UNIVERZITET U NOVOM SADU **FAKULTET TEHNIČKIH NAUKA U NOVOM SADU**

**DIPLOMSKI RAD**

Prenosivi sistem za zaštitu od provale

Kandidat: Mentor:

Katarina Ninković Prof. dr Vladimir Rajs

Novi Sad, 2020.god.

|  |  |
| --- | --- |
|  | UNIVERZITET U NOVOM SADU ⚫**FAKULTET TEHNIČKIH NAUKA**  21000 NOVI SAD, Trg Dositeja Obradovića 6 |
| **KLJUČNA DOKUMENTACIJSKA INFORMACIJA** |

|  |  |  |  |
| --- | --- | --- | --- |
| Redni broj, **RBR**: | |  | |
| Identifikacioni broj, **IBR**: | |  | |
| Тip dokumentacije, **TD**: | | Monografska publikacija | |
| Тip zapisa, **TZ**: | | Tekstualni štampani dokument | |
| Vrsta rada, **VR**: | | Diplomski rad | |
| Аutor, **АU**: | | Katarina Ninković | |
| Меntor, **МN**: | | Prof. dr Vladimir Rajs | |
| Naslov rada, **NR**: | | Prenosivi sistem za zaštitu od provale | |
| Јеzik publikacije, **JP**: | | Srpski | |
| Јеzik izvoda, **JI**: | | Srpski | |
| Zemlja publikovanja, **ZP**: | | Republika Srbija | |
| Uže geografsko područje, **UGP**: | | Autonomna Pokrajina Vojvodina | |
| Godina, **GО**: | | 2020. | |
| Izdavač, **IZ**: | | Autorski reprint | |
| Мesto i adresa, **МА**: | | 21000 Novi Sad, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6 | |
| Fizički opis rada, **FО**: (poglavlja/strana/citata/таbela/slika/grafika/priloga) | | 10 / 42 / 10 / 1 / 55 / 8 / 0 | |
| Naučna oblast, **NO**: | | Automatika | |
| Naučna disciplina, **ND**: | | Računarske komunikacije, Primenjena elektronika | |
| Predmetna odrednica/Ključne reči, **PO**: | | Automotive | |
| **UDK** | |  | |
| Čuva se, **ČU**: | | Biblioteka Fakulteta tehničkih nauka, Trg Dositeja Obradovića 6, Novi Sad | |
| Važna napomena, **VN** | |  | |
| Izvod, **IZ**: | | U projektu je implementiran sistem za detekciju provalnika koji primenom računarske komunikacije (Ethernet - TCP/IP protokola) obaveštava povezane klijentske uređaje o provali i prenosi (eng. “streaming”) snimak uživo. Prenosivi ređaj je *RaspberryPi* mini računar koji radi na *Linux* platformi i izvršava *Python* konzolnu aplikaciju koja rukovodi (eng. “handles”) komunikacijom sa klijentskim uređajima i očitavanjima sa senzora. Sa druge strane, klijent predstavlja platformno uniformnu aplikaciju koja je podržana kako na *Windows*, tako i na *Linux* operativnim sistemima. | |
| Datum prihvatanja teme, **DP**: | | Septembar 2020. | |
| Datum odbrane, **DO**: | |  | |
| Članovi komisije, **КО**: | Predsednik: |  |
|  | Član: |  | Потпис ментора |
|  | Mentor: | doc. dr Vladimir Rajs |  |

|  |  |
| --- | --- |
|  | UNIVERSITY OF NOVI SAD⚫**FACULTY OF TECHNICAL SCIENCES**  21000 NOVI SAD, Trg Dositeja Obradovića 6 |
| **KEY WORDS DOCUMENTATION** |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | Monographic publication | |
| Type of record, **TR**: | | Textual printed material | |
| Contents code, **CC**: | | Bachelor Thesis | |
| Author, **AU**: | | Katarina Ninković | |
| Mentor, **MN**: | | PhD Vladimir Rajs | |
| Title, **TI**: | | Portable system for housebreaking protection | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | English | |
| Country of publication, **CP**: | | Republic of Serbia | |
| Locality of publication, **LP**: | | Autonomous Province of Vojvodina | |
| Publication year, **PY**: | | 2020. | |
| Publisher, **PB**: | | Author’s reprint | |
| Publication place, **PP**: | | Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | | 10 / 42 / 10 / 1 / 55 / 8 / 0 | |
| Scientific field, **SF**: | | Automation | |
| Scientific discipline, **SD**: | | Computer Communications, Applied Electronics | |
| Subject/Key words, **S**/**KW**: | | Automotive | |
| **UC** | |  | |
| Holding data, **HD**: | | Library of the Faculty of Technical Sciences, Novi Sad | |
| Note, **N**: | |  | |
| Abstract, **AB**: | | In this project has been implemented system for housebreaking detection which informs client devices using Computer communication (Ethernet - TCP/IP interface) about housebreaking and it streams live video. Portable device is *RaspberryPi*, it is amini computer which runs on *Linux* platform and it executes *Python* console application which handles communication with client devices and reading values from sesnors. From another side, client represents platform independent which is runnable both on *Windows* and *Linux* operative systems. | |
| Accepted by the Scientific Board on, **ASB**: | | September 2020. | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: | Associate  Definitions of associate  noun  a partner or colleague in business or at work.  he arranged for a close associate to take control of the institute  synonyms: partner, colleague, coworker, workmate, comrade, ally, affiliate, confederate, connection, contact, acquaintance, collaborator, crony, peeps  verb  connect (someone or something) with something else in one's mind.  I associated wealth with freedom  synonyms: link, connect, relate, identify, equate, bracket, set side by side  adjective  joined or connected with an organization or business.  an associate company  2 more definitions  See also  associate  Translations of associate  noun  помоћник  aide, deputy, helper, associate, aid, help  колега  colleague, fellow, associate, fellow worker  заменик  deputy, assistant, substitute, associate, lieutenant, locum tenens  ортак  partner, consort, fellow, associate, bedfellow, yokefellow  учесник  participant, partaker, entrant, participator, partner, privy  verb  везати  tie, bind, link, attach, moor, connect  асоцирати  associate  везивати  bind, connect, relate, associate, tie down  удружити  join, associate, pool, unite, affiliate  adjective  ванредан  associate, extramural, part-time |
|  | Member: |  | Menthor’s sign |
|  | Member, Mentor | PhD Vladimir Rajs |  |

|  |  |  |  |
| --- | --- | --- | --- |
| NA OSNOVU PODNETE PRIJAVE, PRILOŽENE DOKUMENTACIJE I ODREDBI STATUTA FAKULTETA IZDAJE SE ZADATAK ZA DIPLOMSKI RAD, SА SLEDEĆIM ELEMENTIMA: | | | |
| Student: | Katarina Ninković | Broj indeksa: | EE 32/2015 |
| Predmet: | Elektronika | | |
| Mentor: | Doc. dr Vladimir Rajs | | |

**TEMA DIPLOMSKOG RADA:**

|  |
| --- |
| Prenosivi sistem za zaštitu od provale |

**ТEKST ZADATKA :**

|  |
| --- |
| 1. Podići operativni sistem (*Raspbian*) na prenosivom uređaju, kao i neophodne projektne zavisnosti (biblioteke koje se koriste i radno okruženje) 2. Omogućiti komunikaciju između servera i klijenata i pripremiti teren za prenos podataka 3. Omogućiti očitavanje sa senzora detekcije (udaljenosti od objekta), te izveštavanje klijentskih uređaja o prisutnosti provalnika 4. Omogućiti vizualnu detekciju lica uz pomoć kamere priključene za prenosivi uređaj 5. Omogućiti prenos snimka prilikom provale ka svim klijentskim uređajima 6. Realizovati platformno-nezavisnu klijentsku konzolnu |

|  |  |
| --- | --- |
| Šef katedre: | Mentor rada: |
| dr Mirjana Damnjanović | dr Vladimir Rajs |

|  |
| --- |
| Primerak zа: - Studenta; - Mentora; - Katedru; - Studentsku službu fakulteta; |

Obrazac **Q4.09.11** - Izdanje 1

Sadržaj

[Lista skraćenica 3](#_Toc51105463)

[1 Uvod 4](#_Toc51105464)

[2 Blok šema sistema 5](#_Toc51105465)

[3 Komponente i uređaj na serverskoj strani 6](#_Toc51105466)

[3.1 Mašina 6](#_Toc51105467)

[3.2 Senzor udaljenosti 7](#_Toc51105468)

[3.3 Senzor kamere 9](#_Toc51105469)

[4 Alat za detekciju čovekovog lica – OpenCV 10](#_Toc51105470)

[5 TCP/IP soket i mrežno programiranje 11](#_Toc51105471)

[6 Klasifikatori i njihova upotreba 13](#_Toc51105472)

[7 Algoritam rada serverske aplikacije 15](#_Toc51105473)

[8 Algoritam rada klijentske aplikacije 19](#_Toc51105474)

[9 Zaključak 22](#_Toc51105475)

[10 Literatura 23](#_Toc51105476)

# Lista skraćenica

**Spisak skraćenica na engleskom jeziku**

**TCP** Transmmision Control Protocol – Transmisioni kontrolni protokol

**HDMI** High Definition Multimedia Interface – Multimedijski interfejs visoke rezolucije

**USB** Universal Serial Bus – Univerzalna serijska magistrala

**LAN** Local Area Network – Lokalna računarska mreža

**VCC** Voltage Common Collector – Zajednički kolektor napona

**GND** Ground – Masa (Uzemljenje)

**CSI** Camera Serial Interface – Serijski intefejs kamere

**BSD** Berkeley Software Distribution – Distribucija softvera *Berkeley*

**API** Application Programming Interface – Interfejs za programiranje aplikacija

**IP** Internet Protocol – Internet protokol

**UDP** User Datagram Protocol – Korisnički protokol datagrama

**XML** Extensible Markup Language – Proširivi jezik za označavanje

**GPIO** General Purpose Input Output – Ulaz/izlaz opšte namene

**GUI** Graphical User Interface – Grafički korisnički interfejs

# 1 Uvod

Od davnina postoji strah kod ljudi za zaštitu svoje imovine. U današnje vreme sama bezbednost je dobila na porastu zbog tehnološkog napretka, gde svako domaćinstvo olako može da priušti neki oblik zaštitnog sistema, kao što su kamere, alarmi itd.

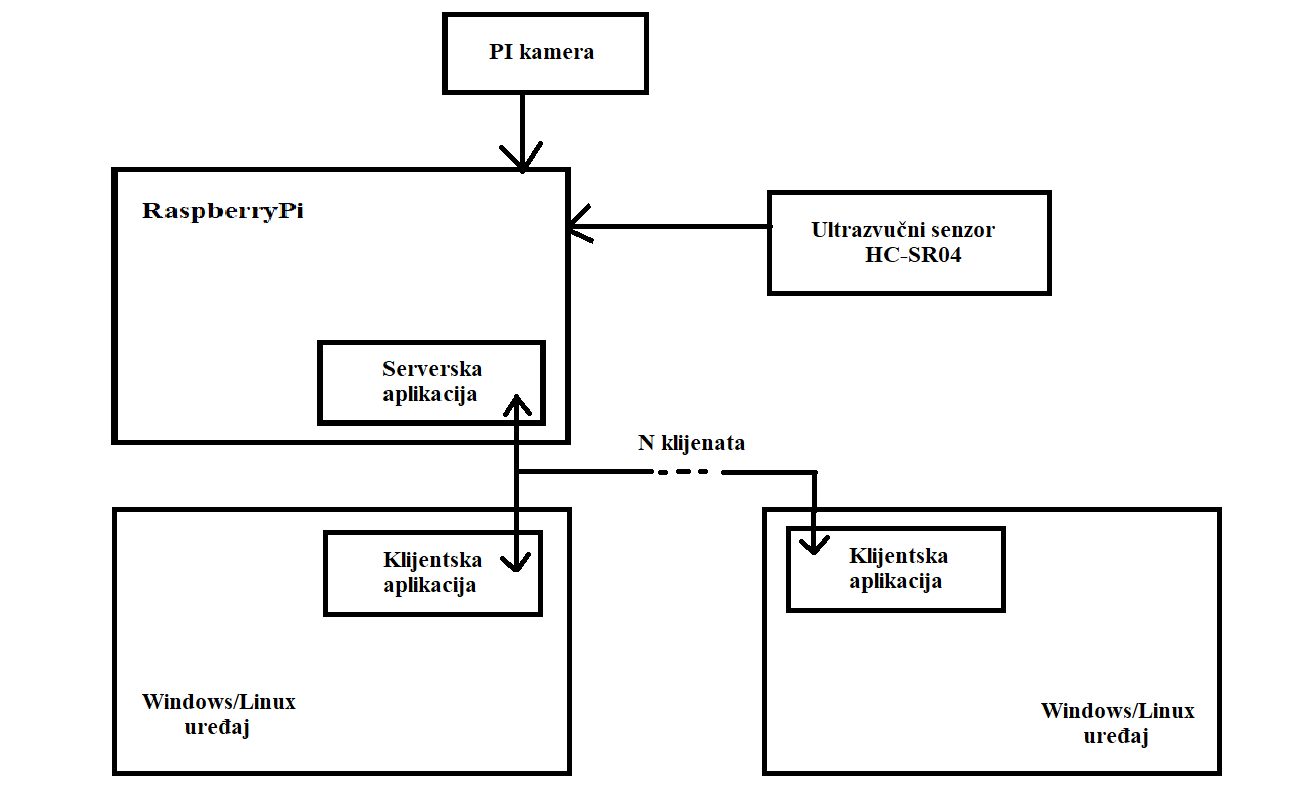
Tema ovog projekta vezana je za kreativan način implementiranja (instalacije) bezbednosnog uređaja. Glavnu ulogu u okviru ovog sistema igra prenosivi uređaj (*RaspberryPi*) koji uz pomoć senzora dolazi do zaključaka da li se u kući nalazi uljez, a potom izveštava korisničku (klijent) aplikaciju.



Slika 1. Bezbednosne kamere

# 2 Blok šema sistema

Kao što se može zaključiti iz uvoda sistem je podeljen u dve celine – serversku (nadzornu) i klijentsku (korisničku) stranu. Na serverskoj strani se nalaze *RaspberryPi* kao srce ove celine, kao i senzor udaljenosti (*HCSR04)* i senzor kamere. Sa korisničke strane može se naći bilo kakav uređaj (računar) koji ima podržano izvršavanje *Python* aplikacije. Ove dve celine komuniciraju uz pomoć *TCP* utičnice (eng. *Transmission Control Protocol socket),* koja uspešno funkcioniše na lokalu (lokalnoj mreži), a informacije koje se šalju pakuju se u *Ethernet* frejm i putuje ka odabranoj destinaciji. Komunikacija je bidirekciona, što znači da oba uređaja će biti kako prijemnik, tako i predajnik istovremeno.



Slika 2. Blok šema celokupnog sistema

# 3 Komponente i uređaj na serverskoj strani

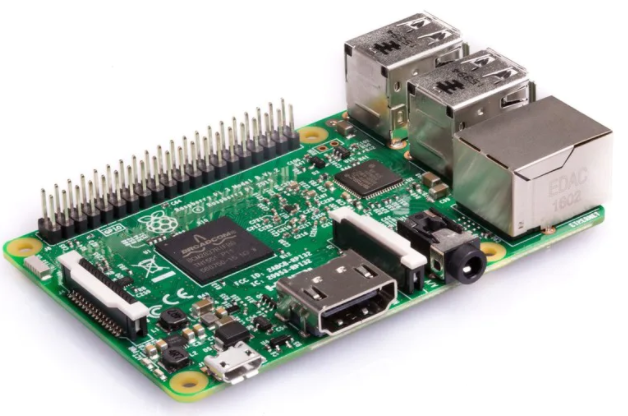
U ovom poglavlju su koncizno opisane hardverske zavisnosti serverskog bloka. Serverski uređaj zajedno čine mašina na kojoj se izvršava aplikacija, kao i senzori koji su povezani na nju.

## 3.1 Mašina

*RaspberryPi* je jeftini računar, veličine kreditne kartice, na kojeg je moguće uključiti monitor računara ili bilo koji drugi eksterni displej sa *HDMI (*eng. *High-Definition Multimedia Interface)* konektorom, kao i miš, tastaturu i ostale *USB* (eng. *Universal Serial Bus)* uređaje koji se mogu povezati na računar. Ovaj uređaj pokriva veliku većinu multimedijalnih mogućnosti kao što su “surfovanje” po internetu, reprodukcija video zapisa, obrade teksta i igranje nisko-grafičkih igara. Jedna od glavnih odlika ovog uređaja je prenosivost, mala veličina, pristupačna cena itd.

U projektu je korišćen *RaspberryPi 3 Model B*, koji je treća generacija ovog proizvoda. Koristi se za mnoge aplikacije i zamenjuje orginalni B+ model kao i *RaspberryPi 2 Model B*. Iako održava popularni izgled *single-board* ploče, ovaj model donosi procesor znatno boljih performansi, 10 puta brži od modela prve generacije. Pored toga, omogućene su i bežične *LAN* i *Bluetooth* veze.

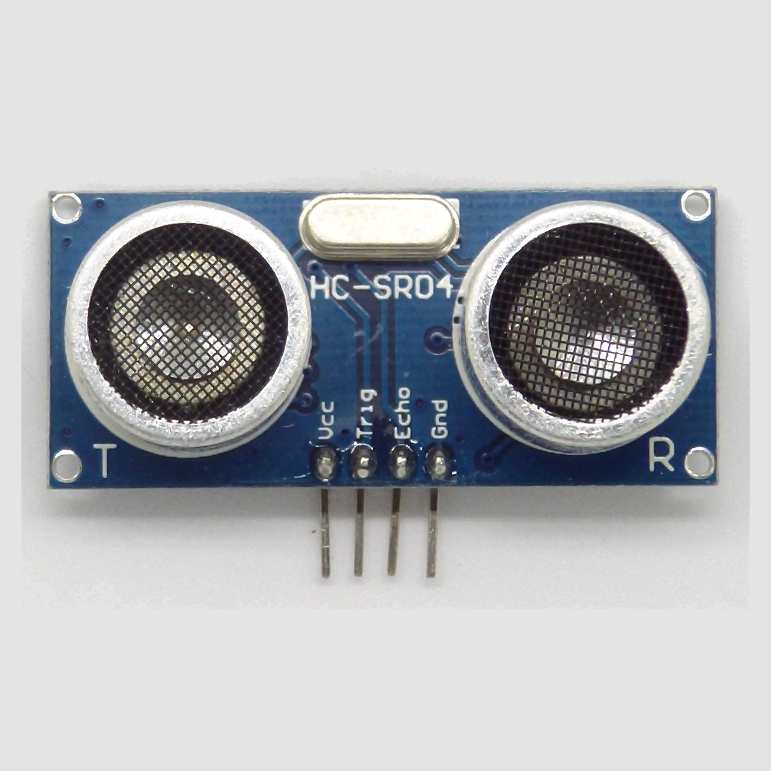
Zasnovan je na *Linux* platformi a poseduje i oficijalni *Raspbian* operativni sistem. Jedna od glavnih karakteristika mu je to što ima izvučene kontakte (pinove) na koje se mogu dovesti različiti senzori i aktuatori. U slučaju ovog projekta na te kontakte doveden je senzor udaljenosti, a putem *USB-a* ili posebnog ulaza je povezana kamera.



Slika 3. Fizički izgled uređaja

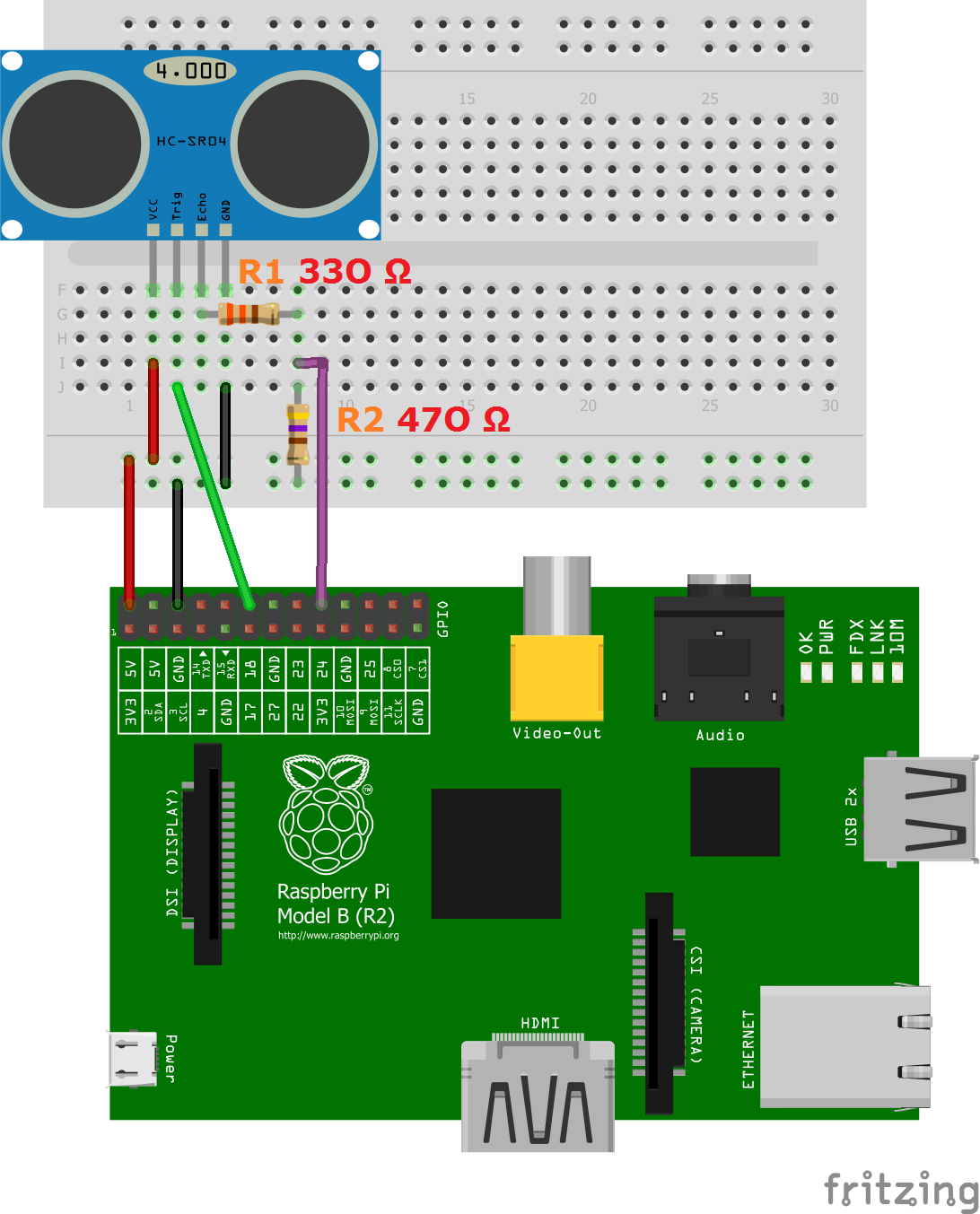
## 3.2 Senzor udaljenosti

U okviru ovog projekta ulogu senzora koji je zadužen za detekciju igra ultrazvučni senzor udaljenosti HC-SR04. U pitanju je ekonomični senzor beskontaktnog merenja i detektuje telo koje se nalazi naspram njega na udaljenostima u opsegu od 2-400 *cm*, sa tačnošću do 3 *mm*.



Slika 4. Fizički izgled senzora udaljenosti

Svaki HC-SR04 modul sadrži ultrazvučni predajnik, prijemnik i kontrolno kolo. Senzor poseduje 4 pina sa kojim se može upravljati senzorom, od toga 3 ulazna i jedan izlazni. Pored pinova zaduženih za napajnje (*VCC* i *GND*), senzor poseduje okidajući (eng. *Trigger*) pin kojim se pobuđuje tako što prima impuls od najmanje 10 *µs*. Pomoću odzivnog (eng. *Echo*) pina moguće je dobiti izmerenu udaljenost.

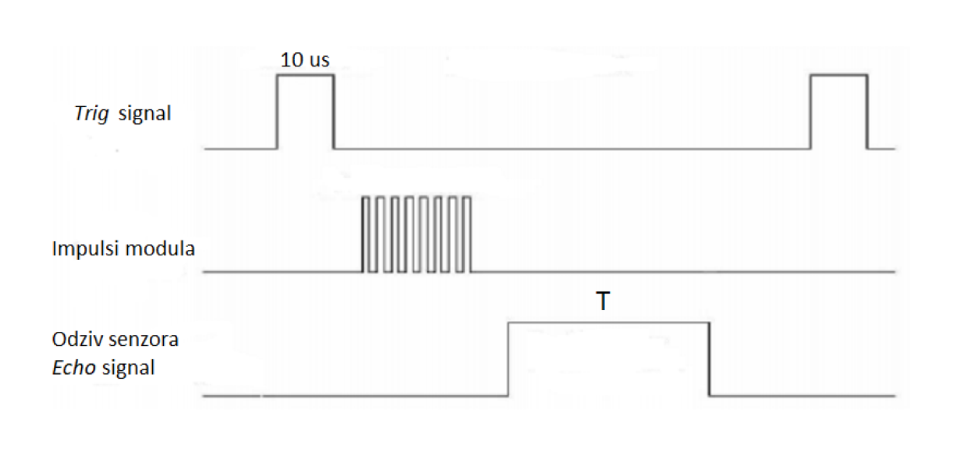


Slika 5. Način povezivanja senzora

Princip rada senzora započinje tako što se senzor pobuđuje impulsom od 10 *µs* preko okidajućeg pina. Kao odgovor na to senzor emituje zvučni „rafal“ od 8 impulsa na 40 *kHz*. Ova povorka je interni signal koji predstavlja „ultrazvučni potpis“ (eng. *Ultrasonic signature*), jer ga prijemnik jednostavno identifikuje i razlikuje od potencijalnih ambijentalnih šumova. Kada se impulsi reflektuju nazad, odnosno senzor je detektovao prepreku, odzivni signal biće na visokoj vrednosti vremensku periodu koji je ekvivalentan „putovanju“ signala ka prepreci i nazad (1).

(1)

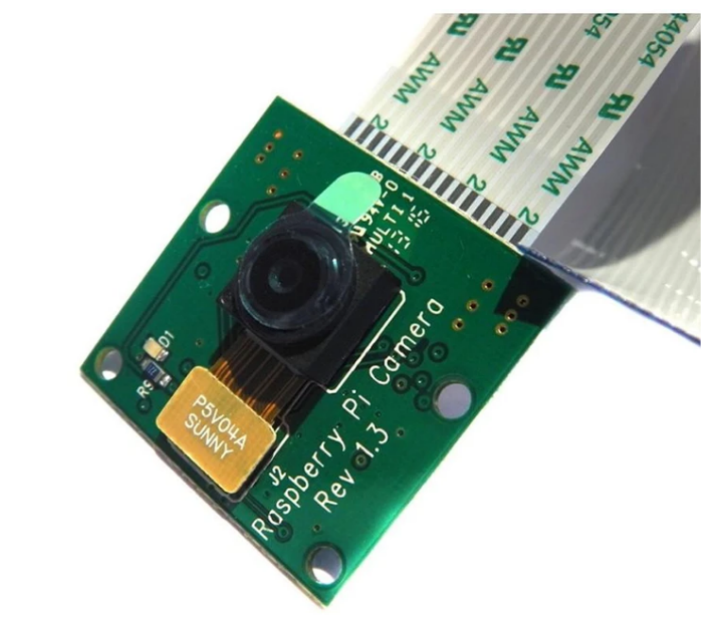
Gde je *D* – Udaljenost (eng. *Distance*), *Vs* – Brzina zvuka (eng. *Velocity of Sound*), *T* – Proteklo vreme (eng. *Time*)



Slika 6. Signali ultrazvučnog senzora

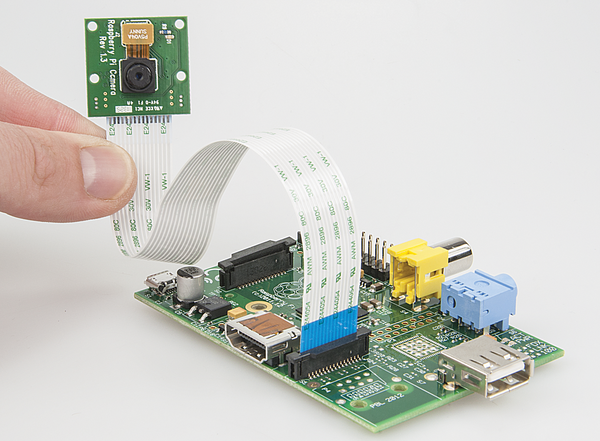
## 3.3 Senzor kamere

Senzor kamere u slučaju ovog uređaja igra *RaspberryPi Camera Module*. U pitanju je specijalno dizajnirani dodatak za *RaspberryPi* od 5 megapiksela, koji sadrži sočiva sa fiksnim fokusom. Senzor pruža statičke slike od 2592x1944 *pix*, a takođe podržava video zapise 1080p30, 720p60 i 640x480p60/90.



Slika 7. Fizički izgled senzora kamere

Senzor se priključuje na *RaspberryPi* pomoću jedne od malih utičnica na gornjoj površini ploče i koristi namenski *CSI* interfejs, dizajniran posebno za povezivanje sa kamerama.



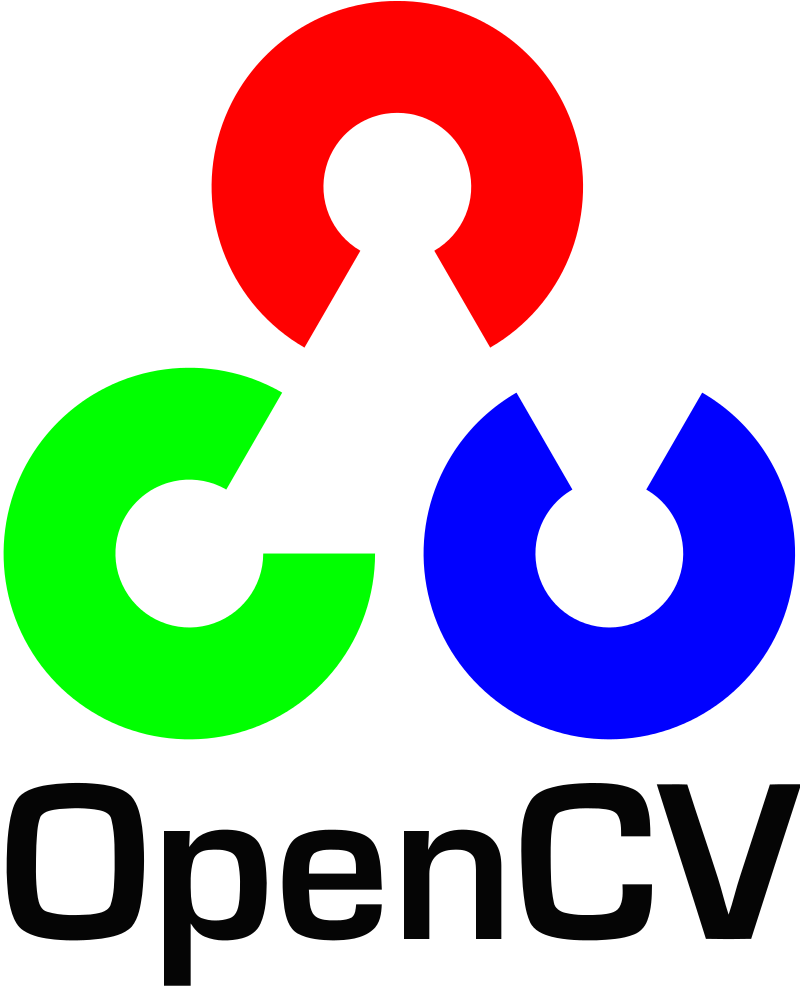
Slika 8. Način povezivanja kamere sa prenosivim uređajem

# 4 Alat za detekciju čovekovog lica – OpenCV

*OpenCV* (*Open Source Computer Vision Library*) je softverska biblioteka otvorenog koda zadužena za računarsku viziju i mašinsko učenje. *OpenCV* je napravljen da omogući zajedinčku infrastrukturu za aplikacije na kojima se izvršava neka vrsta računarske vizije i da ubrza primenu mašinske percepcije u komercijalnim proizvodima. Pošto je *BSD*-licenciran proizvod, *OpenCV* čini poslovanje olakšanim, a kod lako modifikovanim i primenjivim.

Biblioteka poseduje više od 2500 optimizovanih algoritama, što uključuje sveobuhvatan skup kako klasičnih tako i najsavremenijih algoritama za računarsku viziju i mašinsko učenje. Ovi algoritmi mogu biti korišćeni za detekciju i prepoznavanje lica, identifikaciju objekata, klasifikaciju čovekovih pokreta u video zapisima, praćenje pokreta kamere, ka i kretanje objekata, ekstrakciju 3D modela od objekata itd. *OpenCV* poseduje zajednicu od 47 hiljada korisnika i procenjeni broj preuzimanja biblioteke na neverovatnih 18 miliona. Biblioteka se intenzivno koristi u kompanijama, istraživačkim grupama i od strane državnih organa.

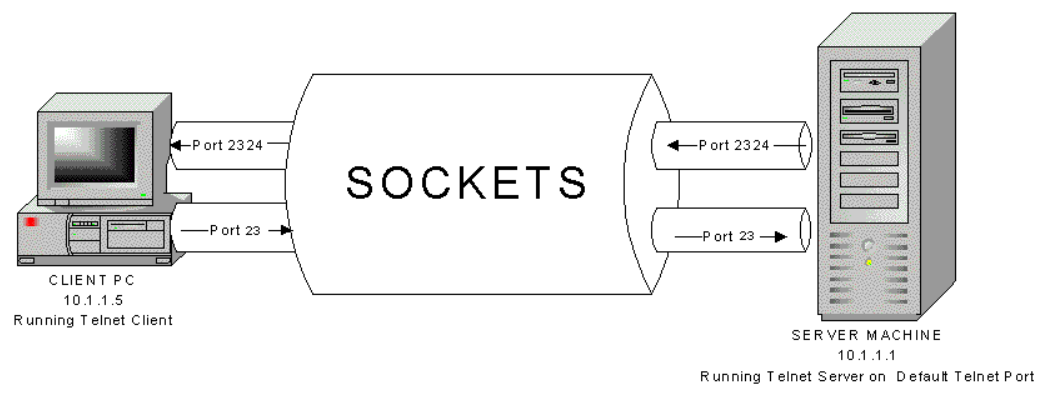
Biblioteka je prvobitno pisana na *C++* programskom jeziku od strane *Intel*-ovog istraživačkog centra u Nižnjem Novgorodu, u Rusiji. Naknadno je napravljena adaptacija za *Python* i *Java* programske jezike, ali i za *MATLAB* interfejs. Podržana je na svim poznatim platformama - *Linux*, *Windows*, *Android* i *MacOS*.



Slika 9. Logo biblioteke

# 5 TCP/IP soket i mrežno programiranje

U oblasti računarskih komunikacija termin soket (eng. *Socket*) ili mrežni soket je krajnja tačka dvosmernog međuprocesnog komunikacionog toka preko računarske mreže bazirane na internet protokolu, kao što je *Ethernet*. Mrežni soketi predstavljaju aplikacioni programski interfejs (*API*) koji je obično podržan od strane operativnog sistema. Oni grade mehanizam za dostavljanje paketa podataka (frejmova) odgovarajućem aplikacionom procesu ili niti, na osnovu kombinacija lokalnih i udaljenih *IP* adresa i portova. Operativni sistem preslikava svaki soket u proces ili nit aplikacije koja komunicira sa udaljenim računarom. Soket adresa je spoj *IP* adrese (lokacije računara) i porta (koji se preslikava u proces ili nit) u jedinstveni identitet.



Slika 10. Ilustrativni primer soketa kao medijuma između klijenta i servera

Internet soket je određen jedinstvenom kombinacijom:

* Protkola (*TCP*, *UDP* ili sirovi *IP*). Zbog ovoga *TCP* port jedne vrednosti ne predstavlja isti soket kao *UDP* port iste te vrednosti.
* Lokalne adrese soketa (lokalna *IP* adresa i broj porta)
* Udaljene adrese soketa. Ova adresa odnosi se samo na uspostavljanje *TCP* soketa. Ovo je potrebno jer *TCP* server može da opslužuje više klijenata u isto vreme. Server pravi po jedan soket za svakog klijenta, dok soketi dele istu lokalnu adresu soketa.

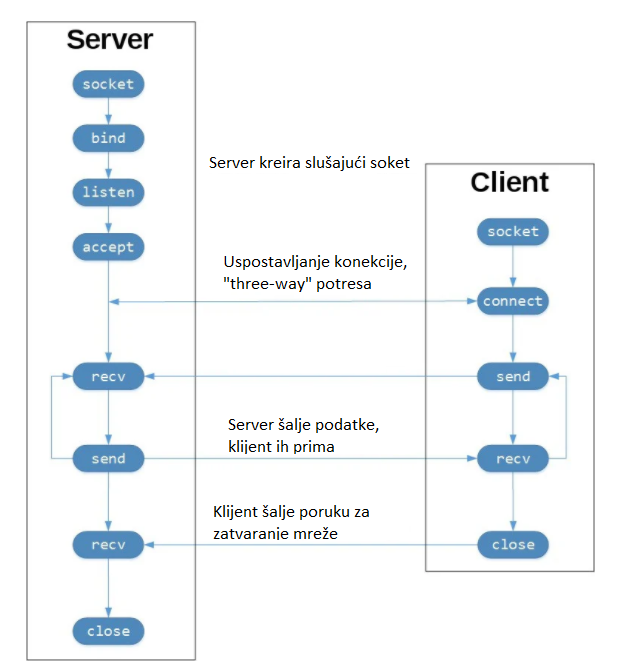
Operativni sistem prosleđuje dolazne *IP* pakete odgovarajućem aplikacionom ili servisnom procesu nakon što iz zaglavlja *IP* i transportnog protokola izvuče podatak o adresi soketa. Unutar aplikacije koja je napravila soket, soket se referiše jedinstvenim celim brojem koji se naziva identifikator soketa ili broj soketa.

U okviru ovog projekta korišćen je *TCP* soket zato što:

1. Pouzdan je – paketi ispušteni u mreži se detektuju na osnovu potvrde (eng. *Acknowledgment*) i pošiljalac ih ponovo prenosi ukoliko isporuka nije uspela.
2. Ima isporuku podataka po redosledu – aplikacija koja dobija podatke ih čita onim redosledom kojim ju je napisao pošiljalac.

Zbog ovih osobina *TCP* *Ethernet* interfejsa i komuniciranja između soketa protok podataka može da izgubi na brzini. U situaciji gde je brzina prenosa od presudnog značaja, a ne sam kvalitet, odnosno podudarnost poslatog paketa, kao i pouzdanost da li će poruka uopšte stići, više se koristi *UDP* protokol. Kao primer primene *TCP*-a može se uzeti slanje elektronske pošte (eng. *e-mail*), gde je od krunskog značaja da sadržaj bude baš onakav kakvog ga je pošiljalac napisao, kao i da zagarantovano stigne. Sa druge strane, primer *UDP*-a bio bi video razgovor.

*Python* soket modul pruža interfejs za *Berkeley Sockets API*. Da bi se soket mogao koristiti neophodno je pri soket instanciranju (instanciranje konstruktora sa parametrima) proslediti dva parametra ili ih naknadno proslediti uz pomoć postavljačke (eng. *Set*) metode *setsockopt* (slučaj instanciranja praznog konstruktora). U pitanju su statičke konstante iz soket klase čija je uloga da definišu način primene instance. Potom je neophodno proslediti serversku *IP* adresu i komunikacioni port. Ukoliko je adresa “živa”, tj. aktivna i port slobodan aplikacija nastavlja dalje bez izuzetka (eng. *Exception*). U slučaju da soket igra ulogu servera onda se uređeni par (adresa, port) prosleđuju na *bind* metodu, dok za klijente je neophodno izvršiti povezivanje (*connect* metoda). Nakon toga razmena podataka je moguća tako što soket podatke šalje preko *send* metode, a prijem preko *recv* metode.



Slika 11. Princip funkcionisanja komunikacije između soketa

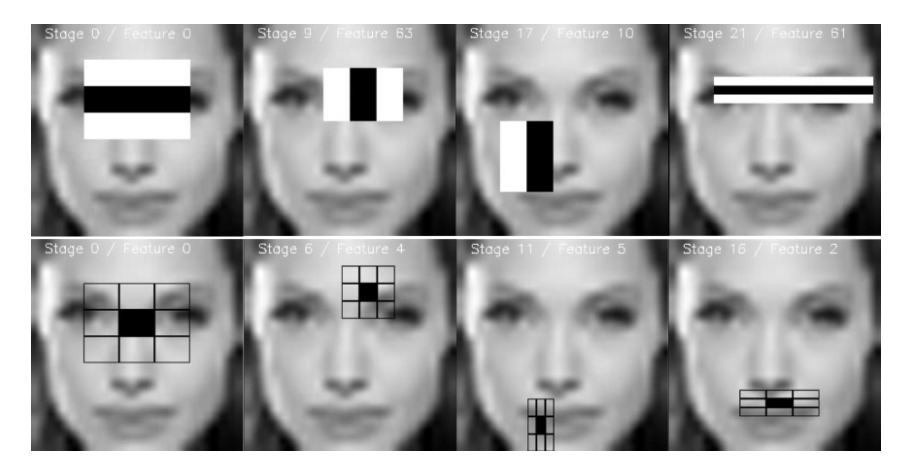
# 6 Klasifikatori i njihova upotreba

Kaskadni klasifikatori, odnosno kaskada pojačanih klasifikatora koji rade sa *HAAR*-sličnim karakteristikama, je specifični slučaj mašinskog učenja preko ansambl metode, koji se zove pojačavanje (eng. *Boosting*). Obično se oslanja na *Adaboost* klasifikatore (i druge modele kao što su *Real Adaboost*, *Gentle Adaboost* itd.).

Kaskadni klasifikatori se treniraju na nekoliko hiljada uzoraka slika koje sadrže objekat koji je potrebno detektovati. Takvi uzorci nazivaju se pozitivi. Takođe, pri trening procesu mora biti sadržano znatno više uzoraka slika na kojima se ciljani objekat ne nalazi, a ti uzorci se nazivaju negativi.

U slučaju detekcije lica, da bi klasifikator uspešno izvršio detekciju neophodno je da koristi alogirtam koji se zove *Viola-Jones* objektno-detekcioni okvir (eng. *Framework*), koji uključuje sledeće korake neophodne za detekciju lica uživo:

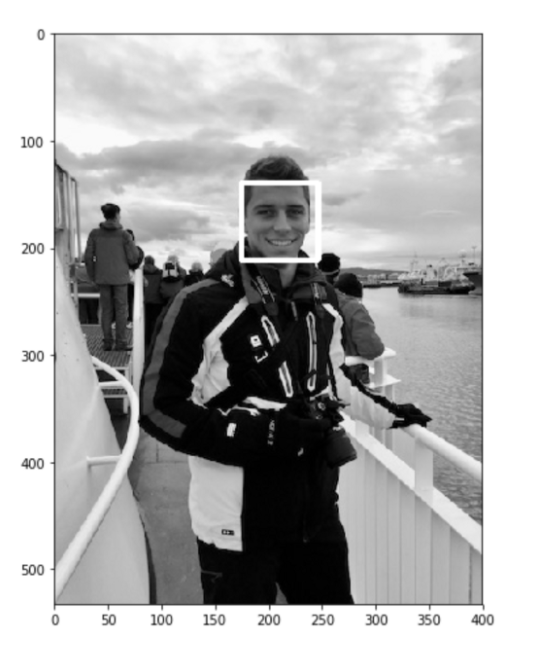
1. Odabir HAAR-karakteristika, izvedenih iz HAAR talasa
2. Kreiranje integralne slike
3. *Adaboost* treniranje
4. Kaskadno klasifikovanje



Slika 12. HAAR karakteristike koje klasifikator traži tokom treniranja

U okviru ovog projekta korišćen je gotovi *OpenCV* kaskadni klasifikator za prepoznavanje čovekovog lika. Klasifikatori su zapakovani u *XML* formatu, te se lako mogu prenositi i deskriptovati na različitim okruženjima. U okviru aplikacije se koriste preko *OpenCV* instance objekta *CascadeClassifier* tako što se prosleđuje putanja fajla. Nakon toga, proces detekcije je takav da se objektu prosleđuje dobijeni frejm, te ukoliko je detektovano jedno ili više lica vraća se niz regiona od interesa, odnosno regiona u kojem se detektovani objekat (čovekov lik) nalazi, u suprotnom povratni niz će biti prazan (*None* u *Python* programskom jeziku).

*OpenCV* klasifikator koji se koristi je poprilično brz, što znači da u većini slučajeva neće praviti dodatna kašnjenja ili ta kašnjenja neće biti od velikog značaja. Ova osobina ne mora da bude generalna za ostale klasifikatore, te je neophodno biti oprezan pri treniranju sopstvenog klasifikatora. Sam proces treniranje može da traje dugo, a rezultati ne moraju da budu na očekivanom nivou.

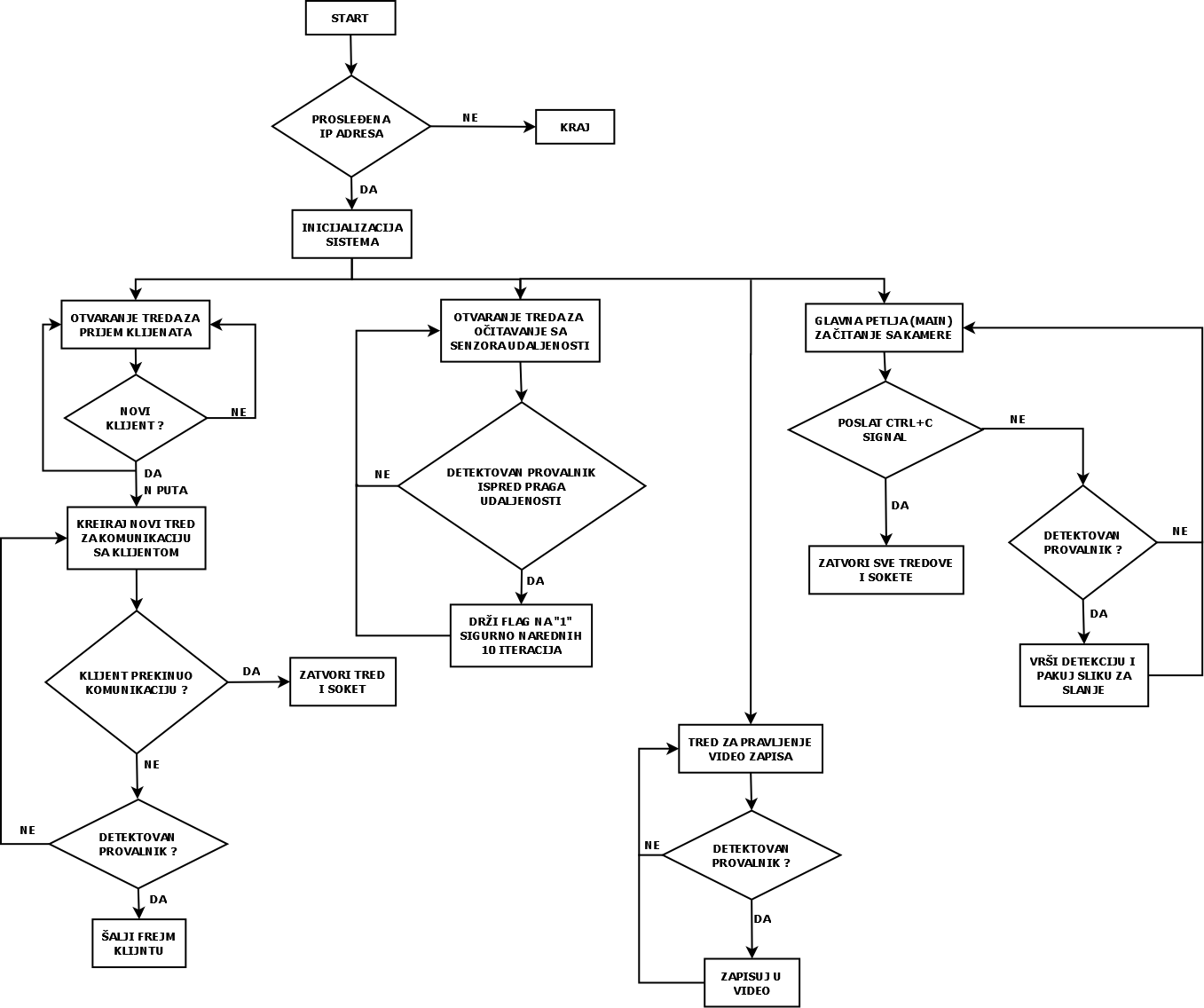


Slika 13. Primer detekcije čovekovog lika uz pomoć kaskadnog klasifikatora

# 7 Algoritam rada serverske aplikacije

U ovom poglavlju opisan je alogirtam rada serverske aplikacije. Početna tačka (eng. *Start*) počinje inicijalizacijom neophodnih zavisnosti, u koje spada:

* *OpenCV* instance zadužene za rukovođenje kamerom povezanom sa uređajem
* Postavljanje ulaznih i izlaznih uređaja (*GPIO* biblioteke *RaspberryPi*-a)
* Utičnice (eng. *Socket*) zadužene za komuniciranje putem *TCP* protokola



Slika 14. Blok dijagram algoritma rada serverske aplikacije

#RPi lib for distance measurement usecase

import RPi.GPIO as GPIO

# GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

#set GPIO direction (IN / OUT)

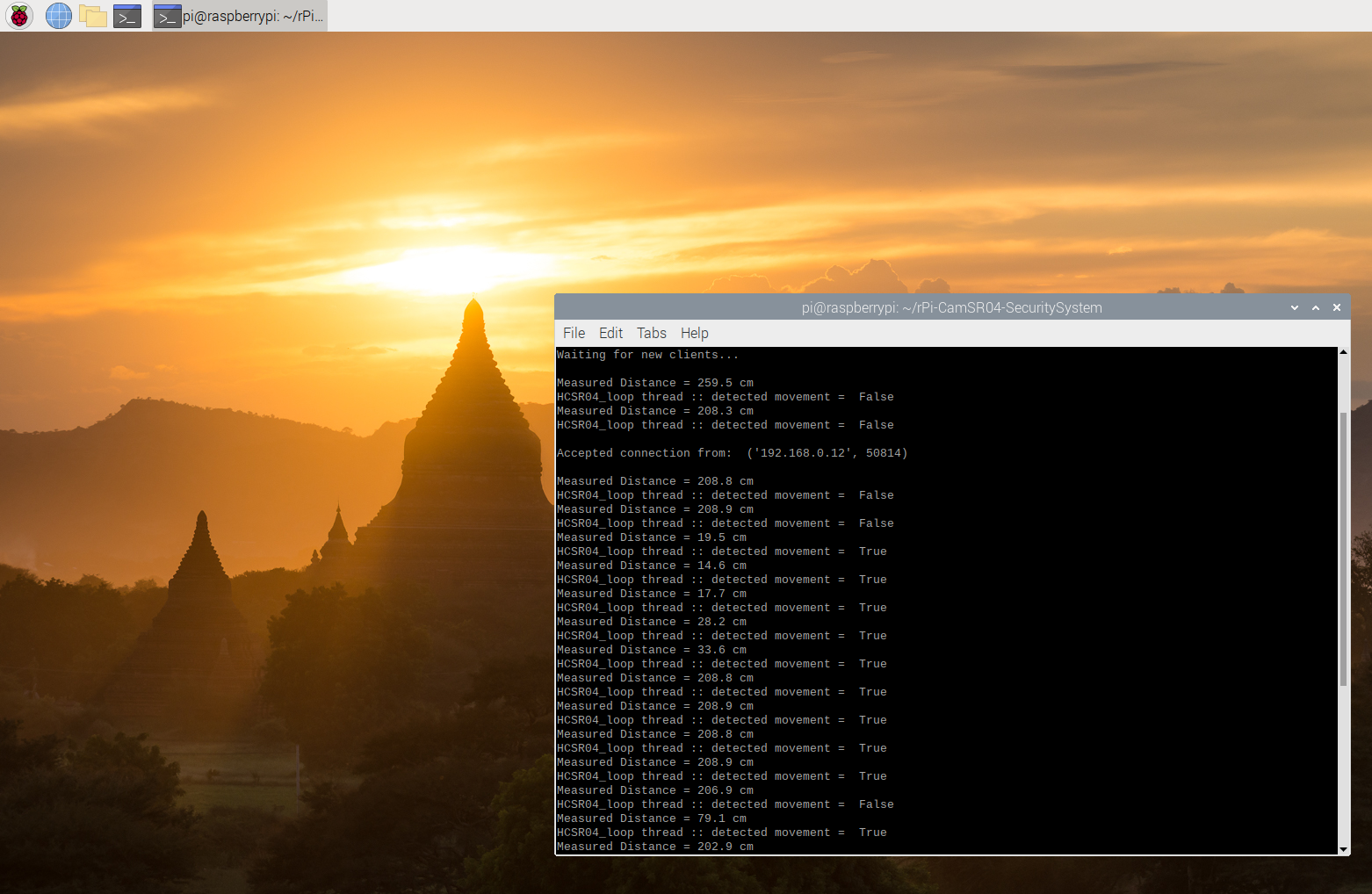
GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

*Deo koda u okviru kojeg se izvršava inicijalizacija ulaznih i izlaznih pinova RaspberryPi uređaja*

Naime, ukoliko nije prosleđen argument, odnosno *IP* adresa koja je dodeljena uređaju iz lokalne mreže, izvršava se terminacija, odnosno izlazak iz aplikacije. Ukoliko je soket dobio adresu i port na kojima može da otpočne komunikaciju sa potencijalnim klijentima, nastupa paralelno izvršavanje nekoliko beskonačnih petlji primenom principa više niti (eng. *Multithreading*).

U okviru jednog treda izvršava se očitavanje sa ultrasoničnog senzora, gde se šalje neophodan okidajući signal, radi proračun i ispisivanje izmerene udaljenosti prepreke od senzora u periodičnoj vremenskoj iteraciji od 500 *ms*. Ukoliko očitana vrednost sa senzora prelazi ispod postavljenog praga (80 *cm*) postavlja se vrednost globalne promenljive (eng. *Flag*) na logičku visoku vrednost (‘1’ – eng. *True*) i kao takav *flag* zagarantovano ostaje u nekoliko iteracija.



Slika 15. Printscreen ispisa detekcije udaljenosti i prijema novog klijenta

Ovaj *flag* aktivira rad senzora kamere u glavnom tredu i izvršavanje detekcije čovekovog lika, te pakovanje slike u binarni niz koji se šalje ka aktivnim klijentima, ukoliko ih u datom trenutku ima. Pre svega, radi uštede procesorskog vremena i boljih performansi aplikacije, ova petlja je pasivna sve do trenutka detekcije provalnika uz pomoć senzora udaljenosti iz gore navedene petlje.

while True:

    try:

        ret, frame = cap.read()

        if detected == True:

            gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

            faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

            for(x, y, w, h) in faces:

                cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 0, 255), 2)

                roi\_gray = gray[y:y+h, x:x+w]

                roi\_color = frame[y:y+h, x:x+w]

            sndMsg = True

            result, frame = cv2.imencode('.jpg', frame, encode\_param)

            data = pickle.dumps(frame, 0)

    except KeyboardInterrupt:

        break

*Deo koda u okviru kojeg se izvršava očitavanje sa senzora kamere i detekcija lica*

Pored ovog treda, ovaj *flag* pobuđuje slanje zapakovane slike na klijentskim tredovima čiji je broj ekvivalentan broju otvorenih klijentskih soketa. Kao i glavni tred, ovi tredovi su takođe pasivni sve do trenutka detekcije (postavljanja promenljive detekcije na logičku visoku vrednost). Pasivnost klijentskih tredova omogućena je zaključavanjem tredova uz pomoć *mutex* i *lock* instanci. Kada dobije signalizaciju da može podatke da pošalje ka klijentu, klijentski soket koristi neblokirajuću funkciju za slanje. Ukoliko bi došlo do diskonekcije sa klijentom, a koju bi prouzrokovala klijentska strana, unutar beskonačne petlje primenjuje se hvatanje izuzetka (eng. *Exception*) za bezbedan izlazak iz petlje i uspešno zatvaranje treda.

Da bi komunikacija sa klijentima bila moguća, tredovi zaduženi za interakciju sa klijentima se inicijalizuju u još jednom tredu u kojem se “dočekuju” novi klijenti. Naime, u ovom tredu nalazi se blokirajuća funkcija prihvatanja novog zahteva za konekciju od strane drugih uređaja. U tom trenutku funkcija zadužena za prihvatanje (eng. *Accept*) kao povratnu vrednost vraća uređeni par koji čine *IP* adresa uređaja i brojčana vrednostotvorenog soketa.

def clientReceivement():

    global sndMsg

    sndMsg = False

    print ("\nWaiting for new clients...\n")

    while True:

        try:

           (client, address) = serversock.accept()

        except OSError:

            break

        th.append(Thread(target=listener, args = (client,address)) )

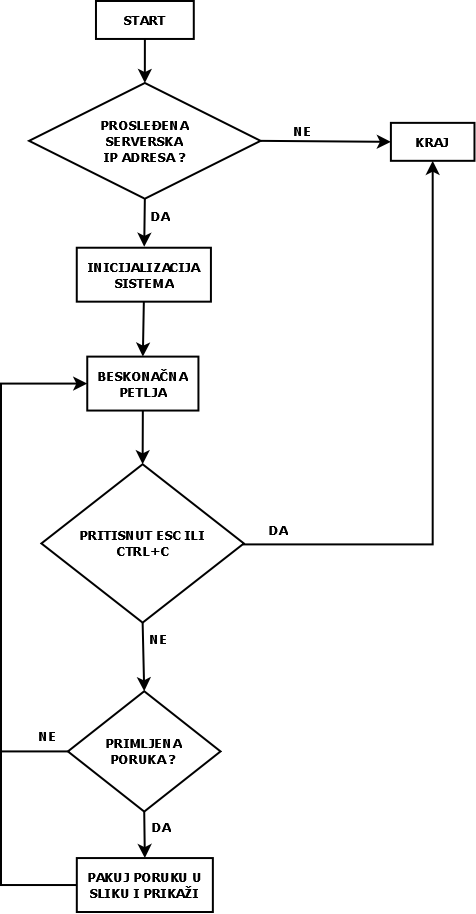
        th[-1].start()

*Deo koda u okviru kojeg se izvršava primanje novih klijenata*

Svaka nova detekcija provalnika zapisuje se u video zapisu koji nosi naziv trenutka njegovog kreiranja (eng. *Timestamp*), a sam video je u *MP4* formatu i skladišti se u eksternoj memoriji uređaja *SD* kartici. Administrator može u bilo kom trenutku prekinuti rad aplikacije “ubijajućim” signalom koji se, između ostalog, može napraviti pritiskom tastera *Ctrl* + *C* na tastaturi. U tom trenutku se izvršava bezbedan izlazak iz aplikacije, što znači da se svi objekti deinicijalizuju, a tredovi zatvore. Da bi se “ubila” prihvatajuća funkcija, šalje se signal za prekid osluškivanja (eng. *Shutdown*) ka glavnoj serverskoj soket instanci, ali i ka ostalim aktivnim klijentima, te se izvršava prekid (eng. *Break*) beskonačnih *While* petlji.

# 8 Algoritam rada klijentske aplikacije

Klijentska aplikacija, kako joj samo ime kaže, zadužena je da se poveže na privatni server i omogući korisniku da isprati rad servera, ukoliko je došlo do provale. Na osnovu ovih uloga može se zaključiti da je klijentska aplikacija znatno jednostavnija od serverske.



Slika 16. Blok dijagram algoritma rada klijentske aplikacije

U slučaju ove aplikacije, startna tačka isključuje inicijalizaciju senzora, odnosno *GPIO* pinova, te pod nju spada samo inicijalizacija klijentskog soketa i neophodnih *OpenCV* instanci. U slučaju klijentske aplikacije takođe je potrebno proslediti *IP* adresu serverskog uređaja i komunikacionog porta sa kojeg se komunikacija očekuje, ukoliko se ne prosledi pravilna *IP* adresa dolazi do izlazka iz aplikacije. Nakon što je klijentski soket uspešno povezan sa serverom započinje komunikacija.

U ustaljenom režimu rada, klijentska aplikacija očekuje prijem nove slike (niza bajtova) uz pomoć prijemske funkcije (eng. *Receive*), koja je, kao i slušajuća funkcija na serverskoj strani, blokirajuća, te svi ostali procesi čekaju da ova funkcija primi novu poruku. Kada je server poslao poruku, poruka se ponovo pakuje u matricu u kojoj svaki bajt predstavlja boju nekog piksela. Takva slika se dalje prosleđuje na ekranizaciju uz pomoć *OpenCV* funkcije za prikaz (eng. *Image* *Show*), te se opciono može sačuvati kao video na serverskoj strani.

while True:

    while len(data) < payload\_size:

        data += client\_socket.recv(4096)

    packed\_msg\_size = data[:payload\_size]

    data = data[payload\_size:]

    msg\_size = struct.unpack(">L", packed\_msg\_size)[0]

    while len(data) < msg\_size:

        data += client\_socket.recv(4096)

    frame\_data = data[:msg\_size]

    data = data[msg\_size:]

    frame=pickle.loads(frame\_data, fix\_imports=True, encoding="bytes")

    finalFrame = cv2.imdecode(frame, cv2.IMREAD\_COLOR)

    FrameRcv = True

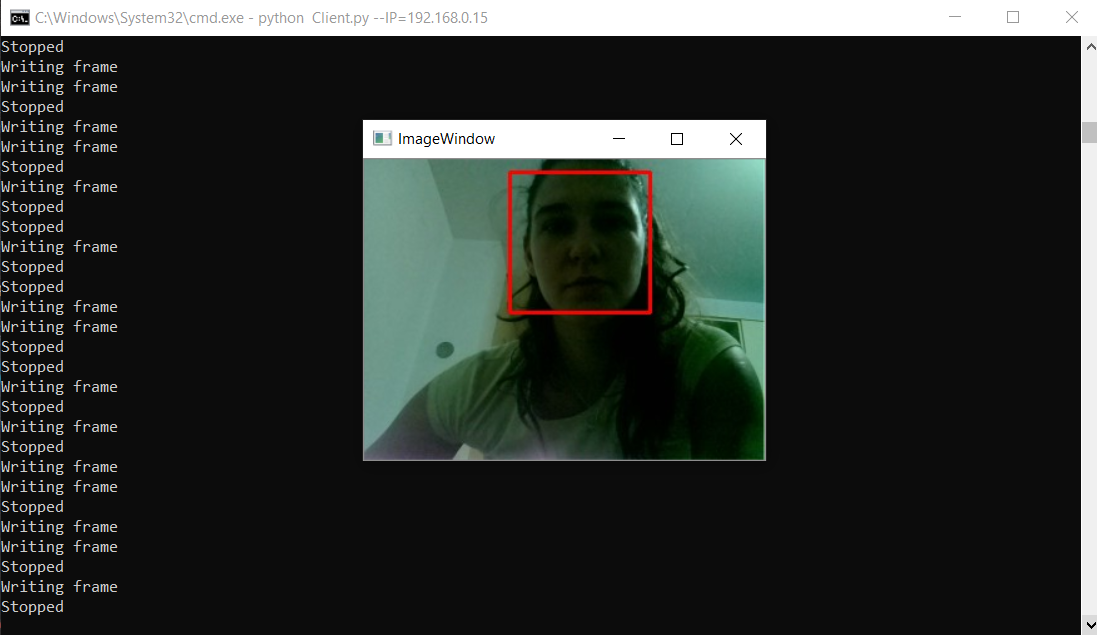
    cv2.imshow('ImageWindow',finalFrame)

    k = cv2.waitKey(1) & 0xff

    if k == 27:

        break

*Deo koda u okviru kojeg se izvršava prijem frejmova od servera i njihovo prikazivanje*



Slika 17. Printscreen prikaza detekcije lica

Izlazak iz aplikacije moguć je u bilo kom trenutku pritiskom tastera *ESC* ili “ubijanjem” sa *Ctrl + C* tastera tastature.

# 9 Zaključak

Nakon završeve implementacije celokupnog sistema, a potom i verifikacije funkcionalnosti može se zaključiti da sistem uspešno izvršava sve predviđene zadatke. Serverski uređaj uspešno prima nove klijente i pohranjuje ih potrebnim podacima u gotovo realnom vremenu, a pritom rukovodi i detekcijom sa senzora udaljenosti, kao i prepoznavanja uz pomoć kaskadnog klasifikatora. Performanse uređaja su na zavidnom nivou, s obzirom da se izvršavaju na malom računaru znatno slabijih performansi od modernih mašina, a protok podataka ne trpi zagušenja na većem broju aktivnih klijenata. Klijentska aplikacija podjednako dobro izvršava svoju funkcionalnost na obe predviđene platforme (*Windows* i *Linux*), te korisniku “olakšava život” brzim pristupom i otvaranjem iste.

U ovom projektu korišćena je moderna verzija *Python* programskog jezika (*Python3*) na obe strane, gde je najveći izazov zapravo bio sama oprimizacija sistema i paralelnog izvršavanja više funkcija. Pre svega, *Multi-threading* je kumovao takvom rezultatu.

Projekat pruža velik dijapazon potencijalnih unapređenja. Pre svega, na serverskoj strani bi se mogao povećati bezbednosni sistem različitim hardverskim dodacima, kako senzorima (npr. veći broj kamera), tako i aktuatorima (npr. zvučnim alarmima). Softverska strana bi mogla da uključi i prepoznavanje ukućana, te da ne pravi uzbunu pri njihovoj detekciji što bi moglo biti rađeno uz pomoć različitih biblioteka, među kojima je najpopularnija *Tensorflow* biblioteka.

Sa druge strane, klijentska aplikacija bi pre svega mogla da omogući pristup i sa mobilnih uređaja, odnosno da bude podržana na *Android* operativnom sistemu, ali i da trenutna aplikacija preraste sa konzolne u *GUI* aplikaciju.

# 10 Literatura

1. <https://sr.wikipedia.org/sr-ec/Raspberry_Pi> – *RaspberryPi*, *Wikipedia* članak, pristupljeno: septembar 2020.
2. [https://sr.wikipedia.org/sr-ec/Internet\_soket](https://sr.wikipedia.org/sr-ec/Internet_soket/) – Soket, *Wikipedia* članak, pristupljeno: septembar 2020.
3. <https://realpython.com/python-sockets/?fbclid=IwAR3ZFysBQGK0TBJp3MLAdu-GqH-IXn5S2SuLKivGhrG71LlyWvY_hMIdD5Q> – Soket programiranje u *Python* programskom jeziku, *RealPython* članak, Nathan Jennings, pristupljeno: septembar 2020.
4. <https://en.wikipedia.org/wiki/Network_socket> - Mrežni soket, *Wikipedia* članak, pristupljeno: septembar 2020.
5. <https://opencv.org/about/> - O *OpenCV* biblioteci, zvanična *OpenCV* stranica, pristupljeno: septembar 2020.
6. <https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1> – Uputstvo za primenu detekcije lica na *Python* programskom jeziku, *Towards Data Science* članak, Mael Fabian, pristupljeno: septembar 2020.
7. <https://www.raspberrypi.org/products/camera-module-v2/> - *RaspberryPi Camera Module V2*, zvanična *RaspberryPi* stranica o proizvodu, pristupljeno: septembar 2020.