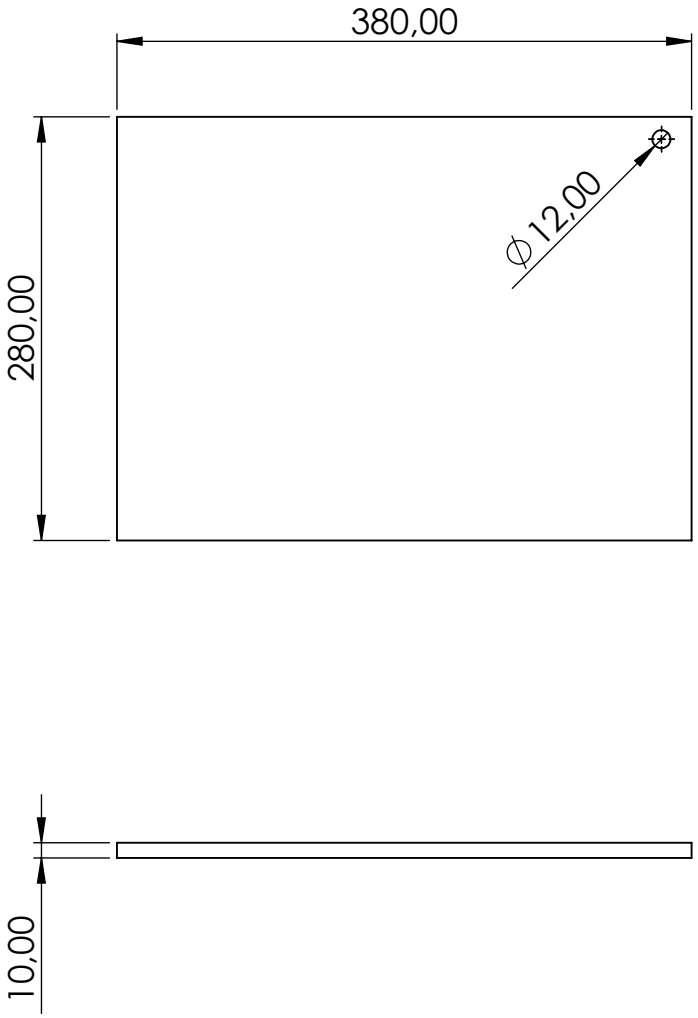
Mjerilo originala M 1:5

Listova: 5

Crtež broj: SB-0000-01

List: 2

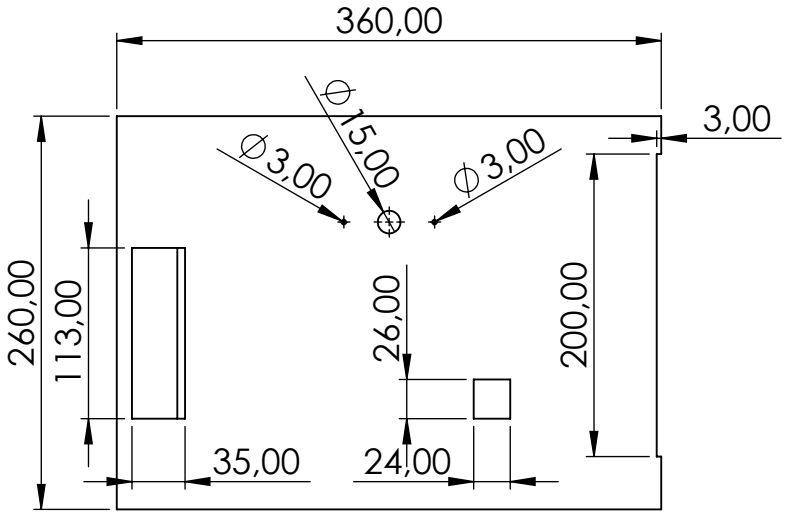
Design by CADLab

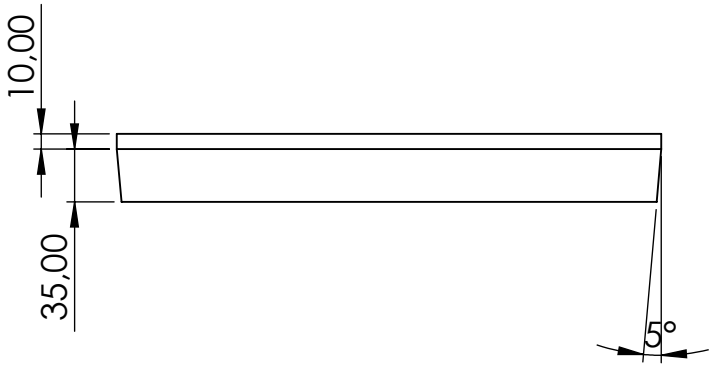


	Datum	Ime i prezime	Potpis
Projektirao	09.09.2023.	Marin Herak	
Razradio	09.09.2023.	Marin Herak	
Crtao	09.09.2023.	Marin Herak	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala M 1:5	Naziv: Zadnji poklopac; Pozicija: 2; Format: A4; Listova: 5; List: 3
	Crtež broj: SB-0000-02

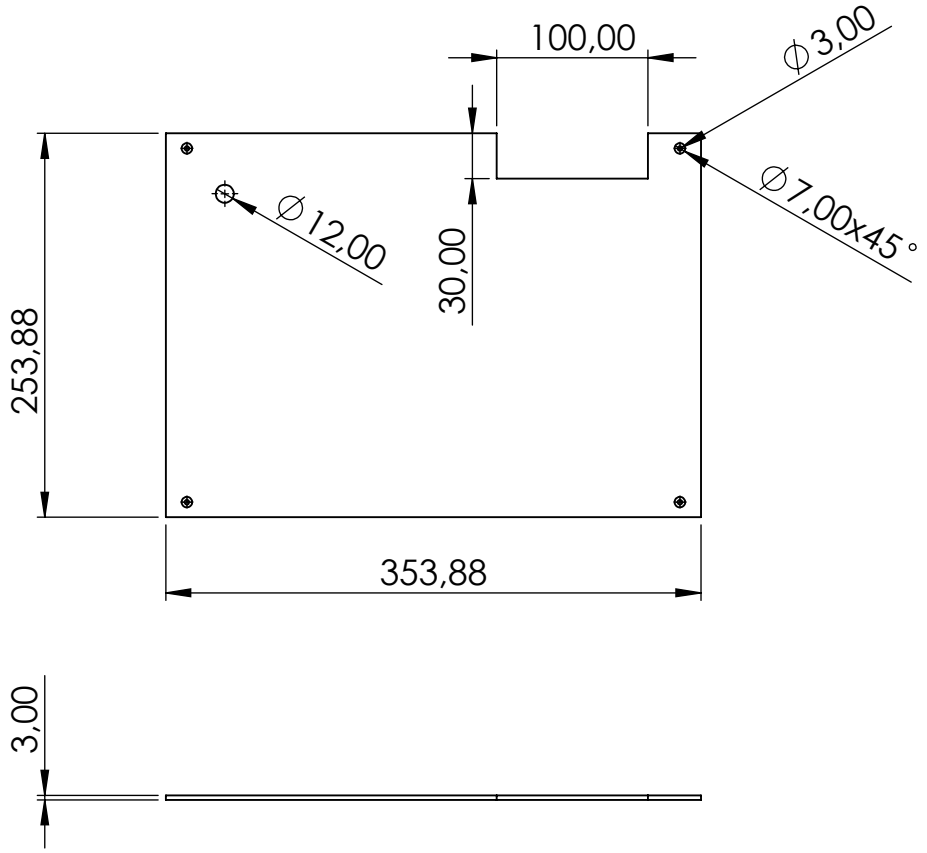




	Datum	Ime i prezime	Potpis
Projektirao	09.09.2023.	Marin Herak	
Razradio	09.09.2023.	Marin Herak	
Crtao	09.09.2023.	Marin Herak	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala: M 1:5	Naziv: Vrata
Pozicija: 5	Format: A4
Listova: 5	List: 4
Crtež broj: SB-0000-03	



	Datum	Ime i prezime	Potpis
Projektirao	09.09.2023.	Marin Herak	
Razradio	09.09.2023.	Marin Herak	
Crtao	09.09.2023.	Marin Herak	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala: M 1:5	Naziv: Poklopac za vrata; Pozicija: 8
Crtež broj: SB-0000-04	Format: A4; Listova: 5; List: 5

Design by CADLab