ŠOLSKI CENTER VELENJE
ELEKTRO IN RAČUNALNIŠKA ŠOLA
Trg mladosti 3, 3320 Velenje

MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

ODKLEPANJE VRAT S POMOČJO PREPOZNAVANJA OBRAZA

Tematsko področje: tehnika

Avtorji:
Tim Jevšenak, 3. letnik
Jure Kotnik, 3. letnik
Aleš Prosenjak, 3. letnik

Mentorja:
Rok Urbanc
Uroš Remenih, inž. informatike

Velenje, 2020

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniška šola, 2020.

Mentorja: Rok Urbanc
Uroš Remenih, inž. informatike

Datum predstavitve: marec 2020

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD ŠC Velenje, šolsko leto 2019/2020

KG program/avtomatizacija/varnost

AV JEVŠENAK, Tim / KOTNIK, Jure / PROSENJAK, Aleš

SA URBANC, Rok / REMENIH, Uroš

KZ 3320 Velenje, SLO, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2020

LI 2022

IJ **ODKLEPANJE VRAT S POMOČJO PREPOZNAVANJA OBRAZA**

TD Raziskovalna naloga

OP VI, 29 str., 1 pregl., 0 graf., 25 sl., 0 pril., 7 vir.

IJ SL

JI sl/en

AI Raziskovalna naloga je osredotočena na tehnologijo prepoznave obraza. Ideja, da bi na šoli imeli vrata, ki se odklepajo s to vrsto tehnologije, se nam je že od samega začetka zdela zanimiva. S tem razlogom smo bili zelo osredotočeni na izdelavo izdelka. V sledeči dokumentaciji je zapisan celoten postopek raziskave, od načina razmišljanja do poteka dela.

KEY WORDS DOCUMENTATION

ND ŠC Velenje, school year 2019/2020

CX program/ automation/ security

AU JEVŠENAK, Tim / KOTNIK, Jure / PROSENJAK, Aleš Prosenjak

AA HRASTNIK, Gregor / REMENIH, Uroš

PP 3320 Velenje, SLO, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2020

PY 2022

TI **ODKLEPANJE VRAT S POMOČJO PREPOZNAVANJA OBRAZA**

DT Research work

NO VI, 29 p., 1 tab., 0 graf, 25 fig., 0 ann., 7 ref.

LA Sl

AL sl/en

AB This research paper focuses on facial recognition technology. The idea of our school having a smart lock, which uses facial recognition to unlock, was interesting to us from the very beginning. For this reason we were focused on actually making the lock. Despite the fact that all of us lacked any form of pre-existing knowledge of this technology, we managed to build and test the lock. Our thought process and research methods are all in the following documentation.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

KRATICE

ŠC – Šolski center

Inž. – inženir

Univ. – univerzitetni

Dipl. – diplomat

Rač. – računalništva

Inf. – informatike

MP – mega pixli

mm - mili meter

OS – operacijski sitem

GB – giga bajt

RAM – read only memory

HDMI – High-Definition Multimedia Interface

USB – universal serial bus

GPIO – general-purpose input/output

A – amper

GUI - graphical user interface

RGB – rows, columns, colors

LBPH - local binary patterns histograms

1. UVOD

Opazili smo, da se tehnologija na večini področij zelo hitro razvija. A eno področje, ki se še ni tako dobro razvilo, je varnost. Zato smo se odločili, da bomo posodobili zastarele ključavnice. To smo storili tako, da smo nadomestili stare ključavnice, za katere si potreboval ključ, in to z novim sistemom prepoznave obraza.

1.1 Hipoteze

1. Za izdelavo pametne ključavnice ne potrebuješ predznanja o delovanju tehnologij prepoznave obraza.

2. Cena izdelane ključavnice ne bo presegla 100 €.

3. Tehnologija bo sposobna ločiti obraz in sliko obraza.

1.2 Namen in cilj

Naš namen je bil izdelati pametno ključavnico, ki deluje na principu prepoznavanja obraza. Na tak način smo upali izboljšati še tako vsakdanje in prav nič posebno opravilo odklepanja vrat. To bi odpravilo potrebo po ključih in bi dodalo naši tehniški šoli novo tehnološko izboljšavo. Prav tako pa smo si za cilj zadali širjenje naših znanj z učenjem zanimive tehnologije prepoznavanja obrazov, v katero smo se poglobili.
Izdelano ključavnico bomo montirali na eno izmed učilnic v petem nadstropju naše šole in opazovali, kako varen je takšen način odklepanja vrat.

2. PREGLED OBJAV

2.1 Varnost

Varnost je oblika svobode ali odpora proti potencialni škodi in nezaželenim prisilnim spremembam, ki jih povzročajo drugi. Upravičenci varnosti so lahko osebe in družbene skupine, predmeti in ustanove, ekosistemi ali kateri koli drug subjekt ali pojav, ki je izpostavljen neželenim spremembam. (vir 1)

2.2 Biometrija

Biometrija je nek način prepoznave ljudi na podlagi njihovih telesnih, fizioloških ter vedenjskih značilnosti, ki jih ima vsak posameznik, so edinstvene in stalne za vsakega posameznika. Z njimi je možno določiti posameznika. Biometrija je danes samo ena od načinov preverjanja identitete posameznika. Preostale načine že poznamo več časa (kratice, PIN-koda, osebno geslo). Biometrija ima pred drugimi načini veliko prednost, saj se kartice lahko izgubijo, PIN in gesla se pozabijo. Biometrične značilnosti pa ostanejo večne, ne morejo se izgubiti ali pa se pozabiti, težko jih je reproducirati ali prenesti na drugo osebo.
Zaradi vse več zahtev po avtomatizaciji, natančnem in hkrati hitrem potrjevanju identitete posameznika, tudi uporaba biometrije narašča. (vir 2)

2.3 Sistem za prepoznavanje obraza

Sistem za prepoznavanje obraza je tehnologija, ki omogoča prepoznavo ali potrditev identitete osebe z digitalne slike ali iz videovira. Sistem deluje tako, da primerja slike izbranih lastnosti obraza dane osebe s slikami obrazov, shranjenih v bazi podatkov. (vir 3)

3. METODA DELA

3.1 Materiali

	IZDELEK	CENA
1 x	Raspberry pi 3 model b	38 €
1 x	Raspberry camera Rev1.3	7 €
1 x	Podatkovni kabel za Raspberry pi kamero	5 €
1 x	Pametna ključavnica	40 €
1 x	Rele	2 €
3 x	Povezovalni kabel	0.20 €

Tabela 1: Materiali, potrebni za izdelavo

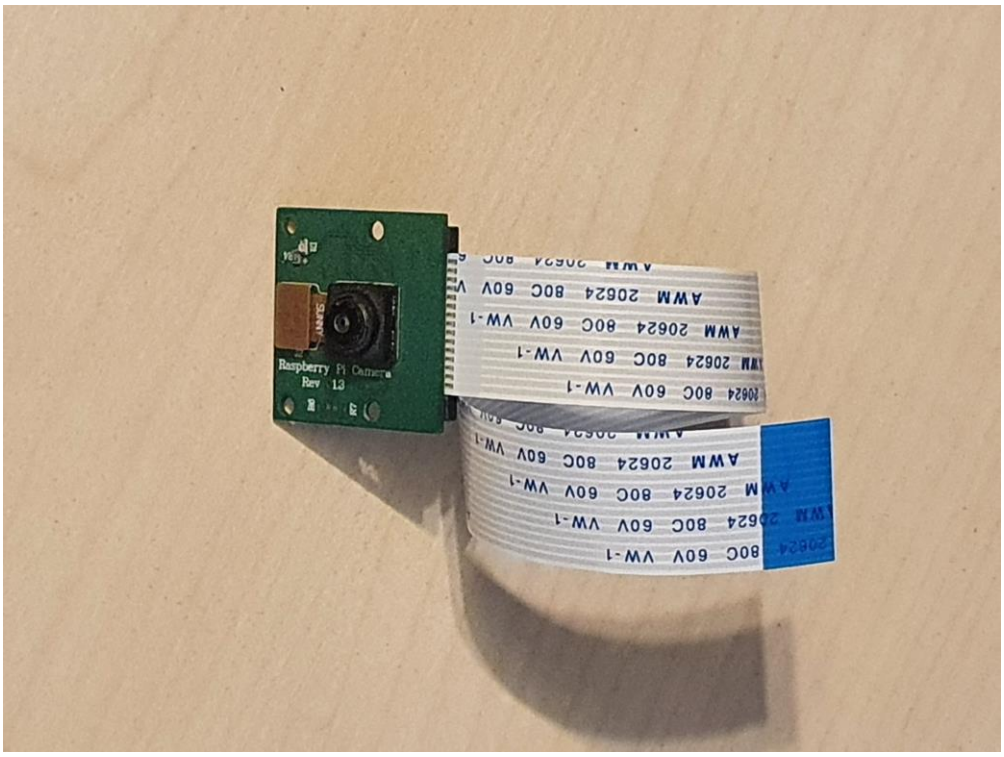
3.1.1 Raspberry pi 3 model b

Raspberry pi 3 model b je eden od modelov mikroračunalnikov Raspberry. Ti so bili razviti, da bi spodbujali učenje računalniške znanosti v šolah. Raspberry pi 3 model b ima Quad Core 1.2 Hz Broadcom Bcm2837 64-bitni procesor, 1 GB RAM-a, 40 GPIO pinov, štiri USB 2 vhode, HDMI-vhod, videovrata, CSI-vrata za Raspberry kamero, DSI-vrata za Raspberry ekran na dotik in pa mikro SD-vrata.

Slika 1: Raspberry pi 3 b

3.1.2 Raspberry camera Rev1.3

Je kamera za mikroračunalnike Raspberry. Ima resolucijo 5 MP, dimenzije 25 x 23 x 8 mm. Podpira videoformate 1080p, 720p in 480p ter CSI vrsto vmesnika. Je popolnoma kompatibilna z Raspberry pi 3 model b.



Slika 2: Raspberry camera Rev 1.3 s kablom za kamero

3.1.3 Pametna ključavnica

Je električna ključavnica, ki ima nekakšen spominski mehanizem. Ta mehanizem spusti manjši signal napetosti, ki odklene ključavnico. Ključavnica ostane odklenjena dokler se vrata ne odprejo. Ključavnica pa potrebuje tudi zunanji vir napetosti.

Slika 3: Pametna ključavnica

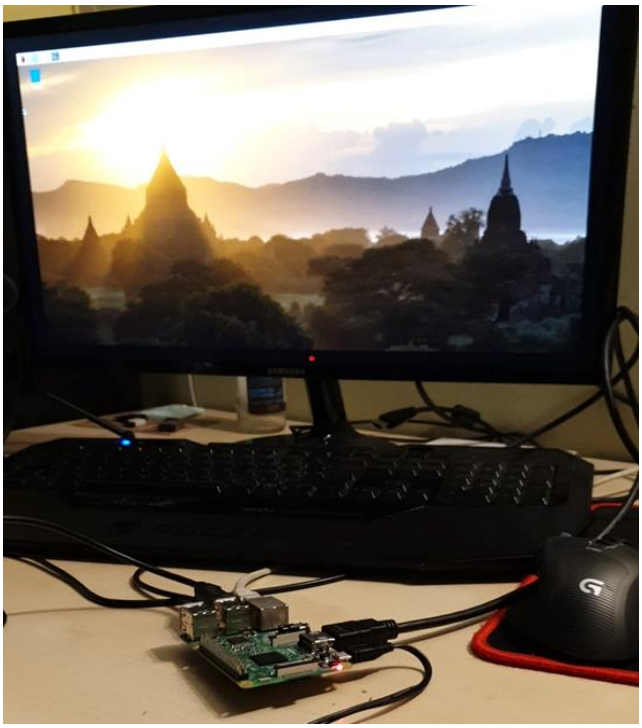
3.1.4 Rele

Rele je elektromagnetno stikalo, ki ga krmili tok skozi magnetno navitje. Lahko nadzira signale 5 V in 12 V. Nadzira lahko izmenično in enosmerno napetost.

Slika 4: Rele

3.1.5 Preostali pripomočki

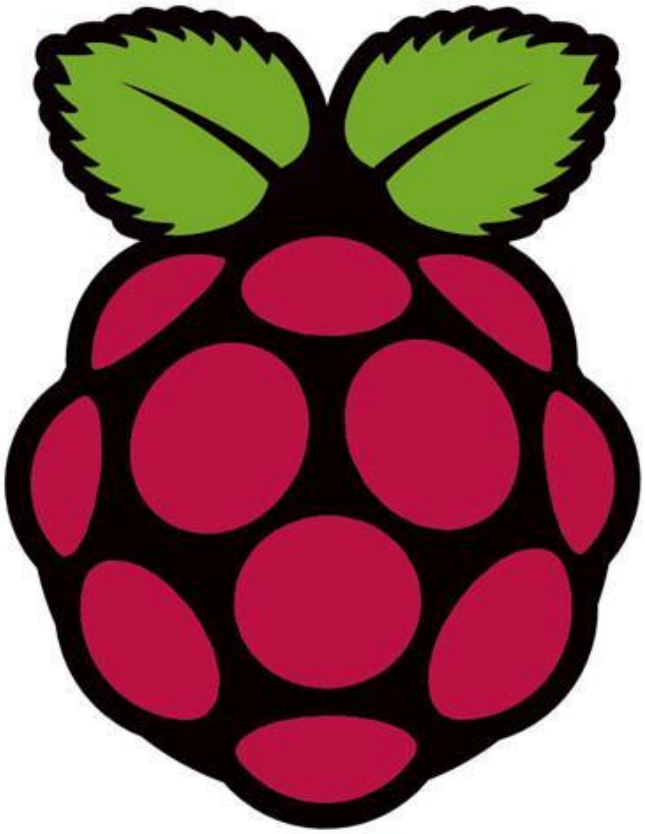
Pri izdelavi projekta smo potrebovali še katerikoli monitor, ki je imel HDMI-vhod, HDMI-kabel za povezavo med monitorjem in Raspberyjem, miško in tipkovnico z USB-priklopom. Potrebovali pa smo tudi mikro USB-napajalnik (2.1 A).



Slika 5: Raspberry, povezan s perifernimi napravami

3.2 Priprava raspberry pi 3 model b

Najprej smo se morali odločiti, kateri operacijski sistem bomo uporabljali na Raspberryju. Izbrali smo OS NOOBS, saj nam je ta omogočal upravljanje z Raspberryjem tudi, ko ni imel internetne povezave. NOOBS je preprost operacijski sistem za uporabo, ki vsebuje Raspbian in LibreELEC. Ponuja tudi izbor alternativnih operacijskih sistemov, ki se nato prenesejo in namestijo. Odločili smo se in prenesli NOOBS.zip datoteko, ki smo jo razširili na SD-kartico. To smo vstavili v Raspberry pi 3 model b ter naložili nanj operacijski sistem NOOBS. Za tem smo posodobili naš operacijski sistem na najnovejšo verzijo. Nato smo naložili depedence. (vir 4)



Slika 6: NOOBS logo [1]

```
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev libgtk-3-dev
sudo apt-get install libatlas-base-dev gfortran
```

Slika 7: Ukazi za namestitev depedenc

Naložili smo še Python 3 orodje za razvoj in pip orodje. Python je tolmačni programski jezik, uporablja se za razvoj različnih aplikacij (GUI, internetne aplikacije …). Pip orodje pa je standardni sistem za upravljanje s paketi, uporablja se za namestitev in upravljanje programskih paketov. (vir 5)

OpenCV (Open Source Computer Vision Library) je knjižnica programske opreme z odprtokodnim računalniškim vidom in strojnim učenjem. OpenCV je bil izdelan za zagotavljanje skupne infrastrukture za aplikacije računalniškega vida in za pospešitev uporabe strojne percepcije v komercialnih izdelkih.

V knjižnici je več kot 2500 optimiziranih algoritmov, ki vključujejo obsežen nabor klasičnih in najsodobnejših algoritmov računalniškega vida in strojnega učenja. Ti algoritmi se lahko uporabljajo za zaznavanje in prepoznavanje obrazov, prepoznavanje predmetov, razvrščanje človeških dejanj v videoposnetke, sledenje premikom kamere, sledenje premikajočim se objektom, izvlečenje 3D-modelov predmetov, izdelavo 3D-oblakov, točk iz stereokamer, lepljenje slik, za ustvarjanje visoke ločljivosti podobe celotnega prizora, iskanje podobnih slik iz baz slik, odstranitev rdečih oči s slik, sledenje premikov oči, prepoznavanje scenografije in postavitev oznak. OpenCV ima več kot 47 tisoč uporabnikov in ocenjeno število prenosov te knjižnice presega 18 milijonov.

Skupaj z uveljavljenimi podjetji, kot so Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, ki uporabljajo to knjižnico, obstajajo številna začetna podjetja, kot so Applied Minds, VideoSurf in Zeitera, ki veliko uporabljajo to knjižnico.

Ima vmesnike C ++, Python, Java in MATLAB ter podpira Windows, Linux, Android in Mac OS. OpenCV se večinoma nagiba k aplikacijam za vid v realnem času. OpenCV je izvirno napisan v jeziku C ++ in ima šablonski vmesnik, ki brez težav deluje s STL-shrambami podatkov. (vir 6)

Slika 8: OpenCV logo [2]

```
sudo apt-get install python3 python3-setuptools python3-dev
```

Slika 9: Ukaz za namestitev Python 3 orodja za razvoj

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
```

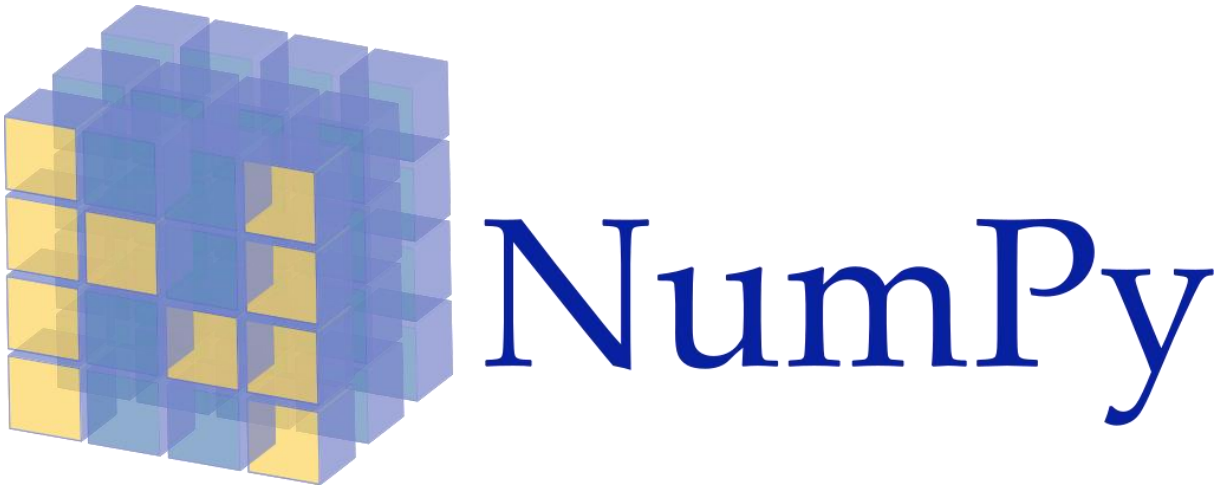
Slika 10: Ukaza za namestitev pip orodja

Po uspešni namestitvi Python in pip orodja pa je bilo potrebno še pridobiti OpenCV 3.4.1 in OpenCV arhiv.

```
cd ~
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.4.1.zip
wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.4.1.zip
unzip opencv.zip
unzip opencv_contrib.zip
```

Slika 11: Ukazi za pridobitev OpenCV 3.4.1 in OpenCV arhiva

Potrebujemo še knjižnico Numpy. Numpy je knjižnica za Python programski jezik, ki ima podporo za večje, več dimenzijske nize in matrice, z veliko kolekcija visokih matematičnih funkcij za upravljanje s temi nizi. Numpy je bil ustvarjen leta 2005, ustvaril ga je Travis Oliphant. Numpy je bil ustvarjen tako, da je vstavil funkcije Numarray v Numeric, z nekaj zelo velikimi modifikacijami. Numpy je odprtokodni program, na katerem še vedno dela veliko ljudi. (vir 7)



Slika 12: NumPy logo[3]

```
sudo pip3 install numpy
```

Slika 13: Ukaz za namestitev knjižnice Numpy

Potem pa je bilo potrebno še zgraditi OpenCV-okolje.

```
cd ~/opencv-3.4.1/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
      -D CMAKE_INSTALL_PREFIX=/usr/local \
      -D INSTALL_PYTHON_EXAMPLES=ON \
      -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
      -D ENABLE_PRECOMPILED_HEADERS=OFF \
      -D BUILD_EXAMPLES=ON ..
```

Slika 14: Ukazi za graditev OpenCV-okolja

Nato smo povečali izmenjali prostor. Prostor smo povečali iz 100 MB na 1024 MB, da smo

olajšali sestavljanje OpenCV na vseh štirih jedrih Raspbery Pi. To pa smo storili tako, da smo odprli */etc/dphys-swapfile* z uporabo nano urejevalnika. Tam smo spremenili vrednost CONF_SWAPSIZE spremenljivke na 1024. Shranili smo spremenjeno datoteko in smo aktivirali nov izmenjalni prostor z uporabo naslednje komande. Za to komando smo ponovno zagnali napravo.

```
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
```

Slika 15:Ukaz za aktivacijo novega izmenjalnega prostora

Nato smo sestavili OpenCV na vseh štirih jedrih z naslednjim ukazom. Ta proces je trajal okoli dve uri.

```
make -j4
```

Slika 16: Ukaz za uporabo vseh štirih jeder

Za tem pa je potrebno samo še namestiti OpenCV

```
sudo make install
sudo ldconfig
```

Slika 17: Ukaz za namestitev OpenCV

Nato smo testirali, če je bil OpenCV uspešno nameščen z naslednjo komando.

```
import cv2
cv2.__version__
```

Slika 18: Ukaz, ki preveri, če je uspešno nameščen OpenCV

Če je bil OpenCV uspešno nameščen, nam ukazna vrstica vrne 3.4.1.

3.3 Izdelava programa

3.3.1 Zbiranje podatkov za prepoznavo obraza

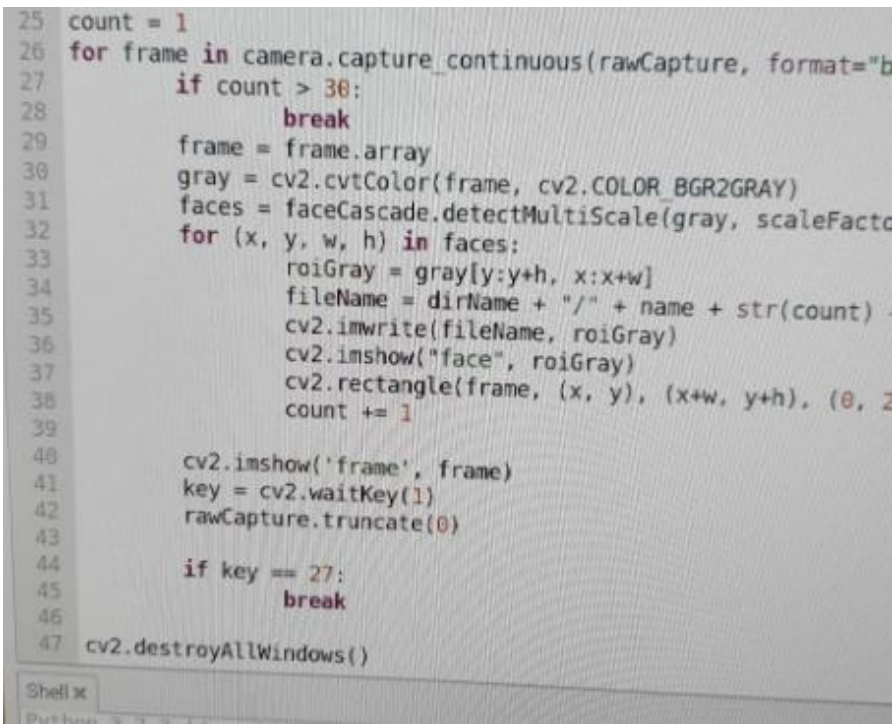
Prva naloga je bila, da zberemo podatke, s katerimi bomo trenirali naš program. Za to smo se odločili, da bomo naredili Python kodo, ki bo naredila trideset posnetkov obraza uporabnika s pomočjo predhodno usposobljenega OpenCV-klasifikatorja. OpenCV že uporablja veliko predhodno usposobljenih klasifikatorjev za obraze, oči, nasmeh itd. Klasifikator, ki smo ga mi uporabili, zazna obraz.
Program deluje v treh različnih fazah: zaznavanje obraza in zbiranje podatkov, urjenje prepoznavalnika obrazov in prepoznavanje obraza. Vsaka od faz je v svoji datoteki. Te datotek pa so shranjene v zavihku skupaj s sproti zbranimi podatki, ki jih program zbira med delovanjem.

3.3.2 Pisanje kode za zbiranje podatkov za prepoznavo obraza

Najprej smo morali uvoziti nekaj potrebnih paketov. Nato smo inicializirali objekt kamere, ki nam je omogočil upravljanje z Raspberry Pi kamero. Nastavili smo resolucijo kamere na 640, 480 in število slik na 30. Uporabili smo PiRGBArray(), ki nam vrne tridimenzionalini RGB niz, organiziran iz nekodiranega RGB-posnetka. Prednost PiRGBArray() je sposobnost, da bere slike s posnetka Raspberry Pi kamer, Numpy niz pa omogoči, da je združljiv z OpenCV. Izogiba pa se tudi pretvarjanju JPEG-formata v OpenCV, saj bi to zelo upočasnilo proces. PiRGBArray() potrebuje dva argumenta, prvi je kamera objekt, drugi pa resolucija. Nato naložimo še kaskade datoteko za zaznavo obraza. Za tem vprašamo uporabnika, da vnese ime. Če že obstaja datoteka z enakim imenom, kot ga je uporabnik vnesel, mu program odgovori “Ime že obstaja” in koda se zapre. Če pa datoteke z imenom ni, bo ustvarilo novo datoteko in vanjo shranilo slike in jo tako poimenovalo.

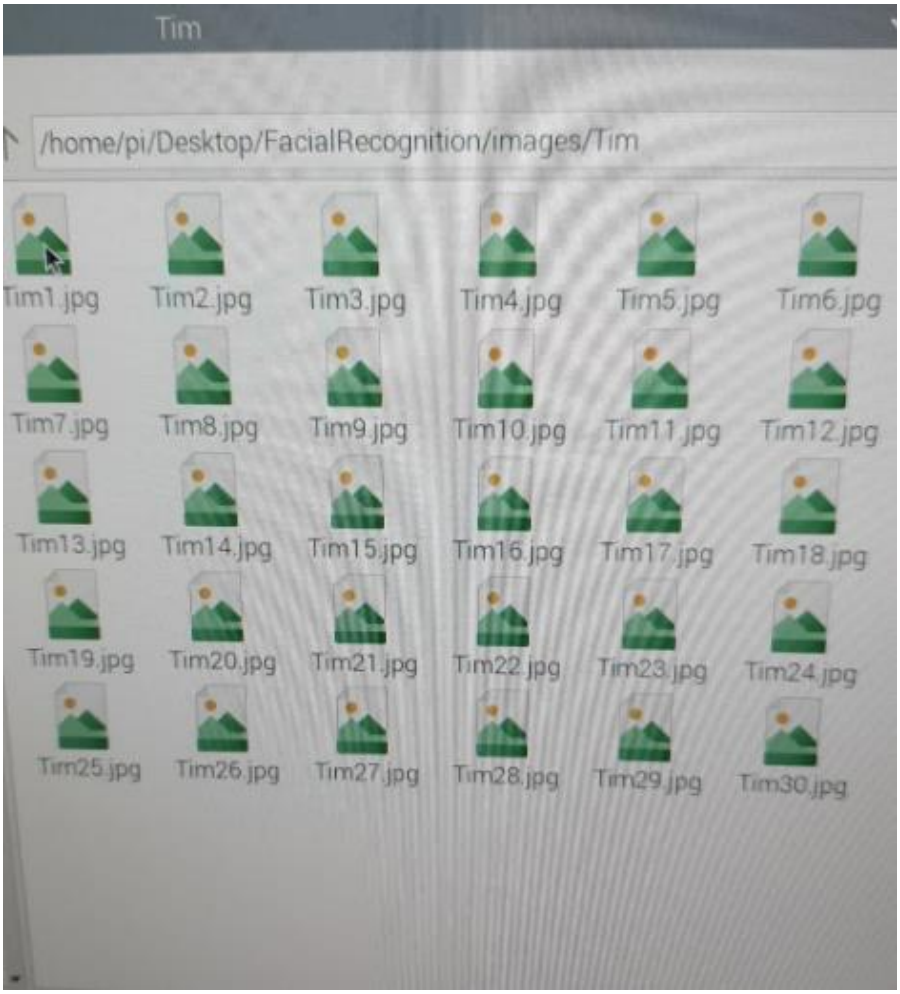
```
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import os
import sys

camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 30
rawCapture = PiRGBArray(camera, size=(640, 480))

faceCascade = cv2.CascadeClassifier("/home/pi/opencv-4.1.2/dat

name = raw_input("What's his/her Name? ")
dirName = "./images/" + name
print(dirName)
if not os.path.exists(dirName):
        os.makedirs(dirName)
        print("Directory Created")
else:
        print("Name already exists")
        sys.exit()

count = 1
for frame in camera.capture_continuous(rawCapture, format="bgr",
```

Slika 19: Prvi del kode za zbiranje podatkov



```
count = 1
for frame in camera.capture_continuous(rawCapture, format="b
        if count > 30:
                break
        frame = frame.array
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, scaleFacto
        for (x, y, w, h) in faces:
                roiGray = gray[y:y+h, x:x+w]
                fileName = dirName + "/" + name + str(count)
                cv2.imwrite(fileName, roiGray)
                cv2.imshow("face", roiGray)
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 2
                count += 1

        cv2.imshow('frame', frame)
        key = cv2.waitKey(1)
        rawCapture.truncate(0)

        if key == 27:
                break

cv2.destroyAllWindows()
```

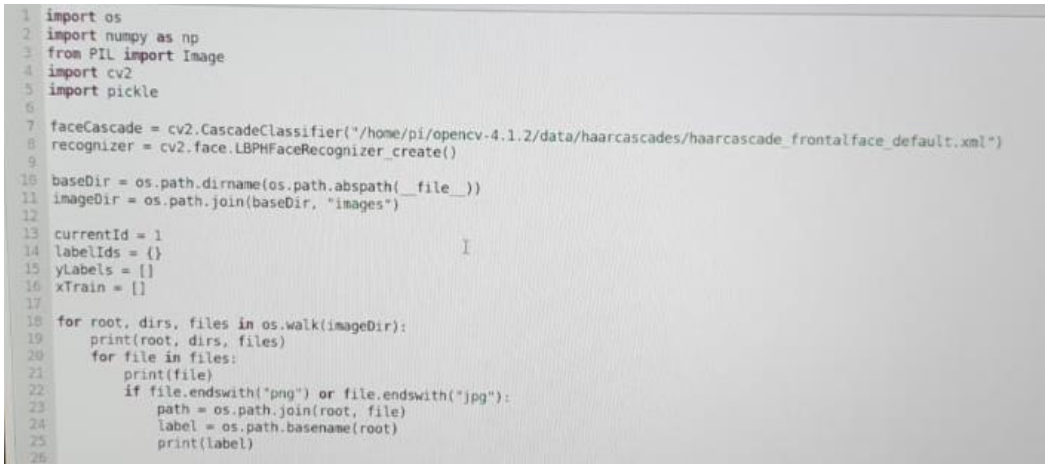
Slika 20: Drugi del kode za zbiranje podatkov



Slika 21: Zavihek s slikami enega uporabnika

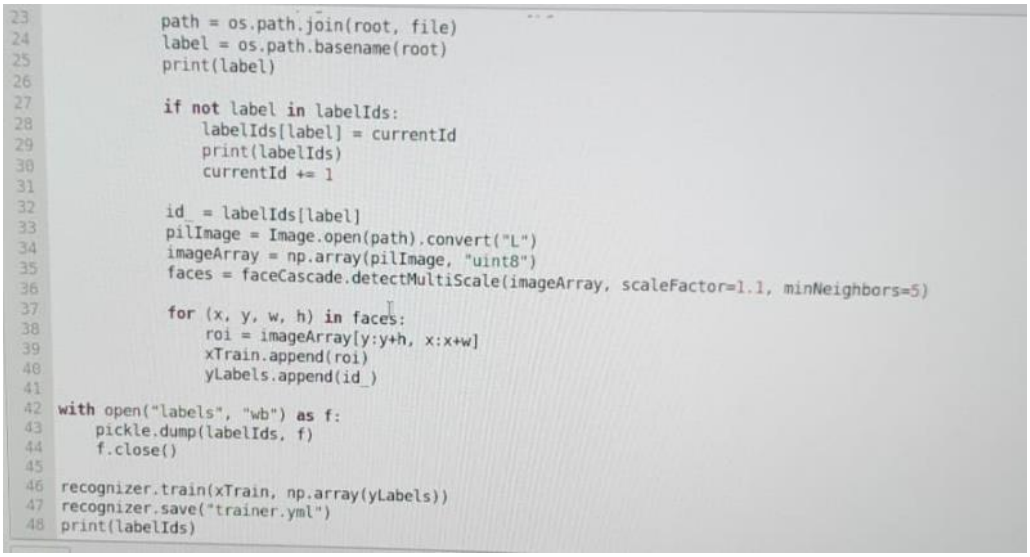
3.3.3 Učenje primerjalnika obrazov

Po tem, ko nam je prejšnji korak naredil in shranil trideset slik izbrane osebe, smo začeli s treniranjem primerjalnika obrazov. Program pregleda vse mape s slikami, in ko zazna sliko (to so datoteke s končnico .png ali .jpg), jo shrani v NumPy tabelo. Nato program ponovno zazna obraze na sliki, tako preveri, da so na vseh slikah obrazi. Sledi dodeljevanje id-ov. Id je številka, po kateri vemo, kateri osebi pripada katera slika. Nato na osnovi prej najdenih slik program izuri prepoznavalnik. Prepoznavalnik obrazov je tudi del knjižnice OpenCv, imenuje se *LBPH (LOCAL BINARY PATTERNS HISTOGRAMS) face recognizer*. Ko se program zaključi, se podatki shranijo v dve datoteki. Id-i se shranijo v datoteko *labels*, izurjen prepoznavalnik pa v datoteko *trainer.yml*.



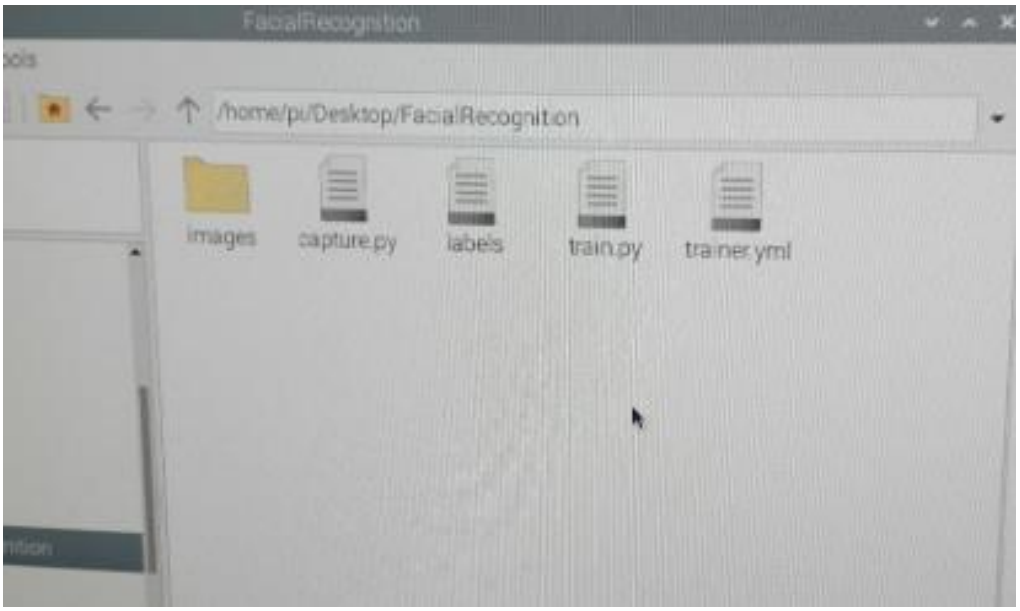
```
import os
import numpy as np
from PIL import Image
import cv2
import pickle

faceCascade = cv2.CascadeClassifier("/home/pi/opencv-4.1.2/data/haarcascades/haarcascade_frontalface_default.xml")
recognizer = cv2.face.LBPHFaceRecognizer_create()

baseDir = os.path.dirname(os.path.abspath(__file__))
imageDir = os.path.join(baseDir, "images")

currentId = 1
labelIds = {}
yLabels = []
xTrain = []

for root, dirs, files in os.walk(imageDir):
    print(root, dirs, files)
    for file in files:
        print(file)
        if file.endswith("png") or file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(root)
            print(label)
```

Slika 22: Prvi del kode za treniranje primerjalnika obrazov



```
            path = os.path.join(root, file)
            label = os.path.basename(root)
            print(label)

            if not label in labelIds:
                labelIds[label] = currentId
                print(labelIds)
                currentId += 1

            id_ = labelIds[label]
            pilImage = Image.open(path).convert("L")
            imageArray = np.array(pilImage, "uint8")
            faces = faceCascade.detectMultiScale(imageArray, scaleFactor=1.1, minNeighbors=5)

            for (x, y, w, h) in faces:
                roi = imageArray[y:y+h, x:x+w]
                xTrain.append(roi)
                yLabels.append(id_)

with open("labels", "wb") as f:
    pickle.dump(labelIds, f)
    f.close()

recognizer.train(xTrain, np.array(yLabels))
recognizer.save("trainer.yml")
print(labelIds)
```

Slika 23: Drugi del kode za treniranje primerjalnika obrazov



Slika 24: Izgled mape po zagonu programa za treniranje

Prepoznavanje obraza

V prejšnji fazi smo že uporabili prepoznavalnik iz knjižnice OpenCv. Tega smo ponovno uporabili za prepoznavanje obrazov. V tej fazi je tudi klasifikator za zaznavo obraza isti kot v prvi fazi. V tem koraku prepoznavalnik obrazov primerja posnetke zaznanih obrazov, ki jih med delovanjem programa posname kamera, in jih primerja s posnetki iz prvega koraka, ki smo jih uporabili za urjenje programa. Prepoznavalnik pri tem, ko zazna obraz, vrne vrednost, ki se ji reče *confidence*, to je vrednost, ki pove, koliko je oseba, posneta s kamero, podobna slikam osebe, ki so shranjene v programu. Višja kot je ta številka, manj je posneta oseba podobna prej shranjenim slikam. Mi smo si za mejo izbrali vrednost 65, saj smo po testiranju ugotovili, da je pri dani kameri to najboljša meja.

```
import os
import numpy as np
from PIL import Image
import cv2
import pickle

faceCascade = cv2.CascadeClassifier("/home/pi/opencv-4.1.2/data/haarcascades/haarcascade_frontalface_default.xml")
recognizer = cv2.face.LBPHFaceRecognizer_create()

baseDir = os.path.dirname(os.path.abspath(__file__))
imageDir = os.path.join(baseDir, "images")

currentId = 1
labelIds = {}
yLabels = []
xTrain = []

for root, dirs, files in os.walk(imageDir):
    print(root, dirs, files)
    for file in files:
        print(file)
        if file.endswith("png") or file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(root)
            print(label)
```

Slika 25: Prvi del kode za prepoznavo obraza

```
        roiGray = gray[y:y+h, x:x+w]

        id_, conf = recognizer.predict(roiGray)

        for name, value in dict.items():
            if value == id_:
                print(name)

        if conf <= 65:
            GPIO.output(relay_pin, GPIO.HIGH)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cv2.putText(frame, name + str(conf), (x, y), font, 2, (0, 0 ,255), 2,cv2.LINE_AA

        else:
            GPIO.output(relay_pin, GPIO.LOW)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('frame', frame)
    key = cv2.waitKey(1)

    rawCapture.truncate(0)

    if key == 27:
        break

cv2.destroyAllWindows()
```

Slika 26: Drugi del kode za prepoznavo obraza

4. REZULTATI

Med izdelovanjem te raziskovalne naloge smo pridobili mnogo koristnih znanj in veščin o tehnologiji prepoznave obraza in prav tako o mikrokrmilniku Raspberry Pi. Temu so pripomogli mnogi spletni vodiči (tutoriali), za katere pa smo ugotovili, da niso vsi koristni. Vendar pa smo vseeno hvaležni za poplavo podatkov, brez katerih takšen način pridobivanja znanj ne bi bil možen. Predvsem pa smo ugotovili, da brez dejanskega dela si niti slučajno ne zapomniš vseh podatkov, ki jih prebereš.

Pomemben delež pri uspešni izdelavi izdelka pa je odigralo tudi sodelovanje vseh treh članov ekipe. Na začetku smo mislili, da bo vsak izmed nas lahko delal po svoje in bomo na koncu samo združili rezultate posameznikovih del. Vendar to ne bi nikoli uspelo. Za uspešen in tekoč potek dela so potrebni odlična komunikacija, primerna razdelitev vlog ter nalog in sprotno poročanje o napakah in težavah. Tako smo s pomočjo te raziskovalne naloge izboljšali tudi naš ekipni duh in sposobnost sodelovanja drug z drugim.

4.1 Pregled hipotez

Ko smo končali pridobivanje novih znanj in nato še izdelali ključavnico, ki se odklepa s prepoznavo obraza, smo imeli dovolj znanih podatkov, da smo lahko potrdili oziroma ovrgli naslednje hipoteze:

1. **Za izdelavo pametne ključavnice ne potrebuješ predznanja o delovanju tehnologij prepoznave obraza.**

Ko smo se prvič lotili dela, smo mislili, da nimamo dovolj predznanja za to na prvi pogled zahtevno nalogo. Ampak ravno to pomankanje znanja nam je bila gonilna sila pri tem, ko smo iskali nove podatke in že obstoječe vire, ki so nam pomagali pridobiti zadostno znanje za izdelavo projekta. Tako da je ta prva hipoteza potrjena, saj nam je kljub nikakršnemu predznanju o tehnologiji prepoznave obraza uspelo izdelati pametno ključavnico.

2. **Cena izdelane ključavnice ne bo presegala 100 €.**

Naš prvoten cilj je bil izdelek končati po čim nižji ceni. Za mejo smo si izbrali 100 €. Ko smo si prvič ogledovali cene, smo pričakovali, da bo končna cena veliko višja, kot pa je na koncu bila. Končna cena naše pametne ključavnice je 92,20 €. To potrdi tudi našo drugo hipotezo, saj stroški niso presegli 100 €.

3. Tehnologija bo sposobna ločiti obraz in sliko obraza.

Že ob prvih pregledih virov smo zaznali, da ima kar nekaj ljudi težave s tem, da njihov program za prepoznavo obraza ne razlikuje med dejanskim obrazom fizične osebe ter sliko, skopirano ali pa prikazano na telefonu. K temu pripomore nekaj dejavnikov, najpogostejša sta ta, da kamera ni sposobna razlikovati med 3D- in 2D-objekti, in to da je klasifikator za zaznavo obraza namenjen 2D-slikam. Tako tudi v našem primeru. Morda tudi zaradi tega, ker smo imeli eno izmed cenejših kamer za Raspberry Pi, program ni ločil med sliko obraza in obrazom. Tako je naša zadnja hipoteza ovržena.

5. DISKUSIJA

V naši raziskovalni nalogi smo želeli izvedeti, kako težko in če je možno narediti sistem za odpiranje vrat s pomočjo obraza pod 100 €. Ugotovili smo, da je to dokaj preprosto narediti, saj obstajajo različne knjižnice, to so podprogrami za pomoč pri izdelavi oziroma razvoju programa. Mi smo uporabili knjižnico OpenCV, s pomočjo katere smo se lotili izdelave programa. Ugotovili smo tudi, da se da narediti sistem, ki bo odpiral vrata za manj kot 100 €, če imaš seveda prej tudi kaj znanja iz programiranja, saj menimo, da je drugače takšno kodo kar zapleteno napisati brez predznanja. Mi smo si za to raziskovalno nalogo zadali tri hipoteze. Prva hipoteza trdi, da je možno izdelati pametno ključavnico s pomočjo odklepanja obraza brez predhodnega znanja. To hipotezo smo morali ovreči, saj je težko napisati program brez kakršnih koli izkušenj od prej, saj moraš poznati vsaj osnove za izdelavo kakršnega koli programa. Za drugo hipotezo smo si zastavili, da cena ključavnice ne bo presegala 100 €, in to hipotezo smo potrdili, saj je bila končna cena izdelave malo manj kot 100 €. Kar pomeni, da je naša naprava najcenejša do sedaj. To nas je tudi presenetilo, saj smo mislili, da bomo morali to hipotezo zavreči, saj smo videli, da imajo ostale naprave kar višjo ceno. Zadnja hipoteza trdi, da bo naša prepoznava obraza ločevala med pravim obrazom in sliko. To hipotezo smo morali tudi ovreči, saj ne moremo biti 100 % prepričani, da ne moreš odpreti vrat s sliko, saj nas nato opozarjajo tudi bolj naprednejši sistemi, ki so ograjeni v mobilne naprave in ostale bolj razvite naprave. Vendar se bo ta tehnologija razvijala še naprej in smo prepričani, da bo to čez kašen čas možno narediti s pomočjo naprednejše kamere.

6. ZAKLJUČEK

Skozi sam postopek izdelovanja raziskovalne naloge smo se soočili z mnogimi težavami, ki smo se jih bili primorani naučiti odpraviti in rešiti. Na tak način smo pridobili mnogo novih zanimivih in koristnih znanj. Šele sedaj, ko smo končali nalogo in izdelali končni izdelek, vidimo, kakšna je razlika v znanju v primerjavi z začetnim. Tudi naša sposobnost za reševanje težav se je izboljšala. Največja in najpogostejša težava med samim delovnim procesom je bilo hitro segrevanje procesorskega čipa na Raspberryju. To se nam je dogajalo zaradi pomanjkanja hlajenja za procesor. Rešitev smo našli v manjšem kosu železa, ki smo ga ohladili in postavili na procesor Raspberryja, ta kos železa je tako hladil procesor in se obnašal kot nekako odlagališče za toploto, ki jo je hitro proizvajal procesor. To je bila za nas najizvirnejša rešitev, saj nam ni bilo potrebno kupiti hlajenja za procesor.

Naučili smo se, da ko se lotiš dela, te ne sme zmotiti vsak mali problem ali napaka. Napako je potrebno popraviti in problem je potrebno rešiti. Če si se pripravljen ob vsakem zapletu predati in opustiti delo, ne boš nikoli ničesar dokončal. Tega dejstva se sedaj po opravljeni nalogi še kako zavedamo. Jasno pa nam je tudi, da nič od narejenega ni bil samo produkt posameznikovega dela, ampak sad našega dobrega sodelovanja in komuniciranja. V tem pogledu smo vsi trije člani ekipe izboljšali tudi naše sposobnosti za sodelovanje drug z drugim.

Na koncu smo dosegli svoj prvotni cilj, in to je bil izdelava pametne ključavnice. Ta izdelek nam je v ponos, saj smo za njegovo izdelavo združili svoja že prej obstoječa znanja o programiranju in jim dodali še nekaj na novo pridobljenih znanj, ki smo se jih naučili z vprašanji mentorjem ali pa preko spletnih vodičev. Zadovoljni smo tudi z dejstvom, da končna cena ni presegala stotih evrov. Seveda pa če se ozremo nazaj na to, kako je potekalo delo in kaj vse smo naredili narobe, ugotovimo, da bi bilo moč marsikaj spremeniti in izboljšati. Vendar pa je to samo še en dober pokazatelj tega, da smo se v resnici tudi nekaj naučili.

7. POVZETEK

V zadnjem obdobju se tehnologija hitro spreminja. Temu se ustrezno prilagajajo vse javne ustanove. Tudi naša šola se trudi slediti novim trendom. A mi še vseeno pogrešamo nekaj res inovativnih tehnologij. Zato smo si za temo te raziskovalne naloge izbrali tehnologijo prepoznave obraza. S pomočjo te smo izdelali ključavnico, ki se odklene ob prepoznavi pravega obraza. Fotografije ustrezne osebe so arhivirane v bazi podatkov. Iz te baze jih dobi naš program in jih primerja s sliko, ki jo posname kamera ob pritisku na gumb. Takšni obliki varnosti se reče biometrična varnost, ki je na trgu uporabljena predvsem v industriji mobilnih telefonov, mi pa smo jo želeli uvesti v naš vsakdan. Ta pristop k vsakdanjemu opravilu odklepanja vrat se nam zdi zanimiv in menimo, da bi bil lahko uporabljen na več učilnicah naše šole.

7.1 A summary

Lately especially in the 21st century technology is changing at a rapid pace. Public institutions take all the right measures to stay up to date with the current technologies. Our school is also among these. But we still see room for improvement. That’s why we decided to base our research paper on facial recognition software. We managed to create a sort of smart lock, that unlocks when detecting the right face. People who have access to the room have pictures of their face stored in a database, from witch our software compares them with pictures taken by the camera. This form of security is called biometrics. Facial recognition is mostly used in the mobile phone industry, mainly for advertising, since it's not a very safe form of security. But we still wanted to implement it into our daily lives. We believe that our smart lock makes something as trivial as unlocking doors modern and fun.

8. ZAHVALA

Radi bi se zahvalili za vso pomoč pri izdelovanju in razvijanju raziskovalne naloge: mentorjema Urošu Remenihu in Roku Urbancu za vso pomoč in podporo, profesorici Lidiji Šuster za lektoriranje, Gregorju Hrastniku, univ. dipl. inž. rač. in inf., ki nam je prav tako pomagal pri izdelavi te raziskovalne naloge, ter recenzentom raziskovalne naloge, staršem in vsem ostalim, ki so pomagali pri nastanku naše raziskovalne naloge.

9. VIRI IN LITERATURA

Vir 1 - https://www.investopedia.com/terms/s/security.asp

(7. 1. 2020)

Vir 2 - https://primerjavaoseb.com/biometrija.html

(22. 1. 2020)

Vir 3 -https://www.americanbar.org/groups/criminal_justice/publications/criminal-justice-magazine/2019/spring/facial-recognition-technology/
(22. 1. 2020)

Vir 4 - https://www.raspberrypi.org/downloads/noobs/

(30. 1. 2020)

Vir 5 - https://www.python.org/doc/essays/blurb/

(30. 1. 2020)

Vir 6 - https://opencv.org/about/

(30. 1. 2020)

Vir 7 - https://numpy.org/

(30. 1. 2020)

9.1 Viri slik

Slika[1]

https://www.google.com/url?sa=i&url=https%3A%2F%2Fswag.raspberrypi.org%2Fproducts%2Fnoobs&psig=AOvVaw3Vg0FXAD-JhpPJgXUFo80Y&ust=1581507171932000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCOiY4byzyecCFQAAAAAdAAAAABAD

(30. 1. 2020)

Slika[2]
https://www.google.com/url?sa=i&url=https%3A%2F%2Fsyncedreview.com%2F2018%2F11%2F21%2Fopencv-4-0-release-ends-3-5-year-wait%2F&psig=AOvVaw3KM7X1vls0UltT6WZ1JjD1&ust=1581518702148000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCMiu_7beyecCFQAAAAAdAAAAABAD
(30. 1. 2020)

Slika[3] https://upload.wikimedia.org/wikipedia/commons/thumb/1/1a/NumPy_logo.svg/1024px-NumPy_logo.svg.png
(30. 1. 2020)