Co-funded by the European Union

PROJEKTY S UMELOU INTELIGENCIOU

Technická škola Pirot

KA220-VET - Partnerstvá v oblasti odborného vzdelávania a prípravy

Názov projektu: Nástroje umelej inteligencie pre školy odborného vzdelávania a prípravy

Dátum dokumentu: marec 2026
Tento materiál bol zostavený a pripravený na účely projektu Erasmus:
Technická škola Pirot (autor: Bojan Ćirić, spoluautori: Boban Blagojević, Aleksandar Madić)
ICEP (autor: Ladislav Mariš, spoluautor: Adelaida Fanfarová)
Agrupamento de Escolas Tomas Cabreira (autor: Sandra Nobre, spoluautori: Rui Dias, Gilherme Mota, Carla Lima)
Preložila: Adelaida Fanfarová

Web: https://book.tsp.edu.rs

Obsah

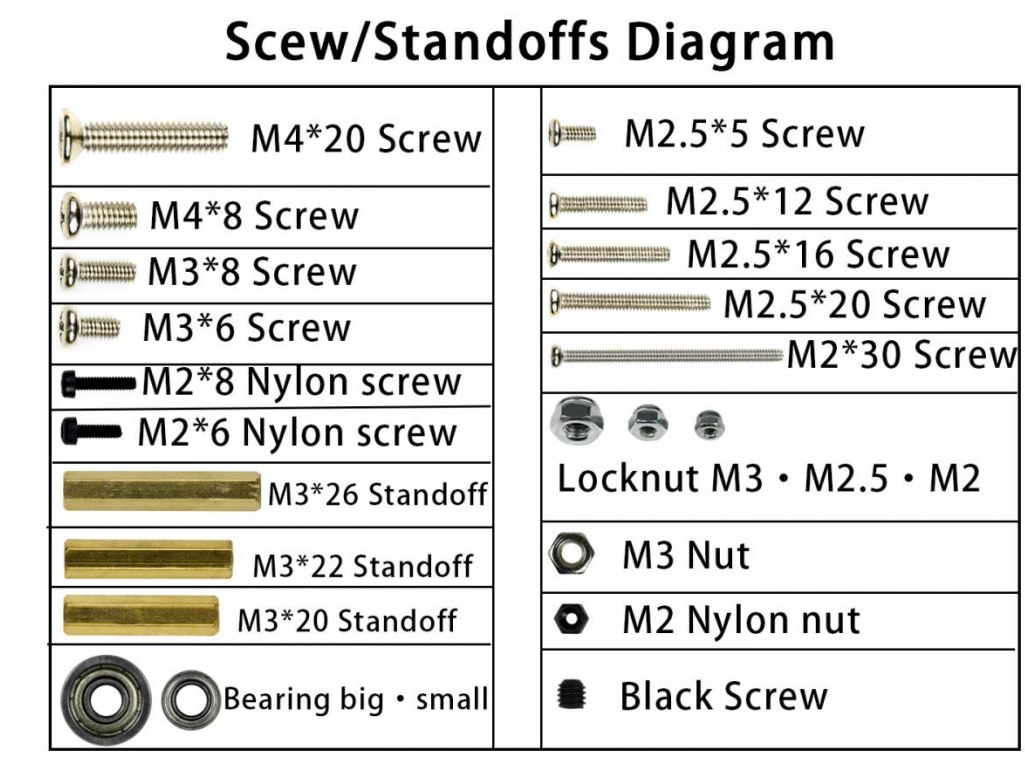
I. PiRACER AI AUTOPILOT

1. PI-RACER DEEP LEARNING AUTOPILOT

1.1 Montážny manuál PiRacer

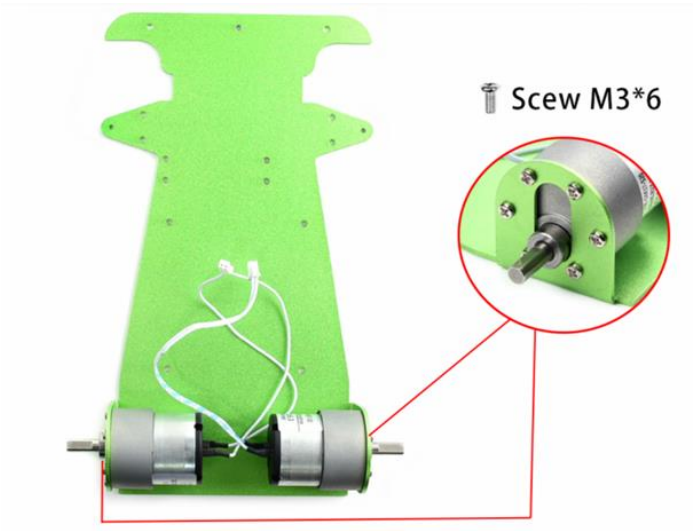
Schéma skrutiek/odstupov

Schéma pre referenciu. Všimnite si, že skrutky, ktoré sú súčasťou servokolia, tu nie sú uvedené.



1. Pripevniť motory k kovovému podvozku

Poznámka: Nepoužívajte M3*8. Je dlhšia a môže poškodiť motor.



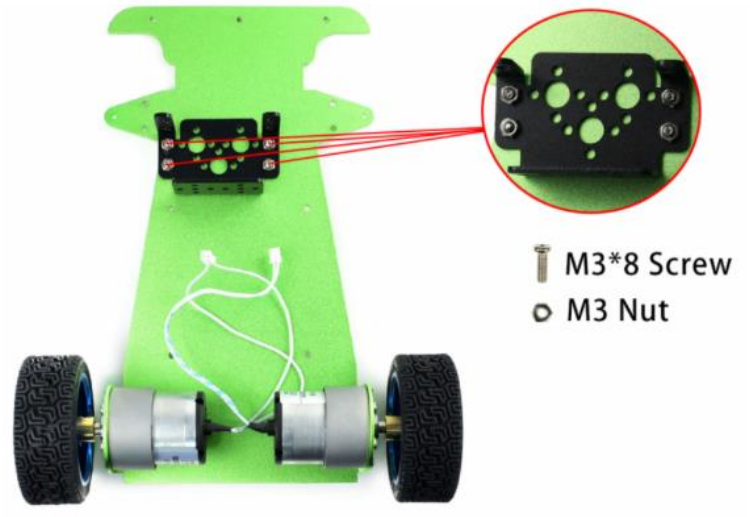
2. Pridať spojky na kolesá

Najprv vložte čiernu skrutku do spojky. Potom pridajte spojku na koleso. Možno budete musieť spojku zatlačiť na koleso. Spojku upevnite k kolesu pomocou skrutky M4*8.

3. Zostaviť kolesá

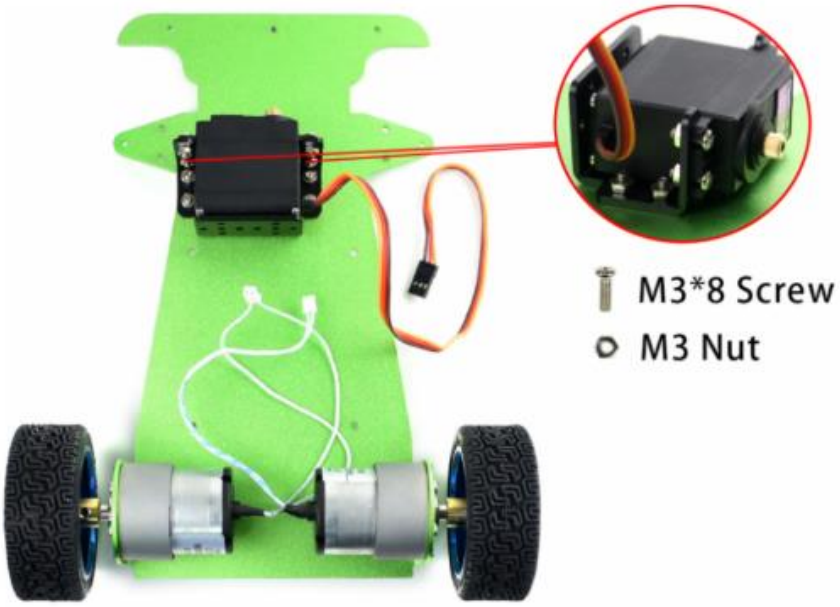
Dotiahnite čiernu skrutku, aby ste spojku upevnili na plochej strane nápravy

4. Držiak serva namontovať na kovové šasi



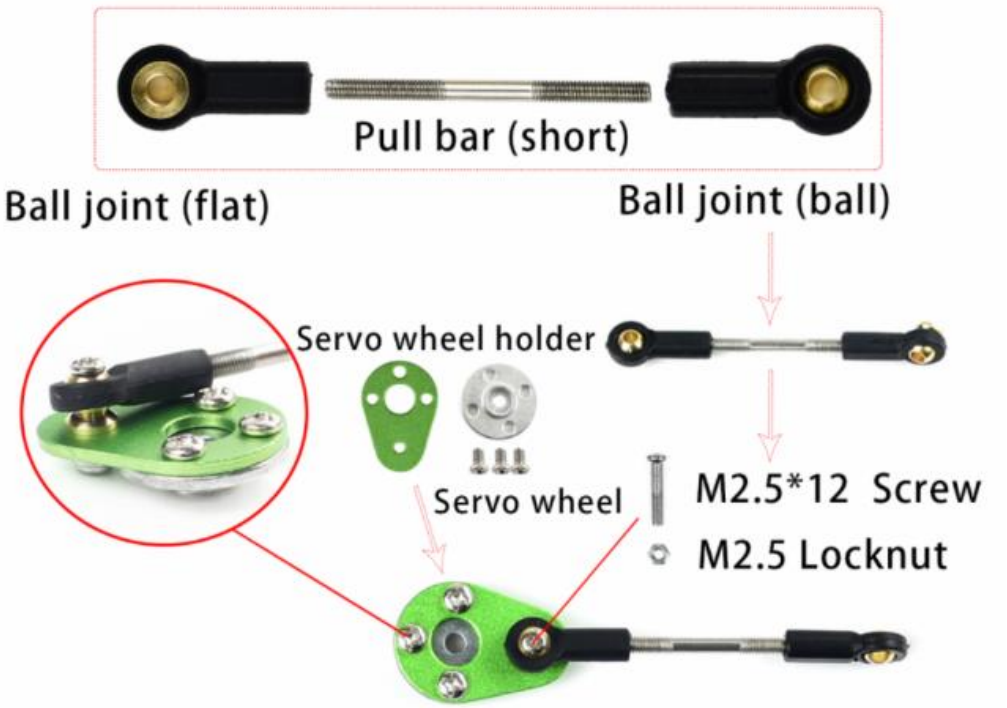
5. Pripevniť servo na držiak pomocou skrutiek a matíc

Uistite sa, že je servo správne zasunuté. Vonkajší hriadeľ by mal byť v strede.



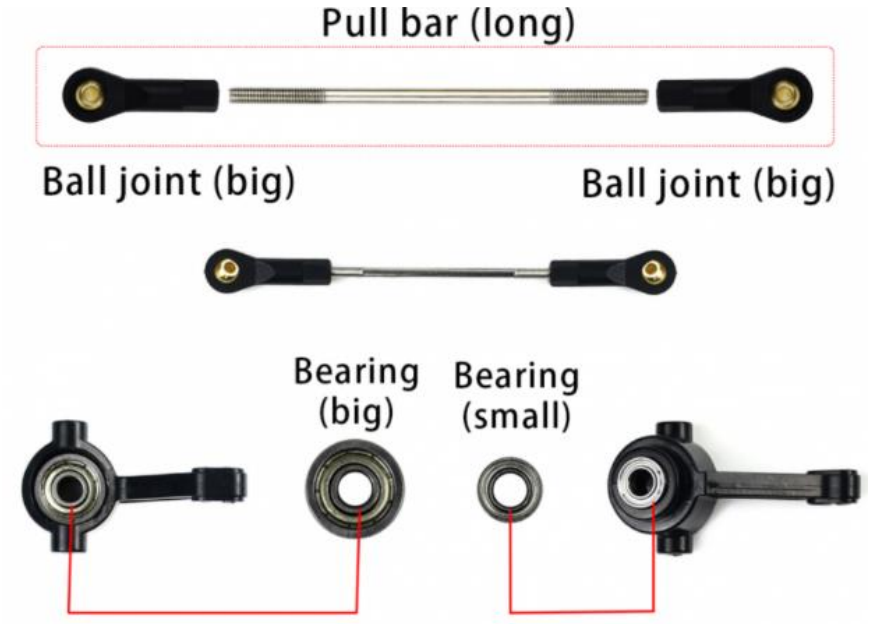
6. Zostaviť servo ťahadlovú tyč

Ťahacia tyč je kombinovaná pomocou dvoch guľových čapov, jedného plochého a jedného guľového, a krátkej tyče. Dva guľové čapy by mali byť na seba kolmé. Pripevnite servo koleso (trúbku) na držiak servo kolesa pomocou skrutiek, ktoré boli súčasťou servo kolesa. Potom pripojte plochý guľový čap k držiaku servokolia. Všimnite si, že drážka servokolia je smerom von.



7. Zostaviť ťahalo predného kolesa a riadiace kĺby

Predné ťahadlá sú kombinované s dvoma guľovými čapmi a dlhou tyčou. Potom daj ložiská do riadiacich kĺbov. Každý kĺb potrebuje malé aj veľké ložisko.



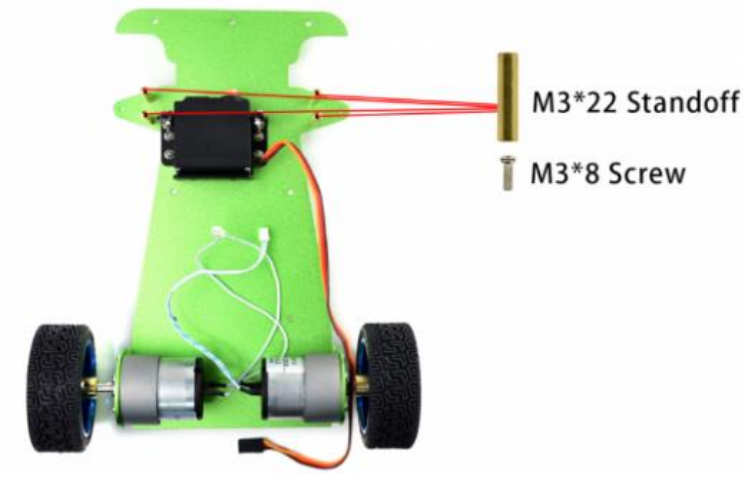
8. Zostaviť servo ťahaciu tyč, prednú ťahaciu tyč a riadiace kĺby

Pripevnite servo ťahadlovú tyč hore, potom predné koleso a nakoniec kĺby. Väčšie ložisko by malo byť bližšie dovnútra.

9. Pripevnite kolesá na riadiaci kĺb

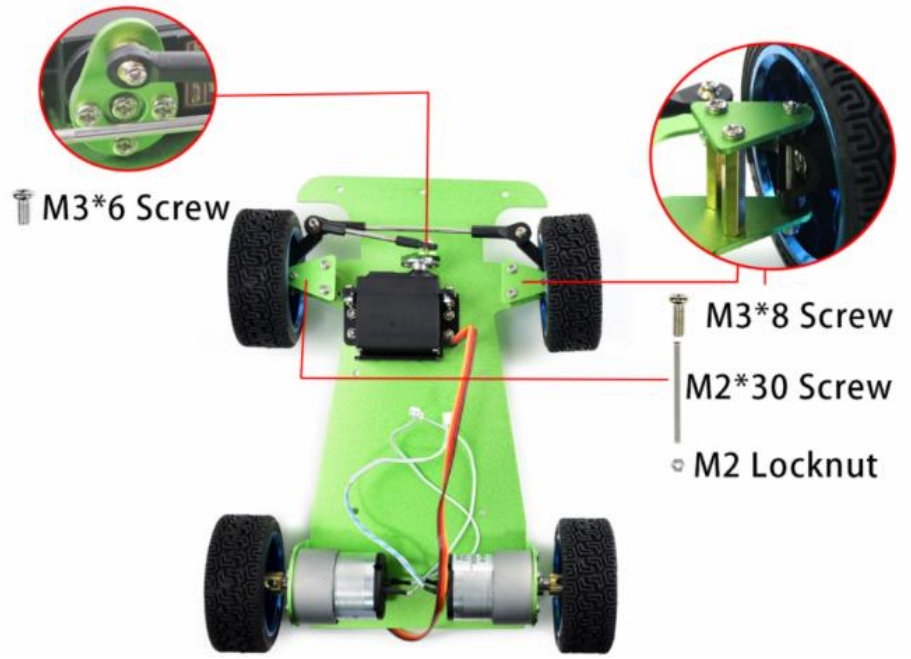
Matica by mala byť na vonkajšej strane kolesa, oproti kĺbu. Uistite sa, že koliesko nie je príliš tesné ani príliš voľné. Otestujte koleso, či sa môže voľne pohybovať.

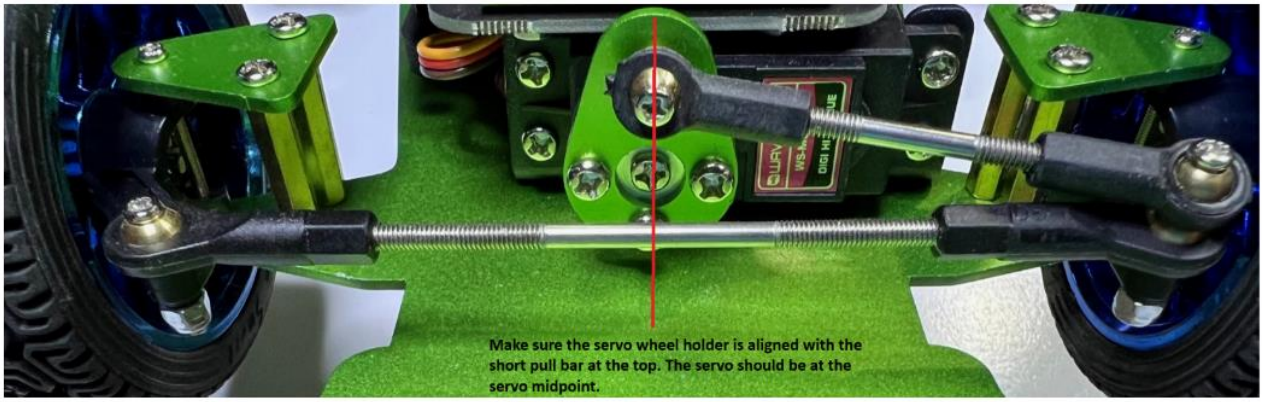
10. Pridajte odstojníky M3 pre predné kolesá



11. Zostaviť kombináciu predných kolies

Polož servo koleso na servo, upevni ho skrutkou M3. Kolesá upevnite skrutkami M2, maticou a trojuholníkovou doskou. Predné kolesá by mali byť rovné dopredu. Ak je to potrebné, upravte dlhú ťahaciu tyč.

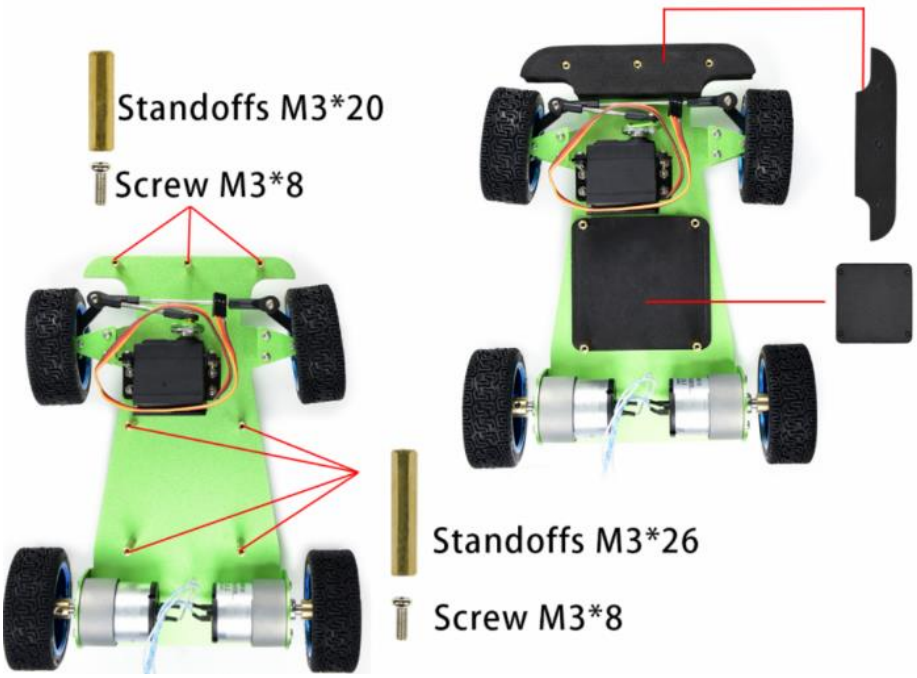




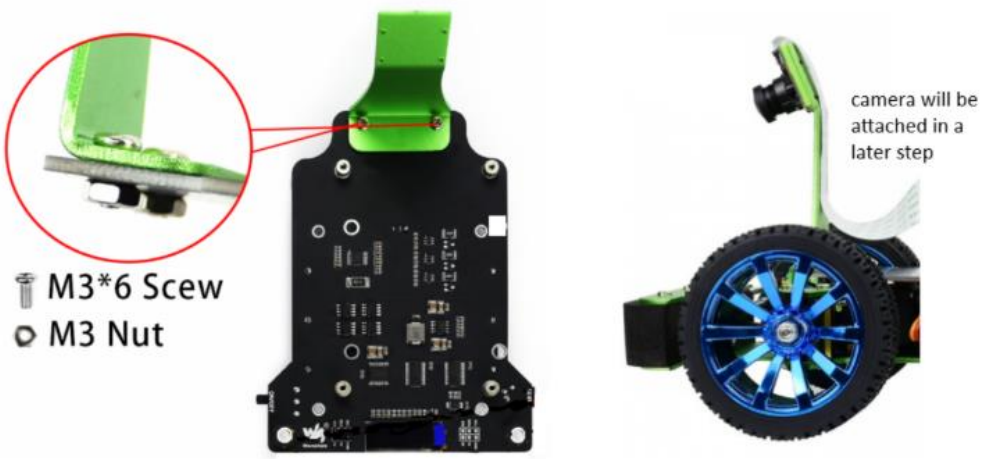
Predné kolesá by mali byť rovné dopredu. Ak je to potrebné, upravte dlhú ťahaciu tyč.

12. Pridaj diaľkové odstupy pre rozšíriacu dosku PiRacer a nárazník

Vložte EVA filcové podložky na nárazník a rozširujúcu dosku PiRacer.

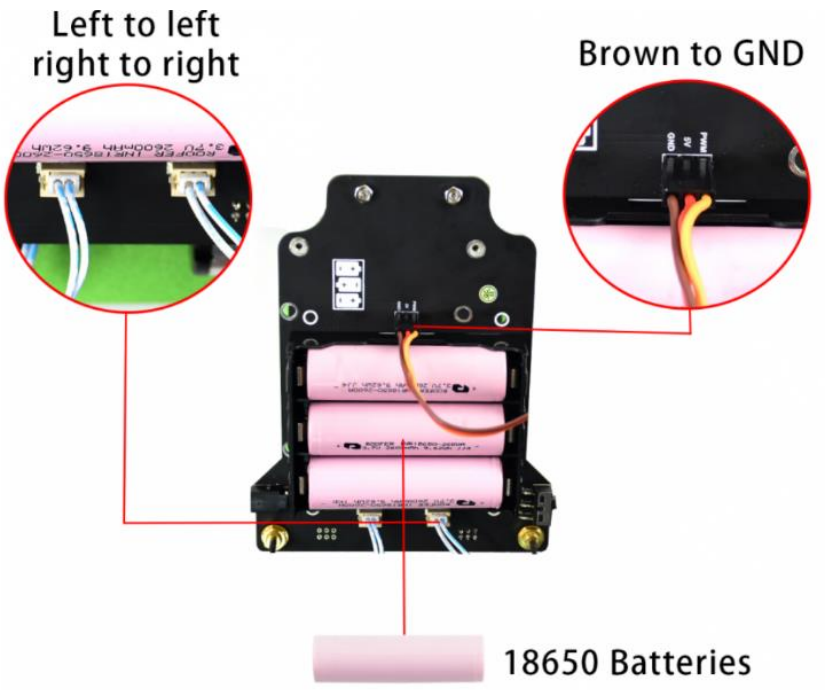


13. Pripojte držiak fotoaparátu na rozširujúcu dosku PiRacer.



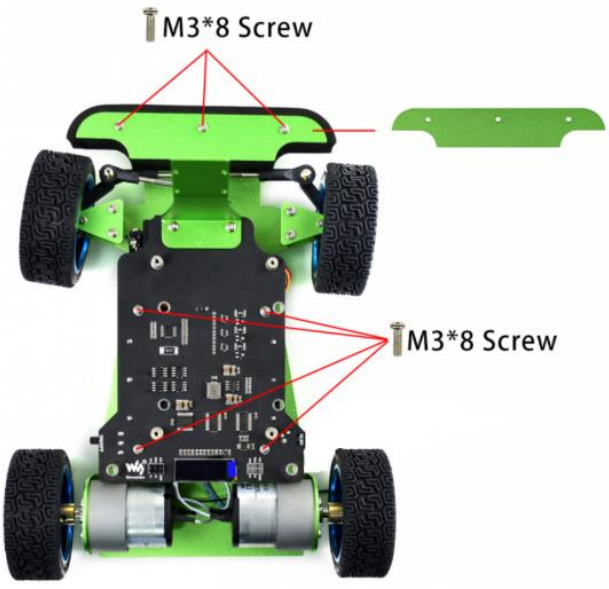
14. Vložte batérie správnym smerom

Pripojte vodiče motorov a servo k rozširujúcej doske PiRacer.



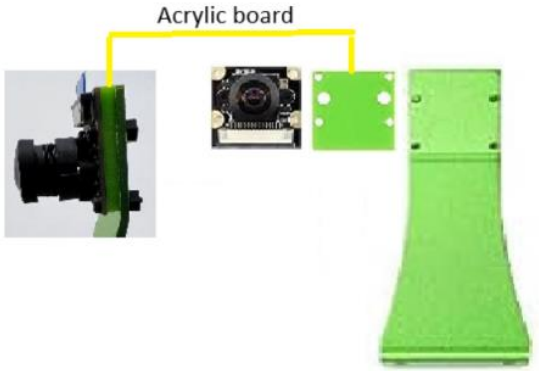
15. Pripojte rozširnú dosku a kovový nárazník

Upravte umiestnenie vodičov motora a servo vodičov, pripevnite rozširujúcu dosku PiRacer na kovové šasi a pripevnite kovový nárazník.

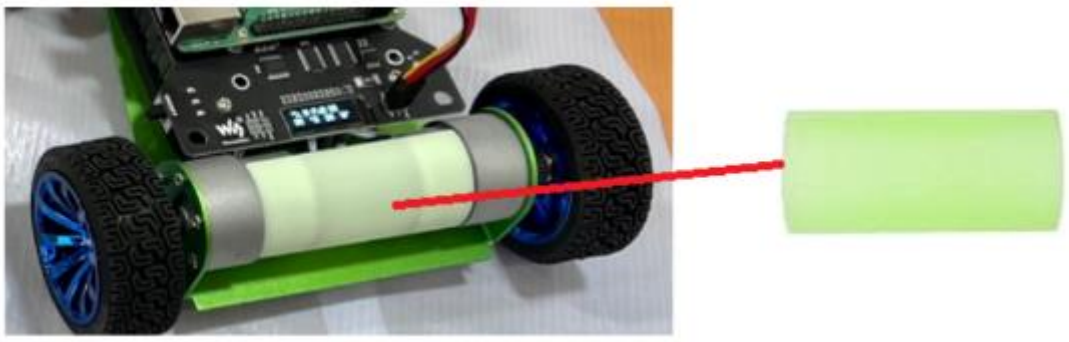


16. Pripevniť fotoaparát na držiak pomocou nylonových skrutiek

Poznámka: Akrylová doska by mala byť medzi fotoaparátom a kovovým držiakom, aby sa predišlo skratu.



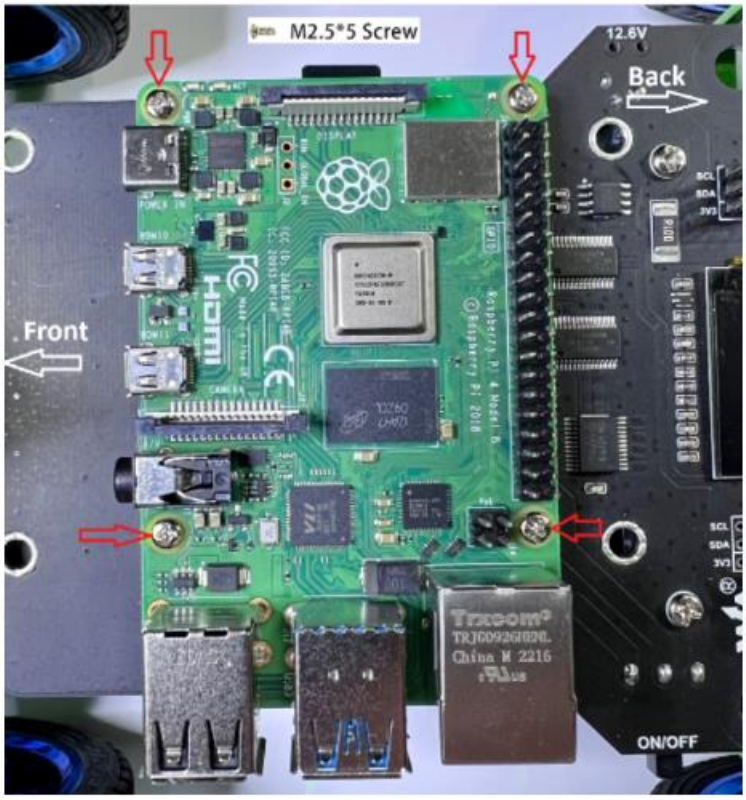
17. Pripojte 3D tlačený kryt motora na jednosmerné motory



18. Pred pokračovaním vypnite Raspberry Pi a odpojte nabíjačku z rozširujúcej dosky PiRacer.

19. Pripojte Raspberry Pi na rozšíriacu dosku PiRacer

GPIO piny by mali byť vzadu na aute.

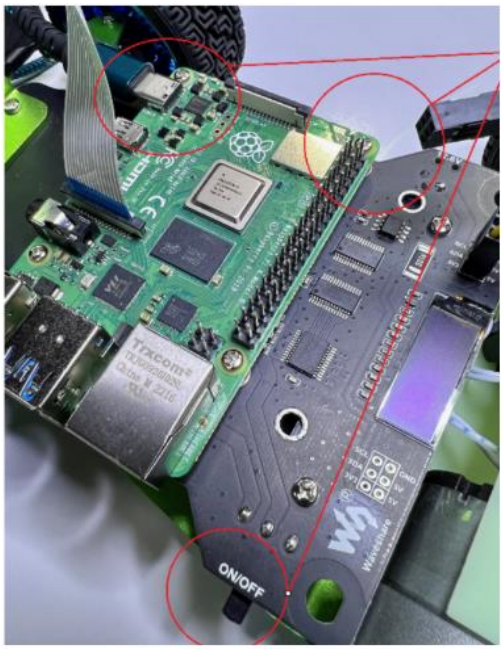


21. Pripojte páskový kábel kamery k vstupu kamery Raspberry Pi

Modrá strana smerom k USB portom Pi.

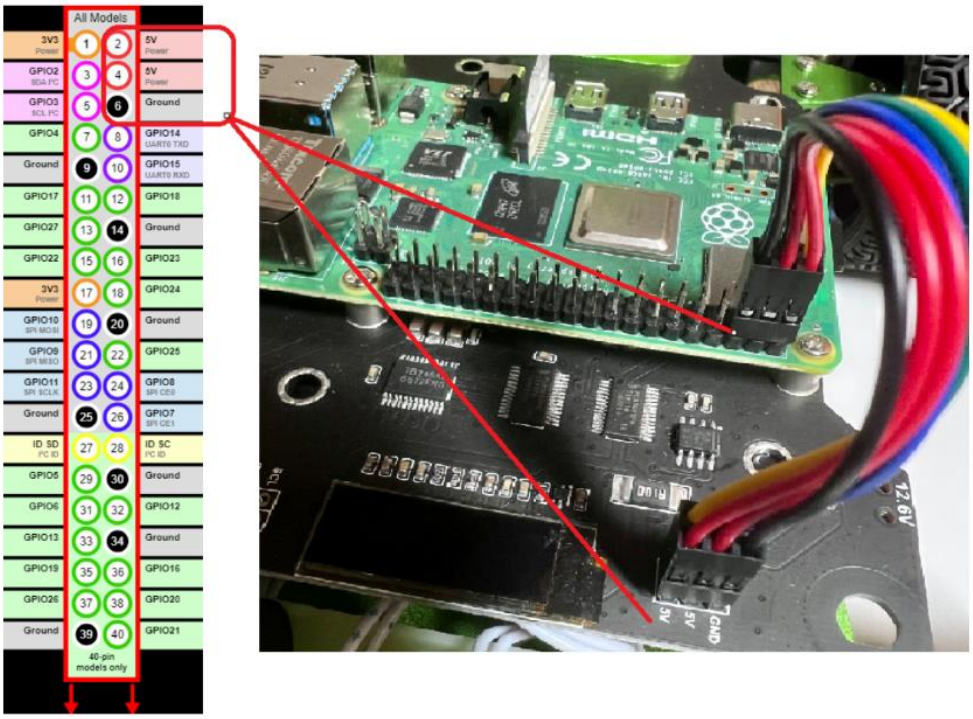
****UPOZORNENIE** Uistite sa, že Raspberry Pi nie je napájané pri pripájaní 6-pinových vodičov**

Tiež nenapájajte Raspberry Pi cez USB-C (externé napájanie), keď je napájané rozširujúcou doskou PiRacer.



22. Pripojiť Raspberry Pi k rozširiavacej doske PiRacer pomocou 6-pinových vodičov

Zladiť červenú|červenú|čiernu (5V|zem) na Pi GPIO a čiernu|červenú|červenú (gnd|5V|5V) na rozšírovacej doske PiRacer.



1.2 Raspian Legacy (Buster) Desktop

Flash OS - Raspian Legacy (Buster) Desktop

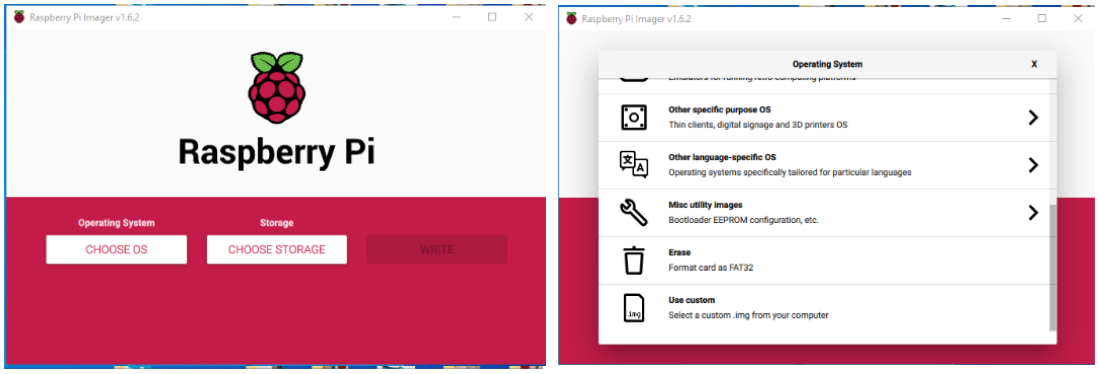
Prejdite do oficiálneho repozitára na stiahnutie Buster OS. **Všetky staršie verzie Raspberry Pi OS nájdete a stiahnete tu a** oficiálny odkaz **na stiahnutie predchádzajúceho Raspberry Pi 'Buster' OS** je tu. Nižšie sa pozrite, ako to vyzerá, stiahnite si obrázok kliknutím na zvýraznený súbor.

Index of /raspios_armhf/images/raspios_armhf-2021-05-28

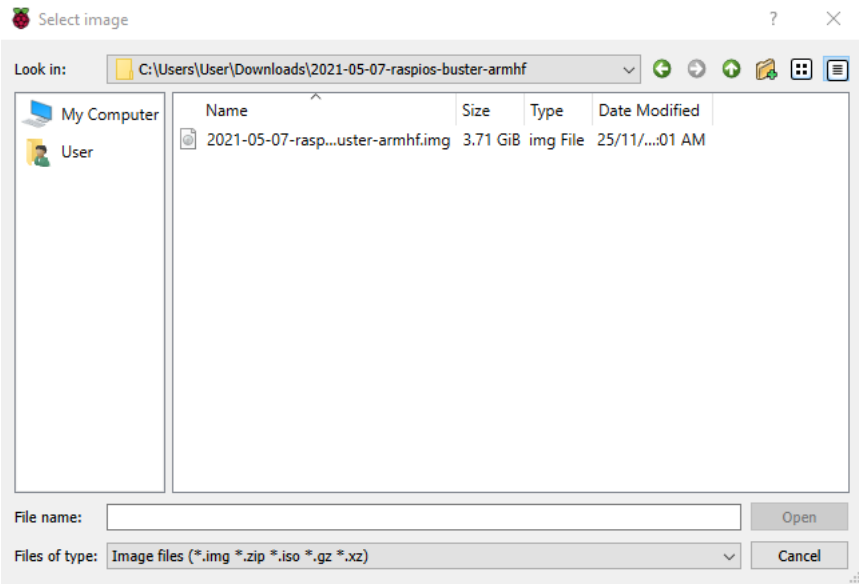
Name	Last modified	Size	Description
Parent Directory		-	
2021-05-07-raspios-buster-armhf.info	2021-05-07 16:07	188K	
2021-05-07-raspios-buster-armhf.zip	2021-05-07 16:12	1.2G	
2021-05-07-raspios-buster-armhf.zip.sha1	2021-05-28 15:45	78	
2021-05-07-raspios-buster-armhf.zip.sha256	2021-05-28 15:45	102	
2021-05-07-raspios-buster-armhf.zip.sig	2021-05-28 15:00	488	
2021-05-07-raspios-buster-armhf.zip.torrent	2021-05-28 15:45	23K	

Stiahnite a nainštalujte Raspberry Pi Imager z https://www.raspberrypi.com/software/

Teraz otvorme oficiálny Raspberry Pi Imager, pozrite si ho na obrázku nižšie. Stojí za zmienku – ak narazíte **na | CTRL+SHIFT+X |** na klávesnici počas používania programu Raspberry Pi Imager sa otvorí Pokročilé skryté menu. Toto skryté menu vám umožňuje predkonfigurovať Raspberry Pi pomocou SSH, WIFI prihlasovacích údajov a nastavení lokalizácie.



Keď to urobíte, otvorí sa prehliadač súborov. Prejdi na ten extrahovaný súbor obrazu disku a klikni naň. Pozrite si to na obrázku nižšie. Potom otvorte obrázok súboru a tým sa Raspberry Pi Imager vybaví operačným systémom Raspberry Pi 'Buster'. Nájdite ho, vyberte a otvorte.



Vlož Micro-SD kartu, ktorú chceš nahrať do počítača. Ak je to potrebné, použite **adaptér USB na Micro-SD**. Potom kliknite na **| VYBERTE ÚLOŽISKO |** a vyberte vloženú Micro-SD kartu. Majte na pamäti, že všetky dáta na vašej Micro-SD karte budú pri flashovaní vymazané alebo trvalo vymazané. Oficiálny Raspberry Pi Imager bude teraz vyzerať ako na obrázku nižšie.

Keď je všetko zoradené (správny operačný systém načítaný a správne úložisko vybrané), môžete teraz kliknúť na **| WRITE |** tlačidlo na spustenie procesu blikania. Pozrite si tento proces blikania na obrázku nižšie.

Keď je blesk dokončený, automaticky virtuálne vysunie Micro-SD kartu z počítača. Takže potom môžete jednoducho fyzicky vybrať svoju Micro-SD kartu a vložiť ju do Raspberry Pi jednodoskového počítača. Potom nastavte Raspberry Pi normálne ako stolný počítač. Po spustení vás privíta staré známe pozadie, pozri obrázok nižšie, a úspešne ste nainštalovali 'Buster' OS na svoj Raspberry Pi. Teraz ste slobodní pohybovať sa po dobre známom digitálnom prostredí.

Postupujte podľa pokynov Pi Wizard na nastavenie polohy, klávesnice, wifi a získajte aktualizáciu

Menu / Preferencie / Konfigurácia Raspberry Pi / Rozhrania

- povoliť kameru a I2C
- voliteľné: povoliť VNC pre vzdialený prístup
- Kliknite OK a reštartujte

Dodatočný softvér

Java 3.8.3 (spúšťaj po jednom)

```
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
```

Ďalšie nástroje

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

AWS CLI

Predvolené používateľské účty na Raspberry Pi

```
sudo apt install awscli -y
Sudo Pip3 Install --Upgrade AWSCLI
Sudo Pip3 Install Boto3

Používateľ - Pi
PW - Raspberry
```

1.3 WaveShare vetva pre softvér DonkeyCar

Inštalačné závislosti

```
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
```

Voliteľné?

```
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
```

Nastavenie virtuálneho prostredia

```
python3 -m virtualenv -p python3 env --system-site-packages
Echo "source ~/env/bin/activate" >> ~/.BASHRC
zdroj ~/.BASHRC
```

Nainštalujte DonkeyCar Python kód - vetva WaveShare pre softvér DonkeyCar 3.1.0

```
 projekty mkdir
CD projekty
Git klon https://github.com/waveshare/donkeycar

CD Donkeycar
git checkout master
pip install -e .[pi]

Pip install tensorflow==1.13.1
```

Nie je potrebné pip install numpy --upgrade

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

Testová verzia tensorflow - mala by ukazovať verziu 1.13.1

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

POZNÁMKA: Model na aute nebolo možné bežať, kým som nepoužil nasledujúce dva modely

```
 Pip inštalácia 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
```

Edit camera.py pridať self.camera.hflip = True

```
sudo nano /domov/pi/projekty/donkeycar/donkeycar/diely/kamera.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
```

Voliteľná inštalácia OpenCV

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

Testujte s:

python -c "import cv2"

1.4 Začiatky s DonkeyCar

Otvorte okno terminálu, zadajte nasledujúci príkaz

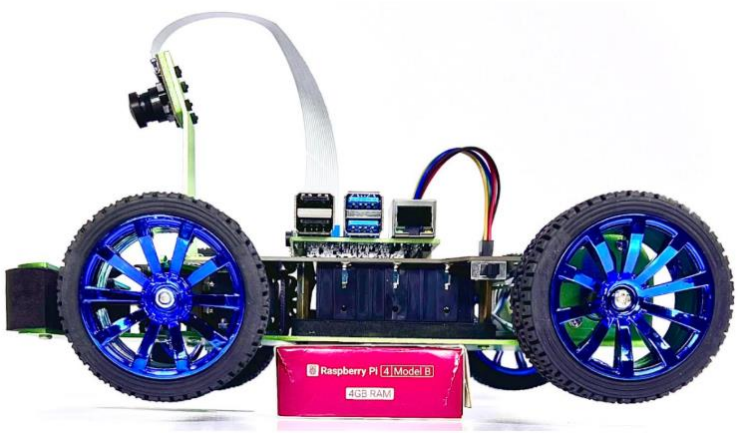
Create DonkeyCar App

Tým sa vytvorí priečinok s názvom mycar so všetkým python kódom potrebným na jazdu auta.

Kalibrujte predné riadenie

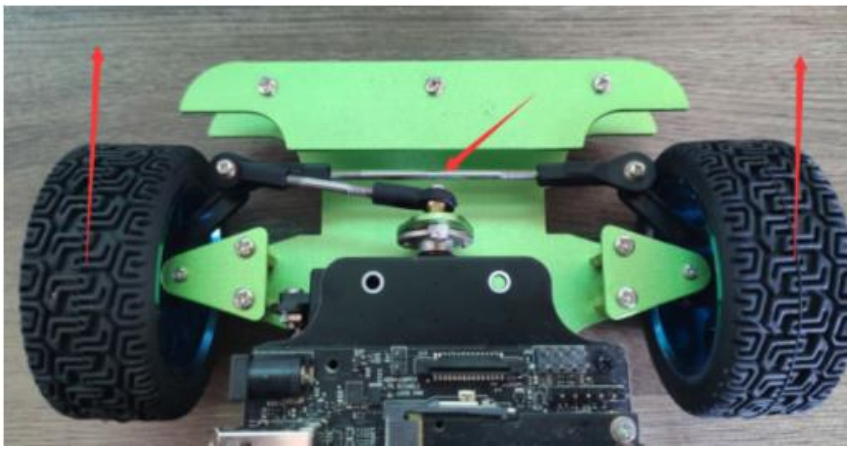
Uistite sa, že vaše auto je nad zemou, aby ste predišli situácii bez kontroly.

Použi malú krabičku, ako bola na Raspberry Pi.

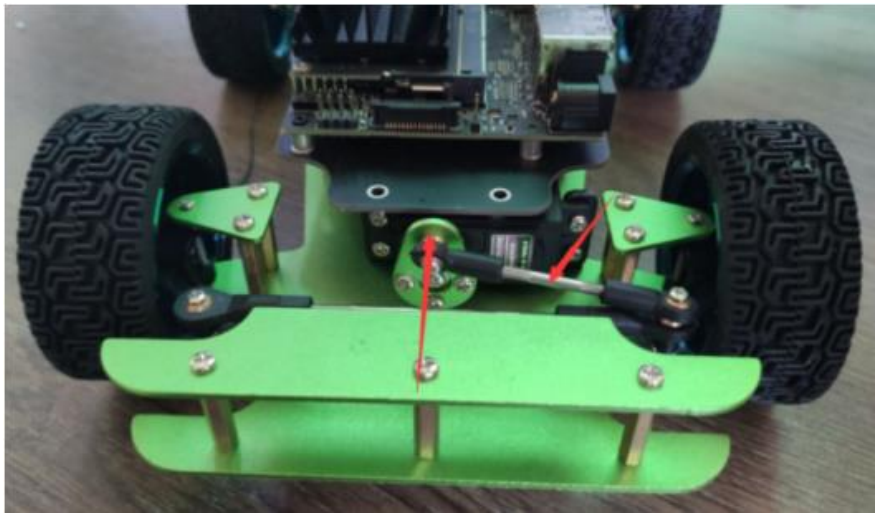


Aby DonkeyCar mohol jazdiť rovno a robiť potrebné zákruty na trati, je potrebné kalibrovať hardvér a softvér auta.

Predné kolesá by mali byť rovné dopredu. Ak je to potrebné, upravte dlhú ťahadlovú tyč



Držiak servo kolies by mal byť zarovnaný s krátkou ťahadlovou tyčou hore. Ak treba, nastavte krátku ťahačku.



Kalibrujte servo softvér

Nájdite ľavý, stredný a pravý riadiaci PWM pre servo tohto auta.

Stred by mal byť v polovici cesty medzi ľavou a pravou stranou. Príklad:

Ľavý 200
pravá 560
Centrum bude 380

V terminálovom okne vstúpte nasledujúce

```
cd ~/mycar
Donkey kalibruj --kanál 0 --zbernica=1
```

Skús hodnoty 300, 400, 500 a uvidíš, ako sa mení riadenie. Zistite si maximálny počet otáčok doľava a doprava.

Keď budete mať čísla, upravte config.py a aktualizujte hodnoty, ktoré ste našli.

Plyn by mal byť nastavený podľa vašich čísel pre riadenie. Plyn je už správne nastavený.

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
```

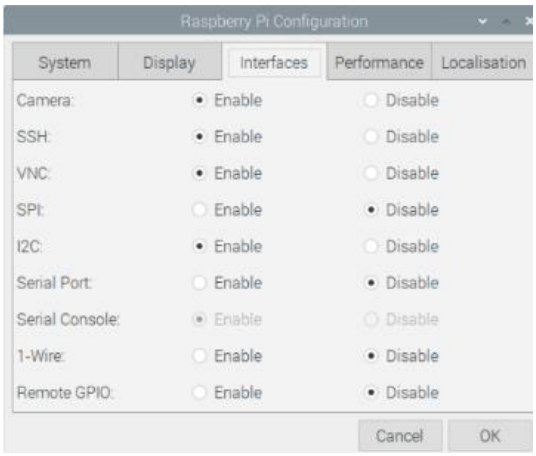
Inštalácia OLED zobrazovacej služby

```
cd ~
Git klon https://github.com/waveshare/pi-display
CD Pi-displej
sudo ./install.sh
cd ~
```

2. VZDIALENÝ PRÍSTUP K RASPBERRY PI POMOCOU REALVNC

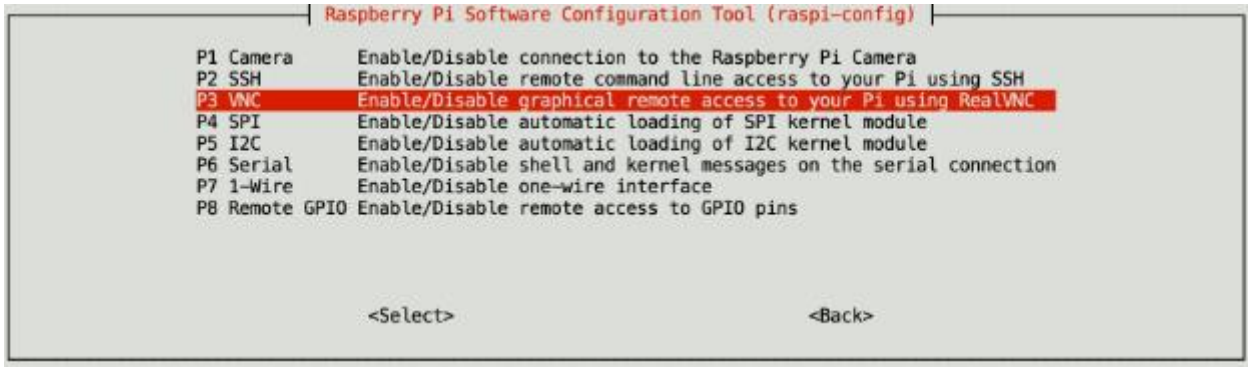
2.1 Povoliť VNC v Raspian OS pomocou a alebo b:

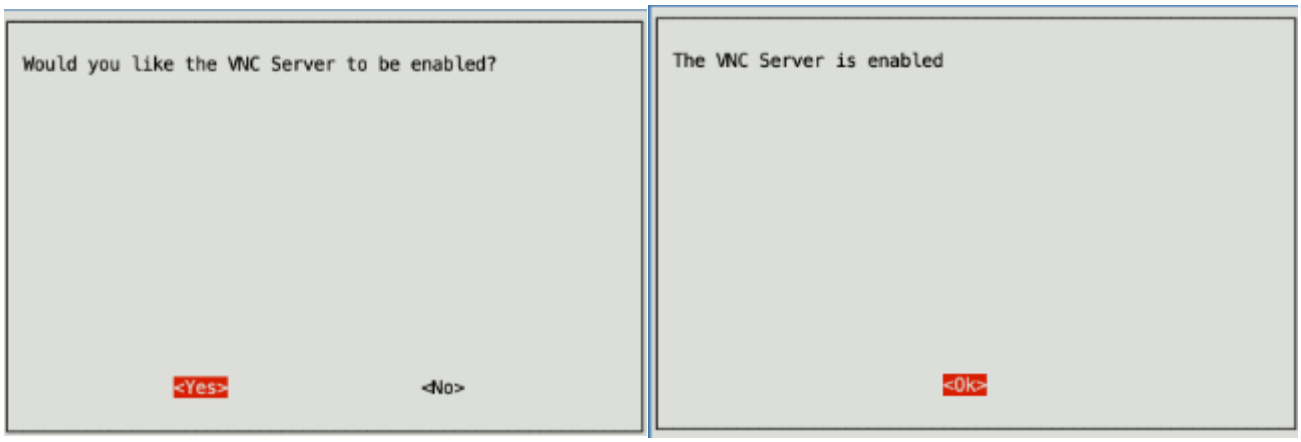
a) Povoliť VNC v nastaveniach - Konfigurácia Raspberry PI – Rozhrania



b) Ďalšou metódou na povolenie VNC je použitie Terminalu, zadajte príkaz

```
sudo raspi-config
```

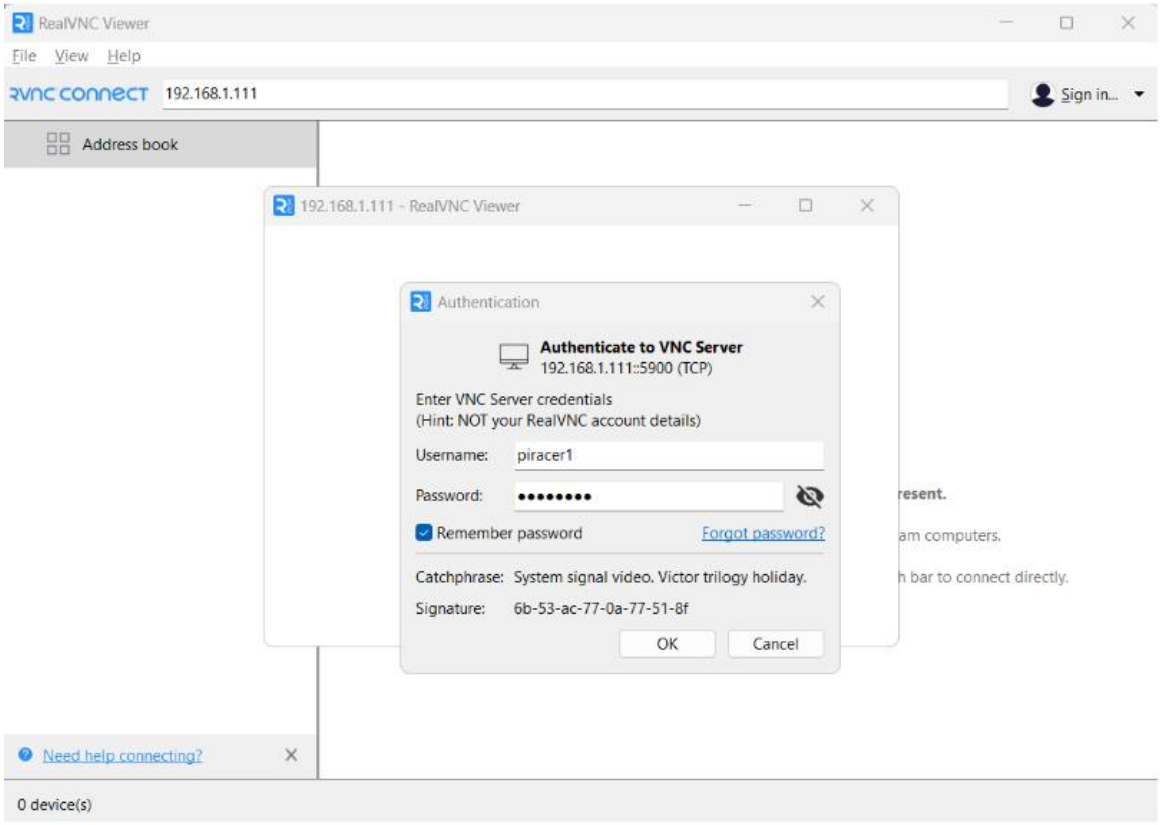




2.2 Inštalácia VNC prehliadača

Na počítači si budete musieť nainštalovať VNC prehliadač, aby ste sa mohli pripojiť k Raspberry Pi. K dispozícii je niekoľko prehliadačov, ale najjednoduchší na nastavenie je **Real VNC Viewer**. Inštalátory pre Windows a Mac si môžete stiahnuť odtiaľto: https://www.realvnc.com/en/connect/download/viewer/

Otvorte Real VNC Viewer a zadajte IP adresu Piracer (prečítajte ju z displeja auta)



3. ZAČNITE ŠOFÉROVAŤ A ZBIERAŤ DÁTA

Naštartuj auto

```
cd ~/mycar
Python Manage.py drive
```

Tento skript spustí drive loop vo vašom aute, ktorý obsahuje časť webového servera na ovládanie auta. Teraz môžete ovládať svoje auto cez webový prehliadač na URL: <hostname.local>:8887

Otvorte prehliadač a pripojte sa k DonkeyCar Monitoru na localhost**:8887**

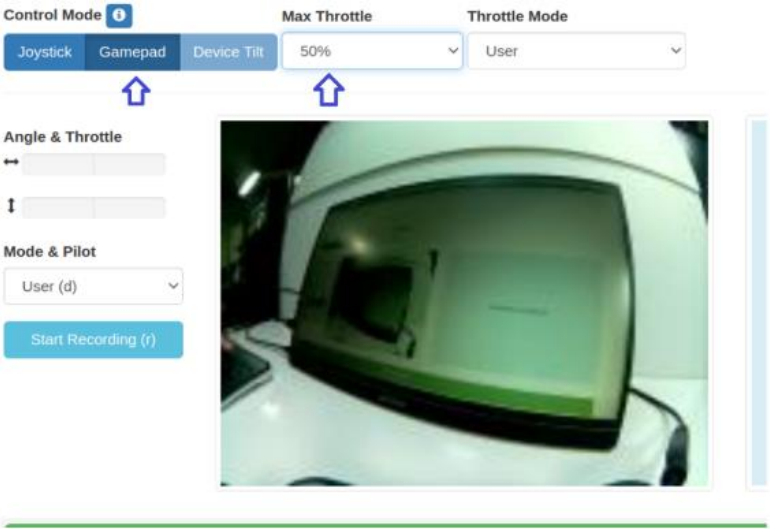
POZNÁMKA: Keď sa auto naštartuje, vytvorí sa priečinok pod /mycar/data s názvom tub_#_date na uloženie dát pre reláciu. Údaje sa zbierajú, keď je plyn zapnutý. Aby ste získali čistý dataset, najlepšou praxou je zastaviť "python manage.py drive" pomocou **Contole + C** a reštartovať, aby ste začali trénovanie s novým /mycar/data/tub.

Existujú 2 možnosti pohonu

1. Používajte gamepad (odporúčané)

Po naštartovaní auta otvorte webový prehliadač na ploche Raspberry Pi, kým je auto pripojené k monitoru.

Vyberte Gamepad a nastavte plyn na 50 %, aby ste mohli začať trénovať. Otestujte zákruty a rýchlosť, keď je auto na Pi boxe.

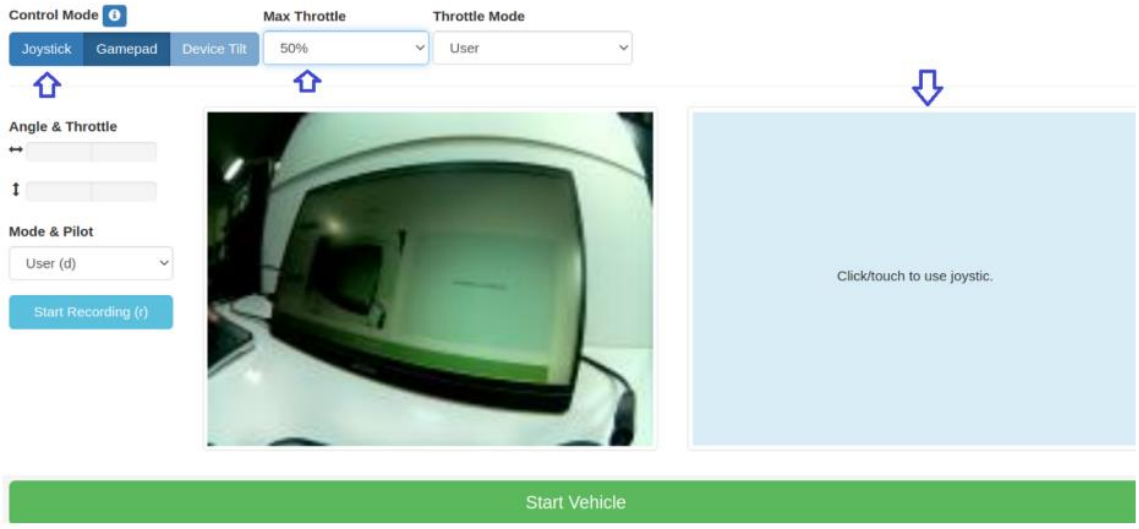


Keď ste pripravení jazdiť na trati, odpojte monitor, položte auto na trať a začnite jazdiť. Začni pomaly, kým sa nebudeš cítiť pohodlne s ovládaním auta.

2. Použite joystick v webovom prehliadači

Budete musieť použiť tablet alebo telefón pripojený na rovnakú wifi. Po naštartovaní auta použite tablet alebo telefón na pripojenie k webovému serveru auta pomocou IP:8887. Auto by malo zobrazovať svoju IP adresu, ak bola v predchádzajúcom kroku povolená OLED Display Service.

Vyberte joystick (oblasť ovládania joysticku je označená klik**/dotyk na použitie joystic**). Použite oblasť joysticku na ovládanie auta prstom.



Po dokončení jazdy zastavte "python manage.py drive" pomocou **Contole + C**.

4. TRAIN DATA - VYTVORENIE INFERENČNÉHO MODELU

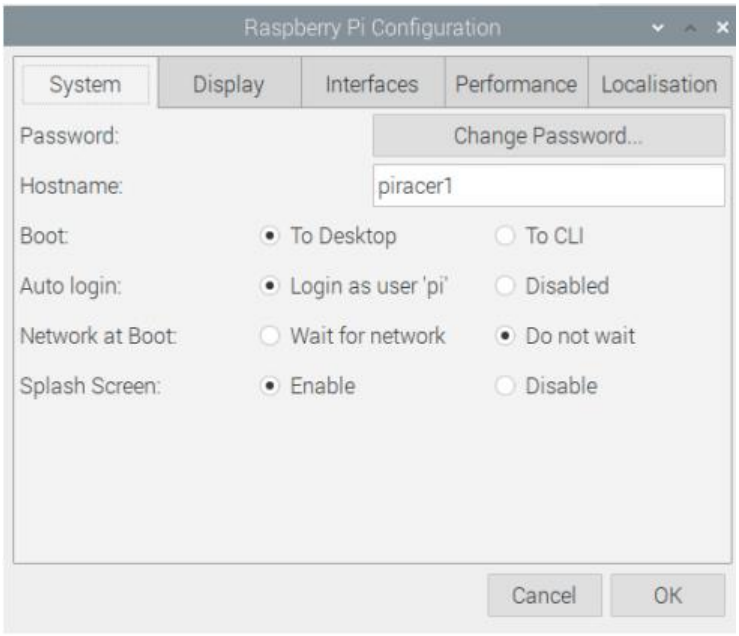
Dátová lokalita je /home/pi/mycar/data. Všetky priečinky s nádobami v tomto adresári sa použijú na vytvorenie vášho modelu. Odstráňte všetko, čo nechcete, aby bolo zahrnuté. Dáta môžete trénovať cez Raspberry Pi alebo cez AWS virtuálny stroj.

1. Trénujte s Raspberry Pi

Train Data:
V novej terminálovej relácii na vašom hostiteľskom PC použite rsync na skopírovanie priečinka cars z Raspberry Pi

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

Názov hostiteľa nájdete v konfigurácii Raspberry Pi:



Train model:
V tom istom termináli teraz môžete spustiť trénovací skript na najnovšom tubu tak, že cestu k tomu tubu pošlete ako argument. Voliteľne môžete predávať masky cesty, ako ./data/* alebo ./data/tub_?_17-08-28, aby ste získali viacero tubov. Napríklad:

```
python  ~/mycar/manage.Py train --tub <názvy priečinkov tub oddelené čiarkou> --model
./models/mypilot.H5
atď:
python  ~/mycar/manage. Py train --tub ./data/tub_1_24-04-06 --model
./models/mypilot.H5
```

Trénovanie modelu bude stáť dlhý čas, prosím, buďte trpezliví. Po trénovaní môžete získať model, musíte modul skopírovať do Raspberry Pi a otestovať ho.

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

5. AUTONÓMNE NAČÍTANIE MODELU A POHONU

Začnite svoje auto s modelom

```
cd ~/mycar
Python Manage.PY Drive --Model ~/MyCar/Models/MyPilot.H5
```

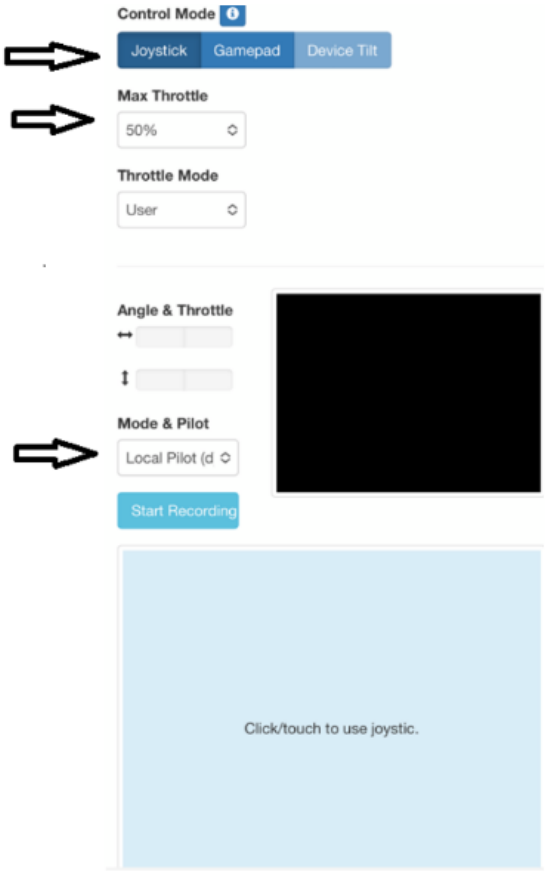
Na autonómnu jazdu alebo riadenie použite joystick v prehliadači na inom zariadení.

Budete musieť použiť tablet alebo telefón pripojený na rovnakú wifi. Po naštartovaní auta použite tablet alebo telefón na pripojenie k webovému serveru auta pomocou IP:8887. Auto by malo zobrazovať svoju IP adresu, ak bola v predchádzajúcom kroku povolená OLED Display Service.

Autonómne riadenie

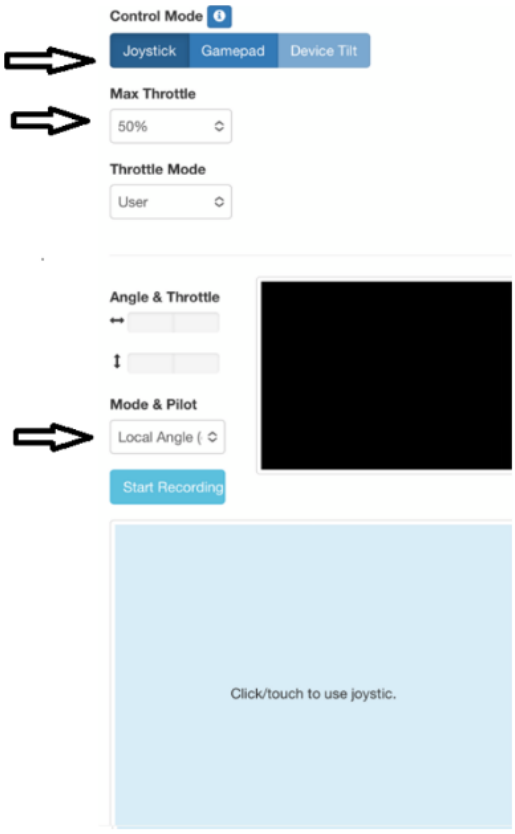
POZNÁMKA: Akonáhle je **vybraný miestny pilot**, auto začne autonómne jazdiť

- **Režim ovládania:** Vyberte **Joystick** (oblasť ovládania joysticku je označená **klik/dotyk na použitie joystic**)
- **Maximálny plyn: Nastavte** maximálny plyn na približne 50 %, aby auto nešlo príliš rýchlo
- **Režim a pilot:** Vyberte miestneho pilota (Miestny pilot je autonómny)



Autonómne riadenie

- **Režim ovládania:** Vyberte **Joystick** (oblasť ovládania joysticku je označená **klik/dotyk na použitie joystic**)
- **Maximálny plyn: Nastavte** maximálny plyn na približne 50 %, aby auto nešlo príliš rýchlo
- **Režim & Pilot:** Vyberte Lokálny uhol (Lokálny uhol je riadenie auta, zatiaľ čo vy pridávate plyn)



Rýchlejší autonómny model?

Použite svoj funkčný model autonómnej jazdy na získavanie nových dát rýchlejšie. Použi Local Angle, aby auto riadilo, zatiaľ čo ty poskytneš rýchlejší plyn.

II. AI S AR/VR

6. ÚVOD

Meta Quest 3 predstavuje zásadný posun od čistej virtuálnej reality (VR) k vysokokvalitnej zmiešanej realite (MR). Využitím vysokorozlíšeného farebného priechodu a výrazne výkonnejšieho čipsetu Snapdragon XR2 Gen 2 umožňuje digitálnemu obsahu bezproblémovo koexistovať s fyzickým svetom. So svojou štíhlejšou "palacinkovou" optikou a 4K+ Infinite Display je momentálne najprístupnejšou silnou spoločnosťou pre spotrebiteľov aj vývojárov.

6.1 Úvod do Meta Quest 3 a zmiešanej reality

Meta Quest 3 využíva technológiu Color Passthrough s vysokým rozlíšením. Na rozdiel od tradičnej VR, kde je svet úplne virtuálny, **Mixed Reality (MR)** používa palubné kamery na premietanie reálneho prostredia a potom na to prekrýva **prvky WebGL**. Váš kód využíva režim immersive-ar, ktorý je jadrom tohto priestorového zážitku.

Systémová architektúra (webový stack)

Aplikácia funguje ako "sendvič" vrstiev:

- **Vrstva 1 (hardvér):** Snímače Quest 3, hĺbkové projektory a RGB kamery.
- **Vrstva 2 (prehliadač):** Meta Quest Browser (založený na Chromiu s podporou WebXR).
- **Vrstva 3 (API):** WebXR Device API, ktoré slúži ako most medzi hardvérom a kódom.
- **Vrstva 4 (Logika**): Vaša JavaScriptová logika využívajúca Three.js na renderovanie a TensorFlow.js na videnie.

7. WEBXR: ALTERNATÍVA ZALOŽENÁ NA PREHLIADAČI

WebXR umožňuje vývojárom vytvárať pohlcujúce zážitky, ktoré bežia priamo v prehliadači Meta Quest. Je postavený na štandardných webových technológiách, vďaka čomu je "VR na webe" rovnako prístupný ako webová stránka.

Prečo si vybrať WebXR?

- Bezproblémový prístup: Používatelia nemusia sťahovať veľké súbory z Meta Store; jednoducho kliknú na URL a kliknú na "Enter VR."
- Frameworky: Využíva výkonné JavaScriptové knižnice ako Three.js, A-Frame (HTML) alebo Babylon.js.
- Multiplatformové: Jedna WebXR aplikácia často beží na Quest 3, Android telefóne (AR režim) alebo stolnom počítači.

7.1 Analýza knižnice

Three.js (Renderovací engine)

Three.js zvláda ťažkú prácu WebGL. Spravuje scénu, kameru a renderer. Vo vašom kóde je renderer nastavený na alfa: true, čo je pre AR kľúčové, pretože umožňuje "prázdnym" častiam prehliadača stať sa oknom do reálneho sveta.

TensorFlow.js & COCO-SSD

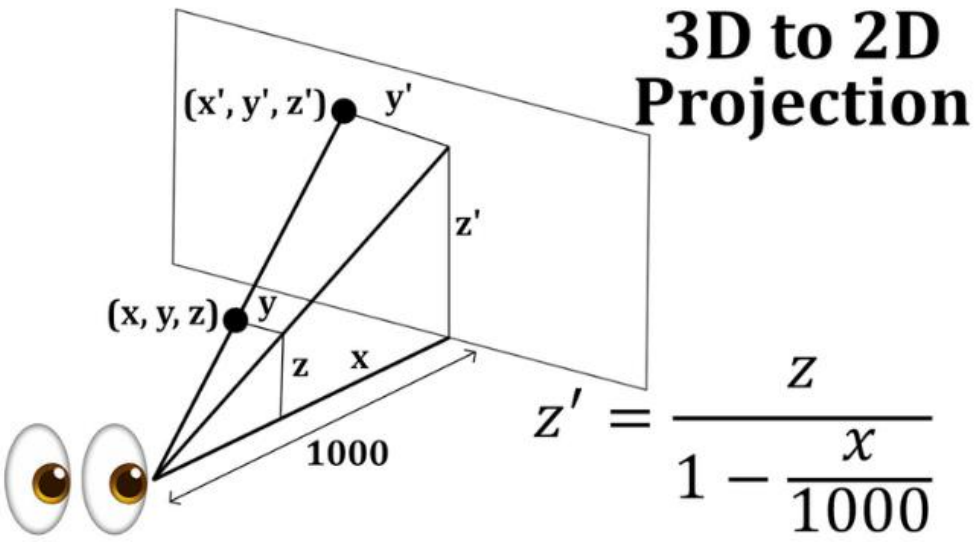
COCO-SSD (Common Objects in Context - Single Shot MultiBox Detector) je model trénovaný na rozpoznávanie 80 tried objektov. Krásou TF.js je jeho **WebGL backend**, ktorý zabezpečuje, že výpočty AI sa vykonávajú na GPU Questu, nie na CPU, čím zabraňuje prehrievaniu zariadenia.

7.2 Matematická projekcia (z 2D do 3D)

Toto je najtechnickejšia časť scenára. AI model vracia bbox (ohraničujúci rámec) v pixeloch (napr. x=100, y=200).

Funkcia detectionTo3D vykonáva neprojekciu:

- **Normalizácia**: Prevádza súradnice obrazovky na rozsah od -1 do $+1$.
- **Výpočet zorného poľa**: Zohľadňuje zorné pole kamery.
- **Vektorový lúč**: Vytvára smerový vektor od polohy hlavy používateľa smerom k objektu.
- **Umiestnenie**: Umiestni 3D označenie na tento vektor v definovanej vzdialenosti (DIST).



7.3 Implementácia testovania zásahov

Testovanie zásahov umožňuje aplikácii "vystreliť" neviditeľný lúč (raycast) do reálneho priestoru, aby detegovala prieniky s podlahami alebo stolmi. Keď ***hitTestSource*** vráti výsledok, kurzor (zelený kruh) sa prichytí do danej pozície (pozícia a orientácia).

7.4 AI pipeline: Detekcia a označovanie

V kóde sa detekcia nevykonáva každý snímok (čo by oneskorilo headset), ale každých $2000ms$ (DETECT_INTERVAL_MS).

- **Label Sprite**: Keďže Three.js nedokáže priamo vykresliť štandardné HTML fonty v 3D scéne, používame CanvasTexture. Text "kreslíme" na neviditeľné 2D HTML plátno a aplikujeme ho ako textúru na 3D sprite, ktorý je vždy otočený k používateľovi (billboarding).

7.5 Nutnosť HTTPS

WebXR a prístup do kamery (getUserMedia) sú dodávateľmi prehliadačov klasifikované ako "Výkonné funkcie". Fungujú len v zabezpečenom kontexte.

- **Výnimka localhost:** Môžete testovať na svojom PC pomocou http://localhost, ale hneď ako sa pokúsite pristupovať k serveru cez headset Quest 3, prehliadač zablokuje XR funkcie, pretože vidí nezabezpečenú sieťovú IP adresu (napr. http://192.168.1.10).
- **Riešenie**: Musíte použiť **tunelovaciu službu** alebo **zabezpečený hosting**.

7.6 Meta Quest Link & Vývojársky režim

Aby ste mohli efektívne spúšťať a ladiť svoj kód, váš Quest 3 musí byť uznaný ako vývojárske zariadenie.

Aktivácia krok za krokom:

- **Developerský účet**: Zaregistrujte sa na dashboard.oculus.comhttps://dashboard.oculus.com/. Budete musieť vytvoriť "Organizáciu" (môže mať akýkoľvek názov).
- Mobilná aplikácia: Otvorte aplikáciu Meta Quest na telefóne, choďte do **menu > Zariadenia > Nastavenia headsetu > Developer Mode** a zapnite ju.
- **Link**: Pripojte svoj Quest 3 k PC pomocou kvalitného USB-C 3.0 kábla alebo cez Air Link (vysokorýchlostné Wi-Fi 6).

7.7 Nadácia Three.js (Boilerplate)

Pred vstupom do AR musíme inicializovať štandardné 3D prostredie. Avšak pre Quest 3 sú dve nastavenia nevyhnutné:

- **Alpha & Antialias**:

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

- **Aktivácia XR**:

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

To Three.js povie, aby počúvali údaje o sledovaní hlavy Questu (6 DOF) a automaticky ich aplikovali na objekt 'kamera'.

7.8 Riadenie AR relácie (Životný cyklus)

Tlačidlo "Enter AR" spustí **navigator.xr.requestSession**. Tu definujeme, aké "superschopnosti" naša aplikácia potrebuje od hardvéru Quest 3.

Povinné a voliteľné funkcie:

- **lokálne poschodie**: Toto hovorí Questu, aby nastavil súradnicu $Y=0$ na skutočnej fyzickej úrovni poschodia.
- **HIT-TEST**: Umožňuje raycast proti reálnej geometrii.
- **detekcia roviny**: Žiada "sémantické" dáta (vie, ktorá sieť je "tabuľka" a ktorá "stena").

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

8. AR + AI DETEKCIA OBJEKTOV

8.1 Prehľad projektu

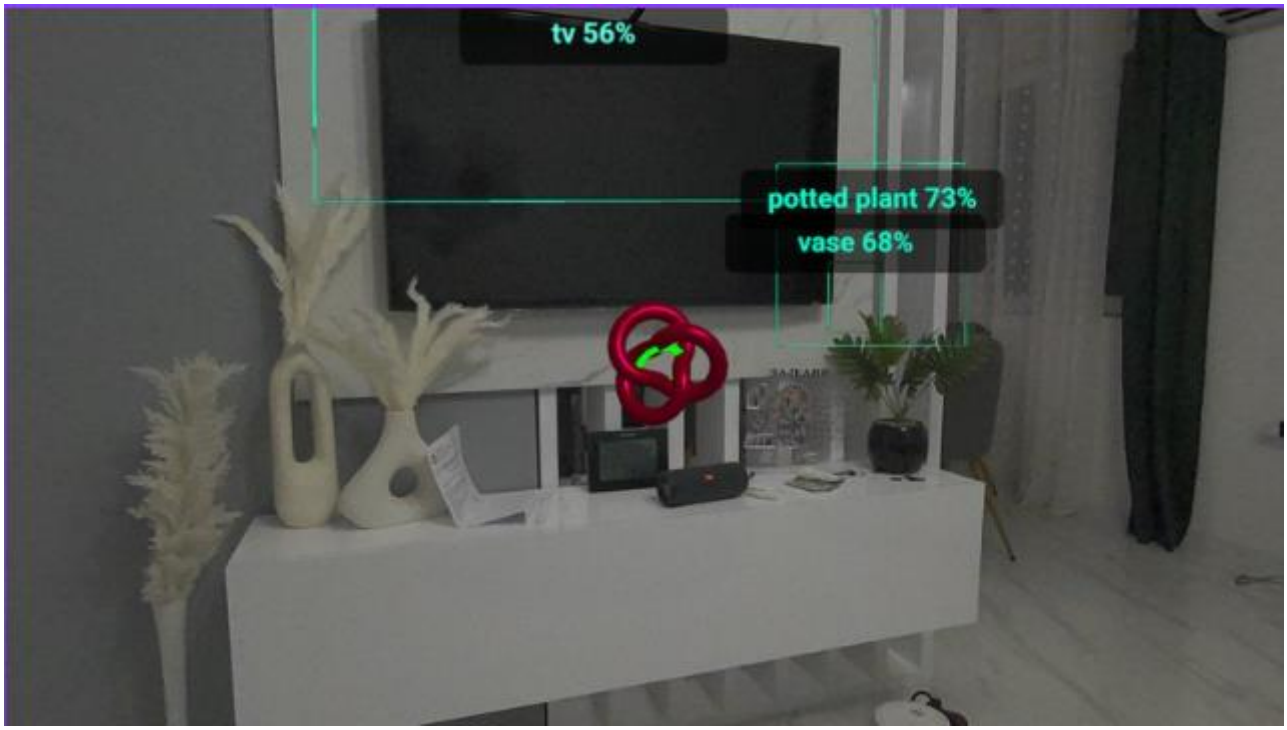
Toto je aplikácia WebXR rozšírenej reality pre Meta Quest 3, ktorá kombinuje:

- **Passthrough AR** — skutočná miestnosť je viditeľná cez kamery na headsete
- **Interakcia hit-test** — virtuálny kurzor, ktorý sa prichytí na reálne povrchy
- **AI detekcia objektov** — TensorFlow.js rozpoznáva objekty v kamerovom zázname a zobrazuje popisy v 3D priestore
- **Porozumenie scéne** — WebXR Plane Detection vizualizuje detekované povrchy (podlaha, steny, stôl)

Živá URL: http://92.113.18.92/

Jeden za druhým:

index.html



8.2 Architektúra projektu

index.html (všetko v jednom)

├── HTML → canvas + UI tlačidlo + skrytý video element

├── CSS → priehľadné pozadie, overlay UI

└── JavaScript

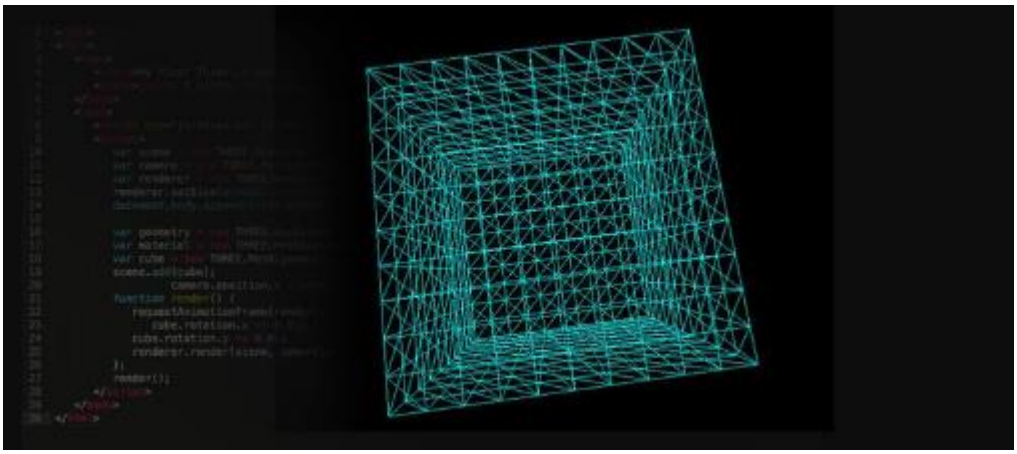
├── Three.js → 3D renderovací engine (scéna, kamera, materiály)

├── WebXR API → AR session, hit-test, plane detection

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

Prečo čistá Three.js (bez A-rámu)?

Počas vývoja sa zistilo, že A-Frame má vlastnú internú slučku vykresľovania, ktorá je v konflikte s explicitnou **immersive-ar** WebXR reláciou. A-Frame začína **immersive-vr** session (nepriehľadná, bez passthrough) namiesto **immersive-ar**. Prechod na **čistý Three.js** nám dal priamu kontrolu nad typom relácie aj nad renderovacím cyklom.



8.3 Technologický stack

Technológia	Verzia	Úloha
Three.js	0.157.0	3D renderovanie, geometria, materiály
WebXR Device API	Natívne pre prehliadač	AR sedenie, test zásahu, detekcia lietadla
TensorFlow.js	4.15.0	Inferenčný engine ML v prehliadači
COCO-SSD	2.2.3	Predtrénovaný model detekcie objektov
getUserMedia API	Natívne pre prehliadač	Kamerový prúd na analýzu TF.js

8.4 Tok aplikácií

Načítanie stránok

→ TF.js model (coco-SSD) sa načítava asynchrónne (~6MB)

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

→ tlačidlo "Enter AR" sa aktivuje

Používateľ kliká na "Enter AR"

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

→ Quest umožňuje Passthrough (kamera viditeľná cez headset)

→ Súčasne: getUserMedia() → kamerový stream pre TF.js

→ renderer.setAnimationLoop() začína (XR render loop)

Každý snímok (~72fps na Quest 3)

- scene.background = null (zabezpečuje transparentnosť)
- Testovanie zásahov → aktualizuje pozíciu kurzora
- Plane Detection → aktualizuje vizualizáciu povrchu
- Každé 2s → runDetection() → AI inferenciu
- renderer.render(scéna, kamera)

8.5 Podrobné vysvetlenie kódu

1. HTML Štruktúra (riadky 1–78)

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

Skrytý <video> prvok prijímajúci tok ***getUserMedia***(). TensorFlow.js ho používa ako vstup na inferenciu — nikdy sa nezobrazuje používateľovi, iba analyzuje.

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

Prekryvná UI vrstva plávajúca nad **Three.js** plátnom. ***Ukazovacie udalosti: Žiadne na*** kontajneri, ale ***ukazovákové udalosti: všetko*** len na tlačidle — takže prekryv neblokuje 3D interakcie.

2. Three.js Inicializácia (riadky 84–109)

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

alpha: true vytvára WebGL plátno s transparentným alfa kanálom — toto je predpoklad pre passthrough. ***xr.enabled = true*** aktivuje integrovanú integráciu Three.js WebXR.

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

Kamera je **pridaná do scény**. To je dôležité, pretože objekty pripojené ku kamere (podriadené entity) musia byť v hierarchii scény.

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

Geometria kruhu je otočená o -90° na osi X, takže leží vodorovne na detekovaných povrchoch. Bez tejto rotácie by stál vertikálne.

3. makeLabel() — Textový sprite (riadky 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;
}
```

TRI. Sprite je objekt, ktorý je vždy otočený ku kamere (**billboarding**) — ideálny pre označenia v 3D priestore. Kreslí sa na HTML plátne, konvertuje na ***CanvasTexture*** a aplikuje sa na ***SpriteMaterial***.

depthTest: false zabezpečuje, že štítok je vždy viditeľný, aj keď je geometricky "za" iným 3D objektom.

4. detectionTo3D() — 2D → 3D projekcia (riadky 140–163)

Toto je matematické jadro integrácie AI-AR.

```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);
```

*Príklad: Ak AI deteguje stoličku v ľavej tretine videa, **nx ≈ -0,17**. To sa premení na záporný uhol (vľavo od osi pohľadu). Štítok je umiestnený 1,8 m pred kamerou týmto smerom.*

5. makeBox3D() a bbox2dSize3D() — 3D ohraničujúce boxy (riadky 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 *vykresľuje iba hrany krabice — efekt tenkého drôteného okraja bez vyplneného povrchu.*

```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
  };
}
```

*Vzorec **totalW** je štandardná perspektívna projekcia: **vo vzdialenosti d** závisí viditeľná šírka od zorného poľa. Z toho odvodíme, koľko metrov zodpovedá pixelom ohraničujúcej krabice.*

6. runDetection() — AI inferenčný pipeline (riadky 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)** prijíma priamo prvok <video> — TF.js interne číta pixelové dáta z aktuálneho video rámca.*

*Pred.bbox formát: **[x, y, šírka, výška]** v pixeloch.*

*Každá detekcia vytvorí pár **JS (sprite, box),** ktorý vydrží 4 sekundy.*

7. WebXR Session — Kľúčové detaily (riadky 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);
```

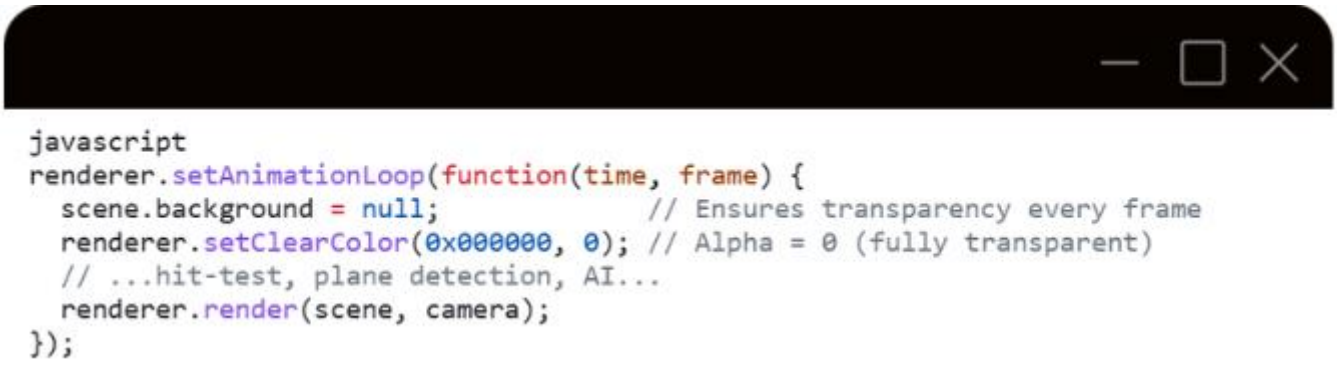
Prečo **immersive-ar** a nie **immersive-vr**?

- **Immersive-VR** = nepriehľadné čierne pozadie (VR prilba)
- **immersive-ar** = priechodná kamera + virtuálny obsah prekrytý

Lokálny referenčný priestor na poschodí fixuje súradnicový systém na podlahu miestnosti — **Y=0** je na úrovni podlahy.

Hit-test a **detekcia roviny** sú voliteľné, pretože nie sú podporované na všetkých zariadeniach a verziách prehliadača — deklarovanie ich ako voliteľných zabraňuje zlyhaniu relácie, ak nie sú dostupné

8. Render Loop (riadky 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);
});
```

Prečo **nastaviť AnimationLoop** a **nepožiadať AnimationFrame**?

Renderovací okruh Three.js XR musí byť integrovaný s WebXR rámcovým callbackom. **renderer.setAnimationLoop()** sa automaticky naviaže na XR reláciu, keď **renderer.xr.enabled = true**. Alternatíva (manuálne volanie **session.requestAnimationFrame(**)) vyžaduje manuálne volanie **renderer.render(),** ale riskuje, že chýbajú správne XR matice, ktoré **Three.js** aplikujú interne.

scene.background = null musí byť volaný **v každom snímku** , pretože Three.js môže túto hodnotu interne resetovať počas určitých operácií.

9. Detekcia lietadiel (riadky 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, ktorý sleduje fyzický povrch. **frame.getPose()** vráti svoju pozíciu vo zvolenom referenčnom priestore.

Mapa **detekovaných libier (Mapa<XRPlane, TRI. Mesh>)** zabraňuje vytváraniu duplicitných sietí pre rovnaký povrch naprieč rámcami.

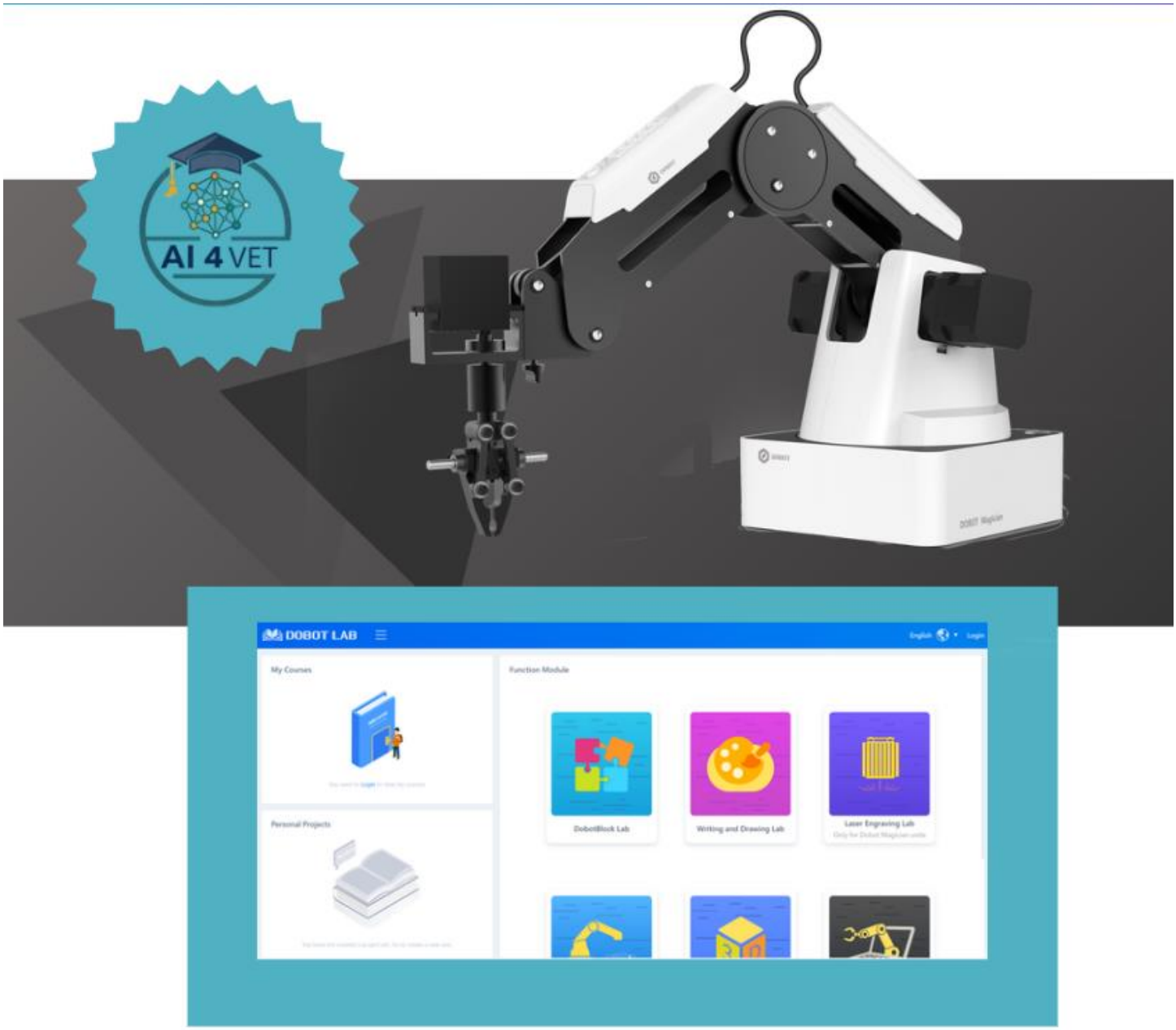
8.6 Známe obmedzenia

Obmedzenia	Dôvod
AI štítky nie sú pixelovo dokonale zarovnané s objektom	Quest passthrough kamery a getUserMedia poskytujú rôzne prúdy s rôznym zorným a optickým kalibrovaním
Detekcia roviny vyžaduje dokončenie nastavenia miestnosti s úlohami	Slúchadlá museli predtým preskenovať miestnosť
getUserMedia môže byť na niektorých zariadeniach zamietnutý	Závisí to od oprávnení prehliadača
AI inferencia nie je v reálnom čase (beží každé 2 sekundy)	Coco-SSD je relatívne ťažký model; ľahšie alternatívy (MobileNet SSD) by poskytli vyššiu priepustnosť

8.7 Zdrojový kód

https://github.com/bciric1/ARVR

http://vr.aiforvet.eu/



III. AI S Dobotom

9. NÁVOD NA INŠTALÁCIU OVLÁDAČA

Pri prvom pripojení Dobotu pripojte Dobot k PC cez USB port. Potom zapnem Dobot, systém automaticky rozpozná príslušný hardvér, vyhľadá správny ovládač a nainštaluje ho. Ak sa však inštalácia nepodarí, môžete ju manuálne nainštalovať znova. Schéma postupu inštalácie je nasledovná:

9.1 Stiahnuť balík ovládača CH340 a nainštalovať ho

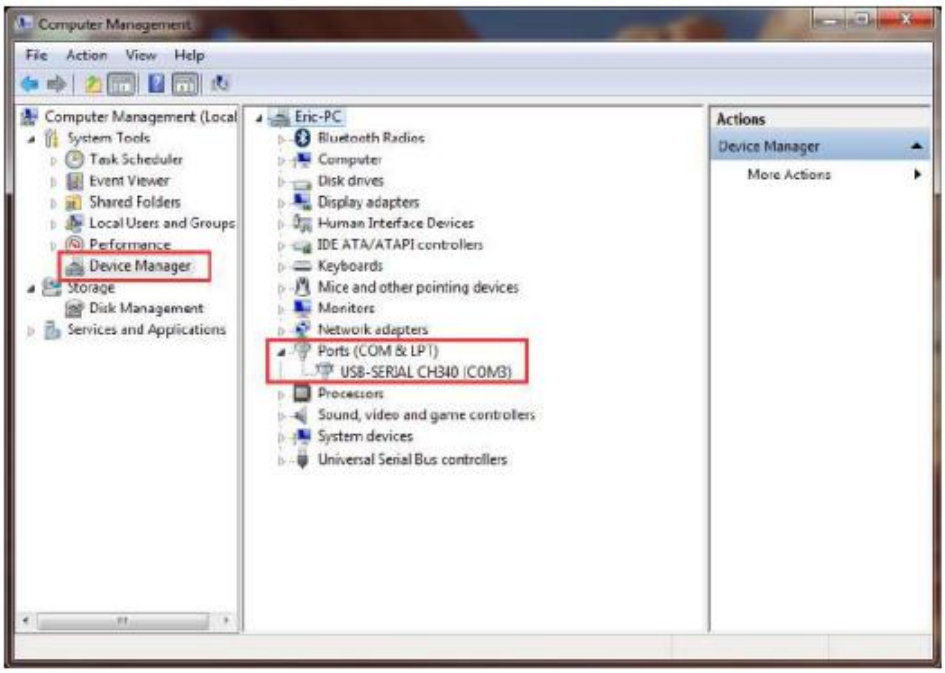
Existujú dve verzie ovládača založené na Windows/Linuxe, preto si prosím vyberte zodpovedajúcu verziu pre váš operačný systém na stiahnutie. Ovládač je možné nájsť na adrese na stiahnutie:

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

Po stiahnutí rozbalte a nainštalujte ovládač

9.2 Skontrolujte, či zariadenie funguje správne v správcovi zariadení

Otvorte správcu zariadení a ak nájdete príslušný COM port "USBSERIAL CH340", ukáže sa, že ovládač bol úspešne nainštalovaný. Správna inštalácia je zobrazená nižšie:



10. PREVÁDZKOVÉ INŠTRUKCIE DOBOTSTUDIO

Softvér používaný Dobot Magician je DobotStudio a najnovšiu verziu si môžete stiahnuť z našej oficiálnej webovej stránky: http://www.dobot.cc/downloadcenter.html?sub_cat=70#sub-download Po úspešnom stiahnutí súboru rozbalte a dvakrát kliknite DobotStudio.exe.

Vyberte zodpovedajúci sériový port Dobot v ľavom hornom rohu DobotStudio a kliknite na "Pripojiť". Po úspešnom spojení sa zobrazí "Odpojenie". Keď je Dobot pripojený, parametre súradníc sa aktualizujú na pravej strane rozhrania

Na hlavnom softvérovom rozhraní je osem modulov:

- Teaching & Playback: Systém na naučenie Dobota, ako sa pohybovať. Umožňuje Dobotovi vykonávať zaznamenané pohyby manuálnym ovládaním.
- Write & Draw: Ovláda Dobot na písanie, kreslenie alebo laserové gravírovanie.
- DobotBlockly： Učí základné programovanie prostredníctvom puzzle rozhrania. Intuitívne a ľahko pochopiteľné.
- Skript: Upraviť skriptovací jazyk na ovládanie Dobota.
- LeapMotion: Ovládaj Dobota gestom.
- Myš: Ovládaj Dobota myšou.
- LaserEngraving: Gravíruje obrázky, tvary a slová cez bitmapu pomocou Dobota.
- Pridajte viac: Pridajte ešte viac funkcií pre Dobot!

10.1 Lineárny režim

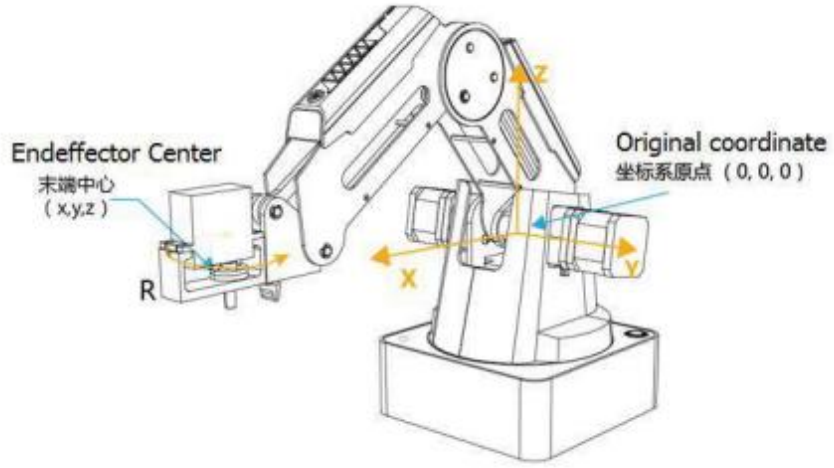
Na základe súradnicového systému osí tela X, Y, Z, pričom počiatok je v strede troch motorov. X, Y, Z je súradnica stredu koncovej plošiny a smer X je kolmý na základňu vpredu, Y je kolmý na základňu doľava a Z je vertikálne nahor. R označuje rotáciu servo-kĺbu vzhľadom na súradnicovú sústavu (proti smeru hodinových ručičiek je kladný smer).

(1) Kliknite na X+,X- a Dobot sa pohybuje pozdĺž X v zápornom alebo kladnom smere;

(2) Kliknite na Y+,Y- a Dobot sa pohybuje pozdĺž Y v negatívnom alebo kladnom smere;

(3) Kliknutím na Z+,Z- sa Dobot pohybuje pozdĺž Z v zápornom alebo kladnom smere;

(4) Kliknite na R+,R- a Dobot sa bude pohybovať pozdĺž R v zápornom alebo kladnom smere.



10.2 Režim posunu

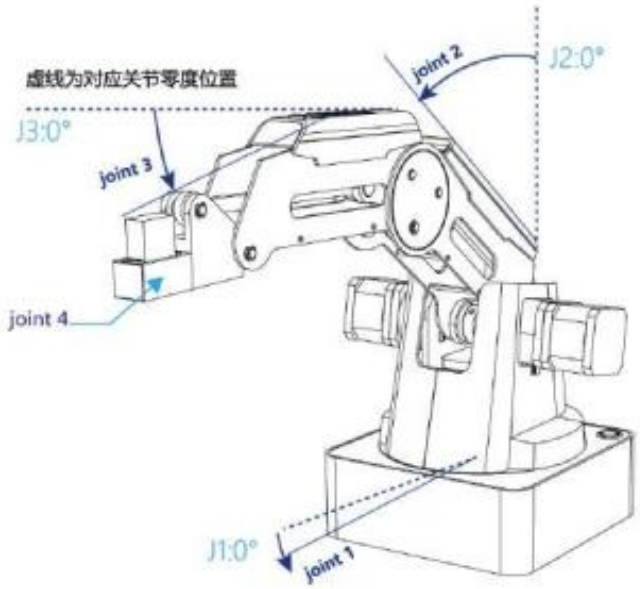
Tento pohyb je zameraný na jednu os. Podržte tlačidlo a zodpovedajúca os sa pohybuje nezávisle. Keď je os maximálna, kĺb sa prestane hýbať. Každá os má kladný smer proti smeru hodinových ručičiek. Kĺb1，2，3，4 označujú základňu, zadné rameno, predlaktie a servo.

(1) Kliknúť na kĺb1+、Kĺb1- a ovládať motor základne Dobotu, aby sa otáčal v zápornom alebo kladnom smere;

(2) Kliknúť na kĺb2+、Kĺb2- a motor zadného ramena ovládať, aby sa otáčal v zápornom alebo kladnom smere;

(3) Kliknúť na Joint3+、Joint3- a ovládať motor predlaktia na rotáciu v zápornom alebo kladnom smere;

(4) Kliknúť na Joint4+、Joint4- a ovládať servor na rotáciu v negatívnom alebo kladnom smere; Medzi nimi je rozsah rotácie Joint4 ±150°

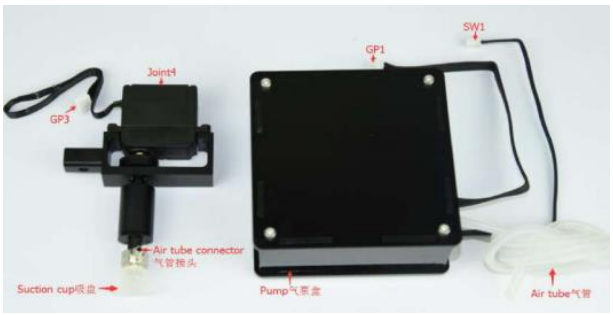


11. VYUČOVANIE A PREHRÁVANIE

Tu sa naučíme, ako sa podlizovať alebo chytiť jednoduché predmety pomocou funkcie Teaching & Playback. Keďže potrebujeme použiť sadu vzduchovej pumpy pre prísavku a sadu gripperu, predstavíme tieto dve sady spolu.

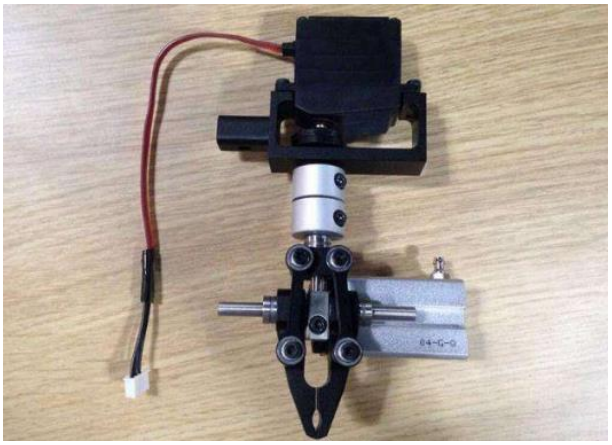
11.1 Súprava vzduchovej pumpy

Predvolenou inštaláciou Dobot Magician je prísavka. Pumpová skrinka a sada prísavky sú zobrazené nižšie:



11.2 Sada pneumatického uchopovača

Pneumatické príslušenstvo je zobrazené na nasledujúcom obrázku:

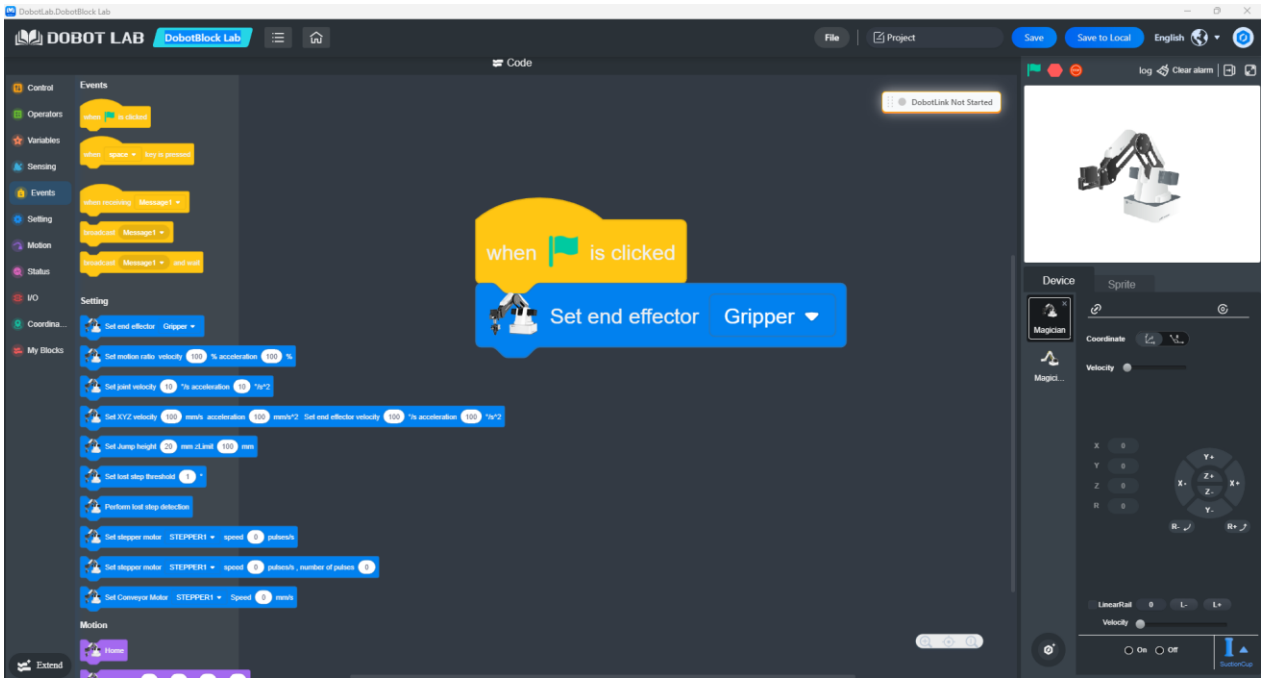


12. PROJEKT: AUTOMATIZOVANÝ SYSTÉM TRIEDENIA A TRIEDENIA ODPADU POMOCOU DOBOT MAGICIAN

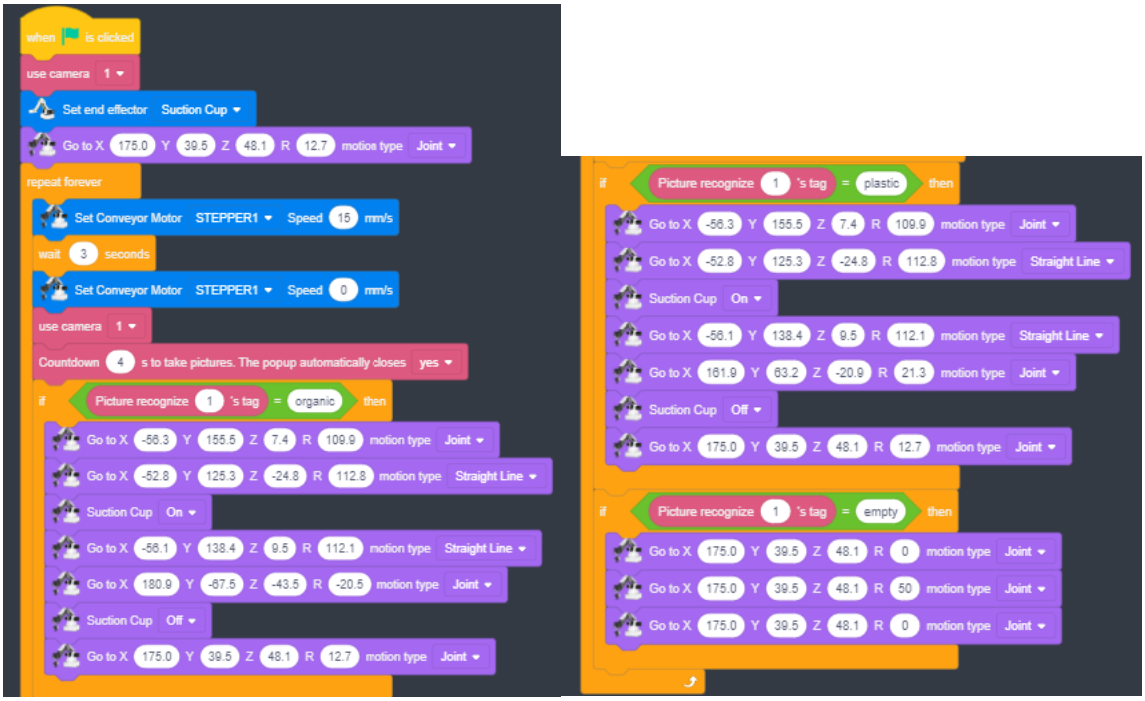
Dobot Blockly je programátorská platforma založená na Google Blockly. V tomto procese môžu používatelia programovať cez formát hádaniek, ktorý je jednoduchý a ľahko pochopiteľný. Používatelia môžu tiež kedykoľvek používať integrované API Dobotu.

12.1 Blokové rozhranie

Otvorte DobotStudio a kliknite na DobotBlock Lab:



12.2 Blokový kód



12.3 Inicializácia a definícia problému

Kontext a motivácia

V modernom priemysle a ekológii je manuálne triedenie odpadu pomalý, neefektívny a často nebezpečný proces. Cieľom tohto projektu je vytvoriť autonómny systém, ktorý využíva **počítačové videnie** a **robotickú manipuláciu** na rozpoznávanie a fyzické oddelenie rôznych materiálov (v tomto prípade organický odpad a plast).

Vyhlásenie problému

Hlavnou výzvou je synchronizácia troch nezávislých podsystémov:

1. **Dopravný systém:** Dopravný pás, ktorý privádza predmety do pracovného priestoru.
2. **Senzorický systém:** Kamera, ktorá musí identifikovať typ predmetu v reálnom čase.
3. **Aktuačný systém:** Robotické rameno, ktoré musí presne vykonávať vyberanie a umiestňovanie na základe spätnej väzby zo senzora.

12.4 Architektúra riešení

Riešenie je založené na platforme **Dobot Magician** a blokovom programovacom prostredí (Blockly), ktoré integruje AI moduly na rozpoznávanie obrázkov.

Hardvérové komponenty

- **Dobot Magician Arm:** Vysoko presný 4-osový robot.
- **Prísavka:** Používa sa ako koncový efektor pre svoju všestrannosť pri manipulácii s rôznymi tvarmi.
- **USB kamera:** Namontovaná nad pásom pre stabilný pohľad zhora.
- **Dopravníkový pás:** Ovládaný krokovým motorom pripojeným k Dobotu.

Softvérová logika

Algoritmus nasleduje **iteratívny uzavretý model**: Spustiť pás -> Zastaviť -> Zachytiť -> Analyzovať -> Zoradiť -> Vrátiť sa domov.

12.5 Podrobná analýza kódových blokov

Program je rozdelený do štyroch kľúčových častí:

Konfigurácia (Setup)

- **použiť kameru 1**: Deklaruje hardvérový vstup pre vizuálne dáta.
- **Nastav koncový efektor [Prísavka]**: Soft-mapuje ovládanie vzduchového čerpadla na výstup robota.

- **Prejdite na X:175, Y:39.5, Z:48.1**: Toto je "Bezpečnostná pozícia." Robot ustupuje, aby neblokoval kameru a zabezpečil optimálnu cestu na akékoľvek miesto na páse.

Logistika (riadenie dopravníkov)

Vo vnútri bloku opakovania navždy:

- **Nastavte dopravníkový motor [STEPPER1] Rýchlosť 15 mm/s**: Aktivuje pás kontrolovanou rýchlosťou, aby zabránil kĺzaniu ľahkých predmetov.
- **počkať 3 sekundy**: Kritická časová konštanta definujúca vzdialenosť medzi položkami.
- **Rýchlosť 0**: Zastaví pás, aby robot mohol vykonať presný "statický pick".

AI inšpekcia (rozpoznávanie obrázkov)

- **Odpočítavanie 4 s**: Poskytuje stabilizačný čas pre zaostrenie kamery a pre AI vyskakovacie okno na spracovanie obrazu.
- **Ak obrázok rozpozná tag 1 = [organický/plastic]**: Volá Deep Learning API na porovnanie aktuálneho rámca s trénovaným datasetom a vráti class tag.

Kinematika a manipulácia

Pre každú vetvu (organickú/plastovú) systém používa dva typy pohybu:

1. **Pohyb kĺbu (PTP):** Pohybuje všetkými kĺbmi súčasne pre najrýchlejší prechod z bodu do bodu.
2. **Priamka (lineárna):** Pohybuje ramenom striktne vertikálne po osi Z, aby sa zabezpečil dokonalý kontakt prísavky bez prevrátenia predmetu.

12.6 Operačný algoritmus (krok za krokom)

- **ŠTART:** Inicializujte nástroje a nastavte koncový efektor.
- **TRANSPORT:** Pohybujte pásom na 3 sekundy, potom zastavte.
- **SENSE:** Zachytiť obrázok a identifikovať značku pomocou AI.
- **ROZHODNUTIE:**
- **Ak organické:** Presuň sa, aby si získal súradnice, zapni sanie, presuň sa do organického zásobníka, vypni odsávanie.
- **Ak je plast:** Presuň sa, aby si získal súradnice, zapni sanie, presuň sa do plastového boxu, vypni odsávanie.
- **Ak je prázdny:** Vykonajte signálny pohyb "trasenia" (otáčanie osi R).
- **RESET:** Vráťte sa na počiatočné bezpečnostné súradnice.
- **LOOP:** Opakujte cyklus donekonečna.

12.7 Technické špecifikácie súradníc

Na základe skriptu boli kalibrované nasledujúce súradnice:

Bod	X (mm)	Y (mm)	Z (mm)	R (°)	Popis
Domov	175.0	39.5	48.1	12.7	Poloha na voľnobeh/čakanie
Prístup	-56.3	155.5	7.4	109.9	Pozícia nad položkou
Chyť	-52.8	125.3	-24.8	112.8	Kontaktný bod (znížený)
Organický zásobník	180.9	-67.5	-43.5	-20.5	Likvidácia na organické materiály
Plastový kontajner	161.9	63.2	-20.9	21.3	Likvidácia plastov

12.8 Riešenie problémov s implementáciou

- **Nepresné chytanie:** Uistite sa, že je predmet v strede. Ak sa to líši, použite vodiace prvky na páse alebo integrujte dynamickú spätnú väzbu X/Y z AI kamery.
- **Chyby v klasifikácii:** Rozpoznávanie AI je citlivé na svetlo. Použite vysoko kontrastnú farbu pásu a konzistentné vonkajšie LED osvetlenie.
- **Bezpečnosť:** Vždy sa uistite, že pracovný priestor je čistý od prekážok pred začatím nekonečného cyklu.

IV. AI S rPI 5

13. ÚVOD

Raspberry Pi je malý počítač veľkosti balíčka kariet a schopný spustiť plnohodnotný desktopový operačný systém Linux pri spotrebe len skromnej spotreby. Obsahuje USB porty na pripojenie klávesnice a myši spolu s rôznymi ďalšími perifériami, ethernetový adaptér a HDMI monitory. Raspberry Pi bol pôvodne navrhnutý na vzdelávanie, no našiel plodné využitie pre hobby nadšencov, domácu automatizáciu, priemyselné aplikácie a ako vhodná technolová technológia pre použitie v školách vo väčšine sveta. Je vyrobený tak, aby spĺňal smernice RoHS (Restriction of Hazardous Substances) a spolieha sa na jednu microSD kartu na svoje uloženie. Beží na variante Linuxu nazývanej Raspberry Pi OS a podporuje širokú škálu open source softvéru vrátane množstva vzdelávacích programov. Navyše, dá sa kúpiť za rozumnú cenu. Tento sprievodca bol napísaný predovšetkým s ohľadom na študentov inžinierstva a informatiky, no bude zaujímavý aj pre tých, ktorí majú záujem dozvedieť sa viac o programovaní a technickej informatike.

13.1 Počiatočné nastavenie Raspberry Pi

Vlož SD kartu s operačným systémom Raspberry Pi do Raspberry Pi a zapoj napájanie. Keď prvýkrát spustíte bežný Raspberry Pi OS, beží grafické desktopové prostredie s priateľským rozhraniem riadeným menu. Pri prvom spustení sa zobrazí dialógové okno, ktoré vás prevedie počiatočným nastavením. Postupujte podľa pokynov na nastavenie klávesnice a používateľského mena. Uveďte používateľské meno a heslo podľa požadovaných požiadaviek. Nakonfigurujte nastavenia WiFi a vyberte možnosť "Aktualizovať softvér" (poznámka: pri prvom nastavovaní Raspberry Pi to môže trvať veľmi dlho).

13.2 Začať s príkazovým riadkom

Existuje tiež-verzia operačného systému založená na príkazovom riadku s názvom Raspberry Pi OS Lite. Táto verzia OS spotrebuje menej energie než bežný Raspberry Pi OS s desktopovým prostredím a dá sa použiť na starších modeloch Raspberry Pi s menším množstvom RAM. Pri nastavovaní Raspberry Pi s Lite OS budete vyzvaní na konfiguráciu klávesnice a zadanie používateľského mena a hesla. Lite verzia OS je vhodná na použitie Raspberry Pi ako servera, zabudovaného systému alebo v aplikácii IoT (Internet vecí). Avšak pre bežné používanie na stole je najlepšia bežná verzia Raspberry Pi OS. Pri používaní Desktop OS je príkazový riadok stále prístupný cez aplikáciu Terminal. Alternatívne je možné k nemu pristupovať aj cez virtuálnu konzolu stlačením CTRL+ALT+F1, čím sa otvorí terminál na celú obrazovku. Ak sa zobrazí prihlasovací prompt, môžete sa prihlásiť pomocou používateľského mena a hesla, ktoré ste nastavili pri nastavovaní. Z virtuálnej konzoly sa môžete vrátiť na plochu stlačením CTRL+ALT+F7. Ďalšie virtuálne konzoly je možné nezávisle pristupovať pomocou CTRL+ALT spolu s klávesmi F2 až F6.

13.3 The Shell

Keď vstúpite do príkazového riadku, budete bežať v Linuxovom shelle. Jednoducho povedané, shell je interpret príkazov, ktorý poskytuje bohatú sadu príkazov, ktoré možno použiť na vykonávanie programov a rozhranie s operačným systémom. Raspberry Pi OS používa štandardne Bash (Bourne Again Shell), shell založený na staršom shelle nazývanom Bourne Shell. BASH je obľúbený medzi používateľmi Linuxu a umožňuje automatické dopĺňanie príkazového riadku pomocou klávesu Tab a môže sa použiť na tvorbu programov nazývaných shell skripty. Existuje množstvo rôznych linuxových shellov, ktoré sa dajú použiť. Ako je uvedené, predvoleným Linuxovým shellom je Bash shell, ale dostupné sú aj iné shelly. Každý náboj má svoje vlastné vlastnosti a možnosti. Napríklad na prepnutie predvoleného obalu z Bash na Z shell (zsh) zadajte nasledujúce

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

Po vydaní týchto príkazov sa odhlásite a potom sa opäť prihlásite a mali by ste teraz bežať s zsh. Rôzne konfiguračné možnosti je možné nastaviť v súbore s názvom .zshrc, ktorý sa nachádza vo vašom domovskom priečinku.

13.4 Shell príkazy

Niektoré z príkazov dostupných v shelle sú zhrnuté nižšie:

Velenie	
CD adresár	Mení aktuálny pracovný adresár na adresár

PWD	Zobrazuje názov aktuálneho pracovného adresára
mkdir adresár	Vytvorte nový adresár s názvom adresár
RMDIR adresár	Odstráni adresár nazývaný adresár
ls	Zobraziť zoznam súborov v aktuálnom adresári
CP F1 F2	Skopírujte súbor zo zdroja f1 do cieľového f2
názov súboru RM	Odstráňte názov súboru
MV F1 F2	Presuňte súbor z f1 na f2
FTP hostiteľ	Prenos súborov na a z hostiteľa

Tieto príkazy predstavujú len časť používateľských príkazov dostupných v Linuxovom shelle. Online manuál nazývaný man (manuál) stránky poskytuje pomoc s mnohými príkazmi a programami, ktoré je možné volať z shellu. Syntax pre vyvolanie ľudskej užitočnosti je nasledovná:

```
mužské meno  príkazu
```

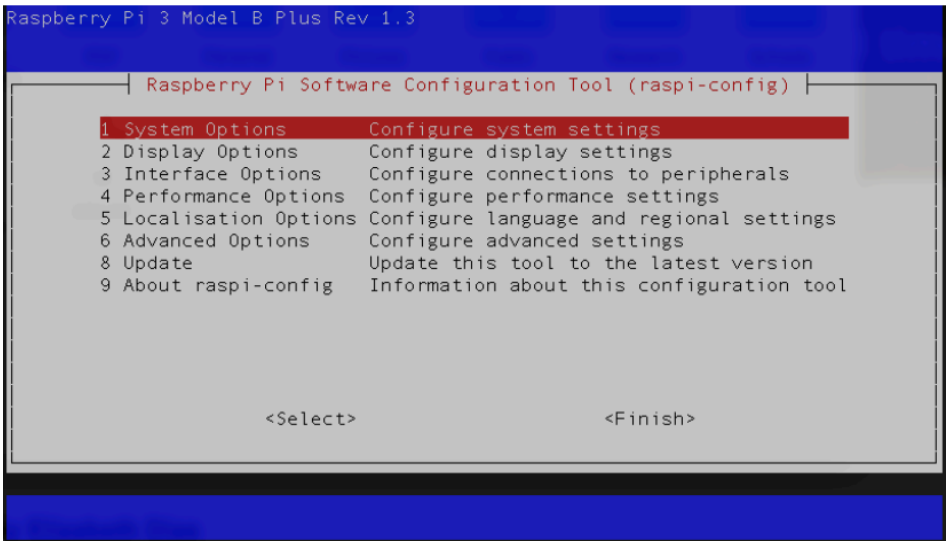
Informácie o špecifikovanom názve príkazu sa potom zobrazia na obrazovke.

13.5 Konfigurácia operačného systému Raspberry Pi

Raspberry Pi OS obsahuje nástroj nazvaný raspi-config, ktorý umožňuje konfiguráciu širokej škály nastavení a služieb. Na prevádzku tohto nástroja napíšte:

```
sudo raspi-config
```

Týmto sa spustí konfiguračný nástroj Raspberry Pi priamo v termináli s hlavným menu, ako je uvedené nižšie:



Konfiguračný nástroj Raspberry Pi je možné použiť na povolenie rozhrania kamery, ako aj sériovej, I2C a SPI komunikácie. Obsahuje tiež možnosť bootovať priamo do príkazového riadku namiesto plochy.

13.6 Pripojenie na diaľku k Raspberry Pi

Je možné spustiť Raspberry Pi bez hlavy, bez obrazovky, klávesnice alebo myši. To je často prípad pri používaní Raspberry Pi v zabudovaných aplikáciách alebo v konfigurácii IoT (Internet vecí). Nasledujúce podsekcie popisujú, ako sa na diaľku pripojiť k Raspberry Pi pomocou jednej z nasledujúcich možností:

- pomocou USBtoTTL sériového kábla
- používaním SSH cez Ethernet alebo WiFi pripojenie
- Služba Raspberry Pi Connect

Niektoré modely Raspberry Pi je možné pripojiť pomocou USB kábla. Funguje to tak, že sa zapne režim USB Gadget, ktorý umožňuje USB portu prezentovať sa ako rôzne typy zariadení. Je známe, že to funguje s Raspberry Pi Zero, nie s väčšinou ostatných modelov. Dva nižšie popísané prístupy by mali fungovať so všetkými modelmi Raspberry Pi.

14. ÚVOD DO PROGRAMOVACÍCH JAZYKOV

Raspberry Pi repozitáre obsahujú podporu COBOL, ako aj bohatú sadu modernejších programovacích jazykov ako Python, C, C++ a Java. Podporuje viac špecializované a staršie profesionálne gramovacie jazyky. Vo všeobecnosti sa paradigmy programovacích jazykov delia do troch všeobecných kategórií:

• procedurálne programovacie jazyky

• objektovo orientované programovacie jazyky

• funkcionálne programovacie jazyky

14.1 Programovací jazyk Python

Python vynašiel začiatkom 90. rokov Guido van Rossum. Python môže byť napísaný buď v procedurálnom alebo objektovo orientovanom paradigme. Je to open source projekt, ktorý je široko dostupný, používa jednoduchú syntax, obsahuje bohaté knižnice a ponúka rôzne programovacie nástroje, zdrojový kód v Pythone sa spúšťa na virtuálnom stroji, ktorý prekladá kód do konkrétneho strojového kódu vykonávaného procesorom. Keďže Python je interpretovaný virtuálnym strojom, je nezávislý od platformy.

Raspberry Pi by malo mať Python nainštalovaný štandardne a môže spúšťať Python programy priamo z príkazového riadku. Na vstup do programu najprv potrebujete obyčajný textový editor. Ak používate desktopové prostredie, existuje priateľské, grafické, integrované vývojové prostredie (IDE) pre Python vhodné pre začiatočníkov s názvom Thonny. Na inštaláciu Thonny napíšte:

```
sudo apt install thonny
```

Pre pokročilejších používateľov v grafickom prostredí poskytuje program vscode vynikajúci editor na programovanie v rôznych jazykoch, vrátane Pythonu. Ako bolo opísané vyššie, vscode je možné spustiť aj na diaľkovú úpravu súborov. Ak používate príkazový riadok, môžete použiť ktorýkoľvek z editorov príkazového riadku popísaných v predchádzajúcich sekciách, vrátane vi, emacs alebo nano. Napríklad na úpravu zdrojového súboru v Pythone s názvom hello.py zadajte nasledovné:

```
Nano , ahoj.py
```

Ďalej zadajte do zdrojového súboru nasledujúci kód:

```
meno = vstup('Ako sa voláš? ')
print ('Ahoj', meno,' vitajte v Raspberry Pi!')
tlač ('Good Bye')
```

Potom uložte a ukončite nano a spustite súbor zadaním:

```
python3 ahoj.py
```

Program by mal bežať podľa očakávania.

14.2 Kompilácia a spustenie programu v C/C++

Linux má rôzne nástroje na podporu vývoja softvéru v C aj C++. V skutočnosti je samotný operačný systém Linux napísaný v jazyku C. Programovací jazyk C je procedurálny jazyk a C++ stavia na C, aby poskytol podporu objektovo orientovaného programovania. Kompilátory, ktoré budeme používať pod Linuxom, sú GNU C Compiler (gcc) a GNU C++ kompilátor (g++). Aby ste zabezpečili inštaláciu nástrojov kompilátora GNU C/C++, napíšte:

```
Sudo APT install GCC G++ GDB Build-Essential
```

Napríklad na zadanie jednoduchého programu v C s názvom hello.c pomocou nano editora zadajte nasledovné:

```
Nano , ahoj.c
```

Pomocou editora zadajte do zdrojového súboru nasledujúci kód:

```
/* A Raspberry Pi C program */
#include <stdio.h>
int main(void)
{
    printf("Ahoj svet.\n");
    printf(""Skompilované a spustené na Raspberry Pi.\n");
    return 0;
}
```

Uložte a ukončite editor. Na kompiláciu zdrojového kódu zadajte nasledujúce na výzve v okne terminálu. Napríklad na kompiláciu programu hello.c vyššie môžete zadať nasledovné:

```
gcc -Stena -o ahoj ahoj.c
```

Kompilátor gcc má množstvo ďalších príkazových možností, ktoré môžete použiť. Pre viac informácií navštívte manové stránky. Upozorňujeme, že kompilátor C++ je možné vyvolať použitím g++ namiesto gcc. Na spustenie kompilovaného programu v aktuálnom pracovnom adresári nezabudnite pred názvom programu zadať ./, aby ste určili cestu ako aktuálny adresár. Napríklad, aby ste spustili program hello.c po jeho kompilácii ako ini, napíšte:

```
.\ahoj
```

Ak kompilátoru nebol poskytnutý žiadny výstupný názov súboru, predvolený názov výstupného súboru bude a.out.

14.3 Kompilácia a spustenie Java programu

Je tiež možné vyvíjať a spúšťať Java programy pomocou Linuxu na Raspberry Pi. Existujú dva rôzne balíky, ktoré je možné nainštalovať: on poskytuje Java Runtime Environment (JRE) a druhý poskytuje Java Development Kit (JDK). JRE vám umožňuje spúšťať Java programy, ale JDK umožňuje kompilovať a spúšťať Java programy. Na inštaláciu vývojového balíka OpenJDK Java zadajte nasledovné:

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

Keď je to nainštalované, môžete skompilovať Java program. Napríklad zadajte nasledujúci jednoduchý Java program s názvom hello.java pomocou obyčajného textového editora:

```
/* Java program */
trieda Main {
    public static void main(String args[]) {
        Systém.von.println("Ahoj svet.\n");
    }
}
```

Skompilujte program tak, že na prompt zadáte nasledujúce:

```
Javac,  ahoj.Java
```

kde myprogram.java je názov Java zdrojového súboru. Na spustenie Java programu zadajte na prompt nasledujúce: **java Main** , kde Main je názov triedy, kde program začína.

15. SKÚMANIE UMELEJ INTELIGENCIE

15.1 Podpora hardvéru a softvéru pre AI

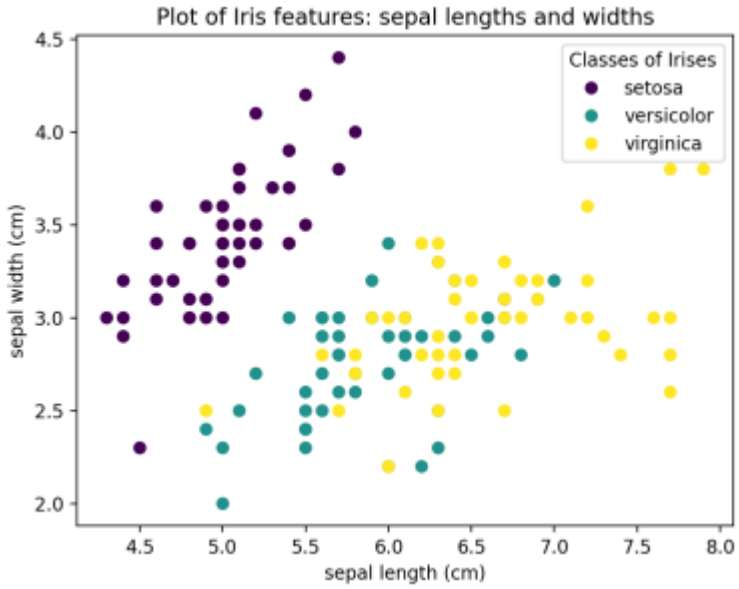
Novšie modely procesora Raspberry Pi obsahujú viacero ARM jadier. Napriek slušným hardvérovým schopnostiam môže Raspberry Pi zápasiť s výpočtovými nárokmi náročnejších aplikácií umelej inteligencie (AI). Pre náročnejšie aplikácie existuje doplnková doska Raspberry Pi — nazývaná HAT (Hardware Attached on Top), ktorá poskytuje NPU (Neural Processing Unit) na zrýchlenie výpočtov strojového učenia. Raspberry Pi AI HAT+ poskytuje vysoko výkonný, energeticky efektívny AI procesor pre Raspberry Pi 5. Okrem hardvérovej podpory existuje široká škála výkonných knižníc strojového učenia, ktoré je možné používať s Raspberry Pi, vrátane scikitlearn a LiteRT. Python obsahuje množstvo výkonných podporných knižníc, ako sú Matplotlib a plotly na kreslenie, NumPy na rýchle operácie s poliami a Pandas na analýzu a manipuláciu s dátami. Nasledujúce časti uvádzajú príklady niektorých z týchto knižníc v praxi.

15.2 SciKit Learn

SciKit Learn je open source knižnica strojového učenia, ktorá pracuje s Pythonom. SciKit Learn poskytuje mnoho rôznych funkcií, vrátane regresných a klastrovacích algoritmov, analýzy hlavných komponentov (PCA), lineárnej diskriminačnej analýzy (LDA) a podporných vektorových strojov (SVM). Všetky formy strojového učenia vyžadujú dátovú sadu, ktorá sa používa na tréning. SciKit obsahuje výber hračkárskych dátových súborov, ktoré možno použiť na experimentovanie s knižnicami strojového učenia. Nasledujúce časti demonštrujú PCA a SVM pomocou klasického datasetu kvetov kosatcov, ktorý je súčasťou zbierky dátových súborov hračiek. Tento súbor dúhoviek pôvodne zostavil biológ Ronald Fisher v známom článku z roku 1936. Táto databáza obsahuje vzorky troch druhov kvetu kosatcov (Iris setosa, Iris virginica a Iris versicolor) a merania dĺžky a šírky kališných lístkov1 aj lupienkov. Knižnica SciKit Learn obsahuje tento súbor dúhoviek na testovanie a demonštračné účely. Nasledujúci Python kód používa Matplotlib na vykreslenie šírky kališného lístka v porovnaní s dĺžkou kališného lístka pre každú z troch tried iris v datasete.

```
z sklearn import datasetov
importovať matplotlib.Pyplot ako PLT
# Načítaj klasickú sadu dát o dúhovke
Iris = dátové súbory.load_iris()
# Vykresliť dĺžky kališných lístkov vs. šírky pre dúhovky v datasete
Obr., Ax = Plt.vedľajšie dejové línie()
scatter = os.scatter(Iris.Data[:, 0], Iris.dáta[:, 1], c=iris.cieľ)
Ax.set_title("Rysy rysov kosatcov: dĺžky a šírky kališných lístkov")
Ax.set(xlabel=iris.feature_names[0], ylabel=iris.feature_names[1])
obr. = sekera.legenda (scatter.legend_elements()[0], iris.target_names,
loc="najlepšie
", názov = "Triedy kosatcov")
Plt.show()
```

Spustenie tohto kódu vedie k nasledujúcemu grafu dátovej sady iris:



15.3 Klasifikácia SVM obrázkov

Nasledujúci príklad je inšpirovaný príkladom SciKit o rozpoznávaní ručne písaných číslic a využíva dataset ručne písaných číslic z repozitára strojového učenia UC Irvine. Táto datasada obsahuje 1797 vzoriek obrázkov zložených zpoľa 8 x 8 pixelov s 10 triedami, kde každá trieda odkazuje na jednu číslicu (09). Krátky program v Pythone na načítanie ručne písaných číslic a ich zobrazenie je zobrazený nižšie:

```
importovať matplotlib.Pyplot ako PLT
zo SKLEARN importujte datasety, metriky, svm
od sklearn.model_selection import train_test_split

# načítať ručne písané číslice dátovej sady
Číslice = dátové súbory.load_digits()

# zobrazuje vzorku desiatich ručne písaných číslic v datasete
Obr., Axes = Plt.vedľajšie dejové línie (nrows=2, ncols=5)
Pre sekeru, číslica v zip (axes.flatten(), digits.obrázky):
    Ax.set_axis_off()
 Ax.imshow(digit,cmap='šedá')
Fig.tight_layout()
Plt.show()
```

Graf zobrazujúci desať ručne písaných číslic v datasete je zobrazený nižšie.

Súbor ručne písaných číslic môžeme rozdeliť na dve sady: tréningovú sadu a sadu na testovanie. SciKit má funkciu na rozdelenie väčšej množiny číslic na dve polovice na tréningové a testovacie množiny nasledovne:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0,5)
```

kde X je pole znakov a y je vektor štítkov, ktoré klasifikujú dáta. Preto môžeme pokračovať v trénovaní SVM s polovicou datasetu na trénovanie a zvyšnú polovicu datasetu použiť na testovanie. Presnosť SVM je možné určiť porovnaním číslic predpovedaných SVM s reálnymi štítkami pre ručne písané číslice a možno hlásiť mieru rozpoznania. Miera rozpoznávania je celkový počet správne identifikovaných obrázkov číslic delený celkovým počtom testovacích snímok. Nasledujúci kód definuje SVM klasifikátor na základe polovice datasetu a následne určuje rýchlosť rozpoznávania pomocou druhej polovice datasetu ako testovacích obrázkov.

```
importovať matplotlib.Pyplot ako PLT
zo SKLEARN importujte datasety, metriky, svm
od sklearn.model_selection import train_test_split

# Čítaj číslice a pretvaruj ich ako vektor obrázkov
Číslice = dátové súbory.load_digits()
n_samples = len(číslice.obrázky)
dáta = číslice.obrázky.reshape((n_samples, -1))

# Vytvoriť podporný vektorový strojový klasifikátor
svm = svm.SVC()
# Rozdeľte dátovú sadu: polovicu na trénovanie a polovicu na testovanie
X_train, X_test, y_train, y_test = train_test_split(
Dáta, číslice.cieľ, test_size=0,5, zamiešanie=Nepravda)

# Trénuj na ručne písaných čísliciach v tréningovej sade
SVM.fit(X_train, y_train)
# Predpovedajte hodnotu číslice pomocou testovacej množiny
Predpovedané = SVM.predpovedať(X_test)

# vypočítaj mieru rozpoznávania
Zápasy = 0
pre x v rozsahu(len(y_test)):
ak y_test[x] == predpovedané[x]:
Zápasy += 1
recognition_rate = (zhody/len(y_test)) * 100;
print(f'Recognition rate: {recognition_rate:.2f}%')
```

Výstup tohto kódu ukazuje nasledovné:

Miera rozpoznateľnosti: 96,11 %

To naznačuje slušnú mieru rozpoznania pri použití SVM na klasifikáciu ručne písaných číslic. Pre viac detailov môžeme vykresliť maticu zmätku na vizualizáciu presnosti algoritmu strojového učenia. Matica zmätku je organizovaná do riadkov a stĺpcov: každý riadok predstavuje jednotlivé triedy v datasete a každý stĺpec predstavuje vykonané predpovede. Ideálne by sa skutočné triedy a predpovede dokonale zosúladili tak, že diagonála matice zmätku bude vyplnená stovkami a všade inde nulami. Diagonálny martix zodpovedá skutočným mieram rozpoznávania, zatiaľ čo všetky ostatné bunky predstavujú výskyt falošných klasifikácií.

16. PRÍKLADY

16.1 Vytvorte si vlastný bezpečnostný systém "detektor rodičov"

Prehľad projektu: Premýšľali ste niekedy, kto sa vkráda do vašej izby, keď nie ste nablízku? V tomto projekte si postavíte inteligentný bezpečnostný systém – často nazývaný "Detektor rodičov", aby ste zistili, kto presne bol vo vašej izbe.

Dosiahnete to kombináciou Raspberry Pi, pohybového senzora a kamery, ktorá automaticky spúšťa a nahráva videozáznam akýchkoľvek votrelcov.

Krok 1: Zhromaždite svoj hardvér

Na zostavenie tejto automatizovanej bezpečnostnej kamery budete potrebovať tri hlavné hardvérové komponenty:

- Raspberry Pi: Toto funguje ako mozog vášho projektu, spracováva signály a spúšťa kód.
- Pasívny infračervený (PIR) pohybový senzor: Táto súčasť funguje ako "oči" vášho systému, deteguje zmeny infračerveného tepla, aby rozpoznala, keď sa osoba pohybuje v blízkosti.
- Modul kamery Raspberry Pi: Toto je zariadenie, ktoré zachytí video dôkazy o votrelcovi.

Krok 2: Pochopte a naladite PIR senzor

Pred zapojením čohokoľvek káblom je potrebné nastaviť fyzické nastavenia priamo na PIR senzore. Ak sa pozriete na zadnú stranu PIR modulu, uvidíte dva oranžové komponenty s malými krížovými zásuvkami, ktoré dokonale pasujú na skrutkovač Philips.

Tieto oranžové ciferníky sa nazývajú potenciometre a umožňujú vám manuálne upravovať správanie senzora:

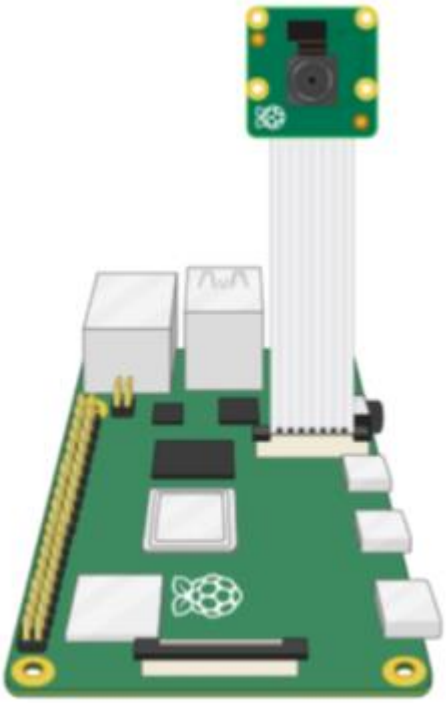
- Citlivosť: Určuje, koľko pohybu je potrebné na spustenie senzora.
- Čas (oneskorenie): Určuje, ako dlho zostáva senzor "aktivovaný" po detekcii pohybu pred jeho resetovaním.

Počiatočné nastavenie: Aby tento projekt fungoval najlepšie, použite skrutkovač na nastavenie potenciometra citlivosti na absolútne maximum a časový potenciometer na absolútne minimum. Tieto nastavenia môžete neskôr vždy upraviť a zmeniť, ak zistíte, že senzor je príliš citlivý alebo zostáva zapnutý príliš dlho.

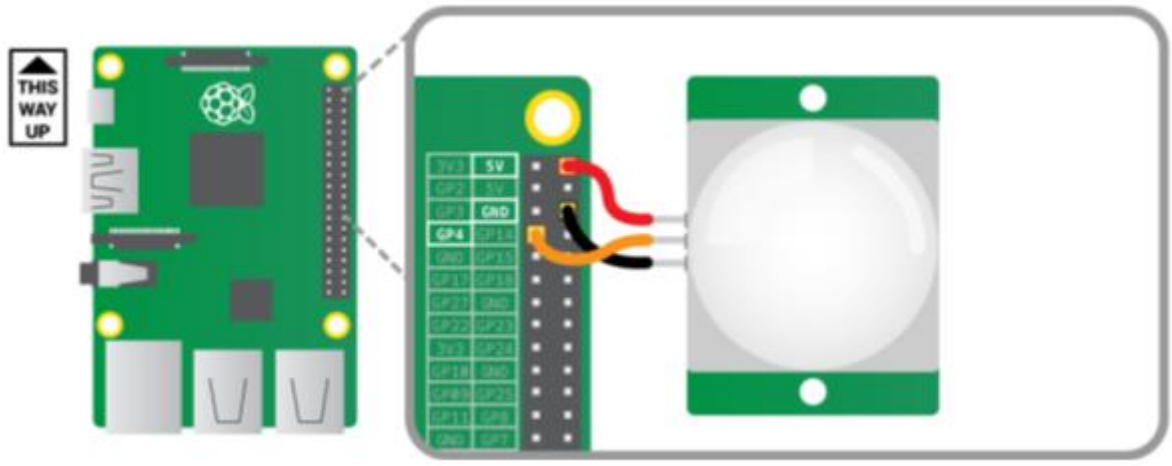
Krok 3: Bezpečne zapojte komponenty

Teraz je čas pripojiť váš hardvér k Raspberry Pi. Kľúčové pravidlo: Vždy sa uistite, že máte Raspberry Pi úplne vypnuté a odpojené od siete pred pripojením hardvéru.

1. Pripojte kameru: Opatrne vložte plochý kábel kamerového modulu do samostatného portu fotoaparátu na Raspberry Pi.



2. Pripojiť PIR senzor: PIR senzor musí poslať svoje pohybové dáta do Raspberry Pi. Dátový výstupný pin PIR senzora pripojíte priamo k pinu označenému GPIO 4 na vašej doske Raspberry Pi. (Budete tiež musieť pripojiť napájacie (VCC) a zemné (GND) piny na príslušné 5V a GND piny na Pi).



Krok 4: Napíšte bezpečnostný softvér (Python)

Keď máte hardvér zapojený, zapnite Raspberry Pi. Otvorte programovacie prostredie s názvom Thonny, vytvorte nový súbor a okamžite ho uložte ako parent_detector.py.

Aby náš hardvér fungoval, musíme importovať konkrétne knižnice. Použijeme MotionSensor z knižnice gpiozero na spracovanie PIR senzora a triedu Camera z knižnice picamzero na ovládanie Camera Module.

Logika kódexu:

1. Kontinuálne monitorovanie: Používame while True: slučka, čo je nekonečná slučka, ktorá udržiava program neustále kontrolovať pohyb.

2. Čakanie na akciu: Program používa pir.wait_for_motion() na pozastavenie kódu, kým senzor nezaznamená pohyb. Keď je pohyb zistený, vytlačí správu na obrazovku.

3. Nahrávanie: Kamera začína zachytávať video pomocou funkcie cam.start_recording().

4. Zastavenie: Nechceme nahrávať prázdnu miestnosť navždy. Kód používa pir.wait_for_no_motion() na čakanie, kým narušiteľ odíde, a potom bezpečne zastaví video súbor pomocou cam.stop_recording().

Riešenie kritickej chyby (problém prepísania): Ak jednoducho povieme kamere, aby súbor uložila ako intruder.mp4, zakaždým, keď do miestnosti vstúpi nová osoba, staré video bude úplne prepísané a vymazané. Aby sme si zabezpečili videozáznam všetkých (otravných rodičov alebo súrodencov), potrebujeme dynamický názov súboru. Môžeme použiť časovú knižnicu v Pythone na automatické zistenie aktuálneho dátumu a času a vloženie ich do názvu súboru videa.

Začnite s týmto kódom a vašimi obľúbenými AI nástrojmi na jeho testovanie:

Toto napíšte do svojho parent_detector.py súboru:

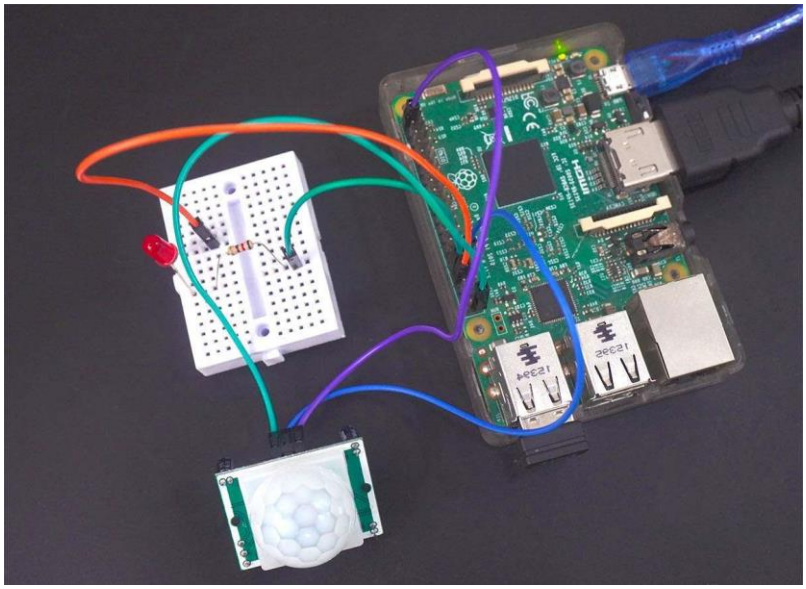
```
1. z gpiozero importovať MotionSensor
 2. z picamzero import fotoaparátu
 3.  Čas importu
 4.
 5. # 1. Inicializovať hardvér
 6. # Nastavte PIR senzor na GPIO 4
 7. pir = MotionSensor(4)
 8.
 9. # Vytvoriť objekt kamery na ovládanie modulu
10. cam = Camera()
11.
12. tlač ("Bezpečnostný systém ozbrojený. Čakanie na votrelcov...")
13.
14. # 2. Hlavná bezpečnostná slučka
15. zatiaľ čo je pravda:
16. #     Program sa tu zastaví, kým nie je zistený pohyb
17.     pir.wait_for_motion()
18. tlač ("UPOZORNENIE: Zistený pohyb!")
19.
20.     #3. Vygenerujte jedinečný názov súboru
21.     # Tým vzniká reťazec ako "20231025-143000" (RokMesiacDeň-HodinaMinútaSekunda)
22.     časová značka = čas.strftime("%Y%m%d-%H%M%S")
23.     názov súboru = f"intruder_{časová značka}.mp4"
24.
25.     # Začnite nahrávať videozáznam do nového spisu
26.     print(f"Nahrávanie videa: {názov súboru}")
```

```
kamera.start_recording(názov súboru)

    #4. Počkaj, kým bude miestnosť opäť prázdna
    pir.wait_for_no_motion()
tlač ("Návrh zastavený.")

    # Zastavte nahrávanie, aby ste video súbor dokončili a bezpečne uložili
kamera.stop_recording()
print("Systém sa vracia do pohotovostného režimu.")
```

Teraz, keď je váš kód úplne napísaný a hardvér bezpečne pripojený, je čas otestovať, či váš rodičovský detektor naozaj funguje v reálnom svete! Najprv kliknite na tlačidlo Run vo vašom programovacom prostredí Thonny, aby ste spustili skript. Keď program beží a čaká, zámerne mávnite rukou priamo pred PIR pohybovým detektorom, aby ste simulovali votrelca vstupujúceho do miestnosti. Okamžite sa pozrite na obrazovku počítača; mali by ste vidieť v oblasti konzoly vytlačené slová "Detekcia pohybu!", čo potvrdzuje, že senzor úspešne spustil kameru na začatie nahrávania.

Po spustení alarmu je potrebné otestovať vypínací mechanizmus. Vystúpte z zorného poľa senzora, zostaňte úplne nehybní alebo jemne zakryte bielu kupolu senzora rukou. Počkajte pár sekúnd, kým vám konzola oznámi, že pohyb sa zastavil a systém sa vráti do pohotovostného režimu. To znamená, že program bezpečne dokončil a uzavrel video súbor. Nakoniec otvorte správcu súborov na Raspberry Pi a prejdite do presne toho istého priečinka, kde ste uložili *parent_detector.py* skript. Teraz by ste mali vidieť úplne nový .mp4 video súbor, ktorý bude mať v názve jedinečný dátum a čas, presne tak, ako sme naprogramovali. Dvakrát kliknite na tento novo vytvorený súbor, aby ste si pozreli zaznamenané dôkazy, overili kvalitu videa a uistili sa, že váš uhol kamery dokonale zachytáva vchod!

Ďalšie obrázky:



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

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

16.2 Rozpoznávanie odhadu pózy YOLO

V tomto sprievodcovi si ich pripravíme s OpenCV a modelovou rodinou YOLO na Raspberry Pi 5. Pozrieme sa na niekoľko rôznych dostupných modelov YOLO, ako ich optimalizovať pre plynulejšie FPS, a tiež na to, ako využiť kľúčové dáta generované modelom, aby ste mohli do svojho ďalšieho projektu implementovať odhad pózy. Toto je jeden z najzábavnejších sprievodcov, aké sme za poslednú dobu vytvorili, tak poďme na to!

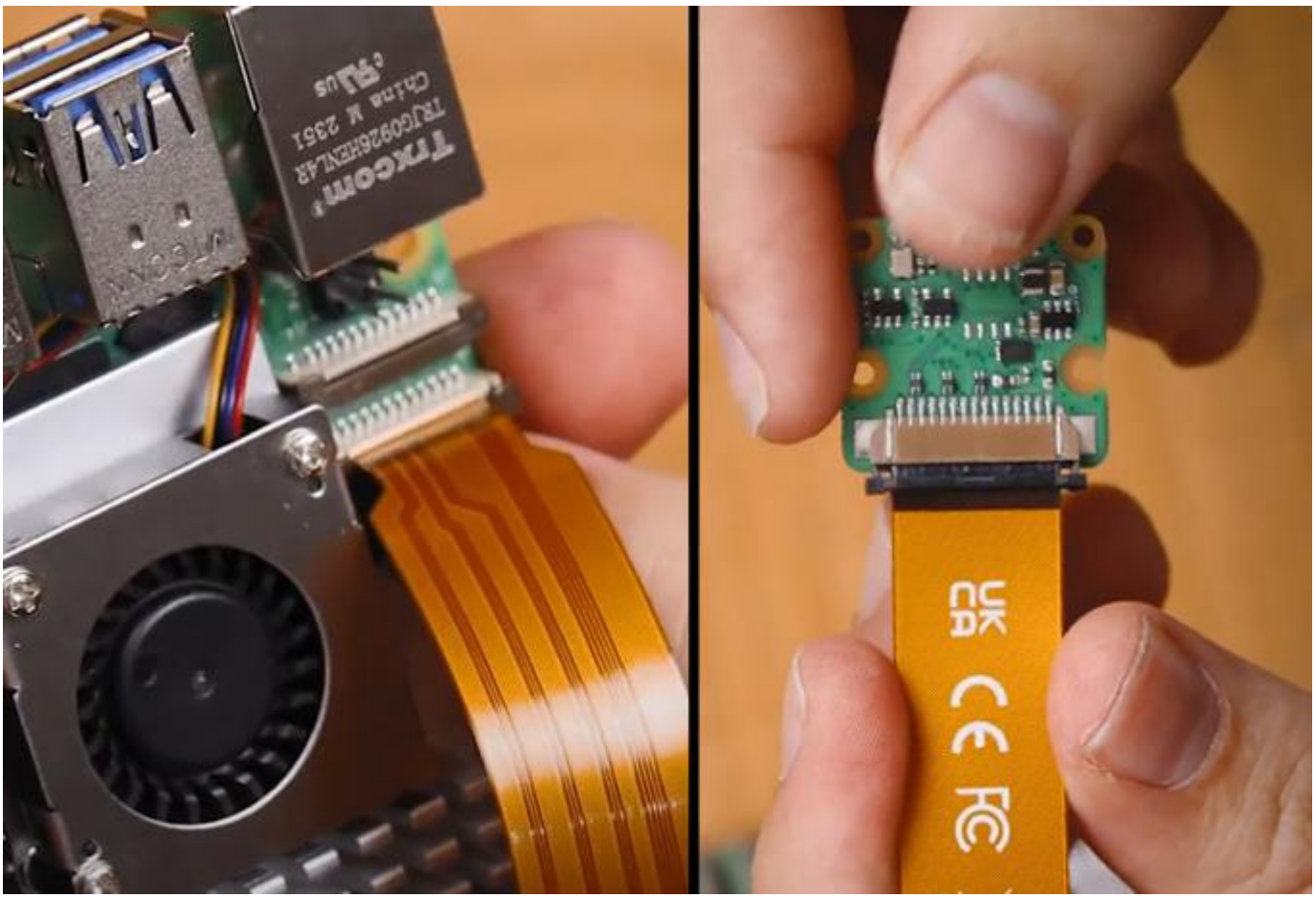
Na sledovanie tohto návodu budete potrebovať:

- Raspberry Pi 5 – Tu bude fungovať buď model s 4GB alebo 8GB. Aj keď by sa to technicky dalo urobiť na Pi 4, je oveľa pomalší ako Pi 5 a nebola by to príjemná skúsenosť, a práve preto sme na Pi 4 netestovali
- Pi Camera - Používame Camera Module V3
- Adaptér – Pi 5 má kábel CSI kamery inej veľkosti a váš fotoaparát môže mať starší hrubší, takže sa oplatí to skontrolovať. Camera Module V3 BUDE potrebovať jeden
- Cooling Solution – Používame aktívny chladič (počítačové videnie naozaj potlačí vaše Pi na maximum)
- Napájanie
- Micro SD karta - Minimálne 16GB veľkosti
- Monitor a kábel Micro-HDMI na HDMI
- Myš a klávesnica

Montáž hardvéru

Čo sa týka montáže hardvéru, tu je to dosť ľahké. Hrubšiu stranu kábla pripojte ku fotoaparátu a tenšiu stranu k Pi 5. Tieto konektory majú záložku – zdvihnite ich a potom vložte kábel do slotu. Keď už tam pekne a správne sedí, zatlačte západku späť dole, aby ste kábel upevnili.

Len dávajte pozor, pretože tieto konektory fungujú len v jednej orientácii a môžu byť krehké, preto ich neohnite príliš (trochu je v poriadku).



Inštalácia Pi OS

Najprv musíme nainštalovať Pi OS na micro SD kartu. Pomocou **Raspberry Pi Imageru** vyberte Raspberry PI 5 ako zariadenie, Raspberry Pi OS (64-bit) ako operačný systém a vašu microSD kartu ako úložné zariadenie.

Rovnaký postup ako pri PiRacer.

Nastavenie virtuálneho prostredia a inštalácia knižníc

S príchodom Bookworm OS v roku 2023 sme teraz povinní používať virtuálne prostredia (venv), pretože ide o izolovaný priestor na Pi, kde môžeme experimentovať bez rizika poškodenia zvyšku nášho Pi OS alebo projektov. V tomto sprievodcovi máme všetky potrebné príkazy a inštrukcie.

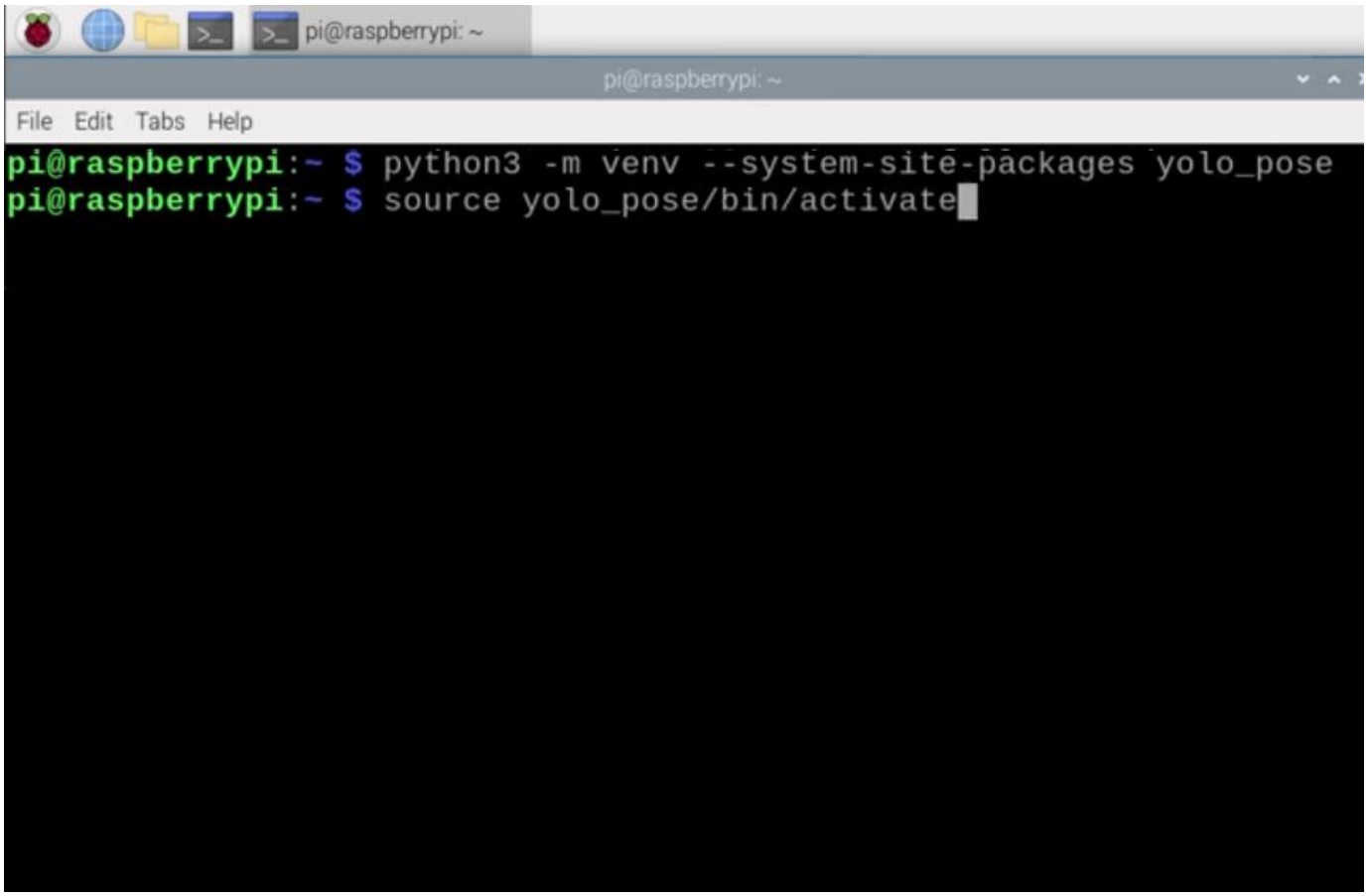
Na vytvorenie virtuálneho prostredia otvorte nové okno terminálu a zadajte:

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

Po vytvorení venv môžeme do neho vstúpiť zadaním:

zdroj yolo_pose/bin/aktivovať

Po tomto zobrazení uvidíte názov vo virtuálnom prostredí naľavo od zeleného textu – to znamená, že v ňom správne pracujeme. Ak budete niekedy potrebovať znovu vstúpiť do tohto prostredia (napríklad ak zatvoríte okno terminálu, prostredie opustíte), stačí znova zadať zdrojový príkaz vyššie.



Teraz, keď pracujeme vo virtuálnom prostredí, môžeme začať inštalovať potrebné balíky. Najprv sa uistite, že PIP (správca balíkov v Pythone) je aktuálny zadaním troch nasledujúcich riadkov:

Aktualizácia Sudo APT

sudo apt install python3-pip-y -y

inštalácia pip -U pip

Potom nainštalujte balík Ultralytics s:

PIP inštalujte ultralytics[export]

Milí ľudia z Ultralytics patria medzi kľúčových vývojárov a údržbárov najnovších modelov YOLO. Tento ich balík urobí veľkú časť ťažkej práce a nainštaluje OpenCV, ako aj všetku potrebnú infraštruktúru na prevádzku YOLO.

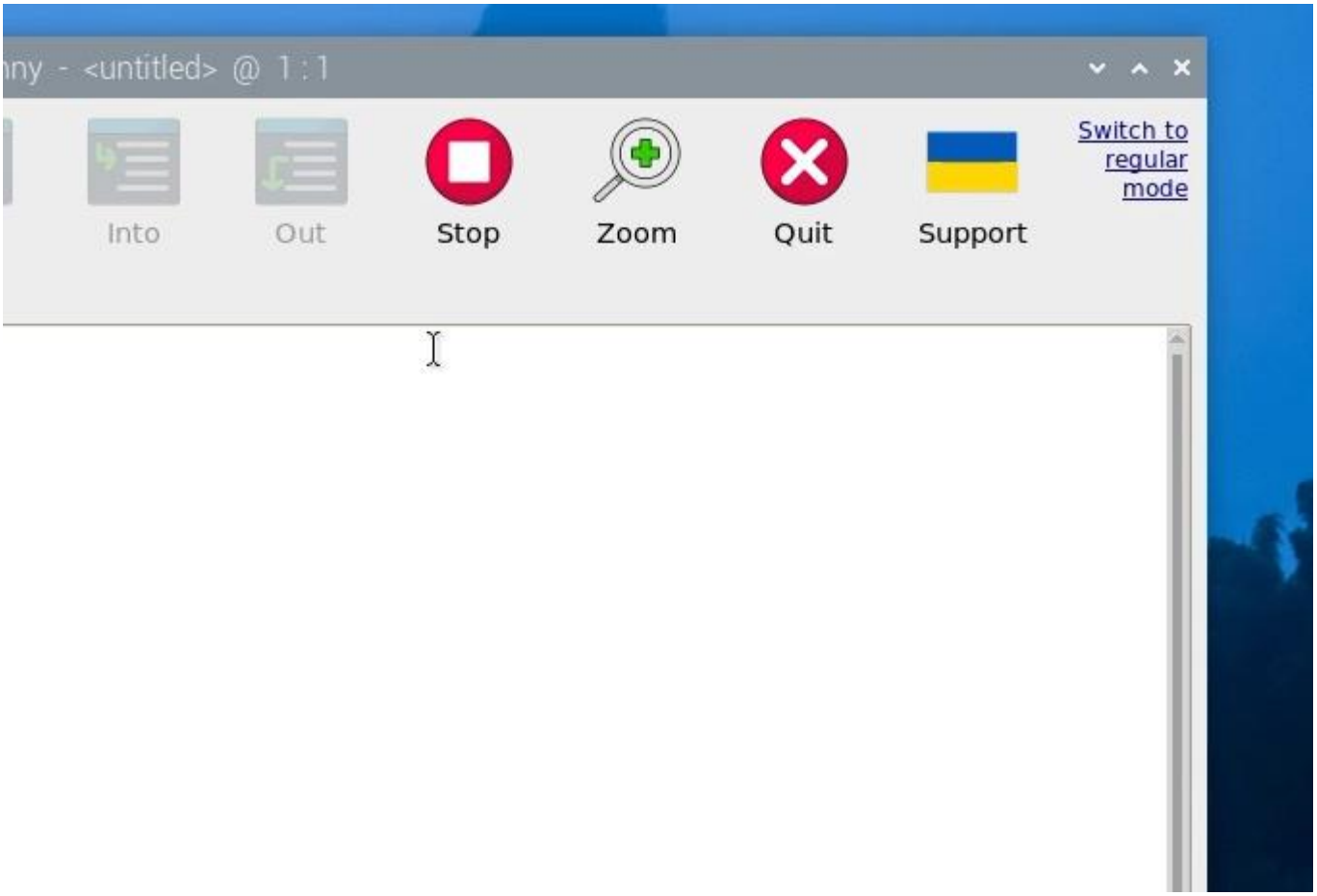
Tento proces tiež nainštaluje pomerne veľké množstvo ďalších balíkov, a preto je náchylný na zlyhania. Ak vaša inštalácia zlyhá (zobrazí sa celá stena červeného textu), stačí znova zadať inštalačný riadok Ultralytics a malo by to pokračovať. V zriedkavých prípadoch môže byť potrebné inštalačnú linku niekoľkokrát zopakovať.

Keď sa dokončí inštalácia, reštartuj Raspberry Pi. Ak chcete byť pokročilým používateľom, môžete to urobiť zadaním do shellu:

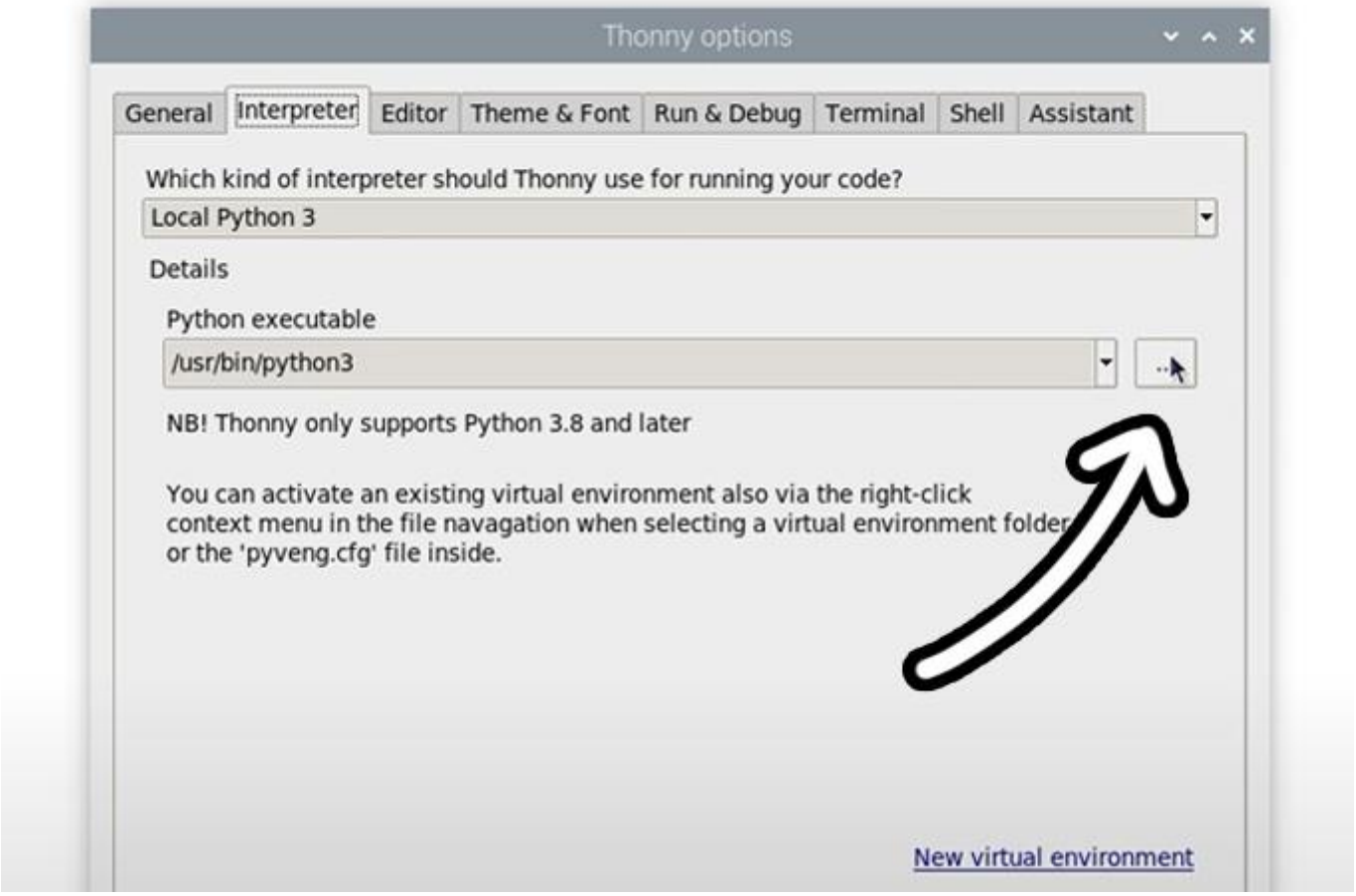
Reštart

Máme ešte jednu úlohu, a tou je nastaviť Thonny tak, aby používal virtuálne prostredie, ktoré sme práve vytvorili. Thonny je program, z ktorého budeme spúšťať všetok náš kód, a potrebujeme, aby fungoval z toho istého venv, aby mal prístup ku knižniciam, ktoré sme nainštalovali.

Prvýkrát, keď otvoríte Thonny, môže to byť v zjednodušenom režime a v pravom hornom rohu uvidíte "prepnúť do bežného režimu". Ak je to prítomné, kliknite naň a reštartujte Thonny zatvorením.



Teraz vstúpte do ponuky možností interpretera výberom Spustiť > Konfigurovať interpreta. Pod možnosťou spustiteľného súboru v Pythone je tlačidlo s tromi bodkami. Vyberte ho a prejdite do Python spustiteľného súboru vo virtuálnom prostredí, ktoré sme práve vytvorili.



Ten bude umiestnený pod home/pi/yolo_pose/bin a v tomto súbore budete musieť vybrať súbor s názvom "python3". Stlačte ok a teraz budete pracovať v tomto venv.

Kedykoľvek otvoríte Thonny, automaticky bude fungovať mimo tohto prostredia. Prostredie, z ktorého pracujete, môžete zmeniť výberom z rozbaľovacieho menu pod Python spustiteľným súborom v tom istom menu možností interpretera. Ak chcete opustiť virtuálne prostredie, vyberte option bin/python3.

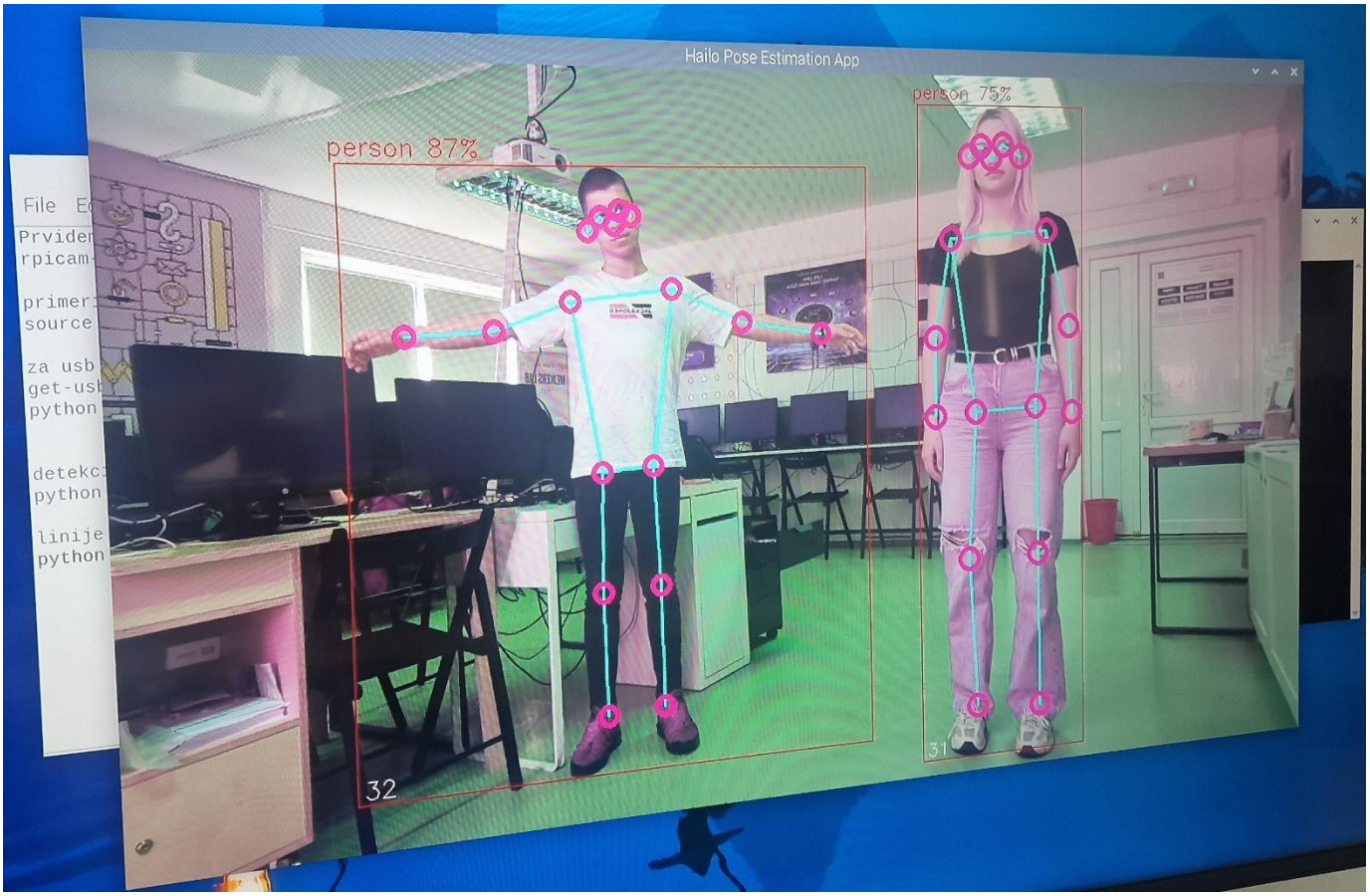
Odhad bežiaceho postoja

Teraz, keď máme nainštalované knižnice a Thonny pracuje vo virtuálnom prostredí, môžeme spustiť náš skript na odhad polohy. Pokračujte a rozbalte **zip priečinok projektu** (stiahnite si ho zo servera) na pohodlné miesto, napríklad na desktop. Tam nájdete prvý scenár, ktorý použijeme *"pose demo.py".* Otvor to, Thonny, a stlač veľké zelené tlačidlo behu. Pri prvom spustení sa môže automaticky nainštalovať niekoľko potrebných vecí navyše a po pár sekundách by sa mal zobraziť náhľad s vaším odhadom pózy.

Malo by sa tu diať niekoľko vecí. Po prvé, YOLO sa bude snažiť detegovať ľudí, a ak jedného rozpozná, nakreslí okolo neho rámeček s hodnotením dôvery na vrchu. Dôležité je, že bude umiestňovať body tam, kde si myslí, že sú niektoré podstatné miesta na vašom tele (tieto sa nazývajú kľúčové body), a nakreslí medzi nimi čiary, aby

odhadol postoj a orientáciu osoby. V pravom hornom rohu bude tiež FPS, na ktorom tento systém beží (čo o chvíľu zlepšíme).

A to je všetko! S týmito niekoľkými krokmi už máme odhad pozície na Pi!

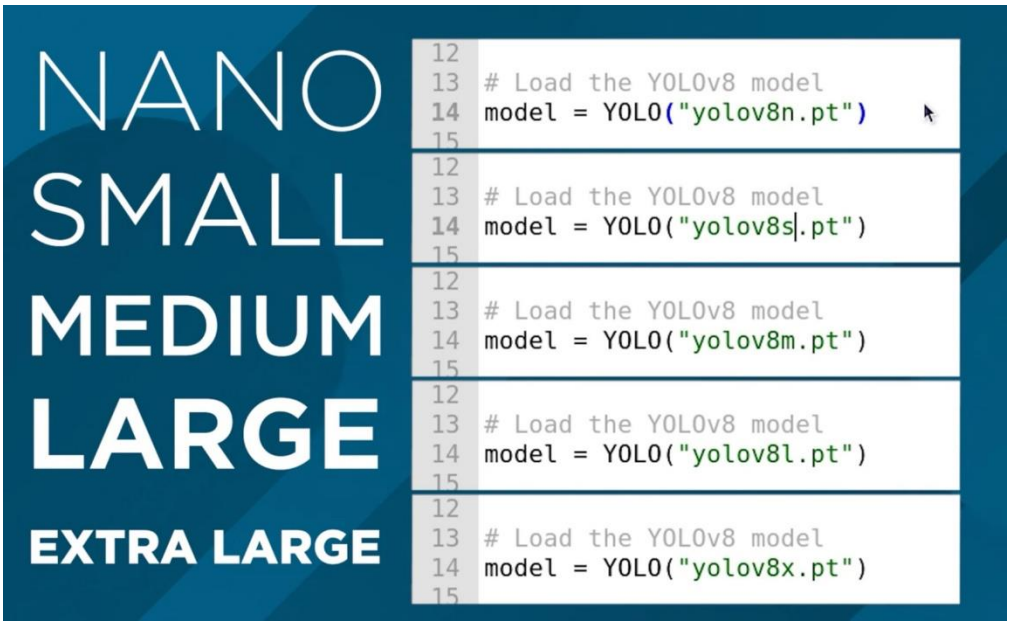


Zmena modelov YOLO

Doteraz používame YOLO11 a jednou z krás tohto balíka Ultralytics je, že môžeme jednoducho vymeniť jeden riadok v kóde a úplne zmeniť model. Môžeme to použiť na spustenie pokročilejšieho modelu YOLO11, alebo dokonca staršieho modelu. Všetko, čo potrebujete zmeniť, je tento riadok tu v nastavení:

```
# Načítaj náš model YOLO11
model = YOLO("yolo11n-pose.pt")
```

Táto séria momentálne používa nano model, ktorý je najmenší, najmenej výkonný, ale najrýchlejší model YOLO11, a môžeme túto radu zmeniť tak, aby používala jednu z rôznych veľkostí tohto modelu, zmenou jedného písmena za "11", ako je znázornené vpravo. Ak zmeníte tento riadok na inú veľkosť modelu a spustíte ho, skript automaticky stiahne nový model (čo môže byť v stovkách Mb pre väčšie modely).



Rozdiel medzi týmito modelmi je kompromisom medzi výkonom odhadu pózy a FPS. Čím väčší model, tým lepšie odhaduje časti tela, ktoré kamera nemusí vidieť, ako aj zložitejšie uhly a snímky s viacerými ľuďmi, avšak môžete očakávať, že sa spracuje len jeden snímok každých 10 sekúnd! V ďalšom kroku to zvýšime.

Nano model na druhej strane beží najrýchlejšie, dosahuje približne 1,5 FPS bez optimalizácie, ale nemá výpočtový výkon väčších modelov. Čo sa týka odhadu pózy, väčšinou stačí nano model, pretože zvyčajne stačí na vaše potreby, ale ak potrebujete niečo výkonnejšie, stále zvyšujte veľkosť modelu, aby vyhovovala vašim potrebám.

V tejto línii môžeme tiež zmeniť verziu YOLO running. Ak chcete, môžete sa vrátiť k staršiemu modelu, alebo použiť novší. Tento sprievodca bude časom zastaraný a ak Ultralytics vydá YOLO13, mali by ste jednoducho zmeniť radu na nasledujúcu, aby ste mohli začať používať novšiu verziu YOLO:

```
# Načítaj náš model YOLO11
model = YOLO("yolo13n-pose.pt")
```

Zvyšovanie rýchlosti spracovania

Existujú dve veci, ktoré môžeme urobiť na zvýšenie FPS na Pi a najefektívnejším spôsobom je konvertovať model do formátu nazývaného NCNN. Toto je formát modelu, ktorý je viac optimalizovaný na procesory založené na ARM, ako sú Raspberry Pi. Otvorte skript s názvom "ncnn conversion.py" a nájdete nasledujúce:

```
z importu ultralytics  YOLO

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

# Exportovať model do formátu NCNN
model.export(format="ncnn", imgsz=640) # vytvára 'yolov11n-pose_ncnn_model'
```

Na použitie tohto skriptu najprv špecifikujte model, ktorý chcete konvertovať. Toto používa rovnaké pomenovávacie konvencie, o ktorých sme hovorili v predchádzajúcej časti. Potom sa ako výstupný formát a rozlíšenie špecifikuje formát modelu "ncnn". Zatiaľ nechajte predvolenú hodnotu 640. Prvýkrát, keď spustíte

tento skript, stiahne sa ešte niekoľko ďalších vecí, ktoré potrebuje, ale skutočná konverzia by mala trvať len pár sekúnd.

Keď je to hotové, v priečinku, v ktorom sa nachádzajú skripty, nájdete nový priečinok s názvom niečo ako "yolo11n-pose_ncnn_model". Skopírujte názov tohto súboru a vráťte sa k nášmu demo skriptu z predtým.

Teraz musíte skriptu povedať, aby použil tento model, ktorý sme vytvorili, zmenou modelovej línie na názov priečinka, ktorý práve vytvoril. Malo by to vyzerať asi takto:

```
# Načítaj náš model YOLO11
model = YOLO("yolo11n-pose_ncnn_model")
```