УНИВЕРЗИТЕТ
У НОВОМ САДУ

ФАКУЛТЕТ
ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Југославија
Деканат: 021 350-413; 021 450-810; Централа: 021 350-122
Рачуноводство: 021 58-220; Студентска служба: 021 350-763
Телефакс: 021 58-133; e-mail: ftndean@uns.ns.ac.yu

PROJEKAT IZ PRIMENJENE ELEKTRONIKE

NAZIV PROJEKTA:

Automobil na remote upravljanje

TEKST ZADATKA:

Projektovati tenkic kojim se upravlja preko wifi-ja ili bluetooth-a uz pomoc raspberry pi-ja

MENTOR PROJEKTA:

dr Vladimir Rajs

PROJEKAT IZRADILI:

Petrovic Nenad EE69/2018

DATUM ODBRANE PROJEKTA

Sadržaj

1. Uvod

Zadatak je napraviti sistem za daljinsko upravljanje automobilom. Izrada projekta se svodi na koriscenje raspberry pi-ja kao mikroracunara I drajvera DC motora I USB camere u traženu celinu. Svaki podsistem ima svoju funkciju i realizovan je kao električno kolo sa određenim komponentama kako bi ispunio željenu funkciju.

U narednim poglavljima će biti objašnjen rad komponenti i u koje svrhe su korištene, opis funkcionalnosti celog sistema i detaljan opis.

U drugom poglavlju opisani su svi delovi sistema , koliko ih ima I softver koriscen za implementaciju projekta.

U trecem poglavlju opisani su delovi u detalje kao I nacin njihove upotrebe , razlog zasto su korisceni , prednosti , mane I alternative I date su slike komponenata.

U cetvrtom poglavlju dat je algoritam rada kao I konkretna implementacija u softveru.

U petom poglavlju dat je zakljucak o funkcionalnosti projekta , mogucim dodacima I unapredjenjima I kratko je diskutovano , sta bi moglo bolje I zasto nije bilo realizovano.

U sestom poglavlju dat je kratak opis kako se sistem koristi , bez ulazenja u detalje .

U sedmom poglavlju data je koriscena literatura , sa linkovima ka datasheetovima svih delova I korisnim tutorijalima kako se koriste I citaju.

Dodatak ilustritativno pokazuje izgled mikroracunara kao I ulogu svakog sub-modula I svih pinova.

2. Opis izrade predmeta projekta

Dva podsistema se primarno dele po njihovoj funkciji, jedan ima funkciju generisanja i slanja kontrolnih signala - predajna ploča (raspberry pi miniracunar), a drugi ima funkciju prijema i obrade tih signala - prijemna ploča (L298N drajver) I servo motor I USB camera.

Delovi	Komada
Raspberry Pi model 4B	x1
L298N motor driver	x1
Servo (stavi koji je	x1
Logitech C270	x1
Tank Chasis	x1
Raspberry Pi Power Supply	x1
RalRT5572 Wireless Adapter	x1

Tabela 1.Spisak delova koriscenih u projektu

Raspberry pi kontrolise L298N driver , camera je povezana na njega I on da je napajanje drajvera (nazalost samo 5V).

Svi GPIO (general purpose input/output) pins su digitalni sto moze biti I prednosti mana , za razliku od drukih mikrokontrolera koji imaju analogne I digitalne pinove.U ovom projektu nema potrebe za analognim pinovima I zanimljivo je da postoje pinovi koji su odredjeni za serijske komuikacije (I2C I SPI).Moguce je kontrolisati neke od pinova I putem interneta.

Za projekat je bio potreban:

Posto je na ploci podignut Linux OS moguce je korisitti bilo koji programski jezik I zato je ovaj modul ekstremno popularan za projekte gde nije potrebno koristiti GPIO pinove , vec se raspberry koristi kao headless uredjaj.Za potrebe ovog projekta prirodan izbor je ili python ili C/C++ ili Java.Odluceno je da se koristi python.

3. Opis fizičke realizacije predmeta projekta

Glavna komponenta koja se koristi je Raspberry Pi model 4B . On je direktno povezan sa svakom komponentom na ploči. Mikroracunar služi za primanje komandnih signala tastature , njihovo tumačenje I togglovanje pinova.Odredjena kombinacija pinova (on & off) se koristi za upravljanje H-mosta I time se pokreće tenkic.Posto je python interpreterski jezik nema potrebe za kompajlerom , vec se pokrece odmah.

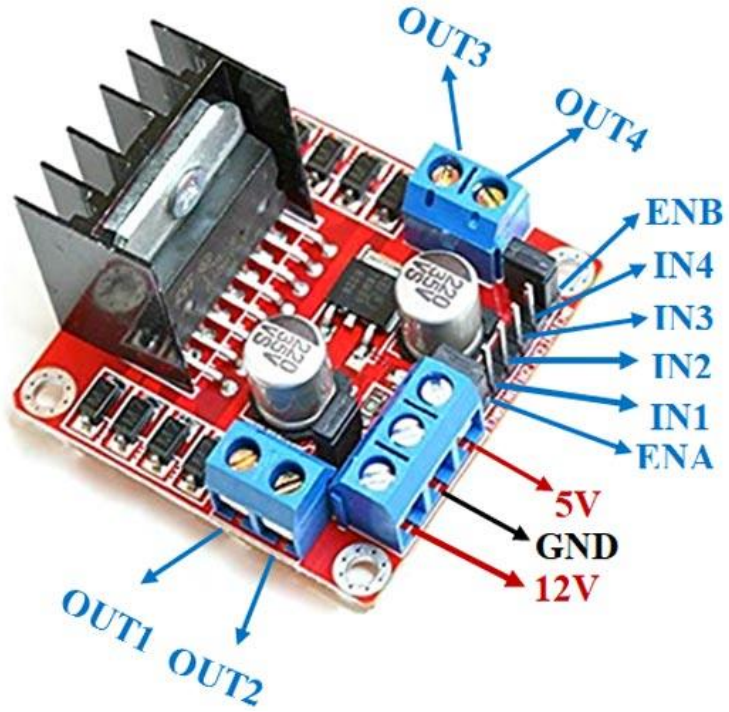
3.1 Opis svih podsistema projekta

1. Raspberry Pi model 4B:

Slika 1. Izgled kola napajanja na šematiku

Kao sto je vec receno ovo je mozak sistema , koriscen je usb port , napajanje je dovedeno preko USB tipa C , GPIO pinovi su povezani na drajver motora kao I pin za napajanje.Detaljni izgleda schematika je dat u prilogu.

2. L298N motor driver:



Slina 2.L298N driver

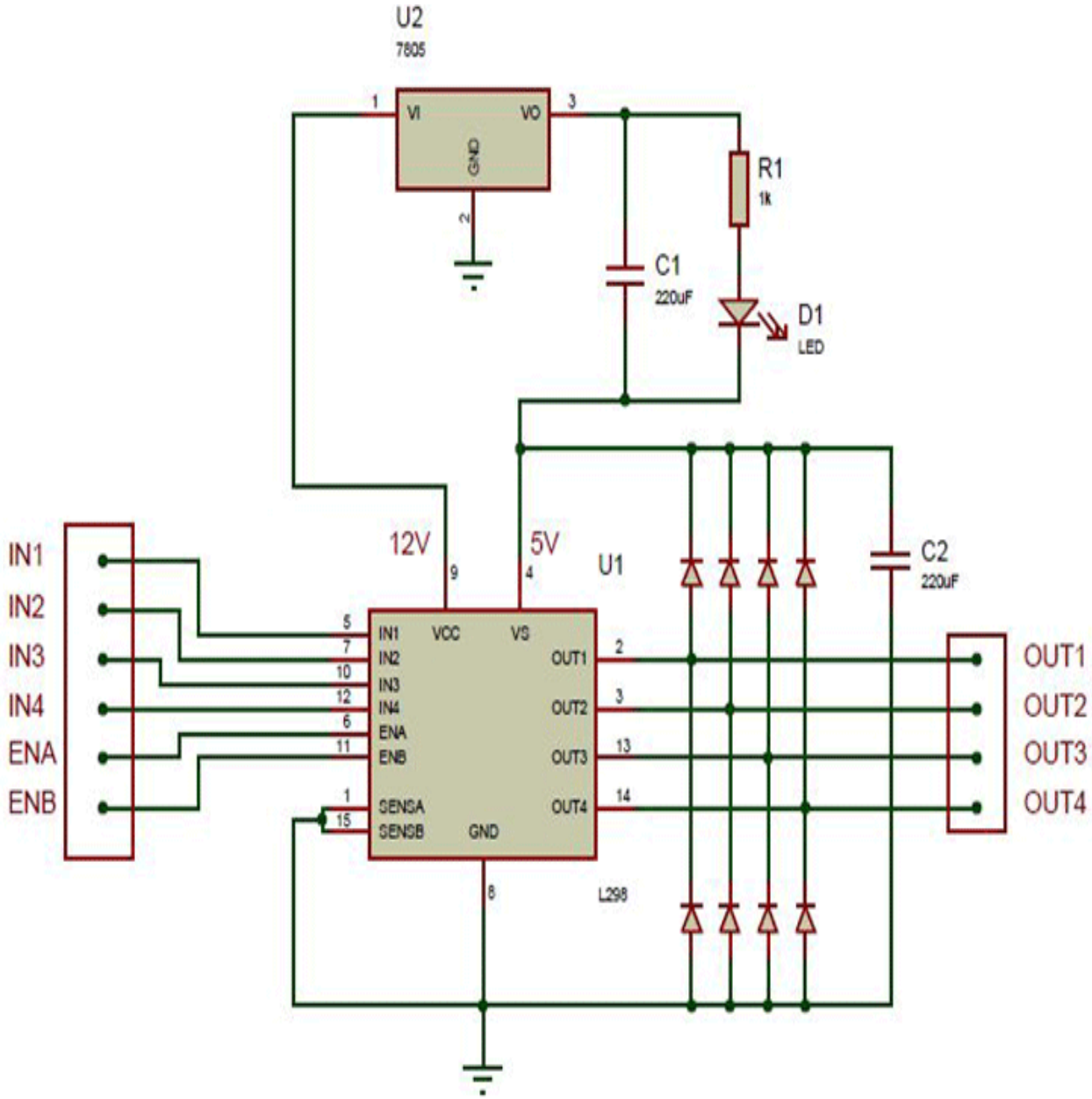
Ovo je dobro poznati I vrlo popularni full bridge dc motor drajver , karakterisu ga 4 pina za kontrolu logickih kola I ENA / ENB pinovi koji su korisceni kao PWM signali.Napon napajanja na motoru je doveden sa pina od 5V ,isto tako I za masu.Ovaj drajver ima na sebi LDO koji jos dodatno umanjuje napon doveden na njegov ulaz , pa ovo moze da pravi problem ako je prikljuceni napon dosta nizak , ali nije problem za visoke napone (reda 12V I vise).Moze se koristiti za kontrolu do 4 motora u jednom smeru ili 2 motora koji imaju promenjiv smer.

Karakteristike drajvera:

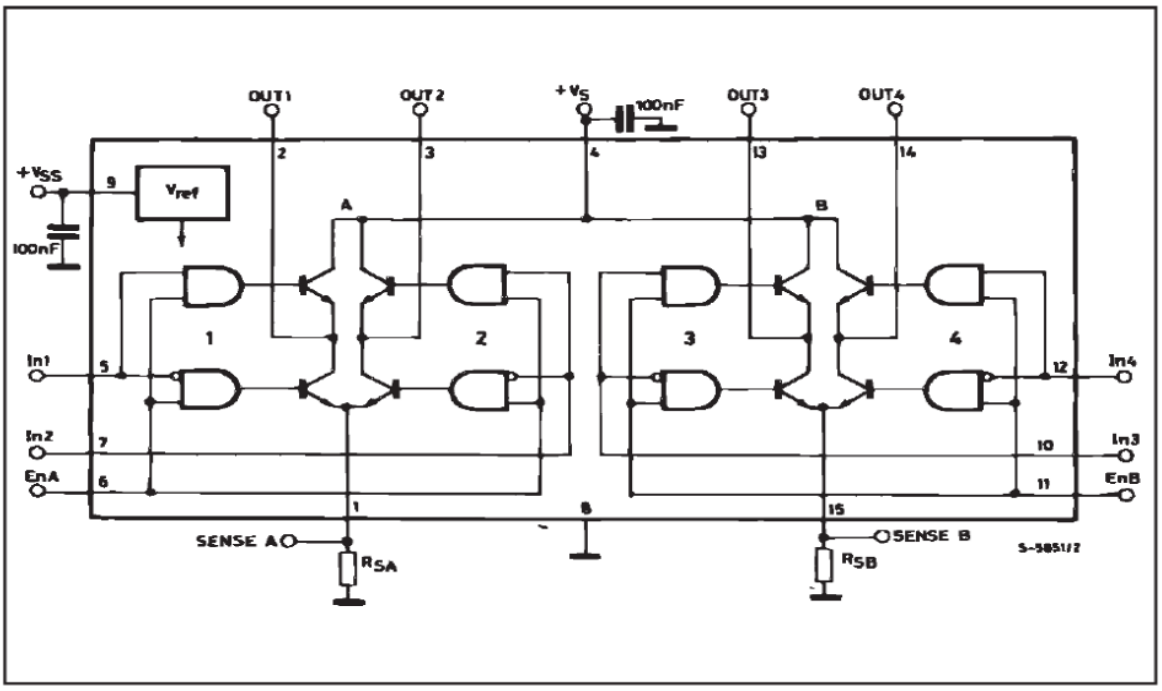
- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Max radni napon na motoru: 46V
- Max struja motora : 2A
- Logicki naponski nivoi: 5V
- Napon na drajveru: 5-35V
- Struja drajvera :2A
- Struja za digitalne komponente :0-36mA
- Disipacija (max): 25W
- Rs optornik u H-mostu za svaki motor
- Hladnjak
- Power-On LED

Ovaj modul je koriscen jer je bio dostupan I nije bilo nikakvih problema sa njim , ali ako postoji potreba za alternativom moguci moduli koji su veoma slicni su :

- L9110S
- Adafruit DRV8833



Sika 3 L298N schematic



Slika 4. H-Most schematic unutar L298N

3. Logitech c270:

Slika 5. Logitech C270 camera

Logitech c270 kamera je koriscena jer predstavlja jednu od najboljih “budzet” verzija kamere na trzistu.Najcesce se koristi za web konferencije I sastanke I u hobby svrhe , takodje ova kamera je bila deo proslog nezavrsenog projekta vezanog za rubikovu kocku I ocitavanje I prepoznavanje boja u razlicitim uslovima osvetljenja .Pruza 30 FPS u 720p , sto je solidno za strimovanje sporijih slika I videa I daje solidnu rezoluciju.Neke karaktereisticne osobine :

Karakteristike kamere

- 720p 30FPS
- Viewing angle 60-80°
- Jeftina
- Plug and play
- Low fps , pa nije bas pogodna za pracenje objekata koji se brzo krecu
- Radi lose u uslovima slabije osveteljenosti

4. Servo motor MG996R:

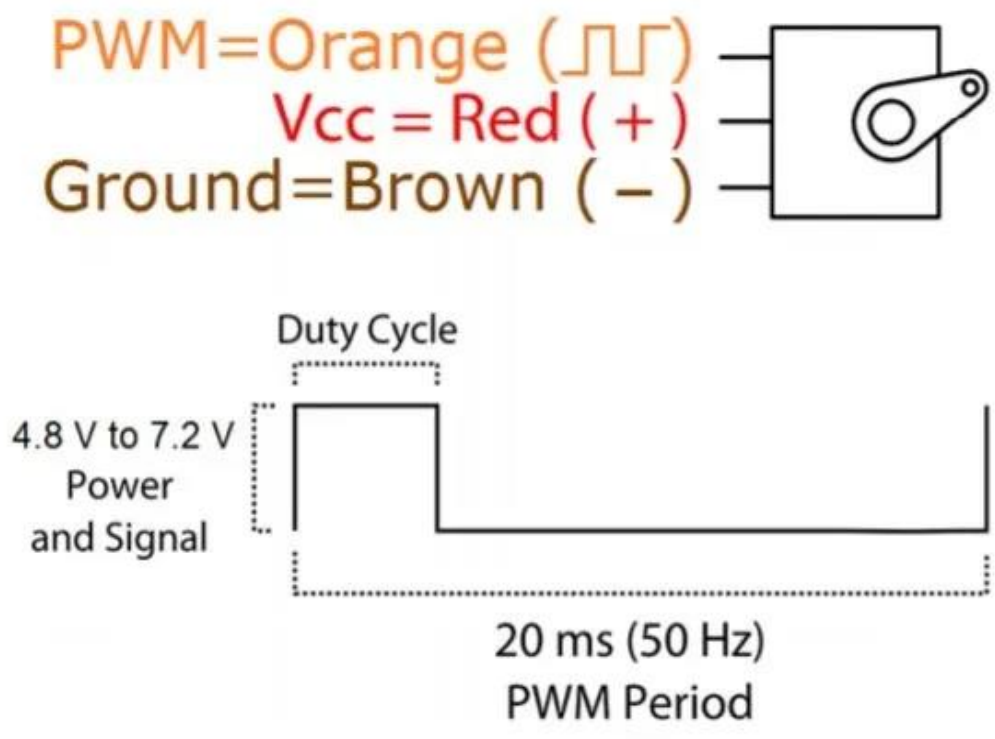
Slika 6. Servo mg996r

MG996R je cak I suvise dobar za ovaj sistem , jer je njegova jedina uloga da rotira kameru levo ili desno.Posto kamera nema ni 400g , ovo nece biti nikakav problem jer servo motori obicno imaju dosta velik moment sile za telo.Ovaj motor je vrlo popular medju hobistima I nasao je mnoge primene svuda pogotovo u robotici , RC kolima I letelicama.Tower Pro je jedan od poznatijih proizvodjaca servo motora I ovaj proizvod je dostupan svuda I karakterisa ga niska cena I visoka pouzdanost (“good bang for your buck”).Neke od karakteristika ovog motora su :

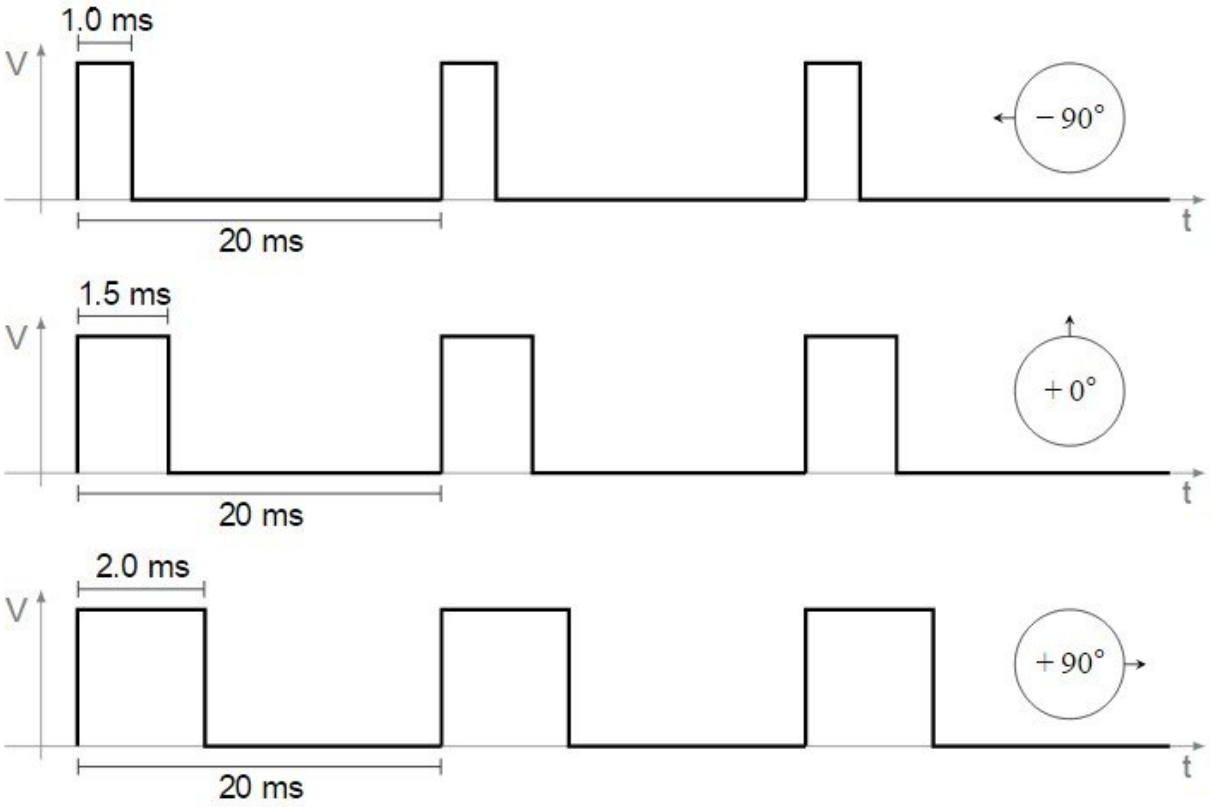
Karakteristike servo motora:

- Stall torque : 9.4 kg/cm (4.8V)
- Radni napon 4.8 – 6.6V
- Deadband width 1us (veoma dobro)
- Operating speed: 0.19sec/60degree (4.8v)
- Mass of 55g
- Rotacija od 0-180°

Vazno je znati koja boja odgovara kom pinu I obicno postoje standardi , cvreni ili “vruci” kraj je tipicno napon napajanja (4.8-6.6V u slucaju MG996R) , braon je masa , dok je narandzasti controlni signal.



Slika 7.Boja zica servo motora

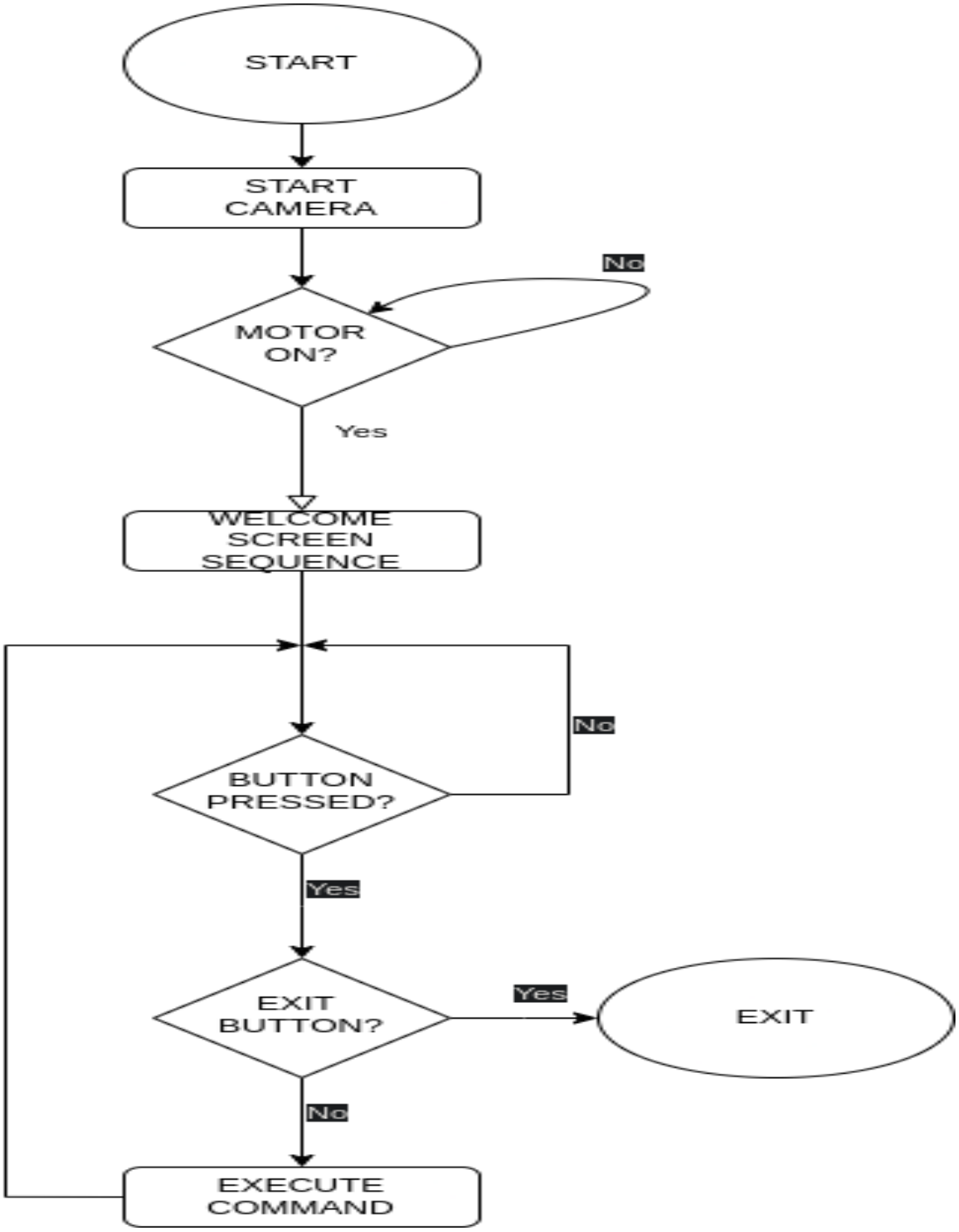


Slika 8.Primer PWM signala za kontrolu servo motora

Servo je najjednostavniji za kontrolu od svih tipova motora (DC , stepper , servo) I zasniva se kontroli faktora ispune od 1ms za roaciju od -90° do 2ms za +90° , svi uglovi izmedju se dobijaju menjanjem faktora ispune izmedju 1ms I 2ms.Servo je pogodan jer nije potrebno posedovati nikakve drajvere za njega , posto on u sebi vec ima sistem negativne povratne sprege I samim tim on “pamti” pocetni polozaj svaki put nakon sto se sistem resetuje , nije potrebno imati ni enkoder,Karakterise ih mala cena , male dimenzije , jednostavnost I dosta velika sila kojom mogu da pomeraju objekte kao I velika zadrzna sila .

4. Opis funkcije rada predmeta projekta

4.1 Ilustracija toka rada



Slika 9.Algoritam rada

4.2 CODE

4.2.1 Selekcija frekvencije PWM-a

Pre nego sto pocnemo sa kontrolom DC motora preko PWM-a potrebno je selektovati pogodnu frekvenciju ovog kontrolnog signala.U daljem izlganju dat je ukratko predlog kako bi to moglo da se izvede I zasto je uzeta bas frekvencija moja je uzeta.

Potrebno je odluciti koju frekvenciju PWM-a koristiti , pa je potrebno znati prednosti I mane visih , odnosno nizih frekvencija , tipicno se uzima opseg (1k-100kHz) I pod visokim frekvencijama se podrazumeva opseg preko 20kHz.

Niske frekvencije	Visoke frekvencije
Buka u cujnom opsegu	Buka je pomerana van cujnog ospega
Veci strujni spajkovi	Manji strujni spajkovi
Motor moze poceti da zapinje	Motor tipicno nema zapinanje
Veci gubici u namotajima	Manji gubici u namotajima
Manje optrecenje za digitalni blok	Vise opterecuje sistem

Tabela 2.Prednosti I mane viskoke I niske frekvencije

Nakon onoga ostalo je još da se odredi faktor ispune PWM signala u slučaju kada autić treba da ide normalno i brzo. Uglavnom su vrednosti faktora ispune u pocetku režimu od 35% , posto je napajanje drajvera 5V I usled postojanja LDO regulatora na L298N drajveru to je napon na motoru oko 3V za pwm od 100% , povecanjem pwm-a moze se doci do 100% u koracima od 20% , isto vazi I za usporavanje.Ovde bi trebalo napomenuti da ovo nije realisticna situacija u kolima ili motociklima ili tenkovima.

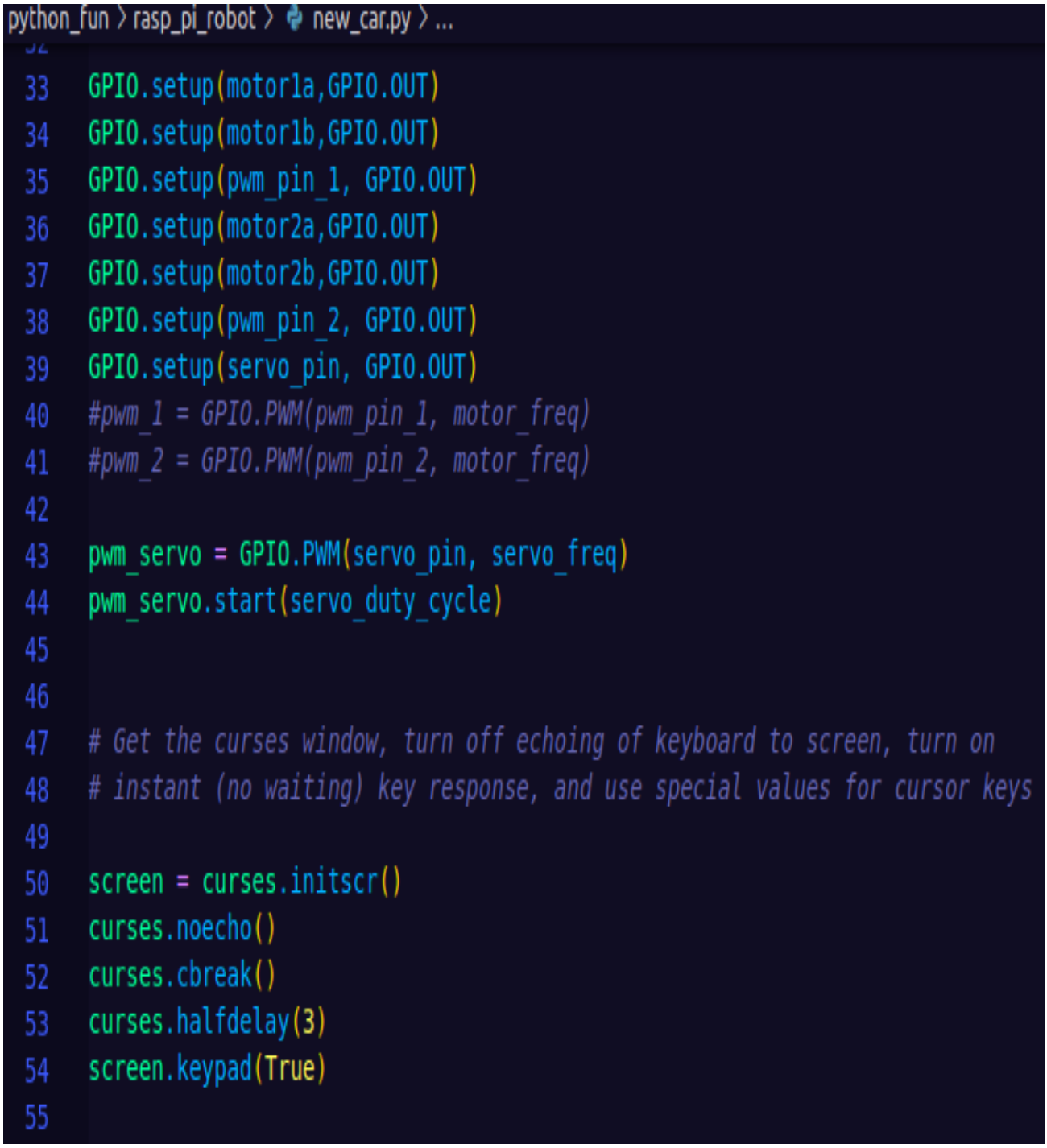
4.2.2 Implementacija koda

python_fun > rasp_pi_robot > new_car.py > ...

```
1   # import curses and GPIO
2   import curses
3   import RPi.GPIO as GPIO
4   import time
5
6   GPIO.setwarnings(False)
7
8       #set GPIO numbering mode and define output pins
9
10  GPIO.setmode(GPIO.BOARD)
11
12  global lower_speed_limit
13  global upper_speed_limit
14  global servo_duty_cycle
15
16  lower_speed_limit = 35                  #min duty cycle
17  upper_speed_limit = 100                 #max duty cycle
18  servo_duty_cycle  = 7.5                 #needs to be changed
19
20  motor1a   = 7
21  motor1b   = 11
22  pwm_pin_2 = 22
23
24  motor2a   = 13
25  motor2b   = 16
26  pwm_pin_1 = 15
27
28  servo_pin = 37
29
30  motor_freq = 2000
31  servo_freq = 50
```

Slika 10.Primer koda , inicijalizacija pinova (1)

Ovaj deo koda je zaduzen za inicijalizaciju pinova.Naime potrebno je odabrati pinove na raspberiju koji nisu namenski (to znai da se ne koriste za serijske komunikacije (I2C , SPI…) I njih spojiti sa L298N drajverom.To je uradjeno na ploci I ovde je softverski receno koji ce se pinovi koristiti.Potom su napravljene globalne promenjive da bi moglo da im se pristupa svuda.To su gornjei donje ogranicenje PWM-a I servo pwm.Takodje je zadata frekvencija PWM signala kojim se drajvuje motor kao sto je I bilo diskutovano.Frekvencija servo motora je standarnih 50Hz.



Slika 11.Primer koda , inicijalizacija pinova (2)

Nakon sto je ovo uradjeno potrebno je “ukljuciti” pinove , to se radi tako sto se postavi da su ti pinovi izlazni , pa mogu da salju signal kroz zice.Na srecu raspberi ima ugradjen pwm modul (hardverski realizovan) pa nije potrebno praviti pwm softverki.Curses modul sluzi za hvatanje inputa od korisnika.Input moze biti preko tastature ili preko misa.Nazalost sa ovim modulom nije moguce koristiti I gamepad.

```
def main(stdscr):

    global lower_speed_limit
    global upper_speed_limit
    global servo_duty_cycle

    motor_speed = 35

                    # Initialize curses


    curses.curs_set(0)
    stdscr.clear()
    stdscr.nodelay(1)               # Make getch() non-blocking
    curses.start_color()
    curses.init_pair(1, curses.COLOR_CYAN, curses.COLOR_BLACK)
    curses.init_pair(2, curses.COLOR_RED, curses.COLOR_BLACK)

    key_is_pressed = False
    motor_is_on = False

    try:
        while True:

            char = screen.getch()
            if char == ord('q'):

                curses.nocbreak()
                stdscr.keypad(False)
                curses.echo()
                curses.endwin()
                pwm_1.stop()
                pwm_2.stop()
                GPIO.cleanup()

                break
```

Slika 12.Primer koda , main loop

Prateci algoritam rada vidimo da je kamera nezavisna I radi uvek.Program se vrti u while True petlji I ceka neki input .Ovo je realizovano pomocu curses modula u pythonu I ovo je primer interrupta , gde neki dogadjaj zapocinje radnju.Ovim se postigla visoka brzina odziva I zaobidjeni su potencijalni problemi koje bi blocking methode mogle da uvedu (deadlock I los odziv kao I dosta tesko debagovanje su primeri nekih

klasicnih problema).

Sada je potrebno implementirati kontrol logiku , naime sta se desava kada se pritisne neko dugme.Kombinacije su sledece :

- q - izlaz iz programa
- t - upali motor
- w - ubrzaj
- s - uspori
- key_up (strelica nagore) – napred
- key_down (strelica nadole) – nazad
- key_left (strelica nalevo) – levo
- key_right (strelica nadesno) – desno
- f - rotacija kamere u desno
- g -rotacija kamere u levo

Bilo koje drugo dugme ne radi nista , osim ako nije Ctrl+z , sto je u python-u keyboard interrupt I komanda za exit.

Na slikama koje slede dat je prikaz sta se desava prilikom pritiska na odredjeno dugme. Deo koji sluzi za ispisivanje u terminal je samo vizuelna indikacija sta se desava I nema nikakvu ulogu u kontroli osim da pomogne vozacu.Ako se vratimo na schematic H-mosta (slika 4) , onda nije tesko zakljuciti koje kombinacije pinova in1 , in2 , in3 , in4 daju komande za napred , nazad , levo , desno.Ovde je bitno napomenuti da tenk skrece tako sto mu se jedna gusenica ne okrece .Na primer da bi skrenuo desno ,desna gusenica treba da miruje dok se leva okrece.Obrnut slucaj je za skretanje ulevo.Brzina motora kontrolise se preko PWM-a odnosno davanjem signala “w” ili “s” za ubrazanje ili usporavanje.Sto se tice same kontrole kamere ona ne zavisi ni od brzine obrtanja motora ni od smera rotacije motora.

```
elif char == ord('t') and motor_is_on == False:

    motor_is_on = True

    stdscr.addstr(0, 8, "  TURNING BATMOBILE ON " , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(1, 8, "  STARTING ENGINE" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(2, 8, "  CALIBRATING CAMERA" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(3, 8, "  CHECKING DRIVERS" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(4, 8, "  UPDATING SOFTWARE" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(5, 8, "  MAKING POPCORN" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(6, 8, "  TURNING ON LIGHTS" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(7, 8, "  TURNING ON MUSIC" , curses.color_pair(2))
    time.sleep(1.4)
    stdscr.refresh()
    stdscr.addstr(8, 8, "  BATMOBILE ON , WELCOME BACK MASTER BRUCE" , curses.color_pair(2))

    pwm_1 = GPIO.PWM(pwm_pin_1, motor_freq)
    pwm_2 = GPIO.PWM(pwm_pin_2, motor_freq)

    pwm_1.start(motor_speed)
    pwm_2.start(motor_speed)

    time.sleep(0.1)
```

Slika 13.Primer koda , start up sekvenca

```
        elif char == curses.KEY_UP:

            GPIO.output(motor1a,GPIO.HIGH)
            GPIO.output(motor1b,GPIO.LOW)
            GPIO.output(motor2a,GPIO.HIGH)
            GPIO.output(motor2b,GPIO.LOW)

            pwm_1.ChangeDutyCycle(motor_speed)
            pwm_2.ChangeDutyCycle(motor_speed)

            stdscr.clear()

            stdscr.addstr(0, 5, " FORWARD ")
            stdscr.addstr(1, 5, "    /\    ")
            stdscr.addstr(2, 5, "   /  \   ")
            stdscr.addstr(3, 5, " /    \  ")
            stdscr.addstr(4, 5, "         ")
            stdscr.addstr(5, 5, "         ")

            time.sleep(0.005)

        elif char == curses.KEY_DOWN:

            GPIO.output(motor1a,GPIO.LOW)
            GPIO.output(motor1b,GPIO.HIGH)
            GPIO.output(motor2a,GPIO.LOW)
            GPIO.output(motor2b,GPIO.HIGH)

            pwm_1.ChangeDutyCycle(motor_speed)
            pwm_2.ChangeDutyCycle(motor_speed)

            stdscr.clear()

            stdscr.addstr(0, 5, "  BACK   ")
            stdscr.addstr(1, 5, " \    /  ")
            stdscr.addstr(2, 5, "  \  /   ")
            stdscr.addstr(3, 5, "   \/    ")
            stdscr.addstr(4, 5, "         ")
            stdscr.addstr(5, 5, "         ")

            time.sleep(0.005)
```

Slika 14.Primer koda , forward & backward logika

```
    elif char == curses.KEY_RIGHT:

        GPIO.output(motor1a,GPIO.HIGH)
        GPIO.output(motor1b,GPIO.LOW)
        GPIO.output(motor2a,GPIO.LOW)
        GPIO.output(motor2b,GPIO.HIGH)

        pwm_1.ChangeDutyCycle(0)
        pwm_2.ChangeDutyCycle(motor_speed)

        stdscr.clear()

        stdscr.addstr(0, 5, "  RIGHT  ")
        stdscr.addstr(1, 5, "   \\    ")
        stdscr.addstr(2, 5, "    \\   ")
        stdscr.addstr(3, 5, "    //   ")
        stdscr.addstr(4, 5, "   //    ")
        stdscr.addstr(5, 5, "         ")

        time.sleep(0.005)

    elif char == curses.KEY_LEFT:

        GPIO.output(motor1a,GPIO.LOW)
        GPIO.output(motor1b,GPIO.HIGH)
        GPIO.output(motor2a,GPIO.HIGH)
        GPIO.output(motor2b,GPIO.LOW)

        pwm_1.ChangeDutyCycle(motor_speed)
        pwm_2.ChangeDutyCycle(0)

        stdscr.clear()

        stdscr.addstr(0, 5, "  LEFT   ")
        stdscr.addstr(1, 5, "  //    ")
        stdscr.addstr(2, 5, " //     ")
        stdscr.addstr(3, 5, " \\     ")
        stdscr.addstr(4, 5, "  \\    ")
        stdscr.addstr(5, 5, "        ")

        time.sleep(0.005)
```

```
    elif char == ord('f'):

        servo_duty_cycle += 0.5
        if servo_duty_cycle >= 12.5:  # Adjust this
            servo_duty_cycle = 12.5

        pwm_servo.ChangeDutyCycle(servo_duty_cycle)

        stdscr.clear()

        stdscr.addstr(0, 5, " CAMERA TURNED RIGHT  ")
        stdscr.addstr(1, 5, "        [       ]---->>    ")
        stdscr.addstr(2, 5, "        [       ]---->>    ")
        stdscr.addstr(3, 5, "           |||             ")
        stdscr.addstr(4, 5, "           |||             ")
        stdscr.addstr(5, 5, "           |||             ")

        time.sleep(0.005)

    elif char == ord('g'):

        servo_duty_cycle -= 0.5

        if servo_duty_cycle <= 2.5:     # Adjust this
            servo_duty_cycle = 2.5

        pwm_servo.ChangeDutyCycle(servo_duty_cycle)

        stdscr.clear()

        stdscr.addstr(0, 5, " CAMERA TURNED LEFT  ")
        stdscr.addstr(1, 5, " <<---- [       ]    ")
        stdscr.addstr(2, 5, " <<---- [       ]    ")
        stdscr.addstr(3, 5, "           |||       ")
        stdscr.addstr(4, 5, "           |||       ")
        stdscr.addstr(5, 5, "           |||       ")

        time.sleep(0.005)
```

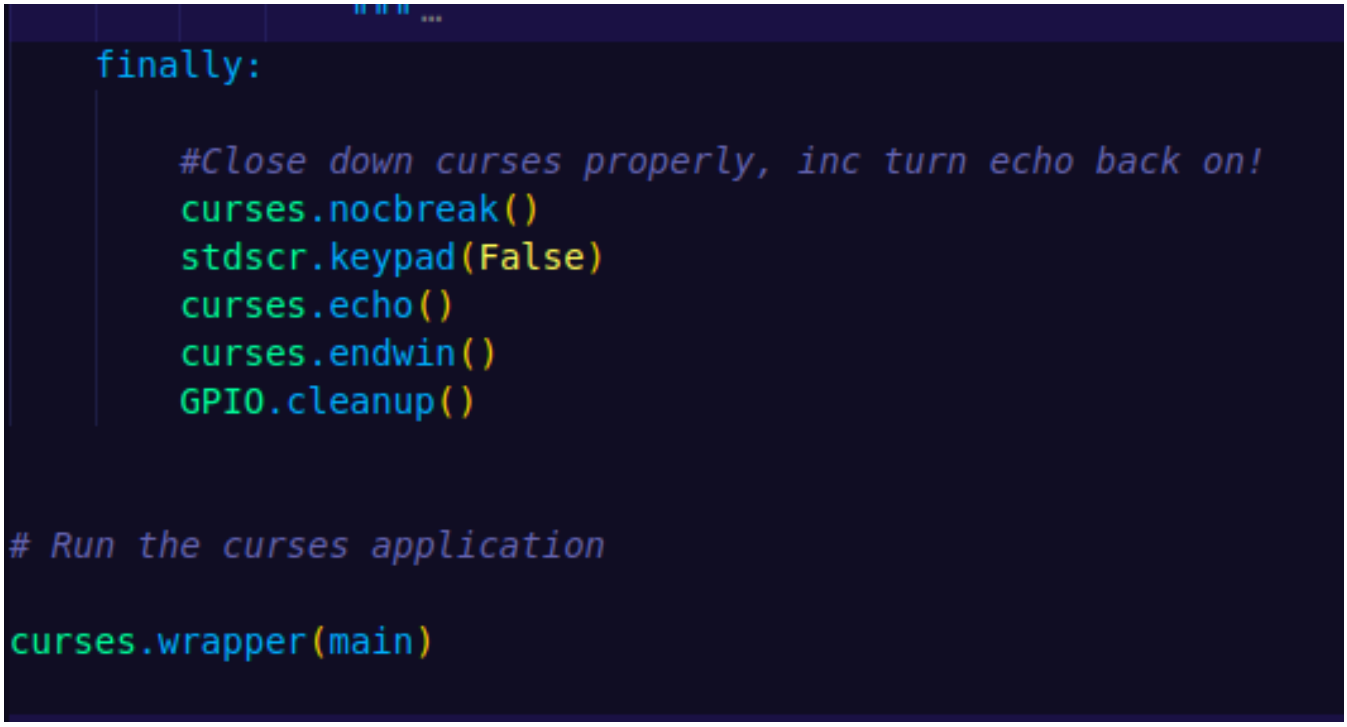
Slika 15.Primer koda , left & right logika

```
elif char == ord('w'):

    if not key_is_pressed:

        key_is_pressed = True
        motor_speed   += 20

        if motor_speed >= upper_speed_limit:
            motor_speed = upper_speed_limit


        pwm_2.ChangeDutyCycle(motor_speed)
        pwm_2.ChangeDutyCycle(motor_speed)

        stdscr.clear()

        stdscr.addstr(0, 5, " SPEEDING UP ")
        stdscr.addstr(1, 5, "         ___  ")
        stdscr.addstr(2, 5, "       /      ")
        stdscr.addstr(3, 5, "      /       ")
        stdscr.addstr(4, 5, "     /        ")
        stdscr.addstr(5, 5, " __/          ")



        stdscr.addstr(8, 0, f"Motor speed: {motor_speed}% ") if motor_speed <= upper_speed_limit
        time.sleep(0.005)
```

Slika 16.Primer koda , speed up logika

```
elif char == ord('s'):


    if not key_is_pressed:

        key_is_pressed = True
        motor_speed   -= 20

        if motor_speed <= lower_speed_limit:
            motor_speed = lower_speed_limit

        pwm_1.ChangeDutyCycle(motor_speed)
        pwm_2.ChangeDutyCycle(motor_speed)

        stdscr.clear()

        stdscr.addstr(0, 5, " SLOWING DOWN ")
        stdscr.addstr(1, 5, " \          ")
        stdscr.addstr(2, 5, "  \         ")
        stdscr.addstr(3, 5, "   \        ")
        stdscr.addstr(4, 5, "    \       ")
        stdscr.addstr(5, 5, "     \_____ ")


        stdscr.addstr(8 , 0 , "LOWER SPEED LIMIT REACHED") if motor_speed <= lower_speed_limit
        time.sleep(0.005)

else:

    key_is_pressed = False
```

Slika 17.Primer koda , slow down logika



Slika 18.Primer koda , final shutdown sekvenca

Nakon sto se program zavrsi potrebno je osloboditi resurse koji su bili napravljeni na pocetku , kada su pinovi inicijalizovani.Sustinski svaki put kad zovemo GPIO.setup(pin ,GPIO.OUT) to znaci da ce pin koji ima neki broj biti zauzet .Kada se program zavrsi neophodno je da se ovaj resurs oslobodi da ne bi doslo da razlicitih konflikata kada se program sledeci put pokrene.Ovo ima I praktican znacaj jer ako se pinovi koji drajvuju motor ne oslobode kada nestane napajanja sa drajvera I ponovo dodje GPIO pinovu sa zapamtili svoje stanje I proseldice datu sekvencu iako to nije ono sto zelimo.Ovo rezultuje tome da algortiam rada nije dobar , odnosno iako nismo pritisnuli “t” motor radi.GPIO.cleanup() ce takodje osloboditi sve gpio pinove koji sluzeza PWM.

5. Zaključak

Na osnovu testiranja laboratorijskim uslovima moze se zaključiti da projekat ispunjava sve trazene funckionalnosti koje su definisane .

Korisnik koristi tastaturu kao input method I svi dugmici rade I nema efekta “ghostinga” I software filter je implementiran .Ekran se redovno ispisuje I nema efekta zakasnelih karaktera.Takodje implementirano je ponasanje u slucaju kada dodje do nepredvidjene greska da program ne bi pukao I GPIO pinovi ostali zauzeti , sto moze da pravi problem kada se program ponovo pokrene.

Kamera moze da radi I nezavisno , ako postoji potreba za tim.Takodje u prilogu je dat jednostavan python kod koji koristi stream sa kamere uz pomoc openCV-ja I pritiskom “space”-a zabelezi sliku , pa ovo otvara mogucnoti za nove primene.

Takodje stream sa kamere se moze gledati na bilo kom uredjaju , posto se koristo jedan od standardnih portova.

5.1 Moguća implementacija izostavljene funkcionalnosti

Jedan od nedostataka projekta je sto nema napajanje preko baterija , jer je autor nekako zagubio 18650 Li-Ion bateriju I battery holder za nju u vidu.Ovaj problem bi se mogao koristiti uz pomoc buck-boost naponskog regulatora TP4056 I standarnih AA ili AAA baterija.

Moguce implementacije bi bile koriscenje dodatnih senzora za bolju orijentaciju u prostoru I automatsko zaustavljanje (“smart brake”) , kada se tenkic nalazi na distanci blizu prepreke.Neki od primera bi bili VL53L0X tof senzor ili HC-SR04 ultrasonic senzor.Takodje bi mogao da se implementira aktivni ili pasivni gimbal za odrzavanje orijentacije kamere.

Naravno bilo bi dobro imati pristup 3D stampacu za pravljenje dodatnih delova I kucista , posto projekati trenutnom stanju izgleda dosta nakaradno.

Funkcija sistema koja je izostavljena je autonomno upravljanje , uz pomoc masinskog ucenja ili machine vision-a, posto autor nije imao dovoljno jak hardwer u trenutku implementacije sistema (Rasp pi 4 je dosao tek kasnije) I masinsko ucenje implementirano na procesoru je neprakticno I predugo bi trajalo.Ako bi se pak autonomno upravljanje koristilo uz pomoc senzora to bi bilo moguce , ali opet uz dosta

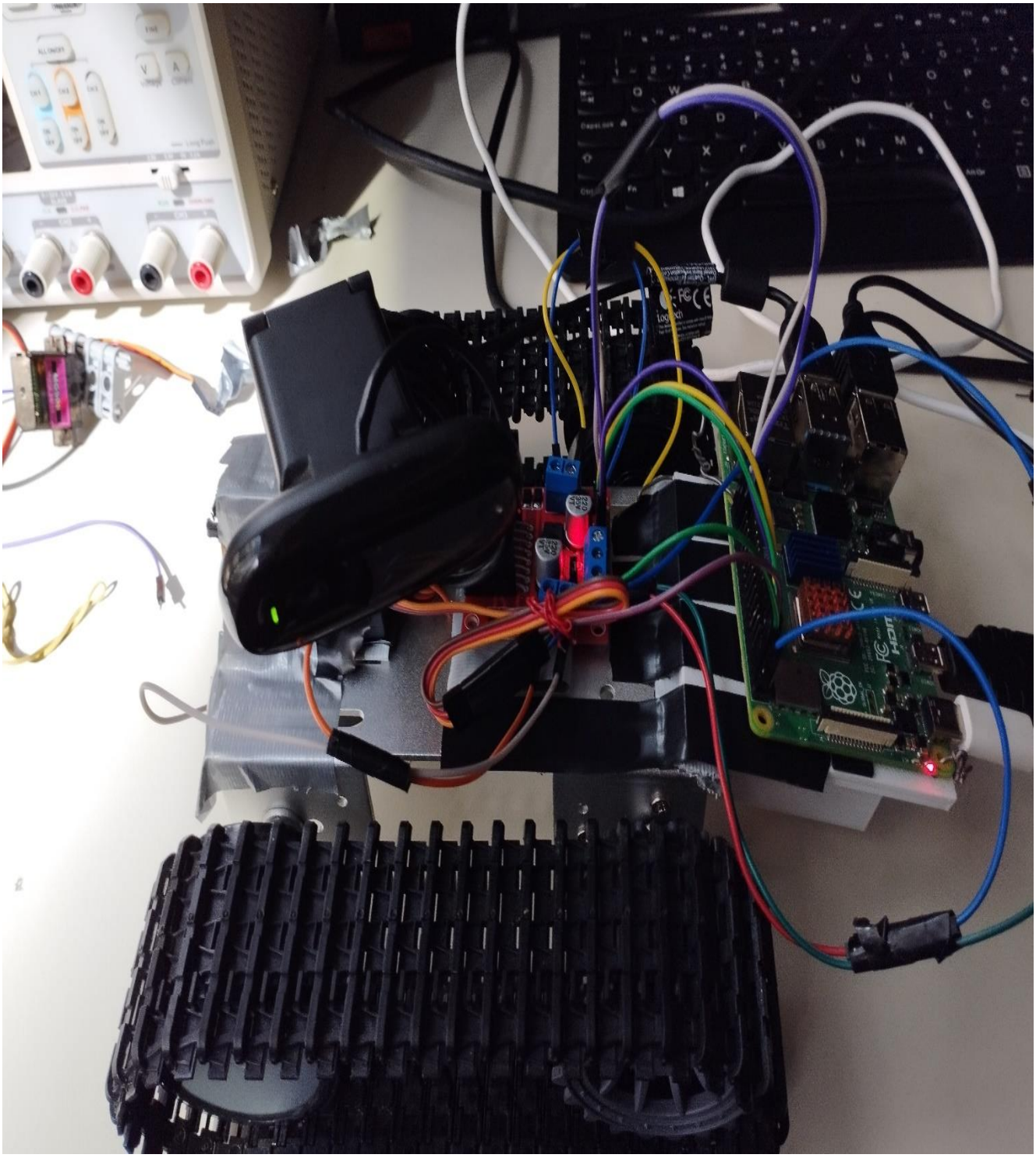
mentalnog dovijanja I uz pomoc kocepata kao sto su sensors fusion I Kalman (KF) filter odnosno EKF.Ako bi se koristili senzori , sledecih par bi bilo dovoljno :

Senzor	Komada
MPU9250	x1
HC-SR04	x1
VL53L1X	x2
Adafruit Ultimate GPS Breakout	x1

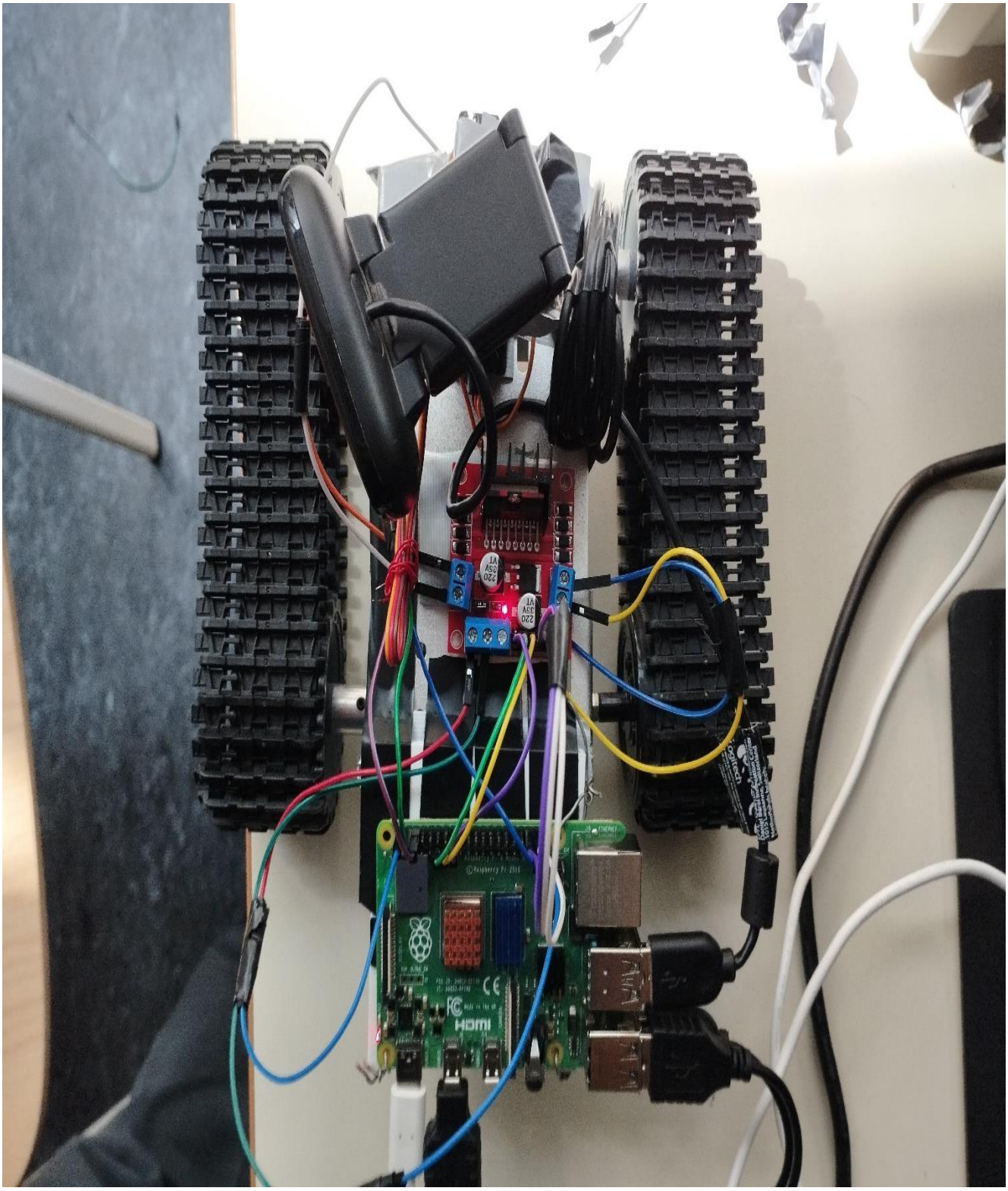
Tabela 3.Spisak potrebnih senzora za autonomnu voznju

Posto postoji dosta senzora koji rade nezavisno , sigurnost u izmerenu poziciju ce biti veca , takodje sistem ce biti mnogo vise otporan na greske I radice u razlicitim uslovima (uslovi manje vidljivosti nisu problem za ultrasonicni senzor , dok bucna sredina ne predstavlja problem za ToF senzor).

5.2 Fizički izgled obe ploče u krajnjem stadijumu izrade



Slika19..Fizicki izgleda tenka sa strane



Slika20.Fizicki izgleda tenka odgore

6. Uputstvo za upotrebu

Dovesti napon napajanja na tenkic (bilo u vidu usb kabla ili preko baterija).

Ulogovati se na uredjaj pomocu SSH -a , VNC-a ili direktno , uci u direktorijum uz pomoc komande

cd /*primenjena*
python3 new_car.py

Nakon sto se ove komande izvrse terminal treba da pocrni I sada je moguce upravljanje.

Ako postoji zelja da se koristi I usb camera treba pokrenuti scriptu pod nazivom

...ovde stavi ime skripte

Potom otvoriti web browser po zelji I uneti sledecu komandu:

https://<ip address of the device >>:9000

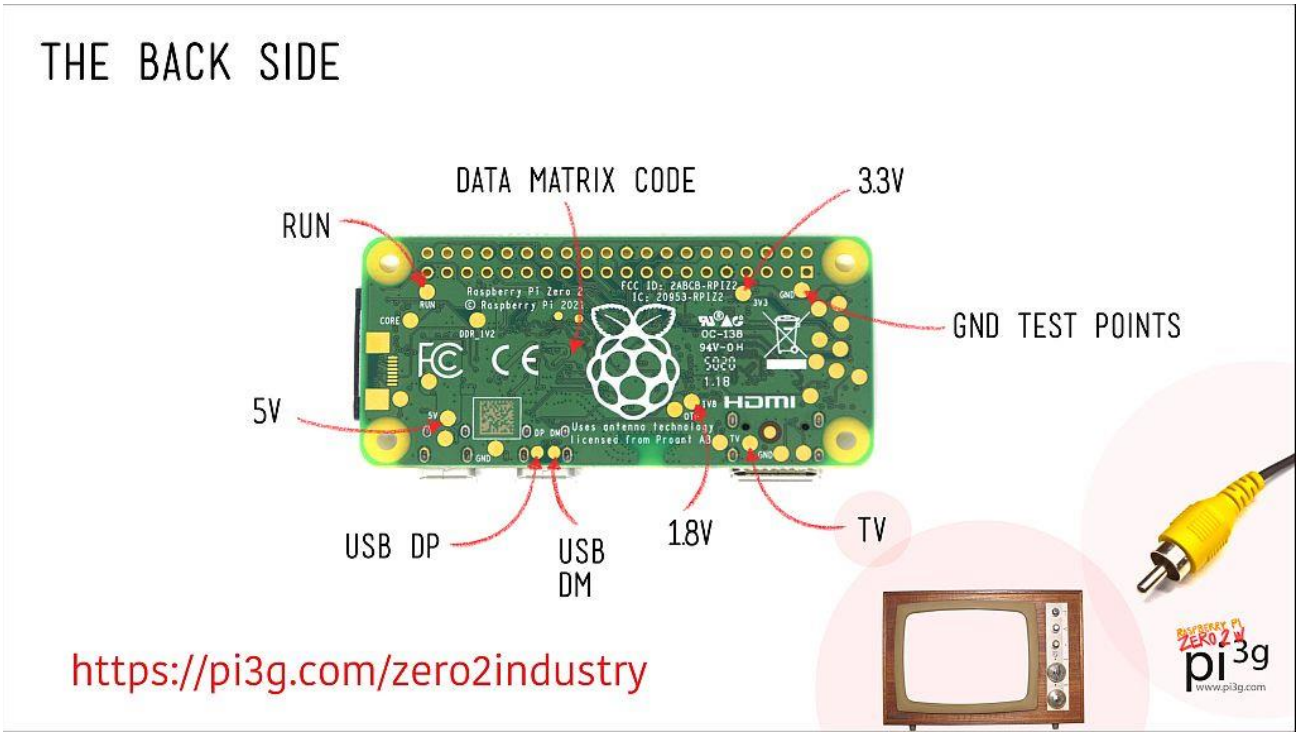
Odabrati stream opciju I podesiti kameru po zelji , ako postoji potreba za tim.

Korišćenje uređaja se svodi na pritiskanje tastera na tastaturu slicno kao u bilo kojoj igrici u kojoj se voze kola ili bilo koje drugo prevozno sredstvo.Postoji jos par dodatnih tastera koji sluze za paljenje tenka (dugme na tastaturi “t”) I rotiranje kamere (“f” & “g”). Registrovanje signala sa svakog tastera rezultuje u različitom ponašanju . Izlazak je uz pomoc dugmeta “q”.

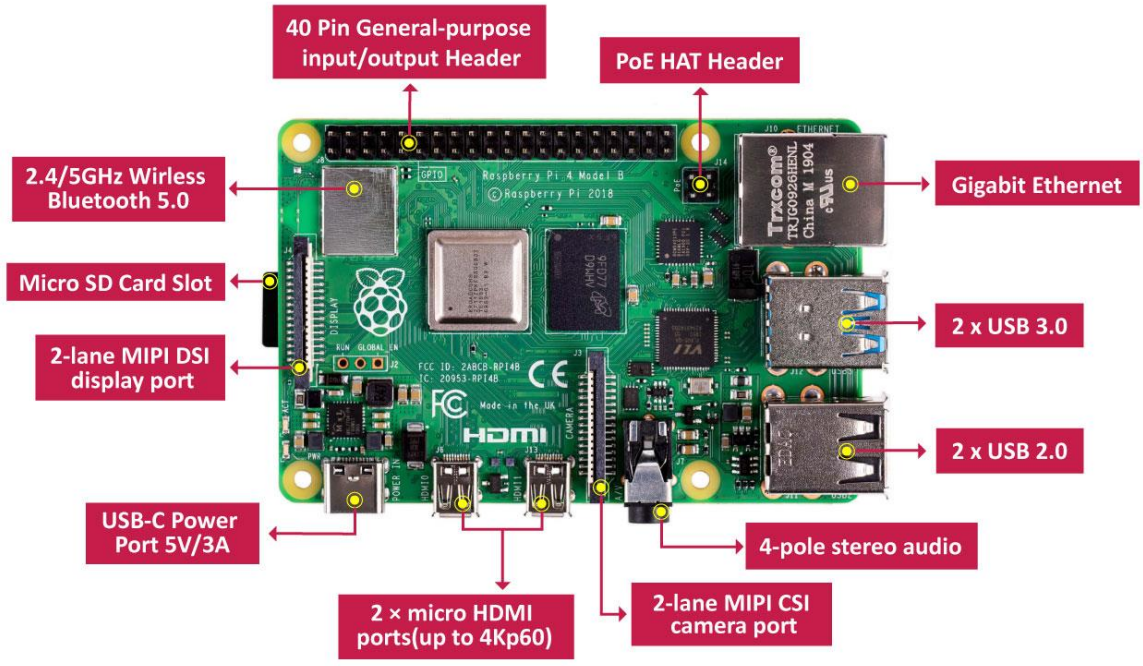
7. Literatura

- https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf
- https://www.adafruit.com/product/746
- https://www.sparkfun.com/products/15569
- https://www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html#featured_resources-0
- https://www.adafruit.com/product/2717
- https://www.etechnophiles.com/l298n-motor-driver-pin-diagram/
- https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf
- https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf
- https://www.raspberrypi.com/documentation/computers/raspberry-pi.html
- https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf
- https://www.towerpro.com.tw/product/mg996r/

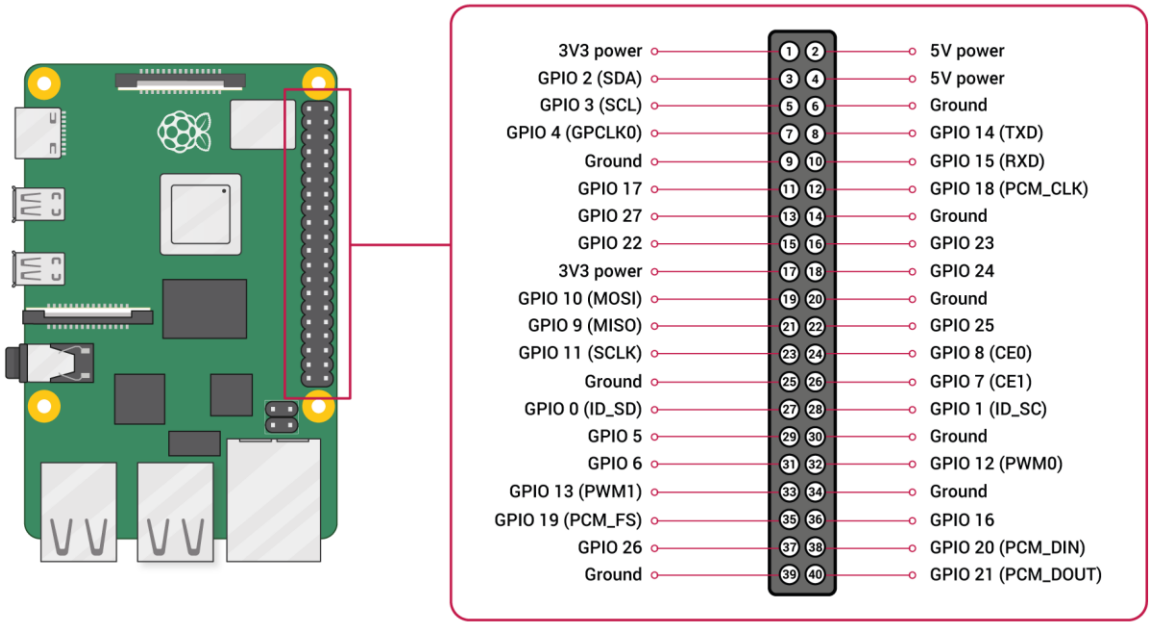
Dodatak A



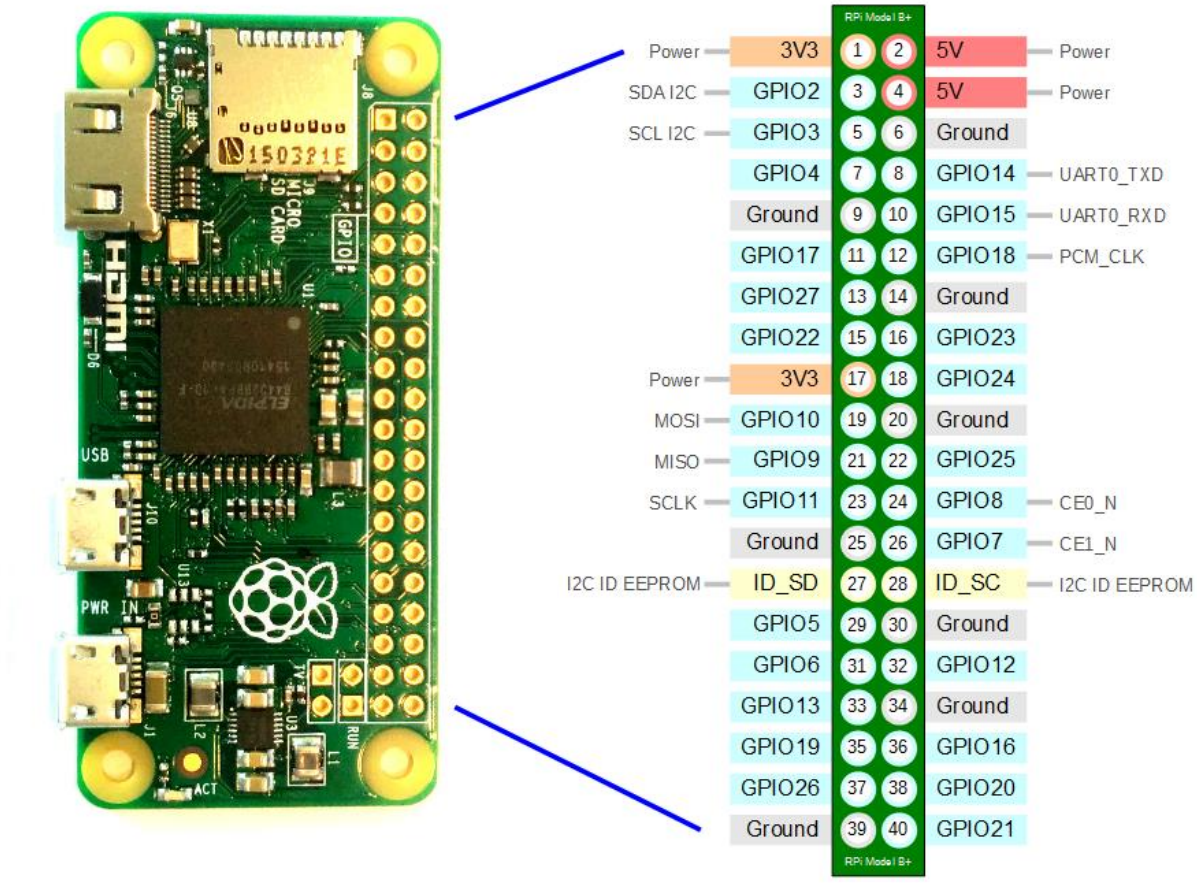
Slika 21 .Raspberry pi Zero W modul , koriscen u pocetku



Slika 22 .Raspberry Pi model B



Slika23.Raspberry Pi model B pinout



Slika 24 .Raspberry Pi Zero W pinout

Dodatak B

Neke od korisnih komandi na linuxu :

cd /<dir> #ulazak u direktorijum pod imenom <dir>
vcgencmd measure_temp # izmeri CPU temperaturu
vcgencmd get_mem arm # prikazi ARM memory
free -h #prikazi koriscene resurse
rm -r <dir> #rekurzivno brisanje celoh direktorijma <dir>
ifconfig #prikaz IP4 I IP6 adresa
hciconfig #prikaz bluetooth adresa
mv <source > <dest> #copy paste komanda
sudo apt-get update && sudo apt-get upgrade -y #update I upgrade sistema
sudo reboot now #restartuj
sudo shutdown -h now #gasi rasp pi
ssh pi@<ip_address> #pokreni ssh protokol I uloguj se
ssh -X pi@<ip_address> #pokreni X sesiju , graficki interface
mkdir <dir_name> #napravi direktorijum pod imenom <dir_name>
python3 <name_of_the_script>.py #pokreni python3 scriptu pod imenom <name_of_the_script>.py
pip3 install <package_name> #instaliraj modul pod imenon <package_name>