

UNIVERZITET “DŽEMAL BIJEDIĆ”

FAKULTET INFORMACIJSKIH TEHNOLOGIJA

ZAVRŠNI RAD

**Detekcija saobraćajne trake za autonomno vođenje robota**

(Road lane detection for autonomous robot guidance)

**Profesor:** **Student:**

Doc.dr. Elmir Babović Faris Hasković, IB150201

Akademska godina: 2019/2020

Mostar, 2020. godina

IZJAVA O AUTORSTVU

Ja, **FARIS (SEAD) HASKOVIĆ**, student Fakulteta informacijskih tehnologija, Univerziteta "Džemal Bijedić" u Mostaru, pod punom moralnom, materijalnom i krivičnom odgovornošću,

**Izjavljujem**

da je rad pod naslovom:

**DETEKCIJA SAOBRAĆAJNE TRAKE ZA AUTONOMNO VOĐENJE ROBOTA**

u potpunosti rezultat sopstvenog istraživanja, gdje su korišteni sadržaji (tekst, ilustracije, tabele itd.) drugih autora jasno označeni pozivanjem na izvor i ne narušavaju bilo čija vlasnička ili autorska prava.

U Mostaru, 24.02.2020.

FARIS HASKOVIĆ, IB150201

**Sadržaj**

[**SAŽETAK** 1](#_Toc33454121)

[**ABSTRACT** 2](#_Toc33454122)

[**1.** **UVOD** 3](#_Toc33454123)

[**2.** **ARDUINO** 4](#_Toc33454124)

[2.1 Programiranje 5](#_Toc33454125)

[2.2 Serijska komunikacija 6](#_Toc33454126)

[2.2.1 Osnovne funkcije 6](#_Toc33454127)

[2.3 Arduino IDE 7](#_Toc33454128)

[2.3.1 Library manager 8](#_Toc33454129)

[2.3.2 Serial monitor 9](#_Toc33454130)

[2.3.3 Serial plotter 9](#_Toc33454131)

[2.4 Varijable 10](#_Toc33454132)

[2.5 Konstante 11](#_Toc33454133)

[2.6 Funkcije 12](#_Toc33454134)

[2.7 Vrste ploča 13](#_Toc33454135)

[2.8 Pinovi 16](#_Toc33454136)

[**3.** **Računarski vid** 17](#_Toc33454137)

[3.1 Modeli boja 18](#_Toc33454138)

[3.2 OpenCV Biblioteka 21](#_Toc33454139)

[3.2.1 Osnovne klase, funkcije i operatori 22](#_Toc33454140)

[3.3 Binarne slike 23](#_Toc33454141)

[3.4 Canny algoritam 24](#_Toc33454142)

[3.5 Morfološke transformacije 25](#_Toc33454143)

[3.5.1 Erozija 26](#_Toc33454144)

[3.5.2 Dilacija 26](#_Toc33454145)

[3.5.3 Otvaranje 27](#_Toc33454146)

[3.5.4 Zatvaranje 27](#_Toc33454147)

[3.6 Hough transformacija 27](#_Toc33454148)

[**4.** **Aplikacija za računarski vid** 29](#_Toc33454149)

[4.1 Dizajn 29](#_Toc33454150)

[4.2 Kod 30](#_Toc33454151)

[4.2.1 Serijsko slanje 38](#_Toc33454152)

[**5.0 Arduino aplikacija** 40](#_Toc33454153)

[**6.0 Robot** 45](#_Toc33454154)

[6.1 HC-SR04 ultrazvučni modul 45](#_Toc33454155)

[6.2 Konstrukcija 48](#_Toc33454156)

[6.2.1 Pleksiglas ploča 50](#_Toc33454157)

[6.2.2 Metalni okvir 51](#_Toc33454158)

[6.2.3 Dvostruki H-Most 51](#_Toc33454159)

[6.2.4 Točkovi i metalna šipka 52](#_Toc33454160)

[6.3 Napajanje 53](#_Toc33454161)

[6.4 Kamera 53](#_Toc33454162)

[**7. Kontrolisano okruženje** 54](#_Toc33454163)

[**8. Rezultati testiranja** 54](#_Toc33454164)

[**9. Budućnost projekta** 56](#_Toc33454165)

[**10. Zaključak** 56](#_Toc33454166)

[**11. Slike** 57](#_Toc33454167)

[**12. Programski kodovi** 58](#_Toc33454168)

[**Literatura** 60](#_Toc33454169)

# **SAŽETAK**

Tema ovog rada je izrada softver rješenja za vođenje robota koji će upotrebom moderne tehnologije i metode 'computer vision-a' odnosno računarskog vida omogućiti autonomno kretanje istog unutar kontrolisanog okruženja. Računarski vid kao područje umjetne inteligencije postaje uobičajeno za kontrolisanje različitih procesa industrijskih robota, autonomnih vozila, video nadzora (prepoznavanje lica, brojanje ljudi, vozila u saobraćaju), analiziranja medicinskih slika, vojnih i bespilotnih letjelica, itd., te se njegovom primjenom u brojnim oblastima značajno olakšava i ubrzava procese koji su prethodno zahtijevali aktivnu ljudsku asistenciju, te spašava veliki broj ljudskih života (npr. deminerski roboti, autonomne letjelice).

U prvim dijelovima rada, opisana je teorija računarskog vida, Arduino razvojnog okruženja, te različite tehnike za transformaciju slike. U praktičnom dijelu rada pojašnjene su funkcionalnosti aplikacije za detekciju saobraćajne trake za autonomno vođenje robota. Kroz ostale dijelove predstavljaju se komponente sistema.

Za korištenje aplikacije nije neophodno informatičko znanje, kako je pokretanje aplikativnih i hardverskih mogućnosti robota moguće jednom tipkom, dok je sa druge strane za modificiranje bilo kakvih vrijednosti aplikacije neophodno napredno informatičko znanje, jer se sve modifikacije vrše unutar programskog koda.

Ključne riječi: Softver, hardver, robotika, autonomija, saobraćajna traka, detekcija, OpenCV, računarski vid, procesiranje slike.

# **ABSTRACT**

Topic of this project is building a software solution for robot guidance which will, by using the modern technology and methods of computer vision, provide the autonomous guidance in the controlled environment. Computer vision as a field of artificial intelligence is becoming a common solution for controlling different kinds of processes in the fields of robotics, autonomous vehicles, video surveillance (face recognition, counting people, traffic), analyzing medical images, military and unmanned aircraft, etc., and using computer vision in many fields significantly eases and speeds up the processes which previously required active human assistance, and also saving lives (for example mines removal and autonomous flying).

In the beginning parts of this paper, theory of computer vision is explained, as well as Arduino development environment, and different sorts of techniques for image transformation. Functionalities of automatic robot guidance application are explained in the practical part of the paper. Other parts of the paper present the systems components.

For using this application, no technical knowledge is needed, as starting the application and hardware capabilities of the robot is possible using a single key-press, on the other hand modifying any values of the application requires advances technical knowledge, as all modifications are done using the code.

Keywords: Software, hardware, robotics, autonomy, road lane, detection, OpenCV, computer vision, image processing.

# **UVOD**

Mobilni roboti, ili specifično, područje za autonomno vođenje robota je izučavano dugi niz godina, te ostaje sektor robotike sa velikom aktivnošću. Živimo u vremenu brzog razvoja novih tehnologija i samim time, konstantnim mogućnostima poboljšanja ovakvih vrsta sistema. Područje računarskog vida postaje sve češći dio naših svakodnevnica, korišten u svim sferama ljudskih života, stoga vrlo je jasna važnost daljnjeg izučavanja ove oblasti.

Softver kreiran tokom ovog projekta je u mogućnosti pratiti putanju, nalik saobraćajnoj traci, te prepoznati prepreke i zaustaviti svoje kretanje. Detekcija se sastoji od određivanja relativne pozicije između robota i saobraćajnih traka, te analize kretanja robota. Nepredvidljivost okruženja čini ovaj zadatak iznimno teškim, te se iz tog razloga ovaj rad fokusira na specifično kontrolisano okruženje.

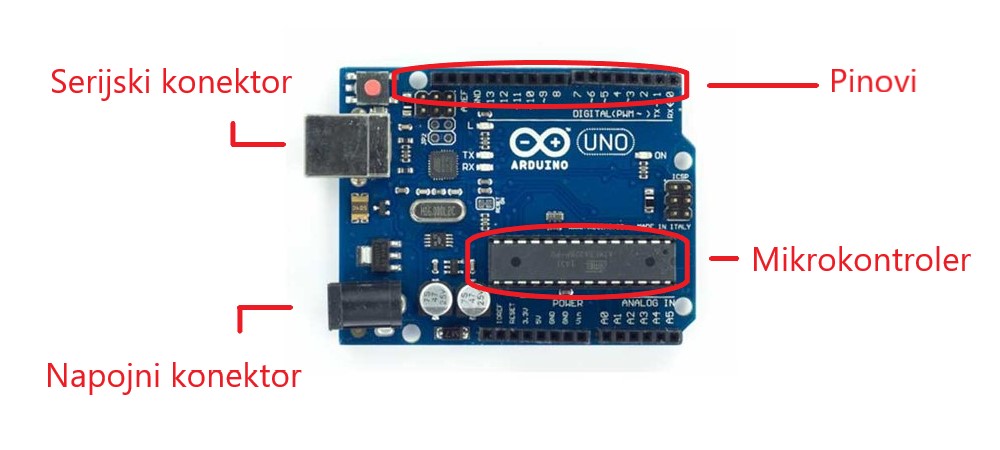
U svrhu testiranja implementiranog softvera, isti je implementiran na robotsku platformu, koja se sadrži od dva motora, kontrolnih ploča, baterije, eksterne kamere, te laptopa koji će biti postavljen na samu platformu.

Moguća primjena ovog softvera je za sve transportne potrebe gdje je moguće napraviti kontrolisano okruženje, npr. u skladištima, za kreiranje autonomnih viljuškara za prevoz tereta ili na aerodromima za autonomni prijevoz prtljaga.

# **ARDUINO**

Arduino je naziv za razvojno okruženje otvorenog koda čija je primarna svrha stvaranje jednostavnih alata za stvaranje digitalnih projekata.[[1]](#footnote-1) Arduino se sastoji od većeg broja dijelova i interfejsa spojenih zajedno na jednoj printanoj ploči. Danas je moguće pronaći brojne varijacije Arduino razvojnih okruženja (Uno, Nano, Mega, Leonardo itd.), od kojih neke sadrže i dodatne dijelove, međutim na osnovnoj ploči naći ćete sljedeće dijelove:

* Pinovi (najčešće 14): Pinovi služe za povezivanje raznih komponenata koje je moguće koristiti unutar Arduino razvojnog okruženja, ovi pinovi su podijeljeni u dvije kategorije:
  + Analogna vrsta pinova u mogućnosti je čitati različite vrjednosti, te se najčešće koriste za precizniju kontrolu (npr. jačina svjetlosti LED diode). Obično se može pronaći 6 analognih pinova na jednoj Arduino ploči.
  + Digitalni pinovi koji su zaduženi samo za upaljeno ili ugašeno stanje (npr. upaljena ili ugašena LED dioda). Obično se može pronaći 14 digitalnih pinova na jednoj Arduino ploči.
* Mikrokontroler: Glavni čip koji omogućava programiranje Arduino-a, kako bi bio u mogućnosti izvršavati komande i praviti odluke zavisno od unosa. Čipov zavise od tipa Arduino-a, ali najčešće sadrži Atmel kontrolere od kojih su najpopularniji ATmega8, ATmega168 i ATmega328. Glavna razlika između njih je količina ugrađene memorije.
* Serijski konektor: Ovaj konektor je na većini novih ploča implementiran kao standardni USB port, te omogućava komunikaciju sa računarom, kao i učitavanje programa na uređaj.
* Napojni konektor: Pruža energiju samom uređaju, te nisku voltažu za napajanje spojenih komponenti.



Slika 1: Arduino razvojna ploča

## 2.1 Programiranje

Arduino kod je pisan u Arduino razvojom okruženju te programskom jeziku zvanom processing koji je baziran na veoma jednostavnom harverskom programskom jeziku sličnom programskom jeziku C.

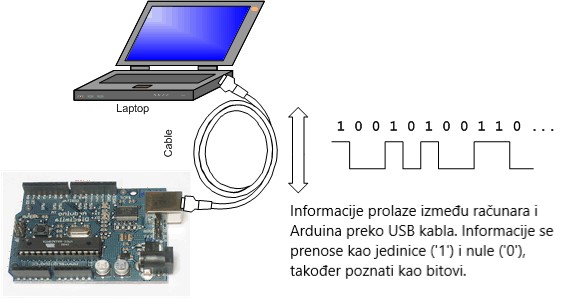
Nakon pisanja sketcha (sinonim za Arduino program), potrebno je poslati ga Arduino ploči na izvršenje. Sketchevi sadrže ekstenziju .ino.

Struktura Arduino programa sadrži minimalno 2 funkcije:

* Setup – Sadrži osnovne dijelove koda koji se izvršavaju samo jednom, kao npr. za inicijalizaciju pinova ili započinjanje serijske komunikacije.
* Loop – Nakon što je Setup funkcija izvršena, sadrži kod koji se izvršava unutar beskonačne petlje.

## 2.2 Serijska komunikacija

Oblasti telekomunikacija predstavljaju serijsku komunikaciju kao proces slanja podataka bit po bit, sekvencijalno, preko komunikacijskog kanala ili kompjuterske sabirnice. Suprotnost serijskoj komunikaciji jeste paralelna komunikacija, gdje se nekoliko bita šalje istovremeno na više paralelnih kanala. Serijska komunikacija posebno je praktična gdje troškovi čine paralelnu komunikaciju nepraktičnom. Serijski kompjuterski sabirnici postaju sve češći, jer poboljšani signal i brzine prijenosa u novijim serijskim tehnologijama počinju da nadmašuju paralelne sabirnice (nije potreban serializer i deserializer). Unutar ovog projekta korištena je serijska komunikacija između računara (C++ aplikacije) i Arduino-a.



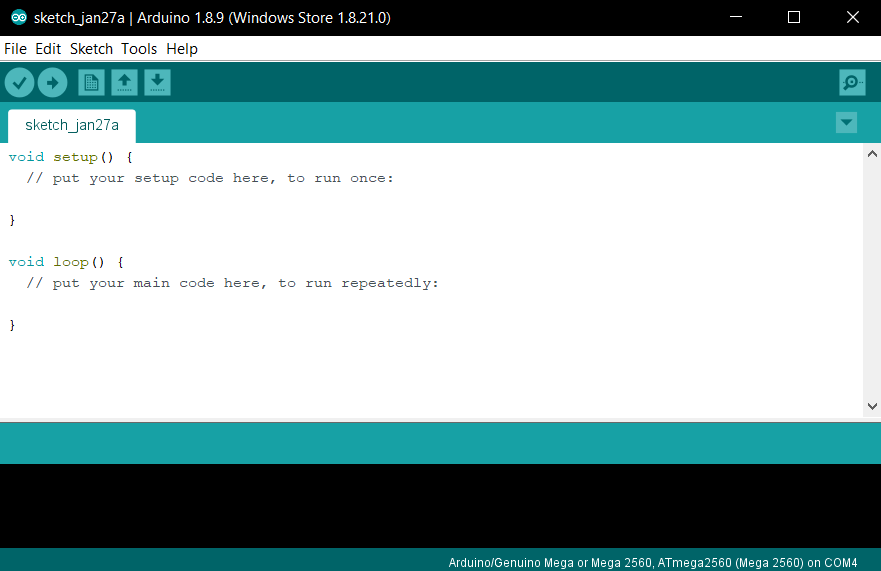
Slika 2: Prikaz serijske komunikacije

### 2.2.1 Osnovne funkcije

* if(serial) – Označava da je specificirani Serijski port spreman. Povratna vrijednost je boolean vrijednost (u zavisnosti od toga da li je serijski port spreman)
* Serial.availabe() – Vraća broj bajtova (karaktera) dostupnih za čitanje sa serijskog porta. Ovo su podaci koji su već stigli i primljeni u serijskom bufferu koji može čuvati 64 bajta. Funkcija kao parametar sadrži Serial port objekat.
* Serial.begin() – Podešava brzinu prenosa u bitovima po sekundi za serijski prijenos podataka. Opcionalni drugi parametar podešava podatke, paritet i stop bitove. Ukoliko se izostavi parametar, zadane vrijednosti su 8 bitova podataka, bez pariteta i 1 stop bit.
* Serial.read() – Čita dolazne serijske podatke. Vraća prvi bajt dostupnih serijskih podataka (ili -1 ako nema dostupnih podataka).
* Serial.write() – Upisuje Binarne podatke na serijski port. Ovi podaci se šalju kao bajtovi ili niz bajtova.
* serialEvent() – Poziva se kada su podaci dostupni. Uz ovu funkciju se koristi Serial.read() za preuzimanje podataka.

## 2.3 Arduino IDE

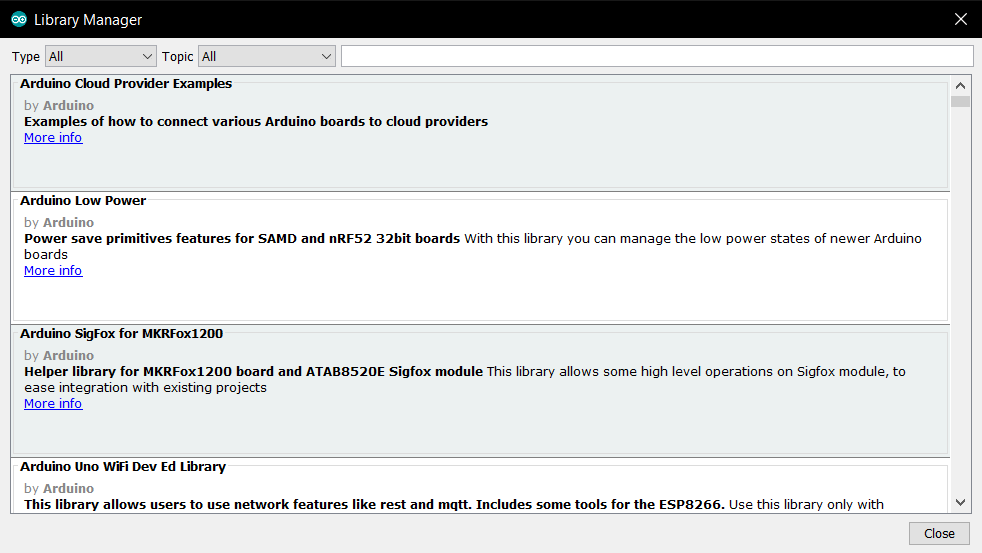
Arduino IDE (Integrated Development Environment) koristi koncept tzv. Sketchbooka, mjesta za pisanje Arduino programa.



Slika 3: Arduino IDE

### 2.3.1 Library manager

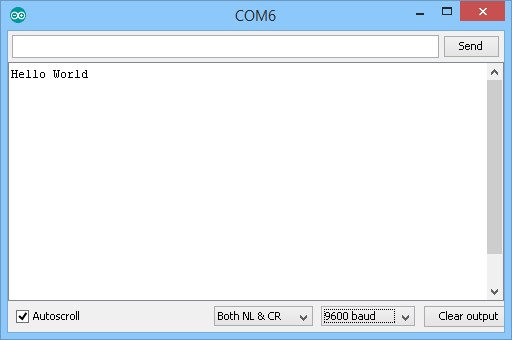
Za dodatne funkcionalnosti Arduino sketcheva, moguće je naknadno dodati biblioteke. Jedan od načina za dodavanje istih jeste korištenje alata pod nazivom Library Manager.



Slika 4: Library manager

Nakon dodavanje biblioteke, Sketch se automatski proširuje sa jednim ili više #include dijelova na početku sketcha. Biblioteke se prilikom procesa slanja podataka na ploču, šalju zajedno sa našim Sketchom, te će zauzeti dodatnu memoriju.

### 2.3.2 Serial monitor

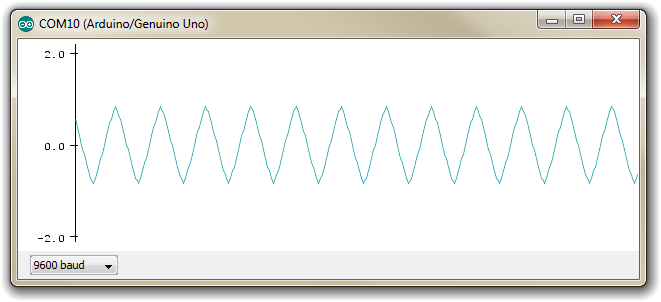


Slika 5: Serial monitor

Serial monitor je još jedan alat koji omogućava slanje i prikaz podataka u serijalnoj komunikaciji sa Arduino razvojnim okruženjem. Baud rate je neophodno podesiti na odgovarajući nivo koji je podešen unutar našeg sketcha sa Serial.begin() funkcijom. Za slanje podataka dovoljno je unijeti tekst, te pritisnuti dugme Send ili tipku Enter na tastaturi.

### 2.3.3 Serial plotter

Serial Plotter je posljednje dodan alat unutar Arduino razvojnog okruženja. Omogućava prikaz serijskih podataka u stvarnom vremenu, te daje mnogo ljepši i pregledniji način vizualizacije serijske komunikacije. Serial plotter je alat koji je dostupan offline (nije neophodna internet konekcija), te zbog ljepšeg prikaza često omogućava lakše pronalaženje grešaka u kodu (debugging). Omogućava različite boje za svaku varijablu, te podržava grafikone negativnih vrijednosti.



Slika 6: Serial plotter

## 2.4 Varijable

Varijable omogućavaju način davanja naziva i pohrane vrijednosti, za kasnije korištenje unutar programa. Osnovni tipovi varijabli unutar Arduino okruženja su sljedeći[[2]](#footnote-2):

* char – Tip podatka koji omogućava pohranu vrijednosti karaktera. Karakteri se pohranjuju kao brojevi, koji odgovarjau svojim vrijednostima unutar ASCII (American Standard Code for Information Interchange) tablice. Veličina char tipa podatka je najmanje 8 bita
* byte – Byte pohranjuje 8 bitni broj bez predznaka, od 0 do 255.
* int – Integer je glavni tip podataka za pohranu brojeva. Na Arduino Uno i Atmega baziranim pločama, integer može čuvati 16 bitne (2 bajta) vrijednosti, te omogućava pohranu vrijednosti od -32,768 do 32,767.
* unsigned int – Može čuvati 16 bitne (2 bajta) vrijednosti, kao i int. Glavna razlika je u tome što unsigned ne čuva negativne vrijednosti, te može čuvati iste od 0 do 65,535.
* long – Tip podataka sa proširenom veličinom, čuva 32 bita (4 bajta), te može pohraniti vrijednosti od 2,147,483,648 do 2,147,483,647.
* unsigned long – Također kao i long, unsigned long čuva 32 bita (4 bajta), te bez negativnih vrijednosti može čuvati vrijednosti od 0 do 4,294,967,295.
* float – Float vrijednosti mogu čuvati brojeve sa decimalnim vrijednostima. Nude veću preciznost od int vrijednosti, te se pohranjuju kao 32 bit-a (4 bajta).
* double – Omogućavaju duplu preciznost iza decimalnog zareza naspram floata na Arduino Due pločama gdje se pohranjuju kao 64 bitne (8 bajta) vrijednosti. Međutim na Uno i Atmega baziranim pločama zauzimaju 4 bajta, te ne omogućavaju dodatnu preciznost.
* String – Tip podatka koji omogućava čuvanje teksta ili niza karaktera. Nema određenu veličinu, jer se proširuje zavisno od količine podataka koje čuva.

## 2.5 Konstante

* Integer Konstante – Ova vrsta konstanti korisna je za jednostavnu konverziju baza brojeva. Početno, integer konstante se tretiraju sa bazom 10 (decimalni brojevi), ali postoje specijalne notacije koje možemo koristiti za konverziju baze brojeva, a to su:
  + Binarni – Notacija B, npr. B1000101, validni karakteri su samo 0 i 1.
  + Oktalni – Notacija 0, npr. 0554 – odgovara broju 364 u decimalnom sistemu. Validni karakteri su od 0-7.
  + Hexadecimalni – Notacija 0x, npr. 0x32 – odgovara broju 50 u decimalnom sistemu, validni karakteri su brojevi od 0-9 i slova A-F. Slova su osjetljiva na veličinu, odnosno mala i velika slova imaju različite vrijednosti.
* HIGH | LOW – Tokom pisanja ili čitanja na digitalni pin, mogu biti samo 2 stanja, HIGH ili LOW.

Kada je pin postavljen na INPUT sa pinMode() i read sa digitalRead(), Atmega Arduino će imati HIGH u dva slučaja:

* + Kada je voltaža veća od 3.0V prisutna na pinu 5 voltnih ploča
  + Kada je voltaža veća od 2.0V prisutna na pinu 3.3 voltnih ploča

Kada je pin postavljen na output sa pinMode() i postavljen na HIGH sa digitalWrite() funkcijom, pin je na 5V na 5 voltnim pločama i 3.3V na 3.3 voltnim pločama.

LOW također može imati različite značenja u zavisnosti od toga u kojem kontekstu se koristi. Ukoliko je pin postavljen na INPUT koristeći pinMode() i read sa digitalRead(), atMega Arduino će imati LOW u slučajevima:

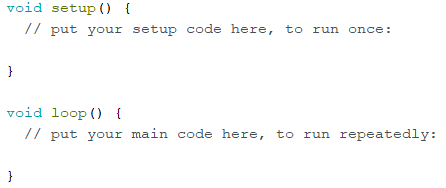
* Kada je voltaža manja od 1.5V prisutna na pinu 5 voltnih ploča
* Kada je voltaža manja od 1.0V prisutna na pinu 3.3 voltnih ploča

Kada je pin postavljen na output koristeći pinMode() i u LOW stanju koristeći digitalWrite() funkciju, pin je na 0 volti u svim slučajevima.

* INPUT | OUTPUT – Digitalni pinovi se mogu koristiti kao INPUT ili OUTPUT. Mijenjanje se vrši pomoću pinMode() funkcije, obično u Setup dijelu Sketcha. Ove izmjene mijenjaju ponašanje pina.

## 2.6 Funkcije

Kreiranjem novog sketcha unutar Arduino IDE-a, kreiraju se 2 osnovne funkcije.



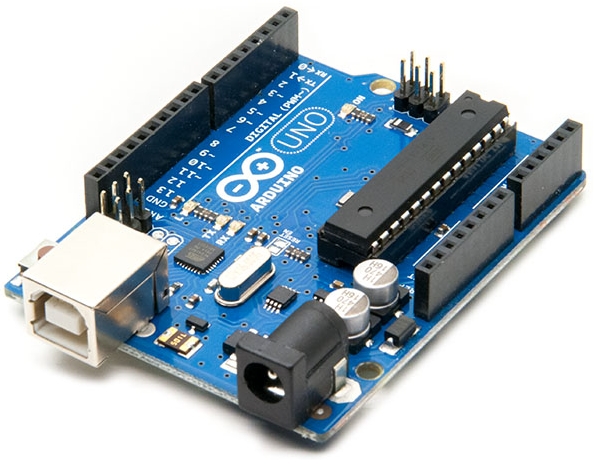
Slika 7: Početne funkcije

* setup() – Jedna od dvije osnovne funkcije koja se izvršava samo jednom npr. za osnovno podešavanje stanja pinova ili za deklarisanje početka serijalne komunikacije i podešavanje baud rate-a.
* loop() – Druga osnovna funkcija, koja izvršava kod u beskonačnoj petlji.
* pinMode() – Funkcija za podešavanje ponašanja pina, za prvi parametar koristi broj pina, a za drugi INPUT ili OUTPUT rezervisanu riječ, zavisno od toga da li ovaj pin želimo koristiti za primanje ili slanje.
* digitalRead() – Čita vrijednost sa specificiranog pina koji je ujedno i jedini parametar ove funkcije, te vraća HIGH ili LOW.
* digitalWrite() – Upisuje stanje na digital pin, prvi parametar ove funkcije jeste pin, a drugi parametar vrijednost HIGH ili LOW.
* analogRead() – Čita vrijednost sa specificiranog analog pina. Arduino mapira voltažu između 0 i maksimalne voltaže ploče (5V ili 3.3V) i pretvara ih u brojčane vrijednosti od 0-1023.
* analogWrite() – Upisuje analognu PWM (pusle with modulation) vrijednost na pin. Veoma korisno ukoliko ne želimo slati maksimalnu voltažu na pin, kao što je slučaj sa digital pinovima, gdje je moguće koristiti samo HIGH ili LOW vrijednosti. Zahvaljujući ovoj mogućnosti, moguće je motoru dati različite brzine ili osvijetliti LED na različite jačine.
* delay() – Pauzira program na specificirani broj milisekundi koje koristi kao svoj jedini parametar.

Za osnovne funkcije serijske komunikacije, pogledajte 3.2.1 Osnovne funkcije.

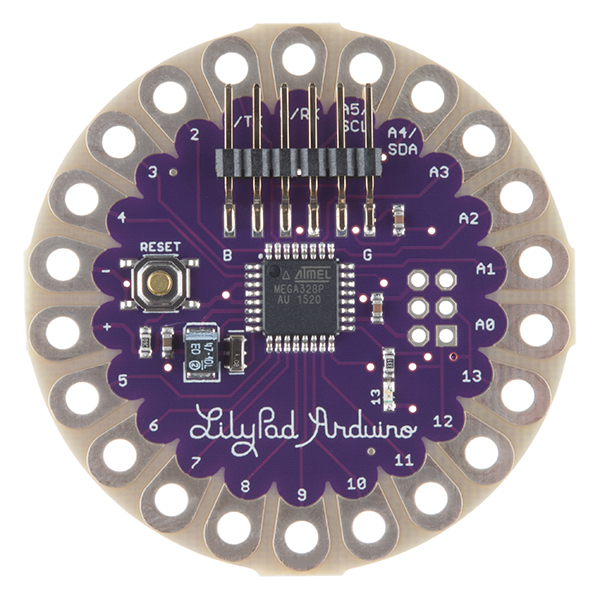
## 2.7 Vrste ploča

* Arduino Uno (R3) je najpopularnija proizvedena Arduino ploča, sastoji se od 14 digitalnih input/output pinova, a 6 od njih se mogu koristiti kao PWM (pulse width modulation) pinovi (za analogOutput), 6 analog inputa, reset dugme, te standardnu USB konekciju.



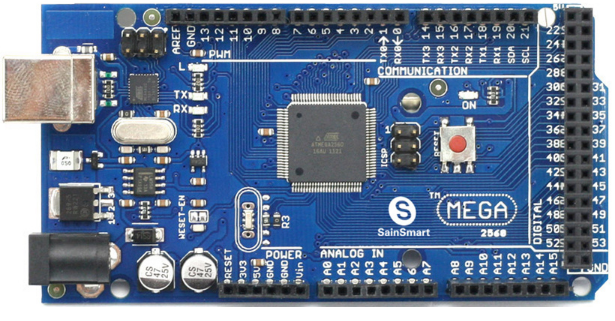
Slika 8: Arduino Uno (R3)

* LilyPad Arduino ploča je zanimljiva varijacija Arduino razvojne ploče, specifična je po tome što je dizajnirana za nošenje na odjevnim predmetima, te je vodootporna. Posjeduje 9 output pinova te 4 input pina. FTDI konektor za spajanje sa računarom, reset dugme, konektora za bateriju.



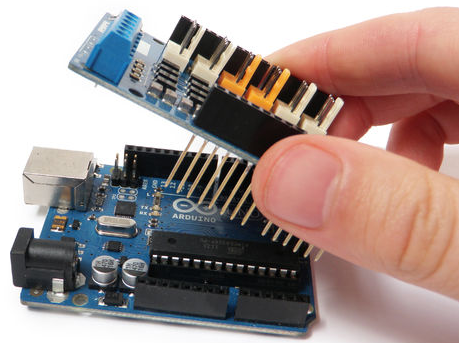
Slika 9: LilyPad Arduino

* Arduino Mega (R3) ima najviše sličnosti sa Uno pločom, te se smatra njenom proširenom verzijom. Posjeduje veliki broj input/output pinova, od kojih se 14 može koristiti kao PWM (pulse width modulation) pinovi. Ova vrsta ploče korištena je za praktični dokaz koncepta ovog rada.



Slika 10: Arduino Mega (R3)

* Arduino Shield su gotove ploče koje se koriste za spajanje na osnovne Arduino ploče. Najčešće na sebi sadrže razne module, te proširuju osnovne funkcionalnosti Arduino-a. Neki primjeri modula koje mogu sadržavati su moduli za povezivanje na internet, bežičnu komunikaciju, LCD monitori, Bluetooth moduli, MicroSD, GSM, itd.



Slika 11: Arduino Shield

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arduino ploča** | **Procesor** | **Memorija** | **Digital I/O** | **Analog I/O** |
| Arduino Uno | 16Mhz ATmega328 | 2KB SRAM, 32KB flash | 14 | 6 |
| LilyPad Arduino | 8Mhz ATmega 168/328V | 1KB SRAM, 16KB flash | 14 | 6 |
| Arduino Mega | 16MHz ATmega2560 | 8KB SRAM, 256KB flash | 54 | 16 |

Tabela 1: Speficikacije Arduino ploča

## 2.8 Pinovi

Zavisno od vrste Arduino ploče, raspored i broj pinova može biti različiti. Sadržani pinovi na osnovnom Arduino Uno okruženju su:

* Napon (pinovi vezani za napon Arduino ploče)
  + Vin – Koristi se za input voltaže kada koristimo eksterni napojni izvor
  + 5V – 5 voltni regulisani napon ploče i ostalih komponentni
  + 3.3V – 3.3 voltni regulisani napon ploče i ostalih komponenti
* Reset – pin koji služi za resetovanje mikrokontrolera, odnosno ponovni početak izvršenja učitanog programa.
* Analog pinovi (oznaka A0-A5) – Koristi se za pružanje analognog ulaza između 0-5V
* Input/Output pinovi (Digitalni pinovi 0-13) – Digitalni pinovi koji se mogu koristiti za ulaz i izlaz.
* Serijski 0(RX),1(TX) – Koriste se za serijsko čitanje/transimisiju podataka, te se ne mogu koristiti kao standardni digital pinovi.
* Eksterni prekidi – Pinovi 2 i 3, rezervisani za okidanje prekida.
* PWM (Pulse Width Modulation) – Standardni Arduino nema klasične analogne izlaze, već koristi metodu modulacije digitalnog pulsa kako bi izvršio simulaciju analognog signala. Pinovi 3,5,6,9,11 omogućavaju 8-bitni PWM izlaz.
* SPI – Pinovi 10 (SS – Slave Select), 11 (MOSI – Master out Slave in), 12 (MISO – Master in Slave out), 13 (SCK – Serial Clock) omogućavaju SPI (Serial Peripheral Interface) komunikaciju. Ova vrsta komunikacije je korištena od strane mikrokontrolera za komunikaciju sa jednim ili više perifernih uređaja ili za komunikaciju između 2 mikrokontrolera. Ono što odlikuje ovu vrstu komunikacije jeste veoma brz protok podataka.
  + Slave Select: Pin koji se može koristiti za omogućavanje/onemogućavanje specifičnih uređaja.
  + Master out Slave in: Koristi se za slanje podataka na periferijske uređaje.
  + Master in Slave out: Koristi se za protok podataka do Master (MOSI) pina.
  + Serial Clock: Sinkronizira clock brzinu prema onom generisanom od strane glavnog uređaja.
* Ugrađeni LED – Pin 13 preko kojeg je moguće kontrolisati ugrađeni LED pin.
* TWI – A4 (SDA), A5 (SCA) pinovi korišteni za komunikaciju sa TWI uređajima.
* AREF – Konfiguracija napona korištenog za analogni input.

# **Računarski vid**

Računarski vid je polje umjetne inteligencije koje obučava računare da interpretiraju i shvataju vizuelni svijet. Korištenje digitalnih slika za kamera, video-a i „deep-learning modela“, mašine mogu precizno identificirati i klasificirati objekte i onda reagovati na ono što oni „vide“.[[3]](#footnote-3)

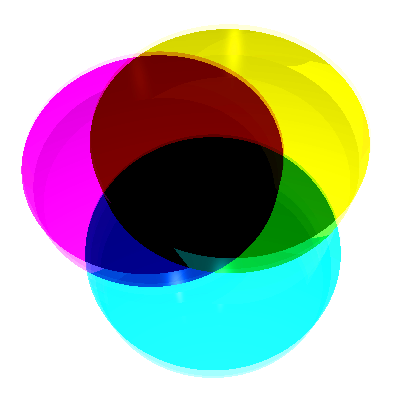
Problem računarskog vida se može čini jednostavnim, jer je vid nešto na što su ljudi navikli, čak i djeca i novorođenčad. Međutim računarski vid ostaje „neriješen problem“, zbog velike kompleksnosti našeg okruženja i našim ograničenim razumijevanjem biološkog vida, ali ipak zahvaljujući napretku u umjetnoj inteligenciji i inovacijama u neuronskim mrežama, ovo polje je znatno napredovalo u prethodnim godinama i uspjelo prevazići ljude u pojedinim zadacima detektovanja i obilježavanja objekata. Također, jedan od glavnih faktora u razvoju računarskog vida jeste velika količina podataka koje danas generišemo kao društvo, (npr. 46,760 Instagram slika po minuti[[4]](#footnote-4)) te se koristi za daljnja poboljšanja računarskog vida.

Prema predikcijama, hardversko i softversko tržište umjetne inteligencije će dostići $48.6 milijardi do 2022. godine[[5]](#footnote-5), s toga je jasno vidljivo da je računarski vid sve više rastuća oblast za koju definitivno možemo očekivati rast i nastavak inovacija u budućnosti.

## 3.1 Modeli boja

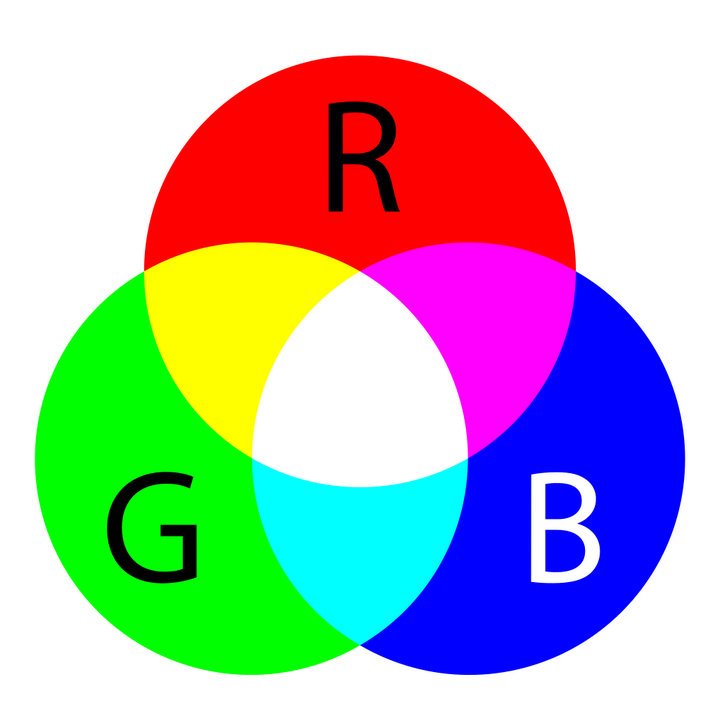
Jednostavno rečeno model boje je način za opis boje. Model opisuje kako će se boja prikazati na računarskom ekranu ili papiru[[6]](#footnote-6). Neophodno je poznavati više modela boja, kako bi što bolje razumjeli i upravljali računarskim vidom. U narednom dijelu su opisani pojedini modeli boja, od kojih su neki korišteni unutar aplikativnog dijela ovog rada.

* CMYK (Cyan, Magenta, Yellow, Black) – Ovaj model zasnovan je postotku prisustva cyan (tirkizne), magenta (varijante crvene), zute i crne. Jako često korišten je za printanje, jer je uz kombiniranje ove 4 boje moguće stvoriti iluziju velikog broja boja. Također ovaj model poznat je kao model oduzimanja – bijela boja je prirodna boja papira ili drugih podloga, a crna je kombinacija svih boja.



Slika 12: CMYK model

* RGB (Red, Green, Blue) – Jedan od najpoznatiji i najkorištenijih modela boje u kompjuterskoj grafici[[7]](#footnote-7). Vrijednosti između 0 i 255 su dodijeljene svakoj od boja (crvena, zelena i plava), te npr. za kreiranje bijele svaka od njih bi imala maksimalnu vrijednost 255. Ovaj model je suprotan prethodno opisanom CMYK modelu, jer je suprotno njemu poznat kao model dodavanja.



Slika 13: RGB model

* HSV/HSB – HSV je skraćenica za Hue, Saturation i Value, često je moguće vidjeti varijaciju HSB (Hue, Saturation, Brightness), a važno je znati da se u oba slučaja radi o istom modelu boja. HSV se veoma često predstavlja u cilindričnom obliku (Slika 14).

Hue je dio za boju u slučaju ovog modela, te se izražava kao broj od 0 do 360 stepeni:

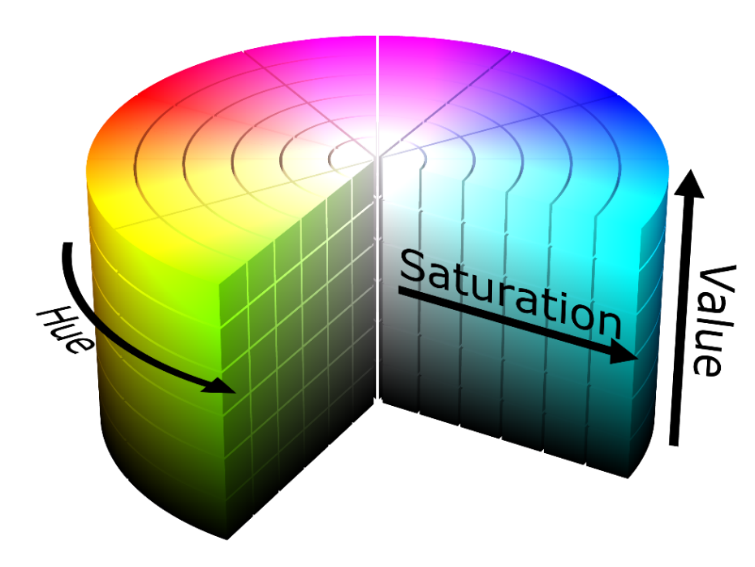
* + Crvena – između 0 i 60 stepeni.
  + Žuta – između 61 i 120 stepeni.
  + Zelena – između 121 i 180 stepeni.
  + Cyan (tirkizna) – između 181 i 240 stepeni.
  + Plava – između 241 i 300 stepeni.
  + Magenta (varijanta crvene) – između 301 i 360 stepeni.



Slika 14: Hue vrijednosti

Saturation opisuje količinu sive unutar određene boje, te se izražava od 0 do 100 procenata, približavanjem ove vrijednosti nuli proizvesti će rezultat izblijeđenosti i više sive boje. Ponekad se saturation prikazuje samo od 0-1, gdje je 0 siva, a 1 primarna boja.

Value/Brightness se koristi u kombinaciji sa saturation dijelom i definiše osvijetljenost ili intenzitet boje, gdje je 0 potpuno crna, a 100 najsvjetlija.



Slika 15: HSV/HSB Model

* HSL (Hue, Saturation Lightness) – Hue se ponaša jednako prethodnom modelu, te ovaj model glavnu razliku pronalazi u saturation i lightness dijelu. Saturation dio se uvijek kreće od potpuno zasićene do ekvivalntne sive (u HSV sa maksimalnim Value, kreće se od zasićene prema bijeloj). Lightness za razliku od value unutar HSV modela postiže punu osvijetljenost boje na 50%, te se nakon toga približava bijeloj sve do 100% kada to postaje u potpunosti (u HSV modelu ide samo do pola, gdje postiže svoju maksimalnu osvijetljenost).



Slika 16: HSL Model

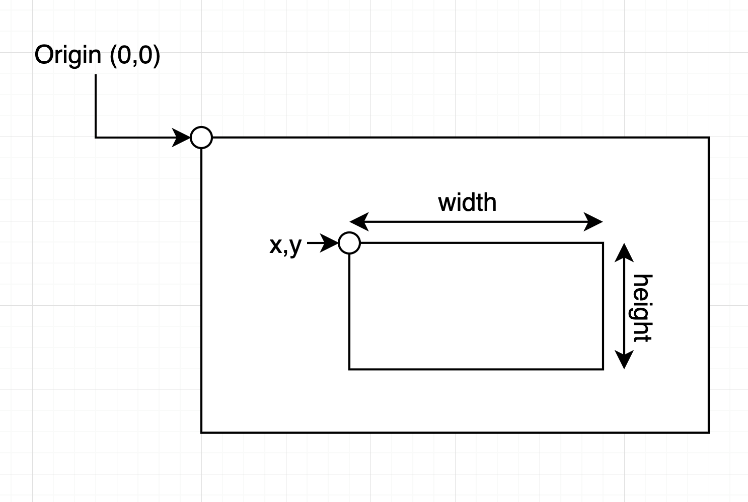
## 3.2 OpenCV Biblioteka

OpenCV je open-source biblioteka koja uključuje nekoliko stotina algoritama za računarski vid (computer vision).[[8]](#footnote-8) Unutar ovog projekta korištena je verzija OpenCV 3, koja je zapravo OpenCV C++ verzija. OpenCV ima strukturu podijeljenu na module, a neki od modula sadržani su sljedeći:

* Jezgrine funkcionalnosti – modul koji definiše osnovne strukture podataka, kao što su višestruko dimenzionalne matrice Mat i osnovne funkcije korištene u ostalim modulima.
* Obrada slike – modul za obradu slika koji uključuje linearno i ne-linearno filtriranje slika, te geometrijske transformacije slike, konverzija modela boja, histograme, itd.
* Video – modul koji uključuje procjenu kretanja, oduzimanje pozadine i algoritme za praćenje objekata.
* objdetect – detekcija objekata i instanci predefinisanih klasa, npr. oči, lica, automobili, auta, itd.
* highgui – interfejs za jednostavne UI mogućnosti.

## 3.2.1 Osnovne klase, funkcije i operatori

* VideoCapture – VideoCapture je klasa za preuzimanje videa sa video fajla, kamera ili sekvenci slika. Kao parametar uzima lokaciju fajla ili integer broj kamere.
* Mat – Klasa n-dimenzionalno gustih matrica. Može se koristiti za pohranu realnih ili kompleksnih vrijednosti, crno bijelih ili slika u boji, vektor polja, histograma, itd.
* >> – Stream operator za čitanje sljedećeg video frejma.
* isOpened() – Funkcija koja provjerava da li je već inicijalizirana klasa VideoCapture.
* Rect – Klasa pomoću kojeg je moguće kreirati kutiju pomoću koordinata koji se prosljeđuju kao parametri. Važno je napomenuti da konstruktor klase Rect, uzima integer vrijednosti, koje obično predstavljaju koordinate, iz gornjeg lijevog ugla ka donjem lijevom, te je najčešći uzrok zabune prilikom kreacije objekata ove klase. Parametri su x,y, sirina (widh) i visina (height).

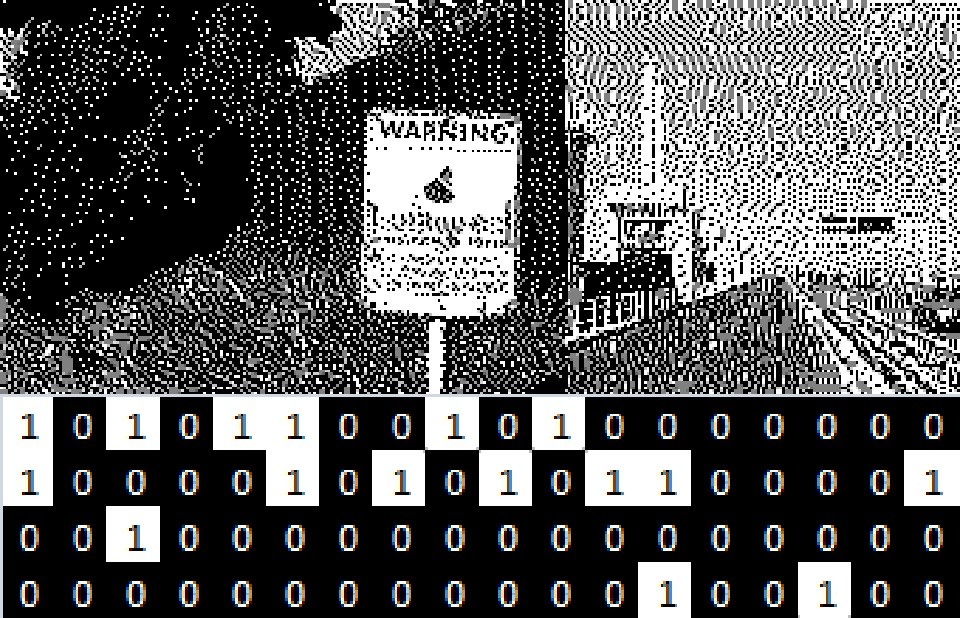


Slika 17: Rectangle konstrukcija

* Mat () – Operator koji kao parametar uzima rectangle, te na osnovu postojećeg Mat-a vraća novi definisan unutar Rectangle granica u postojećem Mat-u.
* Scalar – Klasa za 4 elementni vektor izveden iz Vec klase. Unutar ovog proejkta Scalar je najčešće korišten za definisanje boja, kako je za to veoma pogodan.
* resize() – Funkcija za promjenu veličine slike, za paramater uzima inputArray, outputArray te veličinu.
* cvtColor() – Funkcija za konverziju modela boja, za parametar uzima inputArray, OutputArray, te enumeraciju željene konverzije.
* line() – Funkcija za iscrtavanje linija od početnih koordinata, do završnih koordinata

## 3.3 Binarne slike

Binarne slike su slike koje se sastoje samo od dvije boje, najčešće od crne i bijele, često se nazivaju i „bi-level“ ili „two-level“. Ovo znači da je svaki piksel slike pohranjen kao jedan bit odnosno 1 ili 0. Binarne slike se pojavljuju na razne načine. Najčešće su kreirane od sivih slika za pojednostavljeno procesiranje ili za printanje[[9]](#footnote-9). Binarna slika se može pohraniti u memoriji kao bitmap, niz bitova, te zbog svoje male veličine koriste ga fax mašine i rješenja za menadžment dokumenata.



Slika 18: Binarna slika

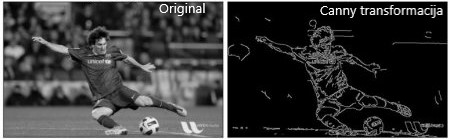
## 3.4 Canny algoritam

Canny algoritam poznatiji i kao canny detektor ivica, jeste algoritam za prepoznavanje različitih vrsta ivica na slikama.

Canny se sadrži od sljedećih koraka:

1. Predprocesiranje – filtriranje šuma na slici
2. Izračun nagiba slike
3. Suzbijanje nemaksimalnih vrijednosti – izbacuje piksele koji se ne smatraju dijelom ivice, samo najbolji kandidati ostaju nakon ovog koraka
4. Podešavanje sa histerozom – minimum i maksimum

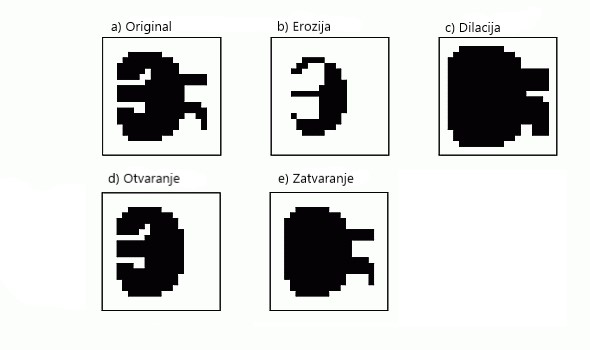
Canny ima tri podesiva parametra: širina Gaussian-a (Gaussian smanjuje efekat šuma prisutan u slici, što veći šum, veća širina), te niski i visoki prag za histerezu. Histereza je tendencija materijala da zadrži jedno od svojih svojstava u odsustvu ili kašnjenju poticaja koji ga je stvorio[[10]](#footnote-10), npr. kod gume je pristuna elastična histereza, jer se guma sporije skuplja kada je smanjena sila koja je izazvala rastezanje. Unutar Canny-a, ukoliko je nagib piksela veći od maksimalnog praga, piksel se prihvata kao ivica, ako je manji od minimalnog praga odbacuje se, a ako je između minimalnog i maksimalnog praga, prihvata se samo ako je povezan sa prihvaćenim pikselom koji je veći od maksimalnog praga. Canny preporučuje postavljanje odnosa između višeg i nižeg na 2:1 ili 3:1[[11]](#footnote-11).



Slika 19: Canny transformacija

## 3.5 Morfološke transformacije

Morfološke transformacije su jednostavne operacije bazirane na obliku slike.[[12]](#footnote-12) Obično se vrše na binarnim slikama. Potrebna su nam dva ulaza, prvi je originalna slika, te druga poznatija kao strukturalni element ili kernel koji odlučuje vrstu operacije. Najpoznatije vrste morfoloških operacija su erozija i dilacija, te također postoje varijante kao što su otvaranje, zatvaranje, gradient i sl. Zatvaranje je vrsta morfološke transformacije korištenja u glavnoj aplikaciji za računarski vid.



Slika 20: Morfološke transformacije

### Erozija

Osnovna ideja erozije jeste kao i sama erozija tla, briše granice prednjeg objekta (treba nastojati uvijek da boja objekta ispred bude bijela). Kernel prelazi preko slike, kao u 2D savijanju. Piksel na originalnoj slici će se prihvatiti (1) samo ako su svi pikseli ispod kernela 1, u suprotnom su serodirani odnosno postavljeni na 0. Rezultat je to da su svi pikseli koji su blizu granice odbačeni, zavisno od veličine kernela, erozija je korisna za uklanjanje malih bijelih šumova (Slika 19). Unutar OpenCV biblioteke dostupna je cv.erode funkcija za postizanje rezultata erozije na slici.

## Dilacija

Dilacija je suprotnost eroziji, ovdje piksel je 1 ukoliko je barem jedan piksel ispod kernela 1. Povećava bijeli dio slike, te u slučajevima kao što je uklanjanje šuma obično je erozija praćena dilacijom iz razloga što erozija uklanja bijeli šum, ali također smanjuje naš objekat, tako da onda vršimo dilaciju. Unutar OpenCV biblioteke dostupna je cv.dilate funkcija za postizanje rezultata dilacije na slici.

## Otvaranje

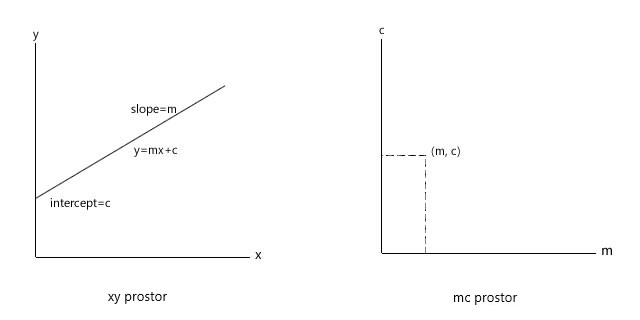
Otvaranje je samo drugi naziv za eroziju praćenu dilacijom. Koristan je za uklanjanje šumova. Otvaranje uklanja male objekte iz prednjeg plana slike, te ih stavlja u pozadinu. Obično se koristi za pronalazak specifičnih oblika na slici. Unutar openCV biblioteke dostupna je cv.morphologyEx funkcija za postizanje rezultata otvaranja na slici.

## Zatvaranje

Zatvaranje je suprotnost otvaranju, odnosno dilaciji praćenoj erozijom. Korisno je za zatvaranje malih rupa unutar prednjeg plana slike, ili malih crnih tački na objektu.

## Hough transformacija

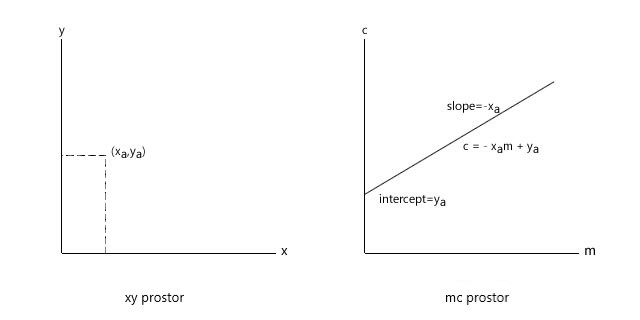
Hough transformacija je popularna tehnika za detekciju bilo kojih oblika, ako je oblik moguće predstaviti u matematičkoj formi. Može detektovati bilo koji oblik čak i kad je pokvaren ili izlomljen. Unutar ovog projekta detektujemo linije, tako da ćemo hough transformaciju objasniti na primjeru linija. Linija je kolekcija tački, ali rad sa kolekcijom tački je teži nego rad sa samo jednom tačkom, zato je važno prikazati liniju kao jednu tačku, ovo se radi kroz m-c prostor.



Slika 21: m-c prostor

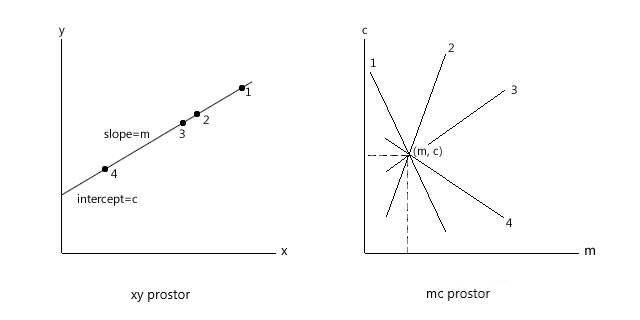
Kao što je prikazano na Slici 21, svaka linija ima dva parametra asocirana sa sobom, nagib (slope) i presjek (intercept). Sa ova dva broja, moguće je potpuno opisati liniju. Dakle osmišljeni su parametri potrebni za mc prostor. Svaka linija u xy prostoru ekvivalentna je jednoj tački u mc prostoru.

Posmatrajmo zamišljenu tačku (xa, ya), za svaku liniju koja prolazi kroz (xa, ya) postoji tačka u mc prostoru. To je tačno zbog sljedećeg: Svaka linija koja prolazi kroz (xa, ya): ya = mxa + c, dalje slijedi **c = - xam + ya**, te dobijamo jednačinu linije u mc prostoru.



Slika 22: Konverzija u mc prostor

Hough transformacija je u biti konverzija tački u xy prostoru u linije u mcd prostoru. Uzimamo ivicu, te za svaku tačku koja nije crna, crtamo linije u mcd prostoru. Očigledno je da će se ne neke linije presjeći, te ti presjeci označavaju parametre linije.



Slika 23: Primjer hough transformacije

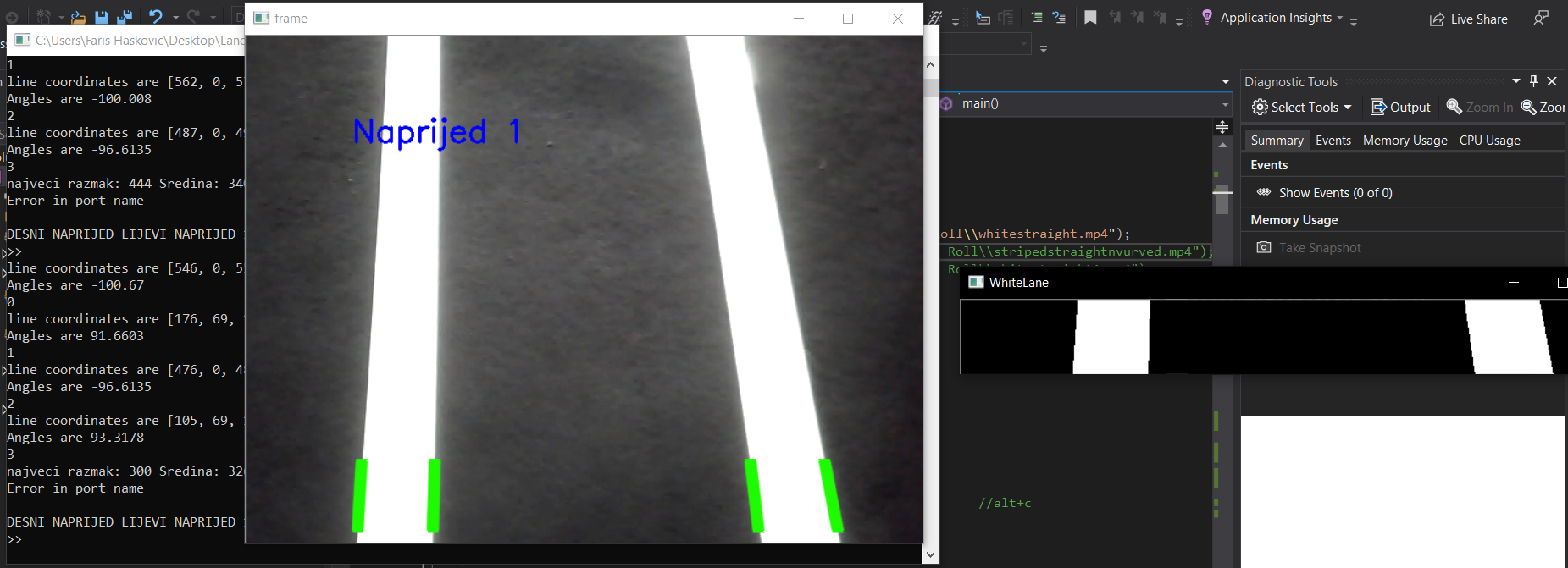
Tačke 1, 2, 3, i 4 su reprezentacije različitih linija u mc prostoru. Presjeci tih linija su ekvivalent originalnoj liniji.

# **Aplikacija za računarski vid**

U ovom dijelu će biti opisana C++ OpenCV aplikacija, koja se koristi za upravljanje robotom i donošenje gotovo svih odluka o kretanju.

## 4.1 Dizajn

Dizajn prikaza, sačinjen je od 3 konzolna prozora. Glavni prozor nazvan „frame“ pokazuje trenutni prikaz kamere, detekciju linija, te tekstualni prikaz akcija koje izvršava odnosno šalje Arduino-u na izvršenje.

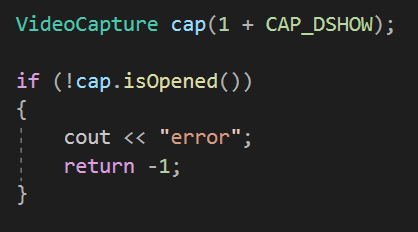


Slika 24: Dizajn izlaza aplikacije

## 4.2 Kod

Jezik korišten unutar izrade ove aplikacije jeste C++, te je korišteno okruženje Visual Studio 2019. Sva implementacija sadržana je unutar main funkcije, kako nije potrebno detaljnije rastavljanje.

Prvenstveno je neophodno kreirati VideoCapture, te provjeriti da li je pravilno inicijaliziran. U programu su parametri postavljeni na 1, kako se unutar projekta koristi sekundarna web kamera, te enumeracijski parametar CAP\_DSHOW, koji govori da input sa kamere prikazuje direktno preko videoInput-a, ovaj parametar kroišten je najviše u procesu debuginga, kako je obični nedirektni prikaz izazivao probleme tokom prikaza sadržaja web kamere. Nakon inicijalizacije, koristimo funkciju isOpened koja vraća true ukoliko je video capturing već inicijaliziran, kako bi ukazali na potencijalnu grešku.



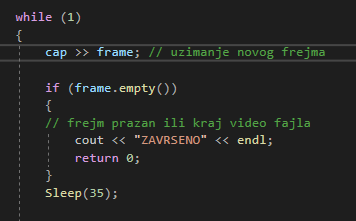
Programski kod 1: VideoCapture

Nakon implementacije VideoCapture klase, definisane su tri Mat klase, frame klasa je glavni Mat za pohranu frejmova sa kamere, whiteLane sadržava detektovane bijele linije, dok će HSV\_Img sadržavati konverziju rectangle dijela videa u HSV prostor boja, radi poređenja da li pripada predodređenom rasponu.



Programski kod 2: Mat klase

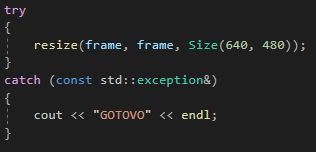
Sva logika programa pokreće se unutar beskonačne petlje, kako se u stvarnom okruženju zahtijeva neprekidno izvršenje programa.



Programski kod 3: Preuzimanje frejma

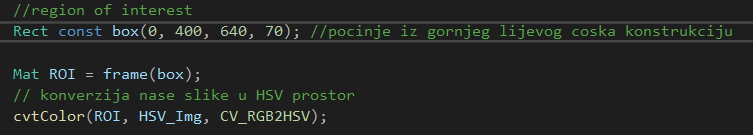
Prvenstveno pokušavamo uzeti novi frejm sa kamere pomoću preklopljenog operatora >> te proslijediti ga u frame Mat. Nakon toga vršimo provjeru da li je uspješno preuzet, odnosno ukoliko je riječ o video fajlu, da li je došao kraj video fajla, kako ne bi došlo do nepredviđenih grešaka.

Sleep() funkcija označava pauziranje izvršenja programa u ovom slučaju na 35 milisekundi, prije nastavka daljnjeg izvršenja, prvenstveno u svrhu sinhroniranja brzina rada sa Arduino okruženjem, kako bi bez ovog dijela koda program slao podatke brže nego što serijalnom vezom Arduino može primiti.



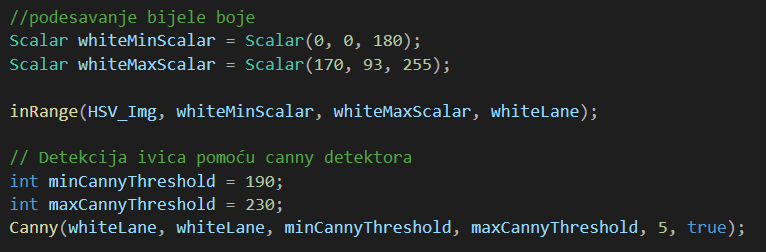
Programski kod 4: Promjena veličine frejma

Neophodno je promijeniti veličinu frejma, kako bi odgovarala našim potrebama. Ovo je izvršeno korišteći resize funkciju, koja prima input i outputArray (Mat), te veličinu koju želimo postići. Također, moguće je da u ovom dijelu koda naiđemo na određene greške, te je ovaj dio koda također obuhvaćen try/catch blokom, kako bi izbjegli neočekivane prekide izvršenja.



Programski kod 5: Rect Konstrukcija

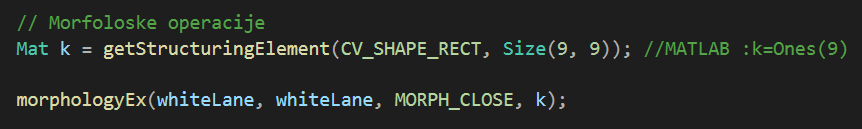
Unutar ovog dijela koda (5) kreiran je objekat klase Rect koji obuhvata donji dio frejma, a razlog jeste neophodnost uzimanja linija koje se nalaze bliže točkovima robota, kako bi se preciznije odredilo kretanje točkova istog. Nakon toga, kreiran je novi Mat koji sadrži izrezanu sliku frejma u granicama prethodno definicanog Rect-a. Na kraju pomoću funkcije cvtColor novi Mat prebacujemo u HSV model boja, te ga smiještamo u HSV\_Img Mat radi daljnjeg poređenja.



Programski kod 6: Canny konstrukcija

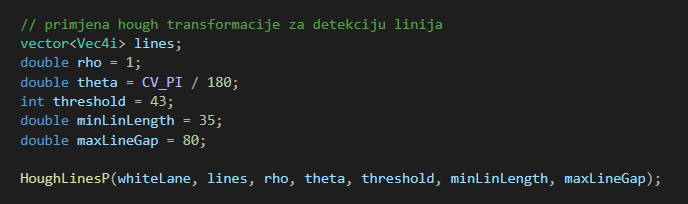
Sljedeći koraci su kreiranje 2 Scalara, koji čuvaju minimalne i maksimalne vrijednosti bijele boje unutar HSV modela boja, te se nakon toga provjerava da li postoje elementi koji pripadaju tom opsegu, te se smještaju unutar whiteLane Mat-a.

Konstrukcija Canny objekta zahtijeva inputArray, outputArray za smještanje ivica, minimalni i maksimalni prag za prihvatanje i odbacivanje ivica, aperture veličinu koja pokazuje veličinu Sobel jezgre. Sobel jezgro se može zamisliti kao operator koji se kreće u X i Y smjeru, te identificira ivice i orijentaciju. Posljednji parametar jeste boolean, koji se odnosi na preciznost, odnosno da li će se koristiti kompleksnija i sporija (true) ili jednostavnija i brža formula (false).



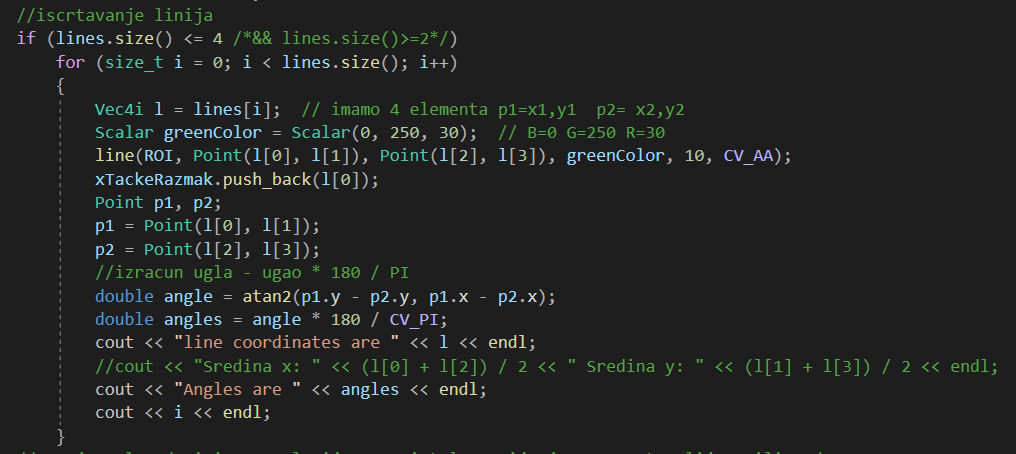
Programski kod 7: Morfoloske operacije

Sljedeće na redu jesu morfološke operacije (Programski kod 7). Prvenstveno unutar novog Mat-a pohranjujemo strukturalni element oblika kvadrata veličine 9,9, te primjenjujemo morfološku operaciju zatvaranja (4.5.4) radi uklanjanja crnih šumova oko slike (kako tražimo bijele linije), te kao kernel postavljamo prethodno kreirani strukturalni element k.



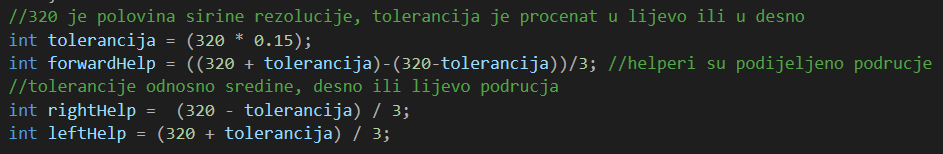
Programski kod 8: Hough transformacija za detekciju linija

Primjena hough transformacije neophodna je za detekciju linija kao sljedeći korak. Detektovanje linije ćemo pohraniti u vektor objekata Vec4i, koji sadrži četri elementi, početak linije (x1,y1) i posljednja dva koja predstavljaju završetak linije (x2, y2). Koristeći HoughLinesP funkciju, moguće je izvršiti hough transformaciju (4.6), te pronaći linije sa frejma whiteLane, te konačno pohraniti ih u lines vektor.



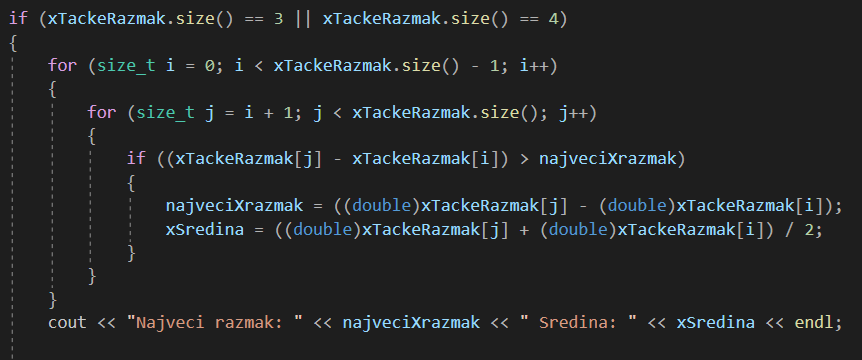
Programski kod 9: Iscrtavanje linija

Naredni korak jeste iscrtavanje linija, želimo se osigurati da u datom trenutku prepoznajemo maksimalno 4 linije, te ih pohranjujemo unutar vektora Vec4i, koji čuva podatke o liniji (početne koordinate i završne koodinate), nakon čega ih iscrtavamo zelenom bojom, debljine 10, sa frakcijskim razmakom bitova u liniji definisanih enumeracijom CV\_AA (16). Zatim je potrebno pohraniti početnu x koordinatu linije, radi daljnjeg poređenja koji će biti objašnjen u sljedećim dijelovima rada. Izračun i ispis uglova, te ispis linija kordinata vrše se radi eventualnih potreba debugginga.



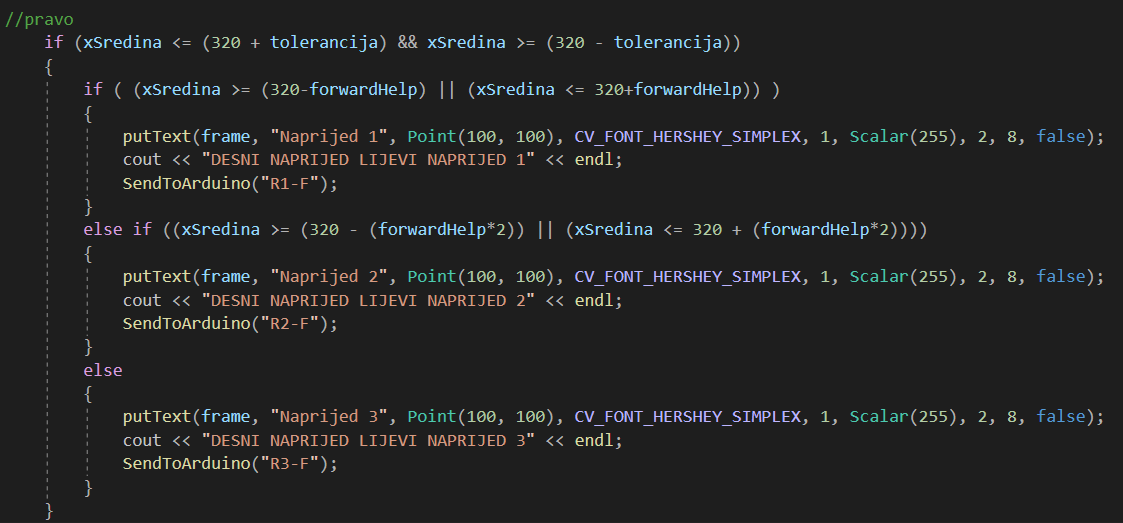
Programski kod 10: Tolerancija

Tolerancija se izračunava u svrhu kretanja robota, a u ovom slučaju tolerancija je postavljena na 15%, odnosno ukoliko se kreće naprijed, robotu će biti dozvoljeno odstupiti 15% od sredine frejma, te dijelimo lijevu i desnu stranu na 3 dijela, kako bi podijelili vrijeme kretanje robota desno ili lijevo u zavisnosti od potreba.



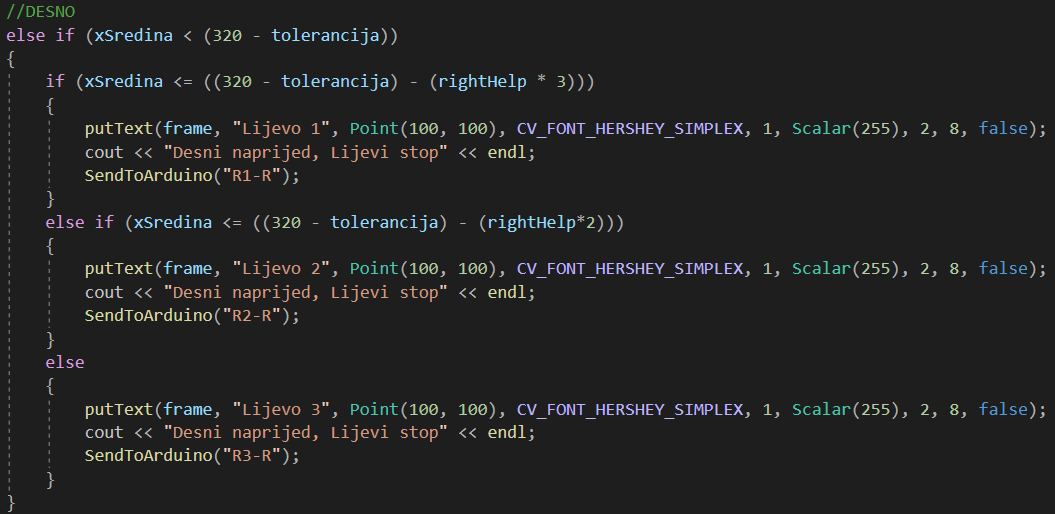
Programski kod 11: Najveci razmak

Naredni korak jeste traženje najvećeg razmaka među linijama, te to činimo samo ukoliko su trenutno pronađene 3 ili 4 ivice linije, sve ostale slučajeve odbacujemo, kako je moguće da u slučaju jedne ili dvije ivice u biti pronađena samo 1 linija, te izračunavamo sredinu između ove dvije ivice po x osi.



Programski kod 12: Kretanje pravo

Kretanje naprijed vrši se samo u slučaju da je sredina po x osi između granica tolerancije sredine. Sjetimo se da je tolerancija podijeljena na 3 dijela, ukoliko je sredina među granicama prvog dijela tolerancije, ispisujemo tekst u glavni frame o kretanju „Naprijed 1“, te Arduino-u serijalno šaljemo komandu „R1-F“ (funkcija za slanje će biti objašnjena u narednim dijelovima). Ukoliko se sredina nalazi među granicama drugog dijela tolerancija, ispisujemo u frejm „Naprijed 2“, te Arduino-u šaljemo komadu „R2-F“, ukoliko ni jedno nije slučaj, znači da se sredina nalazi uz posljednju granicu tolerancije, te to ispisujemo u frejm, te šaljemo komandu „R3-F“.



Programski kod 13: Kretanje desno

Ukoliko se sredina nalazi van granica tolerancije sredine u lijevo, odnosno ukoliko je manja od granica tolerancije, neophodno je izvršiti skretanje u desno. Ukoliko je sredina manja ili jednaka od granice posljednjeg dijela tolerancije, šaljemo komandu Arduino-u „R1-R“, te ispisujemo prikaz unutar konzole. Ukoliko je sredina manja ili jednaka od drugog dijela granice, šaljemo komandu „R2-R“, te u posljednjem slučaju komandu „R3-R“.



Programski kod 14: Kretanje u lijevo

Kretanje ulijevo prati isti principe kao i prethodni slučaj. Ukoliko je sredina veća od granica kretanja naprijed vršit će se kretanje ulijevo, kako bi se robot ponovo usmjerio prema sredini. Ukoliko je sredina veća ili jednaka od posljednjeg dijela granice, šaljemo i ispisujemo komandu „R1-L“, ukoliko je veća ili jednaka od drugog dijela granice, komanda je „R2-L“, te u posljednjem slučaju kada je sredina blizu granice sredine, šaljemo komandu „R3-L“ kako bi se izvršilo minimalno pomijeranje prema sredini.

### 4.2.1 Serijsko slanje

Serijalno slanje podataka izvršeno je uz pomoć SerialPort klase, čiji kod je vidljiv na primjeru (Programski kod 15). Definisano je maksimalno čekanje na odgovor Arduino-a, te maksimalna veličina prijenosa podataka. Konstruktor SerialPort zahtijeva naziv porta na kojem se vrši komunikacija, te na kraju funkcije slanja i primanja podataka zahtijevaju buffer, te veličinu buffera.



Programski kod 15: Serial header

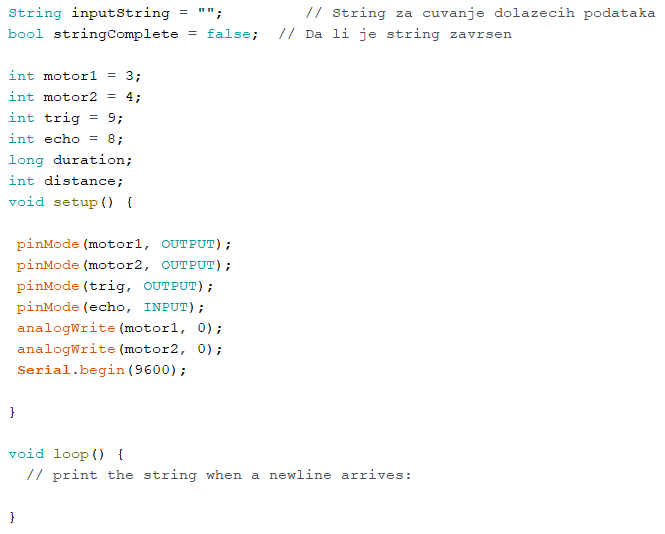
U narednom dijelu (Programski kod 15) prikazana je funkcija za serijalno slanje podataka u Arduino okruženje, funkcija prima string za slanje podataka koji se konvertuje u niz karaktera, jer je kao tip zahtijevan od klase SerialPort. Naredni korak jeste korištenje funkcije slanaj podataka writeSerialPort ili readSerialPort, ukoliko su podaci poslani od strane Arduino-a, te se ispisuju u konzolu. Naposljetku se vrši dealokacija zauzete memorije.



Programski kod 16: Serijalno slanje podataka

# **5.0 Arduino aplikacija**

Primarna svrha Arduino aplikacije jeste izvršenje kontrola za kretanje robota, te upravljanje ultrazvučnim senzorom. U narednom dijelu je opisan programski kod.



Programski kod 17: Arduino setup

Početak uključuje deklaraciju dvije varijable inputString za čuvanje dolaznih podataka, te boolean varijablu koja pokazuje da li je prijenos podataka u potpunosti izvršen. Naredne varijable čuvaju podatke o broju pina koji je korišten od strane motora. Trig i echo varijable čuvaju broj pin-a korišten od strane ultrazvučnog senzora za detekciju kolizije (7.1 HC-SR04 ultrazvučni modul).

Setup funkcija se izvršava samo jednom, te podrazumijeva osnovne postavke uključujući postavljanje pinova motora i trig pina kao izlazne, te echo pin kao ulazni.

Početno stanje motora je ugašeno (bez napona), te analogWrite funkcijom šaljemo vrijednosti 0, kako bi to postigli. Na kraju je neophodno naznačiti početak serijalne komunikacije sa 9600 baud rate-om.

Loop funkcija Arduino programa je prazna, jer se sva logika izvršenja odvija unutar serialEvent fukncije, koja je objašenjena u narednom dijelu.

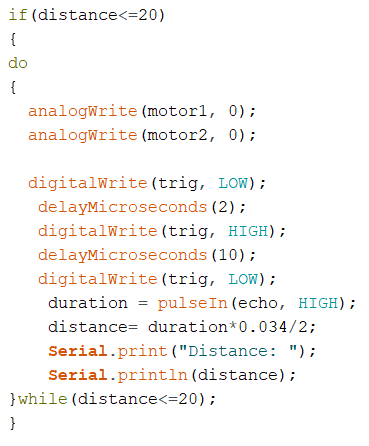


Programski kod 18: Serial event

Funkcija serialEvent se izvršava kada novi podaci stignu u Arduino, odnosno na RX pin. Također, važno je napomenuti da se izvršava između svakog prelaska loop funkcije, tako da je dostupno usporavanje izvršenja postavljajući kašenjenja unutar loop-a.

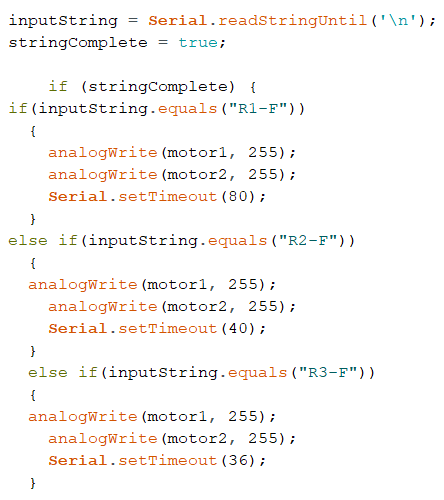
Ukoliko nema serijske komunikacije, motori će biti u stanju mirovanja, dok u slučaju da je serijalna komunikacija aktivna, odnosno podaci dolaze na RX pin, nastavlja se neophodno izvršenje funkcije.

Prvi dio ove funkcije odnosi se na kontrolu ultrazvučnog senzora HC-SR04. Slanje zvučnog signala je u početku isključeno, te se nakon toga mijenja odnosno šalje zvučni signal na 10 mikro sekundi. Tokom 10 mikrosekundi, zvučni senzor će emitovati 8 zvučnih signala, koji se primaju od strane echo pina. Nakon što primi zvučni signal, unutar varijable duration čuvamo vrijeme putovanja talasa. Međutim, važno je napomenuti da se tu čuva vrijeme koje je potrebno za zvučni signal da otputuje i vrati se do senzora, a kako je u ovom slučaju potrebna samo udaljenost objekta, to vrijeme ćemo podijeliti sa 2 i pomnožiti sa 0.034, kako zvuk prelazi 0.034 cm/µs. Nakon što dobijemo udaljenost izraženu u centimetrima, možemo nastaviti sa daljnom logikom izvršenja programa.



Programski kod 19: Ultrazvučna provjera

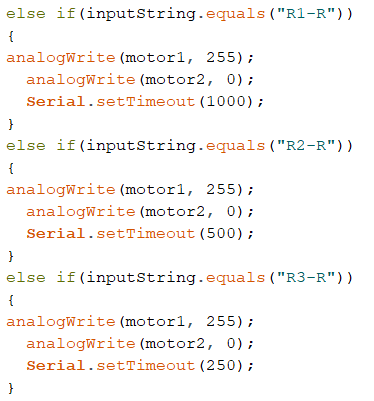
Ukoliko je unutar početne provjere utvrđeno da se objekat nalazi ispred robota u udaljenosti manjoj ili jednakoj od 20cm, zaustavlja se kretanje motora, te se ponovo emituju ultrazvučni signali i vrši provjera udaljenosti sve dok se prepreka ne skloni, odnosno postane udaljena manje od 20cm.



Programski kod 20: Kretanje naprijed

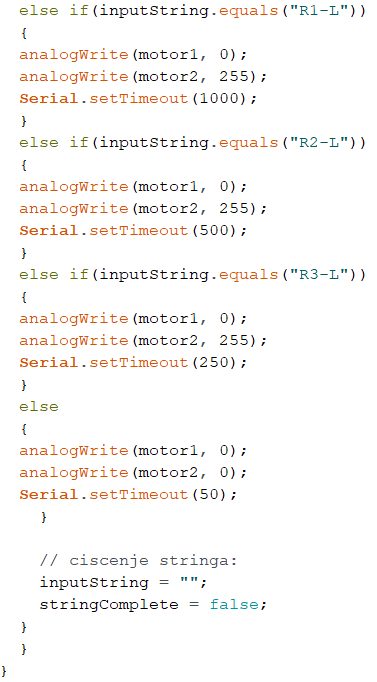
Naredni korak jeste primanje podataka poslanih preko serijalnog porta iz CV (computer vision) aplikacije. Ukoliko se pronađe kraj linije specificiran novom linijom, slanje poruke se smatra završenim, te u tom slučaju se nastavlja izvršenje. Ukoliko je primljena poruka „R1-F“, oba motora se kreću naprijed punom snagom, te se postavlja kašnjenje od 80 milisekundi, kako bi se odgodilo primanje novog frejma. Razlike u ostalim komandama za kretanje naprijed sastoje se od razlike u kašnjenju, odnosno što je robot bliže granicama tolerancije za kretanje naprijed, manje vremena će biti omogućeno za izvršenje kretanja prema naprijed kako bi se spriječilo veće prekoračenje granica.

U slučaju da nije primljena komanda za kretanje naprijed, vrši se provjera za postojanje komande za pokretanje desnog motora (kretanje u lijevo). Ukoliko je zaprimljena komanda „R1-R“, obustavlja se kretanje lijevog motora, te se kretanje desnog motora postavlja na maksimum kako bi postigli efekat okretanja, također u ovom slučaju kreira se pauza od 1000ms, jer se robot nalazi uz samu ivicu frejma, ostali slučajevi „R2-R“ i „R3-R“ razlikuju se u dužini pauziranja komunikacije, kako se nalaze bliže granicama sredine (Programski kod 18).



Programski kod 21: Kretanje u lijevo

Ukoliko nije zaprimljena komanda za kretanje naprijed niti za kretanje u lijevo, vrši se provjera da li je zaprimljena komanda za kretanje u desno. Ukoliko je zaprimljena komanda „R1-L“, obustavlja se kretanje desnog motora, te se kretanje lijevog motora postavlja na maksimum kako bi postigli efekat okretanja. Također u ovom slučaju kreira se pauza od 1000ms, jer se robot nalazi uz samu ivicu frejma, ostali slučajevi „R2-L“ i „R3-L“ razlikuju se u dužini pauziranja komunikacije, kako se nalaze bliže granicama sredine (Programski kod 19). Ukoliko nije pronađena ni jedna od definisanih komandi, jasno je da je došlo do pogreške u komunikaciji, te je zaprimljena strana poruka sa greškom i potpuno se obustavlja kretanje motora, sve do primanja sljedeće ispravne komande.



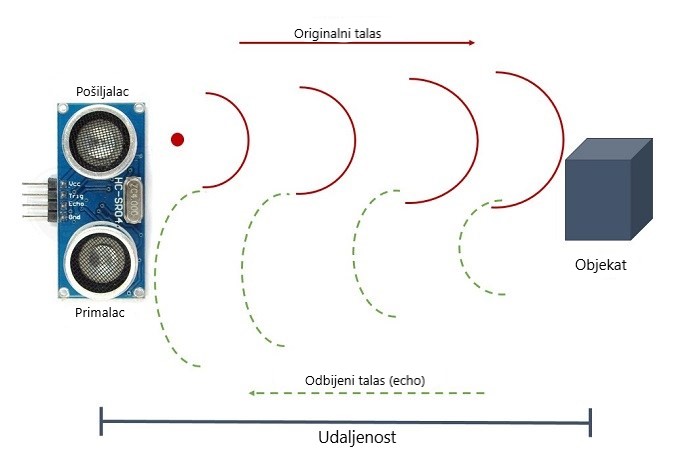
Programski kod 22: Kretanje u desno

# **6.0 Robot**

U sljedećem dijelu bit će opisani hardverski dijelovi koji čine samu platformu za dokaz koncepta aplikacije.

## 6.1 HC-SR04 ultrazvučni modul

HC-SR04 modul je ultrazvučni senzor u ovom projektu korišten za prevenciju kolizije. Emituje frekvenciju od 40,000Hz (40kHz), putuje kroz zrak, te ukoliko je objekat ili prepreka na putu, odbija se nazad u modul. Uzimajući u obzir vrijeme putovanja zvuka moguće je izračunati udaljenost.



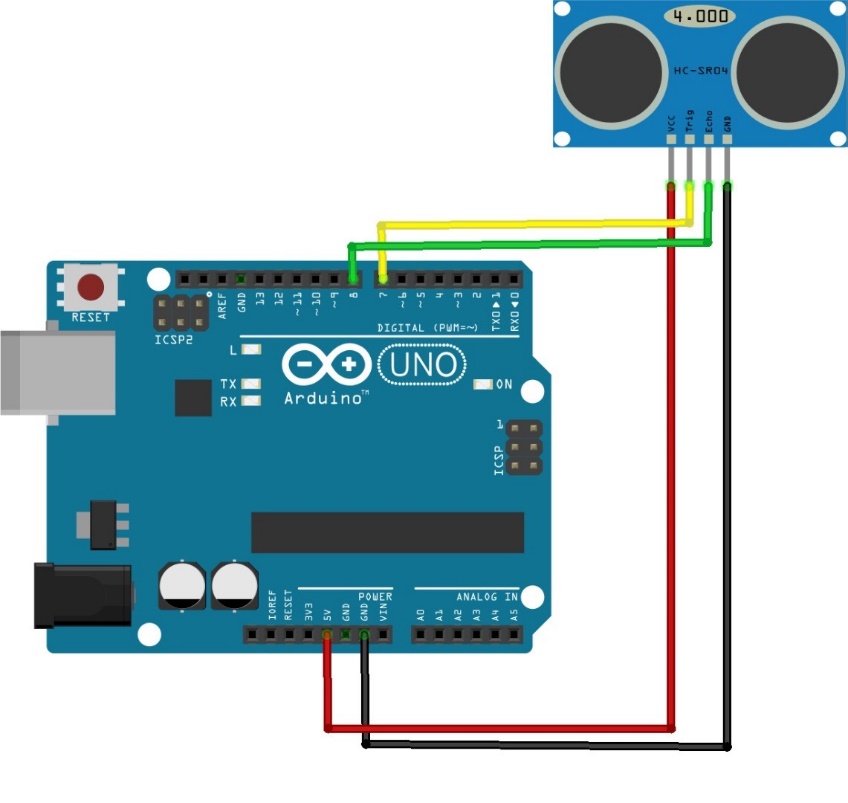
Slika 25: Način komunikacije modula

HC-SR04 ultrazvučni modul ima 4 pina. Uzemljenje (Gnd), VCC, Trig i Echo.



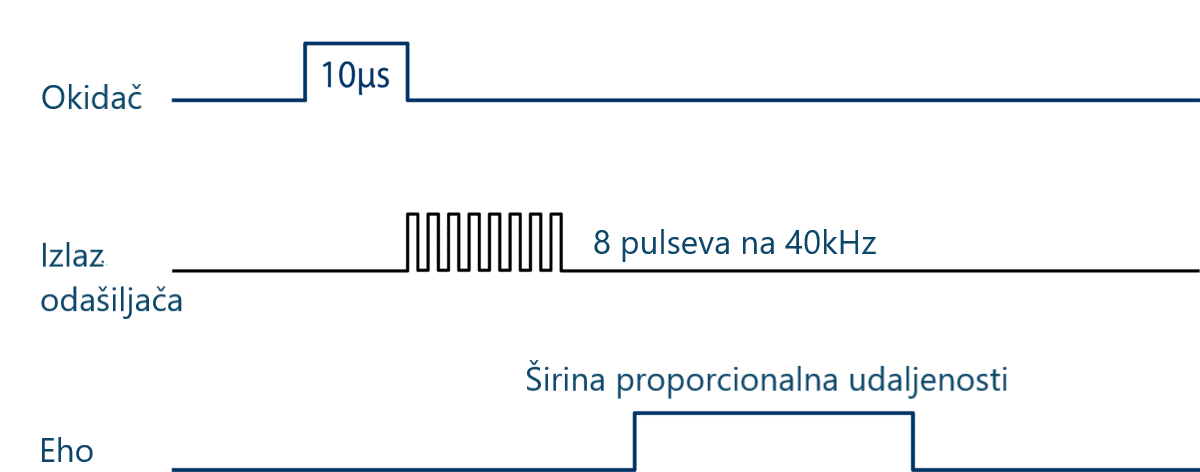
Slika 26: HC-SR04 modul

Uzemljenje se povezuje na Gnd pin, dok se VCC pin povezuje na 5 voltni pin na Arduino razvojnoj ploči (Slika 28).



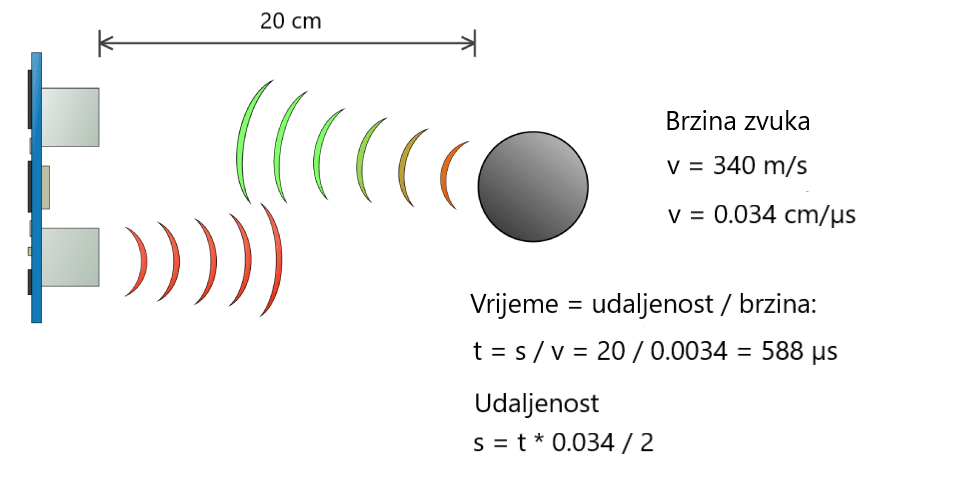
Slika 27: Šema povezivanja senzora

Da bi generisali ultrazvuk, moramo postaviti podesiti stanje Trig pina na upaljeno 10 mikrosekundi. To će poslati 8 „udara“ ultrazvuka koji putuje brzinom zvuka i bit će primljen od Echo pina. Echo pin će vratiti vrijeme putovanja u mikrosekundama koje je zvuk proveo putujući (Slika 27).



Slika 28: Slanje signala

Na primjer, ako je objekat udaljen 20 cm, računajući brzinu zvuka od 340 metara u sekundi, odnosno 0.034 cm po mikrosekundi, zvuk će putovati oko 588 mikro sekundi, ali vrijednost koju Echo pin vrati je vrijeme potrebno za talas da otputuje i vrati se, tako da ćemo tu vrijednost podijeliti sa 2, kako bi dobili samo pola putovanja, odnosno samu udaljenost objekta. Dakle, za dobivanje udaljenosti u centimetrima množimo vrijeme putovanja zvuka sa 0.034 i na kraju podijelimo tu vrijednost sa 2.

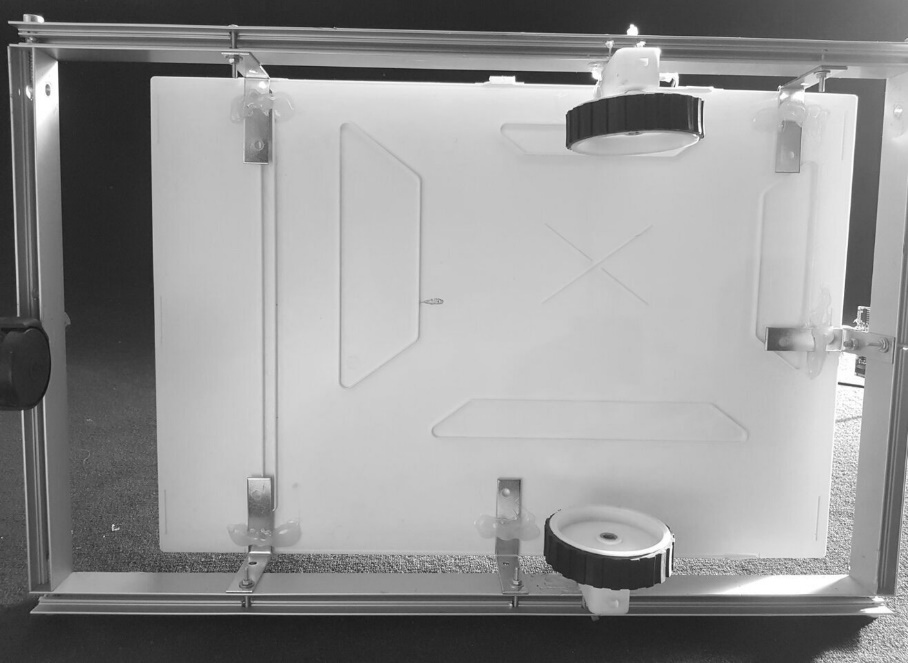


Slika 29: Izračun udaljenosti objekta

## 6.2 Konstrukcija

Konstrukciju robota čini ploča pleksiglasa koja je montirana na metalnom okviru, ispod same konstrukcije nalaze se 2 motora sa točkovima (Slika 34). Na prednjem dijelu okvira pričvršćena je metalna šipka radi pozicioniranja kamere, te ultrazvučni senzor (Slika 32). Na samoj ploči pričvršćena je Arduino Mega ploča, te H-most (Slika 30). Na zadnjoj strani metalne konstrukcije također je pričvrćen dodatni točkić, podešen na visinu od 5cm, kako bi se omogućilo kretanje (Slika 31).

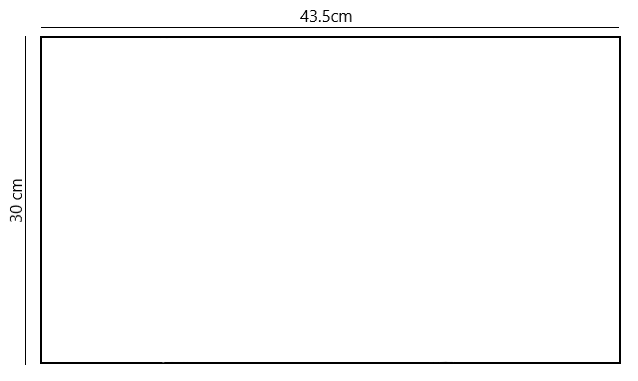
|  |  |
| --- | --- |
| Slika 30: Konstrukcija odozgo | Slika 31: Konstrukcija sa zada |
| Slika 32: Konstrukcija s prijeda | Slika 33: Konstrukcija sa strane |



Slika 34: Konstrukcija odozdo

### 6.2.1 Pleksiglas ploča

Pleksiglas ploča debljine je 3 mm, te dimenzija 43.5 x 30 cm. Glavna svrha pleksiglasne ploče jeste čuvanje laptopa na svojoj površini, koji će serijalnom komunikacijom slati upute kontroleru. Pričvršćena je ljepilom sa 5 L-profila, a zatim su L-profili šarafima pričvršćeni kroz rupe promjera 4.1 mm sa metalnom konstrukcijom. L-profili prave razmak između metalne konstrukcije i pleksiglasne ploče u iznosu od 31 mm.



Slika 35: Tlocrt pleksiglas ploče

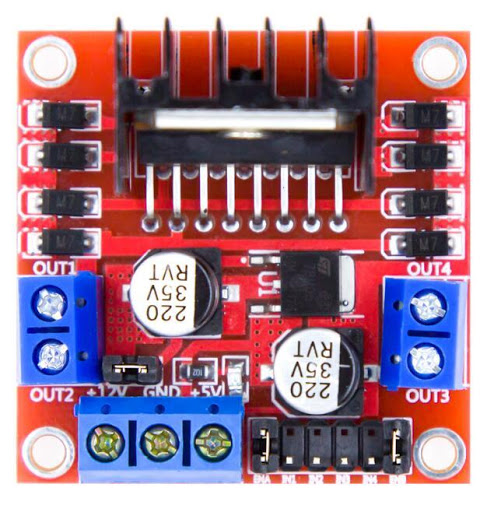
### 6.2.2 Metalni okvir

Metalni okvir dimenzija je 50 x 33 cm, te visine 28 mm i širine 13 mm. Pleksiglasna ploča pričvršćena je L-profilima sa metalnim okvirom u kojem su izbušene rupe promjena 4.1 mm za svaki L-profil, te 2 rupe kroz koje je pričvršćena aluminijska šipka. Također, za metalnu konstrukciju zalijepljeni su motori točkova dvokomponentnim ljepilom i dodatno pričvršćeni PVC vezicom.

### 6.2.3 Dvostruki H-Most

Dvostruki h-most korišten je u ovom projektu za upravljanje motorima. Ova vrsta h-mosta ima dodatan pasivni hladnjak za stabilniji rad. Omogućuje upravljanje 5V do 35V, te može podržati maksimalnu amperažu od 2A. Dimenzije istog su: 43 x 43 x 26 mm. Uopšteno, H-most je jednostavni sklop koji omogućava kontrolu DC motora, odnosno da ide naprijed ili nazad. Zatvaranjem dijagonalnog para prekidača (npr. 1 i 4), a otvaranjem drugog para, struja teče u jednom smjeru (Slika 36). Ako su 2 i 3 zatvoreni a 1 i 4 otvoreni, smjer struje se mijenja (Slika 35). Korištenje dvostrukog H-mosta omogućit će jednostavnu implementaciju kretanja unazad, ukoliko se u budućnosti javi potreba za istim. Razlika između običnog H-mosta i dvostrukog H-mosta, jeste to što dvostruki kao što i samo ime govori sadrži dva H-mosta.

|  |  |
| --- | --- |
| Slika 36: Kretanje struje u h-mostu 1 | Slika 37: Kretanje struje u h-mostu 2 |



Slika 38: Dvostruki h-most

### 6.2.4 Točkovi i metalna šipka

Točkovi koji se nalaze na motorima su plastični. Težina jednog točka sa motorom iznosi oko 50g, prečnik 70 mm, širina točka je 20 mm. Na metalni okvir su fiksirani koristeći dvokomponentno ljepilo. Robotu dodaju 47 mm visine, što ga dovodi na ukupnih 106 mm visine. Metalna šipka koristi se kao postolje za kameru, te joj stvara neophodnu visinu kako bi imala što veće vidno polje i omogućila veću širinu linija. Dimenzija je 210 x 5 mm i pričvršćena je za metalnu konstrukciju sa 2 šarafa kroz rupe promjera 3 mm. Također za metalnu šipku je vrućim ljepilom te PVC vezicom pričvršćen ultrazvučni modul.

## 6.3 Napajanje

Maksimalni Arduino izlaz struje iznosi 5V, te nije dovoljan za pokretanje motora, u tu svrhu neophodno je dovesti vanjski izvor napajanja koji će omogućiti dovoljno voltažu. U tu svrhu, korištena je eksterna Lithium Polymer baterija od 11.1V.

## 6.4 Kamera

Kamera je neizostavan dio sistema računarskog vida, kako omogućava unos slika (videa). Korištena je eksterna web kamera, rezolucije 640 x 480, sa štipaljkom za pričvršćivanje na metalnu konstrukciju. Zbog svoje relativno male rezolucije, visina metalne šipke ima ključnu ulogu u kreiranju većeg vidnog polja.



Slika 39: Web kamera

# **7. Kontrolisano okruženje**

Kontrolisano okruženje kreirano je koristeći crnu podlogu (etison) dimenzija 4 x 1.5 m, po kojem se robot kreće. Također neophodno je imati bijele trake za kretanje na podlozi, te će to u dokazivanju koncepta odnosno praktičnoj demonstraciji biti učinjeno koristeći bijelu izolir traku sa širinom između dvije trake od 60-80 mm.

Svjetlost također igra ključnu ulogu u kreiranju kvalitetnog okruženja, te je testirano najbolje funkcionisanje softvera u okruženjima sa sobnom jačinom svjetlosti tj. 20 – 50 Lux-a.

# **8. Rezultati testiranja**

Testiranje softvera za detekciju saobraćajne trake, kao što je već spomenuto, učinjeno je koristeći fizičku robotsku platformu. Tokom prvih faza testiranja, korišteni su veći i snažniji motori, sa točkovima veće mase i obima. Međutim takva postavka izazivala je brojne probleme pregrijavanja i stvarala brojne probleme pri testiranju. Kretanje robota podijeljeno je na tri brzine odnosno dijela u kasnijem testiranju. Prvobitna zamisao bila je podijeliti kretanje robota (naprijed, desno i lijevo) na po tri brzine zavisno od toga u kojem dijelu granica tolerancije kretanja se nalazi, te kako bi se što više ugladilo kretanje istog. Međutim nakon zamjene motora manjima, pojavilo se ograničenje u smislu snage motora, gdje robota zbog svoje mase (zajedno sa laptopom) nije bilo moguće podijeliti na više od jedne brzine, te je nakon toga podjela radi ostanka dosljednosti prvobitnoj ideji odrađena na sljedeći način. Nakon što Arduino primi kontrolnu poruku, izvršava je, te postavlja određenu pauzu (delay), kako bi dopustio izvršenje prethodne komande na određeno vrijeme prije uzimanja sljedećeg frejma i poruke, npr. za kretanje naprijed prvom (najvećom) brzinom, Arduino će izvršiti komandu te pauzirati izvršenje na 100 ms, kako bi se omogućilo vrijeme izvršenja prije izvršavanja sljedeće poruke. Ovime je izvršen znatan doprinos uglađenosti kretanja robota. Također pored zamjene točkova i motora, neophodno je bilo dodati postolje za kameru, te izvršiti modifikaciju zadnjeg pratećeg točka, kako je prvi zapinjao za podlogu te znatno otežavao kretanje.

Testiranje je izvršeno unutar kontrolisanog okruženja (7. Kontrolisano okruženje) odnosno okruženja ograničene svjetlosti, koristeći dvije bijele trake, napravljene od tzv. izolir trake na crnoj podlozi (Slika 40). Testiranje zaustavljanja, izvršeno je postavljajući razne predmete ispred ultrazvučnog modula tokom kretanja robota (20cm ili bliže) (Slika 41). Mogućnost izbjegavanja kolizije najviše doprinosi sigurnosti, npr. ukoliko bi se softver primijenio na primjeru autonomnog viljuškara (objašnjeno u 9. Budućnost projekta), kretanje će se obustaviti ukoliko teret padne sa viljuškara ili ukoliko se čovjek nađe ispred istog. Mana ovakvog sistema jeste moguće usporavanje kretanja i izvršenja cjelokupnog zadatka (prenosa tereta u ovom slučaju).

|  |  |
| --- | --- |
| Slika 40: Testiranje kretanja | Slika 41: Testiranje zaustavljanja |

# **9. Budućnost projekta**

Jedna od potencijalnih budućnosti projekta jeste unutar skladišta, gdje se i dalje u velikoj mjeri koristi ljudska asistencija. Kreiranjem kontrolisanog okruženja bilo bi moguće kreirati autonomnog viljuškara ili sličan tip vozila koji bi samostalno prevozio teret. Također, unutar ovog projekta implementirana je i detekcija objekata, čime bi se znatno povećala sigurnost drugih radnika unutar skladišta ili osiguralo zaustavljanje u slučaju pada tereta sa vozila.

Druga potencijalna budućnost projekta jeste na aerodromima, tj. omogućavanje jednostavnijeg i sigurnijeg način predaje i preuzimanja prtljaga. Također, jednostavnim izmjenama poput dodavanja vage na platformu robota moguće bi bilo mjeriti masu prtljaga, čime bi se olakšao proces „check-ina“, te povećala sigurnost, s obzirom da ljudska sentimentalnost (radnika) ponekad može promijeniti ishod ovog procesa (prihvatanje više od dozvoljene mase prtljaga). Za proces preuzimanja i stavljanja prtljaga na robot sa autonomnim kretanjem, bio bi zadužen drugi robot fiksiran za traku na kojoj se kreće prtljag.

# **10. Zaključak**

Računarski vid je oblast koja je definitivno u usponu, te će se kao takva i zadržati. Spajanjem svih prethodno opisanih dijelova dobit ćemo i razumjeti finalni proizvod ovog rada tj. softver za detekciju saobraćajne trake za autonomno vođenje robota. Time je zadovoljen cilj ovog rada koji dokazuje superiorne mogućnosti oblasti računarskog vida.

Za detaljnije upoznavanje sa projektom, te praćenje napretka razvoja aplikacije moguće je posjediti Github profil[[13]](#footnote-13), na kojem je dostupan cjelokupni kod softvera.

# **11. Slike**

[Slika 1: Arduino razvojna ploča 5](#_Toc33454171)

[Slika 2: Prikaz serijske komunikacije 6](#_Toc33454172)

[Slika 3: Arduino IDE 7](#_Toc33454173)

[Slika 4: Library manager 8](#_Toc33454174)

[Slika 5: Serial monitor 9](#_Toc33454175)

[Slika 6: Serial plotter 10](#_Toc33454176)

[Slika 7: Početne funkcije 12](#_Toc33454177)

[Slika 8: Arduino Uno (R3) 14](#_Toc33454178)

[Slika 9: LilyPad Arduino 14](#_Toc33454179)

[Slika 10: Arduino Mega (R3) 15](#_Toc33454180)

[Slika 11: Arduino Shield 15](#_Toc33454181)

[Slika 12: CMYK model 18](#_Toc33454182)

[Slika 13: RGB model 19](#_Toc33454183)

[Slika 14: Hue vrijednosti 20](#_Toc33454184)

[Slika 15: HSV/HSB Model 20](#_Toc33454185)

[Slika 16: HSL Model 21](#_Toc33454186)

[Slika 17: Rectangle konstrukcija 22](#_Toc33454187)

[Slika 18: Binarna slika 24](#_Toc33454188)

[Slika 19: Canny transformacija 25](#_Toc33454189)

[Slika 20: Morfološke transformacije 26](#_Toc33454190)

[Slika 21: m-c prostor 28](#_Toc33454191)

[Slika 22: Konverzija u mc prostor 28](#_Toc33454192)

[Slika 23: Primjer hough transformacije 29](#_Toc33454193)

[Slika 24: Dizajn izlaza aplikacije 30](#_Toc33454194)

[Slika 25: Način komunikacije modula 46](#_Toc33454195)

[Slika 26: HC-SR04 modul 46](#_Toc33454196)

[Slika 27: Šema povezivanja senzora 47](#_Toc33454197)

[Slika 28: Slanje signala 47](#_Toc33454198)

[Slika 29: Izračun udaljenosti objekta 48](#_Toc33454199)

[Slika 30: Konstrukcija odozgo 49](#_Toc33454200)

[Slika 31: Konstrukcija sa zada 49](#_Toc33454201)

[Slika 32: Konstrukcija s prijeda 49](#_Toc33454202)

[Slika 33: Konstrukcija sa strane 49](#_Toc33454203)

[Slika 34: Konstrukcija odozdo 50](#_Toc33454204)

[Slika 35: Tlocrt pleksiglas ploče 51](#_Toc33454205)

[Slika 36: Kretanje struje u h-mostu 1 52](#_Toc33454206)

[Slika 37: Kretanje struje u h-mostu 2 52](#_Toc33454207)

[Slika 38: Dvostruki h-most 52](#_Toc33454208)

[Slika 39: Web kamera 53](#_Toc33454209)

[Slika 40: Testiranje kretanja 55](#_Toc33454210)

[Slika 41: Testiranje zaustavljanja 55](#_Toc33454211)

# **12. Programski kodovi**

[Programski kod 1: VideoCapture 31](#_Toc33454212)

[Programski kod 2: Mat klase 31](#_Toc33454213)

[Programski kod 3: Preuzimanje frejma 31](#_Toc33454214)

[Programski kod 4: Promjena veličine frejma 32](#_Toc33454215)

[Programski kod 5: Rect Konstrukcija 32](#_Toc33454216)

[Programski kod 6: Canny konstrukcija 33](#_Toc33454217)

[Programski kod 7: Morfoloske operacije 34](#_Toc33454218)

[Programski kod 8: Hough transformacija za detekciju linija 34](#_Toc33454219)

[Programski kod 9: Iscrtavanje linija 35](#_Toc33454220)

[Programski kod 10: Tolerancija 35](#_Toc33454221)

[Programski kod 11: Najveci razmak 36](#_Toc33454222)

[Programski kod 12: Kretanje pravo 36](#_Toc33454223)

[Programski kod 13: Kretanje desno 37](#_Toc33454224)

[Programski kod 14: Kretanje u lijevo 38](#_Toc33454225)

[Programski kod 15: Serial header 39](#_Toc33454226)

[Programski kod 16: Serijalno slanje podataka 40](#_Toc33454227)

[Programski kod 17: Arduino setup 40](#_Toc33454228)

[Programski kod 18: Serial event 41](#_Toc33454229)

[Programski kod 19: Ultrazvučna provjera 42](#_Toc33454230)

[Programski kod 20: Kretanje naprijed 43](#_Toc33454231)

[Programski kod 21: Kretanje u lijevo 44](#_Toc33454232)

[Programski kod 22: Kretanje u desno 45](#_Toc33454233)

# **Literatura**

[1] „Language Reference“, [Na Mreži], Dostupno: <https://www.arduino.cc/reference/en/> [Posljednji pristup 20.01.2020]

[2] „Arduino Uno Specification“, [Na Mreži], Dostupno: <https://www.elprocus.com/what-is-arduino-uno-r3-pin-diagram-specification-and-applications/> [Posljednji pristup 20.01.2020]

[3] „Arduino Uno Pin Diagram“, [Na Mreži], Dostupno: <https://components101.com/microcontrollers/arduino-uno> [Posljednji pristup 21.01.2020]

[4] „Serial Peripheral Interface“, [Na Mreži], Dostupno: <https://www.arduino.cc/en/reference/SPI> [Posljednji pristup 30.01.2020]

[5] „Analog reference“, [Na Mreži], Dostupno: <https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/> [Posljednji pristup 30.01.2020]

[6] „Computer Vision with the OpenCV Library“, Gary Bradski, Adrian Kaehler, [Na Mreži], Dostupno: <https://books.google.ba/books?id=seAgiOfu2EIC&lpg=PR3&ots=hUL16njFKe&dq=computer%20vision&lr&pg=PR9> [Posljednji pristup 01.02.2020]

[7] „HSL and HSV“, [Na Mreži], Dostupno: <https://psychology.wikia.org/wiki/HSL_and_HSV> [Posljednji pristup 07.02.2020]

[8] „OpenCV“, [Na Mreži], Dostupno: <https://docs.opencv.org/master/> [Posljednji pristup 07.02.2020]

[9] „Canny edge detector“, [Na Mreži], Dostupno: <https://medium.com/@ssatyajitmaitra/what-canny-edge-detection-algorithm-is-all-about-103d94553d21> [Posljednji pristup 19.02.2020]

[10] „Types of Morphological Operations“, [Na Mreži], Dostupno: <https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html> [Posljednji pristup 19.02.2020]

[11] „Binary Image Processing“, [Na Mreži], Dostupno: <https://www.sciencedirect.com/topics/engineering/binary-image> [Posljednji pristup 19.02.2020]

[12] „UltraSonic sensor“, [Na Mreži], Dostupno: <https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/> [Posljednji pristup 20.02.2020]

1. <https://www.arduino.cc/en/Guide/Introduction> [↑](#footnote-ref-1)
2. <https://www.arduino.cc/en/Reference/VariableDeclaration> [↑](#footnote-ref-2)
3. <https://www.sas.com/en_us/insights/analytics/computer-vision.html> [↑](#footnote-ref-3)
4. <https://www.domo.com/learn/data-never-sleeps-5> [↑](#footnote-ref-4)
5. <https://www.tractica.com/newsroom/press-releases/computer-vision-hardware-and-software-market-to-reach-48-6-billion-by-2022/> [↑](#footnote-ref-5)
6. <https://www.sitepoint.com/a-short-guide-to-color-models/> [↑](#footnote-ref-6)
7. <https://www.geeksforgeeks.org/computer-graphics-the-rgb-color-model/> [↑](#footnote-ref-7)
8. <https://docs.opencv.org/> [↑](#footnote-ref-8)
9. <https://www.sciencedirect.com/topics/engineering/binary-image> [↑](#footnote-ref-9)
10. <https://www.dictionary.com/browse/hysteresis> [↑](#footnote-ref-10)
11. <https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html> [↑](#footnote-ref-11)
12. <https://docs.opencv.org/3.4/d4/d76/tutorial_js_morphological_ops.html> [↑](#footnote-ref-12)
13. <https://github.com/faski1> [↑](#footnote-ref-13)