Izrada aplikacije za pametni telefon kao mjerni instrument

Odobašić, Tomislav

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

Preddiplomski sveučilišni studij

IZRADA APLIKACIJE ZA PAMETNI TELEFON KAO MJERNI INSTRUMENT

Završni rad

Tomislav Odobašić

Osijek, 2021.

Sadržaj

1. UVOD

Tema ovog završnog rada je izrada aplikacije za pametni telefon koja će omogućiti njegovo korištenje kao mjernog instrumenta, tako što će pomoću određenih senzora dostupnih u većini pametnih telefona današnjice mjeriti željene veličine potom ih obrađivati i grafički prikazivati. Aplikacija bi se koristila za potrebe eksperimenata iz fizike, kao i za laboratorijske vježbe iz kolegija Fizika. Cilj rada je omogućiti provođenje eksperimenata bez ikakvih drugih mjernih instrumenata osim osobnog pametnog telefona. Također je jako praktično za nastavu na daljinu, gdje bi studenti mogli eksperimente provoditi od kuće.

Aplikacija koristi senzore implementirane u pametni telefon, koji omogućavaju mjerenje potrebnih podataka za provođenje eksperimenta. Očitani podaci se šalju na obradu aplikaciji koja je ostvarena u programskom jeziku Java, u Android Studio razvojnom okruženju.

U nastavku završnog rada slijedi opis po poglavljima. U drugom poglavlju opisana su slična rješenja odnosno slične aplikacije. U trećem poglavlju nalazi se opis korištenih tehnologija uključujući hardverski i softverski dio. U četvrtom poglavlju opisuje se kod aplikacije, a u petom poglavlju nalazi se opis dijelova od kojih se aplikacija sastoji. U šestom poglavlju je opisana izvedba eksperimenta pomoću aplikacije, te usporedba dobivenih rezultata sa rezultatima sličnih aplikacija. Na kraju rada nalazi se zaključak.

1.1. Zadatak završnog rada

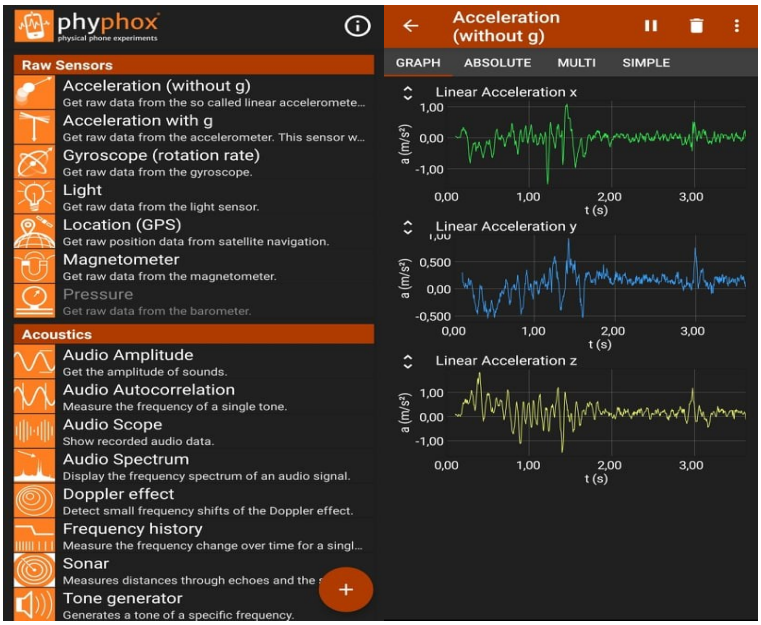
U radu je potrebno navesti i opisati senzore koje ima većina pametnih telefona. Potrebno je izraditi aplikaciju koja će moći koristiti dostupne senzore za provođenje eksperimentalnih mjerenja iz područja fizike. Navesti korake izrade aplikacije i dati kratak opis načina korištenja aplikacije. U aplikaciji napraviti predloške za neka eksperimentalna mjerenja iz fizike na osnovu predložaka za laboratorijske vježbe iz fizike.

2. PREGLED SLIČNIH APLIKACIJA

U ovom poglavlju navedena su slična rješenja tj. slične aplikacije. Na današnjem tržištu postoje razne aplikacije pomoću kojih pametni telefon dobiva mogućnosti rješavanja nekog problema u smislu mjerenja, odnosno da ga se koristi kao funkcionalni mjerni instrument. Osim kao mjerni instrument za provođenje eksperimenata i mjerenja iz područja fizike, neke od ovih aplikacija omogućavaju razna druga mjerenja, poput mjerenja na kojoj nadmorskoj visini se trenutno nalazi uređaj koristeći globalni položajni sustav (GPS, engl. *global positioning system*), očitavanja sa karte i unutarnji barometar pametnog telefona, zatim mjerenja duljine nekih objekata i prostora i razne druge.

2.1. Phyphox

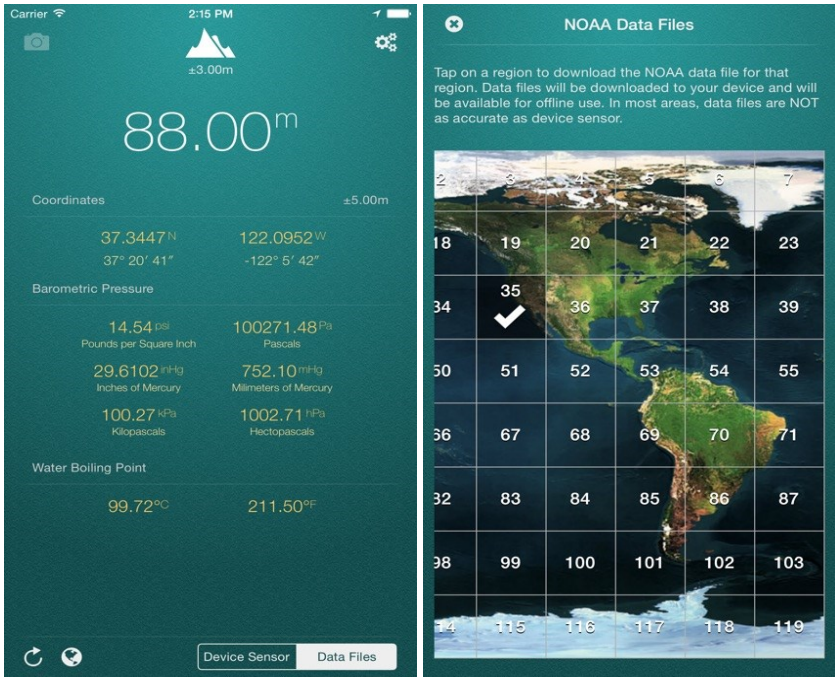
Aplikacija Phyphox omogućuje upotrebu raznih senzora dostupnih u pametnom telefonu za provođenje eksperimenata. Omogućuje izvoz podataka u mnoge uobičajene formate da bi se potom mogli analizirati i obrađivati u nekom od softvera za obradu podataka. Phyphox također omogućuje upravljanje eksperimentom iz bilo kojeg *web* preglednika. Na primjer, phyphox se može kontrolirati sa laptopa i podatke preuzeti izravno na radnu površinu. Još jedna jako zanimljiva mogućnost koju phyphox nudi je stvaranje vlastitog eksperimenta, ukoliko se ne nalazi na listi ponuđenih unutar same aplikacije. [1] Prikaz izgleda korisničkog sučelja je na slici 2.1.



Slika 2.1. Sučelje aplikacije Phyphox (preuzeto s phyphox.org)

2.2. My Altitude

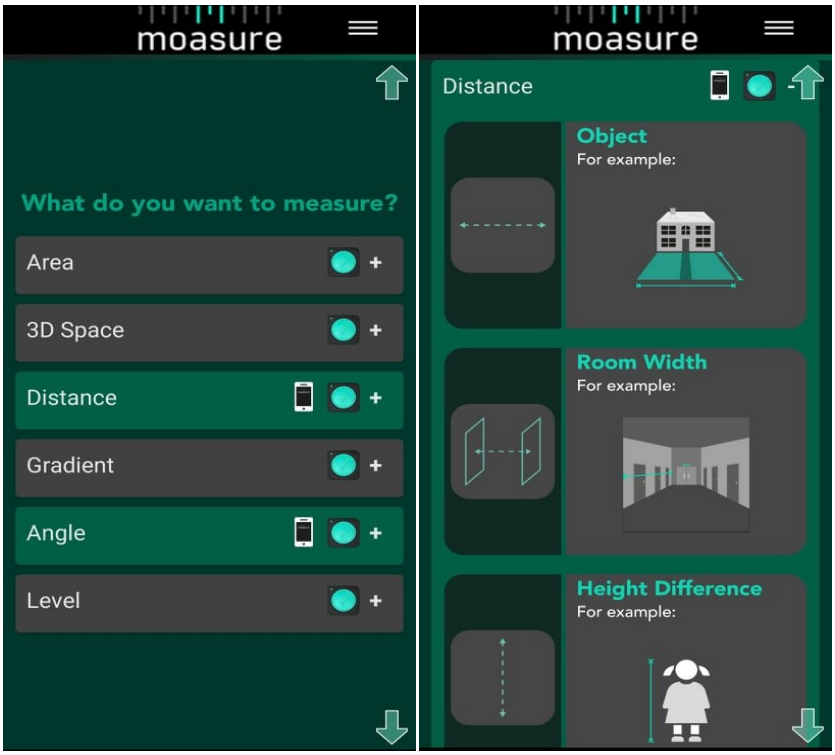
My Altitude je aplikacija namijenjena samo za pametne telefone koji koriste operacijski sustav namijenjen Apple uređajima, odnosno iOS, a ne i za one koji koriste Android operacijski sustav. My altitude koristi GPS signale za točno određivanje mjesta na kojem se trenutno nalazi uređaj, određivanje nadmorske visine, barometarskog tlaka i točke vrenja vode. Za ovu aplikaciju nije potrebna internetska veza, te je najučinkovitija i najbolje radi na otvorenom jer je u zatvorenim područjima snaga i preciznost signala slaba, naime u zatvorenim prostorima GPS signal će blokirati ili reflektirati zidovi . Prema zadanim postavkama koristi GPS uređaja za određivanje nadmorske visine, ali također ima i mogućnost korištenja podatkovnih datoteka Nacionalne Oceanske i Atmosferske Administracije (NOAA) za dobivanje nadmorske visine mjesta uređaja. Naime NOAA svakodnevno prikuplja podatke s meteoroloških stanica, radara, satelita, brodova, bova i senzora, zatim koristeći javno dostupne izvore omogućava se korištenje tih podataka. Aplikacija također omogućuje dobivanje slike trenutne lokacije uređaja i spremanje u foto album. Slika će biti označena koordinatama trenutnog mjesta, zajedno s nadmorskom visinom i lokalnim datumom i vremenom. [2] Prikaz izgleda korisničkog sučelja aplikacije je na slici 2.2.



Slika 2.2. Sučelje aplikacije My Altitude (preuzeto s bejbej.ca)

2.3. Moasure

Moasure je aplikacija praktična i jednostavna za upotrebu, predstavlja više pomagala u jednoj aplikaciji. Jednostavnim premještanjem pametnog telefona s jedne točke na drugu udaljenu točku, Moasure prikazuje udaljenost bilo da se radi o udaljenosti, visinsku razliku o visini ili kut bilo da se radi o kutnom pomaku. Moasure koristi istu tehnologiju koja se nalazi u sustavima za navođenje svemirskih raketa, akcelerometre i žiroskope pametnog telefona pomoću kojih precizno izračunava koliko je pomaknut. Moasure aplikacija ima mogućnost mjerenja duljine, širine, visine predmeta, kao i unutarnje dimenzije neke prostorije. [3] Na slici 2.3. prikazan je izgled korisničkog sučelja aplikacije Moasure.

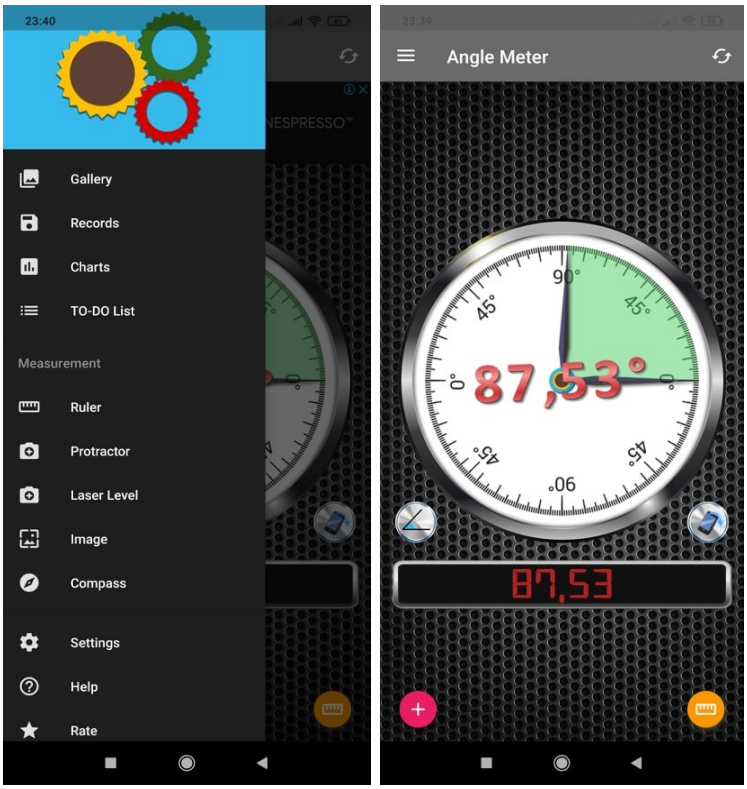


Slika 2.3. Sučelje aplikacije Moasure (preuzeto s moasure.com)

2.4. Angle Meter

Angle Meter je aplikacija za mjerenje nagiba i kutova, dizajnirana je i dostupna samo za korisnike Androida. Aplikacija koristi lučnu tangentu gravitacije između dvije osi kako bi došla do točnih rezultata. Nagib ili kut može se mjeriti i na površinama koje nisu paralelne s tlom, u ovu svrhu aplikacija koristi način relativnog kuta. Moguće je i provjeriti kut na slici koristeći ekran za mjerenje slike. Aplikacija Angle Meter nudi dvije osi orijentacije između kojih se može birati pritiskom na određenu tipku. Aplikacija još nudi mogućnost spremanja zapisa u bazu podataka i

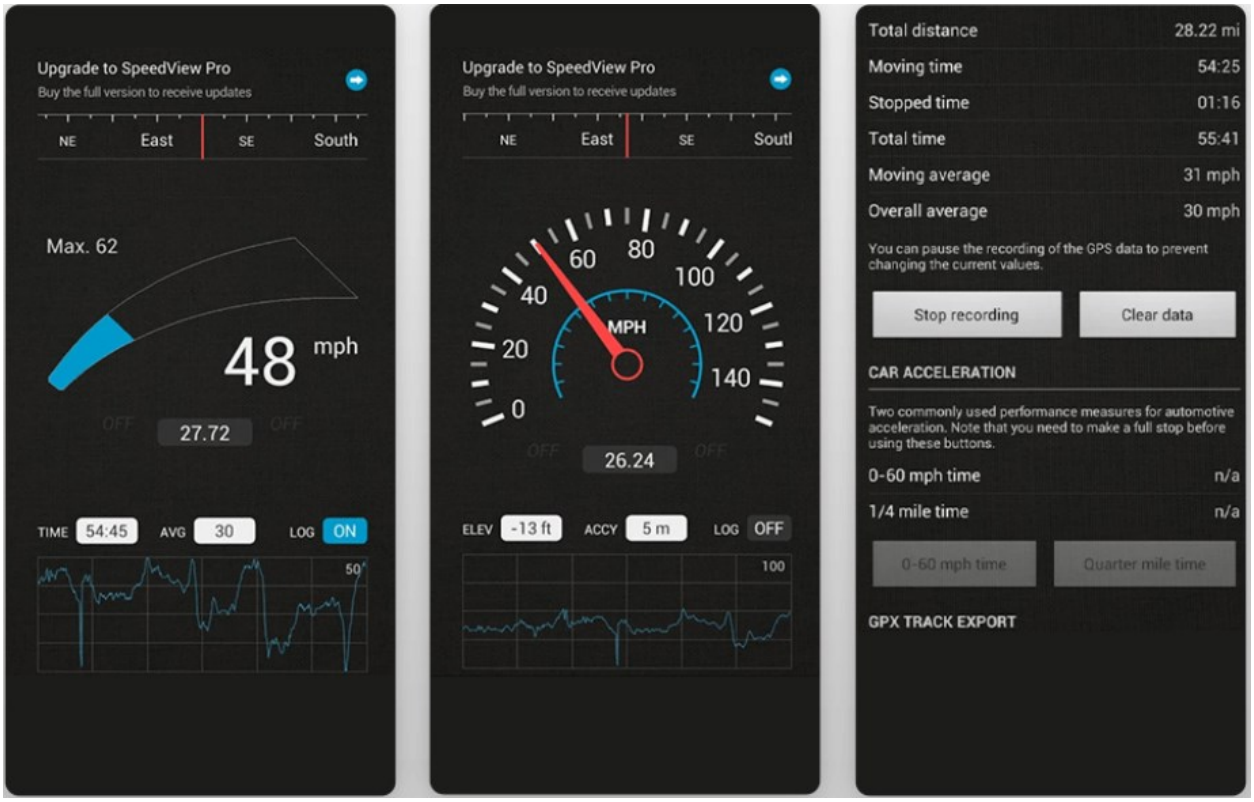
prikazivanje podataka na grafikonu. Na slici 2.4 prikazano je korisničko sučelje aplikacije Angle meter.



Slika 2.4. Sučelje aplikacije Angle Meter (preuzeto s play.google.com)

2.5. SpeedView

Aplikacija SpeedView koristi ugrađeni GPS sustav pametnog telefona za prikaz trenutne maksimalne i prosječne brzine, kao i smjera, ukupne udaljenosti i prijeđenog vremena. Aplikacija je pogodna trčanje, vožnju automobilom, biciklizam ili pješačenje. SpeedView nudi mogućnost postavljanja ograničenja brzine za tri različite vrste cesta tako da obavijesti korisnika vizualno ili zvučno ukoliko ide prebrzo. Aplikacija pokazuje trenutni smjer putovanja, a dostupan je i način rada kompasa. [5] Na slici 2.5. prikazano je korisničko sučelje aplikacije SpeedView.



Slika 2.5. Sučelje aplikacije SpeedView (preuzeto s codesector.com)

3. OPIS KORIŠTENIH TEHNOLOGIJA

U ovom poglavlju opisane su tehnologije koje su korištene u izradi rada. Većina android uređaja ima ugrađene razne senzore koji mogu mjeriti kretanje, orijentaciju, kut, brzinu, ubrzanje i ostale različite uvjete okoliša. Senzori poput ovih imaju mogućnost pružanja neobrađenih podataka s velikom točnošću i preciznošću, korisni su za praćenje i zapažanje promjena u okolišu u blizini uređaja. [6] Veliki broj današnjih pametnih telefona posjeduje sljedeći set ugrađenih senzora: akcelerometar, žiroskop, magnetometar, senzor blizine, senzor svjetlosti i termometar. Nabrojana je većina hardverskih dijelova korištenih u rješavanju zadatka ovog rada, a i drugih zadataka sličnih njemu. Iza svakog hardverskog sklopovlja naravno stoji i programska podrška ili softver, pomoću kojeg se upravlja hardverom na željeni način.

3.1. Hardver

3.1.1. Akcelerometar

Akcelerometar je jedan od najčešćih senzora koji se mogu naći u pametnom telefonu. Akcelerometar je senzor koji proučava je li pametni telefon u pokretu ili je u idealnom stacionarnom stanju. Izračunava orijentaciju pametnog telefona unutar tri osi. Primjer akcelerometra je prebacivanje orijentacije zaslona pametnog telefona između portreta i pejzaža. Bilo koja aplikacija, značajka pametnog telefona, igra koja se aktivira ili reagira na naginjanje pametnog telefona ovisi o podacima akcelerometra.

3.1.2. Žiroskop

Žiroskop je senzor koji daje detaljne informacije orijentacije. Međutim, za razliku od akcelerometra koji mjeri linearno ubrzanje, žiroskop izračunava kutnu brzinu rotacije. Žiroskop i akcelerometar se ustvari međusobno nadopunjuju kako bi mogli pružiti precizne podatke. U raznim izvedbama često su akcelerometar i žiroskop zajedno prisutni na istom čipu ili su kombinirani zajedno. Neki pametni telefoni za proračune koriste samo jedan od ova dva senzora.

3.1.3. Magnetometar

Magnetometar je senzor koji se koristi za otkrivanje magnetskog polja i pružanje informacija koje definiraju orijentaciju pametnog telefona s obzirom na polje. Sve aplikacije koje su neke vrste kompasa koriste senzor magnetometar kako bi otkrili fizički smjer na karti. Upravo iz ovog razloga se digitalna karta automatski okreće u istom smjeru kad se zauzme zavoj tijekom vožnje.

3.1.4. Senzor blizine

Senzor blizine je još jedan uobičajeni senzor prisutan u većini pametnih telefona. Senzor blizine dolazi s infracrvenom svjetlećom diodom (LED, engl. *light emitting diode*) i detektorom infracrvenog zračenja (IR, engl. *infrared radiation*). Senzor blizine zrači zrakom infracrvene svjetlosti koja se reflektira od obližnjeg objekta i taj odraz onda registrira IR skener. Senzor blizine se na pametnom telefonu može pronaći pored ulaza za slušalice. Svrha uključivanja senzora blizine u pametni telefon je otkrivanje kada se koristi za pozivanje. Naime čim slušalica pametnog telefona dodirne tijelo korisnika, senzor blizine to prenosi procesoru (CPU, engl. *central processing unit*) kako bi se isključio zaslon. Na taj način se isto tako izbjegavaju slučajni dodiri i štedi se baterija.

3.1.5. Senzor svjetlosti

Senzor svjetlosti je senzor prisutan kako bi se izmjerila svjetlina okolišnog svjetla. Podaci sa senzora se šalju u pametni telefon gdje se algoritmu operativnog sustava onda podešava osvijetljenost zaslona. Na primjer ako je mjesto tamno, svjetlina zaslona se automatski prigušuje, tako da ne utječe na oči korisnika. S druge strane, pod izravnom sunčevom svjetlošću razina svjetline pametnog telefona je najveća. Danas mnogi proizvođači pametnih telefona koriste napredni svjetlosni senzor koji ima sposobnost razlikovanja plave, crvene, zelene i bijele boje svjetla.

3.1.6. Termometar

Termometar posjeduju gotovo svi današnji pametni telefoni. Termometar je neophodan jer štiti pametni telefon od oštećenja zbog prekomjernog zagrijavanja, što danas i nije tako rijetka pojava. Senzor termometra mjeri temperaturu unutar pametnog telefona i njegove jedinice napajanja. Prilikom naznaka da će uređaj prijeći ograničenje temperature, isključuje se ili gasi procesor dok se uređaj ne ohladi.

3.2. Softver

3.2.1. Operacijski sustav Android

Android je mobilni operativni sustav baziran na softveru otvorenog koda i modificiranoj verziji Linux jezgre. Android je prvenstveno dizajniran za mobilne uređaje koji su osjetljivi na dodir kao što su pametni telefoni, tableti i slično. Android je operacijski sustav koji je razvio projektni tim poznat kao OHA (*Open Handset Alliance*), a komercijalno ga sponzorira Google. Android operacijski sustav predstavljen je u studenom 2007. godine, a prvi komercijalni Android uređaj, HTC Dream, pokrenut je u rujnu 2008. Android je besplatan, a njegov izvorni kod poznat je kao

Android Open Source Project (AOSP), koji je primarno licenciran pod *Apache* licencom. [7] Međutim, većina Android uređaja isporučuje se s unaprijed instaliranim dodatnim vlasničkim softverom, najčešći od njih je Google Mobile Services (GMS) koji unutar svog paketa uključuje osnovne aplikacije kao što su Google Chrome, Google Play i povezane razvojne platforme Google Play Services.

3.2.2. Razvojno okruženje Android Studio

Android Studio je službeno integrirano razvojno okruženje (IDE, engl. *Integrated Development Environment*) za Googleov operacijski sustav Android, izgrađeno je na već postojećem softveru pod imenom IntelliJ IDEA od češke tvrtke za razvoj softvera JetBrains, i dizajnirano je isključivo za razvoj Androida. Ovo okruženje dostupno je za preuzimanje na operativnim sustavima kao što su Windows, macOS i Linux. Android Studio postaje zamjena za Eclipse Android Development Tools (E-ADT) kao primarni IDE za razvoj izvornih Android aplikacija. Razvojno okruženje Android Studio bilo je najavljeno 16. svibnja 2013. na Google I/O konferenciji. U fazi pregleda ranog pristupa počevši od verzije 0.1 bio je u svibnju 2013. godine, a zatim ulazi u beta fazu počevši od verzije 0.8 u lipnju 2014. godine. Prva stabilna gradnja Android Studia je objavljena u prosincu 2014. sa verzijom 1.0. Prevladavajući programski jezik korišten unutar okruženja Android Studio je objektno orijentirani programski jezik po imenu Java, no 7. svibnja 2019. programski jezik Kotlin je zamijenio Javu kao Googleov preferirani jezik za razvoj Android aplikacija. Iako naravno Java je i dalje podržana, kao i programski jezik C++.

3.2.3. Programski jezik Java

Java je objektno orijentirani programski jezik zasnovan na klasama, koji je dizajniran tako da ima što manje ovisnosti o samoj implementaciji. Java je programski jezik opće namjene namijenjen programerima aplikacija koji im omogućuje da jednom napisani program koriste i pokreću bilo gdje, odnosno omogućuje im korištenje principa „piši jednom, pokreni bilo gdje“ (engl. *write once, run anywhere*), što znači da se prevedeni Java kod može izvoditi na svim ostalim platformama koje podržavaju Javu bez potrebe za ponovnim prevođenjem koda. [8] Java programi se obično prevode u Java bajt-kodu koji se mogu izvoditi na bilo kojem Java virtualnom stroju (JVM, engl. *Java Virtual Machine*) bez obzira na temeljnu arhitekturu računala. Sama sintaksa Jave slična je programskim jezicima C i C++. Vrijeme izvođenja Jave pruža dinamičke mogućnosti koje obično nisu dostupne u tradicionalnim sastavljenim jezicima. Od 2019. godine Java postaje jedan od najpopularnijih programskih jezika koji se koriste sudeći po GitHub-u, posebno za klijent-poslužitelj *web* aplikacije, s prijavljenih 9 miliona programera. Java programski jezik

izvorno je razvio James Gosling u tvrtki Sun Microsystems (koju je kasnije kupio Oracle), a objavljena je 1995. godine kao osnovna komponenta Sun Microsystems Java platforme. Izvorne i referentne implementacijske Java kompajlere, virtualne strojeve i knjižnice klasa izvorno je izdao Sun Microsystems pod zaštićenim licencama. Od svibnja 2007. godine, u skladu sa specifikacijama procesa Java Community, Sun Microsystems je licencirao većinu svojih Java tehnologija pod licencom samo za GPL-2.0. Oracle nudi vlastiti HotSpot Java virtualni stroj, međutim službena referentna implementacija je OpenJDK JVM koji je besplatni softver otvorenog koda i koristi ga većina programera, a zadani JVM je za gotovo sve Linux distribucije. Od ožujka 2021. godine najnovija verzija Jave je Java 16, s Javom 11. Glavna načela Jave su: prijenosnost, robusnost, neovisnost o platformi, visoke performanse i korištenje više niti (engl. *thread*). Time magazin je Javu nazvao jednim od deset najboljih proizvoda 1995. godine.

4. OPIS FUNKCIONALNOSTI

U ovom poglavlju opisane su funkcionalnosti aplikacije. Ukratko su opisane i prikazane glavne funkcije odgovorne za ispravan rad aplikacije.

4.1. Kod

Aktivnost je klasa koja ima mogućnost komuniciranja sa korisnikom. Svaka od aktivnosti se u Android Studiu prikazuje kao zasebna klasa u kojoj se određuje njihova funkcionalnost. Aplikacije najčešće imaju više aktivnosti, a uobičajeno da se jedna aktivnost, odnosno ona koja se prikaže korisniku prilikom pokretanja aplikacije, označi kao glavna aktivnost. Aktivnosti se stoga mogu međusobno pokretati kako bi se odradili različiti zadaci.

Prva aktivnost s kojom se korisnik susreće je Menu aktivnost. Svaka aktivnost sastavljena je od dva dijela, izgleda (engl. *layout*) i izvornog koda pomoću kojeg se definira funkcionalnost komponenata koje se nalaze na *layout*-u. Sve *layout* datoteke aplikacije pisane su u *eXtensible Markup Language* (XML) . XML nije klasični programski jezik nego jezik za označavanje koji je nastao kao potreba za stvaranjem jezika koji će biti čitljiv i razumljiv ljudima, a ujedno i računalnim programima.

Metoda *onCreate* je gdje se većina aktivnosti inicijalizira. Najvažnije ovdje je da će se pozvati metoda *setContentView* s *layout* resursom koji definira korisničko sučelje aktivnosti, i pomoću metode *findViewById* dohvatiti *widget*-e u tom korisničkom sučelju s kojima treba programski komunicirati. Unutar klase Menu inicijalizira se metoda *openExperiment* koja je zadužena za prelazak u sljedeću aktivnost. Radi tako da provjerava tekst tipke, i na taj način zaključi koja od dvije ponuđene tipke je pritisnuta. Naime ako je tekst tipke „Pendulum“ otvara se Pendulum aktivnost, a ako je tekst tipke „Spring“ Spring aktivnost (**Slika 4.1.**)

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

public void openExperiment(View view){

    String button_text;
    button_text = ((Button) view).getText().toString();
    if(button_text.equals("Pendulum"))
    {
        Intent intent = new Intent( packageContext: this, Pendulum.class);
            startActivity(intent);
    }
    else if (button_text.equals("Spring"))
    {
        Intent intent = new Intent( packageContext: this, Spring.class);
        startActivity(intent);
    }
}
```

Slika 4.1. Menu aktivnost

Komunikaciju sa senzorima pametnog telefona omogućuju klase *SensorManager* i *Sensor*. *SensorManager* klasa se koristi za stvaranje instance usluge senzora. Klasa *SensorManager* nudi razne metode za pristup i popis senzora, registraciju i poništavanje registracije slušatelja senzorskih događaja i dobivanje informacija o orijentaciji. Također nudi nekoliko konstanti senzora koje se koriste za izvješćivanje o točnosti senzora, postavljanje brzine prikupljanja podataka i kalibriranje senzora. *Sensor* klasa se koristi za stvaranje instance određenog senzora, ovisno o senzoru koji će se koristiti. Nudi razne metode koje omogućuju utvrđivanje mogućnosti senzora. (**Slika 4.2.**)

```
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mAccelero = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
sensorManager.registerListener( listener: Pendulum.this, mAccelero, SensorManager.SENSOR_DELAY_NORMAL);
```

Slika 4.2. Stvaranje instance usluge senzora i dohvaćanje senzora potrebnog za eksperiment

Metoda pomoću koje se dohvaćaju vrijednosti koje senzor očitava je metoda *onSensorChanged*. Rad navedene metode ostvaren je pomoću globalne varijable *readData* koja je *boolean* tipa podataka, čija je inicijalna vrijednost *false*. Metoda radi na način da konstantno provjerava trenutno stanje varijable *readData*, te u trenutku u kojem prepozna da je stanje varijable *true* izvodi

se blok naredbi. Unutar bloka naredbi varijabla *readData* postavlja se ponovno na *false*, te se trenutna vrijednost koju je senzor dohvatio u tom trenutku sprema u listu *Angles*, i ponovno se čeka određeni period sve dok varijabla *readData* ponovno ne postane *true*. (**Slika 4.3.**)

```
@Override
public void onSensorChanged(SensorEvent event) {

    Sensor sensor = event.sensor;
    if(sensor.getType() == Sensor.TYPE_ORIENTATION){
        gyroValue.setText("Angle: " + String.format("%.0f", event.values[2]));

        if(readData) {

            readData = false;

            Angles.add(event.values[2]);
        }
    }
}
```

Slika 4.3. *onSensorChanged* metoda

Većina ostatka funkcionalnosti aplikacije se nalazi unutar metode *startlisting*, koja se poziva pritiskom na gumb *Start*. Prvo što metoda obavlja je provjera je li u *EditText* unesen željeni broj u sekundama, koji obilježava koliko dugo se eksperiment želi promatrati, te ako nije izbacuje se *Toast* poruka (**Slika 4.4.**). Sljedeće o čemu metoda brine je to da se vrijednost koju senzor očitava dodaje u listu svakih 25 milisekundi, što je jednako čak 40 uzoraka u sekundi, zajedno sa pripadajućim vremenom trenutka u kojem je vrijednost dodana. To radi pomoću metode *currentTimeMillis()* klase *System*, koja svojim pozivom vraća proteklo vrijeme od 1. siječnja 1970. Metoda *currentTimeMillis()* se poziva pritiskom na gumb *Start*, te ponovno unutar metode *schedule*, koja se poziva svakih 25 milisekundi. Metoda *schedule* postavlja varijablu *readData* u *true*, poziva metodu *currentTimeMillis()*, oduzima trenutnu vrijednost koju vraća metoda *currentTimeMillis()* od vrijednosti te metode koju je vratila kada je pritisnut gumb *Start*, dijeli tu vrijednost sa 1000 zbog pretvorbe u sekunde, jer metoda *currentTimeMillis()* vraća vrijednost u milisekundama i na kraju tu vrijednost sprema u listu *MyTime*. Ovaj proces se ponavlja svakih 25 milisekundi sve dok vrijeme ne dostigne vrijednost koju je korisnik prethodno unio (**Slika 4.4.**).

```
public void startlisting(View view) {

    if (custom_time.getText().toString().trim().length() <= 0) {

        Toast.makeText( context: Pendulum.this,  text: "Please fill in your wanted time period", Toast.LENGTH_SHORT).show();

    } else {

        long tStart = System.currentTimeMillis();

        Timer timer = new Timer();
        timer.schedule(() -> {

                readData = true;

                long tEnd = System.currentTimeMillis();
                long tDelta = tEnd - tStart;
                double elapsedSeconds = tDelta / 1000.0;

                MyTime.add((float) elapsedSeconds);

        }, delay: 0,  period: 25);
```

Slika 4.4. Dio *startlisting* metode koji prikazuje rad metode *schedule*

CountDownTimer je klasa koja prima dva argumenta, gdje se i prvi i drugi argument predaju u milisekundama, pomoću kojih mjerač vremena mjeri vrijeme do određenog trenutka u budućnosti. Nakon što je mjerač vremena dostigao vrijednost prvog argumenta, poziva se *onFinish()* metoda, unutar koje se zaustavlja rad senzora, vrijednost varijable *readData* postavlja na *false*, te se pomoću *for* petlje dohvaćaju vrijednosti spremljene u liste *Angles* i *MyTime*. Zatim se te vrijednosti pomoću metoda *appendData* i *addSeries* ucrtavaju na koordinatni sustav kako bi se dobio graf koji prikazuje vrijednosti kuta u ovisnosti o vremenu. (**Slika 4.5.**).

```
new CountDownTimer( millisInFuture: actualcustomTime + 50,  countDownInterval: 1000) {
    double x=0,y;
    float maxAvalue=0;
    public void onTick(long millisUntilFinished) {
    }

    public void onFinish() {
        sensorManager.unregisterListener(Pendulum.this);
        readData=false;
        timer.cancel();
        for(int i=1;i<Angles.size();i++){

            listangles.setText(listangles.getText() + "Angle: " + String.format("%.0f", Angles.get(i)) + "        " + "Time: " + String.format("%.3f", MyTime.get(i)) + "\n");
            x = MyTime.get(i);
            y = Angles.get(i);
            seriesPendulum.appendData(new DataPoint(x,y),  scrollToEnd: true,  maxDataPoints: 999999999);

            if( Angles.get(i) > maxAvalue )
            {
                maxAvalue = Angles.get(i);
            }
        }

        String tempmaxA = Float.toString(Float.parseFloat(String.format("%.0f", maxAvalue)));
        maxAnTV.setText("Theta: " + tempmaxA);

        graph.addSeries(seriesPendulum);
        timerTask.cancel();
    }
}.start();
```

Slika 4.5. Rad sa *CountDownTimer* klasom i *onFinish()* metodom

Metoda *export*, koja se poziva pritiskom na gumb *Save* omogućuje izvoz prikupljenih podataka sa senzora i podataka o vremenu u datoteku tipa „.csv“ (*Comma-separated values)*. *Comma-separated values* su datoteke unutar kojih su sve vrijednosti odvojene zarezom, te su pogodne za rad sa programima koji služe za obradu podataka poput *Microsoft Excel*-a. Pritiskom na gumb *Save* kreira se datoteka u unutarnjoj memoriji pametnog telefona u kojoj se nalaze prikupljene vrijednosti kuta i vremena. U unutarnjoj memoriji stvara se datoteka „angles.csv“. Prilikom stvaranja nove datoteke *while* petlja provjerava postoji li već datoteka pod tim imenom, te ako postoji, pomoću brojača stvara datoteku „angles0.csv“, zatim „angles1.csv“ itd. Na ovaj način je omogućeno spremanje više mjerenja odjednom.

```
public void export(View view) throws IOException
{

    int num = 0;
    String filename = "angles" + ".csv";

    File output = new File(getApplicationContext().getExternalFilesDir( type: null),filename);

        while(output.exists()){
            filename = "angles" + (num++) + ".csv";
            output = new File(getApplicationContext().getExternalFilesDir( type: null),filename);
        }

         try {
      FileOutputStream fileout = new FileOutputStream(output.getAbsolutePath());
      OutputStreamWriter outputWriter = new OutputStreamWriter(fileout);

      for(int i = 0; i < Angles.size(); i++) {
          outputWriter.write(("Angle" + "," +  Angles.get(i) + ","));
          outputWriter.write(("Time" + "," + MyTime.get(i).toString() + "\n"));
      }
              Toast.makeText(getBaseContext(), text: "File saved successfully!", Toast.LENGTH_SHORT).show();
              outputWriter.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
}
```

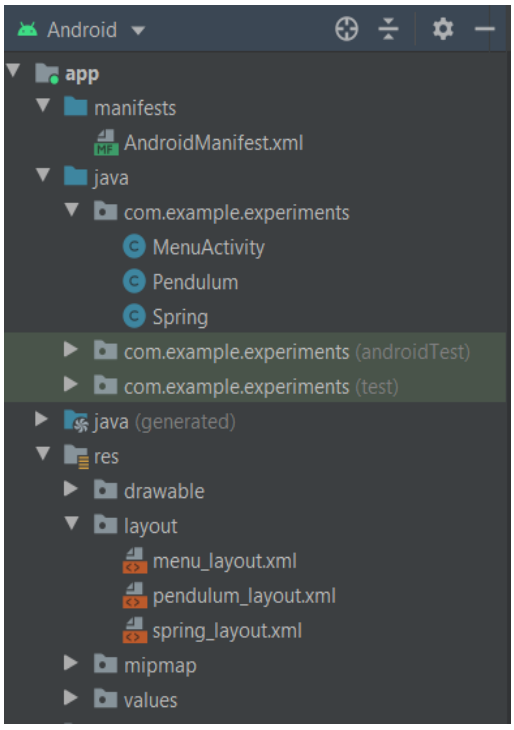
Slika 4.6. Izvoz podataka pomoću metode *export*

5. STRUKTURA APLIKACIJE

Kroz ovo poglavlje dati će se detaljan prikaz strukture aplikacije, to jest pregled dijelova od kojih je sastavljena aplikacija, kao i neki primjeri i opis korištenja same aplikacije. Glavni dijelovi korišteni za izradu ove aplikacije kao što je to spomenuto u prethodnim poglavljima su okruženje Android Studio i programski jezik Java. Android Studio projekt se sastoji od sljedećih ključnih datoteka: [9]

- java
- manifests
- res

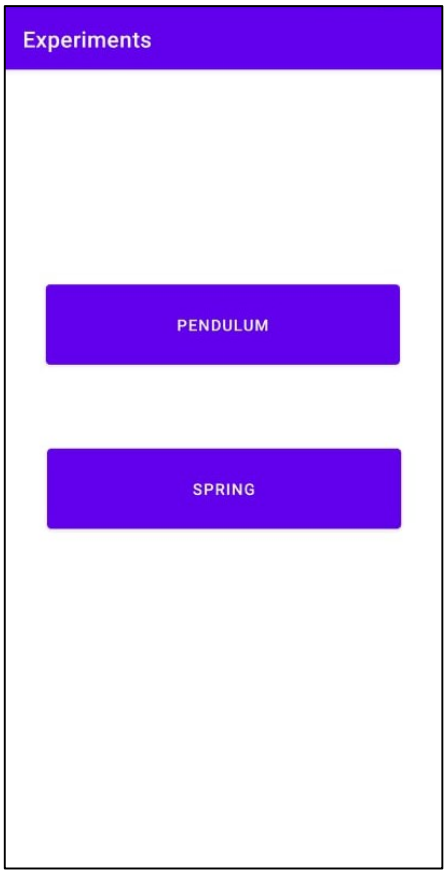
Java mapa sadrži sve mape Java i Kotlin izvornog koda koje se kreiraju tokom razvoja aplikacije. Bilo da se kreira projekt koristeći Java jezik ili Kotlin, novo kreirana mapa te klase će se prema zadanim postavkama kreirati unutar java mape. Nadalje manifests datoteka sadrži informacije o sustavu, poput verzije androida, stanja za datoteku Java i ostalih komponenata aplikacije. Mapa manifests djeluje kao posrednik između Android operacijskog sustava i aplikacije. Res mapa, skraćenica od *resource*, ili mapa resursa je najvažnija mapa jer sadrži sve izvore koji nisu kod, poput slika, XML *layout* datoteka, *user interface (UI)* nizova za Android aplikaciju. Na slici 5.1. prikazan je sadržaj aplikacije.



Slika 5.1. Sadržaj projekta Experiments

5.1. Menu aktivnost

Klasa je nacrt po kojem se stvaraju pojedinačni objekti. Aktivnost je Java klasa s kojom korisnik može komunicirati. Aplikacija je strukturirana u nekoliko aktivnosti koje se izmjenjuju ovisno o radnjama koje korisnik želi izvršiti. Svaka aplikacija najčešće ima jednu glavnu aktivnost koja se prikazuje prilikom pokretanja aplikacije. I unutar ove aplikacije, prilikom samog njenog pokretanja, korisnik nailazi na aktivnost *Menu* koja sadrži dvije tipke, gdje korisnik ima na izbor dvije opcije, ovisno o eksperimentu koji želi izvršiti. (**Slika 5.2.**)



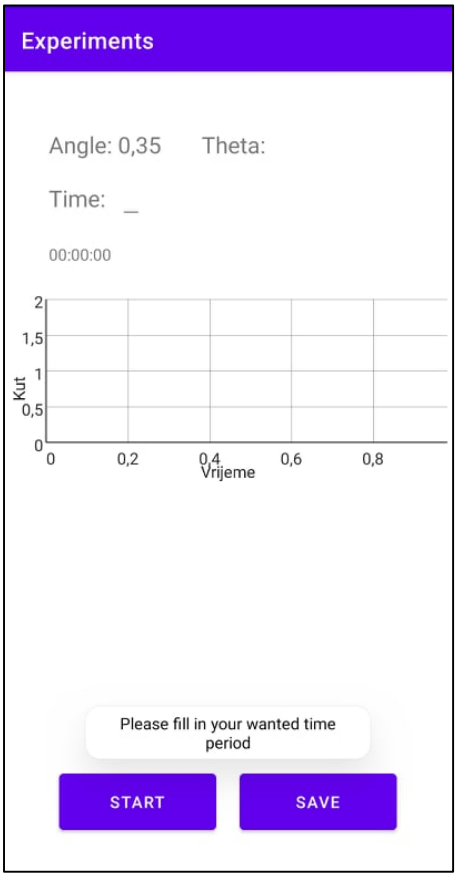
Slika 5.2. Aktivnosti Menu

Ukoliko korisnik želi izvršiti eksperiment njihala (engl. *Pendulum*) odlučuje se na tipku sa natpisom *Pendulum* i klikom na tipku aplikacija ga odvodi na Pendulum aktivnost. Ukoliko korisnik želi izvršiti drugi eksperiment, odnosno eksperiment sa oprugom (engl. *Spring*) odlučuje se za drugu tipku sa natpisom *Spring*, te kao i kod prve tipke, klikom na nju, aplikacija korisnika odvodi na sljedeću aktivnost odnosno aktivnost Spring.

5.2. Pendulum aktivnost

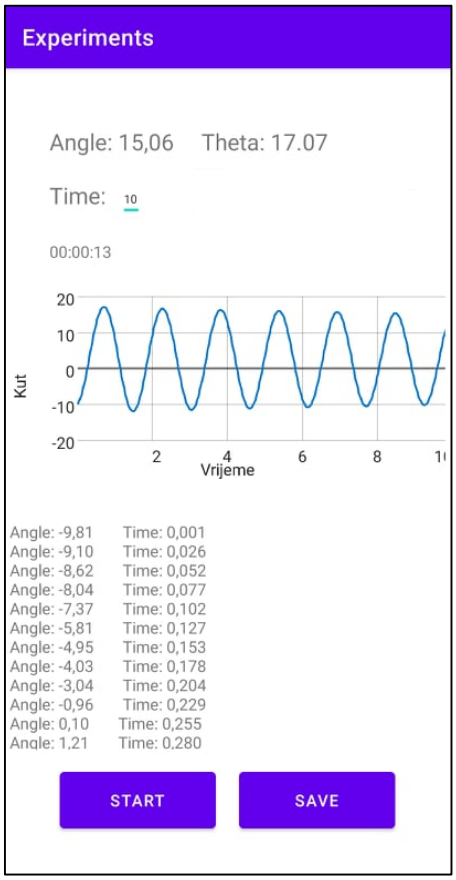
Pendulum aktivnost sastoji se od više komponenti. U aktivnosti postoji prikaz kuta u brojčanim vrijednostima, koje se dohvaćaju putem senzora pametnog telefona. Brojčana vrijednost prikazana na zaslonu pametnog telefona naravno ovisi o njegovom otklonu, odnosno od kuta pod kojim se pametni telefon trenutno nalazi.

Vrijednost koja se prikazuje u nastavku teksta *Angle* se mijenja sa promjenom kuta telefona. Korisnik zatim unosi proizvoljno vrijeme u naznačeno polje sa tekstom *Time*, vrijeme se unosi u sekundama i označava koliko dugo se vremenski želi promatrati eksperiment. Nakon što je korisnik unio željeno vrijeme u odgovarajuće polje predviđeno za to, pritiskom na tipku Start, mjerenje kreće i mjerač vremena kreće brojati do prethodno unesene vrijednosti. Ukoliko korisnik nekim slučajem preskoči korak unošenja vremena, aplikacija ga porukom obavještava da se vrati i da ispuni polje u kojem treba upisati vrijeme promatranja (**Slika 5.3.**).



Slika 5.3. Neispravan unos vremena

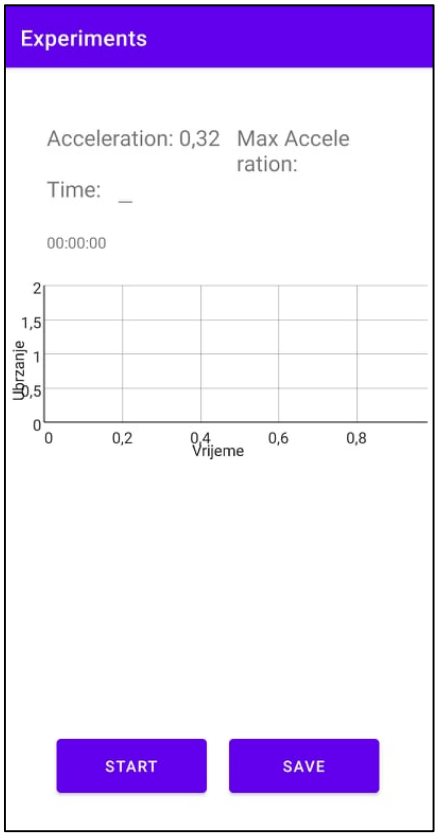
Korisnik bi neposredno prije pritiska gumba prvo trebao postaviti pametni telefon u odgovarajući položaj, odnosno pod kutom od kojeg želi početi promatrati eksperiment. Željeni početni kut moguće je točno odrediti jer dok korisnik otklanja pametni telefon na zaslonu uvijek ima prikazan trenutni kut. Nakon što je proteklo prethodno uneseno vrijeme, aplikacija će na zaslonu iscrtati graf sa vrijednostima kuta u ovisnosti o vremenu, a u predjelu teksta „Theta“ ispisati će se maksimalan kut otklona, odnosno kut pod kojim je pametni telefon ispušten na njihalu (**Slika 5.4.**).



Slika 5.4. Iscrtavanje grafa pomoću aplikacije Experiments

5.3. Spring aktivnost

Princip na koji radi Spring aktivnost, ili aktivnost za izvođenje eksperimenta sa oprugom, je jako sličan kao za Pendulum aktivnost. Razlikuju se u tome, što se u ovoj aktivnosti naravno, ne mjeri kut pomoću senzora, nego akceleracija. Samim time graf iscrtava ovisnost ubrzanja o vremenu. Na slici 5.7. prikazano je sučelje aktivnosti Spring.



Slika 5.7. Aktivnost Spring

6. ISTRAŽIVANJE PRIGUŠENOG TITRANJA KORIŠTENJEM APLIKACIJE

U ovom poglavlju opisan je eksperiment njihala, titrajni sustavi, te laboratorijski postav i promatranje eksperimenta pomoću izrađene aplikacije, zatim su rezultati eksperimenta uspoređeni sa rezultatima slične aplikacije.

U stvarnim titrajnim sustavima tijelo se nalazi pod utjecajem vanjskih sila, što znači da će se njegova amplituda s vremenom smanjivati, a takvo titranje, u kojem se amplituda s vremenom smanjuje nazivamo prigušenim titranjem. Kod stvarnih titrajnih sustava dio mehaničke energije titranja troši se na otpor zraka kad je riječ o njihalu. Jednadžba prigušenog titranja je sljedeća: [10]

$$\frac{d^2x}{dt^2} + 2\delta\frac{dx}{dt} + \omega_0 x = 0 \qquad (6\text{-}1)$$

gdje je ω_0 vlastita kružna frekvencija titranja neprigušenog oscilatora, a δ faktor prigušenja

U slučaju vrlo slabih prigušenja ($\delta < \omega_0$) rješenje jednadžbe (6-1) ima sljedeći oblik:

$$x(t) = Ae^{-\delta t}\sin(\omega t + \varphi_0) \qquad (6\text{-}2)$$

gdje je ω kružna frekvencija prigušenog titranja.

Amplituda titranja opada eksponencijalno s vremenom, i to što je faktor prigušenja veći, to se amplituda brže smanjuje. [10] Omjer dviju susjednih amplituda kod prigušenog titranja uvijek je stalan, te on iznosi:

$$\frac{A(t)}{A(t+T)} = e^{\delta T} \qquad (6\text{-}3)$$

pa slijedi da je logaritamski dekrement titranja:

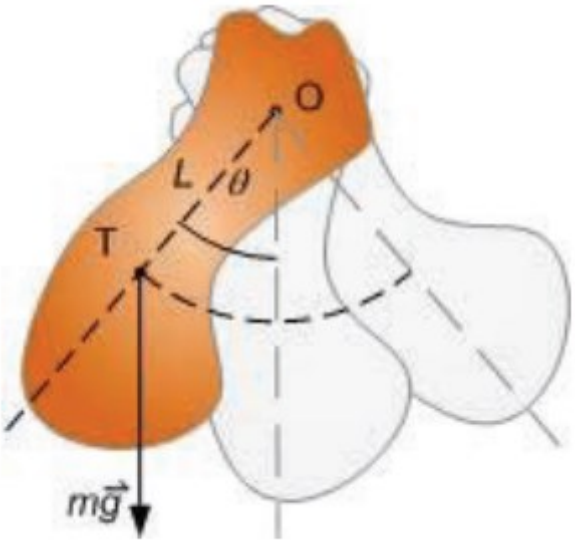
$$\lambda = \ln\left(\frac{A(t)}{A(t+T)}\right) = \delta T \qquad (6\text{-}4)$$

Logaritamski dekrement titranja λ i period titranja T titrajnog sustava je lako izmjeriti pa se pomoću izraza (6-4) može odrediti δ faktora prigušenja. Prigušeni titrajni sustav se također može

opisati Q-faktorom, odnosno faktorom dobrote ili kvalitete titrajnog sustava koji je određen izrazom: [10]

$$Q = \frac{\pi}{\lambda} = \frac{\pi}{\delta T} = \frac{\omega_0}{2\delta} \tag{6-5}$$

Fizikalno njihalo je svako kruto tijelo mase m, koje se njiše oko vodoravne osi koja ne prolazi kroz njegovo težište (**Slika 6.1.**). [10] Za male amplitude titranja, analiza titranja stvarnog fizikalnog njihala je vrlo slična analizi titranja jednostavnog njihala.



Slika 6.1. Fizikalno njihalo [10]

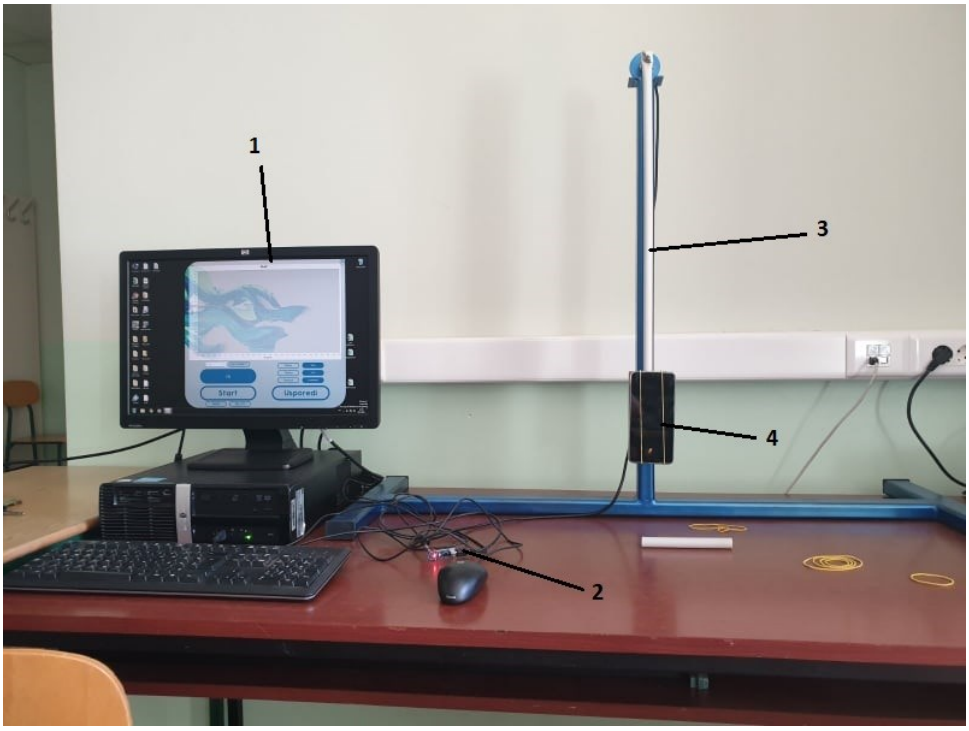
Period titranja fizikalnog njihala iznosi:

$$T = 2\pi\sqrt{\frac{I}{mgL}} \tag{6-6}$$

gdje je I moment tromosti tijela, m masa tijela, L udaljenost između ovjesišta i težišta tijela.

6.1. Eksperimentalni postav i postupak mjerenja eksperimenta

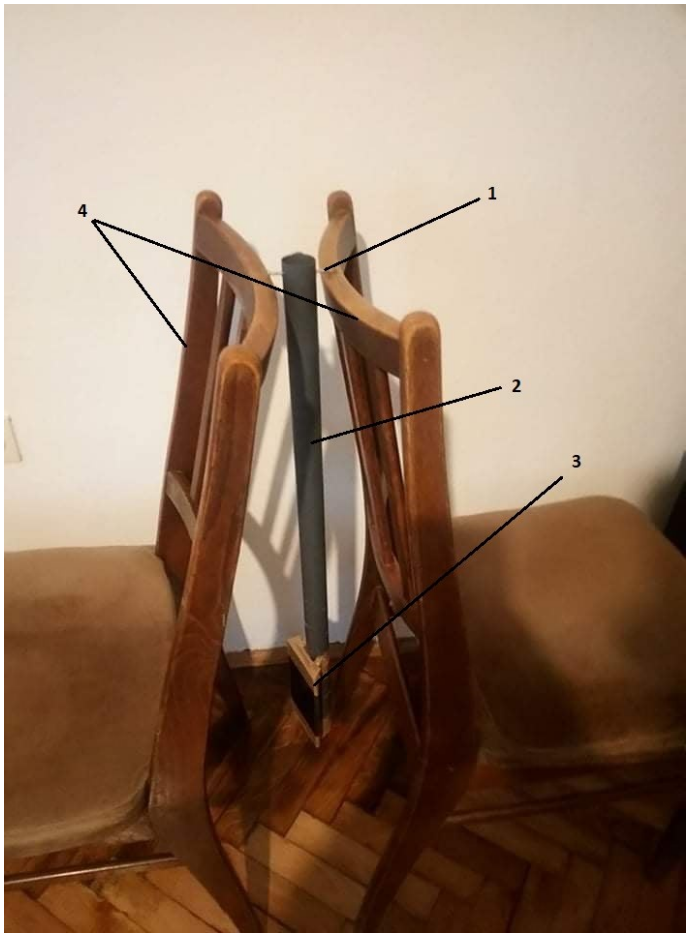
Razmatrani su rezultati na dva jako slična eksperimentalna postava, od kojih je jedan laboratorijski postav, a drugi improvizirani sustav napravljen kod kuće. U ovom eksperimentu fizikalno njihalo predstavlja maketa koja se sastoji od štapa i pametnog telefona koji će obavljati mjerenje kuta putem aplikacije. Štap je u slučaju obadva eksperimentalna postava dug 71.5 cm.



Slika 6.2. Laboratorijski postav eksperimenta

1 – Računalo, 2 – digitalni enkoder, 3 – štap, 4 – pametni telefon.

Prilikom izvođenja eksperimenta koristeći laboratorijski postav pametni telefon je potrebno pričvrstiti za štap tako da je dovoljno stabilan i čvrst da može izdržati njihanje na štapu. Na štapu se nalazi drveno postolje izrađeno odgovarajućih dimenzija za većinu pametnih telefona. Pametni telefon je potrebno pričvrstiti za drveno postolje gumicama kao što je prikazano na slici 6.2.

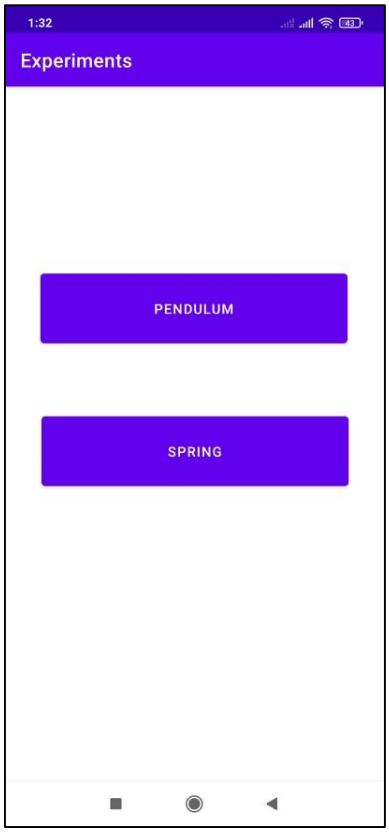


Slika 6.3. Improvizirani kućni eksperimentalni postav

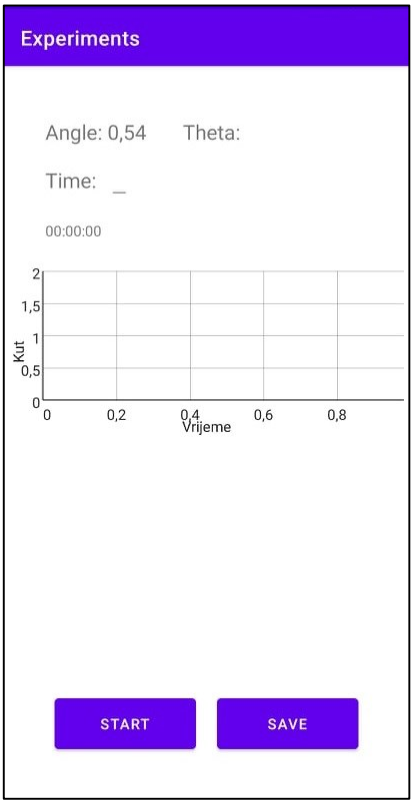
1 – Čavao, 2 – štap, 3 – pametni telefon, 4 - stolice

Prilikom izvođenja eksperimenta koristeći improvizirani kućni postav pametni telefon se za štap pričvršćuje ljepljivom trakom. Kroz vrh štapa prolazi čavao smješten između dvije stolice koji omogućuje titranje sustava kako što je prikazano na slici 6.3.

Eksperiment njihala se provodi tako da nakon što se aplikacija Experiments pokrene, na zaslonu prikazanom na slici 6.4. najprije stisne tipka Pendulum, nakon čega se otvara Pendulum aktivnost. Sučelje Pendulum aktivnosti prikazano je na slici 6.5.

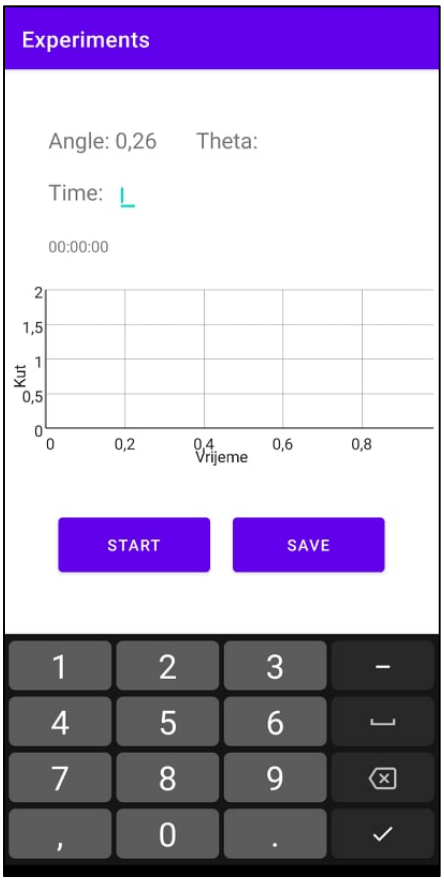


Slika 6.4. Sučelje aplikacije Experiments



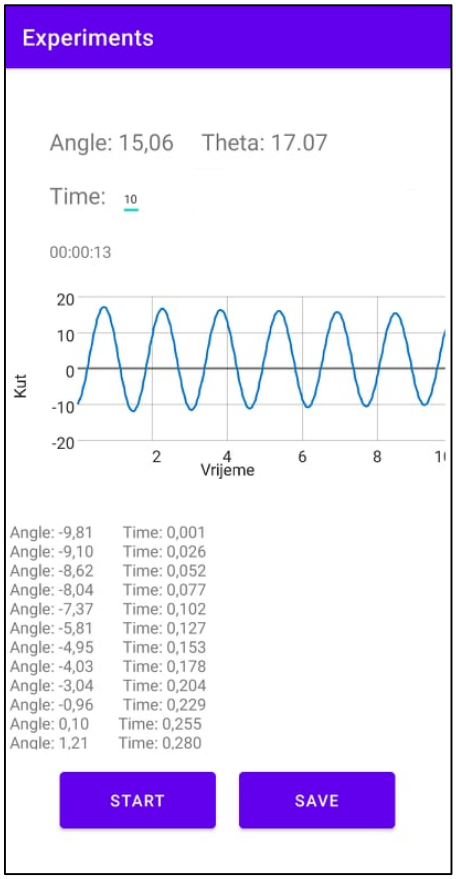
Slika 6.5. Sučelje aplikacije Experiments

Nakon toga je potrebno unijeti željeno vrijeme promatranja eksperimenta u polje *Time* kao što je prikazano na slici 6.6., zatim pričvrstiti pametni telefon za drveno postolje na štapu. Sljedeći korak je postaviti pametni telefon u početni položaj, odnosno otkloniti ga za željeni kut. Na kraju je potrebno pritisnuti tipku *Start*, ispustiti pametni telefon i pustiti ga da titra.



Slika 6.6. Unos željenog vremena

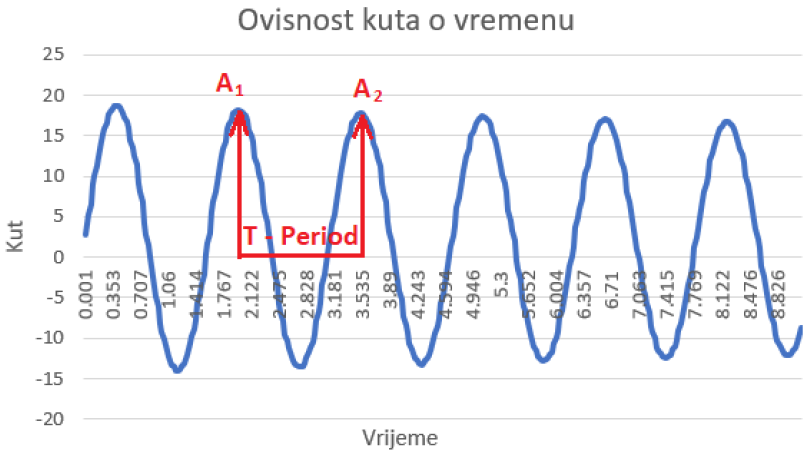
Nakon što je završio prethodno uneseni vremenski period, na zaslonu pametnog telefona se iscrtava graf koji prikazuje ovisnost kuta o vremenu kao što je prikazano na slici 6.7. Na kraju postoji mogućnost izvoza prikupljenih podataka u Excel za dalju obradu pritiskom na tipku *Save*.



Slika 6.7. Graf ovisnosti kuta o vremenu

6.2. Rezultati eksperimenta

Eksperiment titranja fizikalnog njihala izvodi se nekoliko puta sa promjenom kuta otklona i položaja pametnog telefona na štapu, te se promatra kako se mijenja kutna elongacija o vremenu pri titranju fizikalnog njihala (**Slika 6.7.**). Iz ovisnosti kutne elongacije o vremenu može se odrediti amplituda i period titranja. Određivanje amplitude i perioda prikazano je na slici 6.8.



Slika 6.8. Određivanje amplitude i perioda

Prema slici 6.8. period titranja računa se prema sljedećem izrazu:

$$T = t(A_2) - t(A_1) \tag{6-7}$$

gdje je $t(A_1)$ vremenski trenutak prve amplitude, a $t(A_2)$ vremenski trenutak sljedeće amplitude.

Za izračun faktora prigušenja koristi se izvod iz izraza (6-4), te slijedi:

$$\delta T = \ln\left(\frac{A(t)}{A(t+T)}\right)$$

$$\delta = \frac{\ln\left(\frac{A(t)}{A(t+T)}\right)}{T} \tag{6-8}$$

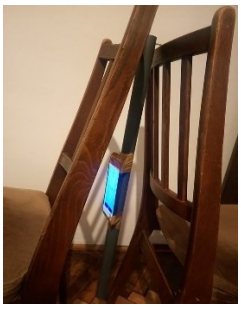
Rezultati mjerenja dok je udaljenost između objesišta i centra mase pametnog telefona 59.5 cm , pri početnom otklonu od 18.65°:

$$T = 2.623 - 1.01 = 1.613\ s$$

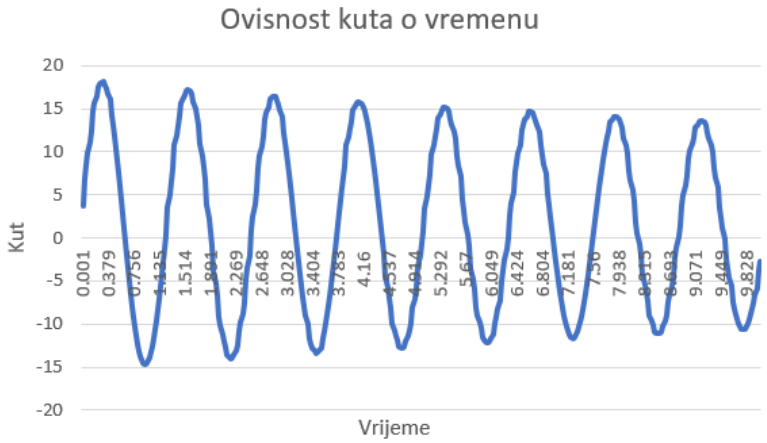
$$\delta = \frac{\ln\left(\frac{18.65}{18.29}\right)}{1.613} = 0.0121$$

Eksperiment se provodi nekoliko puta dok je udaljenost između objesišta i centra mase pametnog telefona 59.5 cm pod različitim kutovima otklona, zatim nekoliko puta dok je udaljenost između objesišta i centra mase pametnog telefona 35 cm. Rezultati ovih mjerenja prikazani su u tablici 6.1.

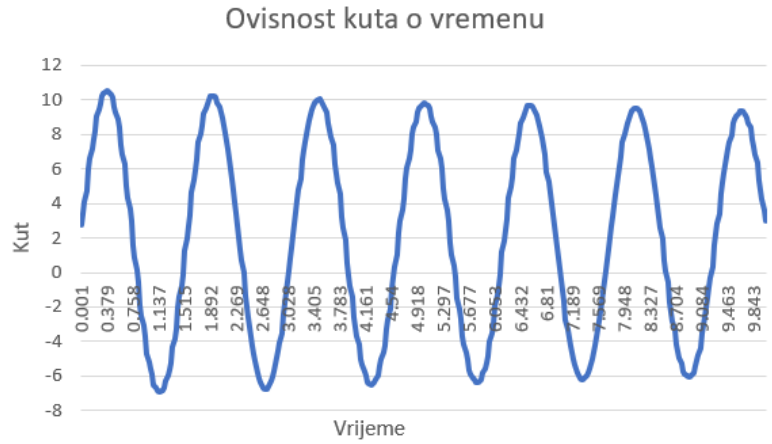
Tablica 6.1. Rezultati mjerenja na improviziranom kućnom eksperimentalnom postavu

Udaljenost između objesišta i pametnog telefona 59.5 cm	$\theta[°]$	A_1 [°]	$A_2[°]$	$T[s]$	$\delta[s^{-1}]$
	10.5°	10.5°	10.23°	1.538	0.0169
	11.75°	11.75°	11.51°	1.563	0.0132
	12.23°	12.23°	11.84°	1.562	0.0207
	14.14°	14.14°	13.64	1.564	0.0230
	16.57°	16.57°	15.95°	1.564	0.0244
Udaljenost između objesišta i pametnog telefona 35 cm	$\theta[°]$	A_1 [°]	$A_2[°]$	$T[s]$	$\delta[s^{-1}]$
	15.29°	15.29°	14.68°	1.285	0.0317
	14.92°	14.92°	14.2°	1.262	0.0392
	18.09°	18.09°	17.17°	1.262	0.0414
	14.06°	14.06°	13.34°	1.263	0.0416
	15.59°	15.59°	14.82°	1.262	0.0401

Prema rezultatima tablice 6.1. može se zaključiti da kada je udaljenost između objesišta i centra mase pametnog telefona 35 cm, period titranja je kraći, a faktor prigušenja veći, što se i da vidjeti na slikama 6.9. i 6.10.

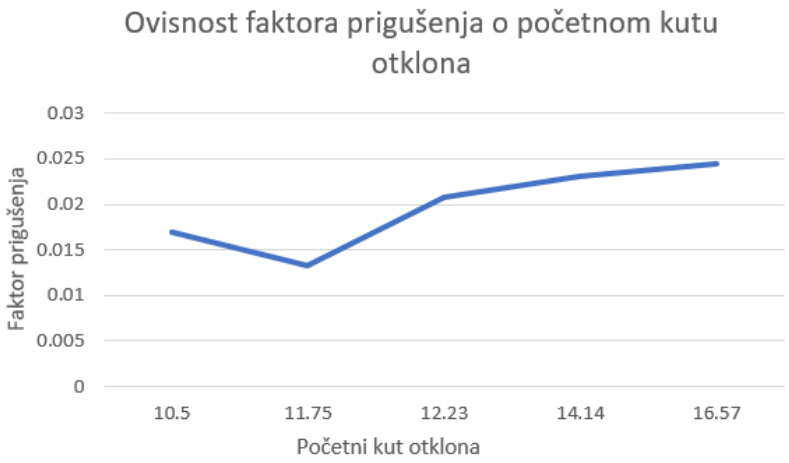


Slika 6.9. Graf ovisnosti kuta o vremenu kada je udaljenost između objesišta i centra mase pametnog telefon 35 cm

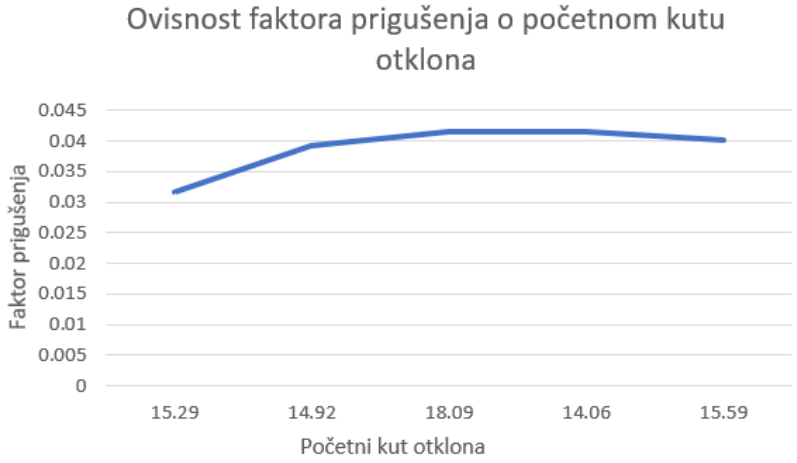


Slika 6.10. Graf ovisnosti kuta o vremenu kada je udaljenost između objesišta i centra mase pametnog telefon 59.5 cm

Prema rezultatima iz tablice 6.1. na slikama 6.11. i 6.12. prikazana je ovisnost faktora prigušenja o početnom kutu otklona.



Slika 6.11. Ovisnost faktora prigušenja o početnom kutu otklona kada je udaljenost između objesišta i pametnog telefona 59.5 cm



Slika 6.12. Ovisnost faktora prigušenja o početnom kutu otklona kada je udaljenost između objesišta i pametnog telefona 35 cm

U nastavku slijedi mjerenje obavljeno pomoću android aplikacije koje će se usporediti sa mjerenjem pomoću računalne aplikacije.

Za početni kut otklona 18.65°, dok se pametni telefon nalazi na kraju štapa, vrijedi da je period:

$$T = 2.623 - 1.01 = 1.613\ s$$

$$T = 1.613\ s$$

Na isti način računaju se i ostali periodi, te je u tablici 6.2. prikazano svih 5 pripadajućih perioda.

Tablica 6.2. Rezultati perioda

	1.	2.	3.	4.	5.
$T(s)$	1.613	1.611	1.615	1.612	1.639

Prema tablici 6.1. proizlazi da je srednja vrijednost perioda:

$$T = \frac{1.613 + 1.611 + 1.615 + 1.612 + 1.639}{5} = 1.618\ s$$

Prema izrazu (6-8) računa se pripadajući faktor prigušenja, te je za prvi period to:

$$\delta = \frac{\ln\left(\frac{A(t)}{A(t+T)}\right)}{T} = \frac{\ln\left(\frac{18.65}{18.29}\right)}{1.613} = 0.0121\ s^{-1}$$

Na isti način se računaju i ostali faktori prigušenja, te je u tablici 6.3. prikazano svih 5 pripadajućih faktora prigušenja.

Tablica 6.3. Rezultati faktora prigušenja

	1.	2.	3.	4.	5.
$\delta(s^{-1})$	0.0121	0.0193	0.0174	0.0187	0.0183

Prema tablici 6.2. proizlazi da je srednja vrijednost faktora prigušenja:

$$\delta = \frac{0.0121 + 0.0193 + 0.0174 + 0.0187 + 0.0183}{5} = 0.01716\ s^{-1}$$

Uspoređuju se rezultati dva slična mjerenja na različitim postavima. Prilikom izvođenja mjerenja koristeći laboratorijski postav, gdje je pametni telefon pričvršćen na kraj štapa pri početnom otklonu od 18.65°, rezultati perioda i faktora prigušenja su sljedeći:

$$T = 2.623 - 1.01 = 1.613\ s$$

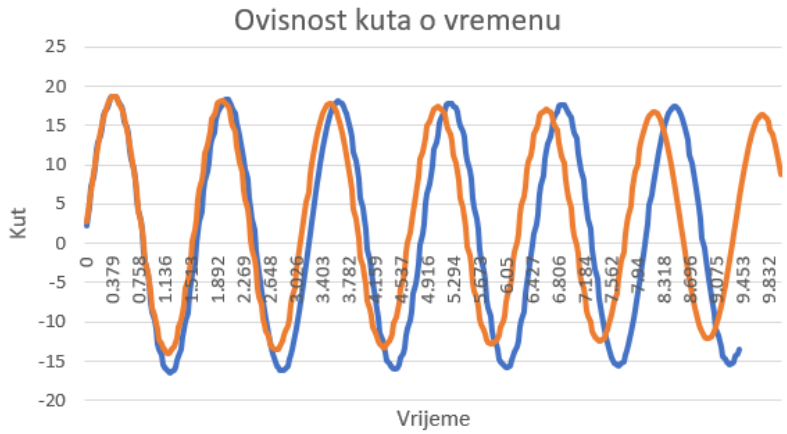
$$\delta = \frac{\ln\left(\frac{18.65}{18.24}\right)}{1.613} = 0.0138\ s^{-1}$$

Prilikom izvođenja mjerenja koristeći improvizirani kućni postav, gdje je pametni telefon pričvršćen na kraj štapa pri početnom otklonu od 18.73°, rezultati su:

$$T = 1.97 - 0.403 = 1.567\ s$$

$$\delta = \frac{\ln\left(\frac{18.73}{18.12}\right)}{1.567} = 0.0211\ s^{-1}$$

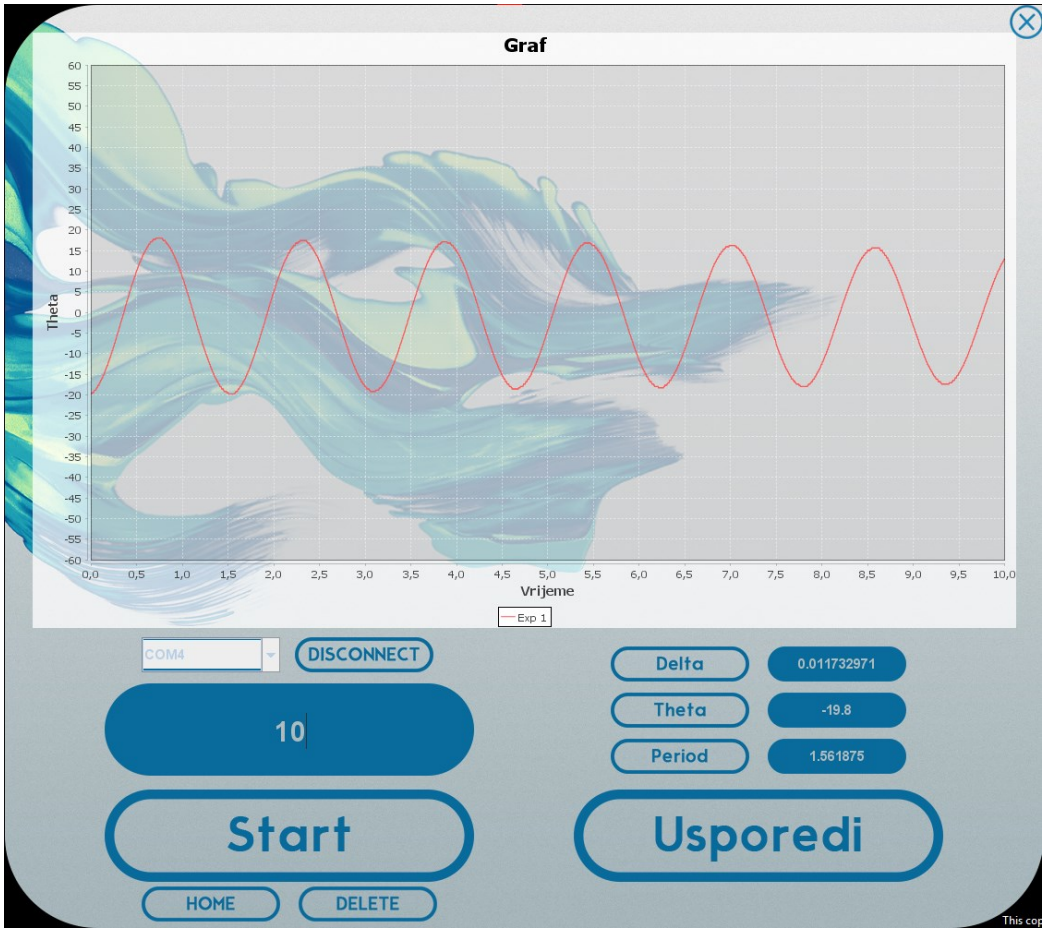
Zaključuje se kako je prilikom korištenja improviziranog kućnog postava faktor prigušenja veći nego kod mjerenja na laboratorijskom postavu. (**Slika 6.13.**)



Slika 6.13. Usporedba rezultata dva različita eksperimentalna postava

6.3. Usporedba rezultata sa sličnim aplikacijama

Rezultati dobiveni mjerenjem pomoću android aplikacije se uspoređuju sa rezultatima dobivenim pomoću računalne aplikacije koja je ostvarena pomoću makete fizikalnog njihala s digitalnim prikazom parametara titranja. Maketa se sastoji od digitalnog (rotacijskog) enkodera, na čiju se glavu mogu staviti različita tijela, koji je spojen na Arduino Mega pločicu. Putem serijskog priključka, podatci koje enkoder očitava šalju se računalnoj aplikaciji koja je realizirana u Java programskom jeziku. Računalna aplikacija ima mogućnost prikaza kutne elongacije fizikalnog njihala u ovisnosti o vremenu na ekranu. (**Slika 6.14.**) [11]



Slika 6.14. Prikaz kutne elongacije fizikalnog njihala u ovisnosti o vremenu [11]

Rezultati mjerenja dobiveni pomoću računalne aplikacije obrađuju se na isti način kao i rezultati dobiveni mjerenjem pomoću android aplikacije, te se također računaju period i faktor prigušenja. Period se računa prema izrazu (6-7), a faktor prigušenja prema izrazu (6-8). Izračunate vrijednosti perioda i faktora prigušenja za mjerenje računalnom aplikacijom prikazane su u tablici 6.4.

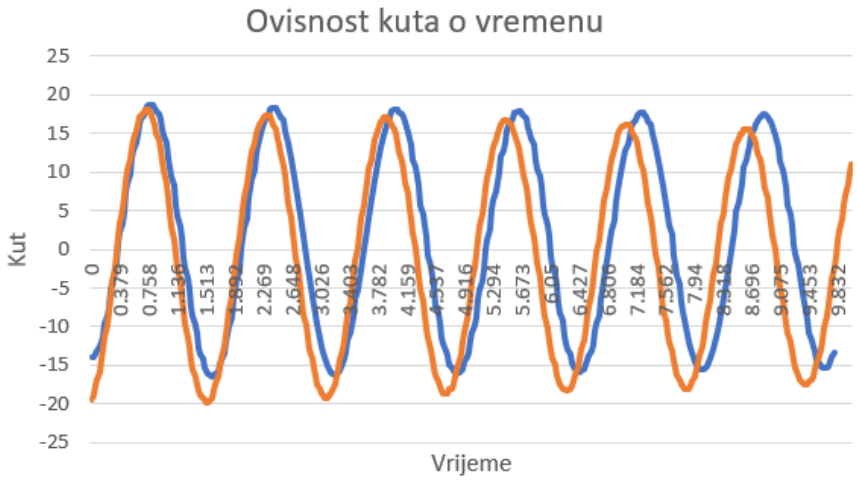
Tablica 6.4. Vrijednosti perioda i faktora prigušenja mjerenja obavljenim pomoću računalne aplikacije

	1.	2.	3.	4.	5.
$T(s)$	1.62	1.565	1.605	1.6	1.58
$\delta(s^{-1})$	0.0174	0.0111	0.0123	0.0228	0.0239

Iz tablice 6.4. računa se srednja vrijednost faktora prigušenja:

$$\delta = \frac{0.0174 + 0.0111 + 0.0123 + 0.0228 + 0.0239}{5} = 0.0175\ s^{-1}$$

Na slici 6.15. imamo usporedbu rezultata android i računalne aplikacije.



Slika 6.15. Usporedba rezultata android i računalne aplikacije

Usporedba rezultata mjerenja iste fizikalne veličine dobivene pomoću dva različita mjerna postupka izražava se relativnom razlikom koja se izračunava prema relaciji: [10]

$$\boldsymbol{relativna\ razlika} = \left|\frac{E_1 - E_2}{\frac{1}{2}(E_1 + E_2)}\right| \times 100\% \qquad (6\text{-}9)$$

Da bi se iskazala relativna razlika u postotcima između ova dva načina mjerenja koristi se izraz (6-9), te je relativna razlika u postotcima jednaka:

$$\left|\frac{E_1 - E_2}{\frac{1}{2}(E_1 + E_2)}\right| \times 100\% = \left|\frac{0.0175 - 0.01716}{\frac{1}{2}(0.0175 + 0.01716)}\right| \times 100\% = 1.96\ \%$$

Prema izrazu (6-9) relativna razlika u postotcima između rezultata dobivenih mjerenjem pomoću android aplikacije i rezultata dobivenih mjerenjem pomoću računalne aplikacije je 1.96%.

7. ZAKLJUČAK

Cilj ovog rada bio je izraditi aplikaciju za pametni telefon pomoću koje će se telefon moći koristiti kao mjerni instrument, te na taj način olakšati i povećati fleksibilnost i mobilnost mjerenja eksperimenata iz fizike. Danas većina ljudi koristi mobilne uređaje i konstantno im imaju pristup, ideja rada je bila da svaki korisnik može mjeriti eksperimente iz fizike preko aplikacije pomoću svog pametnog telefona i njegovih senzora. Ideja je jako praktična jer je potreban samo pametni telefon. Aplikacija je jako pogodna za održavanje nastave na daljinu, pogotovo u današnje vrijeme kada svijetom vlada Covid-19 virus, jer korisnici mogu mjerenje eksperimenata provoditi kod kuće. Aplikacija je uspješno realizirana, te uspoređena sa već postojećim sličnim rješenjem. Iz rezultata usporedbe može se zaključiti da i jedno i drugo rješenje daju slične rezultate.

Sučelje aplikacije je jednostavnog izgleda i aplikacija ne zahtijeva povezanost na internet, što znači da se aplikacija može koristiti bilo gdje u bilo koje vrijeme. Aplikacija pomoću senzora mjeri jako precizno uz period spremanja podataka od 25 milisekundi, iako bi se u budućnosti moglo dodatno proučiti programiranje senzora da period spremanja podataka bude još manji, odnosno da senzor još brže sprema izmjerene podatke. Na kraju mjerenja je omogućen izvoz spremljenih podataka za dalju obradu.

LITERATURA

[1] Phyphox, dostupno na: https://phyphox.org, datum posjete: 22.07.2021.

[2] My Altitude, dostupno na: https://www.bejbej.ca/app/myaltitude, datum posjete: 22.07.2021.

[3] Moasure, dostupno na: https://www.moasure.com, datum posjete: 22.07.2021.

[4] Angle Meter, dostupno na: https://www.slashdigit.com/best-angle-measure-apps, datum posjete: 23.07.2021.

[5] SpeedView, dostupno na: https://www.codesector.com/speedview, datum posjete: 23.07.2021.

[6] Senzori, dostupno na: https://developer.android.com/guide/topics/sensors/sensors_overview, datum posjete: 01.08.2021.

[7] Operacijski sustav Android, dostupno na: https://www.android.com, datum posjete: 04.08.2021.

[8] Java, dostupno na: https://www.javatpoint.com/java-tutorial, datum posjete: 05.08.2021.

[9] Geeks for geeks, dostupno na: https://www.geeksforgeeks.org/android-project-folder-structure, datum posjete: 07.08.2021.

[10] Ž. Mioković, Fizika 1, priručnik za laboratorijske vježbe, Elektrotehnički fakultet Osijek, Sveučilište J.J. Strossmayera u Osijeku, Osijek, 2013.

[11] S. Stanić, Završni rad, Maketa fizikalnog njihala s digitalnim prikazom parametara titranja, Elektrotehnički fakultet Osijek, Sveučilište J.J. Strossmayera u Osijeku, Osijek, 2017.

SAŽETAK

U ovom završnom radu razvijena je Android mobilna aplikacija koja omogućava korisnicima provođenje eksperimenata iz fizike i mjerenje određenih parametara koristeći samo pametni telefon, aplikacija je nazvana Experiments. Aplikacija omogućuje prikaz kutne elongacije u ovisnosti o vremenu na zaslonu pametnog telefona. Objašnjene su metode i funkcije koje se koriste za održavanje glavnih funkcionalnosti aplikacije. Nakon obavljenog eksperimenta omogućen je izvoz podataka za dalju obradu. Rezultati su zatim uspoređeni sa rezultatima dobivenih sličnom aplikacijom. Aplikacija je izrađena u programskom okruženju Android Studio, koristeći se programskim jezikom Java.

Ključne riječi: Android, Eksperiment, Java, Njihalo, Senzor

ABSTRACT

Development of a smart phone application as a measuring instrument

The purpose of this bachelor's thesis was developing an Android application which can display angle as a function of time on a graph. Application allows the user to use smart phone as a measuring device. All the measuring is done through the device's sensors. Main goal of developing an application like this was to enable users to perform experiments even while they're away from the physics laboratory. Most important methods and functions that are in charge of maintaining the main functionalities of the application are precisely described in the project. Application was developed in Java programming language in Android Studio IDE.

Keywords: Android, Experiment, Java, Pendulum, Sensor

ŽIVOTOPIS

Tomislav Odobašić rođen je 04.12.1998. godine u Slavonskom Brodu, Republika Hrvatska. Završio je Osnovnu Školu Orašje u Orašju. Srednju školu je završio u Orašju, smjer elektrotehničar. 2017. godine upisao se na preddiplomski sveučilišni studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.