Dinamičko praćenje procesa i poruka sa Siemens programirljivog logičkog kontrolera

Rajčevac, Matija

Master's thesis / Diplomski rad

2022

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:200:348712

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

MOBILNA APLIKACIJA ZA VIZUALIZACIJU FIZIKALNIH MJERENJA

Diplomski rad

Dominik Tkalčec

Osijek, 2022.

SADRŽAJ

1. UVOD 1

1.1. Zadatak diplomskog rada 1

2. UPOTREBA MOBILNIH UREĐAJA U SVRHU VIZUALIZACIJE FIZIKALNIH MJERENJA 3

2.1. Phyphox 3

2.2. Arduino science journal 4

3. OPIS ALATA POTREBNIH ZA IMPLEMENTACIJU APLIKACIJE 6

3.1. Android operacijski sustav 6

3.1.1. Arhitektura sustava 8

3.2. Programski jezik Kotlin 10

3.3. Android studio programsko okruženje 11

3.3.1. Struktura projekta 12

3.3.2. Korisničko sučelje 13

4. RAZVOJ I IMPLEMENTACIJA APLIKACIJE 15

4.1. Dijagram tijeka 15

4.2. Struktura projekta 17

4.3. Arhitektura projekta 18

4.4. Korištene biblioteke 20

4.5. Podešavanje Arduino programa 21

4.5.1. Definiranje osnovne strukture 21

4.5.2. Slanje konfiguracije dostupnih eksperimenata 22

4.5.3. Slanje podataka primljenih od mjernog uređaja 24

4.6. Podešavanja Android aplikacije 26

5. PRIKAZ RADA APLIKACIJE 28

5.1. Zaslon s popisom uređaja dostupnih za povezivanje (početni zaslon) 28

5.2. Zaslon s popisom eksperimenata 30

5.3. Zaslon s prikazom pojedinosti o eksperimentu 31

5.4. Zaslon postavki 34

5.5. Zaslon s informacijama o aplikaciji 35

6. ZAKLJUČAK 36

LITERATURA 37

SAŽETAK 38

ABSTRACT 39

ŽIVOTOPIS 40

1. UVOD

Tijekom izvođenja laboratorijskih vježbi iz fizike zadatak svakog fizikalnog mjerenja jest utvrditi brojčanu vrijednost izmjerene fizikalne veličine. Zbog nepreciznosti mjernih instrumenata kao i zbog nesavršenosti ljudskih osjetila ni jedno mjerenje nije apsolutno točno. Ako se uzme u obzir ograničena oštrina ljudskog razlučivanja dolazi se do zaključka kako ljudska osjetila nisu dovoljno pouzdana za očitavanja rezultata mjerenja. Zbog navedenog problema javlja se potreba za digitalizacijom fizikalnih veličina koja je u ovom radu riješena korištenjem Arduino Uno mikroupravljača.

Mjerenjem fizikalnih veličina dobije se niz vrijednosti koje se mogu zapisati u tabličnom obliku, ali one same po sebi ne daju dovoljno informacija za donošenje zaključaka. Iz toga razloga nužno je provesti vizualizaciju fizikalnih veličina kod prikazivanja rezultata mjerenja jer se iz nje donosi niz zaključaka o odnosu veličina. Nakon što se mjerene točke prikažu na grafu lako se uočava linearna ovisnost ako postoji takva. Danas, u doba intenzivnog korištenja mobilnih uređaja sve više poduzeća fokusira se na razvoj mobilnih aplikacija koji olakšavaju čovjekovu svakodnevicu pa ni ne čudi spoznaja da postoje aplikacije gotovo za sve. Iz toga razloga krajnji rezultat ovog diplomskog rada je mobilna aplikacija.

Ovaj diplomski rad razrađen je u 4 poglavlja. U prvom poglavlju dan je kratki opis te su navedene značajke pronađenih rješenja aplikacija sa sličnom tematikom. U drugom poglavlju razrađeni su alati korišteni za izradu aplikacije, a u trećem poglavlju prikazan je proces realizacije aplikacije. U četvrtom, posljednjem poglavlju opisan je način rada aplikacije potkrijepljen slikama zaslona izrađene aplikacije.

1.1. Zadatak diplomskog rada

U okviru ovog diplomskog rada potrebno je izraditi mobilnu aplikaciju za Android operacijski sustav koristeći programski jezik Kotlin i MVVM arhitekturu. Aplikacija treba omogućiti prikupljanje fizikalnih mjerenja s mikroupravljača primjenom *Bluetooth* tehnologije. Osim samog prikupljanja podataka potrebno je napraviti i vizualizaciju prikupljenih podataka u tabličnom obliku te u grafičkom obliku. Dodatne funkcionalnosti koje aplikacija mora osigurati su:

- tamni i svijetli način rada
- sučelje na hrvatskom i engleskom jeziku ovisno o jeziku operacijskog sustava

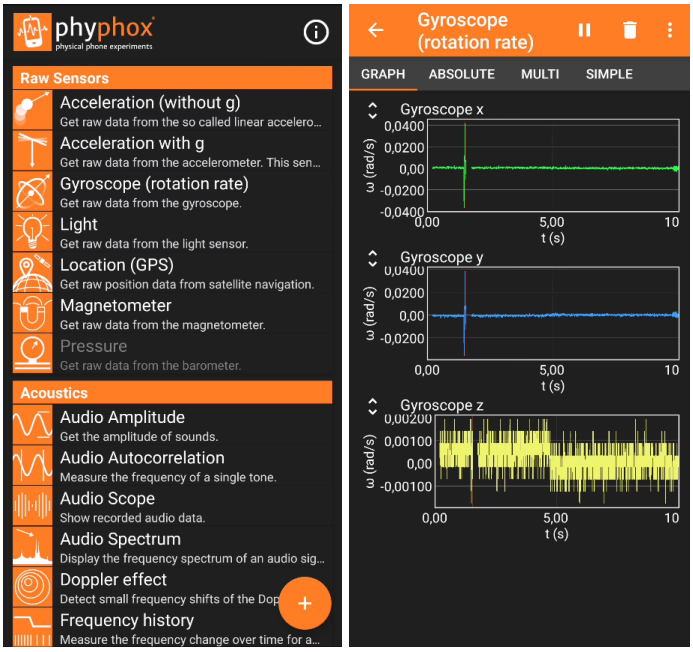
- konfiguriranje načina dohvaćanja podatka s mikroupravljača
- izvoz podataka u Excel datoteku i mogućnost dijeljenja s drugim aplikacijama

2. UPOTREBA MOBILNIH UREĐAJA U SVRHU VIZUALIZACIJE FIZIKALNIH MJERENJA

Posljednjih godina tijekom početka pandemije bolesti COVID-19 uzrokovane koronavirusom sva nastava se morala prilagoditi održavanju na daljinu. Takav tip nastave je izrazito nepovoljan za eksperimentalni rad te je studentima i učenicima onemogućio izvođenje velikog broja eksperimenata. Iz toga razloga počela su se pojavljivati alternativna rješenja koja omogućuju izvođenje jednostavnih fizikalnih eksperimenata korištenjem dostupnih senzora mobilnog uređaja, ali i izvođenje onih naprednijih korištenjem mikroupravljača i *Bluetooth* tehnologije. Pretražujući Internet, pronađeno je nekoliko mobilnih aplikacija koje rješavaju navedeni problem, a neke su opisane u potpoglavljima koji slijede.

2.1. Phyphox

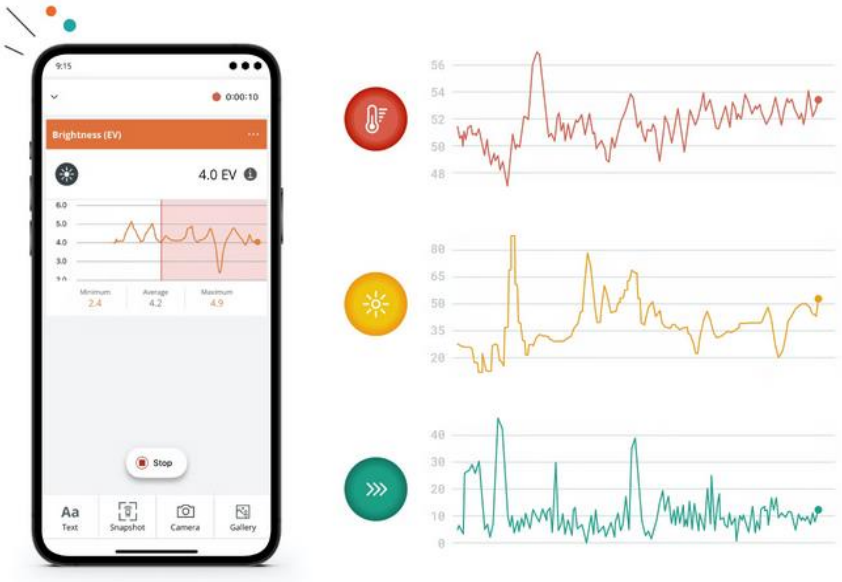
Jedna od najpoznatijih aplikacija za korištenje uređaja kao instrumenta za fizikalna mjerenja, ali i za vizualizaciju istih. Neki od dostupnih senzora za provođenje eksperimenata su akcelerometar, žiroskop, magnetometar, senzor blizine, senzor svjetlosti i termometar [1]. Najvažnije značajke koje nam aplikacija omogućuje su: mogućnost izvoza podataka u Excel, CSV (engl. *Comma-separated values*) i TSV (engl. *Tab-separated values*) format [2], mogućnost udaljenog upravljanja s bilo kojeg uređaja povezanog na mrežu kao i mobilni uređaj te generiranje vlastitih eksperimenata koji se ne nalaze na listi ponuđenih. Još jedna zanimljiva mogućnost koju aplikacija *Phyphox* omogućuje je povezivanje s Arduino mikroupravljačem koristeći BLE (engl. *Bluetooth low-energy*) tehnologiju. *Phyphox* je za tu potrebu razvio vlastitu biblioteku naziva "phyphox BLE" koja omogućuje jednostavnu vizualizaciju podataka s Arduino mikroupravljača u *Phyphox* aplikaciji, ali i primanje podataka sa senzora iz *Phypox* aplikacije u Arduino projekt [3]. Mobilna aplikacija je dostupna za preuzimanje na *Google Play* i *App Store* trgovinama. Prikaz izgleda korisničkog sučelja *Phyhox* aplikacije nalazi se na slici 2.1.



Sl. 2.1. Sučelje aplikacije Phyphox

2.2. Arduino science journal

Druga u redu aplikacija koja uz pomoć mobilnog uređaja omogućava izvođenje i vizualizaciju fizikalnih mjerenja je *Arduino Science Journal*. Ova aplikacija kao i aplikacija opisana u prethodnom poglavlju omogućuje različita mjerenja u ovisnosti o rasponu dostupnih senzora na mobilnom uređaju. Neki od podržanih senzora za izvođenje mjerenja koje aplikacija nudi su: kompas, magnetometar, akcelerometar, senzor svjetlosti i senzor jačine zvuka [4]. *Arduino Science Journal* aplikacija također nudi mogućnost povezivanja Arduino mikroupravljača putem BLE tehnologije. Na glavnoj web stranici proizvođača pronađen je Arduino Nano 33 model kao primjer jednog od mogućih mikroupravljača za povezivanje s aplikacijom [5]. Mobilna aplikacija je dostupna za preuzimanje na *Google Play*, *AppGallery* i *App Store* trgovinama. Izgled korisničkog sučelja *Arduino Science Journal* aplikacije nalazi se na slici 2.2.



Sl. 2.2. Sučelje aplikacije Arduino Science Journal

Prednosti *Arduino Science Journal* aplikacije u odnosu na *Phyphox*:

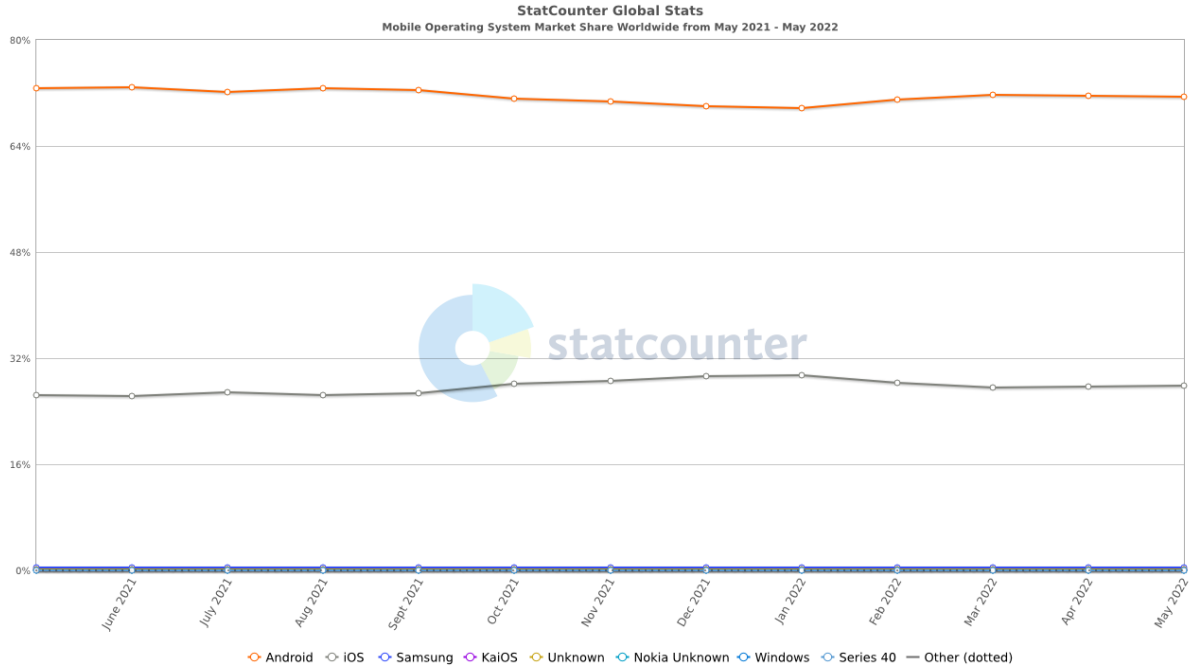
- jednostavniji i moderniji izgled sučelja
- sinkronizacija eksperimenata s Google Diskom
- izvoz eksperimenata u PDF formatu
- prilagođeno za rad sa studentima i učenicima (kreiranje prilagođenih eksperimenata, dijeljenje eksperimenata između studenata i učenika, dodjeljivanje eksperimenata za zadaću)

3. OPIS ALATA POTREBNIH ZA IMPLEMENTACIJU APLIKACIJE

U ovom poglavlju ukratko su opisani alati korišteni za implementaciju mobilne aplikacije. Osim opisa prikupljeni su i statistički podaci korištenih alata kako bi se bolje predočila njihova važnost za daljnji tijek razvoja mobilnih aplikacija. Android operacijski sustav pokriva najveći dio svjetskog tržišta operacijskih sustava za mobilne uređaje, dok Kotlin pripada grupi novih, najbrže rastućih programskih jezika zbog čega su plaće poznavatelja jezika Kotlin pri samom vrhu.

3.1. Android operacijski sustav

Android je operacijski sustav otvorenog koda (engl. *open source*) koji je u vlasništvu tvrtke Google od 2005. godine. Prvobitno je razvijen samo za mobilne uređaje, a danas se koristi u većini pametnih uređaja poput tableta, televizora, sata i automobila. Android danas zauzima najveći dio tržišnog udjela operacijskih sustava za mobilne uređaje odnosno čak 71.43% cjelokupnog tržišta, a prethodi ga iOS s 27.85% [6]. Tržišni udio operacijskih sustava za mobilne uređaje prikazan je na slici 3.1.

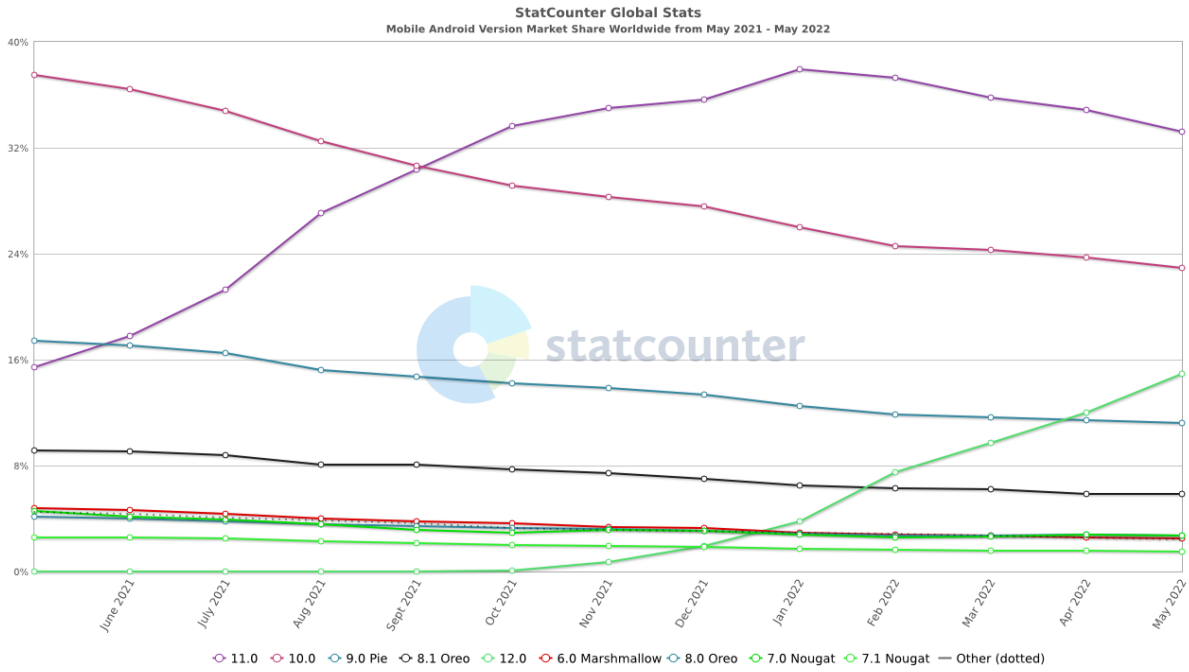


Sl. 3.1. Tržišni udio operacijskih sustava za mobilne uređaje u razdoblje od 05/2021. do 05/2022.

Android operacijski sustav podržava programske jezike koji u pozadini koriste JVM (engl. *Java Virtual Machine*). Najzastupljeniji takvi jezici su Kotlin i Java. Programski jezik Java je sve do 2019. godine bio primarni programski jezik za razvoj Android aplikacija. Zbog potrebe za optimizacijom izrade aplikacija, ali i zbog pojave modernih značajki poput lambda funkcija, ništavnih tipova (engl. *nullable types*) i tokova podataka (engl. *streams*) koje omogućuju čišći, pregledniji i količinski manji kod za iste funkcionalnosti, programski jezik Kotlin je zamijenio njegovu ulogu.

Do danas je razvijeno ukupno 20 verzija sustava Android, a najnovija je Android 12L [7]. Međutim prema podacima s Interneta danas se u svijetu koriste mobilni uređaji sa sljedećim verzijama i postocima (slika 3.2.) [8]:

- 8.0 Oreo s 2.75%
- 8.1 Oreo s 5.83%
- 9.0 Pie s 11.22%
- 10.0 s 22.98%
- 11.0 s 33.24%
- 12.0 s 14.84%.

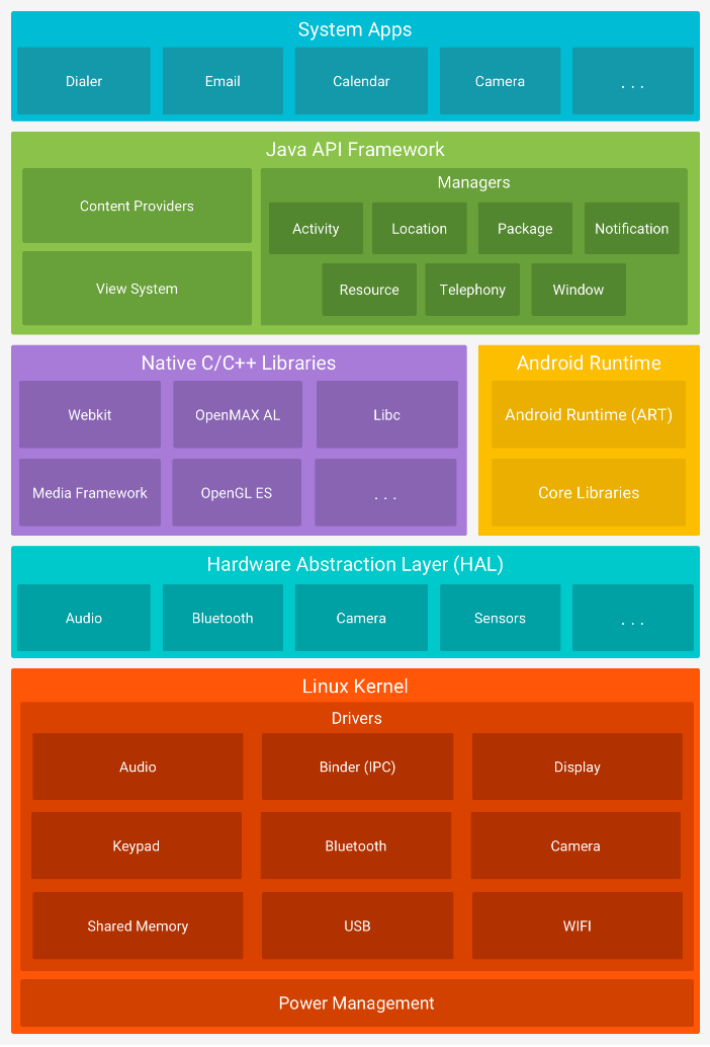


Sl. 3.2. Tržišni udio aktivnih verzija Android sustava na mobilnim uređajima u razdoblje od 05/2021. do 05/2022.

3.1.1. Arhitektura sustava

Što se tiče arhitekture, Android je baziran na Linux operacijskom sustavu i napisan C/C++ programskim jezikom. Linux sustav je također operacijski sustav otvorenog koda, a omogućuje aplikacijama komunikaciju preko posrednika (engl. *middleware*) te pokretanje drugih aplikacija primjerice aplikacije za pokretanje kamere, očitavanje sa senzora, slanje SMS poruka, ostvarivanje poziva, itd. Android arhitektura se može promatrati kao programski stog koji se sastoji od 6 dijelova podijeljenih u 5 razina (slika 3.3.):

- Aplikacije sustava
- Java API okvir
- Nativne C/C++ biblioteke, Android runtime
- Sloj apstrakcije hardvera (engl. *Hardware abstraction layer*)
- Linux jezgra



Sl. 3.3. Prikaz strukture Android operacijskog sustava

Na samom dnu programskog stoga nalazi se Linux koji sadrži upravljačke programe (engl. *drivers*) koji omogućuju komunikaciju različitih procesa i niti unutar istog procesa te upravljačke programe za upravljanje napajanjem (engl. *power management*). Razinu prije Linux-a smješten je HAL (engl. *Hradware Abstraction Layer*), sloj hardvera koji pruža standardizirana sučelja za upravljanje mogućnostima hardvera na višoj razini. HAL se sastoji od više biblioteka pri čemu svaka implementira sučelje za određenu vrstu hardverske komponente. Nakon što viša razina uputi API poziv za pristup željenom hardveru uređaja, Android sustav vrati biblioteku za traženi hardver.

Kod uređaja koji koriste Android verziju 5.0 ili noviju, svaka aplikacija pokreće vlastiti proces i vlastitu instancu ART-a (engl. *Android Runtime*). ART se koristi za pokretanje više virtualnih strojeva kod uređaja s malo memorije pokretanjem datoteka u DEX (engl. *Dalvik executable*) formatu. DEX datoteka predstavlja format bajt koda koji je dizajniran specijalno za Android. Glavne značajke ART-a su: bolja optimiziranost upravljanja memorijom sakupljanjem smeća, bolja podrška za otklanjanje poteškoća, JIT (engl. *Just-In-Time*) i AOF (engl. A*ll-Of-Time*) kompajliranje. JIT predstavlja vrstu kompajliranja gdje se kompajliranje napisanog koda odvija za vrijeme izvođenja aplikacije, dok AOF predstavlja vrstu kompajliranja gdje se kompajliranje određenih dijelova koda odvija prije samog početka izvođenja aplikacije.

Mnoge komponente i usluge za Android sustav kao što su ART i HAL izrađene su od izvornog koda koji zahtjeva izvorne datoteke pisane u C/C++ programskom jeziku. Android platforma pruža Java okvir s API-ima (engl. *Application Programming Interface*) koji omogućuju korištenje funkcionalnosti tih biblioteka. Cijeli skup značajki Android sustava dostupan je pozivima API metoda koje čine temelj potreban za izradu bilo koje aplikacije čijom se upotrebom pojednostavljuje ponovna upotreba osnovnih te modularnih komponenti sustava i usluga.

Na samom vrhu programskog stoga smještene su aplikacije koje dolaze s Android sustavom, a to su aplikacije za e-poštu, slanje SMS poruka, kalendare, pretraživanje interneta, kontakte, glazbu, fotografije, itd. Aplikacije sustava nemaju specijalni status u odnosu na aplikacije koje je korisnik preuzeo s *Google Play* trgovine. Dakle, preuzeta aplikacija također može postati korisnikov zadani web preglednik, zadana tipkovnica, zadana e-pošta, zadani aplikacija za reprodukciju glazbe i sl. Osim toga aplikacije sustava programerima omogućuju pristup iz vlastite aplikacije u svrhu pružanja željenih funkcionalnosti kao npr. pozivanje aplikacije e-pošte u svrhu slanja e-pošte [9].

3.2. Programski jezik Kotlin

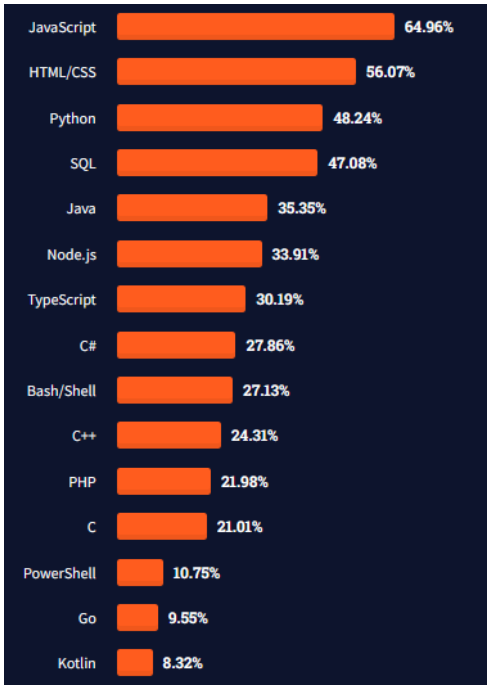
Kotlin je statički tipiziran i objektno orijentiran programski jezik otvorenog koda namijenjen za razvoj višeplatformskih aplikacija. Radi na Java virtualnom stroju (engl. *Java virtual machine*) što omogućuje potpuno kompatibilnost s Java programskim jezikom. Također važno je spomenuti kako pruža višeplatformsku podršku za JavaScript i izvorni kod koristeći LLVM kompajler. Nastao je kao proizvod tvrtke JetBrains, financiran od strane IntelliJ IDEA, a od 2012. godine objavljena je prva verzija dostupna javnosti. Na Google I/O konferenciji 2017. godine Kotlin je prihvaćen kao službeni jezik za razvoj Android aplikacija.

Iako je Android platforma početno napravljena za razvoj aplikacija u Javi to ne predstavlja problem prilikom korištenja programskog jezika Kotlin budući da su međusobno potpuno interoperabilni. Ova osobina omogućuje kombiniranje Kotlin i Java koda. Drugim riječima, Kotlin kod je moguće pokrenuti u Java datoteci i obrnuto, tj. Java kod u Kotlin datoteci [10]. U početku korištenja najviše problema je stvarala brzina kompajliranja te se tako kod koji je koristio prve verzije Kotlina znatno sporije kompajlirao u odnosu na kod pisan u Javi. Taj problem je riješen u novijim verzijama, a danas se u većini slučajeva kod pisan u Kotlinu izvodi brže nego kod pisan u Javi. Razlog tome je što se danas koristi opcija inkrementalnog kompajliranja koristeći Gradle sustav izgradnje (engl. *build system*) koji kompajlira samo novi kod i kod na koji novi kod utječe.

Najvažnije značajke programskog jezika Kotlin: njegove mogućnosti nisu ovisne o verziji Androida ni Jave, jednostavna sintaksa, brže pisanje koda, kraći i čišći kod, jednostavnije i brže ispravljanje grešaka, itd.

Statistike i činjenice o Kotlinu prikupljene na temelju svježih informacija:

1. Kotlin trenutno koristi 8.32% stručnjaka iz industrije (slika 3.4.) [11]:



Sl. 3.4. Lista programskih jezika korišteni za programiranje prema StackOverflow anketi 2021. godine

2. Nalazi se na 18. mjestu najboljih programskih jezika [12]
3. Koriste ga poznate tvrtke Google, Netflix, Amazon, Twitter,…
4. Pinterest i Uber migrirale su postojeće aplikacije s Jave na Kotlin

3.3. Android studio programsko okruženje

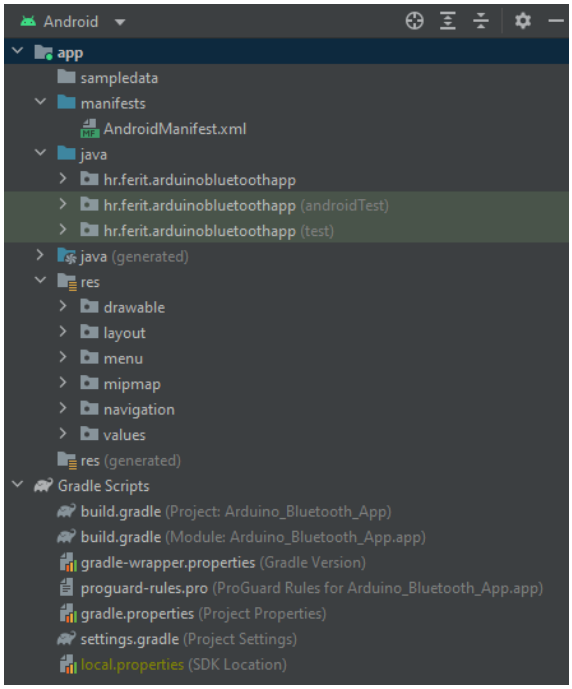
Android Studio je integrirano razvojno okruženje (engl. *Integrated Development Environment – IDE*), izrađeno na postojećem IntelliJ IDEA programskom okruženju, a predstavlja primarni IDE za razvoj aplikacija za Android operacijski sustav. Dostupan je za preuzimanje na Windows, Mac i Linux operacijskom sustavu. Prednosti korištenja Android Studio razvojnog okruženja prilikom razvoja Android aplikacija [13]:

- brz i značajkama bogat emulator
- jednostavno okruženje koje omogućuje razvoj aplikacija za sve Android uređaje
- integracija Gradle sustava za izgradnju
- GitHub integracija koja omogućuje jednostavnije verzioniranje koda
- opsežni alati i okviri za testiranje

- podrška za C++ i NDK (engl. *Native Development Kit*)
- ugrađena podrška za Google Cloud Platform uslugu u oblaku

3.3.1. Struktura projekta

Prema zadanim postavkama, Android Studio prikazuje datoteke projekta u Android prikazu (slika 3.5.). Navedeni prikaz ne odražava stvarnu hijerarhiju datoteka na disku, već je organiziran prema modulima i vrsti datoteka s ciljem jednostavnije navigacije između ključnih datoteka projekta. To se ostvaruje sakrivanjem datoteka koje nije potrebno mijenjati prilikom razvoja aplikacija. Svaki projekt u Android Studio IDE-u sadrži jedan ili više modula koji mogu biti: aplikacijski, bibliotečni i Google App Engine moduli.



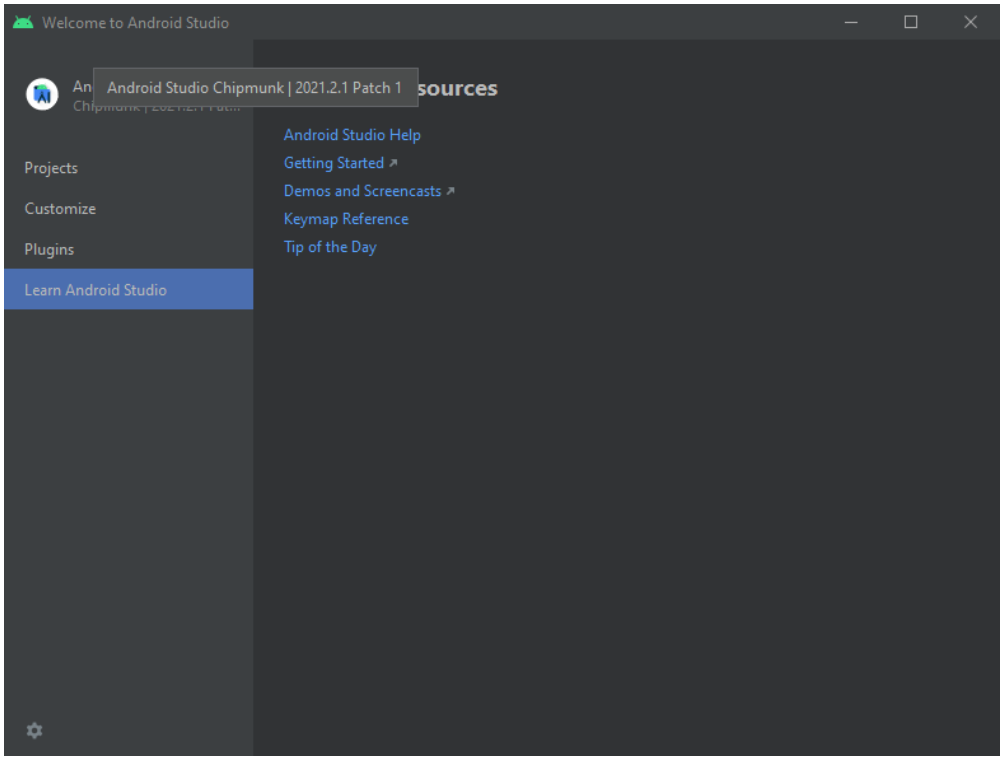
Sl. 3.5. Datoteke projekta u Android prikazu

Unutar svakog aplikacijskog modula, datoteke se prikazuju i sljedećim grupama:

- manifest – sadrži *AndroidManifest.xml* datoteku
- java – sadrži datoteke izvornog koda Java, uključujući JUnit testni kod
- res – sadrži sve resurse koje je potrebno kodirati kao npr. XML izgledi zaslona, UI (engl. *User Interface*) tekst i bitmap slike

3.3.2. Korisničko sučelje

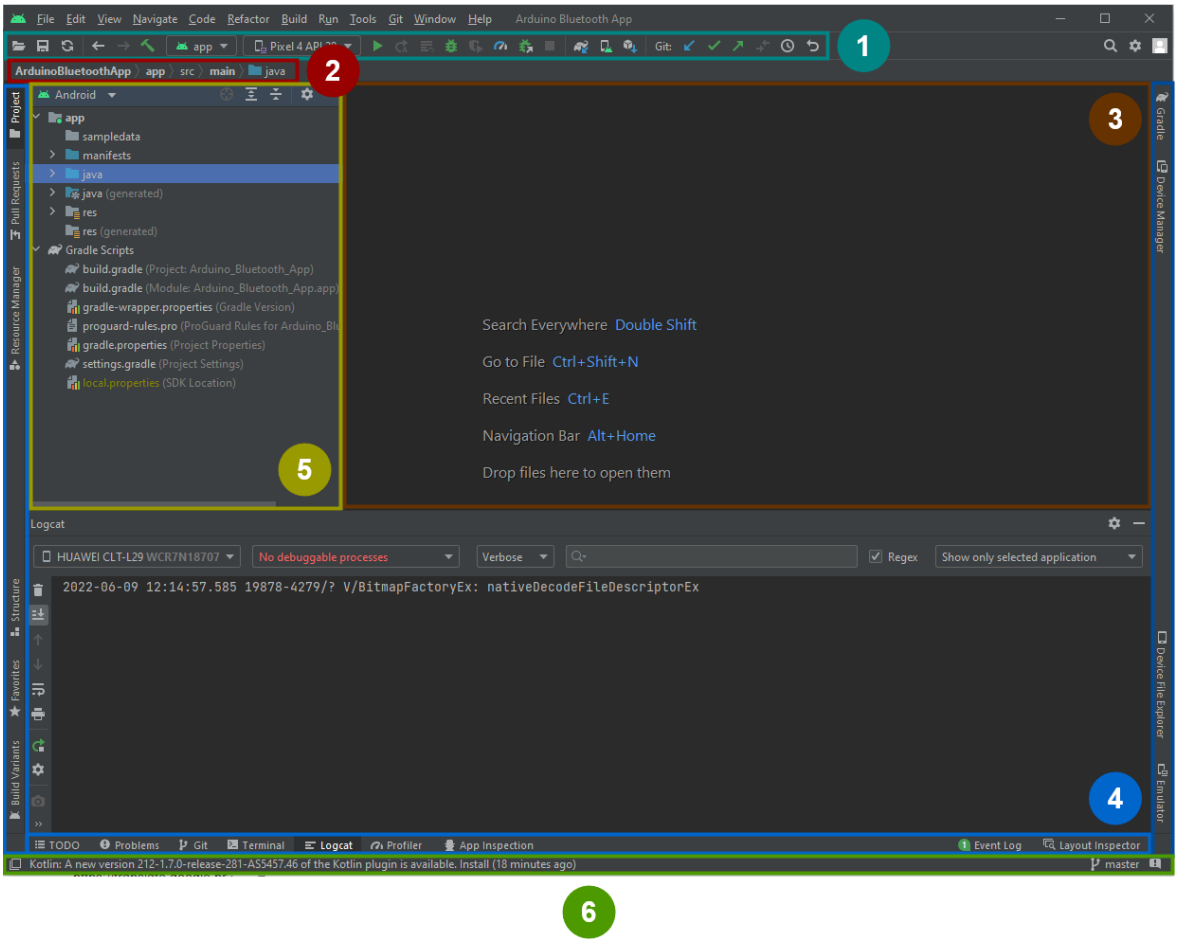
Početni prozor trenutno aktualne verzije Chipmunk - 2021.2.1 Android Studio IDE-a prikazan je na slici 3.6. Izbornik početnog prozora omogućuje korisniku: kreiranje novih i učitavanje postojećih projekata, instalaciju i deinstalaciju raspoloživih dodataka (engl. *plugins*) i izmjenu postavki sučelja kao što su promjena teme, fonta, kombinacije tipki na tipkovnici i sl. Osim toga sadrži i popis korisnih poveznica za pomoć pri korištenju.



Sl. 3.6. Početni prozor Android Studio IDE

Glavni prozor projekta podijeljen je na nekoliko logičkih cjelina (slika 3.7.), a tu su:

1. Alatna traka – prikazuje širok raspon radnji
2. Navigacijska traka – prikazuje putanju do otvorene datoteke
3. Prostor za kreiranje i uređivanje koda – mijenja se ovisno o vrsti datoteke
4. Traka prozora – sadrži gumbe za otvaranje pojedinačnih prozora
5. Alati – pruža pristup određenim zadacima poput upravljanja projektima, pretraživanja, kontrole verzioniranja koda, itd.
6. Statusna traka – prikazuje status i obavijesti projekta i IDE-a te naziv trenutnog *branch-a*



Sl. 3.7. Izgled glavnog prozora projekta u Android Studio IDE

4. RAZVOJ I IMPLEMENTACIJA APLIKACIJE

U ovom poglavlju opisan je proces realizacije aplikacije *Arduino Bluetooth App*. Nadalje, opisan je postupak podešavanja Arduino programa i izrađene aplikacije koji je neophodan za ispravan rad aplikacije. Drugim riječima, kako bi aplikacija radila ispravno potrebno je prilagoditi Arduino program zadanoj konfiguraciji aplikacije ili prilagoditi konfiguraciju aplikacije vlastitom Arduino programu.

4.1. Dijagram tijeka

Dijagram tijeka (engl. *flow chart*) može se definirati kao grafički prikaz algoritma, a koristi se kod projektiranja, dokumentiranja i analiziranja aplikacija i procesa. Dijagrami tijeka pomažu kod opisivanja događaja unutar aplikacije i tako olakšavaju razumijevanje same aplikacije.

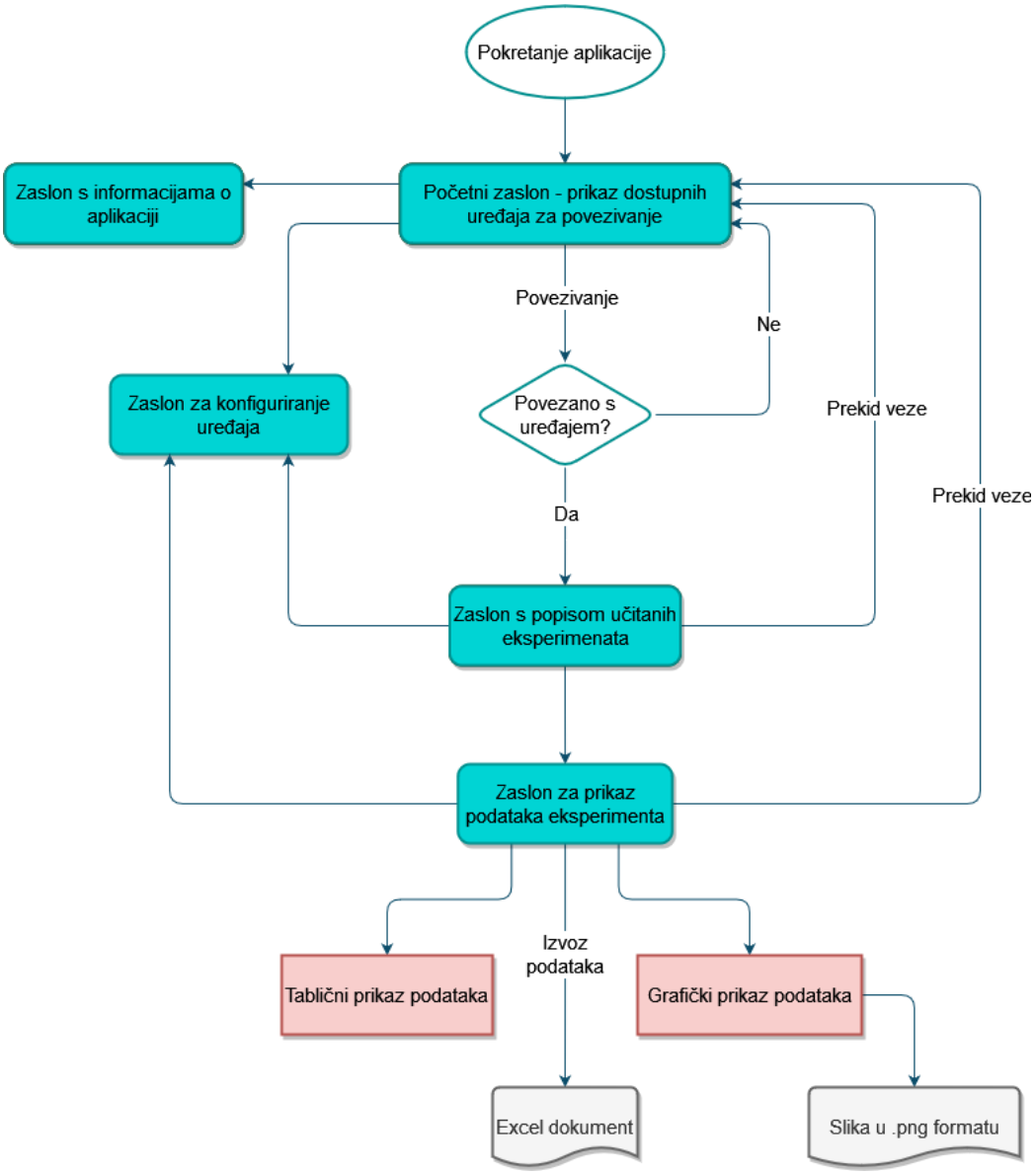
U svrhu jednostavnije implementacije aplikacije izrađen je dijagram tijeka (slika 4.1.). Prilikom crtanja dijagrama korišteni su sljedeći geometrijski simboli:

- elipsa – označava početak aplikacije
- zaobljeni pravokutnik – označava fragment/aktivnost aplikacije
- pravokutnik – označava fragment unutar fragmenta
- romb – označava uvjet odlučivanja
- usmjerena strelica – označava smjer i redoslijed izvođenja aktivnosti/fragmenata

Nakon pokretanja aplikacija otvara se početni zaslon aplikacije s prikazom dostupnih uređaja za povezivanje. Na početnom zaslonu korisnik ima mogućnost povezivanja sa željenim uređajem kao i otvaranje glavnog izbornika. Glavni izbornik sadrži gumb za otvaranje informacija o aplikaciji kao što su verzija, opis, QR kod za dijeljenje i podaci o autoru, te gumb za otvaranje zaslona za konfiguraciju povezanog uređaja. Zaslon za konfiguraciju povezanog uređaja omogućuje korisniku konfiguraciju naredbi, promjenu formata prikaza mjernih jedinica eksperimenta i konfiguraciju kartica na pojedinostima eksperimenta.

Nakon uspješnog povezivanja prikazuje se popis eksperimenata primljenih s uređaja. Korisnik odabirom željenog eksperimenta otvara zaslon s pojedinostima o eksperimentu koji sadrži naziv i opis eksperimenta te kartice koje definiraju prikaz podataka u tabličnom ili grafičkom obliku. Postoji jedna vrsta kartice za tablični prikaz i 3 vrste kartice za grafički prikaz, a to su: kartica za prikaz podataka na linijskom grafu, kartica za prikaz podataka na stupčastom

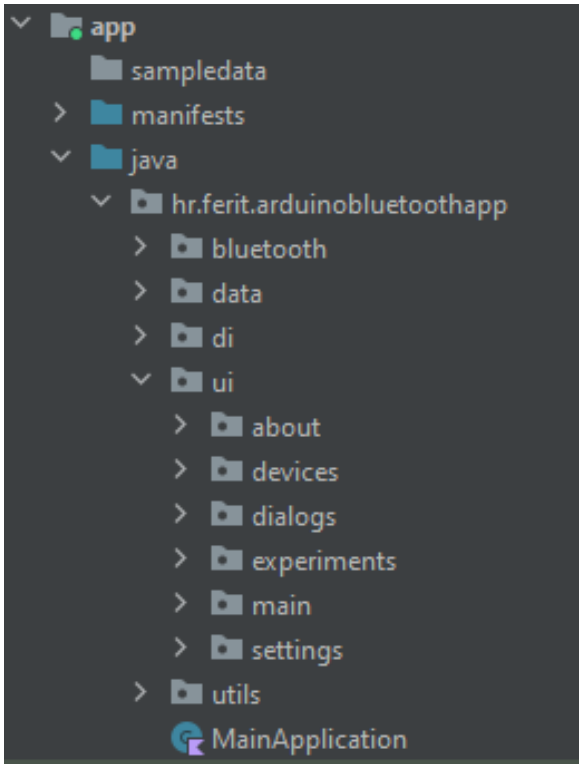
grafu i kartica za prikaz podataka na raspršenom grafu. Na istom zaslonu nalazi se gumb za pokretanje postupka dohvaćanja podataka kao i pomoćni izbornik. Pomoćni izbornik se sastoji od gumba za spremanje grafičkih prikaza u galeriju, gumba za izvoz podataka u Excel datoteku i gumba za prekid veze. Osim izvoza podataka moguće je i njihovo dijeljenje s drugim aplikacijama.



Sl. 4.1. Izgled dijagrama tijeka aplikacije

4.2. Struktura projekta

Prije samog početka razvijanja aplikacije kreirani su osnovni paketi projekta kojima se pojednostavnjuje razvoj aplikacije te omogućuje jednostavnija promjena postojećih i dodavanje novih funkcionalnosti. Osim navedenih karakteristika korištenje paketa omogućuje grupiranje srodnih datoteka te olakšava programerima bržu navigacija kroz projekt.



Sl. 4.2. Izgled strukture projekta

Na slici 4.2. nalazi se struktura projekta *Arduino Bluetooth App* aplikacije iz koje je vidljiva podjela na sljedeće pakete:

- *bluetooth* – sadrži klase i sučelja potrebna za komunikaciju korištenjem *Bluetooth* tehnologije
- *data* – sadrži POJO (engl. *Plain old Java object*) klase koje modeliraju podatke dobivene od uređaja korištenjem *Bluetooth* tehnologije
- *di* – sadrži module za ubrizgavanje ovisnosti (engl. *dependency injection*)
- *ui* – sadrži klase za promjene korisničkog sučelja i potpakete za svaki zaslon u aplikaciji

- *utils* – sadrži pomoćne klase za rad s podacima, objekte s proširenim funkcijama te globalne konstantne potrebne za rad aplikacije

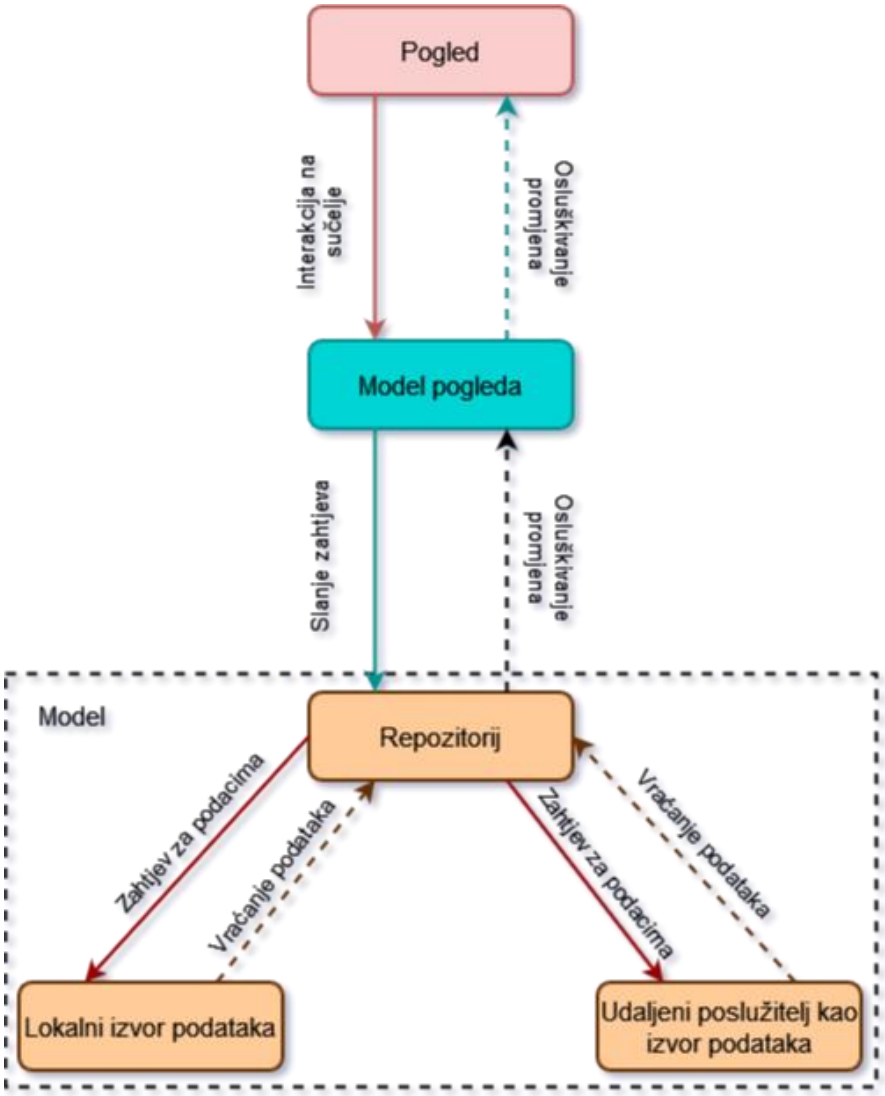
Osim paketa vidljiva je i *MainApplication* klasa aplikacije čiji životni ciklus započinje pokretanjem aplikacije prije prikazivanja prve aktivnosti, a završava kada završi proces koji ju pokreće. Klasa aplikacije sadrži sve komponente potrebne za rad aplikacije. Zadana aplikacijska klasa naziva se *Application*. Promjena zadane aplikacijske klase obavlja se u Android manifest dokumentu, navođenjem putanje do prilagođene odnosno *MainApplication* klase koristeći oznaku *name*.

4.3. Arhitektura projekta

Osim korištenja paketa opisanih u prethodnom potpoglavlju važan faktor u razvijanju kvalitetne aplikacije imaju i arhitekturni obrasci. Arhitekturni obrasci pridonose povećanju skalabilnosti, prilagodljivosti, testabilnosti, održivosti i jednostavnosti.

Prilikom izrade aplikacije korišten je Model-Pogled-Model Pogleda (engl. *Model-View-View Model*, MVVM) arhitekturni obrazac koji razdvaja aplikaciju na više razina prikazanih na slici 4.3.:

- model (engl. *model*) – sadrži podatkovne klase (engl. *data clas*ses) koje pohranjuju podatke s izvora podataka. Izvori podataka s kojih se najčešće primaju podaci su baza podataka ili neki web servis kojemu se pristupa pomoću API metoda, ali izvor podataka može biti i neki vanjski uređaj koji šalje podatke. Model pogleda ne zna s kojega je izvora primio podatke niti na koji način su podaci dohvaćeni.
- model pogleda (engl. *view model*) – posrednički sloj između modela i pogleda. Njegova temeljna zadaća je prilagoditi podatke dobivene od modela u format prikladan za prikaz na korisničkom sučelju. Model pogleda ima zadaću čuvanja stanja o ovisnosti o definiranom životnom ciklusu (engl. *lifecycle*). Životni ciklus modela pogleda može biti od početka aktivnosti kojoj je pridružen pa sve do njezina završetka ili od početka trajanja modela pogleda pa sve do njegova završetka.
- pogled (engl. *view*) – sadrži aktivnosti i fragmente, a zadužen je za praćenje promjene na korisničkom sučelju u ovisnosti o kojima se pozivaju metode u modelu pogleda. Pogled također osluškuje promjene u modelu pogleda te ako dođe do promjene poziva neku metodu iz modela pogleda čiji odgovor vrši promjene na korisničkom sučelju.



Sl. 4.3. Prikaz MVVM arhitekture projekta

Kao što je vidljivo sa slike 4.3., svaka razina vrši interakciju na nižu razinu i osluškuje promjene tj. pogled vrši interakciju na pogled modela, pogled modela vrši interakciju na model, a model vrši interakciju na izvore podataka [14].

4.4. Korištene biblioteke

Ponovna upotrebljivost jednom napisanog koda česta je pojava svakog programera. Ako programer provede neko vrijeme rješavajući problem i shvati kako njegovo rješenje može pomoći drugima, izrada Android biblioteke najbolji je način pakiranja koda za ponovnu upotrebu. Sama činjenica da su osnovni dijelovi Android platforme smješteni u biblioteke govori dovoljno o njihovom značaju.

Biblioteke su neophodan alat za razvoj Android aplikacija jer uvelike olakšavaju postupak razvoja aplikacije. Značajnije biblioteke korištene za potrebe ovog diplomskog rada su:

- Dagger Hilt [15] – standardni način za ubrizgavanje ovisnosti kod razvoja Android aplikacija. Njezinom primjenom smanjuje se potreba pisanja koda koji se stalno ponavlja (engl. *biolerplate code*) prilikom ubrizgavanja ovisnosti. Izgrađen je na temelju biblioteke *Dagger* [16] čiju upotrebu znatno pojednostavljuje.
- RxJava [17] – predstavlja Java implementaciju *ReactiveX* API-a koji omogućuje ulančavanje asinkronih zadataka i događaja u sekvence koje se promatraju (engl. *observable sequences*). Izuzetno korisna biblioteka za dohvaćanje podataka u stvarnom vremenu, povezivanje višestukih poziva, prebacivanje podataka između niti i rukovanje pogreškama.
- MPAndroid Chart [18] – najpopularnija besplatna biblioteka za vizualizaciju podataka korištenjem grafova. Sadrži metode za izradu osnovnih, ali i dodatnih grafova. Omogućuje dinamičko dodavanje točaka i skupova podataka u stvarnom vremenu.
- Splash screen [19] – omogućuje prikaz pozdravnog zaslona prilikom pokretanja aplikacije. Pruža niz metoda za prilagođeni prikaz zaslona kao npr. metodu za promjenu ikonice, pozadine, vremena trajanja animacije i sl.
- Apachi POI [20] – omogućuje stvaranje, prikaz i izmjenu Microsoft Office datoteka korištenjem klasa i metoda pisanih u Java programskom jeziku. Sadrži klase i metode za rad sa svim dokumentima Microsoft Office paketa. Neke komponente ove biblioteke su: HSSF (engl. Horrible Spreadsheet Format) – za manipulaciju .xls datoteka, XSSF (engl. XML Spreadsheet Format) – za manipulaciju .xlsx datoteka, HWPF (engl. Horrible Word Processor Format) – za manipulaciju .doc datoteka, XWPF (XML Word Processor Format) – za manipulaciju .docx datoteka,…

4.5. Podešavanje Arduino programa

Prije početka korištenja *Arduino Bluetooth App* aplikacije potrebno je ostvariti logiku opisanu u daljnjem tekstu.

4.5.1. Definiranje osnovne strukture

Osnovna struktura Arduino programa prikazana je na slici 4.4., a uključuje potrebne biblioteke, njihovu inicijalizaciju te osnovne naredbe.

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>

String cmd;
const byte rxPin = 1;
const byte txPin = 0;
SoftwareSerial bluetoothSerial(rxPin, txPin);

void setup() {
    pinMode(rxPin, INPUT);
    pinMode(txPin, OUTPUT);

    bluetoothSerial.begin(9600);
    cmd = "";
}

void loop() {
    while (!bluetoothSerial.available());
    cmd = bluetoothSerial.readString();

    if (cmd == "dohvati_konfiguraciju") {
        //Ovdje se definira konfiguracija eksperimenata

        bluetoothSerial.print("akcija_dovrsena\n");
    }
    else if (cmd == "dohvati_podatke id_eksperimenta") {
        //Ovdje se formatiraju podaci za slanje

        bluetoothSerial.print("akcija_dovrsena\n");
    }
    else {
        bluetoothSerial.print("akcija_ne_postoji\n");
    }
}
```

Sl. 4.4. Osnovna struktura Arduino programa

Na samom početku programa potrebno je uključiti *SoftwareSerial* i *ArduinoJson* biblioteke. *SoftwareSerial* biblioteka pruža funkcije za serijsku komunikaciju koristeći digitalne

pinove Arduino mikroupravljača, dok *ArduinoJson* biblioteka pruža funkcije za serijalizaciju podataka u JSON (engl. *Java Script Object Notation*) prikaz podataka i deserijalizaciju iz JSON prikaza podataka u željenu strukturu podataka. Nakon uključivanja biblioteka radi se inicijalizacija *SoftwareSerial* objekta čiji konstruktor prima pin za primanje podataka (*rxPin*) i pin za slanje podataka (*txPin*).

U *setup* funkciji potrebno je definirati pin za primanje podataka kao ulazni, a pin za slanje podataka kao izlazni te definirati brzinu prijenosa podataka (engl. *baud rate*).

U *loop* funkciji kontinuirano se dohvaća broj bajtova dostupnih za čitanje sa serijskog priključka odnosno provjerava se da li je neki podatak primljen. Ako je podatak primljen, on se u idućem koraku sprema kao niz znakova u varijablu *cmd* koja predstavlja naredbu na temelju koje se provode daljnje akcije. Na kraju svake akcije potrebno je poslati naredbu koja označava završetak iste. Svaki Arduino programa mora sadržavati 2 osnovne naredbe: naredbu za dohvaćanje konfiguracije dostupnih eksperimenata i naredbu za dohvaćanje podataka željenog eksperimenta.

4.5.2. Slanje konfiguracije dostupnih eksperimenata

Na slici 4.5. prikazan je primjer akcije koja definira konfiguraciju jednog eksperimenta. Navedena akcija radi serijalizaciju podataka u JSON prikaz podataka odnosno kreira niz koji sadrži konfiguracije eksperimenata. Svaku konfiguraciju čine sljedeće varijable:

- jedinstvena oznaka eksperimenta (*id*): cjelobrojna vrijednost
- načini prikaza eksperimentalnih podataka (*tabTypes*): polje niza znakova
- naziv eksperimenta (*name*): niz znakova
- opis eksperimenta (*desc*): niz znakova
- naziv mjerne veličine na x-osi (*xAxisLabel*): niz znakova
- naziv mjerne veličine na y-osi (*yAxisLabel*): niz znakova
- mjerna jedinica na x-osi (*xAxisUnit*): niz znakova
- mjerna jedinica na y-osi (*yAxisUnit*): niz znakova
- nazivi skupova podataka (*dataSetNames*): polje niza znakova

Načini prikaza eksperimentalnih podataka definiraju način na koji se podaci dobiveni s mjernog uređaja prikazuju u Android aplikaciji. Android aplikacija ima mogućnost prikazivanja podataka u tablici i grafički na linijskom, stupčastom i raspršenom grafu. Kako bi aplikacija znala

povezati primljene vrijednost sa željenom vrstom prikaza potrebno je primljene vrijednosti pridružiti određenom načinu prikaza. Postupak pridruživanja objašnjen je u potpoglavlju Podešavanje Android aplikacije.

Nazivi skupova podataka omogućuju korisniku pridruživanje željenih imena određenom skupu podataka.

```
if (cmd == "dohvati_konfiguraciju") {
    StaticJsonDocument<512> experimentsConfig;

    JsonObject experiment_0 = experimentsConfig.createNestedObject();
    experiment_0["id"] = "0";

    JsonArray experiment_0_tabTypes = experiment_0.createNestedArray("tabTypes");
    experiment_0_tabTypes.add("tablica");
    experiment_0_tabTypes.add("linijski_graf");
    experiment_0["name"] = "Eksperiment 1";
    experiment_0["desc"] = "Prikaz ovisnost promjene brzine o vremenu.";
    experiment_0["xAxisLabel"] = "Vrijeme";
    experiment_0["yAxisLabel"] = "Brzina";
    experiment_0["xAxisUnit"] = "ms";
    experiment_0["yAxisUnit"] = "km/h";
    experiment_0["dataSetNames"][0] = "1. mjerenje";

    bluetoothSerial.print(cmd);
    serializeJson(experimentsConfig, bluetoothSerial);
    bluetoothSerial.print("\n");

    bluetoothSerial.print("akcija_dovrsena\n");
}
```

Sl. 4.5. Primjer akcije za konfiguraciju eksperimenta

Iz konfiguracije sa slike 4.5. vidljivo je kako su podaci za navedeni eksperimenta prikazani u tablici i na linijskom grafu. Nadalje, eksperiment sadrži samo jedan skup podataka, naziva 1. mjerenje“. Ostale vrijednost eksperimenta su:

- jedinstveni id eksperimenta: 0
- naziv eksperimenta: Eksperiment 1
- opis eksperimenta: Prikaz ovisnosti promjene brzine o vremenu
- naziv mjerne veličine i mjerne jedinice na x-osi: Vrijeme (ms)
- naziv mjerne veličine i mjerne jedinice na y-osi: Brzina (km/h)

Izgled JSON prikaza podataka dobivenog nakon primjene funkcije *serializeJson* prikazan je na slici 4.6.. Kako bi Android aplikacija znala napraviti deserijalizaciju potrebno je poslati redom: naziv primljene naredbe, kreirani JSON prikaz podataka i oznaku za novi red. Na kraju akcije potrebno je obavijestiti aplikaciju da je akcija izvršena, a to se postiže slanjem naredbe koja označava kraj akcije.

Sl. 4.6. Izgled konfiguracije u JSON prikazu podataka

4.5.3. Slanje podataka primljenih od mjernog uređaja

Na slici 4.7. prikazan je primjer akcije koja simulira prikupljanje podataka s mjernog uređaja. Podaci su povezani su s konfiguracijom eksperimenta preko njegovog id-a. Dopušteni formati za slanje podataka su:

1. Format s vremenom uzorkovanja: <indeks podatka, vrijeme uzorkovanja, id skupa podataka>y vrijednost
2. XY format: <indeks podatka, id skupa podataka>x vrijednost, y vrijednost

Primjer na navedenoj slici koristi format s vremenom uzorkovanja za slanje podataka što znači da mikroupravljač šalje samo y vrijednost, dok se x vrijednost dobije na temelju indeksa podatka i vremena uzorkovanja.

Simulacija pristizanja podataka realizirana je pomoću *for* petlje unutar koje se generiraju i šalju podaci s razmakom od 30ms. Nakon što se pošalju svi podaci potrebno je obavijestiti aplikaciju da je akcija završena isto kao i kod akcije opisane u prethodnom potpoglavlju.

```
if (cmd == "dohvati_podatke 0") {
    int i, dataSetId = 0;
    float samplingTime = 1.0;
    for (i = 1; i <= 50; i++)
    {
        float yValue = 0.5 + (i + 1);
        bluetoothSerial.print("<");
        bluetoothSerial.print(i);
        bluetoothSerial.print(",");
        bluetoothSerial.print(samplingTime);
        bluetoothSerial.print(",");
        bluetoothSerial.print(dataSetId);
        bluetoothSerial.print(">");
        bluetoothSerial.print(yValue);
        bluetoothSerial.print("\n");

        delay(30);
    }

    bluetoothSerial.print("akcija_dovrsena\n");
}
```

Sl. 4.7. Primjer akcije za dohvaćanje podataka

Izgled primljenih podataka nakon pozivanja akcije za dohvaćanje eksperimenta id 0 prikazan je na slici 4.8.

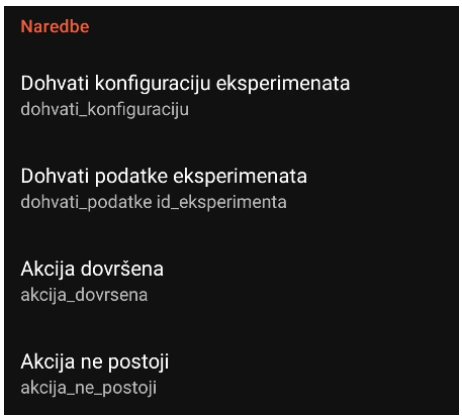
```
<1,1.00,0>2.50
<2,1.00,0>3.50
<3,1.00,0>4.50
<4,1.00,0>5.50
<5,1.00,0>6.50
<6,1.00,0>7.50
<7,1.00,0>8.50
<8,1.00,0>9.50
<9,1.00,0>10.50
<10,1.00,0>11.50
<11,1.00,0>12.50
<12,1.00,0>13.50
<13,1.00,0>14.50
<14,1.00,0>15.50
<15,1.00,0>16.50
        .
        .
        .
<50,1.00,0>51.50
```

Sl. 4.8. Izgled primljenih podataka korištenjem formata s vremenom uzorkovanja

4.6. Podešavanje Android aplikacije

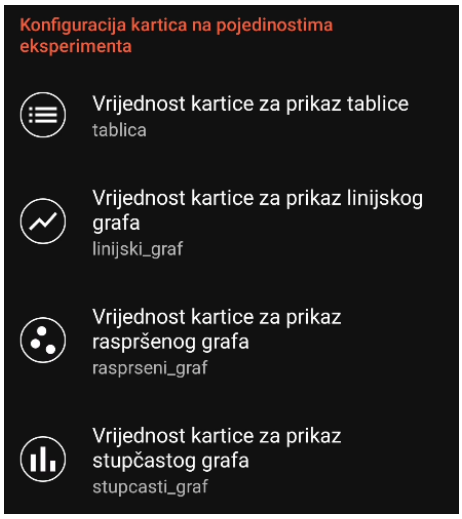
Nakon što je dovršeno podešavanje programa za Arduino mikroupravljač, potrebno je podesiti *Arduino Bluetooth App* aplikaciju kako bi znala komunicirati s povezanim uređajem. Podešavanje se obavlja na zaslonu „Postavke“ gdje je moguće promijeniti osnovne naredbe te povezati način prikazivanja podataka s vrijednostima poslanim u konfiguraciji eksperimenta.

Izgled postavki osnovnih naredbi nakon podešavanja na temelju programa (slika 4.4.) napisanog u potpoglavlju Definiranje osnovne strukture vidljiv je na slici 4.9.



Sl. 4.9. Izgled postavki osnovnih naredbi

Izgled postavki nakon promjene vrijednosti prema vrsti prikaza podataka u konfiguraciji eksperimenta (slika 4.5.) vidljiv je na slici 4.10.



Sl. 4.10. Izgled postavki vrijednosti za prikaz podataka

Aplikacija ima zadane vrijednosti osnovnih naredbi, kao i zadane vrijednosti koje povezuju način prikazivanja podataka s konfiguracijom što znači da nije nužno mijenjati zadane vrijednosti ako se one koriste prilikom pisanja Arduino programa. Drugim riječima, ako se prilikom pisanja Arduino programa koriste zadane naredbe:

- /config – za dohvaćanje konfiguracije eksperimenata
- /data experiment_id – za dohvaćanje podataka eksperimenta
- /done – za obavještenje aplikacije o kraju akcije
- /no exist – za obavještenje aplikacije o nepostojanju akcije

te ako se koriste zadane vrijednosti za povezivanje načina prikaza podataka u konfiguraciji eksperimenta:

- TABLE – za prikaz podataka u tablici
- LINE – za prikaz podataka na linijskom grafu
- SCATTER – za prikaz podataka na raspršenom grafu
- BAR – za prikaz podataka na stupčastom grafu

nije potrebno dodatno podešavati aplikaciju.

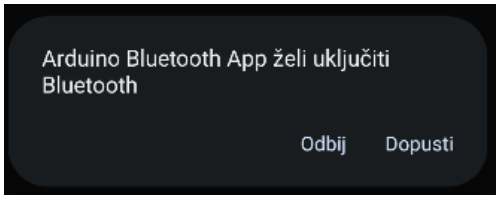
5. PRIKAZ RADA APLIKACIJE

Korisničko sučelje aplikacija *Arduino Bluetooth App* podržava tamni (engl. *dark*) i svijetli (engl. *light*) način rada te hrvatski i engleski jezik. Obje značajke se postavljaju automatski ovisno o postavkama Android sustava. Pokretanjem aplikacije prikazuje se pozdravni zaslon (engl. *splash screen*) s logom aplikacije (slika 5.1.) koji traje kratko vrijeme nakon kojega se prikazuje početni zaslon aplikacije odnosno lista uređaja dostupnih za povezivanje korištenjem *Bluetooth* tehnologije.

Sl. 5.1. Izgled pozdravnog zaslona

5.1. Zaslon s popisom uređaja dostupnih za povezivanje (početni zaslon)

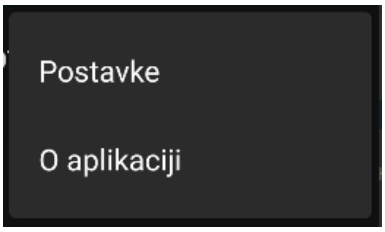
Kako bi aplikacija prikazala popis uređaja dostupnih za povezivanje potrebno je posjedovati mobilni uređaj koji podržava značajku *Bluetooth*, ali isto tako *Bluetooth* mora biti uključen. Ako *Bluetooth* nije uključen, aplikacija automatski traži njegovo uključivanje (slika 5.2.). Dodatno, na mobilnim uređajima s Android sustavom 31 ili novijim potrebno je dati dopuštenje *Arduino Bluetooth App* aplikaciji da traži uređaje u blizini, poveže se s njima i odredi njihov približni položaj (slika 5.3).



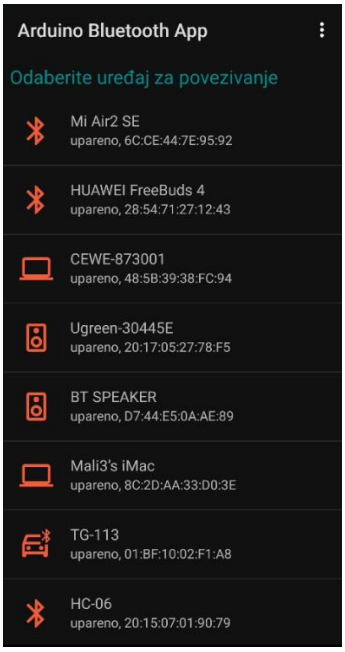
Sl. 5.2. Izgled prozora za uključivanje značajke Bluetooth

Sl. 5.3. Izgled prozora za davanje dopuštenja aplikaciji

Početni zaslon se sastoji od glavnog izbornika (slika 5.4.) i popisa uređaja dostupnih za povezivanje (slika 5.5.). Osvježivanje popisa dostupnih uređaja ostvaruje se prevlačenjem popisa prema dolje. Pritiskom na gumb „Postavke“ iz glavnog izbornika otvara se Zaslon postavki, dok se pritiskom na gumb „O aplikaciji“ otvara Zaslon s informacijama o aplikaciji.

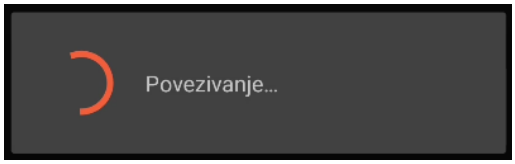


Sl. 5.4. Izgled glavnog izbornika



Sl. 5.5. Izgled početnog zaslona

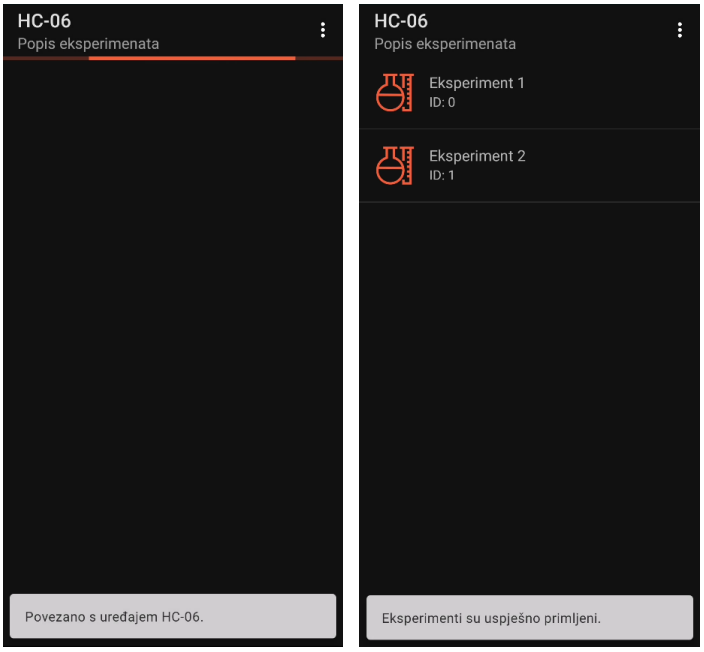
Povezivanje s uređajem obavlja se pritiskom na željeni uređaj pri čemu se otvara prozor s indikatorom napretka i porukom obavijesti (slika 5.6.).



Sl. 5.6. Izgled prozora za povezivanje

5.2. Zaslon s popisom eksperimenata

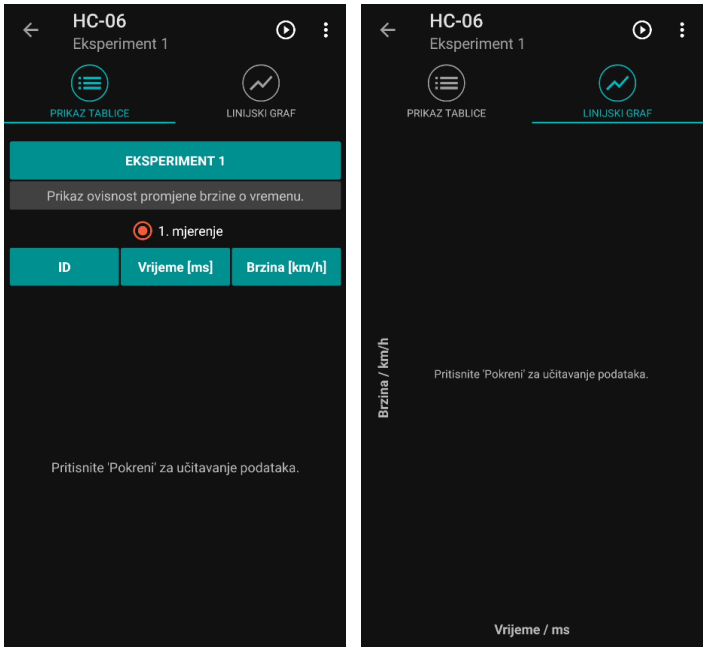
Nakon uspješnog povezivanja s uređajem, uređaju se šalje naredba za dohvaćanje konfiguracije eksperimenata pri čemu aplikacija započinje postupak dohvaćanja dostupnih eksperimenata (slika 5.7.). Prekid veze s uređajem ostvaruje se pritiskom na navigacijski gumb za povratak ili pritiskom na gumb „Prekid veze“ iz izbornika. Pritiskom na eksperiment iz popisa eksperimenata otvaraju se pojedinosti odabranog eksperimenta.



Sl. 5.7. Prikaz postupka dohvaćanja eksperimenata

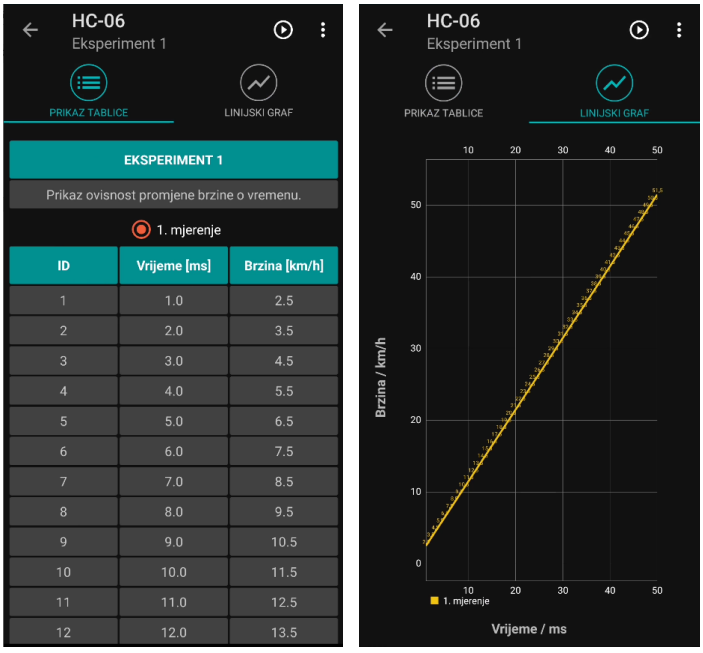
5.3. Zaslon s prikazom pojedinosti o eksperimentu

Nakon ulaska u pojedinosti eksperimenta prikazuju se kartice (engl. *tabs*) definirane u konfiguraciji odabranog eksperimenta (slika 5.8.). Budući da je u konfiguraciji (slika 4.5.) navedeno kako se podaci trebaju prikazati u tablici i na linijskom grafu, prikazuju se samo te dvije kartice. Osim kartica, postavlja se zaglavlje tablice kod tabličnog te x i y osi grafa kod grafičkog prikaza. Promjena prikaza podataka moguća je pritiskom na ikonicu ili naziv željene kartice te pomicanjem lijevo-desno (engl. *swipe*).



Sl. 5.8. Prikaz pojedinosti odabranog eksperimenta

S obzirom na to da podaci još uvijek nisu dohvaćeni, aplikacija prikazuje samo poruku za dohvaćanje podataka. Postupak dohvaćanja i vizualizacije podataka prikazan je na slici 5.9., a započinje pritiskom na gumb „Pokreni“ (slika 5.10.). Vizualizacija podataka se vrši u trenutku pristizanja svakog podatka kod prikaza na linijskom i raspršenom grafu, dok se kod prikaza u tablici i na stupčastom grafu vizualizacija vrši nakon što svi podaci pristignu.

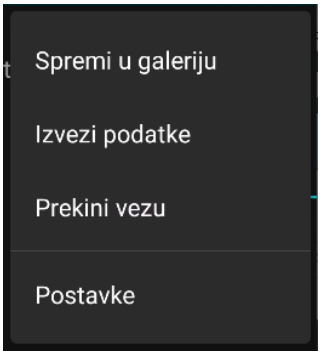


Sl. 5.9. Prikaz postupka dohvaćanja i vizualizacije podataka

Sl. 5.10. Izgled ikonice za dohvaćanje podataka

Aplikacija radi grupaciju dohvaćenih podataka prema skupovima podataka tj. kod tabličnog prikaza svaki skup podataka nalazi se u zasebnoj tablici, dok se kod grafičkog prikaza svaki skup podataka označuje drugom bojom. Prelaskom na zaslon s prikazom pojedinosti o eksperimentu prikazuje se novi izbornik (slika 5.11.), drugačiji od izbornika na prethodnom zaslonu. Novo prikazani izbornik omogućuje:

- spremanje slike grafa u galeriju
- izvoz podataka u Excel dokument i njegovo dijeljenje
- prekidanje veze s uređajem
- promjenu konfiguracije uređaja



Sl. 5.11. Izgled novo prikazanog izbornika

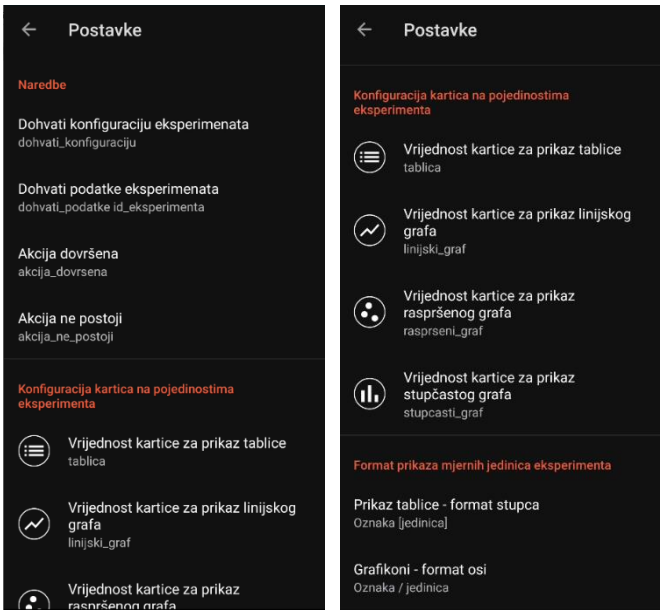
5.4. Zaslon postavki

Izgled zaslona Postavke prikazan je na slici 5.12., a sadrži sljedeće 3 kategorije: naredbe, konfiguracije kartica na pojedinostima eksperimenta i formata prikaza mjernih jedinica eksperimenta.

Kategorija naredbe sadrži opcije za promjenu naredbi koje se šalju uređaju s kojim je aplikacija povezana u svrhu dohvaćanja konfiguracije eksperimenata i njihovih podataka. Osim promjene naredbi koje aplikacija šalje moguća je i promjena naredbi koje aplikacija prima. Drugim riječima, aplikacija pruža mogućnost promjene naredbe koja označava kraj izvođenja trenutne akcije na povezanom uređaju, ali i promjenu naredbe koja označava da tražena akcija ne postoji.

Kategorija za konfiguraciju kartica na pojedinostima eksperimenta sadrži vrijednosti koje povezuju način prikazivanja podataka s konfiguracijom promatranog eksperimenta. Drugim riječima, korisnik prilikom pisanja konfiguracije u Arduino programu određuje način na koji se podaci prikazuju u aplikaciji.

Kategorija za promjenu formata prikaza mjernih jedinica eksperimenta određuje način prikaza naziva jedinica u odnosu na mjernu jedinicu. Korisnik može postaviti drugačiji format za prikaz u tablici i drugačiji format za prikaz na osi-ima grafa.



Sl. 5.12. Izgled zaslona Postavke

5.5. Zaslon s informacijama o aplikaciji

Izgled zaslona „O aplikaciji“ prikazan je na slici 5.13., a sadrži osnovne informacije o aplikaciji odnosno logo, verziju, build i kratki opis aplikacije te QR kod s poveznicom na link za preuzimanje aplikacije. Također sadrži i ikonicu za prikaz LinkedIn profila razvojnog programera aplikacije.

Sl. 5.13. Izgled zaslona s informacijama o aplikaciji

6. ZAKLJUČAK

Izvođenje fizikalnih mjerenja neophodno je za svladavanje i razumijevanje fizike i njezinih zakonitosti. Rezultati mjerenja u većini slučajeva ne daju dovoljno informacija već je potrebna dodatna obrada i vizualizacija. Zbog težnje pojednostavljivanja svakodnevnih obveza korištenjem informatičkih uređaja javila se potreba za digitalizacijom rezultata, a samim time i potreba za izradom programa i aplikacija za njihovu obradu i prikaz.

Fokusirajući se na izvođenje fizikalnih mjerenja na laboratorijskim vježbama iz kolegija Fizike, zadatak ovog diplomskog rada je razvoj aplikacije koja olakšava i ubrzava postupak očitavanju rezultata mjerenja, ali i vizualizacije podataka. Osim studentima iz Hrvatske aplikacija će pomoći i studentima drugih država budući da podržava sučelje na hrvatskom i engleskom jeziku. Razvijena aplikacija u konačnici osigurava kraće vrijeme provedbe eksperimenata na laboratorijskim vježbama. Kroz rad su prikazana trenutna rješenja mobilnih aplikacija koja rješavaju navedeni problem, analizirane su tehnologije korištene pri razvoju aplikacije te je opisan proces realizacije i načina rada aplikacije.

Trenutno korištena biblioteka za crtanje grafova nije potpuno optimizirana i ne podržava sve varijante rada s podacima koji pristižu u stvarnom vremenu. Kako bi se aplikacija dodatno unaprijedila potrebno je izraditi vlastiti biblioteku optimiziranu za crtanje grafova u stvarnom vremenu ili zamijeniti trenutnu biblioteku drugom, prilagođenom za rad u stvarnom vremenu.

LITERATURA

[1] „Supported sensors“, *phyphox*. https://phyphox.org/sensors/ (pristupljeno 25. svibanj 2022.).
[2] „Export formats“, *phyphox*. https://phyphox.org/export-formats/ (pristupljeno 25. svibanj 2022.).
[3] „Arduino library“, *phyphox*. https://phyphox.org/arduino/ (pristupljeno 25. svibanj 2022.).
[4] P. kroz prozor, „Eksperimenti“, *Pogled kroz prozor*, 31. listopad 2021. https://pogledkrozprozor.wordpress.com/2021/10/31/eksperimenti/ (pristupljeno 02. ožujak 2022.).
[5] „Arduino Science Journal“. https://www.arduino.cc/education/science-journal (pristupljeno 25. svibanj 2022.).
[6] „Mobile Operating System Market Share Worldwide“, *StatCounter Global Stats*. https://gs.statcounter.com/os-market-share/mobile/worldwide (pristupljeno 07. lipanj 2022.).
[7] „Android version history“, *Wikipedia*. 02. lipanj 2022. Pristupljeno: 08. lipanj 2022. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Android_version_history&oldid=1091144965
[8] „Mobile Android Version Market Share Worldwide“, *StatCounter Global Stats*. https://gs.statcounter.com/android-version-market-share/mobile/worldwide/ (pristupljeno 07. lipanj 2022.).
[9] „Platform Architecture | Android Developers“. https://developer.android.com/guide/platform (pristupljeno 07. lipanj 2022.).
[10] „Uvod u Kotlin — Koje su prednosti novog zvaničnog jezika za razvoj Android aplikacija?“, *Startit*, 16. lipanj 2017. https://startit.rs/uvod-u-kotlin-koje-su-prednosti-zvanicnog-jezika-za-razvoj-android-aplikacija/ (pristupljeno 08. lipanj 2022.).
[11] „Stack Overflow Developer Survey 2021“, *Stack Overflow*. https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021 (pristupljeno 08. lipanj 2022.).
[12] S. O'Grady, „The RedMonk Programming Language Rankings: January 2022“, *tecosystems*, 28. ožujak 2022. https://redmonk.com/sogrady/2022/03/28/language-rankings-1-22/ (pristupljeno 08. lipanj 2022.).
[13] „Meet Android Studio“, *Android Developers*. https://developer.android.com/studio/intro (pristupljeno 09. lipanj 2022.).
[14] „Uvod u MVVM arhitekturu“, *Web Programiranje*, 31. srpanj 2019. https://www.webprogramiranje.org/uvod-u-mvvm-arhitekturu/ (pristupljeno 04. srpanj 2022.).
[15] „Hilt“. https://dagger.dev/hilt/ (pristupljeno 09. srpanj 2022.).
[16] „Dagger“. https://dagger.dev/ (pristupljeno 09. srpanj 2022.).
[17] *RxAndroid: Reactive Extensions for Android*. ReactiveX, 2022. Pristupljeno: 09. srpanj 2022. [Na internetu]. Dostupno na: https://github.com/ReactiveX/RxAndroid
[18] P. Jahoda, *MPAndroidChart*. 2022. Pristupljeno: 09. srpanj 2022. [Na internetu]. Dostupno na: https://github.com/PhilJay/MPAndroidChart
[19] „Splash screens“, *Android Developers*. https://developer.android.com/guide/topics/ui/splash-screen (pristupljeno 09. srpanj 2022.).
[20] „Apache POI - the Java API for Microsoft Documents“. https://poi.apache.org/ (pristupljeno 09. srpanj 2022.).

SAŽETAK

Primarni cilj ovog diplomskog rada je razvoj Android aplikacije koja olakšava učenicima i studentima praćenje te razumijevanje eksperimentalnih mjerenja iz područja fizike. Predstavljeno je moderno rješenje, jednostavnog sučelja koje omogućuje povezivanje na uređaj korištenjem *Bluetooth* tehnologije, dohvaćanje podataka s povezanog uređaja te vizualizaciju dohvaćenih podataka u tabličnom i grafičkom obliku. Također, omogućen je izvoz podataka dobivenih mjerenjem radi daljnje obrade i analize. Aplikacija *Arduino Bluetooth App* razvijena je u Android Studio razvojnom okruženju koristeći programski jezik Kotlin i MVVM arhitekturni obrazac. Razvijena aplikacija omogućuje učinkovito dohvaćanje podataka s povezanog uređaja, njihovu deserijalizaciju i vizualizaciju u stvarnom vremenu. Sučelje aplikacija za vizualizaciju podataka je konfigurabilno tj. ovisi o postavkama mobilne aplikacije i konfiguraciji dohvaćenoj s povezanog uređaja.

Ključne riječi: Android, Bluetooth, fizikalna mjerenja, vizualizacija podataka

ABSTRACT

MOBILE APPLICATION FOR VISUALIZATION OF MEASUREMENTS

ŽIVOTOPIS

Autor ovo diplomskog rada, Dominik Tkalčec, rođen je 13. listopada 1997. godine u Virovitici. Odrastao je u malenom selu Otrovanec, pokraj Virovitice gdje živi i danas. Pohađao je osnovnu škola „Petra Preradovića“ u Pitomači. Nakon osnovne škole upisuje Tehničku školu u Virovitici te 2016. godine stječe zvanje „Računalni tehničar za strojarstvo“. Iste godine sudjeluje na Županijskom natjecanju učenika strojarskih zanimanja u kategoriji „CNC tehnologije – glodanje“ gdje se uspješno plasirao na Državno natjecanje. Maturirao je 2016. godine, nakon čega upisuje „Stručni sveučilišni studij Informatike“ na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Na posljednjoj godini studija radi kao demonstrator na kolegijima „Arhitektura računalnih sustava“ i „Programiranje II“. Stručnu praksu odradio je 2019. godine u tvrtki Atos za *back-end* developer-a. Iste godine završava upisani studij te stječe zvanje „Stručni prvostupnik inženjer elektrotehnike“. Daljnje visokoškolsko obrazovanje nastavlja upisom Razlikovne godine za prijelaz sa stručnog na diplomski sveučilišni studij. U 8. mjesecu 2020. godine počinje raditi kao Junior Android developer u tvrtki GDi. Iste godine nastavlja obrazovanje na diplomskom studiju Računarstva, smjer Programsko inženjerstvo. Svoje obrazovanje na diplomskom studija Računarstva završava 2022. godine.

_____

Potpis autora