Co-funded by the European Union

AI PROJEKTI

Tehnička škola Pirot

KA220-VET - Partnerstva za saradnju u oblasti stručnog obrazovanja i obuke

Naziv projekta: AI alati za strukovne škole

Datum dokumenta: mart 2026

Ovaj materijal je prikupljen i pripremljen za potrebe Erasmus projekta od:
Tehnička škola Pirot (autor: Bojan Ćirić, koautori: Boban Blagojević, Aleksandar Madić)
ICEP (autor: Ladislav Mariš, koautor: Adelaida Fanfarova)
Agrupamento de Escolas Tomas Cabreira (autor: Sandra Nobre, koautori: Rui Dias, Gilherme Mota, Carla Lima)
Prevel: Bojana Stojanović

INFORMACIJE O ADRESI

Takovska 22, Pirot, Srbija
Veb stranica: https://book.tsp.edu.rs

Sadržaj

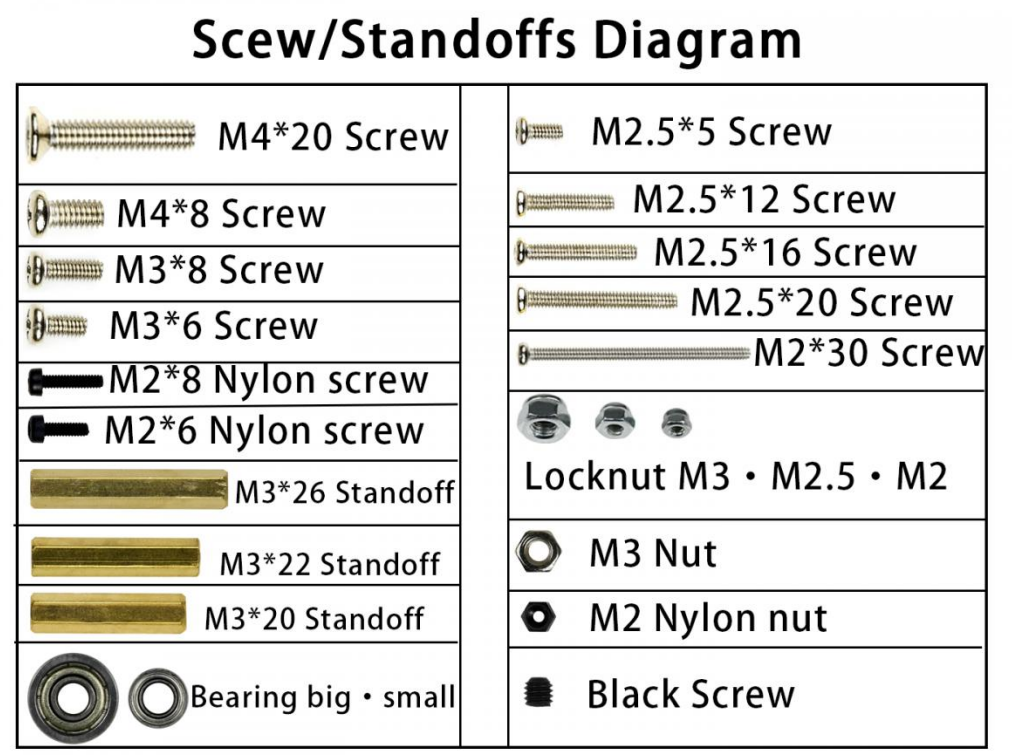
I. PiRACER AI AUTOPILOT

1. PI-RACER DEEP LEARNING AUTOPILOT

1.1 PiRacer Uputstvo za montažu

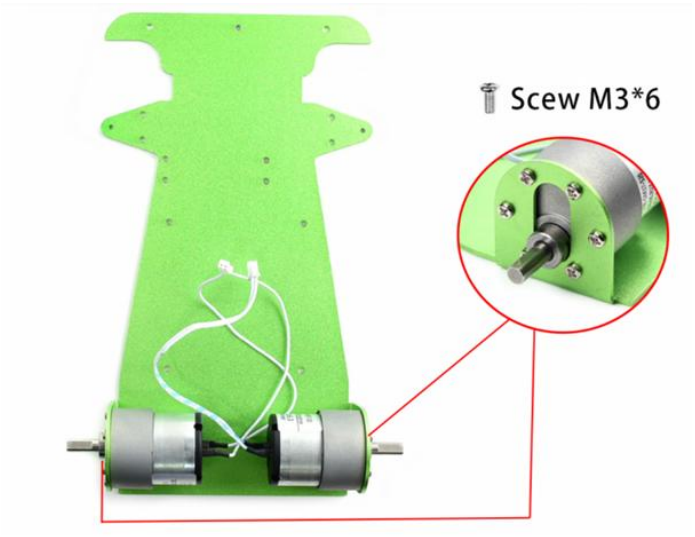
Dijagram vijaka / odstupanja

Dijagram za referencu. Imajte na umu da šrafovi koji dolaze sa servo točkom nisu navedeni ovde.



1. Pričvrstite motore na metalnu šasiju

Napomena: Nemojte koristiti M3 * 8. To je duže i može oštetiti motor.



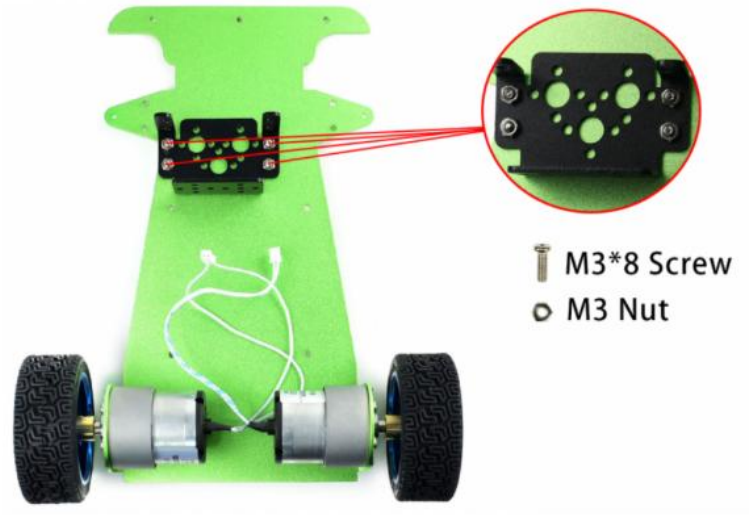
2. Dodajte spojnice na točkove

Prvo, ubacite crni vijak u spojnicu. Zatim dodajte spojnicu na točak. Možda ćete morati da pritisnete spojnicu u točak. Pričvrstite spojnicu na točak pomoću M4 * 8 vijka.

3. Sastavite točkove

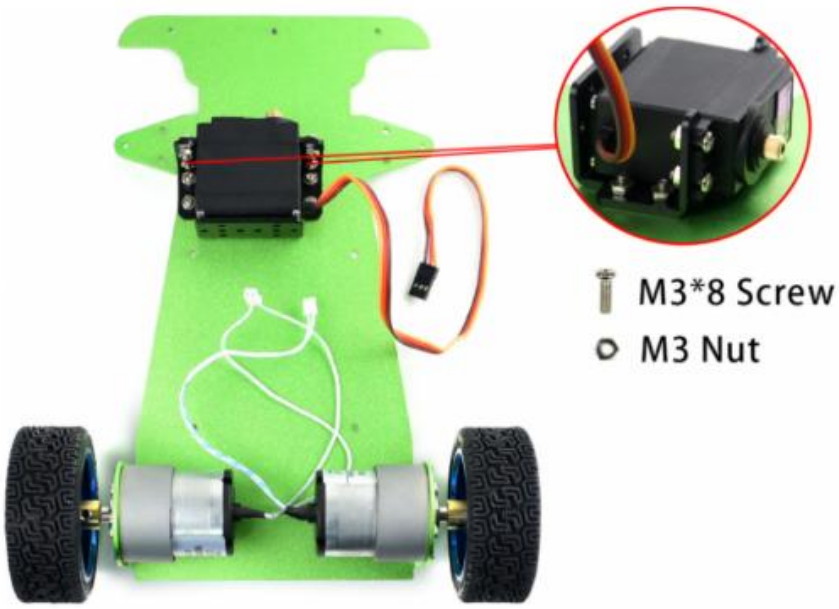
Zategnite crni vijak da biste pričvrstili spojnicu na ravnu stranu osovine

4. Montirajte držač servo na metalnu šasiju



5. Pričvrstite servo na držač i pomoću vijaka i matica

Proverite da li je servo ispravno. Spoljna osovina treba da bude u centru.

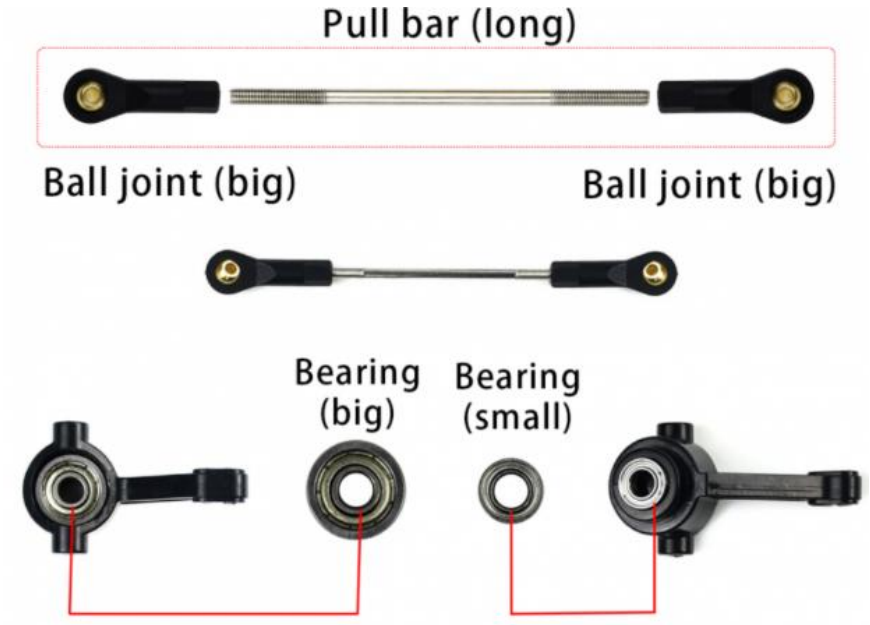


6. Sastavite servo vučnu šipku

Pull bar se kombinuje pomoću dva kuglična zgloba, jedan ravan i jedna lopta, i kratka šipka. Dva kuglična zgloba treba da budu okomita jedna na drugu. Pričvrstite servo točak (rog) na držač servo točka pomoću vijaka koji ste dobili sa servo točkom. Zatim pričvrstite ravni kuglični zglob na držač servo točka. Imajte na umu da je žleb servo točka je prema spolja.

7. Sastavite vučnu šipku prednjeg točka i zglobove upravljača

Vučna šipka prednjeg točka se kombinuje pomoću dva kuglična zgloba i duge šipke. Zatim stavite ležajeve u zglobove upravljača. Svaki zglob treba mali i veliki ležaj.



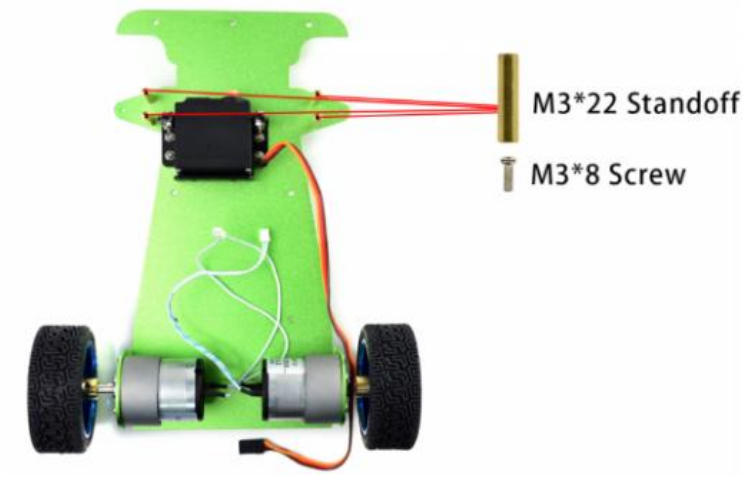
8. Sastavite servo pull bard, prednji točkovi pull bard, i zglobovi upravljača

Pričvrstite servo pull bar na vrhu, a zatim prednji točak pull bar, i na kraju zglobova. Veći ležaj treba prema unutra.

9. Pričvrstite točkove na zglob upravljača

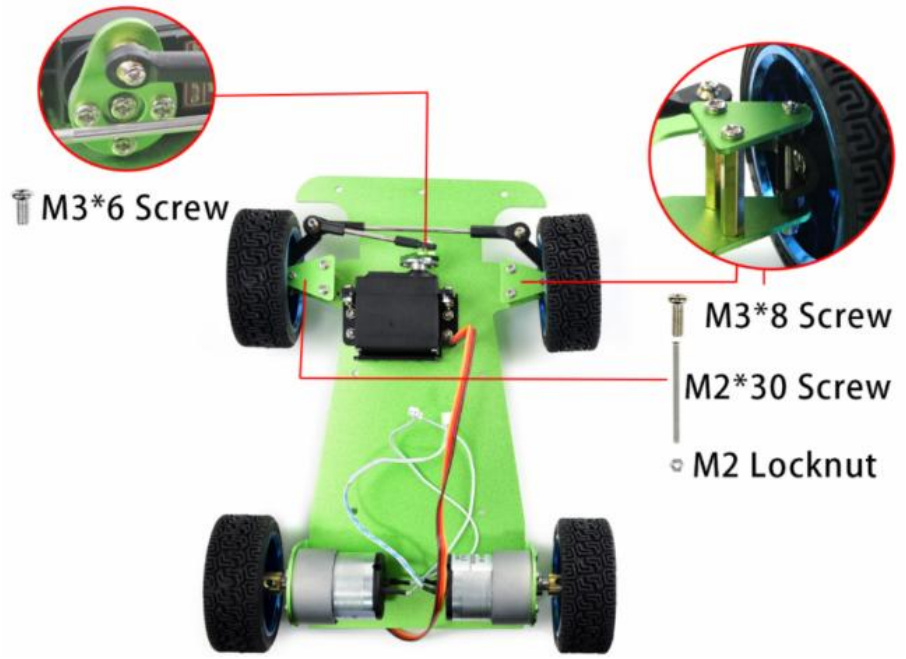
Matica treba da bude na spoljnoj strani točka, nasuprot zgloba. Uverite se da točak nije previše zategnut ili previše labav. Testirajte točak da biste bili sigurni da može slobodno da se kreće.

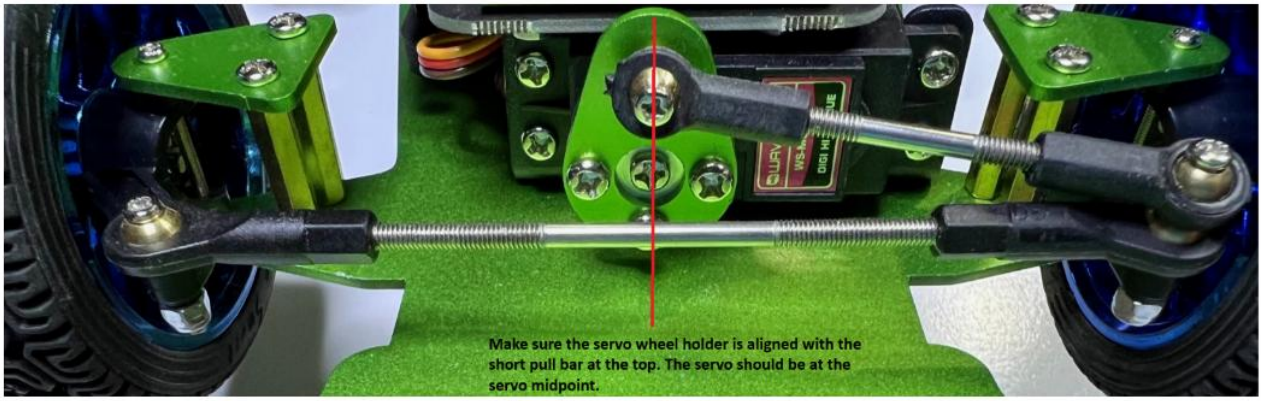
10. Dodajte M3 standoffs za prednje točkove



11. Sastavite kombinaciju prednjih točkova

Stavite servo točak na servo, pričvrstite ga pomoću M3 vijka. Pričvrstite točkove pomoću M2 vijaka i kontra matice i trougla ploče. Prednji točkovi treba da budu ravno napred. Podesite dugu traku za povlačenje ako je potrebno.

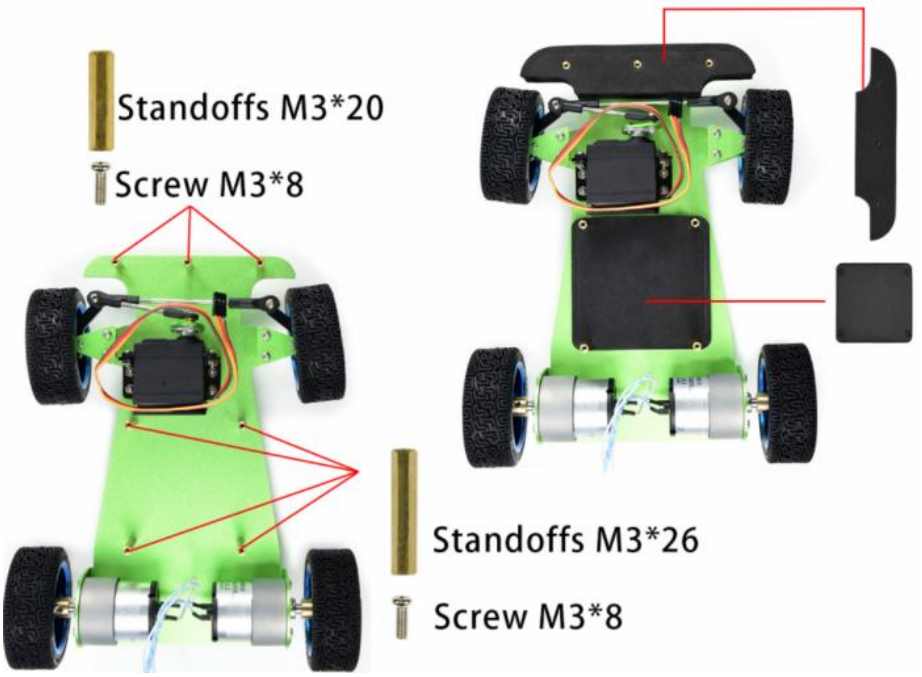




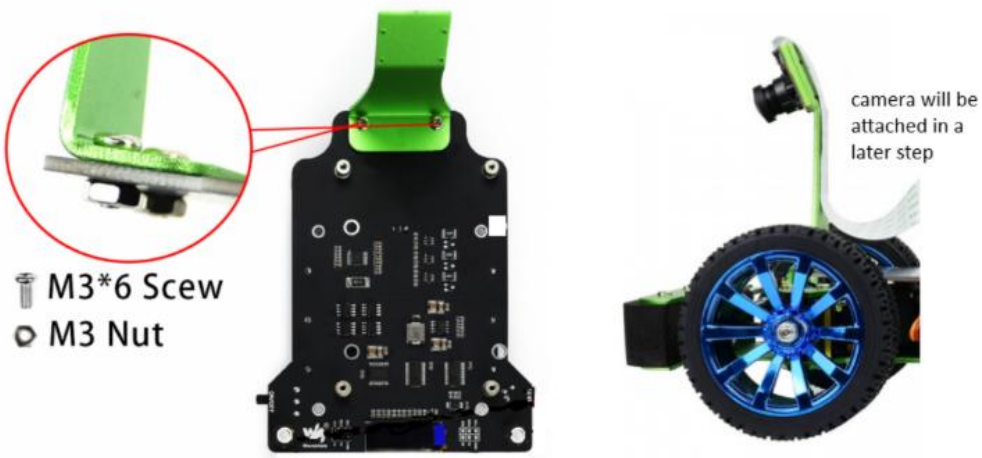
Prednji točkovi treba da budu ravno napred. Podesite dugu traku za povlačenje ako je potrebno.

12. Dodajte standoffs za PiRacer Ekpansion odbora i branik

Ubacite EVA filc jastučiće za branik i PiRacer Ekpansion odbor.

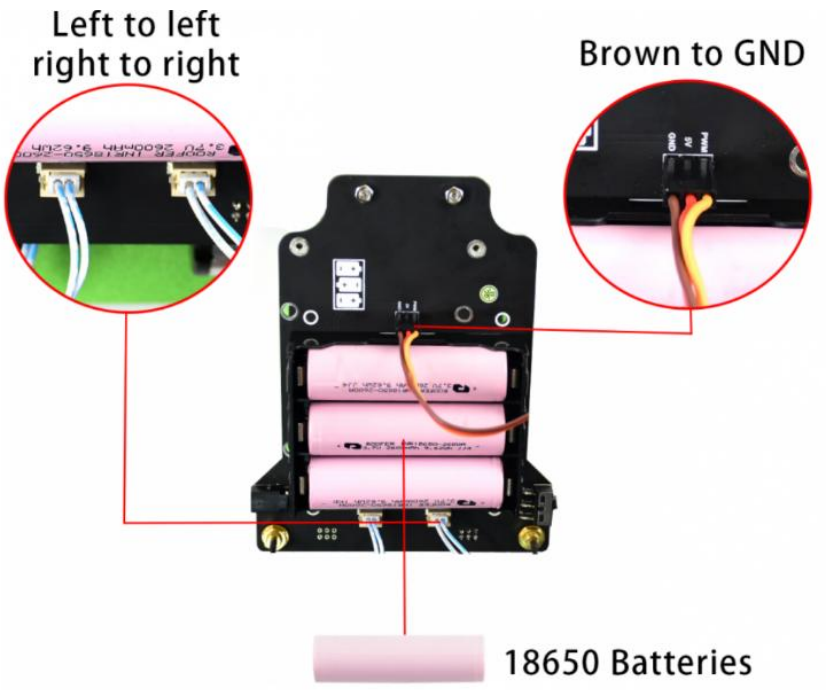


13. Pričvrstite držač kamere na PiRacer Ekpansion ploču.



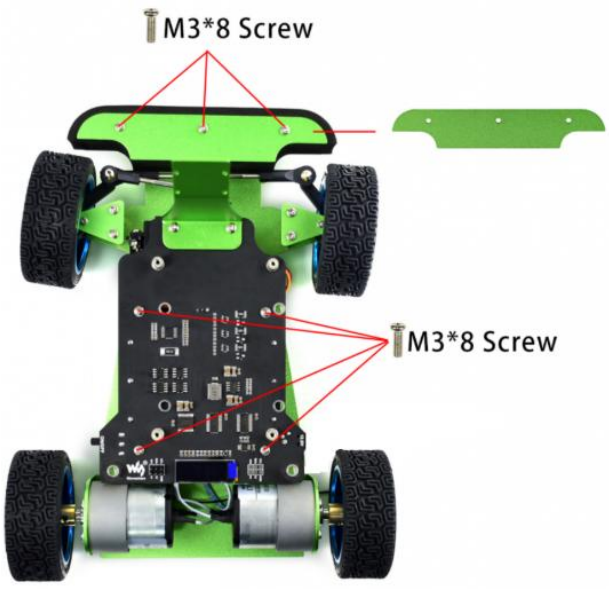
14. Umetnite baterije u pravom smeru

Povežite žice motora i servo na PiRacer Ekpansion odbor.



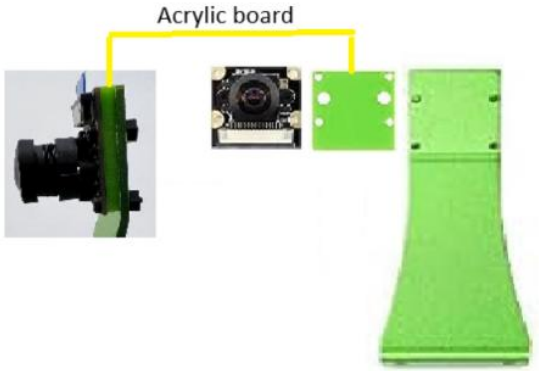
15. Pričvrstite ploču za proširenje i metalni branik

Podesite položaj motora i servo žica i pričvrstite PiRacer Ekpansion ploču na metalnu šasiju i pričvrstite metalni branik.

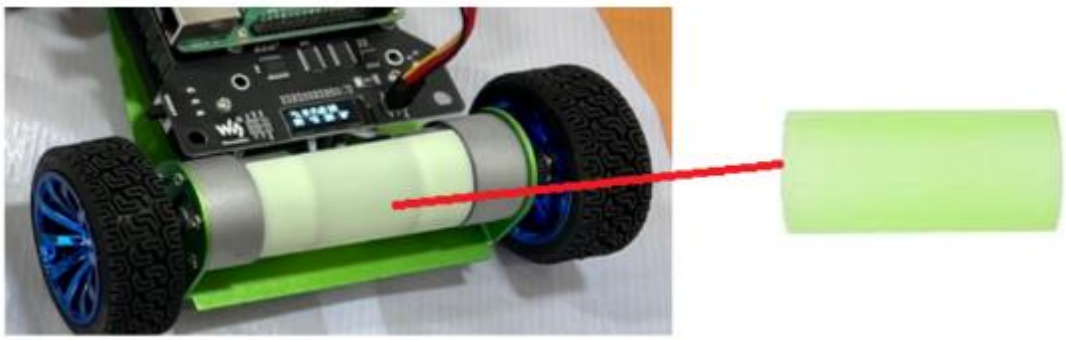


16. Montirajte kameru na držač pomoću najlonskih vijaka

Napomena: Akrilna ploča treba da bude između kamere i metalnog držača kako bi se izbeglo kratko spajanje.



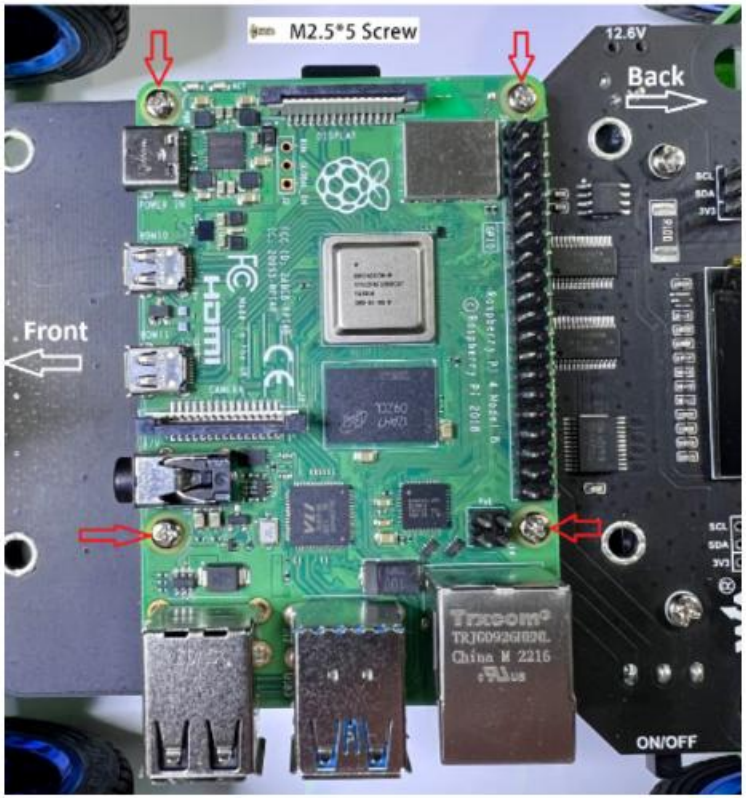
17. Pričvrstite kućište motora 3D-štampano na DC motore



18. Isključite Raspberry Pi i isključite punjač iz PiRacer Ekpansion ploče pre nego što nastavite.

19. Pričvrstite Raspberry Pi na PiRacer ploču za proširenje

GPIO igle treba da budu na zadnjem delu automobila.

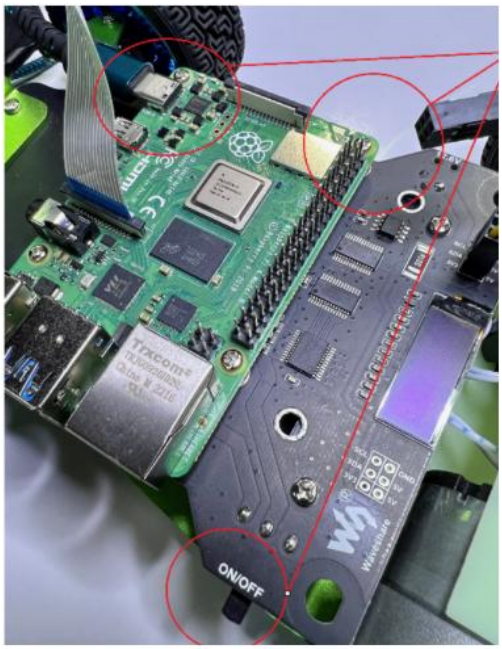


21. Pričvrstite trakasti kabl kamere na ulaz Raspberry Pi kamere

Plava strana prema USB portovima Pi je.

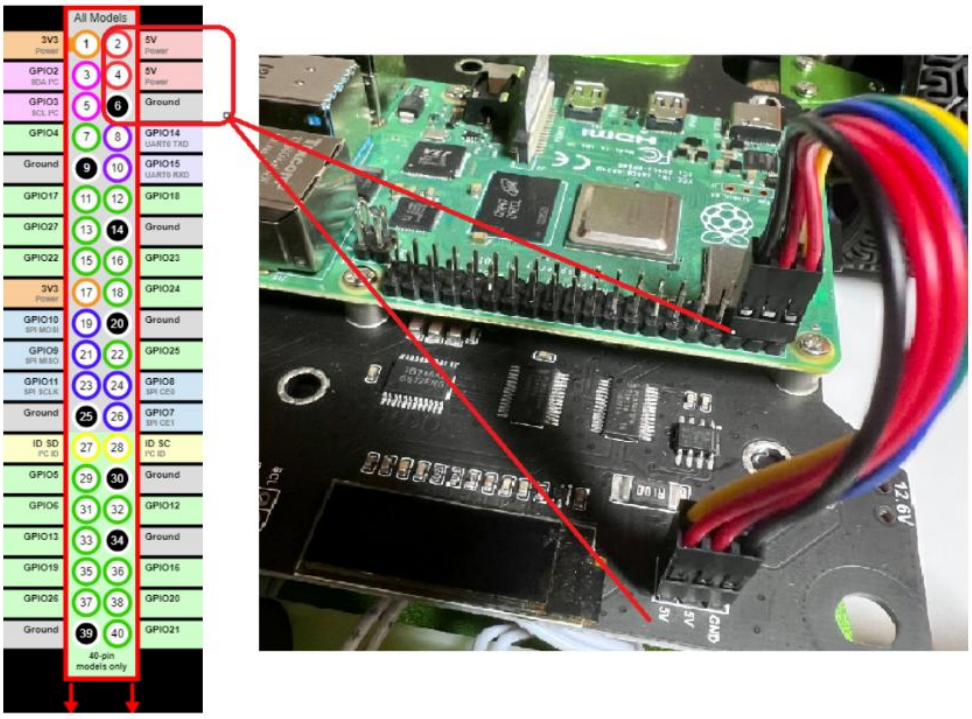
**** UPOZORENjE ** Uverite se da Raspberry Pi se ne napaja prilikom povezivanja 6 pin žice**

Takođe, ne napajajte Raspberry Pi preko USB-C (eksterno napajanje) dok ga napaja PiRacer Ekpansion ploča.



22. Povežite Raspberry Pi na PiRacer Ekpansion ploču pomoću 6PIN žica

Uporedite crveno|crveno|crno (5V|5V|uzemljenje) na Pi GPIO i crno|crveno|crveno (gnd|5V|5V) na PiRacer Expansion board.



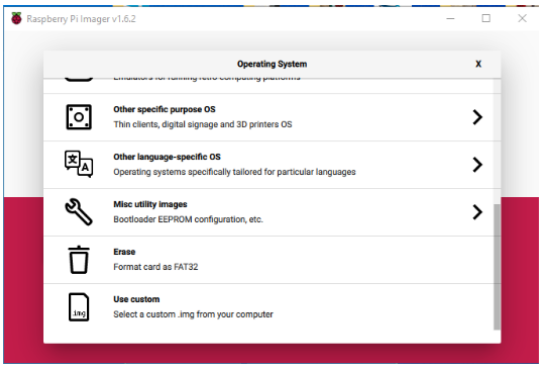
1.2 Raspian Legaci (Buster) desktop

Flash OS - Raspian Legaci (Buster) desktop

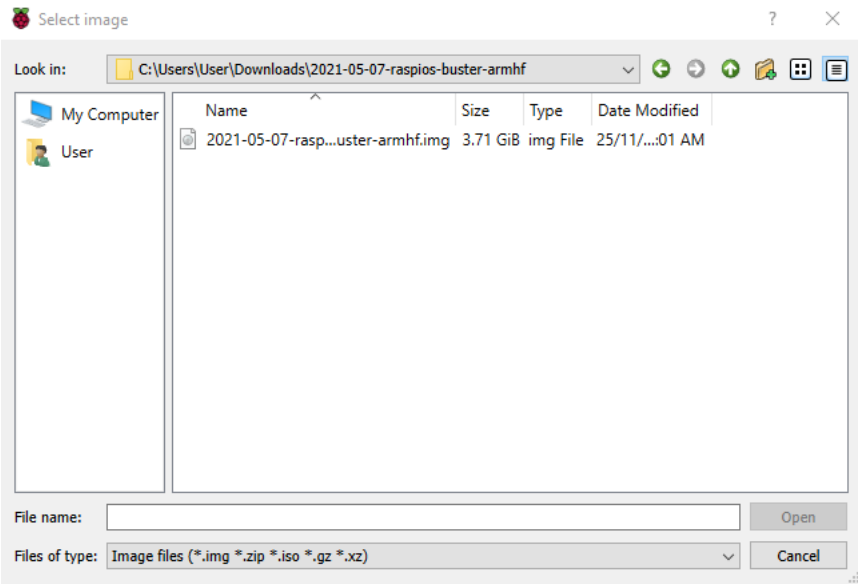
Idite na zvanično spremište za preuzimanje za Buster OS. **Sve ranije verzije Raspberry Pi OS-a mogu se naći i preuzeti ovde,** a direktno prethodni Raspberry Pi 'Buster' OS **zvanični link za preuzimanje je ovde**. Pogledajte dole kako to izgleda, preuzmite slikovnu datoteku klikom na istaknutu datoteku.

Preuzmite i instalirajte Raspberry Pi Imager sa https://www.raspberrypi.com/software/

Sada, hajde da otvorimo zvanični Raspberry Pi Imager, pogledajte ga na slici ispod. Vredi napomenuti ovde - ako pritisnete **| CTRL+SHIFT+X |** na tastaturi dok je u programu Raspberry Pi Imager otvoriće napredni skriveni meni. Ovaj skriveni meni vam omogućava da unapred konfigurišete Raspberry Pi sa SSH, VIFI akreditivima i podešavanjima lokalizacije.



Nakon što je to uradio, otvoriće se eksplorer datoteka. Idite na taj izdvojeni Disc Image File i kliknite na njega. Pogledajte ovo na slici ispod. Zatim otvorite sliku datoteke i to će spremiti Raspberry Pi Imager sa Raspberry Pi 'Buster' OS-om. Pronađi ga, izaberite ga i otvorite.



Ubacite Micro-SD karticu koju želite da se flash-uje u računar. Koristite **USB na Micro-SD adapter** ako je potrebno. Zatim kliknite na **| CHOOSE STORAGE |** dugme i izaberite umetnuti Micro-SD. Imajte na umu da će svi podaci koji su bili na vašoj Micro-SD kartici biti obrisani / trajno izbrisani nakon flash-ovanja. Zvanični Raspberry Pi Imager će sada izgledati kao na slici ispod.

Sa svime poredanim (desni OS je napunjen i ispravan Storage izabran) sada možete kliknuti na **|WRITE|** dugme za početak procesa flash-ovanja. Pogledajte ovaj proces na slici ispod.

Kada se flasho-vanje završi, automatski će isključiti Micro-SD karticu sa računara. Dakle, onda možete jednostavno fizički izvaditi svoj Micro-SD i ubaciti ga u Raspberry Pi. Zatim podesite Raspberry Pi normalno kao desktop računar. Kada se pokrene, dočekaće vas stara poznata pozadina, pogledajte sliku ispod, tako da ste uspešno flash-ovali 'Buster' OS na vaš Raspberry Pi. Sada ste spremni za nastavak upravljanja dobro poznatim OS-om.

Pratite Pi Wizard instrukciju da podesite lokaciju, tastaturu, WiFi, i dobiti ažuriranje

Menu / Preferences / Raspberry Pi Configuration/ Interfaces

- omogućiti kameru i I2C
- opciono: omogućite VNC za daljinski pristup
- Kliknite na dugme OK i ponovo pokrenite

Dodatni softver

Java 3.8.3 (pokrećite jednu po jednu komandu)

```
sudo apt install build-essential libncurses5-dev libgdbm-dev libnss3-dev libssl-dev
libreadline-dev libffi-dev -y
wget https://www.python.org/ftp/python/3.8.3/Python-3.8.3.tgz
tar -zxvf Python-3.8.3.tgz
cd Python-3.8.3
sudo ./configure --enable-optimizations
sudo make -j 4
sudo make altinstall
sudo python3.8 -m pip install boto3
sudo python3.8 -m pip install tqdm
```

Dodatni alati

```
sudo apt install chromium-browser -y
sudo apt-get install zip unzip -y
```

AVS CLI

Podrazumevani korisnički account na Raspberry Pi

```
sudo apt install awscli -y
sudo pip3 install --upgrade awscli
sudo pip3 install boto3

user - pi
pw - raspberry
```

1.3 VaveShare Branch za DonkeyCar Software

Instalirajte zavisnosti

```
sudo apt-get install build-essential python3 python3-dev python3-pip
python3-virtualenv python3-numpy python3-picamera python3-pandas
python3-rpi.gpio i2c-tools avahi-utils joystick libopenjp2-7-dev
libtiff5-dev gfortran libatlas-base-dev libopenblas-dev
libhdf5-serial-dev git ntp -y
```

Opciono?

```
sudo apt-get install libilmbase-dev libopenexr-dev libgstreamer1.0-dev
libjasper-dev libwebp-dev libatlas-base-dev libavcodec-dev libavformat-dev
libswscale-dev libqtgui4 libqt4-test -y
```

Podešavanje virtuelnog okruženja

```
python3 -m virtualenv -p python3 env --system-site-packages
echo "source ~/env/bin/activate" >> ~/.bashrc
source ~/.bashrc
```

Instalirajte DonkeyCar Python kod - VaveShare Branch za DonkeyCar Softvare 3.1.0

```
mkdir projects
cd projects
git clone https://github.com/waveshare/donkeycar

cd donkeycar
git checkout master
pip install -e .[pi]

pip install tensorflow==1.13.1
```

Nije potrebno pip install numpi --upgrade

```
pip install protobuf == 3.20.*
```

Test tenzorflov verzija - treba da pokaže 1.13.1

```
python -c "import tensorflow; print(tensorflow.__version__)"
```

NAPOMENA: Nije mogao pokrenuti model na automobilu dok ne pokrenete sledeća dva

```
pip install https://github.com/lhelontra/tensorflow-on-arm/releases/download/v2.2.0/
        tensorflow-2.2.0-cp37-none-linux_armv7l.whl

pip install tensorflow==1.13.1
```

Izmeni camera.py da dodate self.camera.hflip = True

```
sudo nano /home/pi/projects/donkeycar/donkeycar/parts/camera.py
```

```
class PiCamera(BaseCamera):
    def __init__(self, image_w=160, image_h=120, image_d=3, framerate=20):
        from picamera.array import PiRGBArray
        from picamera import PiCamera

        resolution = (image_w, image_h)
        # initialize the camera and stream
        self.camera = PiCamera() #PiCamera gets resolution (height, width)
        self.camera.vflip = True
        self.camera.hflip = True
        self.camera.resolution = resolution
        self.camera.framerate = framerate
        self.rawCapture = PiRGBArray(self.camera, size=resolution)
        self.stream = self.camera.capture_continuous(self.rawCapture,
            format="rgb", use_video_port=True)

        # initialize the frame and the variable used to indicate
        # if the thread should be stopped
```

Opciono Instalirajte OpenCV

```
sudo apt install python3-opencv -y
```

Testirajte sa:

python -c "import cv2"

1.4 Prvi koraci sa DonkeyCar

Otvorite prozor terminala unesite sledeću komandu

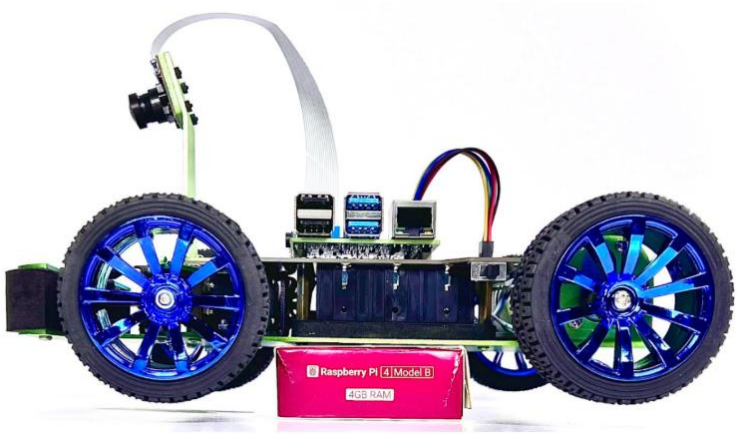
Kreirajte aplikaciju DonkeyCar

Ovo će stvoriti fasciklu pod nazivom mycar sa svim Python kodom potrebnim za vožnju automobila.

Kalibrirajte prednji upravljač

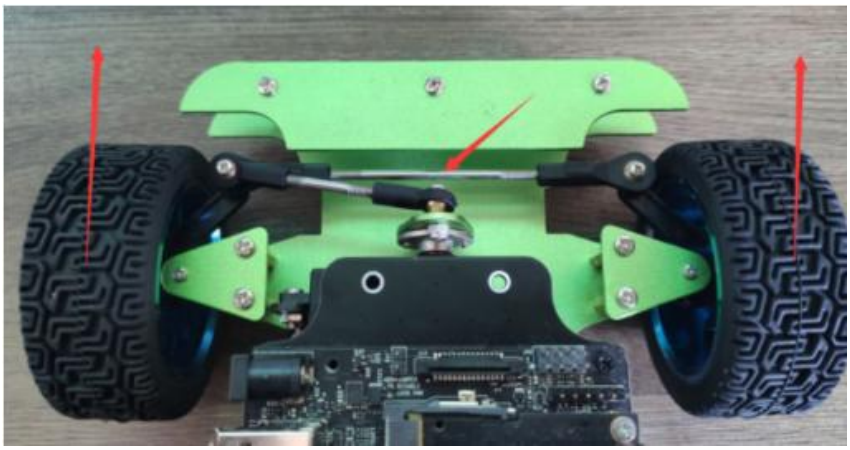
Uverite se da je vaš automobil sa zemlje kako biste sprečili odbeglu situaciju.

Koristite malu kutiju kao što je na Raspberry Pi ušao.

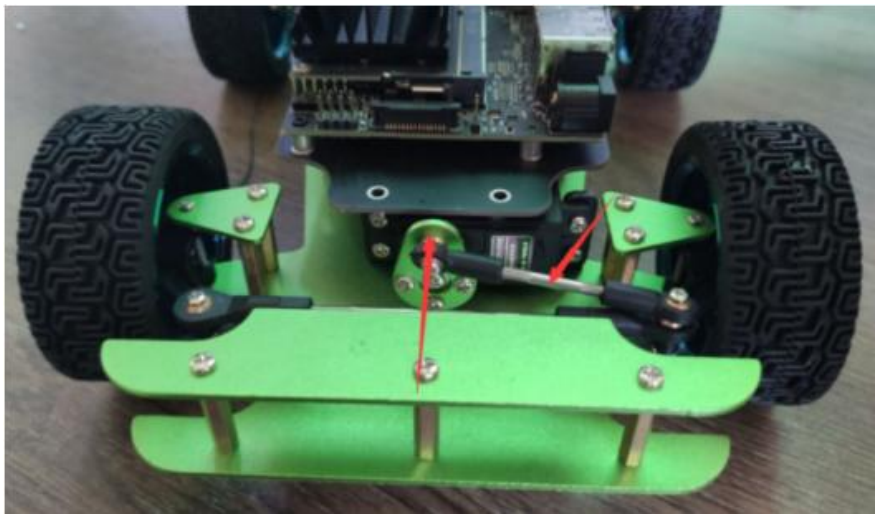


Da biste bili sigurni da DonkeyCar može voziti pravo i napraviti potrebne skretanja na stazi, hardver i softver automobila moraju biti kalibrirani.

Prednji točkovi treba da budu pravo napred. Po potrebi podesite dugu polugu za povlačenje



Držač servo točkova treba da bude usklađen sa kratkom polugom na vrhu. Po potrebi podesite kratku polugu.



Kalibrirajte servo softver

Kako da pronađite levo, centralno i desno upravljanje PVM za servo ovog automobila.

Centar treba da bude na pola puta između leve i desne strane. Primer:

Left 200
right 560
centar će biti 380

U prozoru terminala unesite sledeće

```
cd ~/mycar
donkey calibrate --channel 0 --bus=1
```

Probajte vrednosti 300, 400, 500 i vidite kako se upravljač menja. Shvatite maksimalno skretanje ulevo i udesno.

Kada imate brojeve, uredite config.py i ažurirajte vrednosti koje ste pronašli.

Gas (throttle) treba podesiti pomoću vaših brojeva za upravljanje. Gas je već ispravno podešen.

Nano config.py

```
#STEERING
STEERING_CHANNEL = 0            #channel on the 9685 pwm board 0-15
STEERING_LEFT_PWM = 200         #pwm value for full left steering
STEERING_RIGHT_PWM = 560        #pwm value for full right steering

#THROTTLE
THROTTLE_CHANNEL = 0            #channel on the 9685 pwm board 0-15
THROTTLE_FORWARD_PWM = 4095     #pwm value for max forward throttle
THROTTLE_STOPPED_PWM = 0        #pwm value for no movement
THROTTLE_REVERSE_PWM = -4095    #pwm value for max reverse throttle
```

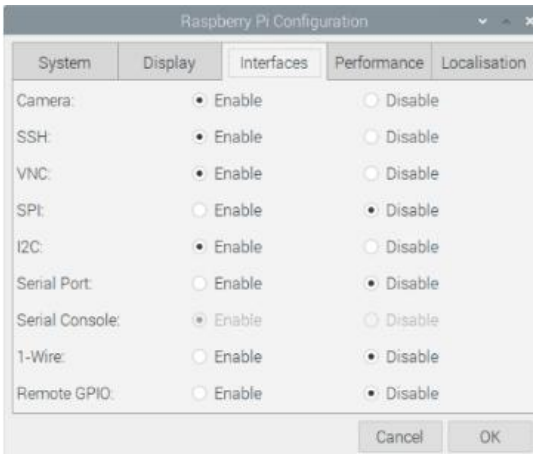
Instalirajte uslugu OLED displeja

```
cd ~
git klon https://github.com / vaveshare / pi-displej
ЦD Pi-displej
sudo ./install.OMILjENO
cd ~
```

2. RASPBERRY PI DALJINSKI PRISTUP KORISTEĆI REALVNC

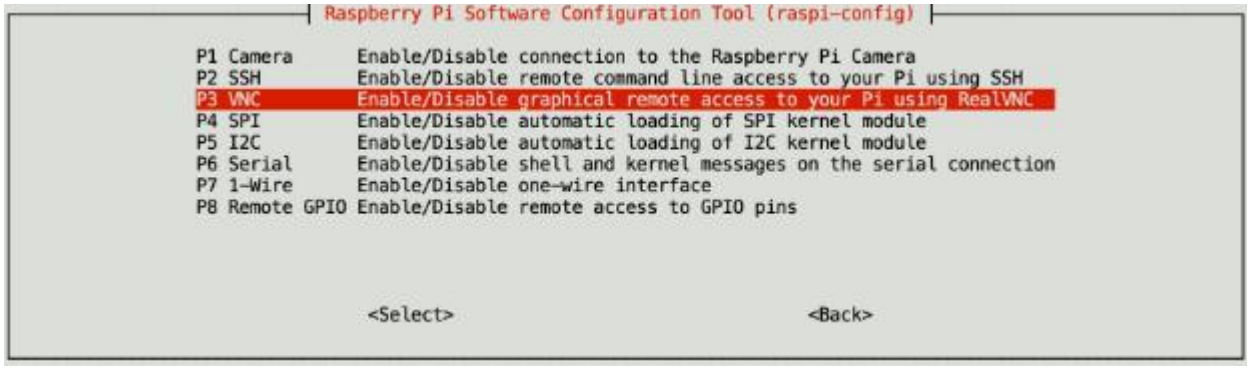
2.1 Omogućite VNC u Raspian OS na a ili b način:

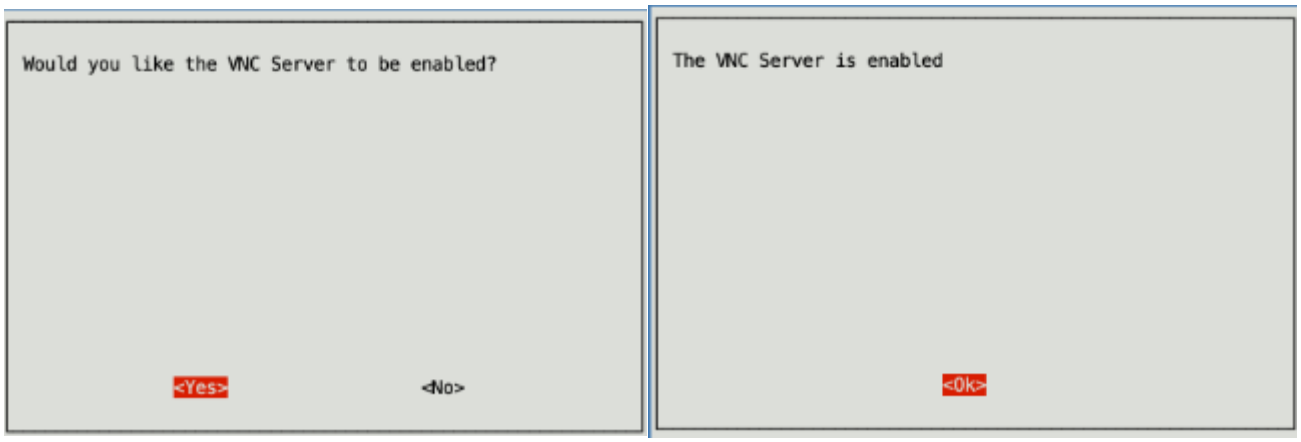
a) Omogući VNC u podešavanjima - Konfiguracija Raspberry PI - Interfejsi



b) Drugi način da se omogući VNC koristi terminal, unesite komandu

```
sudo raspi-config
```

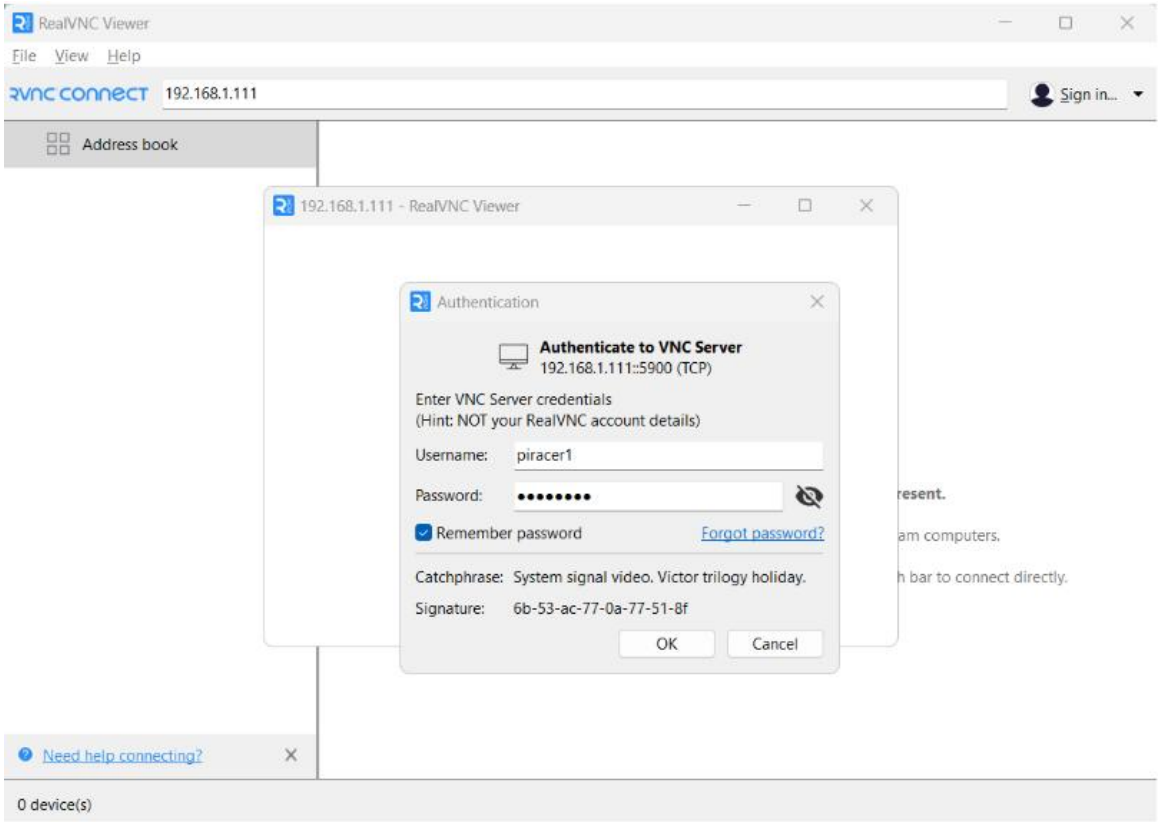




2.2 Instalirajte VNC Viever

Moraćete da instalirate VNC Viever na računaru, tako da možete da se povežete sa vama Raspberry Pi. Na raspolaganju je veliki broj gledalaca, ali najlakše je podesiti Real **VNC Viever**. Možete preuzeti Vindovs i Mac instalatere odavde: https://www.realvnc.com/en/connect/download/viewer/

Otvorite Real VNC Viever i unesite Piracer IP adresu (pročitajte ga sa ekrana automobila)



3. POČNITE VOZITI I PRIKUPLJAJTE PODATKE

Pokrenite svoj automobil

```
cd ~/mycar
python manage.py drive
```

Ova skripta će pokrenuti pogonsku petlju u vašem automobilu koja uključuje deo koji je veb server za vas da kontrolišete svoj automobil. Sada možete da kontrolišete svoj automobil iz veb pretraživača OS-a na automobilu na URL-u: <host_ime_vašeg_automobila.local>:8887

Otvorite pregledač i povežite se sa DonkeyCar Monitor-om na **localhost:8887**

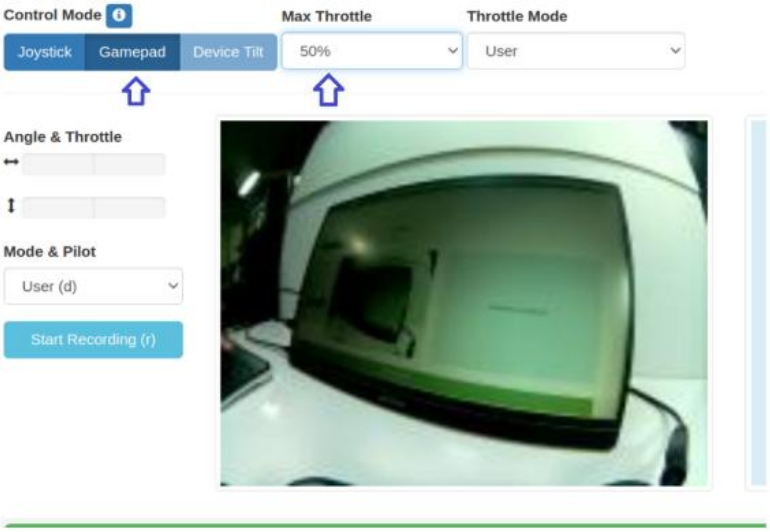
NAPOMENA: Kada se automobil pokrene, fascikla kreirana pod /mycar/data pod nazivom tub_#_date za čuvanje podataka za sesiju. Podaci se prikupljaju kada je gas uključen to jest kada se automobil kreće. Da biste prikupili nov skup podataka, najbolje je da zaustavite " python manage.py drive" koristeći **Contole + C** i ponovo pokrenete da biste započeli obuku sa novim /micar/data/tub.

Postoje 2 opcije vožnje

1. Koristite gamepad (preporučeno)

Nakon pokretanja automobila, otvorite veb pretraživač sa Raspberry Pi radne površine dok je automobil povezan sa monitorom.

Izaberite Gamepad i podesite gas na 50% da počne da vežba. Test skretanja i brzinu dok je automobil na Pi kutiji.

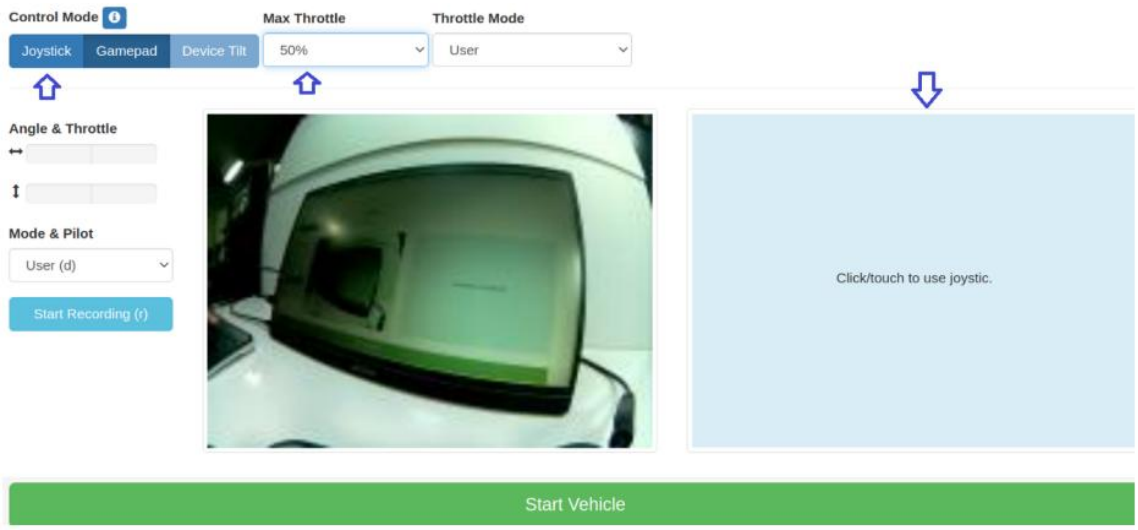


Kada ste spremni za vožnju na stazi, isključite monitor, postavite automobil na stazu i počnite da vozite. Počnite polako dok ne budete udobno rukovanje automobilom.

2. Koristite džojstik veb pretraživača

Moraćete da koristite laptop, tablet ili telefon povezan na isti WiFi (LAN mrežu). Nakon pokretanja automobila, koristite tablet ili telefon za povezivanje sa veb serverom automobila koristeći IP_automobila: 8887. Automobil bi trebalo da prikaže svoju IP adresu jet je usluga OLED displeja omogućena u prethodnom koraku.

Izaberite džojstik (upravljačka oblast džojstika je označena sa **click/touch to use joystic).** Koristite područje džojstika za vožnju automobila prstom.



Kada završite sa vožnjom, zaustavite "python manage.py drive " koristeći **Contol + C**.

4. TRAIN DATA - KREIRAJTE TRENIRANI MODEL

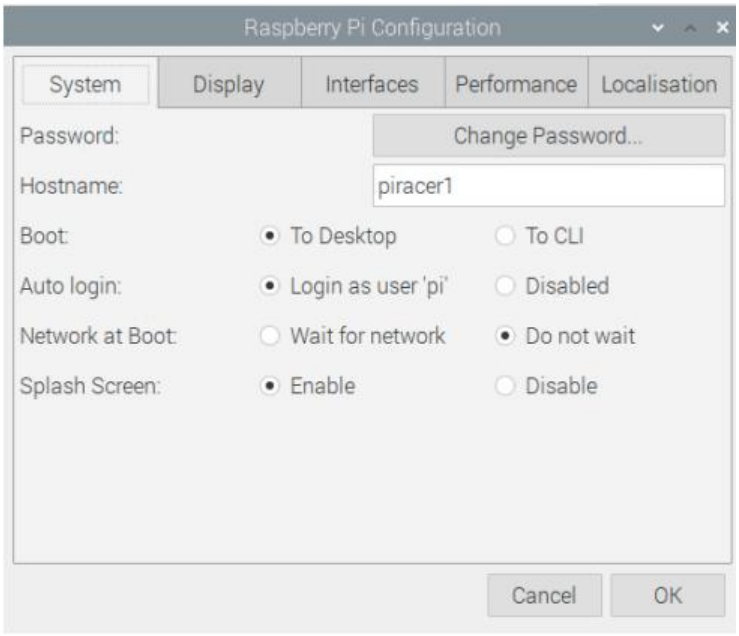
Lokacija podataka je /home/pi/mycar/data. Svi tub folderi u ovom folderu će se koristiti za kreiranje vašeg modela. Uklonite sve što ne želite da bude uključeno. Možete trenirati podatke sa Raspberry Pi ili sa AWS virtuelnom mašinom.

1. Treniraj sa Raspberry Pi

Podaci za treniranje:
U novoj terminal sesiji na vašem računaru koristite rsync da kopirate fasciklu automobila iz Raspberry Pi

```
rsync -rv --progress --partial <hostname>@<your_pi_ip_address>:~/mycar/data/
~/mycar/data/
npr:
rsync -rv --progress --partial piracer1@192.168.1.111:~/mycar/data/ ~/mycar/data/
```

Naziv hostname možete pronaći u Raspberry Pi konfiguraciji:



Trenirajte Model:
U istom terminalu sada možete pokrenuti skriptu za treniranje na najnovijem tub folderu, tako što ćete proslediti putanju do tub foldera kao argument. Opciono možete proslediti maske putanje, kao što su ./data/* ili ./data/tub_?_17-08-28 kako biste prosledili više tub foldera.
Na primer:

```
python ~/mycar/manage.py train --tub <tub folder names comma separated> --model
./models/mypilot.h5
npr.:
python ~/mycar/manage.py train --tub ./data/tub_1_24-04-06 --model
./models/mypilot.h5
```

Trebaće dosta vremena za treniranje modela, pa budite strpljivi. Nakon treninga, možete preuzeti model, potrebno je da kopirate modul na vaš Raspberry Pi i testirate ga.

```
rsinc -rv --shov-progress --partial ~/mycar/models/
<hostname>@<your_ip_address>:~/mycar/models/
itd:
rsync -rv --show-progress --partial ~/mycar/models/
piracer1@192.168.1.111:~/mycar/models/
```

5. UČITAJTE MODEL I VOZITE AUTONOMNO

Pokrenite svoj automobil sa modelom

```
cd ~/mycar
python manage.py drive --model ~/mycar/models/mypilot.h5
```

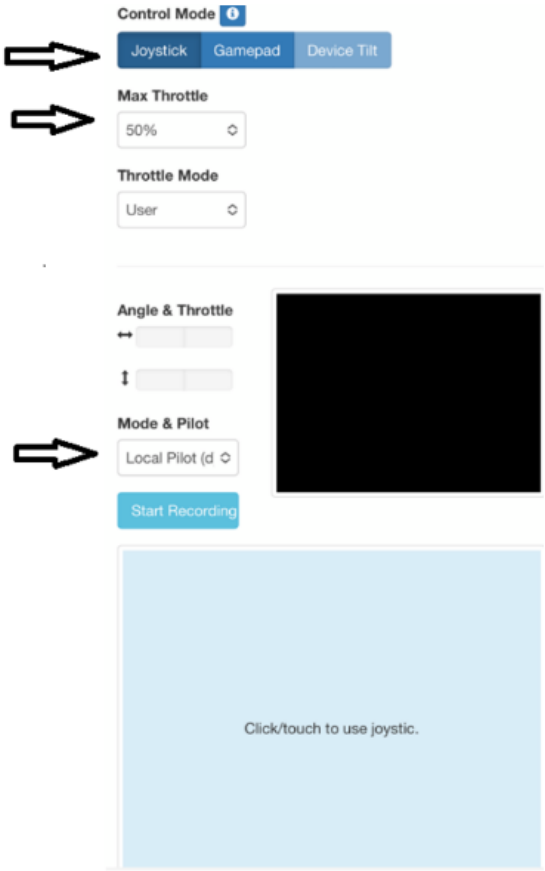
Za autonomnu vožnju ili upravljanje, koristite džojstik preko veb pretraživača (npr. Chrome) na drugom uređaju (npr. laptopu, računaru…).

Moraćete da koristite računar, laptop, tablet ili telefon povezan na isti WiFi to jest na istu lokalnu mrežu. Na tom uređaju je potrebno povezati se sa veb serverom automobila koristeći IPautomobila: 8887 (npr. 192.168.1.11:8887). Automobil bi trebalo da prikaže svoju IP adresu na svom OLED displeju jer je omogućena u prethodnom koraku.

Autonomna vožnja

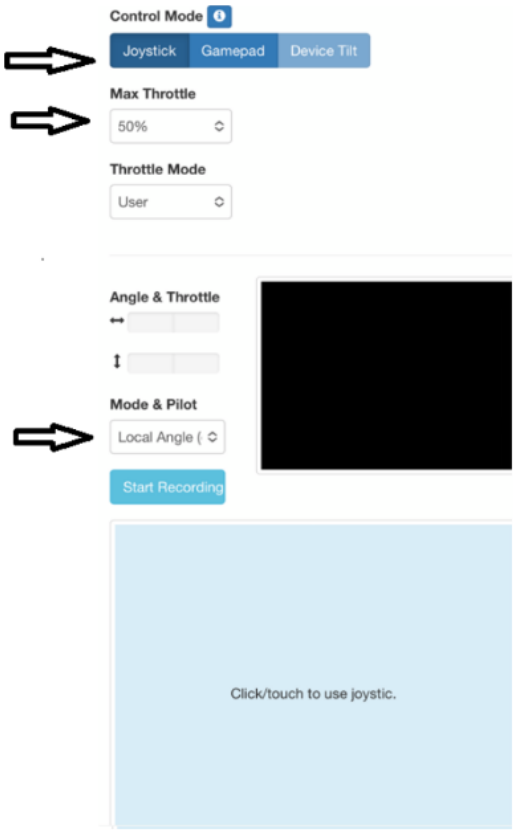
NAPOMENA: Čim se izabere **lokalni pilot**, automobil će početi autonomnu vožnju

- **Režim kontrole:** Izaberite ** Džojstik ** (upravljačka oblast džojstika je označena sa **click/touch to use joystic**)
- **Max Throttle:** Podesite Max throttle na oko 50% da automobil ne bi išao prebrzo
- **Mode & Pilot:** Izaberite Local Pilot (Local Pilot je autonoman)



Autonomno upravljanje

- **Režim kontrole:** Izaberite ** Džojstik ** (upravljačka oblast džojstika je označena sa **click/touch to use joystic**)
- **Max Throttle:** Podesite Max throttle na oko 50% da automobil ne bi išao prebrzo
- **Mode & Pilot:** Izaberite Local Angle (Local Angle je automatsko upravljanje automobilom dok brzinu regulišete džojstikom)



Brži autonomni model?

Koristite svoj model autonomne vožnje kako biste prikupili nove podatke sa većom brzinom. Možete iskoristiti Local Angle režim tako da dok automobil upravlja autonomno vozite maksimalnom mogućom brzinom.

II. AI SA AR / VR

6. UVOD

Meta Quest 3 označava ključni pomak od čiste virtuelne stvarnosti (VR) do mešovite stvarnosti visokog kvaliteta (MR). Koristeći prolaz u boji visoke rezolucije i znatno snažniji Snapdragon KSR2 Gen 2 čipset, omogućava digitalnom sadržaju da neprimetno koegzistira sa fizičkim svetom. Sa svojom tanjom optikom "palačinke" i 4K+ Infinite Display, trenutno je najpristupačniji uređaj za potrošače i programere.

6.1 Uvod u Meta Quest 3 i mešovitu stvarnost

Meta Quest 3 koristi tehnologiju Color Passthrough visoke rezolucije. Za razliku od tradicionalnog VR-a, gde je svet potpuno virtuelni, **mešovita stvarnost (MR)** koristi ugrađene kamere za projektovanje stvarnog okruženja, a zatim prekriva VebGL elemente na vrhu. Vaš kod koristi immersive-ar mod, koji je srž ovog prostornog iskustva.

Arhitektura sistema (Veb Stack)

Aplikacija funkcioniše kao "sendvič" slojeva:

- **Sloj 1 (hardver):** Quest 3 senzori, dubinski projektori i RGB kamere.
- **Sloj 2 (Browser):** Meta Quest Browser (zasnovan na Chromium-u sa podrškom za VebXR).
- **Sloj 3 (API):** VebXR Device API, koji deluje kao most između hardvera i koda.
- **Sloj 4 (logika):** Vaša JavaScript logika koja koristi Three.js za renderovanje i TensorFlow.js za viziju.

7. VEBXR: ALTERNATIVA ZASNOVANA NA PRETRAŽIVAČU

VebXR omogućava programerima da kreiraju impresivna iskustva koja se pokreću direktno u pretraživaču Meta Quest. Izgrađen je na standardnim veb tehnologijama, čineći "VR na vebu" lakim za pristup kao veb sajt.

Zašto izabrati VebXR?

- Frictionless Access: Korisnici ne moraju da preuzimaju velike datoteke iz Meta Store-a; oni jednostavno kliknu na URL i pritisnu "Enter VR."
- Frameworks: Koristi moćne JavaScript biblioteke kao što su Three.js, A-Frame (zasnovan na HTML-u) ili Babylon.js.
- Cross-Platform: Jedna VebXR aplikacija često može da radi na Quest 3, Android telefonu (AR režim) ili desktop računaru.

7.1 Analiza biblioteke

Three.js (Render motor)

Three.js se bavi teškim podizanjem VebGL-a. Upravlja scenom, kamerom i rendovanjem. U vašem kodu, renderer je postavljen na alfa: true, što je od vitalnog značaja za AR jer omogućava "praznim" delovima pretraživača da postanu prozor u stvarni svet.

TensorFlow.js & COCO-SSD

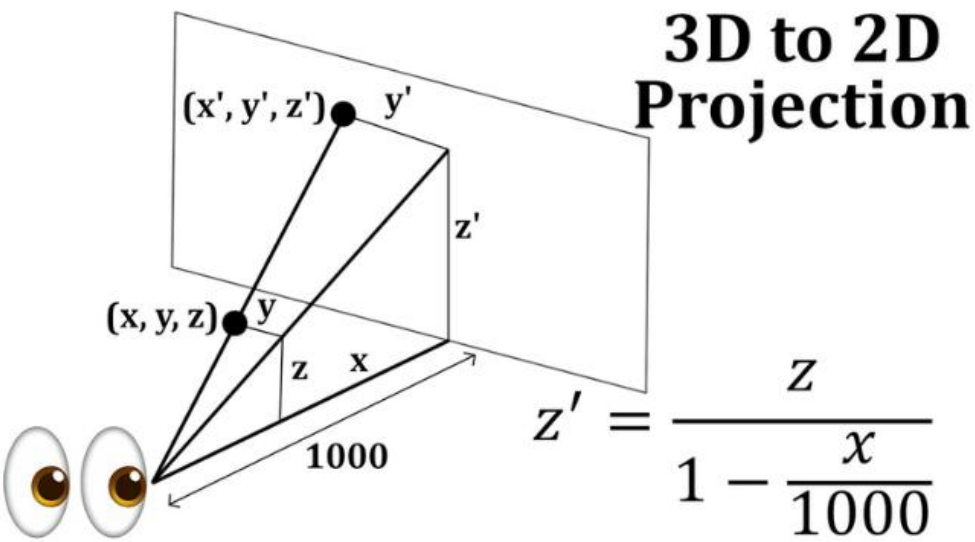
COCO-SSD (Common Objects in Context - Single Shot MultiBox Detector) je model obučen da prepozna 80 klasa objekata. Lepota TF.js je **njegov VebGL backend**, koji osigurava da se AI proračuni izvode na GPU-u Quest-a, a ne na CPU-u, sprečavajući pregrevanje uređaja.

7.2 Matematička projekcija (2D do 3D)

Ovo je najtehničkiji deo scenarija. AI model vraća bbox (bounding box) u pikselima (npr. x = 100, i = 200).

Funkcija detectionTo3D vrši un-projekciju:

- **Normalizacija**: Pretvara koordinate ekrana u rasponu od -1 do +1.
- **FOV kalkulacija**: faktoriše u vidnom polju kamere.
- **Vektorski zrak**: Kreira vektor pravca od položaja glave korisnika prema objektu.
- **Pozicioniranje**: Postavlja 3D oznaku na taj vektor na definisanoj udaljenosti (DIST).



7.3 Implementacija Hit-Testing

Hit-testing omogućava aplikaciji da "ispaljuje" nevidljivi zrak (raycast) u stvarni prostor kako bi otkrio ukrštanja sa podovima ili stolovima. Kada ***hitTestSource*** vrati rezultat, kursor (zeleni prsten) se prebacuje na tu pozu (položaj i orijentacija).

7.4 AI Pipeline: Detekcija i označavanje

U kodu, detekcija se ne vrši svaki kadar (što bi izazvalo seckanje), već svakih 2000ms (DETECT_INTERVAL_MS).

- **Label Sprite**: Pošto Three.js ne može da renderuje standardne HTML fontove direktno u 3D sceni, koristimo CanvasTexture. Tekst "crtamo" na nevidljivom 2D HTML platnu i primenjujemo ga kao teksturu na 3D Sprite koji je uvek okrenut prema korisniku (bilbordovanje).

7.5 Neophodnost HTTPS-a

VebXR i pristup kameri (getUserMedia) su klasifikovani kao "moćne funkcije" od strane proizvođača pretraživača. Oni će raditi samo u sigurnom kontekstu.

- **Izuzetak lokalnog hosta**: Možete testirati na računaru koristeći http://localhost, ali čim pokušate da pristupite tom serveru sa naočara Quest 3, pretraživač će blokirati XR funkcije jer vidi nesigurnu mrežnu IP adresu (npr. http://192.168.1.10).
- **Rešenje**: Morate koristiti **uslugu tunelovanja** ili **siguran hosting**.

7.6 Meta Quest Link & Developer Mode

Da biste efikasno pokrenuli i otklonili greške u kodu, vaš Quest 3 mora biti prepoznat kao uređaj za programere.

Aktivacija korak po korak:

- **Nalog programera**: Registrujte se na **dashboard.oculus.com**. Moraćete da kreirate "Organizaciju" (to može biti bilo koje ime).
- Mobilna aplikacija: Otvorite aplikaciju Meta Quest na svom telefonu, idite na **Menu > Devices > Headset Settings > Developer Mode** i uključite ga.
- **Link**: Povežite Quest 3 sa računarom pomoću visokokvalitetnog USB-C 3.0 kabla ili preko Air Link (velike brzine Wi-Fi 6).

7.7 Three.js Foundation (The Boilerplate)

Pre ulaska u AR, moramo inicijalizovati standardno 3D okruženje. Međutim, za Quest 3, dva podešavanja su obavezna:

- **Alfa i Antialias**:

```javascript
const renderer = new THREE.WebGLRenderer({ antialias: true, alpha: true });
```

- **XR aktivacija**:

```
enderer.xr.enabled = true;
```

Ovo upućuje Three.js da slušaj podatke o praćenju glave Quest-a (6DOF) i automatski ga primenjuju na objekat "kamere".

7.8 Upravljanje AR sesijom (životni ciklus)

Dugme "Enter AR" pokreće **navigator.xr.requestSession**. Ovo je mesto gde definišemo koje "supermoći" Quest 3 hardvera trebaju našoj aplikaciji.

Potrebne i opcione karakteristike:

- **local-floor**: Ovo govori Quest-u da postavi koordinatu Y=0 na stvarnom nivou fizičkog poda.
- **hit-test**: Uključuje mogućnost raicast na geometriju u stvarnom svetu.
- **plane-detection**: Zahteva "semantičke" podatke (znajući koja mreža je 'sto' u odnosu na 'zid').

```JavaScript
const session = await navigator.xr.requestSession('immersive-ar', {
    requiredFeatures: ['local-floor'],
    optionalFeatures: ['hit-test', 'plane-detection']
});
```

8. AR + AI DETEKCIJA OBJEKATA DEMO

8.1 Pregled projekta

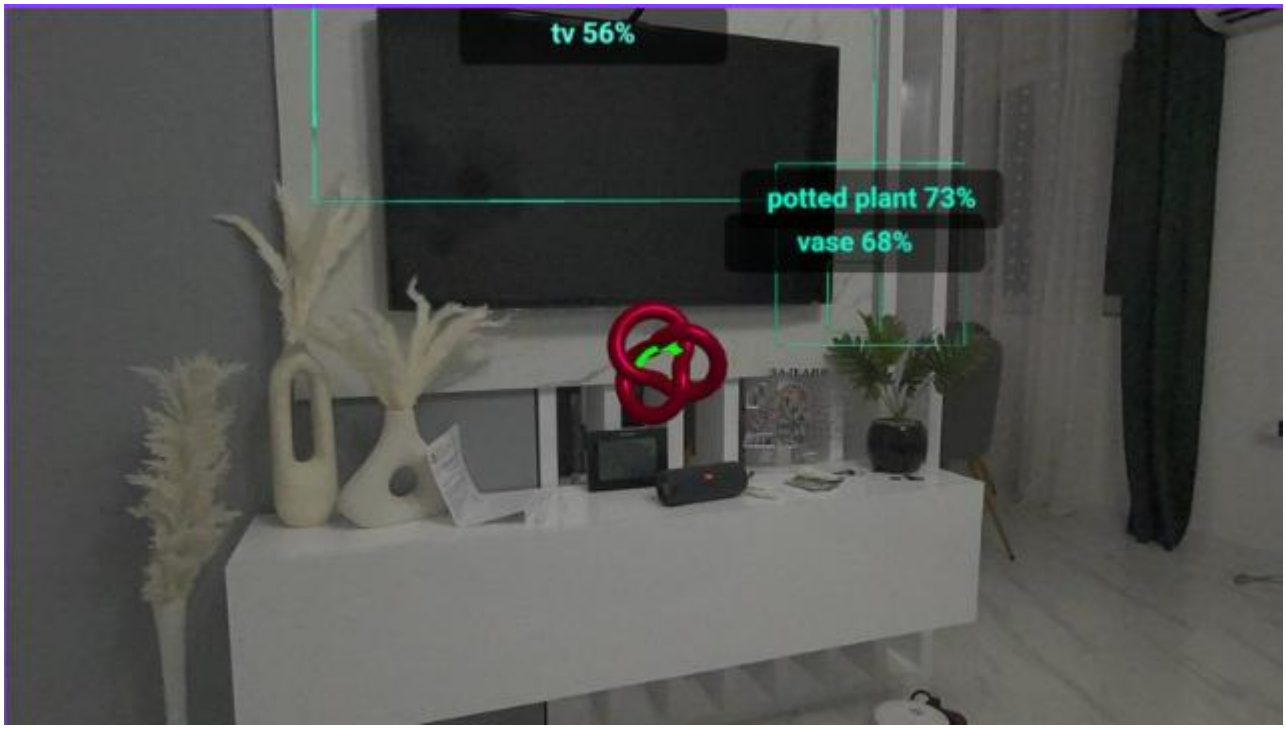
Ovo je aplikacija VebXR Augmented Realiti za Meta Quest 3 koja kombinuje:

- **Passthrough AR** — stvarna soba je vidljiva kroz kamere za slušalice
- **Interakcija hit-test** — virtuelni kursor koji se lepi na površine u stvarnom svetu
- **AI detekcija** objekata — TensorFlow.js prepoznaje objekte u feedu kamere i prikazuje oznake u 3D prostoru
- **Razumevanje scene** — WebXR Plane Detection vizualizuje otkrivene površine (pod, zidovi, sto)

URL uživo: http://92.113.18.92/

Pojedinačna datoteka:

index.html



8.2 Arhitektura projekta

index.html (sve-u-jednom)

├── HTML → platno + dugme UI + skriveni video element

├── CSS → transparentna pozadina, preklapanje UI

└── JavaScript

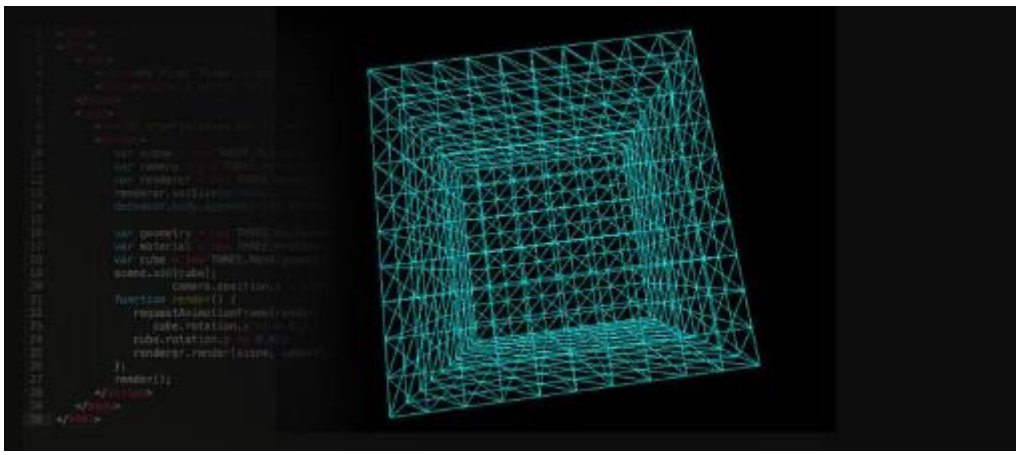
├── Three.js → 3D rendering engine (scena, kamera, materijali)

├── VebXR API → AR sesija, hit-test, detekcija prostora

└── TF.js → AI podrška (coco-ssd model)

Zašto čisti Three.js (a ne A-frame)?

Tokom razvoja otkriveno je da A-Frame ima svoju unutrašnju petlju za renderovanje koja je u sukobu sa eksplicitnom **imerzivnom-ar** VebXR sesijom. A-Frame započinje **immersive-vr** sesiju (neprozirno, bez prolaza) umesto **immersive-ar**. Prelazak na čistu **Three.js** nam je dao direktnu kontrolu i nad tipom sesije i petljom rendera.



8.3 Tehnologija Stack

Tehnologija	Verzija	Uloge
Three.js	0.157.0	3D rendering, geometrija, materijali
VebXR API uređaja	Izvorni pretraživač	AR sesija, hit-test, detekcija prostora
TensorFlow.js	4.15.0	Mehanizam za korišćenje ML-a u pretraživaču
COCO-SSD	2.2.3	Unapred obučeni model detekcije objekata
getUserMedia API	Izvorni pretraživač	Video strim kamere za analizu TF.js

8.4 Tok aplikacije

Učitavanje stranica

→ TF.js model (coco-SSD) se učitava asinhrono (~ 6MB)

→ Provera: navigator.xr.isSessionSupported('immersive-ar')

→ "Enter AR" dugme postaje aktivno

Korisnik klikne na "Enter AR"

→ Zahtev: navigator.xr.requestSession('immersive-ar', {...})

→ Quest omogućava Passthrough (kamera vidljiva preko headset-a)

→ Istovremeno: getUserMedia () → strim kamere za TF.js

→ renderer.setAnimationLoop () se pokreće (XR render petlja)

Svaki kadar (~ 72fps na Quest 3)

- scene.background = null (obezbeđuje transparentnost)
- Hit-test → ažurira poziciju kursora
- Plane Detection → ažurira vizuelizaciju površine
- Svakih 3s → runDetection() → AI obrada
- renderer.render(scene, camera)

8.5 Detaljno objašnjenje koda

1. HTML Struktura (linije 1–78)

```
HTML
<video id="tfvideo" playsinline muted autoplay></video>
```

Skriveni element <**video**> koji prima stream ***getUserMedia***(). TensorFlow.js ga koristi kao ulaz za AI obradu — nikada se ne prikazuje korisniku, već se samo analizira.

```
html
<div id="ui"> ... <button id="ar-button"> </div>
```

Sloj prekrivača korisničkog interfejsa koji lebdi iznad **Three.js** platna. ***pointer-events: none*** on the container but ***pointer-events: all*** on the button only — so the overlay doesn't block 3D interactions.

2. Three.js Inicijalizacija (linije 84–109)

```javascript
const renderer = new THREE.WebGLRenderer({ antialias: true, alpha: true });
renderer.xr.enabled = true;
```

alpha: true kreira WebGL platno sa transparentnim alfa kanalom — ovo je preduslov za prolaz. ***kstr.enabled = true*** aktivira **ugrađenu VebXR integraciju Three.js-a.**

```javascript
const camera = new THREE.PerspectiveCamera(70, aspect, 0.01, 20);
scene.add(camera);
```

Kamera se dodaje **na scenu**. Ovo je važno jer objekti vezani za kameru (potomci entiteti) moraju biti u hijerarhiji scene.

```javascript
const cursorGeo = new THREE.RingGeometry(0.04, 0.06, 32);
cursorGeo.rotateX(-Math.PI / 2);
```

Geometrija prstena se rotira -90 ° na X osi tako da leži ravno horizontalno na detektovanim površinama. Bez ove rotacije stajao bi vertikalno.

3. makeLabel() — Text Sprite (linije 113–138)

```javascript
function makeLabel(text, color, bgColor) {
  const canvas = document.createElement('canvas'); // 512x128 px
  // Draws rounded rect + text
  const texture = new THREE.CanvasTexture(canvas);
  const sprite  = new THREE.Sprite(material);
  sprite.scale.set(0.6, 0.15, 1); // 0.6m wide in 3D space
  return sprite;
}
```

3. Sprajt je objekat koji je uvek okrenut prema kameri (**bilbord**) — idealan za oznake u 3D prostoru. Nacrtan je na HTML platnu, pretvoren u ***CanvasTexture*** i primenjen na ***SpriteMaterial***.

depthTest: false osigurava da je oznaka uvek vidljiva čak i ako je geometrijski "iza" drugog 3D objekta.

4. detectionTo3D() — 2D → 3D projekcija (linije 140–163)

Ovo je matematičko jezgro AI-AR integracije.

```javascript
// Normalize bbox center to [-0.5, 0.5]
const nx = (bbox.x + bbox.width/2)  / videoWidth  - 0.5;
const ny = (bbox.y + bbox.height/2) / videoHeight - 0.5;
// Apply camera FOV (70° horizontal)
const fovH = 70 * Math.PI / 180;
const dir  = new THREE.Vector3(
  Math.tan(nx * fovH),
  Math.tan(-ny * fovV), // flip Y (image top-down, WebGL bottom-up)
  -1                    // forward in camera space (Three.js uses -Z)
).normalize();
// Rotate into world space using the current XR camera orientation
dir.applyQuaternion(camera.quaternion);
// World position = camera position + direction × distance
return camera.position.clone().addScaledVector(dir, placeDist);
```

*Primer: Ako AI otkrije stolicu u levoj trećini video zapisa, **nx ≈ -0.17**. Ovo se pretvara u negativan ugao (levo od ose pogleda). Oznaka je postavljena 1,8 m ispred kamere u tom pravcu.*

5. makeBox3D() i bbox2dSize3D() — 3D Bounding Boxes (linije 165–182)

```javascript
function makeBox3D(w, h, colorHex) {
  const edges = new THREE.EdgesGeometry(new THREE.BoxGeometry(w, h, 0.01));
  return new THREE.LineSegments(edges, material);
}
```

EdgesGeometry + LineSegments *renderuje samo ivice pravougaonika — efekat tanke žičane granice bez ispunjene površine.*

```javascript
function bbox2dSize3D(bboxW, bboxH, videoW, videoH, dist) {
  const fovH   = 70 * Math.PI / 180;
  const totalW = 2 * dist * Math.tan(fovH / 2); // total visible width at given dist
  return {
    w: (bboxW / videoW) * totalW,  // bbox fraction → meters
    h: (bboxH / videoH) * totalH
  };
}
```

TotalW *formula je standardna perspektivna projekcija:* ***na udaljenosti d****, vidljiva širina zavisi od FOV. Iz ovoga izvodimo koliko metara odgovara pikselima graničnog okvira.*

6. runDetection() — AI Inference Pipeline (linije 205–247)

```javascript
async function runDetection() {
  const predictions = await cocoModel.detect(tfVideo);
  // Clear old labels and boxes
  activeLabels.forEach(({ sprite, box }) => { scene.remove(sprite); scene.remove(box); });
  activeLabels = [];
  predictions.forEach(pred => {
    if (pred.score < 0.5) return; // Confidence threshold: 50%
    // ...create sprite + box...
    activeLabels.push({ sprite, box, expireAt: Date.now() + 4000 });
  });
}
```

cocoModel.detect(tfVideo) *prihvata element <video> direktno — TF.js interno čita podatke piksela iz trenutnog video okvira.*

pred.bbox *format:* ***[x, y, širina, visina]*** *u pikselima.*

Svaka detekcija stvara ***JS (sprite, box)*** *par koji egzistira 4 sekunde.*

7. WebXR Session — Ključni detalji (linije 274–285)

```javascript
const session = await navigator.xr.requestSession('immersive-ar', {
  requiredFeatures: ['local-floor'],
  optionalFeatures: ['hit-test', 'plane-detection']
});
renderer.xr.setReferenceSpaceType('local-floor');
await renderer.xr.setSession(session);
```

Zašto **immersive-ar**, a ne **immersive-vr**?

- **Immersive-VR** = neprozirna crna pozadina (VR iskustvo sa kacigom)
- **Immersive-AR** = prolazna kamera + virtuelni sadržaj prekriven na vrhu

local-floor referentni prostor popravlja koordinatni sistem na podu prostorije — **y=0** je na nivou poda.

Hit-Test i **plane-detection** su opcioni jer nisu podržani na svim uređajima i verzijama pretraživača — proglašavamo ih opcionim kako bi sprečili grešku sesije ako nisu dostupni

8. Render Loop (linije 335–395)

```javascript
renderer.setAnimationLoop(function(time, frame) {
  scene.background = null;              // Ensures transparency every frame
  renderer.setClearColor(0x000000, 0); // Alpha = 0 (fully transparent)
  // ...hit-test, plane detection, AI...
  renderer.render(scene, camera);
});
```

Zašto **setAnimationLoop** i ne **requestAnimationFrame**?

Three.js XR render petlja mora biti integrisana sa povratnim pozivom WebXR okvira. **renderer.setAnimationLoop ()** automatski se vezuje za XR sesiju kada **renderer.xr.enabled = true**. Alternativa (ručno pozivanje **session.requestAnimationFrame(**)) zahteva ručno pozivanje **renderer.render(),** ali rizikuje da nedostaju ispravne matrice XR prikaza koje **Three.js** primenjuju interno.

scene.background = null mora biti pozvan **svaki frejm,** jer Three.js može interno resetovati ovu vrednost tokom određenih operacija.

9. Detekcija prostora (linije 355–378)

```javascript
session.detectedPlanes.forEach(plane => {
  if (!detectedPlanes.has(plane)) {
    // New surface detected — create a mesh
    const label = plane.semanticLabel; // 'floor', 'wall', 'table'...
    const mesh = new THREE.Mesh(PlaneGeometry, transparentMaterial);
    scene.add(mesh);
    detectedPlanes.set(plane, mesh); // store reference
  }
  // Every frame, update the surface pose
  const pose = frame.getPose(plane.planeSpace, xrRefSpace);
  mesh.position.copy(pose.transform.position);
  mesh.quaternion.copy(pose.transform.orientation);
});
```

plane.planeSpace je XRSpace koji prati fizičku površinu. **frame.getPose()** vraća svoju poziciju u izabranom referentnom prostoru.

detectedPlanes (Map<XRPlane, TRI. Mesh>) sprečava stvaranje duplih mreža za istu površinu preko okvira.

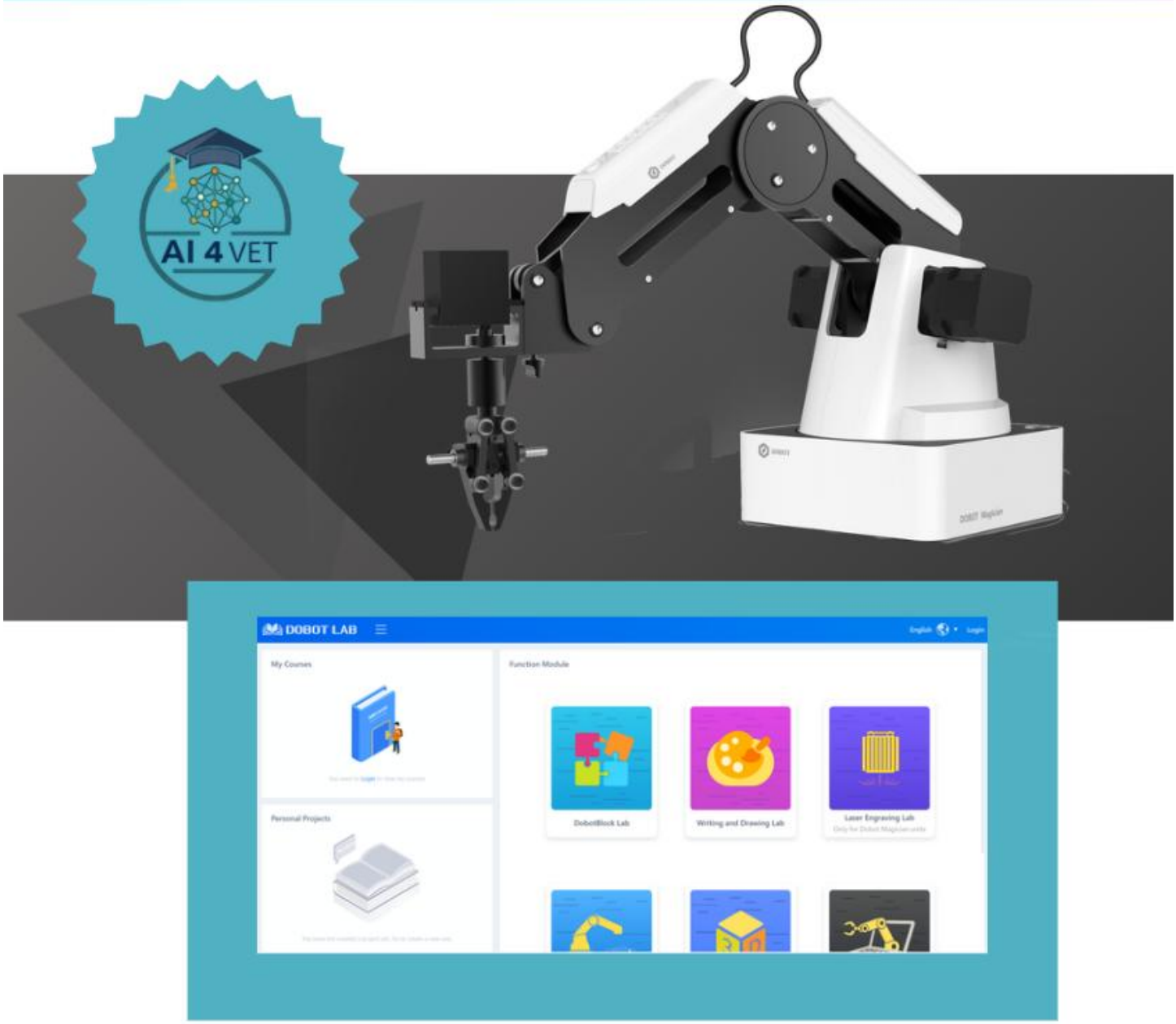
8.6 Poznata ograničenja

Ograničenje	Obrazloženje
AI oznake nisu savršeno poravnate sa objektom	Quest passthrough kamere i getUserMedia isporučuju različite streamove sa različitim FOV i optičkom kalibracijom
Detekcija prostora zahteva da se Quest Room Setup završi	headset mora prethodno skenirati sobu
getUserMedia može biti odbijen na nekim uređajima	Zavisi od dozvola pretraživača
AI obrada nije u realnom vremenu (radi svakih 2s)	Coco-SSD je relativno težak model; lakše alternative (MobileNet SSD) će dati veću brzinu obrade

8.7 Izvorni kod

https://github.com/bciric1/ARVR

http://vr.aiforvet.eu/



III. AI SA Dobotom

9. UPUTSTVO ZA INSTALACIJU DRAJVERA

Kada prvi put povežete Dobot, povežite Dobot sa računarom preko USB porta. Zatim, uključite Dobot, onda će sistem automatski prepoznati odgovarajući hardver i da će tražiti ispravan upravljački program i instalirati ga. Međutim, ako ne uspe da se instalira, možete ga ponovo instalirati ručno. Dijagram toka instalacije na sledeći način:

9.1 Preuzmite CH340 drajver paket i instalirajte ga

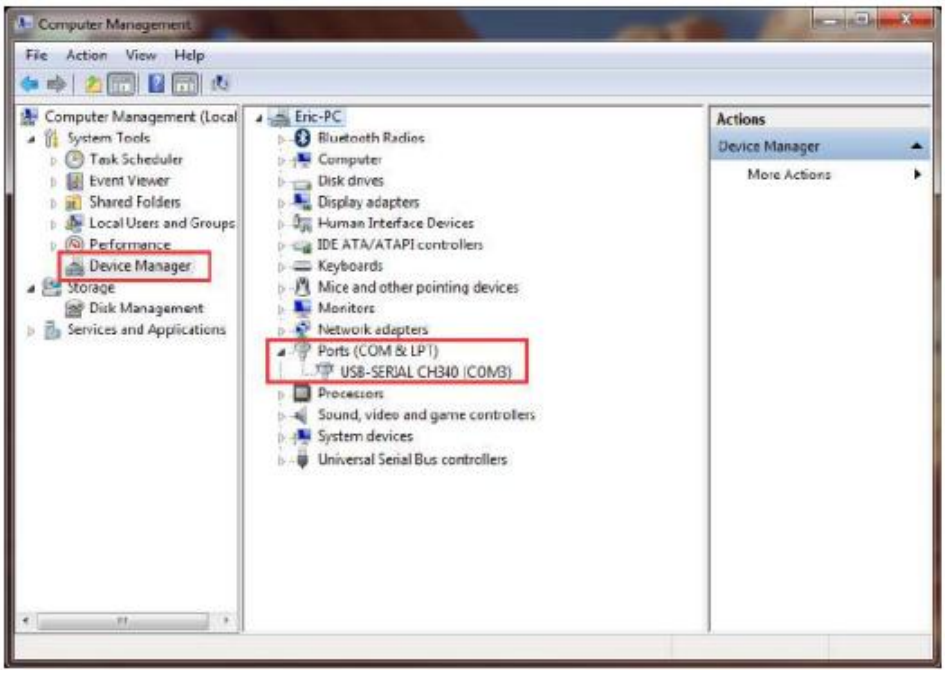
Postoje dve verzije drajvera zasnovane na Windows/Linux-u, pa vas molimo da izaberete odgovarajuću verziju za vaš operativni sistem za preuzimanje. Vozač se može nalaziti na adresi za preuzimanje:

http://www.dobot.cc/downloadcenter.html?sub_cat=70#sub-download

Nakon preuzimanja, raspakujte i instalirajte drajver

9.2 Proverite da li oprema može ispravno da radi u device manager-u

Otvorite menadžer uređaja i ako možete da pronađete odgovarajući COM port "USBSERIAL CH340", onda pokazuje da je drajver uspešno instaliran. Ispravna instalacija je prikazana ispod:



10. UPUTSTVO ZA UPOTREBU DOBOTSTUDIO

Softver koji koristi Dobot Magician je DobotStudio, a najnoviju verziju možete preuzeti sa zvaničnog sajta: http://www.dobot.cc/downloadcenter.html?sub_cat=70#sub-download
Nakon što se datoteka uspešno preuzme, raspakujte i dvaput kliknite na DobotStudio.exe.

Izaberite odgovarajući Dobot serijski port, u gornjem levom uglu DobotStudio, i kliknite na "Connect". Nakon uspešnog povezivanja, biće prikazan "Disconnect". Kada je Dobot povezan, koordinatni parametri će biti ažurirani na desnoj strani interfejsa

Postoji osam modula na glavnom softverskom interfejsu:

- Teaching & Playback: Sistem za podučavanje Dobota kako da se kreće. Omogućava Dobotu da postigne snimljene pokrete ručnom kontrolom.
- Write & Draw: Kontroliše Dobot za pisanje, crtanje ili lasersko graviranje.
- DobotBlockly：Uči osnovno programiranje kroz interfejs slagalice. Intuitivan i lak za razumevanje.
- Script: Koristite skriptni jezik za kontrolu Dobota.
- LeapMotion: Kontrolišite Dobot gestom.
- Mouse: Kontrolišite Dobot pomoću miša.
- LaserEngraving: Gravira slike, oblike i reči kroz bitmapu sa Dobotom.
- Add More: Dodajte još više funkcija za Dobot!

10.1 Linearni režim

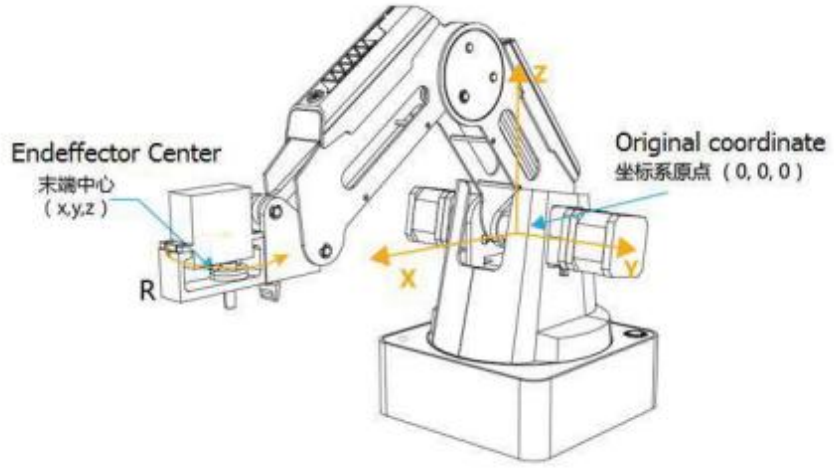
Na osnovu koordinatnog sistema tela osa X, Y, Z sa poreklom u centru tri motora. X, Y, Z je koordinata centra krajnje platforme, a pravac X je okomit na bazu napred, Y je okomit na bazu prema levoj strani, a Z je vertikalno prema gore. R označava rotaciju servo zgloba u odnosu na koordinatni okvir (suprotno smeru kazaljke na satu je pozitivan pravac).

(1) Kliknite X+, X- i Dobot će se kretati duž X u negativnom ili pozitivnom smeru;

(2) Kliknite Y+, Y- i Dobot će se kretati duž Y u negativnom ili pozitivnom smeru;

(3) Kliknite na Z +, Z- i Dobot će se kretati duž Z u negativnom ili pozitivnom smeru;

(4) Kliknite R +, R- i Dobot će se kretati duž R u negativnom ili pozitivnom smeru.



10.2 Jog režim

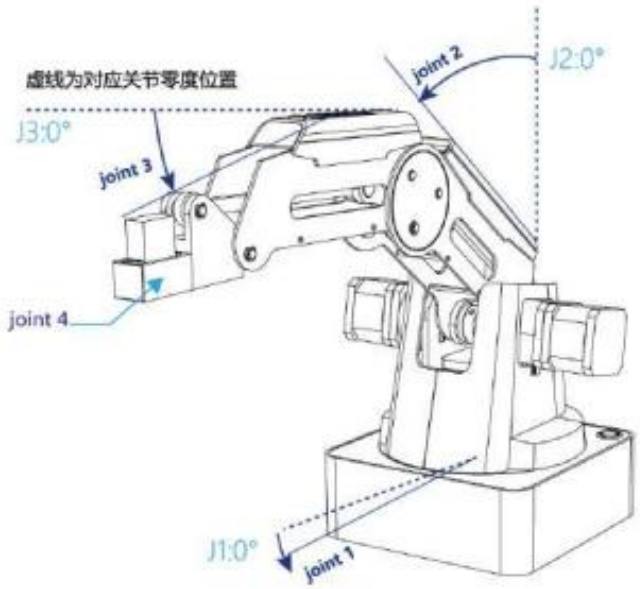
Ovaj pokret je usmeren na jednu osu. Držite pritisnuto dugme, a odgovarajuća osa će se kretati nezavisno. Kada je osa maksimalno napolje, zglob će prestati da se kreće. Svaka osa ima smeru suprotnom od kazaljke na satu kao pozitivan pravac. Joint1，2，3，4 se odnose na bazu, zadnju ruku, podlakticu i servo respektivno.

(1) Kliknite Joint1+，Joint1- i kontroliše motor Dobot baze da rotira u negativnom ili pozitivnom smeru;

(2) Kliknite Joint2+，Joint2- i kontrola motor zadnje ruke da rotira u negativnom ili pozitivnom smeru;

(3) Kliknite Joint3+，Joint3- i kontrola motora podlaktice da rotira u negativnom ili pozitivnom smeru;

(4) Kliknite Joint4 +，Joint4- i kontrola servor da rotira u negativnom ili pozitivnom smeru; Među ovim, opseg rotacije Joint4 je ±150 °

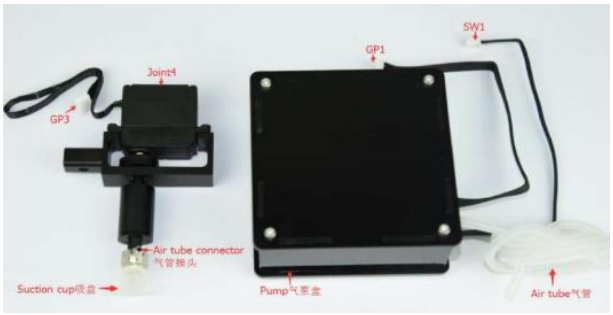


11. AKTUATORI NA DOBOTU

Ovde ćemo naučiti kako da usisamo ili zgrabimo jednostavne predmete koristeći funkciju Teaching & Playback. Zato što moramo da koristimo komplet pumpe za vazduh za usisnu čašu i komplet hvataljke, mi ćemo predstaviti ova dva kompleta zajedno.

11.1 Komplet pumpe za vazduh

Podrazumevana instalacija Dobot Magician je usisnica. Kutija pumpe i komplet za usisnicu prikazani su ispod:



11.2 Pneumatska hvataljka

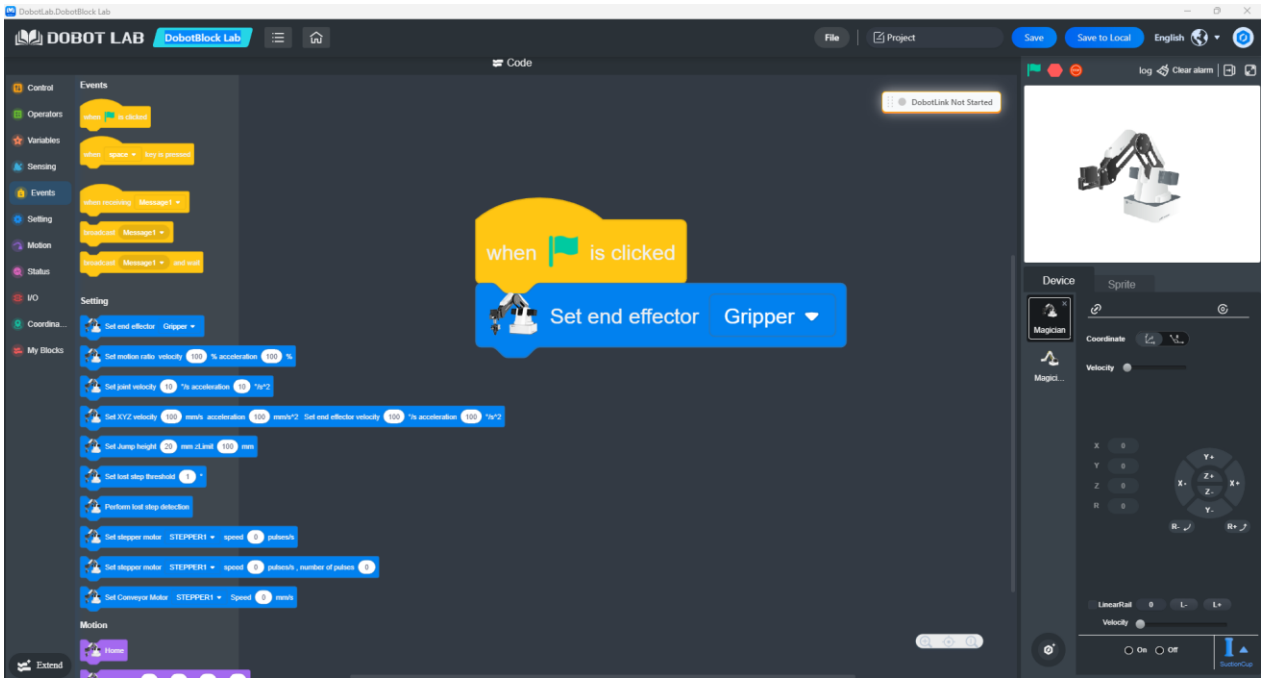
Pneumatski pribor su prikazani na sledećoj slici:

12. PROJEKAT: AUTOMATIZOVANI SISTEM ZA KLASIFIKACIJU I SORTIRANJE OTPADA POMOĆU DOBOT MAGICIAN

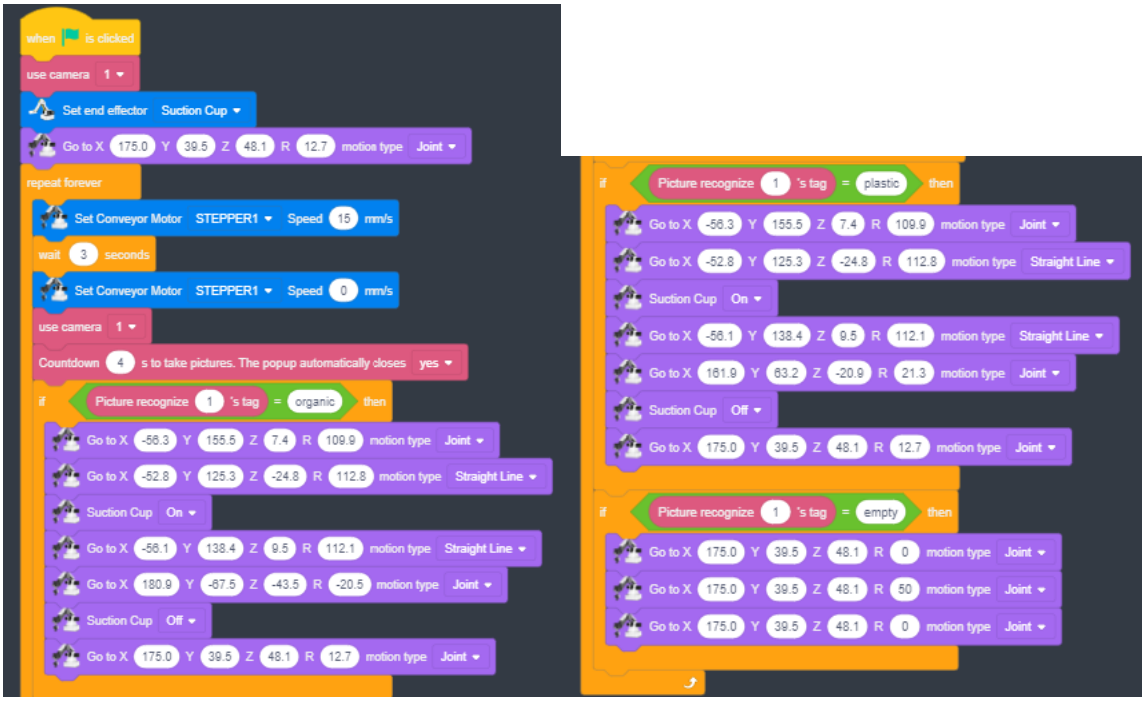
Dobot Blockly je programska platforma zasnovana na Google Blockly. U njemu korisnici mogu programirati kroz format slagalice, koji je jednostavan i lako razumljiv. Takođe, korisnici mogu koristiti integrisani API Dobota u bilo kom trenutku.

12.1 Blockly interfejs

Otvorite DobotStudio i kliknite na DobotBlock Lab:



12.2 Blok kod



12.3 Inicijalizacija i definisanje problema

Kontekst i motivacija

U modernoj industriji i ekologiji, ručno sortiranje otpada je spor, neefikasan i često opasan proces. Cilj ovog projekta je da se stvori autonomni sistem koji koristi **kompjuterski vid** i **robotsku manipulaciju** za prepoznavanje i fizičko odvajanje različitih materijala (u ovom slučaju organskog otpada i plastike).

Definisanje problema

Primarni izazov leži u sinhronizaciji tri nezavisna podsistema:

1. **Transportni sistem:** Pokretna traka koja donosi predmete u radni prostor.
2. **Senzorski sistem:** Kamera koja mora da identifikuje tip predmeta u realnom vremenu.
3. **Sistem za aktiviranje:** Robotska ruka koja mora precizno izvršiti branje i postavljanje na osnovu povratnih informacija senzora.

12.4 Arhitektura rešenja

Rešenje je zasnovano na platformi **Dobot Magician** i programskom okruženju zasnovanom na blokovima (Blockli) koje integriše AI module za prepoznavanje slika.

Hardverske komponente

- **Dobot Magician Arm:** Visoko precizni 4-osni robot.
- **Usisnica:** Koristi se kao krajnji efektor za svoju svestranost u rukovanju različitih oblika.
- **USB kamera:** Montirana iznad trake za stabilan pogled odozgo na dole.
- **Transportna traka:** Kontroliše se preko koračnog motora povezanog sa Dobotom.

Softver Logika

Algoritam prati **iterativni model zatvorene petlje**: Start Belt -> Stop -> Capture -> Analyze -> Sort -> Return to Home.

12.5 Detaljna analiza bloka koda

Program je podeljen u četiri ključna segmenta:

Konfiguracija (podešavanje)

- **koristite kameru 1**: Objavljuje hardverski ulaz za vizuelne podatke.
- **Set end effector [Suction Cup]**: Soft-mapira kontrolu vazdušne pumpe na izlazu robota.

- **Idi na X:175, Y:39.5, Z:48.1**: Ovo je "Bezbednosna pozicija". Robot se povlači kako bi izbegao ometanje kamere i obezbeđuje optimalan put do bilo koje tačke na pojasu.

Logistika (kontrola transportera)

Unutar bloka ponavljanja zauvek:

- **Set transportni motor [STEPPER1] Brzina 15 mm / s**: Aktivira pojas kontrolisanom brzinom kako bi se sprečilo klizanje lakih predmeta.
- **sačekajte 3 sekunde**: Kritična vremenska konstanta koja definiše rastojanje između stavki.
- **Brzina 3**: Zaustavlja traku tako da robot može da izvrši precizan "statički izbor".

AI inspekcija (prepoznavanje slike)

- **Odbrojavanje 4 s**: Obezbeđuje vreme stabilizacije za kameru da se fokusira i za AI popup za obradu slike.
- **Ako slika prepozna oznaku 1 = [organski / plastični]**: Poziva API za duboko učenje da uporedi trenutni okvir sa obučenim skupom podataka i vraća oznaku klase.

Kinematika i manipulacija

Za svaku granu (organska / plastična), sistem koristi dva tipa kretanja:

1. **Zajedničko kretanje (PTP):** Pomera sve zglobove istovremeno za najbrži point-to-point putovanja.
2. **Ravna linija (linearna):** Pomera ruku strogo vertikalno na Z-osi kako bi se osiguralo da usisna čaša čini savršen kontakt bez kucanja stavku preko.

12.6 Operativni algoritam (korak po korak)

- **START:** Inicijalizujte instrumente i podesite krajnji efektor.
- **TRANSPORT:** Pomerite pojas za 3 sekunde, a zatim zaustavite.
- **SENSE:** Snimanje slike i identifikuju oznaku preko AI.
- **DECISION:**
- **If Organic:** Pomerite se da zgrabite koordinate, uključite usisavanje, pređite na organsku kantu, isključite usisavanje.
- **If Plastic:** Pomerite se da biste zgrabili koordinate, uključite usisavanje, pređite na plastičnu kantu, isključite usisavanje.
- **If Empty:** Izvršite signalni "shake" pokret (rotiranje R-ose).
- **RESET:** Povratak na početne koordinate bezbednosti.
- **LOOP:** Ponavljajte ciklus na neodređeno vreme.

12.7 Tehničke specifikacije koordinata

Na osnovu skripte kalibrisane su sledeće koordinate:

Tačka	X (mm)	Y (mm)	Z (mm)	R (°)	Pretraga
Home	175.0	39.5	48.1	12.7	Stanje mirovanja / čekanja
Approach	-56.3	155.5	7.4	109.9	Položaj iznad stavke
Grab	-52.8	125.3	-24.8	112.8	Kontakt tačka (spuštena)
Organic Bin	180.9	-67.5	-43.5	-20.5	Odlaganje organskih materija
Plastic Bin	161.9	63.2	-20.9	21.3	Odlaganje plastike

12.8 Rešavanje problema sa implementacijom

- **Netačno hvatanje:** Uverite se da je stavka centrirana. Ako se razlikuje, koristite vodiče na pojasu ili integrišite dinamičke X/Y povratne informacije iz AI kamere.
- **Greške u klasifikaciji:** AI prepoznavanje je osetljivo na svetlost. Koristite boju pojasa visokog kontrasta i konzistentno spoljno LED osvetljenje.
- **Bezbednost:** Uvek proverite da li je radni prostor čist od prepreka pre nego što započnete ponavljanje zauvek petlje.

IV. AI SA rPI 5

13. UVOD

Raspberry Pi je mali računar veličine špila karata i sposoban da pokrene punu Linuk desktop operativni sistem dok troši samo skromnu snagu. To uključuje USB portove za povezivanje tastature i miša zajedno sa raznim drugim periferijama, Ethernet adapter, i HDMI monitora veze. Raspberry Pi je prvobitno konstruisan za obrazovanje, ali je pronašao plodnu upotrebu za hobiste, kućnu automatizaciju, industrijske aplikacije i kao odgovarajuću tehnologiju za upotrebu u školama u većinskom svetu. Proizveden je u skladu sa direktivama RoHS (Ograničenje opasnih materija) i oslanja se na jednu microSD karticu za skladištenje. Pokreće varijantu Linuksa koja se zove Raspberry Pi OS i podržava širok spektar softvera otvorenog koda, uključujući mnoštvo obrazovnih programa. Štaviše, može se kupiti po skromnoj ceni. Ovaj vodič je napisan prvenstveno sa studentima inženjerstva i informatike na umu, ali će biti od interesa za druge koji imaju veliki interes da nauče više o programiranju i tehničkom računarstvu.

13.1 Inicijalno podešavanje Raspberry Pi

Ubacite SD karticu sa Raspberry Pi OS u Raspberry Pi i primenite napajanje. Kada prvi put pokrenete redovni Raspberry Pi OS, on će pokrenuti grafičko radno okruženje sa prijateljskim interfejsom vođenim menijem. Nakon prvog pokretanja, dijaloški okvir će se pojaviti koji će vas voditi kroz početno podešavanje. Pratite uputstva da biste konfigurisali tastaturu i korisničko ime. Navedite korisničko ime i lozinku kao što je zatraženo. Konfigurišite ViFi podešavanja i izaberite opciju "Ažuriraj softver" (imajte na umu da ovo može potrajati jako dugo kada prvi put podesite Raspberry Pi).

13.2 Prvi koraci sa komandnom linijom

Tu je-i verzija operativnog sistema zasnovana na komandnoj liniji koja se zove Raspberry Pi OS Lite. Ova verzija operativnog sistema troši manje energije od redovnog Raspberry Pi OS-a koji pokreće desktop en vironment i može se koristiti na starijim modelima Raspberry Pi sa manje RAM-a. Prilikom podešavanja Raspberry Pi sa Lite OS-om, od vas će se tražiti da konfigurišete tastaturu i navedete korisničko ime i lozinku. Lite verzija operativnog sistema je pogodna za korišćenje Raspberry Pi kao servera, kao ugrađenog sistema ili u aplikaciji IoT (Internet of Things). Međutim, za redovnu upotrebu na radnoj površini, najbolja je redovna verzija Raspberry Pi OS-a. Kada koristite Desktop OS, komandnoj liniji se i dalje može pristupiti pomoću aplikacije Terminal. Alternativno, može mu se pristupiti i pomoću virtuelne konzole pritiskom na CTRL + ALT + F3 koji će ući u terminal preko celog ekrana. Ako se pojavi poziv za prijavljivanje, možete se prijaviti koristeći korisničko ime i lozinku koju ste konfigurisali tokom podešavanja. Na radnu površinu možete se vratiti sa virtuelne konzole pritiskom na CTRL + ALT + F7. Dodatnim virtuelnim konzolama može se samostalno pristupiti pomoću CTRL + ALT zajedno sa tasterima F2 do F6.

13.3 Shell

Kada uđete u komandnu liniju, radićete u Linux shell-u. Jednostavno rečeno, shell je komandni tumač koji obezbeđuje bogat skup komandi koje se mogu koristiti za izvršavanje programa i interfejs sa operativnim sistemom. Raspberry Pi OS podrazumevano koristi Bash (Bourne Again Shell), ljusku zasnovanu na starijoj ljusci koja se zove Bourne Shell. BASH je popularan među korisnicima Linuka i ima automatsko popunjavanje komandne linije pomoću tastera tabulatora i može se koristiti za kreiranje programa koji se nazivaju shell skripte. Postoji mnoštvo različitih Linuk školjki koje se mogu koristiti. Kao što je naznačeno, podrazumevana Linuk shell je Bash shell, ali su dostupne i druge ljuske. Svaka školjka ima svoje karakteristike i opcije. Na primer, da prebacite podrazumevanu shell iz Bash u Z shell (zsh), upišite sledeće

```
sudo apt install zsh -y
chsh -s /bin/zsh
```

Nakon izdavanja ove komande, odjavite se, a zatim se vratite i sada bi trebalo da pokrenete zsh. Različite opcije konfiguracije mogu se podesiti unutar datoteke pod nazivom .zshrc koja se nalazi u vašem početnom folderu.

13.4 Shell komande

Neke od komandi dostupnih u ljusci su sažete u nastavku:

Pretraga	
cd directory	Izmenjuje tekući radni folder u folder

pwd	Prikazuje ime tekućeg radnog foldera
mkdir directory	Kreirajte novi direktorijum koji se zove direktorijum
rmdir directory	Uklanja folder koji se zove direktorijum
ls	Prikaz liste fajlova u tekućem direktorijumu
cp f1 f2	Kopirajte datoteku iz izvora f1 na odredište f2
rm filename	Uklonite ime datoteke
mv f1 f2	Premestite fajl sa f1 na f2
ftp host	Prenesite datoteke na i od domaćina

Ove komande predstavljaju samo deo korisničkih komandi dostupnih u Linuks shell. Online priručnik koji se naziva man (uputstvo) stranice pruža pomoć za mnoge komande i programe koji se mogu pozvati iz shell-a. Sintaksa za pozivanje man je sledeća:

```
man command-name
```

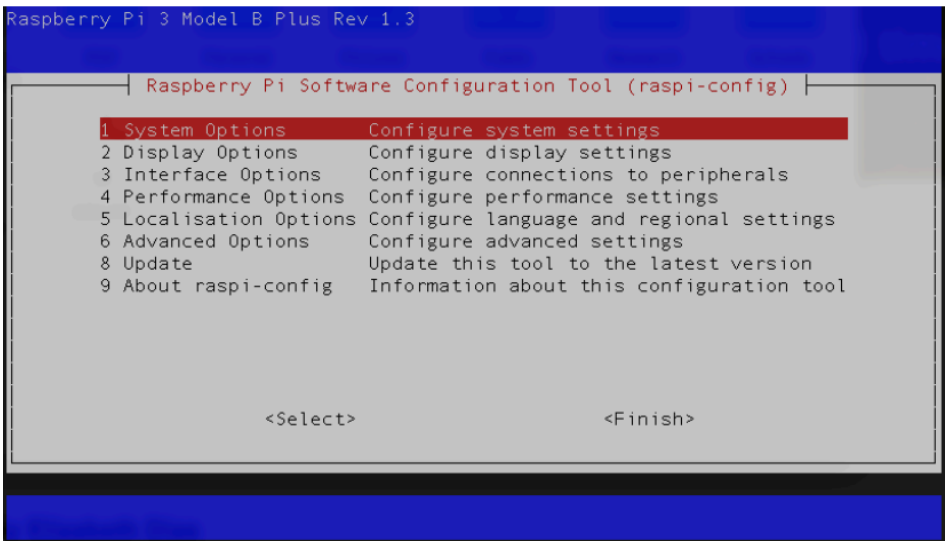
Informacije u vezi sa navedenom komandom će se zatim prikazati na ekranu.

13.5 Konfigurisanje Raspberry Pi OS-a

Raspberry Pi OS dolazi sa uslužnim programom koji se zove raspi-config koji se može koristiti za konfigurisanje širokog spektra podešavanja i usluga. Da biste pokrenuli ovaj uslužni program, upišite:

```
sudo raspi-config
```

Ovo će pokrenuti uslužni program za konfiguraciju Raspberry Pi unutar terminala sa glavnim menijem kao što je prikazano ispod:



Uslužni program za konfiguraciju Raspberry Pi može se koristiti za omogućavanje interfejsa kamere, kao i serijske, I3C i SPI komunikacije. Takođe uključuje opciju da se pokrene direktno u komandnu liniju, a ne na radnu površinu.

13.6 Daljinsko povezivanje sa Raspberry Pi

Moguće je pokrenuti Raspberry Pi bez glave, bez ekrana, tastature ili miša. Ovo je često slučaj kada se koristi Raspberry Pi u ugrađenoj aplikaciji ili IoT (Internet of Things) konfiguracija. Sledeći pododjeljci opisuju kako se daljinski povezati sa svojim Raspberry Pi koristeći jednu od sledećih opcija:

- pomoću USBtoTTL serijskog kabla
- koristeći SSH preko Ethernet ili ViFi veze
- Raspberry Pi Connect servis

Neki modeli Raspberry Pi mogu se povezati pomoću USB kabla. Ovo funkcioniše tako što omogućava USB Gadget Mode, koji omogućava USB port da se predstavi kao niz različitih tipova uređaja. Poznato je da ovo radi sa Raspberry Pi Zero, a ne sa većinom drugih modela. Dva pristupa opisana u nastavku treba da rade sa svim modelima Raspberry Pi.

14. UVOD U PROGRAMSKE JEZIKE

Raspberry Pi spremišta uključuju podršku za COBOL, kao i bogat skup modernijih programskih jezika kao što su Python, C, C ++ i Java. Ima podršku za više nišnih i nasleđenih jezika za programiranje. Generalno, paradigme za programske jezike spadaju u tri opšte kategorije:

• proceduralni programski jezici

• objektno orijentisani programski jezici

• funkcionalni programski jezici

14.1 Python programski jezik

Python je izmislio početkom 1990-ih Guido van Rossum. Python se može napisati korišćenjem proceduralne ili objektno orijentisane paradigme. To je projekat otvorenog koda koji je široko dostupan, koristi jednostavnu sintaksu i uključuje bogate biblioteke i nudi razne programske alate, Python izvorni kod se pokreće na virtuelnoj mašini koja prevodi kod u specifični mašinski kod koji izvršava procesor. Pošto Python tumači virtuelna mašina, on je nezavisan od platforme.

Raspberry Pi bi trebalo da podrazumevano instalira Python i može da pokreće Python programe direktno iz komandne linije. Da biste ušli u program, prvo vam je potreban editor običnog teksta. Ako koristite desktop okruženje, postoji prijateljsko, grafičko, integrisano razvojno okruženje (IDE) za Python pogodno za početnike pod nazivom Thonny. Da biste instalirali Thonny, upišite:

```
sudo apt install thonny
```

Za naprednije korisnike u grafičkom okruženju, program **vscode** pruža odličan editor za kodiranje na različitim jezicima, uključujući i Python. Kao što je ranije opisano, vscode može da se pokrene za uređivanje fajlova na daljinu. Ako koristite komandnu liniju, možete koristiti bilo koji od uređivača komandne linije opisanih u

prethodnim odeljcima, uključujući vi, emacs ili nano. Na primer, da biste uredili izvornu datoteku Python pod nazivom hello.py, upišite sledeće:

```
nano hello.py
```

Zatim unesite sledeći kod u izvornu datoteku:

```
name = input('What is your name? ')
print('Hi', name,' welcome to the Raspberry Pi!')
print('Good Bye')
```

Zatim, sačuvajte i izađite nano i pokrenite datoteku kucanjem:

```
python3 hello.py
```

Program bi trebalo da radi kao što se očekuje.

14.2 Kompajliranje i pokretanje programa C / C ++

Linuk ima razne alate za podršku razvoju softvera u C i C++. U stvari, sam operativni sistem Linuk je napisan u C. Programski jezik C je proceduralni jezik, a C++ gradi na C-u kako bi pružio podršku objektno orijentisanom programiranju. Kompajleri koje ćemo koristiti na Linuksu su GNU-ov Ц kompajler (gцц) i GNU-ov C++ kompajler (g++). Da biste bili sigurni da su instalirani GNU-ovi alati za prevođenje C/C++, upišite:

```
sudo apt install gcc g++ gdb build-essential
```

Na primer, da biste ušli u jednostavan C program pod nazivom hello.c koristeći nano editor, upišite sledeće:

```
nano hello.c
```

Koristeći editor, unesite sledeći kod u izvornu datoteku:

```
/* A Raspberry Pi C program */
#include <stdio.h>
int main(void)
{
    printf("Hello world.\n");
    printf("Compiled and run on a Raspberry Pi.\n");
    return 0;
}
```

Sačuvajte i izađite iz editora. Da biste kompajlirali izvorni kod, upišite sledeće na upit u prozoru terminala. Na primer, da biste sastavili program hello.c iznad, možete uneti:

```
gcc -Wall -o hello hello.c
```

GCC kompajler ima brojne druge opcije komandne linije koje možete koristiti. Za više informacija pogledajte man stranice. Imajte na umu da se C++ kompajler može pozvati korišćenjem g++ umesto gcc. Da biste pokrenuli kompajlirani program u trenutnom radnom direktorijumu, ne zaboravite da navedete ./ ispred imena programa

da odredite putanju kao trenutni direktorijum. Na primer, da biste pokrenuli program hello.c nakon kompajliranja kao ini, upišite:

```
.\hello
```

Ako kompajleru nije dato izlazno ime fajla, podrazumevano izlazno ime fajla će biti a.out.

14.3 Kompajliranje i pokretanje Java programa

Takođe je moguće razviti i pokrenuti Java programe koristeći Linuk na Raspberry Pi. Postoje dva različita paketa koji se mogu instalirati: on obezbeđuje Java Runtime Environment (JRE) i drugi obezbeđuje Java Development Kit (JDK). JRE vam samo omogućava da pokrenete Java programe, ali JDK omogućava kompajliranje i pokretanje Java programa. Da biste instalirali OpenJDK Java razvojni komplet, upišite sledeće:

```
sudo apt install default-jdk
```

Kada se ovo instalira, možete kompajlirati Java program. Na primer, unesite sledeći jednostavan Java program pod nazivom hello.java koristeći editor običnog teksta:

```
/* A Java program */
class Main {
    public static void main(String args[]) {
        System.out.println("Hello world.\n");
    }
}
```

Sastavite program tako što ćete ukucati sledeće u promptu:

```
javac hello.java
```

gde je myprogram.java ime Java izvorne datoteke. Da biste pokrenuli Java program, upišite sledeće u promptu: **java Main** gde je Main ime klase u kojoj program počinje.

15. ISTRAŽIVANJE VEŠTAČKE INTELIGENCIJE

15.1 Hardverska i softverska podrška za AI

Nedavni modeli Raspberry Pi procesora uključuju više ARM jezgara. Uprkos svojim respektabilnim hardverskim mogućnostima, Raspberry Pi može da se bori sa računarskim zahtevima zahtevnijih veštačke inteligencije (AI) aplikacije. Za zahtevnije aplikacije, postoji Raspberry Pi addon ploča - koja se naziva HAT (Hardware Attached on Top) koja obezbeđuje NPU (Neural Processing Unit) za ubrzavanje računara mašinskog učenja. Raspberry Pi AI HAT + obezbeđuje visokoučinkovit, energetski efikasan AI procesor za Raspberry Pi 3. Pored hardverske podrške, postoji širok spektar moćnih biblioteka za mašinsko učenje koje se mogu koristiti sa Raspberry Pi, uključujući

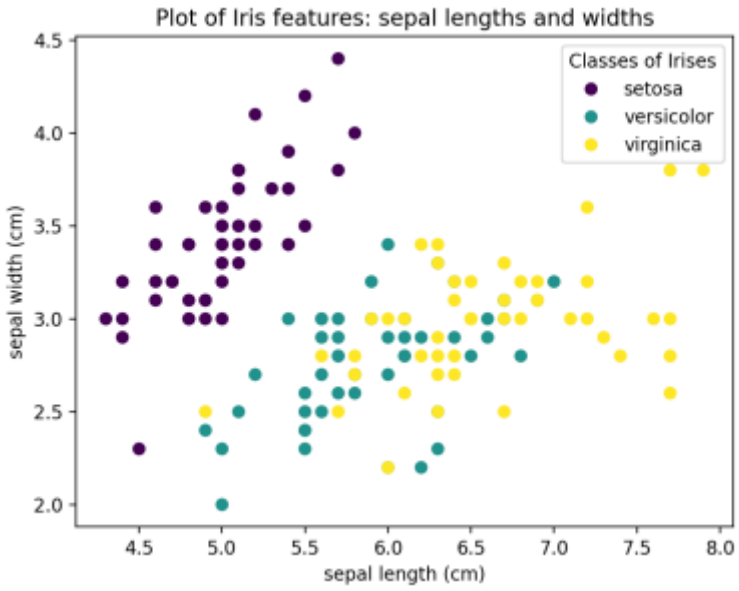
scikitlearn i LiteRT. Python uključuje brojne moćne biblioteke za podršku, kao što su Matplotlib i plotli za crtanje, NumPi za obezbeđivanje brzih operacija na nizovima i Pandas za analizu i manipulaciju podataka. Sledeći odeljci daju primere nekih od ovih biblioteka u akciji.

15.2 SciKit Learn

SciKit Learn je biblioteka za mašinsko učenje otvorenog koda koja radi sa Pythonom. SciKit Learn pruža mnogo različitih funkcija, uključujući regresijske i klasterne algoritme, analizu glavnih komponenti (PCA), linearnu diskriminantnu analizu (LDA) i vektorske mašine za podršku (SVM). Svi oblici mašinskog učenja zahtevaju skup podataka koji se koristi za obuku. SciKit uključuje selekciju skupova podataka za „igranje" koji se mogu koristiti za eksperimentisanje sa bibliotekama mašinskog učenja. Sledeći odeljci pokazuju PCA i SVM koristeći klasični skup podataka o cvetu irisa koji je deo zbirke skupova podataka za „igranje". Ovaj skup podataka irisa je prvobitno sastavio biolog Ronald Fisher u poznatom radu iz 1936. godine. Ovaj skup podataka sadrži uzorke tri vrste cveta Iris (Iris setosa, Iris virginica, i Iris versicolor) i merenja dužine i širine oba sepals1 i latica. Biblioteka SciKit Learn uključuje ovaj skup podataka irisa za potrebe testiranja i demonstracije. Sledeći Python kod koristi Matplotlib za crtanje širine čašice u odnosu na dužinu čašice za svaku od tri klase irisa u skupu podataka.

```
from sklearn import datasets
import matplotlib.pyplot as plt
# Load the classic iris dataset
iris = datasets.load_iris()
# Plot sepal lengths vs. widths for irises in dataset
fig, ax = plt.subplots()
scatter = ax.scatter(iris.data[:, 0], iris.data[:, 1], c=iris.target)
ax.set_title("Plot of Iris features: sepal lengths and widths")
ax.set(xlabel=iris.feature_names[0], ylabel=iris.feature_names[1])
fig = ax.legend(scatter.legend_elements()[0], iris.target_names, loc="best
", title="Classes of Irises")
plt.show()
```

Pokretanje ovog koda rezultira sledeći skupu podataka irisa:



15.3 SVM klasifikacija slika

Sledeći primer je inspirisan SciKit primerom o prepoznavanju rukom pisanih cifara i koristi rukom pisane cifre skup podataka iz UC Irvine repozitorijuma mašinskog učenja. Ovaj skup podataka ima 1797 uzoraka slika koji se sastoje od 8x8 niza piksela sa 10 klasa gde se svaka klasa odnosi na jednu cifru (0-9). Kratak Python program za učitavanje rukom pisanih cifara skup podataka i prikaz ih je prikazan ispod:

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, svm
from sklearn.model_selection import train_test_split

# load hand-written digits dataset
digits = datasets.load_digits()

# display a sample of ten hand-written digits in the dataset
fig, axes = plt.subplots(nrows=2, ncols=5)
for ax, digit in zip(axes.flatten(), digits.images):
    ax.set_axis_off()
    ax.imshow(digit,cmap='gray')
fig.tight_layout()
plt.show()
```

Grafik koji prikazuje deset rukom pisanih cifara u skupu podataka prikazano je ispod.

Možemo podeliti skup podataka rukom pisanih cifara u dva seta: skup za obuku i skup za testiranje. SciKit ima funkciju za uzimanje većeg skupa podataka cifara i razdvajanje na pola u skupove za obuku i testiranje na sledeći način:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

gde je X niz karakteristika i Y je vektor oznaka koje klasifikuju podatke. Stoga možemo nastaviti da treniramo SVM koristeći polovinu skupa podataka za obuku, a zatim koristimo preostalu polovinu skupa podataka za testiranje. Tačnost SVM-a može se odrediti upoređivanjem cifara predviđenih od strane SVM-a sa stvarnim oznakama za rukom pisane cifre i stopa prepoznavanja se može prijaviti. Stopa prepoznavanja je ukupan broj ispravno identifikovanih cifara slika podeljen sa ukupnim brojem test slika. Sledeći kod definiše SVM klasifikator na osnovu polovine skupa podataka, a zatim određuje stopu prepoznavanja koristeći drugu polovinu skupa podataka kao test slike.

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics, svm
from sklearn.model_selection import train_test_split

# Read digits and reshape digits as a vector of images
digits = datasets.load_digits()
```

```
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))

# Create a support vector machine classifier
svm = svm.SVC()
# Split the dataset: half for training and half for testing
X_train, X_test, y_train, y_test = train_test_split(
data, digits.target, test_size=0.5, shuffle=False)

# Train on hand-written digits in the training set
svm.fit(X_train, y_train)
# Predict the value of the digit using the testing set
predicted = svm.predict(X_test)

# compute the recognition rate
matches = 0
for x in range(len(y_test)):
if y_test[x] == predicted[x]:
matches += 1
recognition_rate = (matches/len(y_test)) * 100;
print(f'Recognition rate: {recognition_rate:.2f}%')
```

Izlaz ovog koda pokazuje sledeće:

Stopa prepoznavanja: 96.11%

Ovo ukazuje na respektabilnu stopu prepoznavanja koristeći SVM za klasifikaciju rukom pisanih cifara. Za više detalja, možemo iscrtati matricu konfuzije kako bismo vizualizovali tačnost algoritma mašinskog učenja. Matrica konfuzije je organizovana u redove i kolone: svaki red predstavlja svaku od klasa u skupu podataka i svaka kolona predstavlja predviđanja koja su napravljena. U idealnom slučaju, stvarne klase i predviđanja će se savršeno uskladiti tako da je dijagonala matrice konfuzije popunjena sa 100 i sa 0 svuda drugde. Dijagonalna martice odgovara pravim stopama prepoznavanja, dok sve ostale ćelije predstavljaju pojave lažnih klasifikacija.

16. PRIMERI

16.1 Izgradite sopstveni sigurnosni sistem "Parent Detector"

Pregled projekta: Da li ste se ikada zapitali ko se ušunja u vašu sobu kada niste u blizini? U ovom projektu ćete izgraditi pametan sigurnosni sistem - često nazvan "Detektor roditelja" da biste saznali tačno ko je bio u vašoj sobi.

To ćete postići kombinovanjem Raspberry Pi, senzora pokreta i kamere za automatsko pokretanje i snimanje video snimaka bilo kakvih uljeza.

Korak 1: Skupite svoj hardver

Da biste izgradili ovu automatizovanu sigurnosnu kameru, trebat će vam tri glavne hardverske komponente:

- Raspberry Pi: Ovo deluje kao mozak vašeg projekta, obrađuje signale i pokreće kod.
- Pasivni infracrveni (PIR) senzor pokreta: Ova komponenta deluje kao "oči" vašeg sistema, otkrivajući promene u infracrvenoj toploti da oseti kada se osoba kreće u blizini.
- Raspberry Pi modul kamere: Ovo je uređaj koji će snimiti video dokaze uljeza.

Korak 2: Razumeti i podesiti PIR senzor

Pre nego što priključite bilo šta sa žicama, potrebno je da podesite fizička podešavanja na samom PIR senzoru. Ako pogledate zadnju stranu PIR modula, videćete dve narandžaste komponente sa malim utičnicama u obliku krsta koje savršeno odgovaraju Philips odvijaču.

Ovi narandžasti brojčanici se nazivaju potenciometri i omogućavaju vam da ručno podesite kako se senzor ponaša:

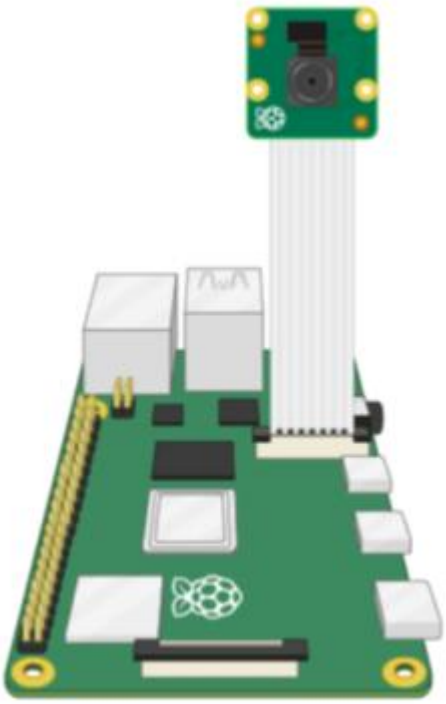
- Osetljivost: Određuje koliko pokreta je potrebno da se pokrene senzor.
- Vreme (kašnjenje): Određuje koliko dugo senzor ostaje "aktiviran" nakon otkrivanja pokreta pre nego što se resetuje.

Inicijalno podešavanje: Da bi ovaj projekat najbolje funkcionisao, pomoću odvijača podesite potenciometar osetljivosti na apsolutni maksimum i podesite vremenski potenciometar na apsolutni minimum. Uvek možete podesiti i promeniti ova podešavanja kasnije ako otkrijete da je senzor previše osetljiv ili ostaje predugo.

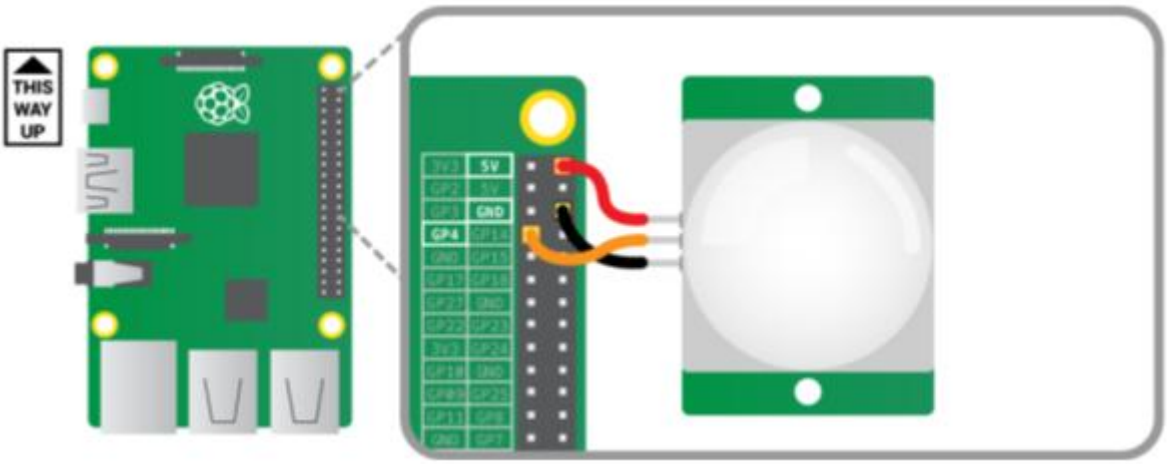
Korak 3: Bezbedno ožičite komponente

Sada je vreme da povežete hardver sa Raspberry Pi. Ključno pravilo: Uvek proverite da li je vaš Raspberry Pi potpuno isključen i isključen pre povezivanja hardvera.

1. Povežite kameru: Pažljivo umetnite trakasti kabl modula kamere u namenski port kamere na Raspberry Pi.



2. Povežite PIR senzor: PIR senzor treba da pošalje svoje podatke o kretanju na Raspberry Pi. Spojit ćete izlazni pin podataka PIR senzora direktno na pin označen GPIO 3 na vašoj Raspberry Pi ploči. (Takođe ćete morati da povežete napajanje (VCC) i uzemljenje (GND) pinove na odgovarajuće 5V i GND pinove na Pi).



Korak 4: Napišite sigurnosni softver (Python)

Kada je vaš hardver ožičen, uključite Raspberry Pi. Otvorite programsko okruženje pod nazivom Thonni, kreirajte novu datoteku i odmah je sačuvajte kao parent_detector.py.

Da bi naš hardver radio, moramo da uvezemo određene biblioteke. Koristićemo MotionSensor iz biblioteke gpiozero za rukovanje PIR senzorom, a klasu kamere iz biblioteke picamzero za kontrolu modula kamere.

Logika kodeksa:

1. Kontinuirano praćenje: Koristimo neko vreme Istina: petlja, koja je beskonačna petlja koja održava program zauvek proverava kretanje.

2. Čekanje akcije: Program koristi pir.wait_for_motion() da pauzira kod dok senzor ne detektuje kretanje. Kada se detektuje kretanje, on štampa poruku na ekranu.

3. Snimanje: Kamera počinje snimanje video zapisa pomoću funkcije cam.start_recording ().

4. Zaustavljanje: Ne želimo da zauvek snimimo praznu sobu. Kod koristi pir.wait_for_no_motion() da sačeka dok uljez ne ode, a zatim bezbedno zaustavlja video fajl koristeći cam.stop_recording().

Rešavanje kritične greške (problem prepisivanja): Ako jednostavno kažemo kameri da sačuva datoteku kao intruder.mp4, svaki put kada nova osoba uđe u sobu, stari video će biti potpuno prepisan i izbrisan. Da bismo osigurali da vodimo video dnevnik svih (dosadnih roditelja ili braće i sestara), potrebno nam je dinamično ime datoteke. Možemo da koristimo Python vremensku biblioteku da automatski saznamo trenutni datum i vreme i ubrizgamo ga u ime datoteke video zapisa.

Počnite sa ovim kodom i vašim omiljenim AI alatima za testiranje ovog koda:

Upišite ovo u datoteku parent_detector.py:

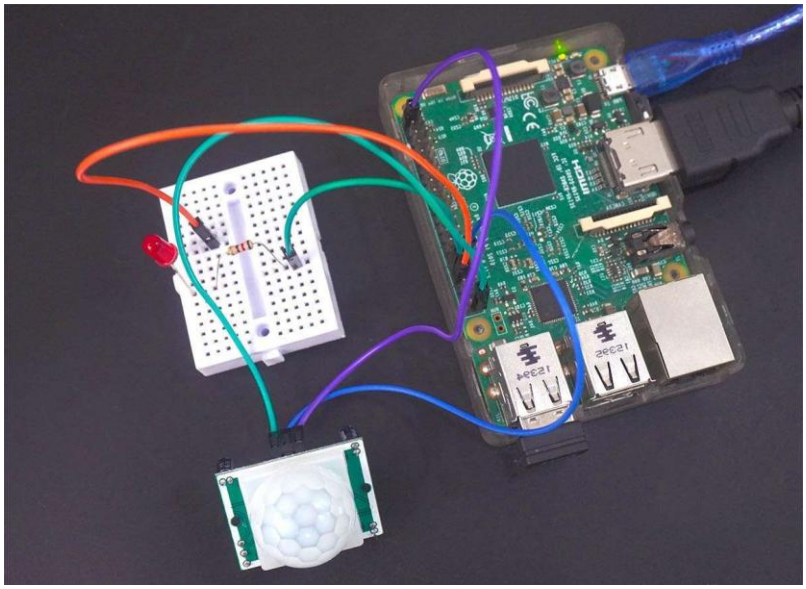
```
1. from gpiozero import MotionSensor
 2. from picamzero import Camera
 3. import time
 4.
 5. # 1. Initialize Hardware
 6. # Set up the PIR sensor on GPIO 4
 7. pir = MotionSensor(4)
 8.
 9. # Create a Camera object to control the module
10. cam = Camera()
11.
12. print("Security System Armed. Waiting for intruders...")
13.
14. # 2. Main Security Loop
15. while True:
16.     # Program pauses here until movement is detected
17.     pir.wait_for_motion()
18.     print("WARNING: Motion detected!")
19.
20.     # 3. Generate a Unique Filename
21.     # This creates a string like "20231025-143000" (YearMonthDay-HourMinuteSecond)
22.     timestamp = time.strftime("%Y%m%d-%H%M%S")
23.     filename = f"intruder_{timestamp}.mp4"
24.
25.     # Start recording video evidence to the new file
26.     print(f"Recording video: {filename}")
27.     cam.start_recording(filename)
28.
29.     # 4. Wait for the room to be empty again
30.     pir.wait_for_no_motion()
31.     print("Motion stopped.")
```

```
32.
33.     # Stop recording to finalize and save the video file safely
34.     cam.stop_recording()
35.     print("System returning to standby mode.")
```

Sada kada je vaš kod u potpunosti napisan i vaš hardver je bezbedno povezan, vreme je da testirate da li vaš roditelj detektor zaista radi u stvarnom svetu! Prvo, kliknite na dugme Pokreni unutar vašeg Thonni programskog okruženja da biste pokrenuli skriptu. Kada se program pokrene i čeka, namerno mahnite rukom direktno ispred PIR detektora pokreta kako biste simulirali uljeza koji ulazi u sobu. Odmah pogledajte ekran računara; trebalo bi da vidite reči "Motion detected!" odštampane u području konzole, potvrđujući da je senzor uspešno pokrenuo kameru da započne snimanje.

Nakon aktiviranja alarma, potrebno je testirati mehanizam za isključivanje. Izađite iz vidokruga senzora, držite se potpuno mirno ili rukom nežno prekrijte belu kupolu senzora. Sačekajte nekoliko sekundi dok vam konzola ne kaže da je kretanje zaustavljeno i da se sistem vraća u stanje pripravnosti. Ovo ukazuje da je program bezbedno završio i zatvorio video datoteku. Na kraju, otvorite menadžer datoteka Raspberry Pi i idite na istu fasciklu u kojoj ste sačuvali *parent_detector.py* skriptu. Sada bi trebalo da vidite potpuno novu .mp4 video datoteku koja vas čeka, sa jedinstvenim datumom i vremenskom oznakom u svom nazivu baš kao što smo programirali. Dvaput kliknite na ovu novokreiranu datoteku da biste gledali snimljene dokaze, proverili kvalitet video zapisa i uverili se da ugao kamere savršeno snima vrata!

Dodatne slike:



Izvor: https://www.electronicwings.com/raspberry-pi/pir-motion-sensor-interfacing-with-raspberry-pi

Izvor: https://thepihut.com/products/pir-camera-case-for-raspberry-pi-4-3

16.2 IOLO Prepoznavanje procene položaja

U ovom vodiču ćemo postaviti neke sa OpenCV-om i IOLO porodicom modela za procenu položaja na Raspberry Pi 3. Pogledaćemo nekoliko različitih dostupnih YOLO modela, kao i kako ih optimizirati kako biste dobili glatkiji FPS, kao i kako koristiti ključne podatke generisane modelom kako biste mogli implementirati procenu poze u svoj sledeći projekat. Ovo je bio jedan od najzabavnijih vodiča koje smo napravili u neko vreme, pa hajde da uđemo u to!

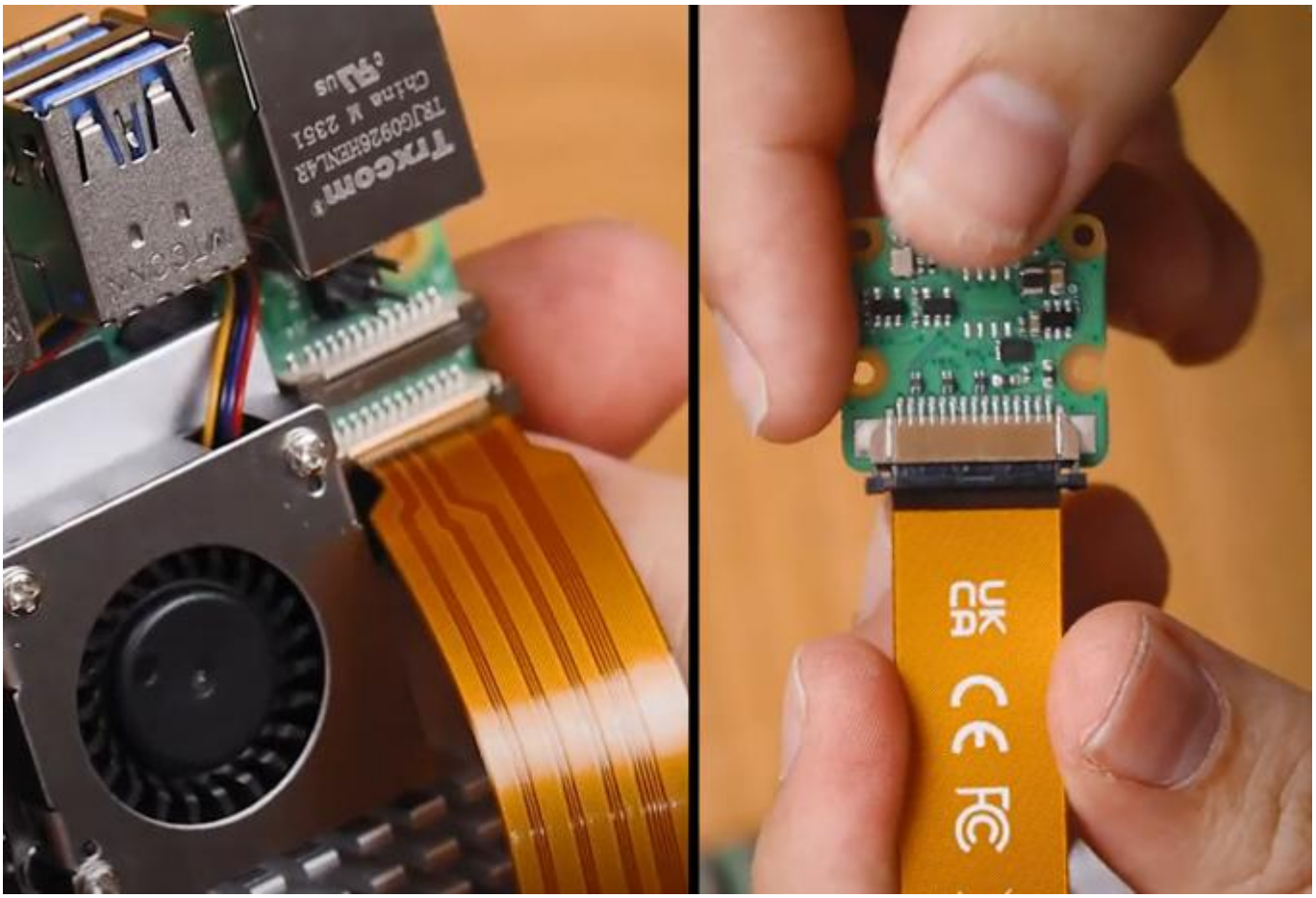
Da biste pratili ovaj vodič, trebaće vam:

- Raspberry Pi 5 - Ovde će raditi ili 4GB ili 8GB model. Iako bi se to tehnički moglo uraditi na Pi 4, to je daleko sporije od Pi 5 i ne bi bilo lepo iskustvo, i iz tih razloga, nismo testirali na Pi 4
- Pi kamera - Mi koristimo modul kamere V3
- Adapter kabl - Pi 5 dolazi sa CSI kablom za kameru različite veličine, a vaša kamera može doći sa starijim debljim, tako da je vredno dvostruke provere. Modul kamere V3 će trebati jedan
- Rešenje za hlađenje - Koristimo aktivni hladnjak (kompjuterski vid će zaista gurnuti vaš Pi do svojih granica)
- Napajanje
- Micro SD kartica - Najmanje 3GB veličine
- Monitor i Micro-HDMI na HDMI kabl
- Miš i tastatura

hardver Skupština

Što se tiče hardverske montaže, ovde je prilično lagano. Povežite deblju stranu kabla na kameru, a tanju stranu na Pi 5. Ovi konektori imaju jezičak na njima - podignite ih, a zatim umetnite kabl u utor. Kada sedi tamo lepo i kvadratno, gurnite jezičak nazad dole da stegnete kabl na svoje mesto.

Samo pripazite jer ovi konektori rade samo u jednoj orijentaciji, a mogu biti krhki, tako da izbegavajte da ih čvrsto savijate (malo je u redu).



Instaliranje Pi OS-a

Prvo, moramo da instaliramo Pi OS na mikro SD karticu. Koristeći **Raspberry Pi Imager**, izaberite Raspberry PI 5 kao uređaj, Raspberry Pi OS (64-bitni) kao operativni sistem, i vaš microSD kartica kao uređaj za skladištenje.

Isti postupak kao i za PiRacer.

Uspostavljanje virtuelnog okruženja i instaliranje biblioteka

Sa uvođenjem Bookvorm OS-a 2023. godine, sada se od nas traži da koristimo virtuelna okruženja (ili venv), jer su oni izolovani prostor na Pi gde možemo eksperimentisati bez rizika da naškodimo ostatku našeg Pi OS-a ili projekata. U ovom vodiču imamo sve potrebne komande i uputstva.

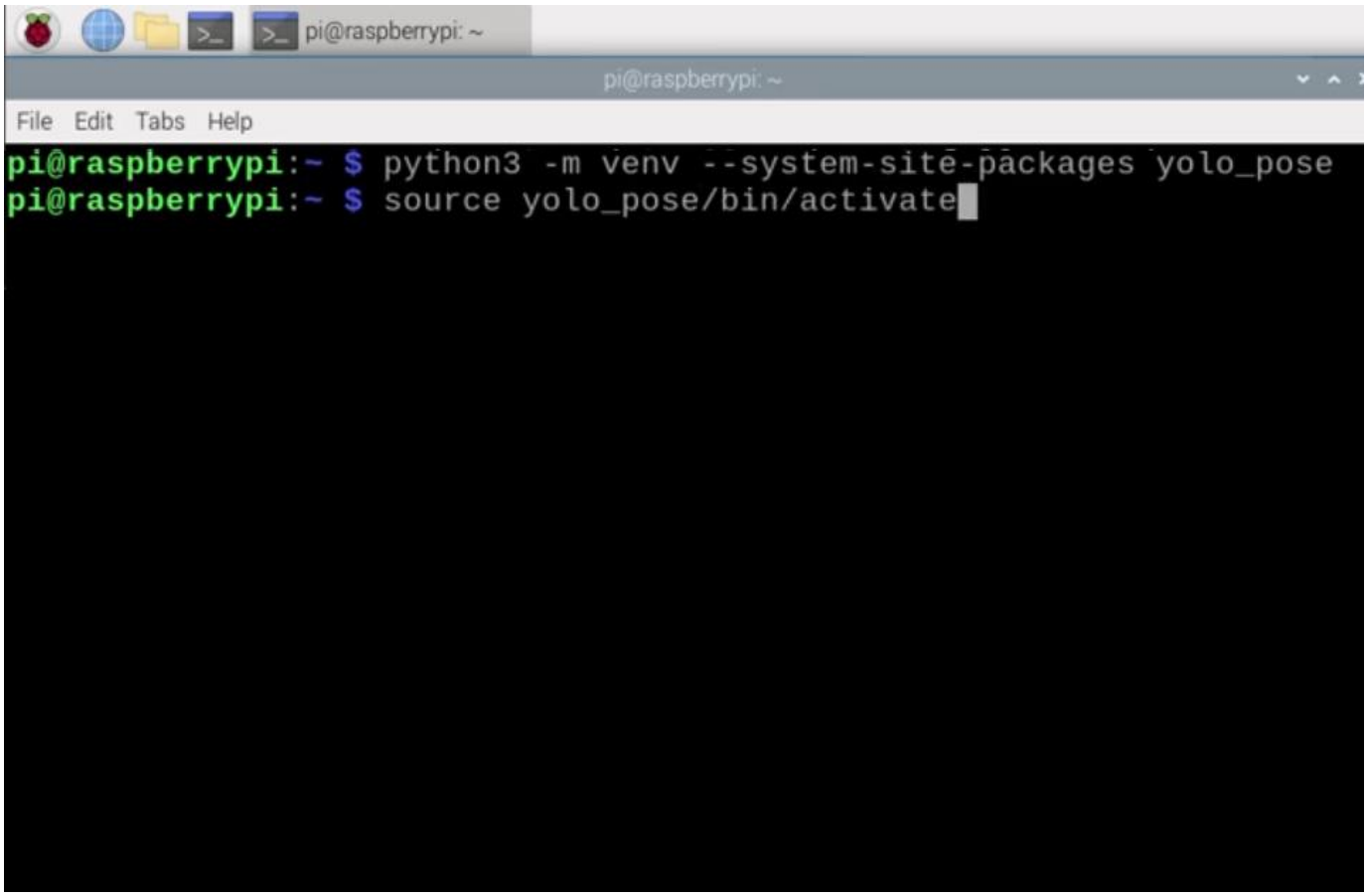
Da biste kreirali virtuelno okruženje, otvorite novi prozor terminala i ukucajte:

python3 -m venv --system-site-packages yolo_pose

Nakon kreiranja venv-a, možemo ući u njega ukucavanjem:

source yolo_pose/bin/activate

Nakon što to učinite, videćete ime u virtuelnom okruženju levo od zelenog teksta - to znači da ispravno radimo u njemu. Ako ikada budete imali potrebu da ponovo uđete u ovo okruženje (na primer, ako zatvorite prozor terminala izaći ćete iz okruženja), samo ponovo ukucajte komandu izvora iznad.



Sada kada radimo u virtuelnom okruženju, možemo početi sa instaliranjem potrebnih paketa. Prvo, uverite se da je PIP (Python menadžer paketa) ažuriran unosom sledeće tri linije:

sudo apt update

sudo apt install python3-pip -y

pip install -U pip

Zatim instalirajte Ultralitiцs paket sa:

PIP instalira ultralitiku[izvoz]

Divni ljudi u Ultralitiцs su bili jedan od ključnih programera i održavatelja najnovijih IOLO modela. Ovaj njihov paket će obaviti veliki deo teškog posla i instaliraće OpenCV kao i svu potrebnu infrastrukturu za pokretanje IOLO-a.

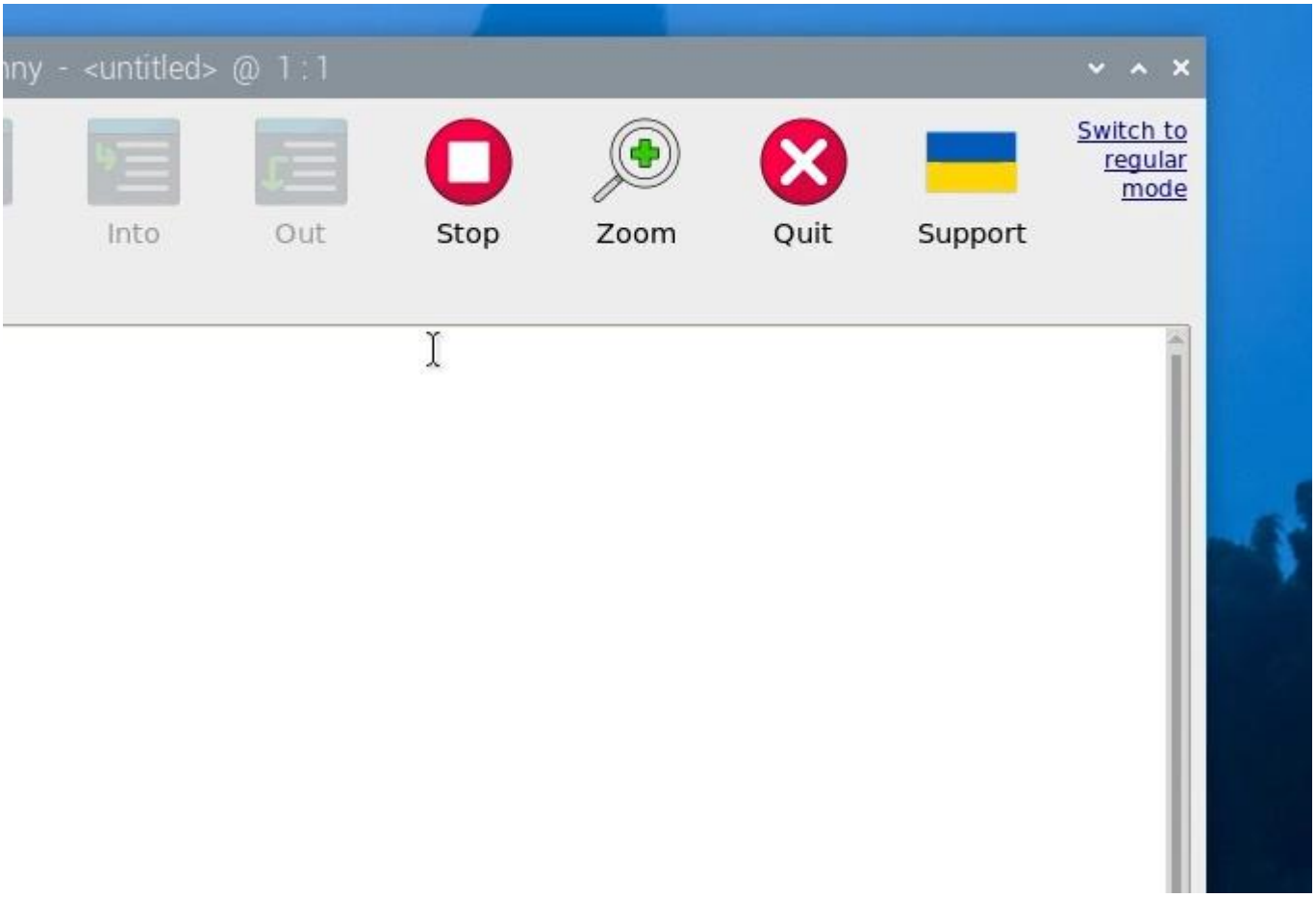
Ovaj proces će takođe instalirati prilično veliku količinu drugih paketa, i kao rezultat toga, sklon je neuspehu. Ako vaša instalacija ne uspe (prikazaće ceo zid crvenog teksta), samo ponovo ukucajte liniju za instalaciju Ultralitika i trebalo bi da se nastavi. U retkim slučajevima, instalacijska linija će možda morati da se ponovi nekoliko puta.

Kada se završi instalacija, ponovo pokrenite Raspberry Pi. Ako želite da budete napredni korisnik, možete to učiniti tako što ćete ukucati u ljusku:

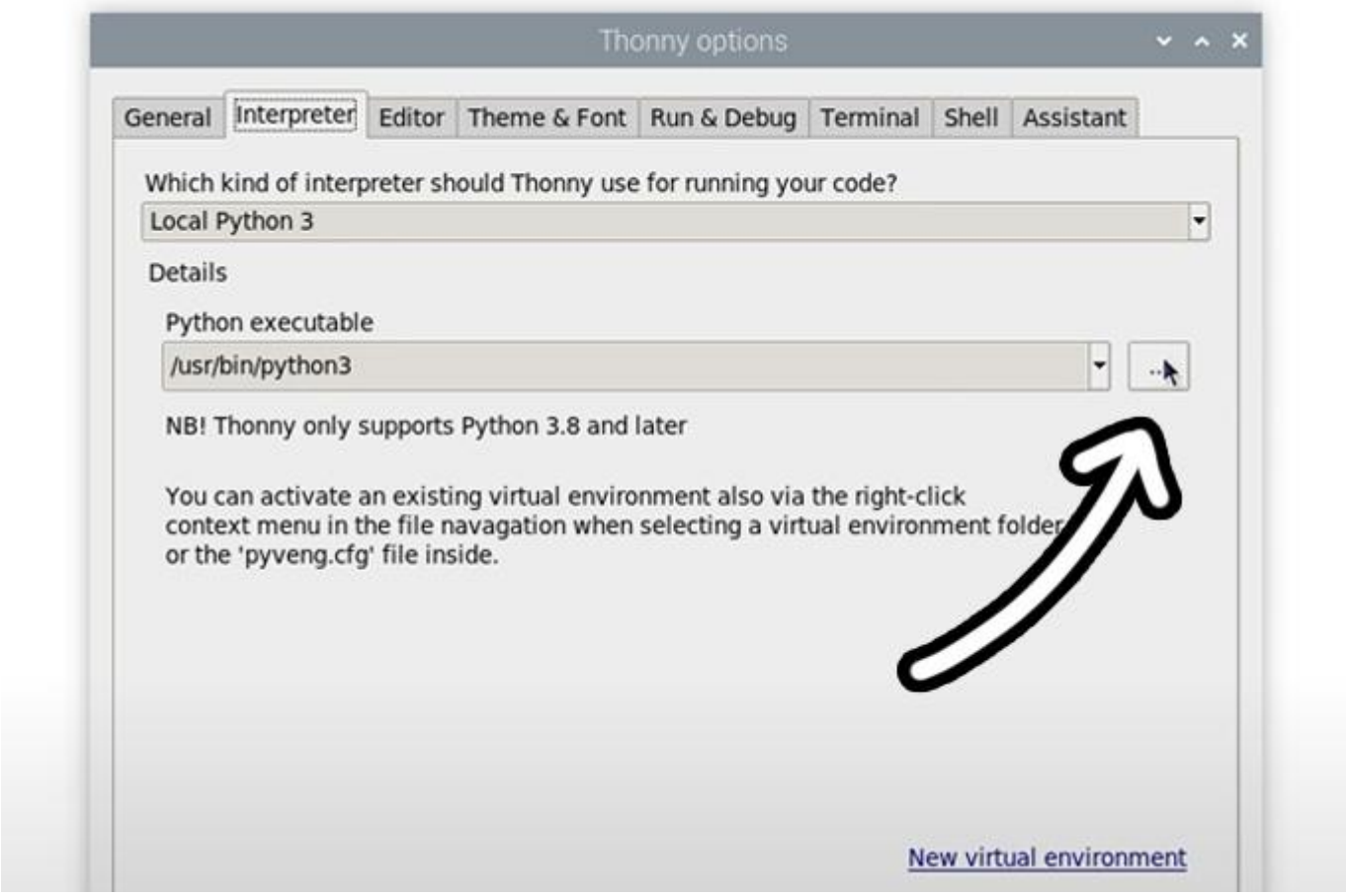
Ponovo pokretanje

Imamo još jednu stvar da uradimo, a to je da podesite Thonni da koristi virtuelno okruženje koje smo upravo stvorili. Thonny je program iz kojeg ćemo pokrenuti sav naš kod i moramo ga naterati da radi iz istog venv-a, tako da ima pristup bibliotekama koje smo instalirali.

Prvi put kada otvorite Thonni može biti u pojednostavljenom režimu, a videćete "prebacivanje na redovni režim" u gornjem desnom uglu. Ako je ovo prisutno, kliknite na njega i ponovo pokrenite Thonni tako što ćete ga zatvoriti.



Sada uđite u meni opcija prevodioca izborom Pokreni > Konfiguriši prevodioca. Pod Python izvršnom opcijom, postoji dugme sa 3 tačke. Izaberite ga i idite na izvršnu datoteku Python u virtuelnom okruženju koje smo upravo kreirali.



Ovo će se nalaziti pod home/pi/yolo_pose/bin i u ovoj datoteci ćete morati da izaberete datoteku pod nazivom "Python3". Hit u redu i sada ćete raditi u ovom venv.

Kad god otvorite Thonni, sada će automatski raditi iz ovog okruženja. Možete promeniti okruženje iz kojeg radite tako što ćete ga izabrati iz padajućeg menija ispod Python izvršne datoteke u istom meniju opcija prevodioca. Ako želite da izađete iz virtuelnog okruženja, izaberite opciju bin / Python3.

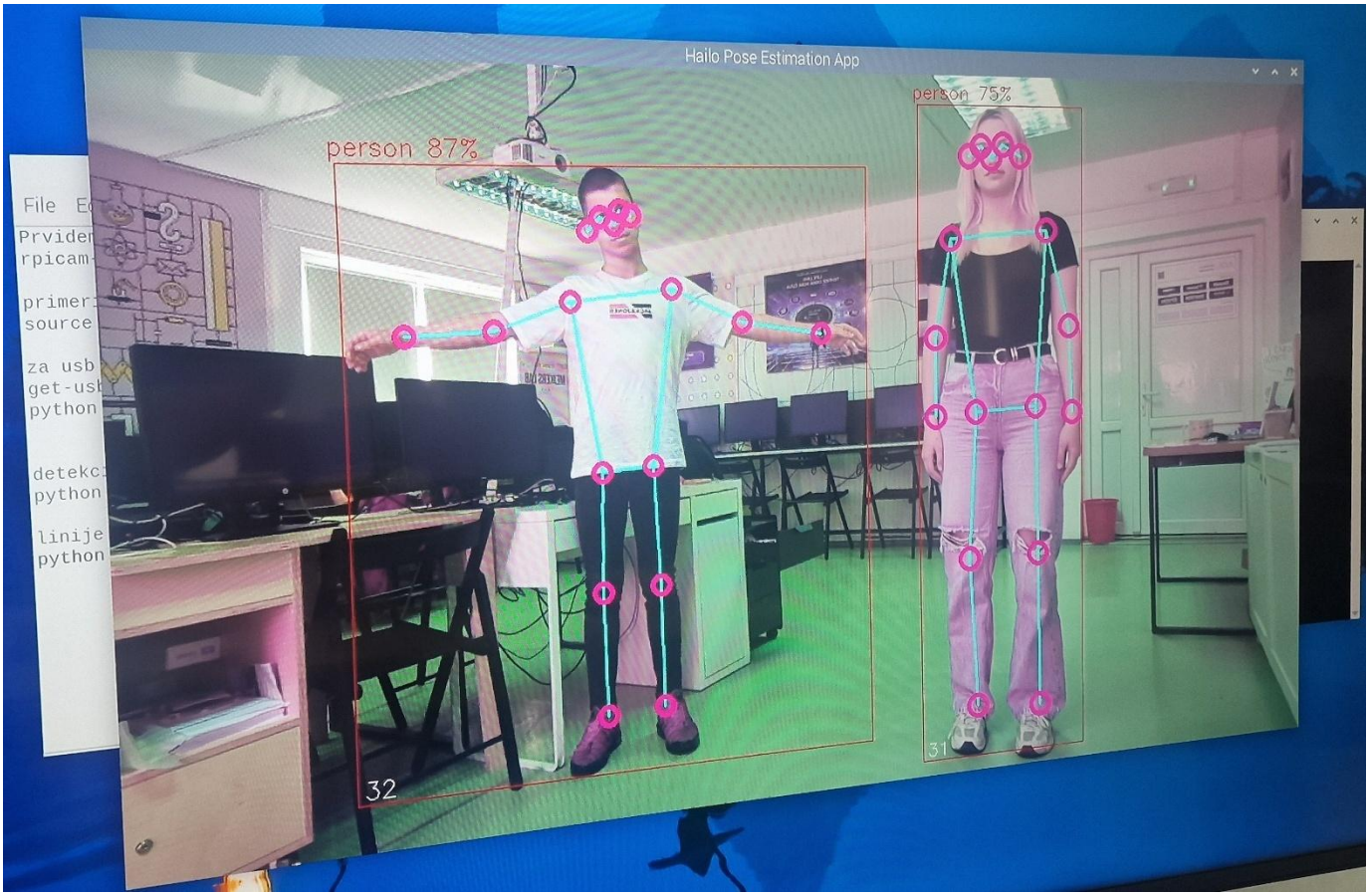
Procena poze trčanja

Sada kada imamo instalirane biblioteke i Thonni radi iz virtuelnog okruženja, možemo pokrenuti našu skriptu za procenu poze. Samo napred i izvadite **zip fasciklu projekta** (preuzmite sa našeg servera) na pogodno mesto kao što je radna površina. Tamo ćete naći prvu skriptu koju ćemo koristiti *"pose demo.py".* Otvori ga Thonni i pritisnite veliko zeleno dugme za trčanje. Prvi put kada pokrenete ovo može instalirati nekoliko dodatnih potrebnih stvari (sve automatski), a nakon nekoliko sekundi trebalo bi da vidite prozor za pregled sa vašom procenom poze.

Nekoliko stvari bi trebalo da se dešava ovde. Prvo će IOLO pokušati da otkrije ljude, a ako ih prepozna, nacrtaće okvir oko njega sa ocenom poverenja na vrhu. Važno je da će se postavljati tačke na kojima misli da su neka

bitna mesta vašeg tela (to se zovu ključne tačke), i to će biti crtanje linija između ovih tačaka da proceni pozu i orijentaciju osobe. U gornjem desnom uglu će takođe biti FPS na kojem se radi (koji ćemo malo poboljšati).

I to je to! Sa ovih nekoliko koraka, već imamo našu procenu Pi trčanja poze!

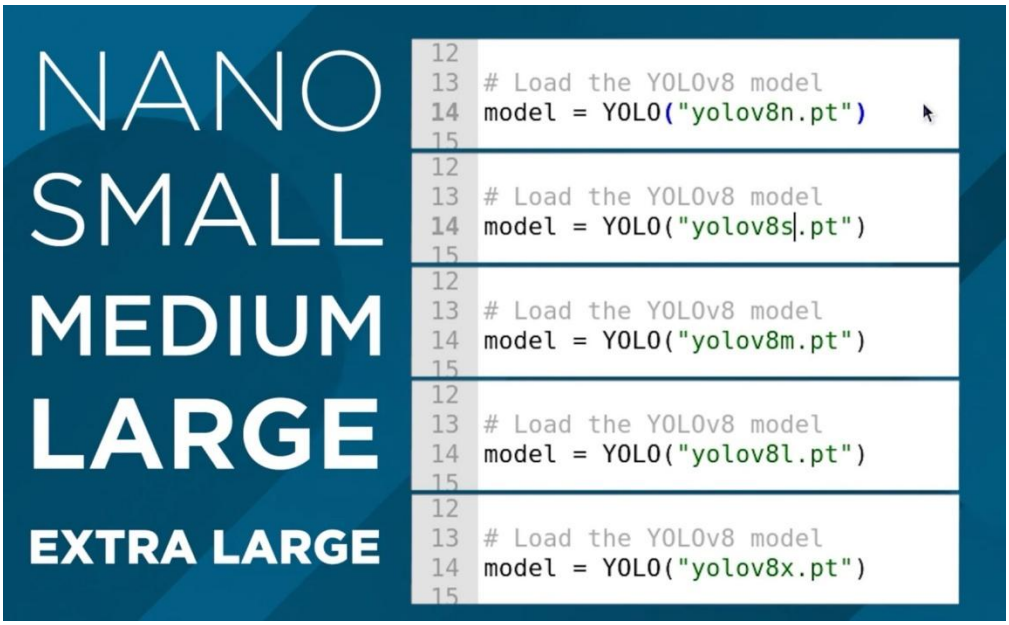


Promena IOLO modela

Do sada smo pokrenuli IOLO3, a jedna od lepota ovog Ultralitics paketa je da možemo jednostavno zameniti jednu liniju u kodu da bismo u potpunosti promenili model. Ovo možemo koristiti za pokretanje naprednijeg IOLO3 modela, ili čak starijeg modela. Sve što treba da promenite je ova linija ovde u podešavanju:

```
# Load our YOLO11 model
model = YOLO("yolo11n-pose.pt")
```

Ova linija trenutno koristi nano model koji je najmanji, najmanje moćan, ali najbrži model IOLO11, a ovu liniju možemo promeniti da pokrenemo jednu od različitih veličina u kojima ovaj model dolazi promenom jednog slova posle "11" kao što je prikazano na desnoj strani. Ako promenite ovu liniju na drugu veličinu modela i pokrenete ga, skripta će automatski preuzeti novi model (koji može biti u 100s Mb za veće modele).



Razlika između ovih modela je kompromis između performansi procene poza i FPS-a. Što je veći model, to je bolje u proceni delova vašeg tela koji se možda ne vide od strane kamere, kao i složenijih uglova i okvira sa više ljudi u njemu, međutim, možete očekivati da ćete dobiti samo 1 okvir obrađen svakih 10 sekundi! Mi ćemo to povećati u sledećem koraku.

Nano model, s druge strane, radi najbrže, dobija oko 1.5 FPS bez optimizacije, ali nema procesorsku snagu većih modela. Za procenu poze, možete se izvući sa nano modelom većinu vremena jer je obično dovoljno dobar za vaše potrebe, ali ako vam je potrebno nešto malo moćnije, nastavite da povećavate veličinu modela kako bi odgovarala vašim potrebama.

U ovoj liniji, takođe možemo da promenimo verziju IOLO trčanja. Možete se vratiti na stariji model ako želite, ili možete koristiti noviji model. Ovaj vodič će na kraju biti zastareo i ako Ultralitiцs objavi IOLO13, jednostavno bi trebalo da budete u mogućnosti da promenite liniju na sledeće da počnete da koristite noviju verziju IOLO:

```
# Load our YOLO11 model
model = YOLO("yolo13n-pose.pt")
```

Povećanje brzine obrade

Postoje 2 stvari koje možemo da uradimo da povećamo FPS na Pi, a najefikasniji način je da pretvorimo model u format koji se zove NCNN. Ovo je format modela koji je optimizovan za rad na procesorima zasnovanim na ARM-u kao što je Raspberry Pi. Otvorite skriptu pod nazivom "ncnn conversion.py" i naći ćete sledeće:

```
from ultralytics import YOLO

# Load a YOLO11n PyTorch model
model = YOLO("yolo11n-pose.pt")

# Export the model to NCNN format
model.export(format="ncnn", imgsz=640)  # creates 'yolov11n-pose_ncnn_model'
```

Da biste koristili ovu skriptu, najpre navedite model koji želite da konvertujete. Ovo koristi iste konvencije imenovanja o kojima smo govorili u poslednjem odeljku. Zatim je format modela "ncnn" naveden kao izlazni format, kao i rezolucija. Za sada držite ovo na podrazumevanom 640. Prvi put kada pokrenete ovu skriptu ona će

preuzeti još neke dodatne stvari koje su joj potrebne, ali bi trebalo da traje samo nekoliko sekundi da pokrene stvarnu konverziju.

Kada se to završi, u folderu skripte žive u vašem će naći novu fasciklu koja se zove nešto u stilu "iolo11n-pose_ncnn_model". Kopirajte ime ovog fajla i vratite se na našu demo skriptu od ranije.

Sada morate da kažete skripti da koristi ovaj model koji smo kreirali promenom linije modela na ime te fascikle koju je upravo kreirao. Trebalo bi da izgleda ovako:

```
# Load our YOLO11 model
model = YOLO("yolo11n-pose_ncnn_model")
```