# Razvoj programske podrške za upravljanje Raspberry Pi mobilnom robotskom platformom s Android uređaja

**Dumančić, David**

**Undergraduate thesis / Završni rad**

**2020**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:220682

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-25**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni studij**

# RAZVOJ PROGRAMSKE PODRŠKE ZA UPRAVLJANJE RASPBERRY PI MOBILNOM ROBOTSKOM PLATFORMOM S ANDROID UREĐAJA

**Završni rad**

**David Dumančić**

**Osijek,2020.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju**

**Osijek, 25.09.2020.**

**Odboru za završne i diplomske ispite**

# Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

| | |
|---|---|
| **Ime i prezime studenta:** | David Dumančić |
| **Studij, smjer:** | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3910, 24.09.2019. |
| **OIB studenta:** | 49516408454 |
| **Mentor:** | Izv. prof. dr. sc. Damir Blažević |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | |
| **Naslov završnog rada:** | Razvoj programske podrške za upravljanje Raspberry Pi mobilnom robotskom platformom s Android uređaja |
| **Znanstvena grana rada:** | **Procesno računarstvo (zn. polje računarstvo)** |
| **Predložena ocjena završnog rada:** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 2 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 25.09.2020. |
| **Datum potvrde ocjene Odbora:** | 30.09.2020. |

| | |
|---|---|
| Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija: | Potpis: |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 09.10.2020.**

| | |
|---|---|
| **Ime i prezime studenta:** | David Dumančić |
| **Studij**: | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3910, 24.09.2019. |
| **Turnitin podudaranje [%]:** | 11 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj programske podrške za upravljanje Raspberry Pi mobilnom robotskom platformom s Android uređaja**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Damir Blažević

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

1.UVOD ..... 1
1.1 Zadatak Završnog rada ..... 1
2. MOBILNA PLATFORMA ..... 2
2.1 Raspberry Pi Zero W ..... 3
2.2 Realizacija mobilne platforme ..... 5
2.3 Raspberry Pi operacijski sustav ..... 8
2.4 Python ..... 8
2.5 Definiranje kontrole motora ..... 8
2.6 Primanje naredbi s druge adrese ..... 10
3. PRAKTIČNI ZADATAK ..... 13
3.1 Android Studio ..... 13
3.2 Programski jezik Java ..... 15
3.3 XML ..... 15
3.4 Activity ..... 17
3.5 Fragment ..... 20
3.6 Slanje naredbi ..... 24
4. ZAKLJUČAK ..... 27
LITERATURA ..... 28
SAŽETAK ..... 29
ABSTRACT ..... 30
ŽIVOTOPIS ..... 31
PRILOZI ..... 32

# 1.UVOD

Jednu od najvećih prednosti digitalnog doba predstavlja bežično upravljanje. Posljednjih godina je to još podignuto na višu razinu jer osim što je bežično, moguće je kontrolirati raznim uređajima uz pomoć mobilnog uređaja. To pridonosi sigurnijoj radnoj okolini, manje fizičkog rada i jednostavnije korištenje što pridonosi većoj produktivnosti.

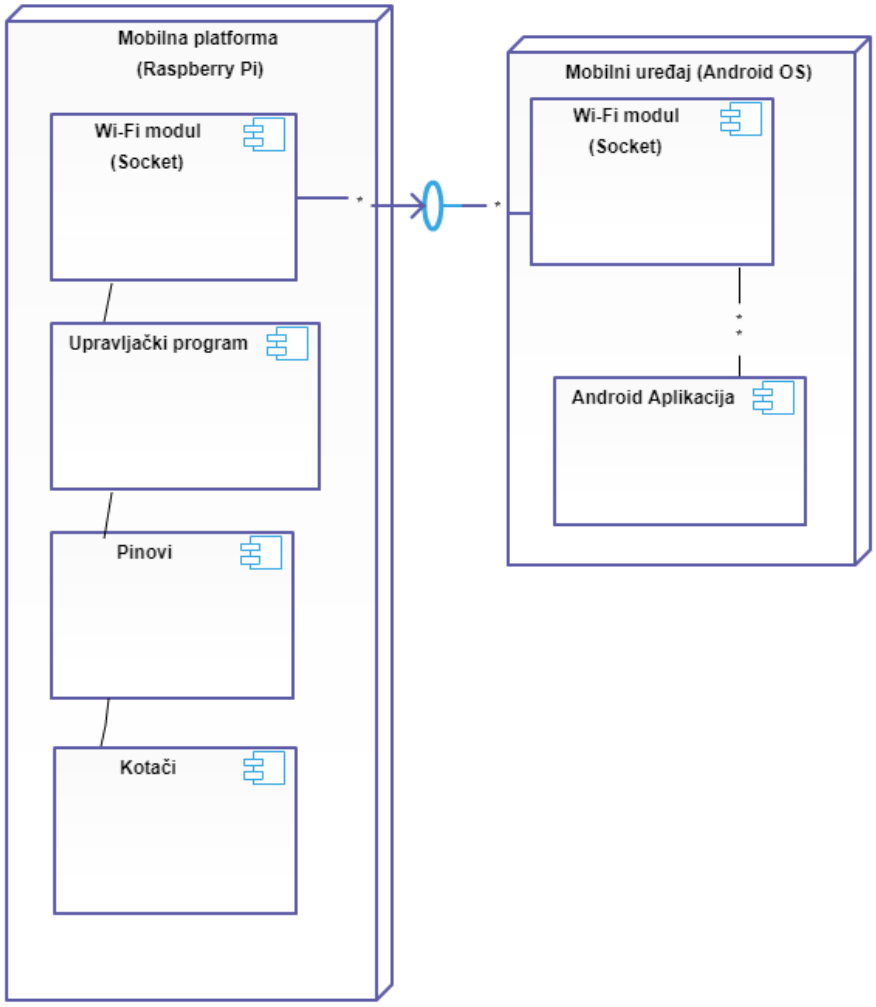
U ovom radu opisana je izrada robotske mobilne platforme i izrada aplikacije koja omogućuje bežično upravljanje navedenom platformom. Izrada je prikazana u četiri poglavlja. Prvo poglavlje opisuje temu koja je obrađena u ovom radu. U poglavlju dva opisana je izrada mobilne platforme zasnovanu na radu Raspberry Pi [1] računala uz dodatak drugih komponenti te njihovu ulogu. U trećem poglavlju prikazana je izrada aplikacije za android uređaje uz objašnjenje korištenih tehnologija, pokrijepljenih programskim kodom, korištenih za realizaciju iste. U četvrtom poglavlju je iznesen zaključak.

## 1.1 Zadatak Završnog rada

U ovom radu opisano je korištenje Raspberry Pi računala za definiranje osnovnih ponašanja pokretne platforme, dizajniranje i izrada Android aplikacije za upravljanje mobilne platforme uz korištenje WiFi komunikacije za razmjenu informacija.

# 2. MOBILNA PLATFORMA

Robotska mobilna platforma treba osigurati vlastito kretanje u smjeru i trajanju prema odabiru korisnika s udaljenog položaja. Android aplikacija treba detektirati odabir korisnika i prenijeti naredbe do mobilne platforme. Ukoliko su zadovoljeni ti uvjeti, radi se o bežičnom upravljanju u stvarnom vremenu. Dijagram koji prikazuje pojednostavljene odnose aplikacije i mobilne platforme prikazan je na slici 2.1.



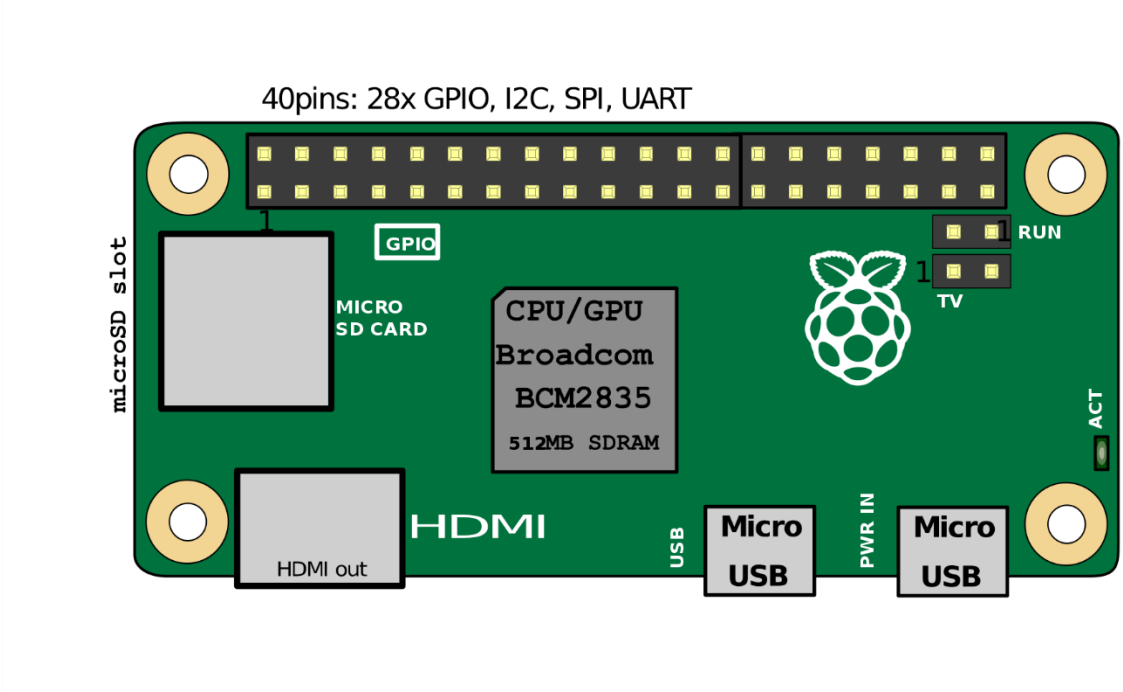

Slika 2.1 Odnos aplikacije i mobilne platforme

Mobilna platforma zasnovana je na radu Raspberry Pi računala. Raspberry Pi je serija malih jednopločnih računala (engl. *Single Board Computer*) te predstavlja potpuno funkcionalno

računalo na jednoj tiskanoj ploči. Razvijen je za potrebe edukacije i razvoja računarstva. Često je korišten u istraživačkim projektima zbog svoje niske cijene i lake prenosivosti.

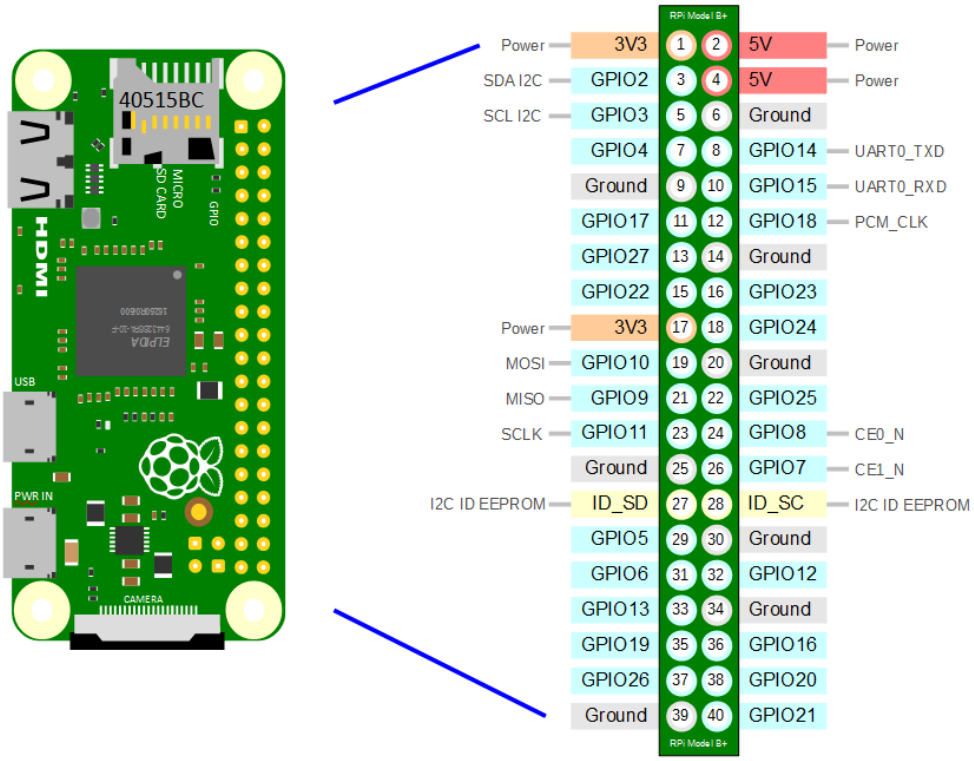
## 2.1 Raspberry Pi Zero W

Raspberry Pi Zero W [2] je najmanje računalo iz serije Raspberry Pi. Sadrži jednojezgreni procesor frekvencije 1 GHz , 512 MB RAM, mini HDMI port, micro USB priključak za napajanje, micro USB OTG port za priključke. Izgled Raspberry Pi Zero W računala prikazan je na slici 2.2.




Slika 2.2 Raspberry Pi Zero W

Osim navedenog Raspberry Pi Zero W sadrži 40 GPIO (engl. *General purpose input*/output) pinova (Slika 2.3). Dva pina od 5V i dva pina od 3.3V se nalaze na ploči uz 8 pinova koji služe za uzemljenje (0V), navedene pinove nije moguće definirati kao ulaz ili izlaz. Ostalim pinovima je moguće manipulirati kako bismo dobili željeni rezultat. Pinovi definirani kao izlaz mogu biti postavljeni u logičku jedinicu (3.3V) ili logičku nulu (0V). Isto tako pinovi definirani kao ulazni mogu biti očitani kao logička jedinica (3.3V) ili logička nula (0V) .



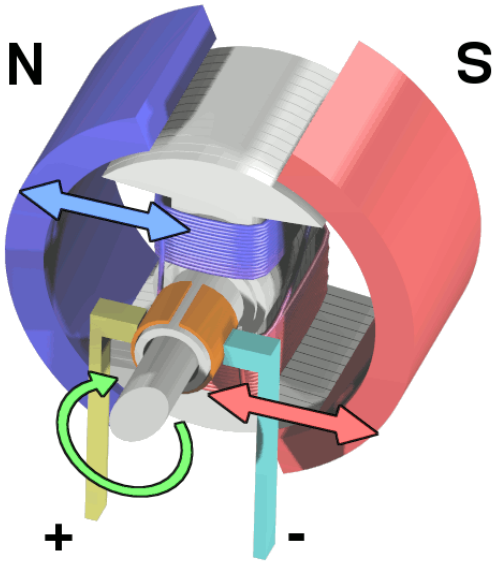

Slika 2.3 Raspored pinova

## 2.2 Realizacija mobilne platforme

Pokretna mobilna platforma osim Raspberry pi računala sadrži:

### 2.2.1. Istosmjerni motor

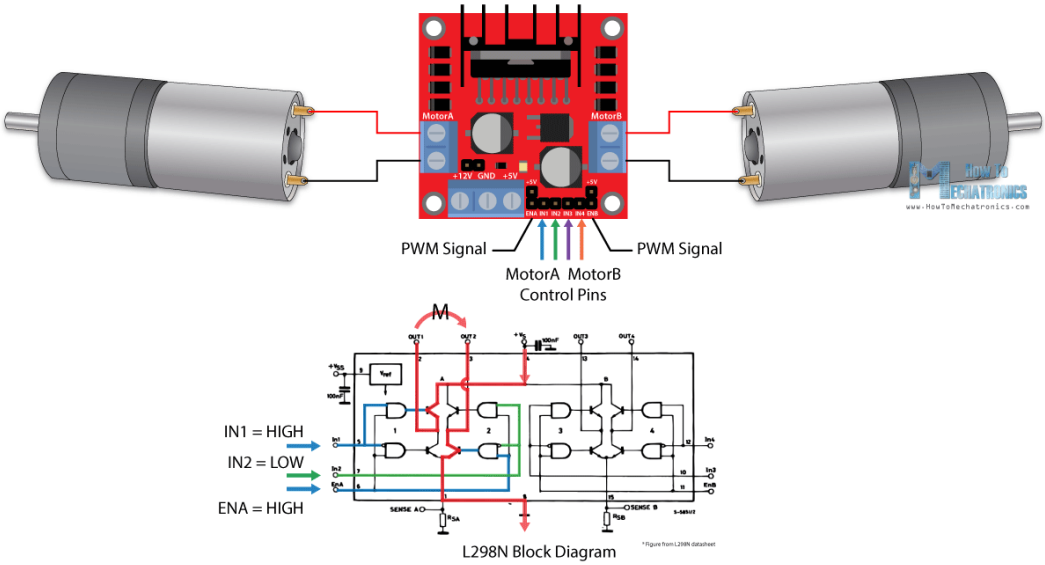
Istosmjerni motor [3] je elektromehanički uređaj koji električnu energiju pretvara u mehaničku odnosno pretvara istosmjernu struju u kružno gibanje. Glavni dijelovi motora su rotor i stator. Osnovno načelo koje koriste svi motori je da na vodič kroz koji teče struja u magnetskom polju djeluje sila. Kod istosmjernog motora magneti na statoru stvaraju magnetsko polje, a kroz namotaje rotora se propušta električna struja te na namotaje djeluje sila koja zakreće rotor. Za potrebe ove mobilne platforme koriste se dva istosmjerna motora. Prikaz istosmjernog motora prikazan je na slici 2.4.




Slika 2.4 Prikaz istosmjernog motora

### 2.2.2. L298N sklop za kontrolu motora

L298N [4] je sklop koji omogućuje zamjenu polariteta na svojim izlazima pa tako omogućuje upravljanje vrtnje motora u oba smjera. Kao ulaze prihvaća signale koje šalje Raspberry Pi računalo preko svojih pinova te na osnovu njih određuje polaritet na izlazu a time i smjer vrtnje motora. Prikaz L298N sklopa i njegov blok dijagram prikazan je na slici 2.5.




Slika 2.5.Spoj L298N sklopa s istosmjernim motorima i blok dijagram

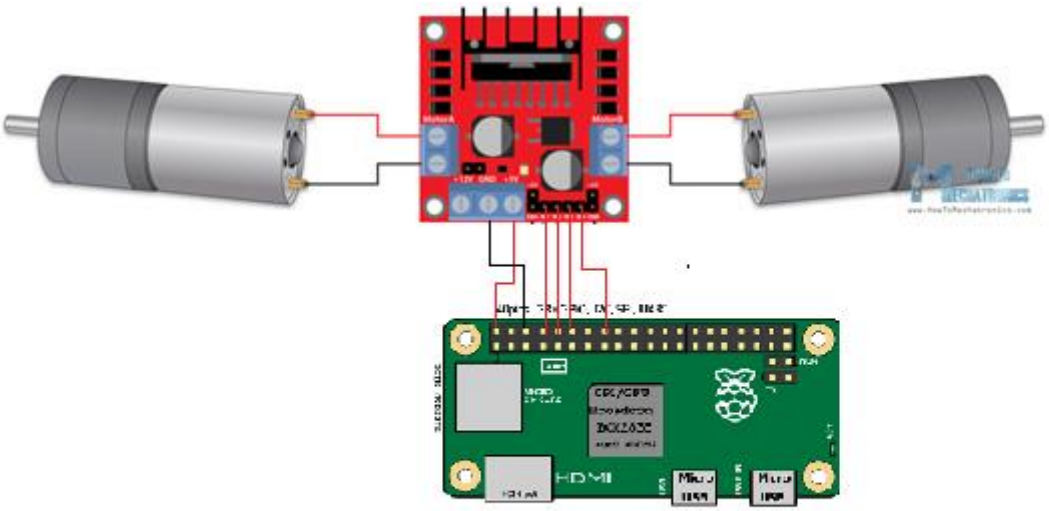
### 2.2.3. Raspberry Pi Kamera v2 modul

Senzor rezolucije 8 megapiksela koji se može koristiti za prijenos videa visoke rezolucije i fotografije.

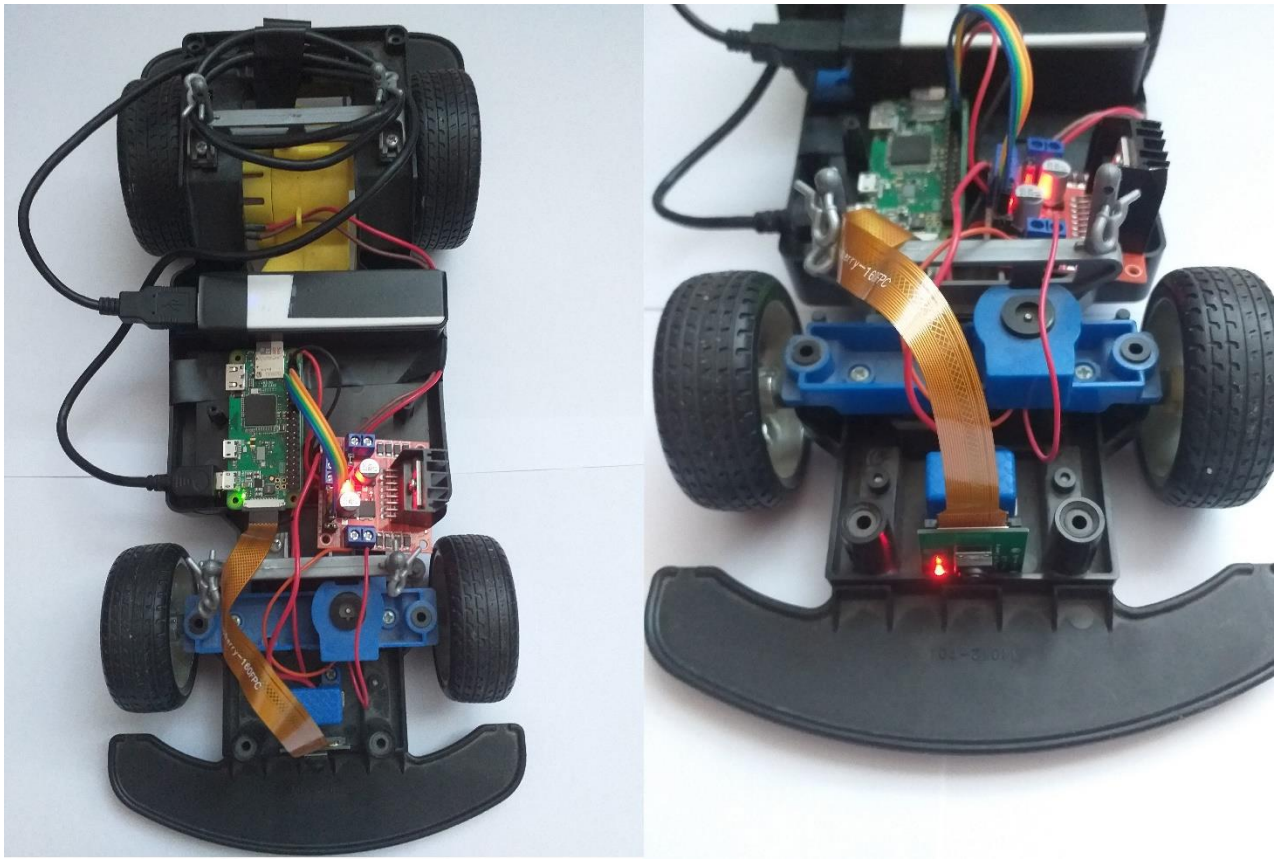
### 2.2.4. Baterija za napajanje

Za napajanje motora korišteno je 6 baterija AA 1.5V, dok je za napajanje Raspberry Pi uređaja korištena prijenosna punjiva baterija. Prema slici 2.5 vidimo kako su pinovi 8,10,12,16 Raspberry Pi računala spojeni na kontrolne pinove motora na L298N sklopu, na taj način je moguće upravljati motorima uz pomoć računala koje šalje 4 signala, sklop L298N zatim obrađuje dobivene signale i na temelju njih određuje koji motor radi i u kojem smjeru. Pinovi 2 i 6 su iskorišteni za napajanje sklopa.

Shema spajanja Raspberry Pi računala s L298 sklopom prikazana je na slici 2.6. Realizacija cijele mobilne platforme s navedenim dijelovima prikazana je na slici 2.7.



Slika 2.6 Shema spajanja Raspberry Pi računala sa sklopom L298N



Slika 2.7 Pokretna mobilna platforma

## 2.3 Raspberry Pi operacijski sustav

Raspberry Pi OS [6] (ranije poznat kao Raspbian) je operacijski sustav kreiran 2012. godine. Zasnovan je na Debianu i optimiziran za Raspberry Pi sklopovlje. Raspberry Pi OS je projekt zajednice pod aktivnim i stalnim razvojem s naglaskom na stabilnost i performanse.
Za pokretanje sustava potrebno je priključiti memorijsku karticu s raspakiranim operacijskim sustavom. Raspberry Pi OS sadrži modificirano LXDE grafičko sučelje. Upravljanje računalom može se postići na više načina. Raspberry Pi sadrži HDMI izlaz kako bi sučelje bilo prikazano na vanjskom monitoru, za ulazne uređaje poput miša i tastature sadrži micro USB priključak. Osim navedenog, Raspberry Pi OS-u se može pristupiti i preko daljinskog upravljanja uz aplikaciju koju nudi Windows (*engl. Remote Desktop Connection*) koja prikazuje grafičko sučelje ili aplikaciju PuTTY – besplatni emulator terminala, serijska konzola i aplikacija za prijenos podataka.

## 2.4 Python

Python je programski jezik visoke razine, interpretiran, opće namjene. Prva javna inačica je objavljena 1990. godine. Koristi automatsku memorijsku alokaciju i dopušta programerima korištenje više stilova programiranja. Kompatibilan je s Raspberry Pi OS sustavom što je iskorišteno u ovom slučaju kako bi prethodno spojena mobilna platforma dobila očekivanu funkcionalnost kretanja i upravljanja istom. Python je korišten za razvoj programskog rješenja u tri razine: kontrola motora, primanje naredbi s android uređaja i video prijenos slike s kamere na android uređaj. Kontrola motora predstavlja osnovnu razinu u kojoj se, manipuliranjem izlaznim signalima Raspberry Pi računala pomoću Python programa, aktiviraju motori koji se koriste u raznim kombinacijama za ostvarivanje željenih kretnji platforme. Kretnje se mogu iskoristiti na više načina, u ovom slučaju poželjno je da korisnik odlučuje o smjeru i trajanju kretnji. Za to je zadužena druga razina programa. Primanje informacije i odlučivanje koju funkciju pozvati s osnovne razine. Video prijenos slike s kamere odvija se u drugom smjeru, u tom slučaju Raspberry Pi računalo emitira video uživo kojeg je moguće dohvatiti s drugog uređaja. Za njihov rad korištene su biblioteke RPi.GPIO, picamera, socket, server.

## 2.5 Definiranje kontrole motora

Za uspješnu kontrolu motora potreban je uvoz biblioteke RPi.GPIO. Navedeni paket pruža klasi kontrolu GPIO [7] pinova opisanih u poglavlju 2.1. Kako su pinovi 8,12,10,16 spojeni na kontrolne pinove motora na L298N sklopu (Slika 2.5) ,potrebno je iste definirati u

programu kao izlazne vrijednosti kako bi unutar programa korisnik svojevoljno mogao mijenjati njihove vrijednosti (logička nula ili logička jedinica). Definiramo ih pomoću funkcije GPIO.setup() koja za argumente prihvaća kanal (broj pina) i oznaku za izlaz ili ulaz, ovisno o potrebi. Postoji više vrsta definiranja kanala, kako bi se definirale oznake pinova identične brojevima na ploči koristimo funkciju GPIO.setmode(GPIO.BOARD). Definiranje izlaznih pinova prikazano je na slici 2.8

```
import RPi.GPIO as GPIO

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(8,GPIO.OUT)
    GPIO.setup(12,GPIO.OUT)
    GPIO.setup(10,GPIO.OUT)
    GPIO.setup(16,GPIO.OUT)
```

Slika 2.8 Definiranje izlaznih pinova

Nakon što su definirane izlazne vrijednosti moguće je definirati funkcije koje predstavljaju četiri različite aktivacije motora (kretanje naprijed, kretanje nazad, skretanje u lijevo, skretanje u desno) i četiri deaktivacije motora za navedene slučajeve uz jednu funkciju za isključivanje svih motora. Funkcije su prikazane na slici 2.9.

```
def MotorUp():
    print ('Motor UP')
    GPIO.output(10,True)
def MotorDown():
    print ('Motor DOWN')
    GPIO.output(8,True)
def MotorLeft():
    print ('Motor LEFT')
    GPIO.output(12,True)
def MotorRight():
    print ('Motor RIGHT')
    GPIO.output(16,True)
def MotorUpStop():
    print ('Motor UP Stop')
    GPIO.output(10,False)
def MotorDownStop():
    print ('Motor DOWN Stop')
    GPIO.output(8,False)
def MotorLeftStop():
    print ('Motor LEFT Stop')
    GPIO.output(12,False)
def MotorRightStop():
    print ('Motor RIGHT Stop')
    GPIO.output(16,False)
def MotorStop():
    print ('Motor STOP')
    GPIO.output(8,False)
    GPIO.output(10,False)
    GPIO.output(12,False)
    GPIO.output(16,False)
```

Slika 2.9 Implementacija funkcija za aktivaciju i deaktivaciju motora

Svaka funkcija postavlja određeni pin u jedno od dva moguća stanja (logička jedinica ili logička nula), to stanje se dalje prenosi do sklopa L298N koji na osnovu toga aktivira motor ukoliko je dobio logičku jedinicu ili gasi određeni motor ukoliko je dobio logičku nulu.

## 2.6 Primanje naredbi s druge adrese

Za ostvarivanje komunikacije između programa za upravljanje motora i Android aplikacije koristimo sučelje *Socket* [8]. *Socket* predstavlja način povezivanja dva čvora u mreži kako bi komunicirali jedan s drugim. Jedan *socket* sluša port na određenoj IP adresi, dok drugi ostvaruje tu vezu. Možemo ih nazvati server i klijent. U ovom slučaju koristimo Stream *Socket* koji ostvaruje vezu pomoću TCP protokola (*engl. Transmission Control Protocol)*, koji predstavlja vezu jedan-na-jedan. Potrebno je uvesti biblioteku *Socket* koja sadrži potrebne metode za rad i MotorControl.py koji je opisan u poglavlju 2.5. Polje stringova ctrCmd predstavlja znak za svako ponašanje, gdje velika slova predstavljaju aktivaciju motora za naprijed, nazad, lijevo, desno (W, S, A, D) a mala slova za deaktivacju istih (w, s ,a, d). Koriste

se znakovi jer u slučaju većih stringova treba više vremena za slanje i njihovo čitanje. HOST string je prazan kako bi mogli primati podatke s bilo koje adrese. Potrebno je odrediti port za spajanje i definirati *socket*. *Socket* instanca prima dva parametra : AF_INET označava korištenje IP protokol verzije 4 (IPv4) i SOCK_STREAM označava vezu uz pomoć TCP protokola. Implementacija *socket* tehnologije prikazana je na slici 2.10.

```
import MotorControl
from socket import *
from time import ctime
import time
import RPi.GPIO as GPIO


ctrCmd = ['W','S','A','D','w','s','a','d','0']

HOST = ''
PORT = 21567
BUFSIZE = 1024
ADDR = (HOST,PORT)

tcpSerSock = socket(AF_INET, SOCK_STREAM)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(5)
```

Slika 2.10 Implementacija tehnologije *socket*

*Socket* pomoću metode listen() sluša sve podatke koji dolaze na njegovu adresu. U beskonačnoj petlji ih prihvaća metodom accept() i preuzima podatke metodom recv(), sprema ih u string data i uspoređuje s poljem stringova ctrCmd. Ukoliko je preneseni string jedan od navedenih znakova, program poziva potrebnu funkciju iz programa MotorControl i pokreće ili zaustavlja motor. Metode za upravljanje motorima prikazane su na slici 2.11.

```
while True:
        print ('Waiting for connection')
        tcpCliSock,addr = tcpSerSock.accept()
        print ('...connected from :', addr)


        try:
                while True:
                        data =''
                        data1=data
                        data = tcpCliSock.recv(BUFSIZE).decode('utf-8')
                        print (data)
                        if data == ctrCmd[0]:
                            MotorControl.MotorUp()
                        elif data == ctrCmd[1]:
                            MotorControl.MotorDown()
                        elif data == ctrCmd[2]:
                            MotorControl.MotorLeft()
                        elif data == ctrCmd[3]:
                            MotorControl.MotorRight()
                        elif data == ctrCmd[4]:
                            MotorControl.MotorUpStop()
                        elif data == ctrCmd[5]:
                            MotorControl.MotorDownStop()
                        elif data == ctrCmd[6]:
                            MotorControl.MotorLeftStop()
                        elif data == ctrCmd[7]:
                            MotorControl.MotorRightStop()
                        elif data == ctrCmd[8]:
                            MotorControl.MotorStop()
                        elif not data1:
                                break
        except KeyboardInterrupt:
                GPIO.cleanup()
                MotorControl.close()
tcpSerSock.close();
```

Slika 2.11 Pozivanje metoda za upravljanje

# 3. IZRADA ANDROID APLIKACIJE

Za izradu praktičnog zadatka potrebno je izraditi i dizajnirati Android aplikaciju za upravljanje mobilnim robotom temeljenim na Raspberry Pi platformi koja je opisana u drugom poglavlju. Upravljanje je je zasnovano na razmjeni informacija uz korištenje WiFi komunikacije. Mobilna aplikacija napravljena je u razvojnom okruženju Android Studio. Za razvoj programskog rješenja aplikacije korišten je programski jezik Java, dok je za opis dizajna aplikacije korišten XML jezik. U ovom poglavlju će biti opisan i prikazan postupak izrade aplikacije uz detaljnije objašnjena implementacija korištenih tehnologija za rad same aplikacije (Aktivnost, Fragment) i ostalih tehnologija za upravljanje mobilnom platformom (*Socket*, korištenje tipki i senzora Android uređaja).

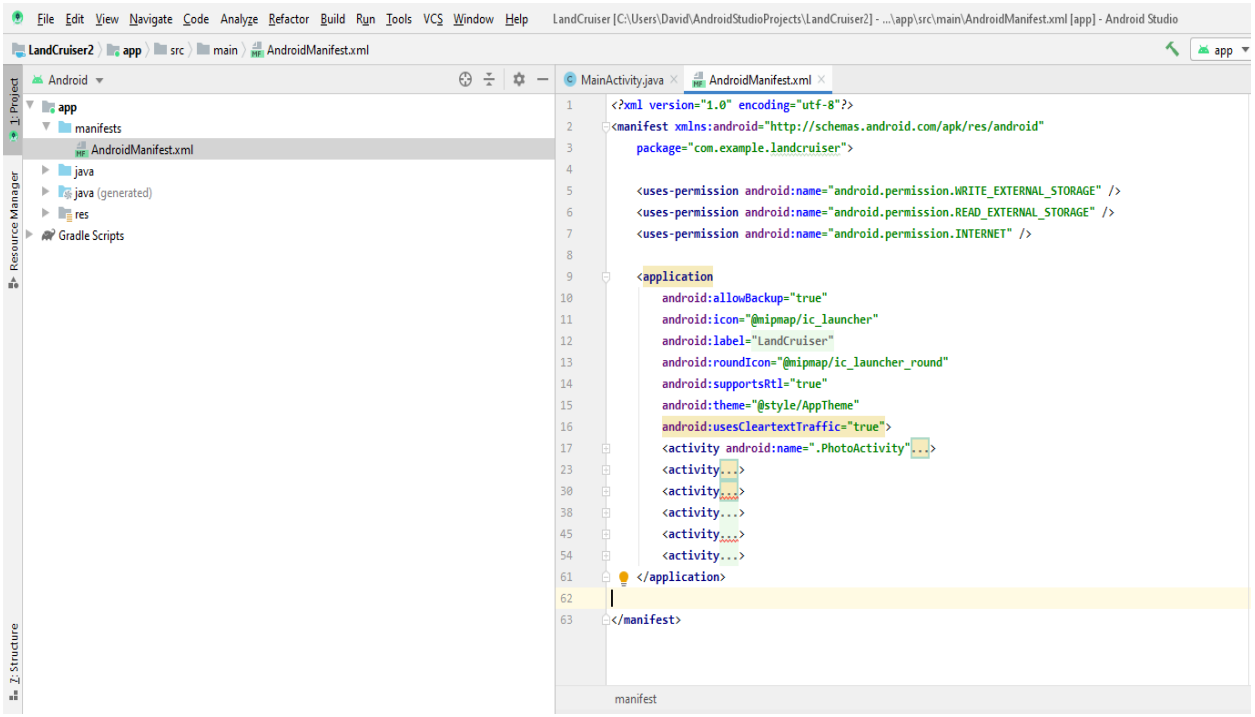
## 3.1 Android Studio

Android studio [9] je službeno integrirano razvojno okruženje (*engl. Integrated Development Environment, IDE)* za razvoj Android aplikacija zasnovano na IntelliJ IDEA programskoj podršci. Dostupan je na Windows, MacOS i Linux platformi. Osim programskog jezika Java, podržava Kotlin i C++.

Svaka aplikacija se sastoji od 3 datoteke : manifest, java, res

### 3.1.1. Manifest datoteka

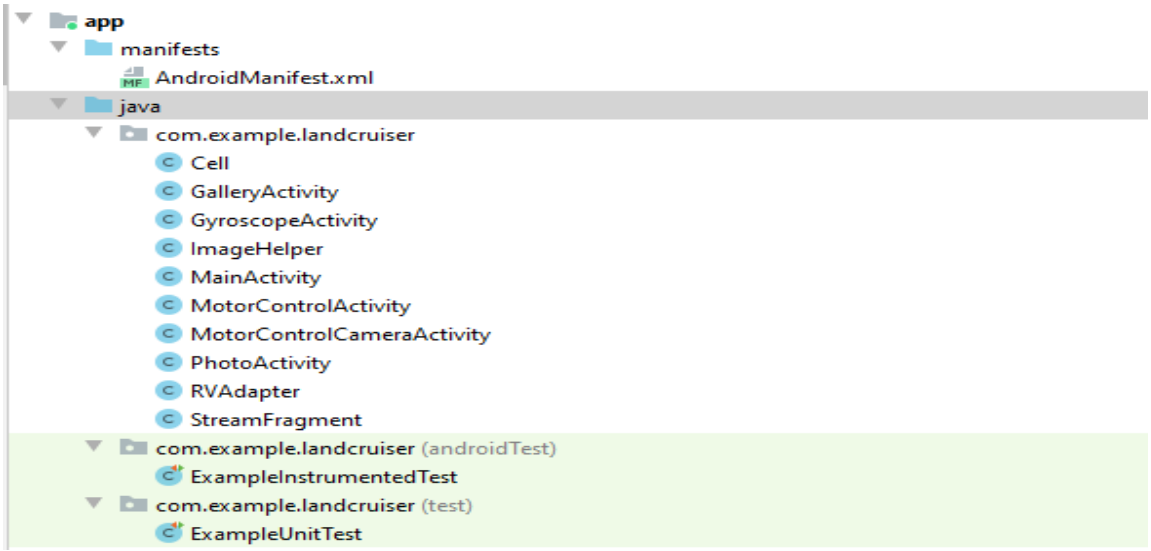
Sadrži AndroidManifest.xml datoteku koja sadrži meta podatke aplikacije, dozvole, sve aktivnosti i njihovu hijerarhiju. U slučaju za ovu aplikaciju na slici 3.1. vidimo kako AndroidManifest.xml ima dodane dozvole za čitanje i pisanje podataka sa vanjske memorije (linija koda 5 i 6) koje koristimo za spremanje fotografija na SD karticu i dozvolu za korištenje interneta (linija koda 7) što koristimo kako bi slali naredbe s aplikacije mobilnoj platformi i prikazali video prijenos s kamere mobilne platforme u aplikaciji. Svaka aktivnost (*engl. Activity* ) je navedena u Android manifest datoteci.




Slika 3.1 AndroidManifest datoteka

### 3.1.2. Java datoteka

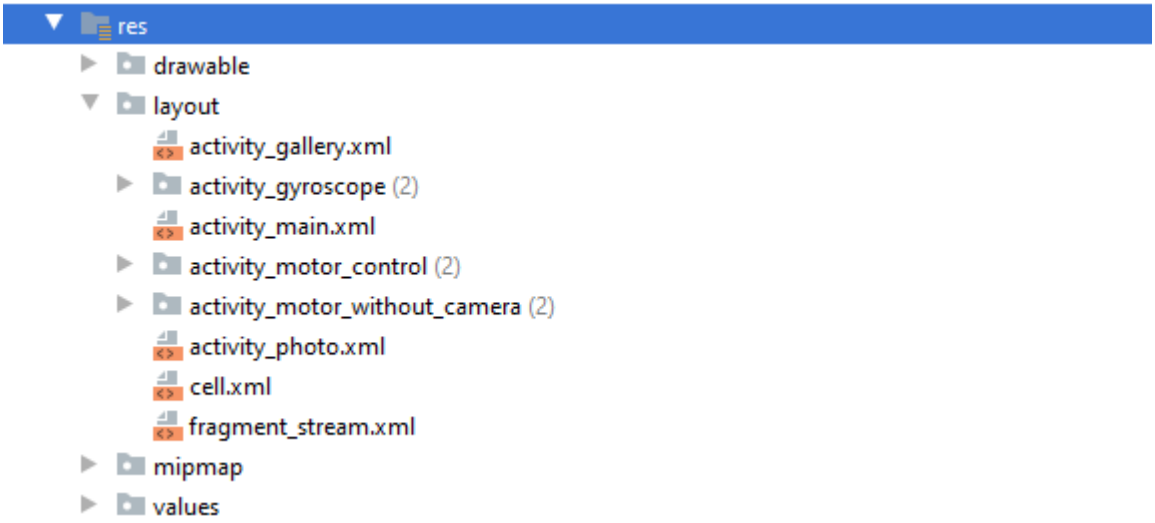
Sadrži sav izvorni kod. Svaka klasa je lako dostupna u Java datoteci. Osim toga sadrži i JUnit kod za testiranje.




Slika 3.2 Java datoteka

### 3.1.3. Res datoteka

Predstavlja sve resurse koji ne predstavljaju kod aplikacije, kao što je XML izgled (*engl. XML Layout*), UI stringovi i bitmap slike. Resursi služe za definiranje izgleda aplikacije i svih njenih komponenti. Svaka aktivnost ima svoj izgled napisan XML jezikom kojim definiramo smještaj komponenti, veličinu, boju, ubacivanje slika itd.




Slika 3.3 res datoteka

## 3.2 Programski jezik Java

Java [10] je objektno orijentirani jezik koji kreira programska rješenja za više platformi. Aplikacije napisane u Javi mogu se pokrenuti na svim uređajima koji podržavaju JVM (*engl. Java Virtual Machine)* bez obzira na arhitekturu računala. Sintaksa Jave je slična prograskim jezicima C i C++ i jedan je od najpopularnijih jezika danas.

## 3.3 XML

XML [11] (*engl. Extensible Markup Language*) je opisni jezik koji određujemo izgled aplikacije. Svaki dio aplikacije (Activity,Fragment) sadrži sadrži svoju XML datoteku koja definira veličinu, oblik, boju svakog vizualnog elementa korisničkog sučelja i dizajn njega samog. Sastoji se od 2 dijela. Prvi dio je zaglavlje. U zaglavlju se navode podaci koji opisuju XML dokument kao što je verzija preporuke prema čijima je pravilima napravljen i kodna stranica. Drugi dio je sadržaj dokumenta u kojem se nalazi korisni sadržaj koji je omeđen XML oznakama. Za potrebe ove aplikacije korišten je relativni prikaz (*engl. Relative layout*). Relativni prikaz omogućuje smještanje elemenata unutar sebe, relativno u odnosu na druge elemente ili u odnosu na vlastite točke vezanja. Kako bi se pojedini elementi mogli smjestiti u odnosu na druge

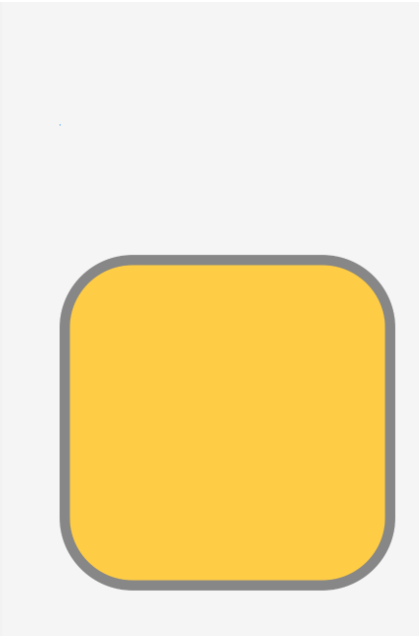
elemente koristimo identifikatore. Slika 3.4 Prikazuje XML datoteku početne aktivnosti. Svaka aktivnost može imati više prikaza što osim estetskog napretka omogućuje drugačiji raspored elemenata prilikom horizontalnog načina rada.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView...>
    <TextView
        android:id="@+id/tvChooseMode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Choose Mode:"
        android:layout_below="@id/etIPAddress"
        android:textSize="23dp"
        />
    <ImageView...>
    <EditText
        android:id="@+id/etIPAddress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="192.168.1.8"
        android:inputType="number"
        android:textSize="35dp"
        android:layout_below="@id/tvIPaddress"
        />
    <Button...>
    <ImageButton...>
    <TextView...>
    <RadioGroup
        android:id="@+id/rgMode"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_below="@+id/tvChooseMode"
        >
        <RadioButton...>
        <RadioButton...>
        <RadioButton...>
    </RadioGroup>
    <ImageView...>
</RelativeLayout>
```

Slika 3.4 activity_main.xml

Uz definiranje rasporeda i izgleda cijelog zaslona moguće je manipulirati izgledom pojedinih elemenata. Na taj način su modificirani gumbi (*engl. Button*). Osim estetskog napretka modificirani gumbi u ovom slučaju omogućuju lakše korištenje pri horizontalnom načinu rada, gdje video prijenos koji dolazi s Raspberry Pi uređaja je moguće pokrenuti preko cijelog zaslona. Gumbi postaju transparentni i ne zaklanjaju pogled na video prijenos.

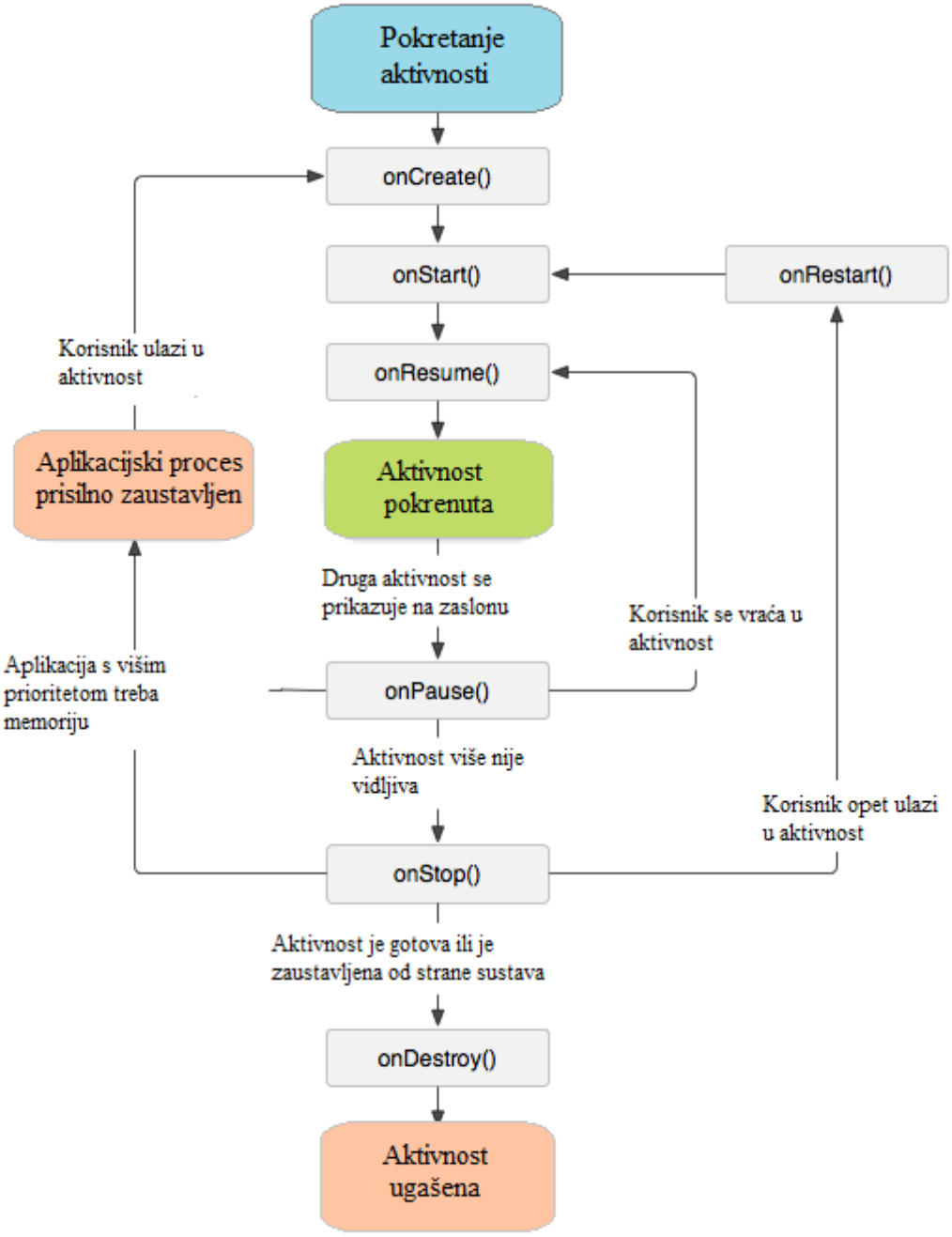
```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <corners
        android:radius="20dp"
        />
    <solid
        android:color="#FFCD45"
        />
    <padding
        android:left="0dp"
        android:top="0dp"
        android:right="0dp"
        android:bottom="0dp"
        />
    <size
        android:width="100dp"
        android:height="100dp"
        />
    <stroke
        android:width="3dp"
        android:color="#878787"
        />
</shape>
```



Slika 3.5 buttonshape.xml

## 3.4 Activity

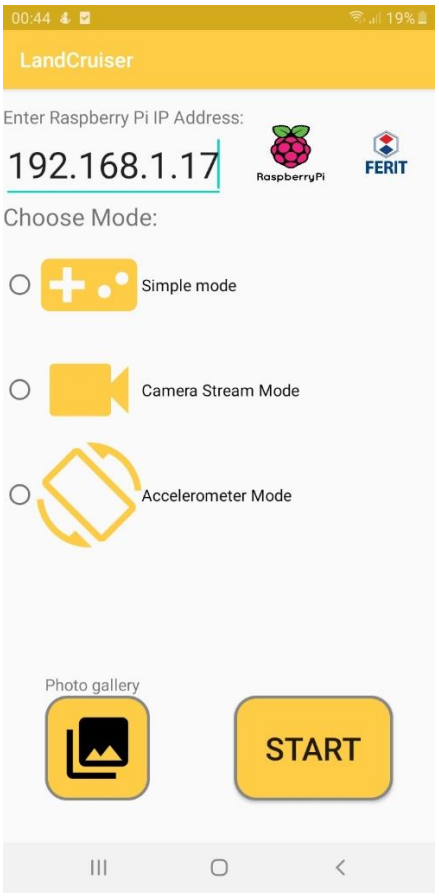
Aktivnost [12] (*engl.*Activity) je nalik formi u klasičnom programiranju desktop aplikacija. Predstavlja jedan zaslon aplikacije. Skoro sve aktivnosti imaju interakciju s korisnikom, stoga klasa Activity stvara prozor u kojem se nalaze ostale ulazno-izlazne komponente. Activity nema samo jednu ulaznu točku kao neki programi, ima veći broj metoda koje se pozivaju kada je potrebno i predstavljaju životni ciklus aktivnosti. Metode životnog ciklusa i redoslijed njihovog izvršavanja prikazani su na slici 3.6.




Slika 3.6 Životni ciklus aktivnosti

Dvije najvažnije metode koje moraju implementirati skoro sve klase koje nasljeđuju aktivnost su: onCreate() i onPause(). Metoda onCreate() je mjesto na kojem inicijaliziramo aktivnost. Budući da je sučelje napisano u XML kodu odvojeno od koda aplikacije, potrebno ga je „napuhati“ (*engl. Inflate*), odnosno potrebno je iz XML opisa kreirati objekte koji predstavljaju elemente korisničkog sučelja. Metoda onPause() se poziva kada korisnik prekida interakciju s aktivnošću. Tada je potrebno spremiti sve podatke i promjene za daljnji rad trenutnog zaslona

ukoliko je potrebno. U slučaju ove aplikacije uz početnu aktivnost koja je prikazana kao izbornik, svaki navedeni način rada stvoren je kao jedna aktivnost. Početna aktivnost koja se otvara pri ulasku u aplikaciju naziva se „MainActivity". Sadrži polje za unos IP adrese od Raspberry Pi uređaja s kojim komunicira, 3 radio gumba koja služe za odabir načina rada, gumb za pokretanje galerije fotografija i gumb „START“ za pokretanje druge aktivnosti ovisno o odabiru radio gumba. Izgled početne aktivnosti prikazan je na slici 3.7.




Slika 3.7 Izgled početne aktivnosti

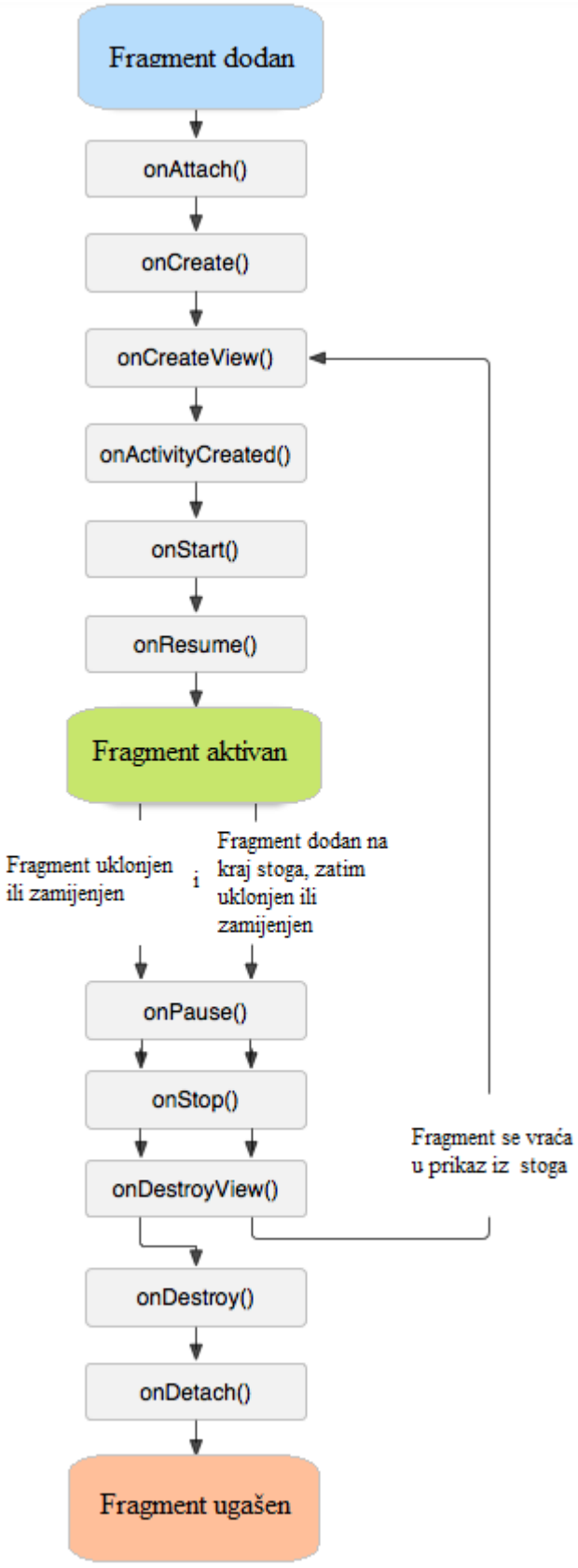
Prilikom pritiska na gumb „START“ provjerava se koji je način rada odabran i sljedeći taj odabir pokreće sljedeću aktivnost. Pokrećemo ju stvarajući objekt namjere (*engl. Intent*) i kao argument mu dajemo IP adresu koju prenosimo na sljedeću aktivnost (slika 3.8)

```
btnSubmitIP.setOnClickListener((view) → {
        setIPaddress(etIPAddress.getText().toString());
        if(mode==1)
        {
            Intent intent = new Intent(view.getContext(), MotorControlActivity.class);
            intent.putExtra( name: "IP_ADDRESS",getIPaddress());
            MotorControlActivity activity=new MotorControlActivity();
            view.getContext().startActivity(intent);

        }
        else if(mode==2)
        {
            Intent intent = new Intent(view.getContext(), MotorControlCameraActivity.class);
            intent.putExtra( name: "IP_ADDRESS",getIPaddress());
            MotorControlCameraActivity activity=new MotorControlCameraActivity();
            view.getContext().startActivity(intent);

        }

        else if(mode==3)
        {
            Intent intent = new Intent(view.getContext(), GyroscopeActivity.class);
            intent.putExtra( name: "IP_ADDRESS",getIPaddress());
            GyroscopeActivity activity=new GyroscopeActivity();
            view.getContext().startActivity(intent);
        }
```

Slika 3.8 Odabir drugih aktivnosti i njihovo pokretanje

## 3.5 Fragment

Fragmenti [13] predstavljaju klasu koja omogućuje modularan dizajn aktivnosti. Moguće je kombinirati više fragmenata u jednu aktivnost i koristiti jedan fragment u više aktivnosti. To je ostvareno time što fragment ima vlastiti životni ciklus, prima vlastite ulazne događaje koje se mogu dodavati ili uklanjati dok aktivnost radi. Fragment je usko vezan uz aktivnost u kojoj se nalazi jer kada je aktivnost pauzirana (metoda onPause() ) pauziraju se i svi fragmenti unutar navedene aktivnosti. Kada je aktivnost uništena (metoda onDestroy() ) uništavaju se i svi fragmenti unutar aktivnosti. Međutim kada je aktivnost pokrenuta možemo manipulirati svakim fragmentom nezavisno, što nam omogućuje vlastiti životni ciklus fragmenta i vlastite metode (slika 3.9).



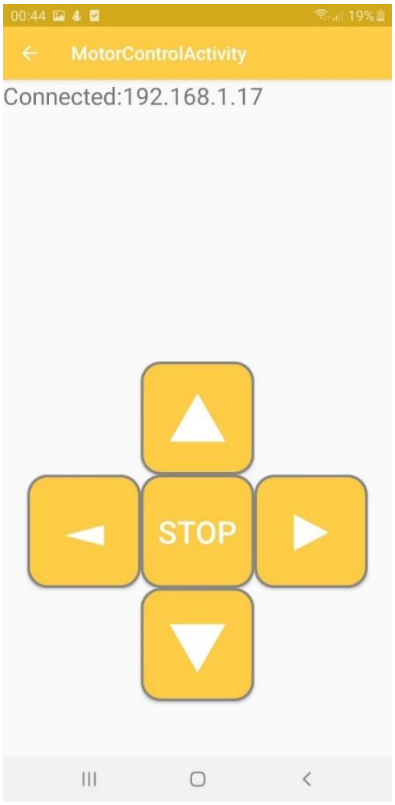

Slika 3.9 Životni ciklus fragmenta

Za svrhe ove aplikacije fragment je iskorišten za emitiranje video prijenosa uživo s Raspberry Pi kamere. Fragment se instancira unutar aktivnosti gdje se koristi, točnije u metodi onCreate(). Predajemo mu IP adresu kao argument preko objekta klase „Bundle“ i pozivom funkcije commit() fragment započinje svoj životni ciklus. Unutar klase fragmenta definiran je rad video prijenosa i fragment ga obavlja samostalno.

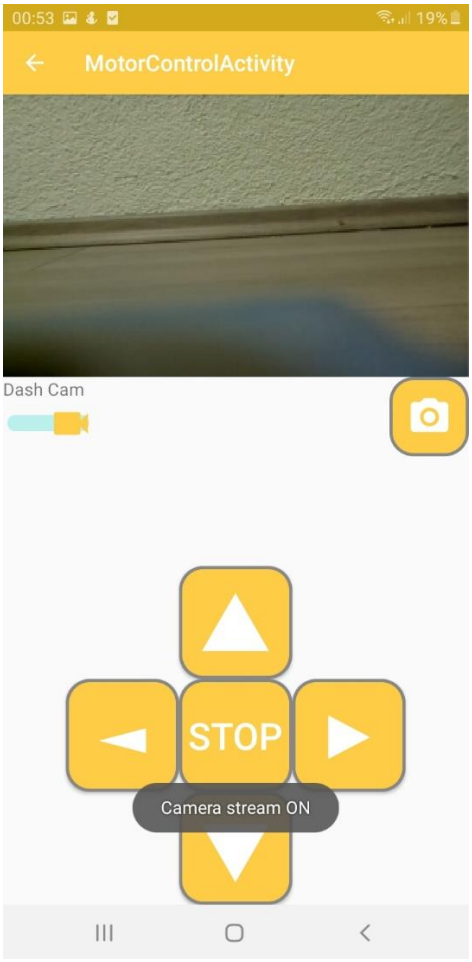
```
Bundle bundle=new Bundle();
bundle.putString("etText",IPAddress);
fragmentManager=getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
final StreamFragment fragment=new StreamFragment();
fragment.setArguments(bundle);
fragmentTransaction.add(R.id.streamFragment,fragment);
fragmentTransaction.commit();
```

Slika 3.10 Stvaranje fragmenta

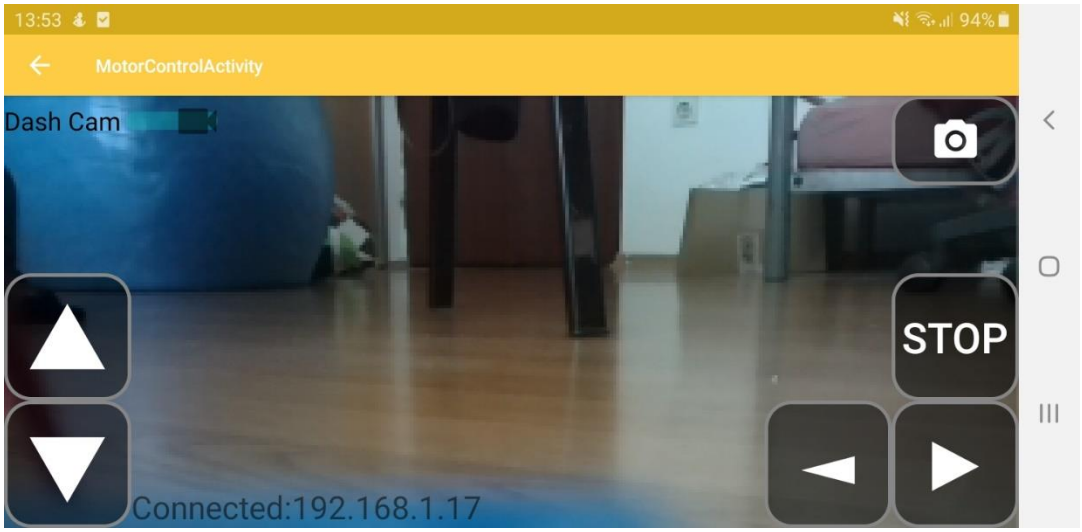
Na slici 3.11 prikazana je aktivnost bez fragmenta, a na slici 3.12 je prikazan način rada koji koristi fragment. Kod horizontalnog načina rada (slika 3.13) fragment se nalazi preko cijelog zaslona u pozadini.




Slika 3.11 Način rada bez kamere




Slika 3.12 Način rada s kamerom




Slika 3.13 Horizontalni način rada s kamerom

## 3.6 Slanje naredbi

Kako bi uspješno upravljali mobilnom platformom potrebno je prenijeti informaciju da je korisnik pritisnuo gumb koji označava određeni smjer kretanja. Prvo se stvaraju slušatelji događaja (*engl. Event listener*) koji detektiraju svaki pritisak na gumb. U ovom slučaju koristi se „onTouchListener“ koji osim pritiska gumba bilježi i njegovo puštanje što je vrlo važno u ovom slučaju kako motori ne bi stalno ostali u aktivnom stanju. Nadalje, nakon što slušatelji događaja reagiraju na promjenu, potrebno je prenijeti informaciju do mobilne platforme. Kako bi to ostvarili koristi se tehnologija *Socket* [13]. Predstavlja jedinstvenu vezu između dvije točke u mreži (*engl. Point-to-point connection*). Komuniciraju na osnovi klijent / server ili na kompleksnijim sustavima koristeći veći broj *socketa*. Primanje podataka pomoću *socketa* opisano je u poglavlju 2.6 (Primanje podataka s druge adrese). Kako bi došlo do razmjene informacija potrebno je uskladiti brojeve porta na oba kraja sustava. Implementacija *socketa* za android aplikaciju prikazana je na slici 3.14.

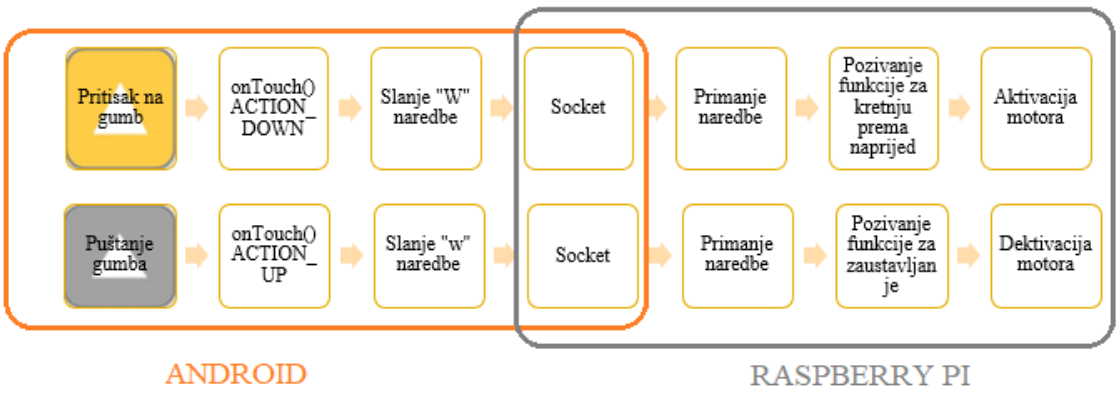
```
public void getIPandPort(){
    String IPandPort=IPAddress+":21567";
    Log.d( tag: "MYTEST", msg: "IP String "+IPandPort);
    String temp[]=IPandPort.split( regex: ":");
    wifiModuleIP=temp[0];
    wifiModulePort=Integer.valueOf(temp[1]);
    Log.d( tag: "MY TEST", msg: "IP:"+wifiModulePort);
    Log.d( tag: "MY TEST", msg: "PORT:"+wifiModulePort);

}

public class Soket_AsyncTask extends AsyncTask<Void,Void,Void> {
    Socket socket;

    @Override
    protected Void doInBackground(Void... params) {
        try{
            InetAddress inetAddress=InetAddress.getByName(MotorControlCameraActivity.wifiModuleIP);
            socket=new java.net.Socket(inetAddress, MotorControlCameraActivity.wifiModulePort);
            DataOutputStream dataOutputStream=new DataOutputStream(socket.getOutputStream());
            dataOutputStream.writeBytes(CMD);
            dataOutputStream.close();
            socket.close();
        }catch (UnknownHostException e ){
            e.printStackTrace();
        }catch (IOException e){
            e.printStackTrace();
        }
        return null;
    }
```

Slika 3.14 Implementacija *socketa*

Kako je prijašnje opisano u poglavlju 2.6 (Primanje podataka s druge adrese), kako bi se aktivirali motori mobilne platforme potrebno je poslati znakove (W, S, A, D),dok za gašenje istih je potrebno poslati znakove (w, s, a, d) koje kasnije obrađuje program MotorControl.py kojeg pokreće Raspberry Pi računalo. Svakom slovu je dodjeljen jedan gumb. Na pritisak jednog od gumba za željeni smjer metoda onTouch() šalje jedan od znakova (W, S, A, D). Puštanjem istog gumba metoda onTouch() šalje odgovarajući znak (w, s, a, d) i time se zaustavlja kretanje u trenutnom smjeru. Primjer slanja znaka „W“ za kretanje platforme prema naprijed i slanje znaka „w“ za prekid kretanja prikazan je na slici 3.15.

```
btnUp.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_DOWN) {
            getIPandPort();
            CMD="W";
            MotorControlActivity.Soket_AsyncTask cmd_sendinfo = new MotorControlActivity.Soket_AsyncTask();
            cmd_sendinfo.execute();
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            getIPandPort();
            CMD="w";
            MotorControlActivity.Soket_AsyncTask cmd_sendinfo = new MotorControlActivity.Soket_AsyncTask();
            cmd_sendinfo.execute();
        }
        return true;
    }
```

Slika 3.15 Slanje informacija



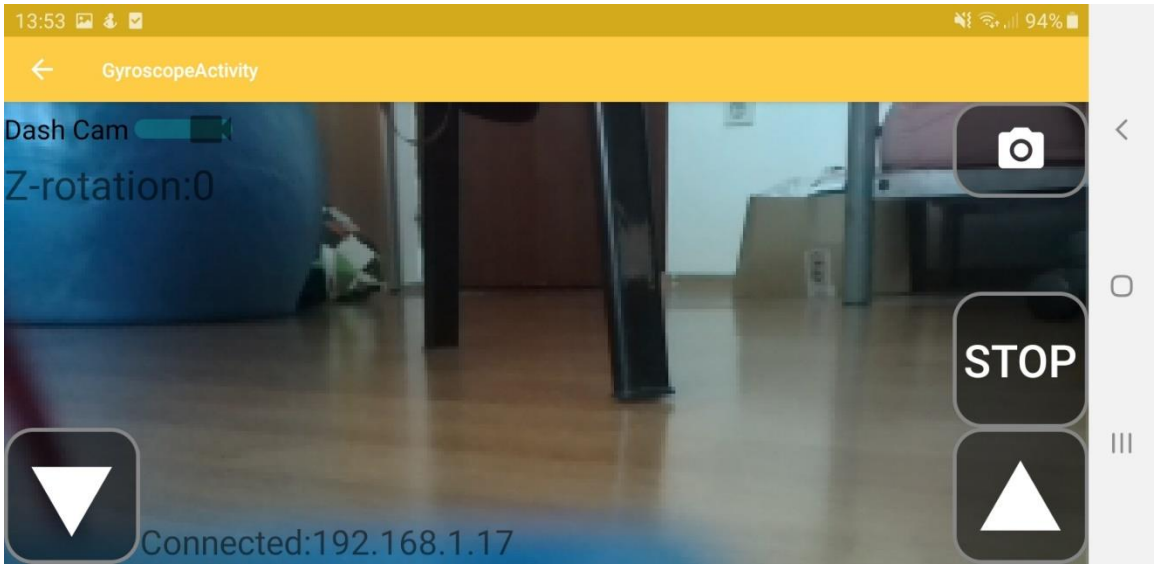

Slika 3.16. Analogija slanja podataka

Uz upravljanje korišteno na prva dva prikazana načina rada uz kameru ili bez, treći način rada koristi upravljanje mobilnom platformom koristeći senzore android uređaja (Slika 3.17).

Korišteni senzor mjeri akceleraciju na 3 osi (x, y, z). Za svrhe ove aplikacije kako bi simulirali skretanje volana na automobilu uzimamo samo y os senzora u brojčanom obliku. Vrijednost y osi manju od -4 tumačimo kao skretanje u lijevo, time *socket* šalje znak „A“, isto tako vrijednost y osi veću od 4 tumačimo kao skretanje u desno i šalje se znak „D“.

```
sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
accSensor =sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

if(accSensor ==null){
    Toast.makeText( context: this, text: "The device has no Gyroscope",Toast.LENGTH_SHORT).show();
    finish();
}
accelerometerEventListener = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {

        if(sensorValue!=(int)sensorEvent.values[1]){
            sensorValue= (int)sensorEvent.values[1];
            onSensorValueChanged(sensorValue);
        }
    }
```

Slika 3.17 Implementacija senzora




Slika 3.18 Način rada rotiranjem uređaja

# 4. ZAKLJUČAK

Raspberry Pi računalo uz mala ulaganja ima vrlo velik opseg mogućnosti za ispunjenje zadanih ciljeva. U ovom slučaju umjesto korištenja tradicionalnih radio frekvencija za upravljanje mobilnim platformama, implementirano je Raspberry pi računalo koje se uz dodatne komponente koristi za upravljanje električnih istosmjernih motora. Računalo uspješno kontrolira motore i time omogućuje kretanje. Kako bi korisnik svojevoljno birao smjer i vrijeme kretanja, u radu je opisana izrada Android aplikacije. Aplikacija uspješno omogućuje primanje naredbi od korisnika, zatim te iste naredbe šalje, uz korištenje WiFi komunikacije, do Raspberry Pi računala. Aplikacija je kompatibilna s bilo kojim Android uređajem što omogućuje veću pristupačnost, lakše korištenje a samim time i primjenu u drugim granama tehnologije uz robotiku.

# LITERATURA


[1] Raspberry Pi, Wikipedia, 2020.
Dostupno na :https://en.wikipedia.org/wiki/Raspberry_Pi [12.07.2020.]

[2] Specifications, Raspberry Pi Foundation, 2020.
Dostupno na: https://www.raspberrypi.org/products/raspberry-pi-zero/ [12.07.2020.]

[3] Istosmjerni DC motor, e-radionica, 2020.
Dostupno na: https://e-radionica.com/hr/blog/2018/11/12/istosmjerni-dc-motor/ [12.07.2020]

[4] Arduino DC Motor Control -L298N H Bridge, How to Mechatronics, 2020.
Dostupno na: https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/ [13.07.2020.]

[5] Raspberry Pi OS, DistroWatch.com, 2020.
Dostupno na: https://distrowatch.com/table.php?distribution=raspios [13.07.2020.]

[6] Raspberry Pi Foundation,GPIO, 2020. Dostupno na:
https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md
[24.07.2020.]

[7] socket — Low-level networking interface, Python Software Foundation, 2020.
Dostupno na: https://docs.python.org/3/library/socket.html [25.07.2020.]

[8] Meet Android Studio, Android Developers, Google 2020.
Dostupno na: https://developer.android.com/studio/intro [12.08.2020.]

[9] Java (programming language), Wikipedia, 2020.
Dostupno na: https://en.wikipedia.org/wiki/Java_(programming_language) [18.08.2020.]

[10] XML Tutorial, W3Schools, 2020. Dostupno na: https://www.w3schools.com/xml/
[22.08.2020.]

[11] Activity, Android Developers, Google, 2020. Dostupno na:
https://developer.android.com/reference/android/app/Activity [22.08.2020.]

[12] Fragment, Android Developers, Google, 2020. Dostupno na:
https://developer.android.com/guide/components/fragments [23.08.2020.]

# SAŽETAK

U ovom završnom radu obrađena je tema izrade i dizajna Android aplikacije za daljinsko upravljanje mobilnom robotskom platformom, temeljenoj na Raspberry Pi računalu. Opisano je Raspberry Pi računalo uz dodatne komponente koje tvore mobilnu platformu. Detaljno objašnjene su sve tehnologije koje Android uređaju omogućuju daljinsko upravljanje mobilnom platformom, uz korištenje WiFi komunikacije za razmjenu informacija. Prikazana je funkcionalnost aplikacije, dani su primjeri programskog koda za rad aplikacije i mobilne robotske platforme.

Ključne riječi : Android aplikacija, daljinsko upravljanje, mobilna robotska platforma, Raspberry pi,

## ABSTRACT

This final article addresses the topic of creating an Android application for remote control of mobile robotic platform, based on Raspberry pi computer. A Raspberry Pi computer with additional components that make up a mobile platform is described. All technologies that enable the Android device to remotely control the mobile platform, with the use of WiFi communication for information exchange, are explained in detail. The functionality of the application is presented, examples of program code for the operation of the application and the mobile robotic platform are given.



Keywords : Android app, remote control, mobile robot platform, Raspberry pi

# ŽIVOTOPIS

David Dumančić rođen je 12.02.1998. u Požegi. Pohađao je Osnovnu Školu Antuna Kanižlića u Požegi. Srednjoškolsko obrazovanje nastavlja u Tehničkoj Školi Požega koju završava 2016. godine sa završnim radom pod nazivom „Cloud Computing – Google spremnik podataka i fotografija“ te upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Trenutno završava treću godinu preddiplomskog studija i planira nastaviti studiranje na nekom od diplomskih studija.

# PRILOZI

**MotorControl.py**

```
import RPi.GPIO as GPIO
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(8,GPIO.OUT)
    GPIO.setup(12,GPIO.OUT)
    GPIO.setup(10,GPIO.OUT)
    GPIO.setup(16,GPIO.OUT)

def MotorUp():
    print ('Motor UP')
    GPIO.output(10,True)
def MotorDown():
    print ('Motor DOWN')
    GPIO.output(8,True)
def MotorLeft():
    print ('Motor LEFT')
    GPIO.output(12,True)
def MotorRight():
    print ('Motor RIGHT')
    GPIO.output(16,True)
def MotorUpStop():
    print ('Motor UP Stop')
    GPIO.output(10,False)
def MotorDownStop():
    print ('Motor DOWN Stop')
    GPIO.output(8,False)
def MotorLeftStop():
    print ('Motor LEFT Stop')
    GPIO.output(12,False)
def MotorRightStop():
    print ('Motor RIGHT Stop')
    GPIO.output(16,False)
def MotorStop():
    print ('Motor STOP')
    GPIO.output(8,False)
    GPIO.output(10,False)
    GPIO.output(12,False)
    GPIO.output(16,False)

if __name__=='__main__':
    setup()
```

**PiServer.py**

```
import MotorControl
from socket import *
from time import ctime
import time
import RPi.GPIO as GPIO


MotorControl.setup()
GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setup(8,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)
GPIO.setup(10,GPIO.OUT)
GPIO.setup(16,GPIO.OUT)

ctrCmd = ['W','S','A','D','w','s','a','d','0']

HOST = ''
PORT = 21567
BUFSIZE = 1024
ADDR = (HOST,PORT)

tcpSerSock = socket(AF_INET, SOCK_STREAM)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(5)

while True:
        print ('Waiting for connection')
        tcpCliSock,addr = tcpSerSock.accept()
        print ('...connected from :', addr)


        try:
                while True:
                        data =''
                        data1=data
                        data = tcpCliSock.recv(BUFSIZE).decode('utf-8')
                        print (data)
                        if data == ctrCmd[0]:
                            MotorControl.MotorUp()
                        elif data == ctrCmd[1]:
                            MotorControl.MotorDown()
                        elif data == ctrCmd[2]:
                            MotorControl.MotorLeft()
                        elif data == ctrCmd[3]:
                            MotorControl.MotorRight()
                        elif data == ctrCmd[4]:
                            MotorControl.MotorUpStop()
                        elif data == ctrCmd[5]:
                            MotorControl.MotorDownStop()
                        elif data == ctrCmd[6]:
                            MotorControl.MotorLeftStop()
                        elif data == ctrCmd[7]:
                            MotorControl.MotorRightStop()
                        elif data == ctrCmd[8]:
                            MotorControl.MotorStop()
                        elif not data1:
                                break
        except KeyboardInterrupt:
                GPIO.cleanup()
                MotorControl.close()
tcpSerSock.close();
```

**PiCamera.py**

```
# Web streaming example
# Source code from the official PiCamera package
# http://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming

import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Camera</h1></center>
<center><img src="stream.mjpg" width="640" height="480"></center>
</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()

        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace;
boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
```

```
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='320x240', framerate=12) as camera:
    output = StreamingOutput()
    #Uncomment the next line to change your Pi's Camera rotation (in degrees)
    camera.rotation = 180
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()
```

**MainActivity.java**

```
package com.example.landcruiser;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Button btnSubmitIP;
    EditText etIPAddress;
    private String IPaddress;
    RadioGroup radioGroup;
    RadioButton radioButtonMode1;
    RadioButton radioButtonMode2;
    ImageButton ibGallery;
    int mode=0;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnSubmitIP=findViewById(R.id.btnSubmitIP);
        etIPAddress=findViewById(R.id.etIPAddress);
        radioGroup=findViewById(R.id.rgMode);
        radioButtonMode1=findViewById(R.id.rbMode1);
        radioButtonMode2=findViewById(R.id.rbMode2);
        ibGallery=findViewById(R.id.ibGallery);

        ibGallery.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(view.getContext(),
GalleryActivity.class);
                view.getContext().startActivity(intent);
            }
        });
        btnSubmitIP.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                setIPaddress(etIPAddress.getText().toString());
                if(mode==1)
                {
                    Intent intent = new Intent(view.getContext(),
MotorControlActivity.class);
                    intent.putExtra("IP_ADDRESS",getIPaddress());
                    MotorControlActivity activity=new MotorControlActivity();
                    view.getContext().startActivity(intent);

                }
                else if(mode==2)
                {
                    Intent intent = new Intent(view.getContext(),
MotorControlCameraActivity.class);
                    intent.putExtra("IP_ADDRESS",getIPaddress());
                    MotorControlCameraActivity activity=new
MotorControlCameraActivity();
                    view.getContext().startActivity(intent);

                }

                else if(mode==3)
                {
                    Intent intent = new Intent(view.getContext(),
GyroscopeActivity.class);
                    intent.putExtra("IP_ADDRESS",getIPaddress());
                    GyroscopeActivity activity=new GyroscopeActivity();
                    view.getContext().startActivity(intent);
                }
                else
                    {
                        Toast toast=
Toast.makeText(getApplicationContext(),"Please choose
mode",Toast.LENGTH_SHORT);
                        toast.show();
                    }
            }
        });
    }
    public void onRadioButtonClicked(View view) {
```

```
        boolean checked = ((RadioButton) view).isChecked();

        switch(view.getId()) {
            case R.id.rbMode1:
                if (checked)
                    mode=1;
                    break;
            case R.id.rbMode2:
                if (checked)
                    mode=2;
                    break;
            case R.id.rbMode3:
                if (checked)
                    mode=3;
                break;
        }
    }
    public String getIPaddress() {
        return IPaddress;
    }

    public void setIPaddress(String IPaddress) {
        this.IPaddress = IPaddress;
    }
}
```

**MotorControlActivity.java**

```
package com.example.landcruiser;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;

public class MotorControlActivity extends AppCompatActivity {
    TextView tvText;
    String IPAddress;
    Button btnUp;
    Button btnDown;
    Button btnStop;
    Button btnLeft;
    Button btnRight;
    Socket MyAppSocket=null;
    public static String wifiModuleIP="";
    public static int wifiModulePort=0;
    public static String CMD="0";
    @SuppressLint("ClickableViewAccessibility")
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_motor_without_camera);
        btnUp=(Button) findViewById(R.id.btnUp);
        btnDown=(Button)findViewById(R.id.btnDown);
        btnStop=(Button)findViewById(R.id.btnStop);
        btnLeft=(Button)findViewById(R.id.btnLeft);
        btnRight=(Button)findViewById(R.id.btnRight);
        tvText=findViewById(R.id.tvText);
        //upbutton
        getSupportActionBar().setTitle("MotorControlActivity");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        //primanje ip adrese od prvog activitija
        IPAddress =getIntent().getStringExtra("IP_ADDRESS");
        tvText.setText(IPAddress);
        btnStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getIPandPort();
                CMD="0";
                MotorControlActivity.Soket_AsyncTask cmd_increase_servo = new
MotorControlActivity.Soket_AsyncTask();
                cmd_increase_servo.execute();
            }
        });
        btnUp.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="W";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="w";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });
        btnRight.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="D";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="d";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
```

```
                }
                return true;
            }

        });
        btnLeft.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="A";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="a";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });
        btnDown.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="S";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="s";
                    MotorControlActivity.Soket_AsyncTask cmd_increase_servo =
new MotorControlActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });

    }
    public void getIPandPort(){
        String IPandPort=IPAddress+":21567";
        Log.d("MYTEST","IP String "+IPandPort);
        String temp[]=IPandPort.split(":");
        wifiModuleIP=temp[0];
        wifiModulePort=Integer.valueOf(temp[1]);
        Log.d("MY TEST","IP:"+wifiModulePort);
        Log.d("MY TEST","PORT:"+wifiModulePort);

    }

    public class Soket_AsyncTask extends AsyncTask<Void,Void,Void> {
        Socket socket;
```

```
        @Override
        protected Void doInBackground(Void... params) {
            try{
                InetAddress
inetAddress=InetAddress.getByName(MotorControlCameraActivity.wifiModuleIP);
                socket=new java.net.Socket(inetAddress,
MotorControlCameraActivity.wifiModulePort);
                DataOutputStream dataOutputStream=new
DataOutputStream(socket.getOutputStream());
                dataOutputStream.writeBytes(CMD);
                dataOutputStream.close();
                socket.close();
            }catch (UnknownHostException e ){
                e.printStackTrace();
            }catch (IOException e){
                e.printStackTrace();
            }
            return null;
        }
    }
}
```

**MotorControlCameraActivity.java**

```
package com.example.landcruiser;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Date;

public class MotorControlCameraActivity extends AppCompatActivity {
```

```
TextView tvText;
String IPAddress;
Button btnUp;
Button btnDown;
Button btnStop;
Button btnLeft;
Button btnRight;
ImageButton btnScreenshot;
Switch onoffSwitch;
Socket MyAppSocket=null;
public boolean cameraOn= true;
public static String wifiModuleIP="";
public static int wifiModulePort=0;
public static String CMD="0";
private StreamFragment streamFragment;
private FragmentManager fragmentManager;
    @SuppressLint("ClickableViewAccessibility")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_motor_control);
        final View rootView =
getWindow().getDecorView().findViewById(android.R.id.content);
        btnUp=(Button) findViewById(R.id.btnUp);
        btnDown=(Button)findViewById(R.id.btnDown);
        btnStop=(Button)findViewById(R.id.btnStop);
        btnLeft=(Button)findViewById(R.id.btnLeft);
        btnRight=(Button)findViewById(R.id.btnRight);
        btnScreenshot=(ImageButton)findViewById(R.id.btnScreenshot);
        onoffSwitch=(Switch)findViewById(R.id.onoffSwitch1);
        tvText=findViewById(R.id.tvText);
        getSupportActionBar().setTitle("MotorControlActivity");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        IPAddress =getIntent().getStringExtra("IP_ADDRESS");
        tvText.setText(IPAddress);
        Bundle bundle=new Bundle();
        bundle.putString("etText",IPAddress);
        fragmentManager=getSupportFragmentManager();
        FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
        final StreamFragment fragment=new StreamFragment();
        fragment.setArguments(bundle);
        fragmentTransaction.add(R.id.streamFragment,fragment);
        fragmentTransaction.commit();

        onoffSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if(isChecked==false)
                    stopStream();
                else if(isChecked==true){
                    startStream();
                }
            }

        });

        btnStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                getIPandPort();
                CMD="0";
                Soket_AsyncTask cmd_increase_servo = new Soket_AsyncTask();
                cmd_increase_servo.execute();
            }
        });
        btnUp.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="W";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="w";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });
        btnRight.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="D";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="d";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });
        btnLeft.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="A";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="a";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
```

```
                }
                return true;
            }

        });
        btnDown.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    getIPandPort();
                    CMD="S";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    getIPandPort();
                    CMD="s";
                    Soket_AsyncTask cmd_increase_servo = new
Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });

if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.M&&checkSelfPermission(Manifest
.permission.WRITE_EXTERNAL_STORAGE)!= PackageManager.PERMISSION_GRANTED){
            requestPermissions(new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},1000);
        }
        else{
        }
        btnScreenshot.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                View
rootView=getWindow().getDecorView().findViewById(android.R.id.content);//slik
a cijeli app
                Bitmap bitmap=getScreenshot(rootView);
                store(bitmap,"ScreenShot-
"+java.text.SimpleDateFormat.getDateTimeInstance().format(new
Date())+".png");
            }
        });

    }

    @Override
    protected void onPause() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        if(fragment.isStreamOn())stopStream();
        super.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }
```

```
    @Override
    protected void onStop() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        if(fragment.isStreamOn())stopStream();
        super.onStop();
    }

    private void stopStream() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        fragment.stopStream();
        Toast.makeText(this, "Camera stream OFF", Toast.LENGTH_SHORT).show();
    }

    private void startStream() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        fragment.startStream();
        Toast.makeText(this,"Camera stream ON",Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        if(requestCode==1000){
            if(grantResults[0] == RESULT_OK){
                Toast.makeText(this,"Permission
granted",Toast.LENGTH_SHORT).show();
            }
            else{
                Toast.makeText(this,"Permission
denied",Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    public static Bitmap getScreenshot(View view){
        View screenView=view.getRootView();
        screenView.setDrawingCacheEnabled(true);
        Bitmap bitmap=Bitmap.createBitmap(screenView.getDrawingCache());
        screenView.setDrawingCacheEnabled(false);
        return bitmap;
    }
    public void store(Bitmap bm,String fileName){
        String
dirPath=Environment.getExternalStorageDirectory().getAbsolutePath()+"/MyFIles
";
        File dir = new File(dirPath);
        if(!dir.exists()){
            dir.mkdirs();
        }
        File file=new File(dirPath,fileName);
            try{
                FileOutputStream fos=new FileOutputStream(file);
                bm.compress(Bitmap.CompressFormat.PNG,100,fos);
```

```
                fos.flush();
                fos.close();
                Toast.makeText(this,"Saved",Toast.LENGTH_SHORT).show();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
                Toast.makeText(this,"Error!",Toast.LENGTH_SHORT).show();
            } catch (IOException e) {
                e.printStackTrace();
            }
    }


    public void getIPandPort(){
        String IPandPort=IPAddress+":21567";
        Log.d("MYTEST","IP String "+IPandPort);
        String temp[]=IPandPort.split(":");
        wifiModuleIP=temp[0];
        wifiModulePort=Integer.valueOf(temp[1]);
        Log.d("MY TEST","IP:"+wifiModulePort);
        Log.d("MY TEST","PORT:"+wifiModulePort);

    }


    public class Soket_AsyncTask extends AsyncTask<Void,Void,Void> {
        Socket socket;
        String response;
        @Override
        protected Void doInBackground(Void... params) {
            try{
                InetAddress
inetAddress=InetAddress.getByName(MotorControlCameraActivity.wifiModuleIP);
                socket=new java.net.Socket(inetAddress,
MotorControlCameraActivity.wifiModulePort);
                DataOutputStream dataOutputStream=new
DataOutputStream(socket.getOutputStream());
                dataOutputStream.writeBytes(CMD);
                dataOutputStream.close();
                socket.close();
            }catch (UnknownHostException e ){
                e.printStackTrace();

            }catch (IOException e){
                e.printStackTrace();
            }
            return null;
        }
    }
}
```

**StreamFragment.java**

```
package com.example.landcruiser;

import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import com.longdo.mjpegviewer.MjpegView;


public class StreamFragment extends Fragment {
    private String IPAddress;
    private MjpegView mview;
    private boolean streamOn=false;

    public StreamFragment() {
    }

    public static StreamFragment newInstance(String IPAddress) {
        StreamFragment fragment = new StreamFragment();
        Bundle args = new Bundle();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {

        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        String strtext = getArguments().getString("etText");
        setIPAddress(getArguments().getString("etText"));
        return inflater.inflate(R.layout.fragment_stream, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        //MJPEg
        mview=view.findViewById(R.id.mjpegview);
        mview.setAdjustHeight(true);
        mview.setMode(MjpegView.MODE_FIT_WIDTH);
        mview.setUrl("http://"+getIPAddress()+":8000/stream.mjpg");
        mview.setRecycleBitmap(true);
    }

    public void setIPAddress(String IPAddress) {
        this.IPAddress = IPAddress;
    }
    public String getIPAddress(){
```

```
        return this.IPAddress;
    }
    public void startStream(){
        mview.startStream();
        setStreamState(true);
    };
    public void stopStream(){
        mview.stopStream();
        setStreamState(false);
    }

    public boolean isStreamOn() {
        return streamOn;
    }

    public void setStreamState(boolean streamState) {
        this.streamOn = streamState;
    }
}
```

**GyroscopeActivity.java**

```
package com.example.landcruiser;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Date;
```

```
public class GyroscopeActivity extends AppCompatActivity {
    TextView tvText;
    String IPAddress;
    Button btnUp;
    Button btnDown;
    Button btnStop;
    Socket MyAppSocket=null;
    TextView tvRotation;
    Switch onoffSwitch;
    ImageButton btnScreenshot;

    public static String wifiModuleIP="";
    public static int wifiModulePort=0;
    public static String CMD="0";
    private StreamFragment streamFragment;
    private FragmentManager fragmentManager;
    private SensorManager sensorManager;
    private Sensor gyroscopeSensor;
    private SensorEventListener gyroscopeEventListener;
    private int sensorValue=0;
    private boolean motor_w=false;
    private boolean motor_a=false;
    private boolean motor_s=false;
    private boolean motor_d=false;

    public boolean isMotor_w() {
        return motor_w;
    }

    public void setMotor_w(boolean motor_w) {
        this.motor_w = motor_w;
    }

    public boolean isMotor_a() {
        return motor_a;
    }

    public void setMotor_a(boolean motor_a) {
        this.motor_a = motor_a;
    }

    public boolean isMotor_s() {
        return motor_s;
    }

    public void setMotor_s(boolean motor_s) {
        this.motor_s = motor_s;
    }

    public boolean isMotor_d() {
        return motor_d;
    }

    public void setMotor_d(boolean motor_d) {
        this.motor_d = motor_d;
    }
    public void setMotorsTo0(){
        this.motor_a=false;
        this.motor_w=false;
        this.motor_s=false;
        this.motor_d=false;
```

```
    }
    @SuppressLint("ClickableViewAccessibility")
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gyroscope);
        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);

gyroscopeSensor=sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        btnUp=(Button) findViewById(R.id.btnUp);
        btnDown=(Button)findViewById(R.id.btnDown);
        btnStop=(Button)findViewById(R.id.btnStop);
        tvText=findViewById(R.id.tvText);
        tvRotation=findViewById(R.id.tvRotation);
        onoffSwitch=findViewById(R.id.onoffSwitch1);
        btnScreenshot=(ImageButton)findViewById(R.id.btnScreenshot);


        if(gyroscopeSensor==null){
            Toast.makeText(this,"The device has no
Gyroscope",Toast.LENGTH_SHORT).show();
            finish();
        }
        gyroscopeEventListener = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent sensorEvent) {

                if(sensorValue!=(int)sensorEvent.values[1]){
                    sensorValue= (int)sensorEvent.values[1];
                    onSensorValueChanged(sensorValue);
                }
            }

            @Override
            public void onAccuracyChanged(Sensor sensor, int i) {

            }
        };
        onoffSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if(isChecked==false)
                    stopStream();
                else if(isChecked==true){
                    startStream();
                }
            }

        });
        getSupportActionBar().setTitle("GyroscopeActivity");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        IPAddress =getIntent().getStringExtra("IP_ADDRESS");
        tvText.setText(IPAddress);
        Bundle bundle=new Bundle();
        bundle.putString("etText",IPAddress);
        fragmentManager=getSupportFragmentManager();
        FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
```

```
        StreamFragment fragment=new StreamFragment();
        fragment.setArguments(bundle);
        fragmentTransaction.add(R.id.streamFragment,fragment);
        fragmentTransaction.commit();
        btnStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                setMotorsTo0();
                getIPandPort();
                CMD="0";
                GyroscopeActivity.Soket_AsyncTask cmd_increase_servo = new
GyroscopeActivity.Soket_AsyncTask();
                cmd_increase_servo.execute();
            }
        });
        btnUp.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    setMotor_w(true);
                    getIPandPort();
                    CMD="W";
                    GyroscopeActivity.Soket_AsyncTask cmd_increase_servo =
new GyroscopeActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    setMotor_w(false);
                    getIPandPort();
                    CMD="w";
                    GyroscopeActivity.Soket_AsyncTask cmd_increase_servo =
new GyroscopeActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });

        btnDown.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN) {
                    setMotor_s(true);
                    getIPandPort();
                    CMD="S";
                    GyroscopeActivity.Soket_AsyncTask cmd_increase_servo =
new GyroscopeActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    setMotor_s(false);
                    getIPandPort();
                    CMD="s";
                    GyroscopeActivity.Soket_AsyncTask cmd_increase_servo =
new GyroscopeActivity.Soket_AsyncTask();
                    cmd_increase_servo.execute();
                }
                return true;
            }

        });
        //screenshot
```

```
if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.M&&checkSelfPermission(Manifest
.permission.WRITE_EXTERNAL_STORAGE)!= PackageManager.PERMISSION_GRANTED){
            requestPermissions(new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},1000);
        }
        else{
            //nista za sad
        }
        btnScreenshot.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                View
rootView=getWindow().getDecorView().findViewById(android.R.id.content);//slik
a cijeli app
                Bitmap bitmap=getScreenshot(rootView);
                store(bitmap,"ScreenShot-
"+java.text.SimpleDateFormat.getDateTimeInstance().format(new
Date())+".png");
            }
        });
    }
    public static Bitmap getScreenshot(View view){
        View screenView=view.getRootView();
        screenView.setDrawingCacheEnabled(true);
        Bitmap bitmap=Bitmap.createBitmap(screenView.getDrawingCache());
        screenView.setDrawingCacheEnabled(false);
        return bitmap;
    }
    //store metoda on the device
    public void store(Bitmap bm,String fileName){
        String dirPath=
Environment.getExternalStorageDirectory().getAbsolutePath()+"/MyFIles";
        File dir = new File(dirPath);
        if(!dir.exists()){
            dir.mkdirs();
        }
        File file=new File(dirPath,fileName);
        try{
            FileOutputStream fos=new FileOutputStream(file);
            bm.compress(Bitmap.CompressFormat.PNG,100,fos);
            fos.flush();
            fos.close();
            Toast.makeText(this,"Saved",Toast.LENGTH_SHORT).show();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            Toast.makeText(this,"Error!",Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    private void stopStream() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        fragment.stopStream();
        Toast.makeText(this, "Camera stream OFF", Toast.LENGTH_SHORT).show();
    }
```

```
    private void startStream() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        fragment.startStream();
        Toast.makeText(this,"Camera stream ON",Toast.LENGTH_SHORT).show();
    }

    private void onSensorValueChanged(int sv) {
        tvRotation.setText(Integer.toString(sv));
            if(sv>=4&& !isMotor_d() && !isMotor_a()){
                setMotor_d(true);
                getIPandPort();
                CMD="D";
                GyroscopeActivity.Soket_AsyncTask cmd_increase_servo = new
GyroscopeActivity.Soket_AsyncTask();
                cmd_increase_servo.execute();
            }else if (sv<=-4&&!isMotor_d() && !isMotor_a()){
                setMotor_a(true);
                getIPandPort();
                CMD="A";
                GyroscopeActivity.Soket_AsyncTask cmd_increase_servo = new
GyroscopeActivity.Soket_AsyncTask();
                cmd_increase_servo.execute();
            }else if (sv<3&&isMotor_d()){
                setMotor_d(false);
                getIPandPort();
                CMD="d";
                GyroscopeActivity.Soket_AsyncTask cmd_increase_servo = new
GyroscopeActivity.Soket_AsyncTask();
                cmd_increase_servo.execute();
            }
              else if (sv>-3&&isMotor_a()){
                setMotor_a(false);
                getIPandPort();
                CMD="a";
                GyroscopeActivity.Soket_AsyncTask cmd_increase_servo = new
GyroscopeActivity.Soket_AsyncTask();
                cmd_increase_servo.execute();
            }

}
    public void getIPandPort(){
        String IPandPort=IPAddress+":21567";
        Log.d("MYTEST","IP String "+IPandPort);
        String temp[]=IPandPort.split(":");
        wifiModuleIP=temp[0];
        wifiModulePort=Integer.valueOf(temp[1]);
        Log.d("MY TEST","IP:"+wifiModulePort);
        Log.d("MY TEST","PORT:"+wifiModulePort);

    }



    public class Soket_AsyncTask extends AsyncTask<Void,Void,Void> {
        Socket socket;

        @Override
        protected Void doInBackground(Void... params) {
            try{
```

```
                InetAddress
inetAddress=InetAddress.getByName(GyroscopeActivity.wifiModuleIP);
                socket=new
java.net.Socket(inetAddress,GyroscopeActivity.wifiModulePort);
                DataOutputStream dataOutputStream=new
DataOutputStream(socket.getOutputStream());
                dataOutputStream.writeBytes(CMD);
                dataOutputStream.close();
                socket.close();
            }catch (UnknownHostException e ){
                e.printStackTrace();
            }catch (IOException e){
                e.printStackTrace();
            }
            return null;
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        //startStream();

sensorManager.registerListener(gyroscopeEventListener,gyroscopeSensor,SensorM
anager.SENSOR_DELAY_FASTEST);
    }

    @Override
    protected void onPause() {
        super.onPause();
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        if(fragment.isStreamOn())stopStream();
        sensorManager.unregisterListener(gyroscopeEventListener);
    }

    @Override
    protected void onStop() {
        StreamFragment
fragment=(StreamFragment)getSupportFragmentManager().findFragmentById(R.id.st
reamFragment);
        if(fragment.isStreamOn())stopStream();
        super.onStop();
    }
}
```

**PhotoActvity.java**

```
package com.example.landcruiser;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
```

```
import android.widget.ImageView;

public class PhotoActivity extends AppCompatActivity {
private String path;
private ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_photo);
        path=getIntent().getStringExtra("path");
        imageView=(ImageView)findViewById(R.id.ivFSPhoto);
        Bitmap bm = BitmapFactory.decodeFile(path);
        imageView.setImageBitmap(bm);
    }
}
```

**GalleryActivity.java**

```
package com.example.landcruiser;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.widget.Toast;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class GalleryActivity extends AppCompatActivity {
    List<Cell> allFilesPath;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gallery);

        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&
checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE)!=
PackageManager.PERMISSION_GRANTED){
            requestPermissions(new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},1000);
        }
        else{
            showImages();
        }

    }
    private void showImages() {
        String path=
Environment.getExternalStorageDirectory().getAbsolutePath()+"/MyFIles";
        allFilesPath = new ArrayList<>();
```

```
        allFilesPath= listAllFiles(path);

        RecyclerView recyclerView=(RecyclerView)
findViewById(R.id.rvGallery);
        recyclerView.setHasFixedSize(true);
        RecyclerView.LayoutManager layoutManager=new
GridLayoutManager(getApplicationContext(),2);
        recyclerView.setLayoutManager(layoutManager);
        ArrayList<Cell>cells=prepareData();
        RVAdapter adapter= new RVAdapter(getApplicationContext(),cells);
        recyclerView.setAdapter(adapter);

    }
    private ArrayList<Cell>prepareData(){
        ArrayList<Cell>allImages=new ArrayList<>();
        for(Cell c :allFilesPath){
            Cell cell = new Cell();
            cell.setTitle(c.getTitle());
            cell.setPath(c.getPath());
            allImages.add(cell);
        }
        return allImages;
    }
    private List<Cell>listAllFiles(String pathName){
        List<Cell>allFiles = new ArrayList<>();
        File file =new File(pathName);
        File[] files =file.listFiles();
        if (files!= null){
            for (File f:files){
                Cell cell=new Cell();
                cell.setTitle(f.getName());
                cell.setPath(f.getAbsolutePath());
                allFiles.add(cell);
            }
        }
        return allFiles;
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        if(requestCode==1000){
            if(grantResults[0]==PackageManager.PERMISSION_GRANTED){
                showImages();
            }
            else{
                Toast.makeText(this, "Permission denied",
Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }
}
```

**RVAdapter.java**

```
package com.example.landcruiser;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.io.File;
import java.util.ArrayList;

public class RVAdapter extends RecyclerView.Adapter<RVAdapter.ViewHolder> {
    private ArrayList<Cell> galleryList;
    private Context context;

    public RVAdapter(Context context, ArrayList<Cell> galleryList){
        this.galleryList=galleryList;
        this.context=context;
    }

    @NonNull
    @Override
    public RVAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int viewType) {
        View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.cell,viewGroup,f
alse);
        return new RVAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder viewHolder, final int
position) {
        viewHolder.img.setScaleType(ImageView.ScaleType.CENTER_CROP);
        setImageFromPath(galleryList.get(position).getPath(),viewHolder.img);
        viewHolder.img.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(context,
""+galleryList.get(position).getTitle(), Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(view.getContext(),
PhotoActivity.class);
                intent.putExtra("path",galleryList.get(position).getPath());
                view.getContext().startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        return galleryList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder{
        private ImageView img;
```

```
        public ViewHolder(View view){
            super(view);
            img = (ImageView)view.findViewById(R.id.img);
        }
    }
    private void setImageFromPath(String path,ImageView image){
        File imgFile = new File(path);
        if (imgFile.exists()){
            Bitmap
myBitmap=ImageHelper.decodeSampledBitmapFromPath(imgFile.getAbsolutePath(),20
0,200);
            image.setImageBitmap(myBitmap);
        }
    }
}
```