Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Lenarčič

Mobilna aplikacija za upravljanje metronoma

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00330/2012
Datum: 04.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATEJ LENARČIČ**

Naslov: **MOBILNA APLIKACIJA ZA UPRAVLJANJE METRONOMA**
MOBILE APPLICATION TO CONTROL METRONOME

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Metronom je naprava, ki glasbenikom pomaga s produciranjem konstantnega števila udarcev na minuto. Izdelajte načrt za metronom in ga na podlagi načrta tudi izdelajte. Metronom naj upravlja mobilna aplikacija platforme Android, ki preko vmesnika USB upravlja z metronomom. Uporabite vmesnik IOIO, za prenos podatkov pa protokola ADB (Android Debug Bridge) in OpenAccessory

Mentor:
doc. dr. Rok Rupnik

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matej Lenarčič, z vpisno številko **63060151**, sem avtor diplomskega dela z naslovom:

Mobilna aplikacija za upravljanje metronoma

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. marca 2013 Podpis avtorja:

Kazalo

Povzetek

Pri igranju glasbenega inštrumenta, posebno bobnov, je zelo pomemben natančen tempo izvajanja. Za to glasbeniki uporabljajo napravo, imenovano metronom, ki daje konstanten tempo, glede na željeno število udarcev na minuto. V diplomski nalogi je razvit metronom na platformi Android z zunanjo enoto, ki je povezana z mobilno napravo prek vmesnika USB in nam omogoča zunanje upravljanje mobilne naprave. Najprej so opisane uporabljene metode in orodja. Nato je opisana zunanja enota, njeni sestavni deli ter shema. Tukaj je tudi opisana komunikacija med Androidom ter zunanjo enoto prek vmesnika IOIO. Za prenos podatkov med napravama se uporabljata porotokola ADB (angl. Android Debug Bridge) ali OpenAccessory, glede na podporo same Android naprave. Sledi opis zagotavljanja natančnega metronoma na osnovi cikličnega predvajanja zvočnega posnetka v formatu PCM (angl. Pulse Code Modulation). Z uporabo metod že vgrajenega razreda AudioTrack je mogoče predhodno posnet klik skrajšati glede na število udarcev na minuto ter ga predvajati v neskončni zanki. Izdelek diplomske naloge ima tudi možnost določanja tempa na podlagi zaporednih pritiskov nekega gumba in tako sproti ugotovimo, koliko udarcev na minuto potrebujemo. Proti koncu je opisana praktična uporaba izdelka z opisom ključnih gradnikov uporabniškega vmesnika, kot zadnje poglavje pa možne izboljšave, kot bi bila uporaba brezžične tehnologije Bluetooth.

Ključne besede:

Glasba, metronom, USB, IOIO, ADB, OpenAccessory, AudioTrack

Abstract

When playing musical instrument, especially drumkit, it is very important that tempo is accurate. In order to improve playing accuracy, musicians use device called metronome, which provides constant time, depending on desired beats per minute (BPM). In this thesis, we developed metronome on Android platform and external unit, which is connected with mobile device using USB interface and enables us to control Android device externally. In the beginning, methods and tools used in this thesis are described, following by description of external unit, its components and circuit diagram. Communication between Android and external unit via IOIO protocol is also described here. To transfer data between devices, ADB (Android Debug Bridge) and OpenAccessory protocols are used, depending on Android device support. Then we describe how to provide very accurate metronome based on cyclical playing of a sound clip in PCM (Pulse Code Modulation) format. Using builtin AudioTrack methods, we can cut the previously recorded click according to specified beats per minute and play it in an infinite loop. Application, developed in this thesis, also enables us to tap tempo to easily determine how many beats per minute we actually need. In the end, we describe practical use of this product and also some key user interface components are described. Finally there is a description of possible improvements including using Bluetooth wireless technology.

Keywords:

Music, metronome, USB, IOIO, ADB, OpenAccessory, AudioTrack

Poglavje 1

Uvod

V glasbi je najbolj pomemben stalen in enakomeren tempo izvajanja. Za to glasbeniki pogosto uporabljajo napravo, ki jim daje enakomeren pulz - metronom. Daje nam trajanje osnovne dobe oziroma tempo.

Tempo označuje hitrost izvajanja glasbenih del in se nanaša na trajanje osnovne ritmične enote v taktu. (Npr. pri 3/2 ali tri-polovinskem taktu je osnovna ritmična enota polovinka, pri 4/4 ali štiri-četrtinskem taktu je to četrtinka).[1] Kot enakomerno hitrost izvajanja lahko določamo z opisnimi oznakami (andante, allegro...), ali natančno, z metronomsko oznako, s katero definiramo, koliko danih ritmičnih enot moramo zaigrati v eni minuti - udarcev na minuto (angl. beats per minute - BPM).

Za natančno merjenje tempov, v fizičnem svetu poznamo več vrst metronomov, digitalnih ter analognih, tako cenejših, kot tudi dražjih. V diplomski nalogi skušamo takšen metronom narediti za mobilni telefon Android, da je najbolj priročen in ga imamo vedno s seboj.

Vendar aplikacij za Android, ki imajo funkcionalnost metronoma, ne manjka. Aplikacija, razvita v diplomski nalogi ima poleg osnovne, še dodatno funkcionalnost - priklop zunanjega modula in nadzor mobilne aplikacije od zunaj. To je najbolj pomembno pri živem igranju na odru. Namreč, pogosto ni časa in tudi koncentracije, da bi sproti menjavali število udarcev na minuto iz pesmi v pesem prek vmesnika. V diplomski nalogi rešujemo težavo

preklapljanja med pesmimi s pomočjo zunanjega modula, priklopljenega na vmesnik USB telefona.

Poglavje 2

Uporabljene metode in orodja

Za razvoj diplomske naloge smo se posluževali samo odprtokodnih rešitev. Uporabljena razvojna orodja tudi niso odvisna od operacijskega sistema, zato smo uporabili operacijski sistem Ubuntu, različice 12.04.

2.1 Android in razvojno okolje IntelliJ IDEA

2.1.1 Googlov operacijski sistem Android

Je odprtokodni programski jezik in operacijski sistem, deluje na različnih manjših napravah, zaenkrat najbolj uporabljen na mobilnih telefonih. Jedro je osnovano iz jedra operacijskega sistema Linux, z izboljšavami in optimizacijami za uporabo na manjših mobilnih napravah.

V začetku ga je razvijalo podjetje Android Inc.[2] V operacijskem sistemu je Google kasneje spoznal velik potencial, zato je podjetje vzel pod svoje okrilje ter ustanovil odprto združenje mobilnih naprav, imenovano Open Handset Alliance. Tu si podjetja prizadevajo k razvoju standardov za mobilne naprave ter večjih inovacij.

Prednosti operacijskega sistema Android:

- cenejše in lažje razvijanje programov, saj je Android odprtokoden sistem. Posledično je tudi večina aplikacij, napisanih zanj, zastojnska,

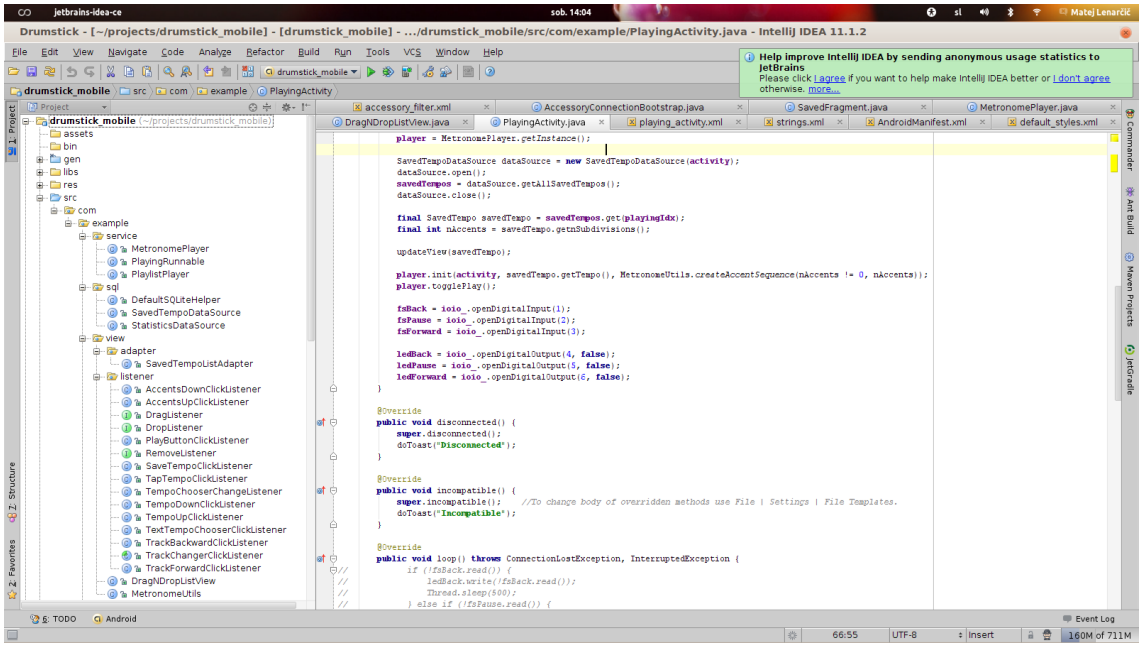
- cenejši, hitrejši in lažji razvoj mobilnih telefonov, ker proizvajalcem ni potrebno več razvijati operacijskih sistemov, ampak zgolj posamezne komponente sistema,
- enostaven, odziven, omogoča večopravilnost ter
- samodejno sinhroniziran z Googleovimi storitvami.

Razvoj aplikacij v veliki večini poteka v programskem jeziku Java, razvijalcu se ponavadi ni potrebno ukvarjati z delovanjem v ozadju. Vse ključne funkcionalnosti naprave so dostopne prek Android API-jev (angl. Application Programming Interface - Aplikacijski programski vmesnik). Dokumentacija je kar izčrpna, vsebuje veliko realnih primerov [3], zelo veliko dodatne dokumentacije in zastojnske podpore pa lahko dobimo tudi drugje na internetu.

V sami diplomski nalogi je sicer potrebno poznati tudi kakšno posebnost, ki je ne najdemo v klasičnih tipih mobilnih aplikacij. Posebna težava lahko nastane pri delu z nitmi, ki jim operacijski sistem Android lahko ponudi premalo procesorskih ciklov - posledično pride do nestabilnega metronoma. Prav tako je dobro poznati osnovno delovanje protokolov, ki skrbijo za povezavo mobilne naprave z zunanjimi napravami (ADB, OpenAccessory).

2.1.2 IntelliJ IDEA

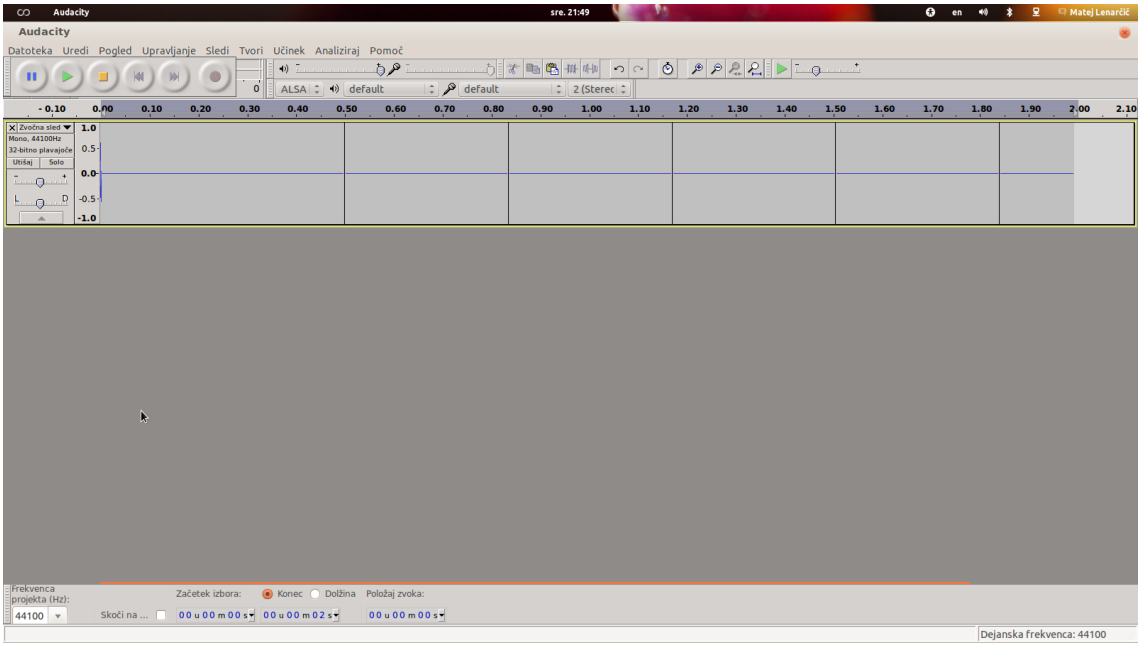
IntelliJ IDEA je integrirano razvojno okolje (angl. Integrated Development Environment). IDE je okolje za razvoj aplikacij, ki omogoča programskim razvijalcem lažje delo. V celoti je razvito v programskem jeziku Java. Posebnost okolja je, da je posebej usmerjeno k samem pisanju in strukturiranju kode ter ne toliko k vizualnim čarovnikom, ki sami opravijo veliko dela z le nekaj kliki. Poznamo dve različici okolja: IntelliJ IDEA Ultimate - plačljiva ter IntelliJ IDEA Community Edition - zastonjska. Analiziranje ter optimizacija kode pri obeh enaka; razlika je le v ogrodjih (angl. framework), ki jih podpirata. Različica Ultimate je bolj za poslovno rabo in razvoj na Java EE (za



Slika 2.1: Zaslon razvojnega okolja IntelliJ IDEA.

poslovno rabo, angl. enterprise edition) strežnikih, medtem ko ima različica Community Edition bolj okrnjeno podporo za splet, k sreči ima pa popolno podporo za razvoj Android aplikacij, kar nam je v veliko pomoč. Zato smo uporabili Community Edition. Nekatere njegove pomembnejše funkcije[4]:

- inteligentni urejevalnik kode,
- integracija testnih okolij JUnit in TestNG,
- podpora za Javo 7,
- podpora za pakiranje projektnih datotek z Maven, Gradle in Ant,
- podpora za Groovy programski jezik (dodatno tudi Scala in Clojure),
- podpora za Google Android,
- podpora za Subversion, Git/GitHub, Mercurial in CVS,



Slika 2.2: Zaslon urejevalnika zvočnih posnetkov Audacity.

- odlično pozna XML strukturo, tudi sprotno učenje prek imenskega prostora (angl. namespace),
- vizualni urejevalnik uporabniških vmesnikov Swing.

Razvojno okolje je posebej močno v optimizaciji kode, logični in strukturalni analizi ter učinkoviti in nevsiljivi pomoči pri kodiranju. Razvijalcu je lahko tudi posebno všeč dostopnost prek tipkovnice in bližnjičnih tipk, kar pomeni, da dejansko lahko popolnoma kodiramo tudi samo s pomočjo tipkovnice.

2.2 Audacity

Za predvajanje klika ga je potrebno tudi računalniško ustvariti. Ustvaril sem ga v odprtokodnem programu Audacity, ki ima omenjeno funkcijo ustvarjanja metronoma že privzeto vključeno. Urejanje zvočnih posnetkov je mogoče v

večih kanalih hkrati ali ločeno, delo z zvočnimi posnetki je pa preprosto. Uporabniška izkušnja deluje na podobnem principu, kot bi urejali datoteko z besedilom (pozna vse ključne možnosti - urejanje, kopiranje, lepljenje).

2.3 IOIO s protokoloma ADB ter OpenAccessory

2.3.1 Vmesnik IOIO

Razvil ga je Ytai Ben-Tsvi, inženir iz Izraela, sedaj zaposlen pri Googlu. Proizvaja in prodaja se pa v Združenih državah Amerike, pri podjetju SparkFun. Vmesnik IOIO je posebej odličen zato, ker se prilagaja gostiteljevi napravi, odvisno od podpore, ki jo ponuja za zunanje naprave.

Protokol ADB

Včasih je bilo mogoče do mobilnega telefona z Androidom oddaljeno dostopati le prek protokola ADB (angl. Android Debug Bridge - Androidov most za razhroščevanje). V prvi vrsti je namenjen za razvoj aplikacij iz računalnika ter nadziranje delovanja z računalnikom. Ytai je tako dobil idejo: zakaj ne bi obstajala naprava, ki se predstavi kot računalnik oz. gostitelj, vzpostavi povezavo ter pošilja in prejema podatke iz Android naprave. Na ta način lahko komunicira s praktično vsakim mobilnim telefonom Android z operacijskim sistemom različice 1.6 ali več.

Protokol OpenAccessory

Za protokolom ADB je, šele leta 2011, Google razvil poseben protokol za komunikacijo z zunanjimi napravami. Imenuje se OpenAccessory. Ima kar nekaj prednosti:

- Mobilna naprava z Androidom je ob vklopu prek vmesnika USB sposobna sama prepoznati gostiteljevo napravo ter opcijsko tudi zagnati želeno aplikacijo, glede na priključeno napravo,

- protokol sedaj lahko deluje tudi brez kabla, in sicer z uporabo Bluetootha,
- med napravama potekajo hitrejši prenosi.

Slaba lastnost je edino podpora za starejše naprave Android - deluje le od različice 2.3.5 naprej.

USB gostiteljski način ter USB OTG

Na koncu lahko omenimo še mogoče najnovejši, na splošno najbolj standariziran, ampak hkrati tudi zaenkrat najmanj podprt protokol na mobilnih napravah - USB gostiteljski način (angl. USB host mode).

Vmesnik IOIO običajno ni podpiral USB gostiteljskega načina. Vendar so ravno v času sklepnega razvoja diplomske naloge razvili nov model, ki ima tudi to podporo. Imenuje pa se IOIO OTG oz. način Na poti (angl. OTG - On the go). Gre za specifikacijo, ki omogoča manjšim napravam USB - digitalnim zvočnim predvajalnikom ali mobilnim telefonom, da se obnašajo kot gostitelji. To pomeni, da lahko nanje priključimo ključke USB, miško, tipkovnico, na nekatere celo tiskalnik in podobno. Za razliko od standardnih sistemov USB, lahko sistemi USB OTG opustijo funkcijo gostovanja ter se obnašajo kot navadne naprave USB.

Podpora USB OTG na mobilnih telefonih je zadnje čase sicer vse bolj prisotna. Dejansko nam omogoča, da lahko na mobilni telefon priključimo USB razdelilec (angl. hub) ter nato hkrati uporabljamo tipkovnico, miško, tudi zunanje pogone USB. Trenutno najmodernejši mobilni telefoni (na primer Samsung Galaxy S3) podpirajo tudi hkraten izhod HDMI in USB gostiteljskega načina prek istega vmesnika USB z uporabo protokola MHL. Izdelovalci odprtokodnega projekta Ubuntu so že izkoristili prednosti zmogljivih mobilnih naprav. Napovedali so, da razvijajo sistem, da bomo lahko mobilni telefon zunaj uporabljali običajno kot do sedaj, ko bomo pa prišli npr. domov, ga bomo pa lahko vključili v posebeno namizno postajo in iz telefona uporabljali namizno različico operacijskega sistema prek miške in tipkovnice.

Vse se seveda zaganja in opravlja na procesorju mobilnega telefona. Verjetno je do vsakodnevne uporabe takšnega sistema le vprašanje časa.

Poglavje 3

Zunanja enota

Zunanja enota je, če odštejemo vmesnik IOIO, zelo preprosta naprava. Napaja se preko 3,3V adapterja pod napetostjo 1A, kar je tudi zahteva vmesnika IOIO. Povezava z mobilno napravo se izvaja prek vmesnika USB, standardno Micro B, kot je prikazano na sliki 3.1.

Sestavni deli, ki omogočajo interakcijo z uporabnikom, so povezani na vmesnik IOIO prek različnih stičišč (angl. pins). Vmesnik IOIO [5] ima takšnih stičišč 48. Vsi lahko delujejo kot vhodi oziroma kot izhodi ter vsi so sposobni prenašanja analognih in digitalnih meritev, odvisno od incializacije s strani Android aplikacije.

3.1 Sestavni deli

Zunanjo enoto sestavljajo naslednji sestavni deli:

- 3 nožna stikala,
- 3 lučke LED,
- ter seveda vmesnik IOIO.

Slika 3.1: Zunanja enota z vsemi pripadajočimi kabli za priključitev.

3.2 Shema priključitve

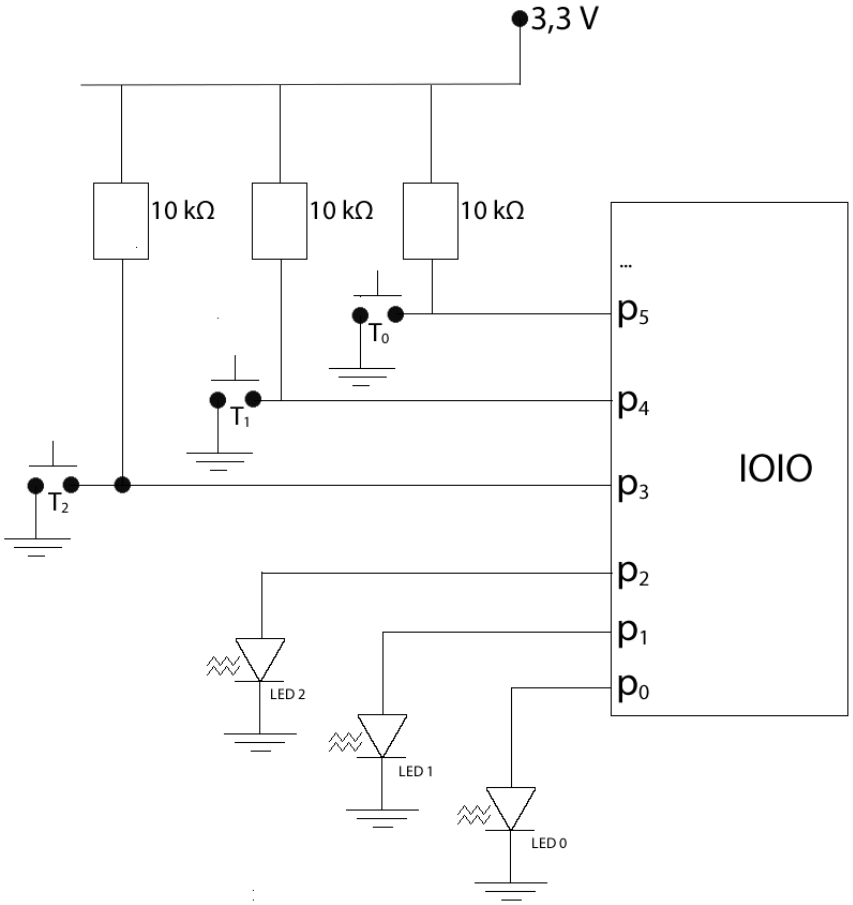
Notranjost zunanje enote je priključena tako, kot je razvidno iz slike 3.2.

3.3 Komunikacija

3.3.1 Komunikacija prek aktivnosti v Javi

Na Android strani je koda, ki komunicira z vmesnikom IOIO, razmeroma preprosta.

Večino stvari uredi samo ogrodje IOIO, ki ga dobimo na internetu. Zahteva sicer nekaj konfiguracije, vendar je kar dobro opisana na sami spletni strani proizvajalca ter vsebuje tudi nekaj živih primerov delujočih aplikacij. Ogrodje nato samo poskrbi, da ni težav s preverjanjem, če je naprava priklopljena ter deluje, prav tako tudi funkcionalnost naprave dosežemo na razmeroma lahek način. Ogrodje samodejno ustvari svojo lastno nit, prek



Slika 3.2: Shema priključitve vmesnika IOIO.

katere komunicira s priključeno napravo. V ospredju se, kar se nanaša na IOIO, ne dogaja veliko in je uporabniška izkušnja zato lahko zelo dobra. V ločenih nitih pa ogrodje samodejno ustvari zanke, v katerih lahko preverjamo in obdelujemo vrednosti podatkov ter jih uporabimo drugje.

Da izkoristimo možnosti, ki nam jih IOIO ponuja, moramo le razširiti (angl. Extend) razred aktivnosti (angl. activity), ki podeduje iz abstraktnega razreda IOIOActivity. V njem povozimo nekatere metode:

setup()

Zgodi se ob inicializaciji, prvem zagonu aktivnosti. V njej ponavadi povežemo objekte v Javi ter vhodne in izhodne naprave, priključene na naš vmesnik IOIO. Določimo tudi, katere pozicije na vmesniku IOIO bodo v uporabi in kakšna bo njihova funkcija (vhodna ali izhodna).

loop()

Je funkcija, ki se kliče cel čas znotraj nove niti. V njej preverjamo vrednosti vhodnih enot ter nastavljamo vrednosti izhodnih. Ker se odvija znotraj niti, lahko uporabimo tudi metodo Thread.sleep(nMilisekund) ter tako poskrbimo, da neka vrednost obstane dlje časa prisotna. Med takšnim klicem sicer vse ostale funkcije odpovejo, vendar je lahko v našem primeru pritiskanja na nožna stikala (kjer moramo poskrbeti, da se en pritisk ne upošteva kot večkratni), takšno čakanje dobrodošlo.

disconnect()

Kliče se ob izgubi povezave. Tukaj lahko nastavimo kakšne vrednosti na privzete, prikažemo uporabniku sporočilo, itd.

3.3.2 Samodejno zaganjanje aplikacije (samo prek protokola OpenAccessory)

V `androidManifest.xml` lahko dodamo, kaj naj se zažene ob priklopu zunanje naprave na mobilno napravo Android, na sledeči način:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example"
          android:versionCode="1"
          android:versionName="1.0">

<!-- obvezna knjižnica za delo s protokolom OpenAcessory -->
<uses-library android:name="com.android.future.usb.accessory"
              android:required="false" />

<activity android:name="Main"
          android:label="@string/app_name"
          android:logo="@drawable/drumstick_logo">

<!-- (standardne nastavitve aplikacije) -->

<!-- tukaj povemo, da aplikacijo zanima dogodek, ko
     priključimo zunanjo napravo -->
<intent-filter>
<action
    android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
</intent-filter>
<meta-data
    android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
    android:resource="@xml/accessory_filter" />
</activity>

</application>
</manifest>
```

V tem primeru smo na dogodek priključitve zunanje enote registrirali kar aktivnost `Main`. Namesto le-te bi lahko tudi `PlayingActivity`, ki skrbi za predvanjanje in upravljanje prek zunanje enote, a sem ocenil, da je s stališča

uporabniške izkušnje tako bolj smiselno. Tako se potem zažene določena aktivnost oz. uporabniku prikaže sporočilo.

Poglavje 4

Aplikacija

Pri izdelavi mobilne aplikacije smo morali biti pozorni na nekaj posebnosti, ki jih v drugih mobilnih aplikacijah ne najdemo. Naša aplikacija potrebuje procesorski čas tako za nit, ki dela v ospredju, kot tudi za nit, ki deluje v ozadju in skrbi za nemoteno predvajanje metronoma.

4.1 Analiza

4.1.1 Glavni objekti v aplikaciji

Prvi zaslon aplikacije nam mora omogočati nastavljanje tempa ter nekaterih njegovih lastnosti. Obstajati mora možnost morebitne kasnejše razširitve aplikacije ter samih nastavitev. V operacijskem sistemu Android imamo na voljo že preprosto relacijsko bazo SQLite, ki sicer za svojo interno shrambo potrebuje zgolj eno datoteko, pozna pa nek preprost nabor SQL poizvedb. V programskem jeziku Java je seveda lažje delo s samimi objekti kot sprotno pisanje SQL poizvedb, zato je dobra praksaa razviti tudi vmesni nivo med SQLite relacijsko bazo ter objekti v Javi. Cilj je, da bi upravljali samo z navadnimi objekti, ki predstavljajo določeno entiteto, shranjevanje naj bo samodejno oz. na programerjev ukaz.

Sama struktura entitet je pri aplikaciji zelo preprosta. Potrebujemo dejansko samo eno entiteto, ki nosi nastavitve prvega zaslona, dajmo ji ime

`SavedTempo`. Vsebovati mora:

- id,
- ime tempa,
- hitrost tempa (udarcev na minuto),
- osnovno dobo,
- število dob v taktu,
- poudari prvo dobo da / ne.

Osnovni zaslon ima sicer manj opcij, ima pa funkcijo "Poudari na", ki dejansko pomeni, na koliko udarcev se sproži močnejši udarec. V kateri izmed naslednjih različic bi lahko vključili malo bolj napredne nastavitve (funkcionalno sicer zelo podobne obstoječim), želeli pa bi, da so združljive z vsemi različicami. Zaenkrat je zato bolje, da pretvorimo opcijo ene nastavitve v več atributov, in sicer:

- osnovna doba fiksno 4 (četrtinke), saj nenazadnje ni važna - šlo bi samo za lažje prepisovanje tempa iz različnega notnega materiala. Namreč če vzamemo osnovno dobo četrtinko ali osminko in ji dodelimo hitrost 120 udarcev na minuto, je to povsod 120 udarcev; ne glede na poudarke in podobno,
- število dob v taktu - to je praktično opcija "Poudari na" - privzeto 4,
- poudari prvo dobo da / ne - privzeto izključeno. Pomeni, če naj se sploh uporabljajo nastavitve poudarjanja.

4.2 Načrtovanje

4.2.1 Natančen metronom

Za sam razvoj metronoma se nam najbolj ponuja sledeče zaporedje izvajanja:

- predvajaj klik,
- čakaj določeno število milisekund glede na hitrost (udarcev na minuto),
- ponavljaj.

Vendar se kmalu izkaže, da tukaj naletimo na težavo. Na internetu najdemo že nekaj primerov takšnih metronomov, ki pa sploh niso natančni. Zakaj je temu tako - ker se predvajanje klika ne more vršiti v glavni niti, ji sistem Android zaradi upravljanja z energijo ne dodeli dovolj procesorskih ciklov. Zato takšen način pri nas ne pride v poštev.

Ponuja se nam rešitev; lahko povišamo prioriteto niti, ki predvaja metronom, z uporabo metode `android.os.Process.setThreadPriority(prioriteta)` [6]. Mogoče takšen način deluje razmeroma dobro, vendar je v realnem svetu lahko veliko različnih situacij, kot je prejemanje SMS sporočila, klicev in podobno. Takšne situacije lahko vsilijo svojo nit nad tisto, ki se že izvaja, kar spet ni ugodno.

Po manjših preizkušnjah ugotovimo, da se najbolj obnese uporaba vgrajenega predvajalnika surovega PCM (angl. raw pulse code modulation) formata, z razredom `AudioTrack`. Gre za metodo, ki digitalno predstavi vzorčene analogne signale. Je standardna oblika digitalnega zvočnega zapisa v računalnikih in različnih formatih Blu-ray, DVD ter CD, kot tudi drugje, kot so na primer digitalni telefonski sistemi. Tok PCM je digitalna predstavitev analognega signala, kjer je magnituda analognega signala vzorčena na stadardno določen interval - frekvenco vzorčenja (angl. sampling rate). Na takšen interval se izbere približek vrednosti znotraj območja digitalne bitne globine (angl. bit depth).[7]

Kar se tiče samega predvajanja, gre za popolnoma isti način kot je na primer sprotno sprejemanje radijske postaje iz interneta. Tukaj je sistem Android tudi prisiljen, da za nit nameni dovolj procesorskega časa, saj pri nobenem predvajanju ne sme (razen v res ekstremnih slučajih), priti do preskakovanja.

Postopek, ki ga moramo opraviti pri predvajanju:

- glede na izbrano število udarcev na minuto izračunamo, koliko milisekund mora preteči med dvema klikoma (običajno med 400 in 1500 milisekund),
- odpremo razred za predvajanje AudioTrack,
- število milisekund iz prve alineje pretvorimo v sekunde ter pomnožimo s frekvenco vzorčenja datoteke s klikom – pri nas je to 44100 Hz. Tako dobimo število bajtov, ki jih moramo naložiti v našo instanco razreda `AudioTrack`. Iz toka (angl. stream) preberemo natančno toliko bajtov,
- začnemo ciklati predvajanje. Ker se predvajanje vrši sinhrono v niti, ki ga je zagnal, ni skrbi, da bi prišlo do preskakovanja. Vmes preko lokalnih parametrov objekta preverjamo, če želi uporabnik ustaviti predvajanje ali spremeniti tempo izvajanja.

4.2.2 Razred za predvajanje

V celotni aplikaciji ne sme priti do podvajanja instanc predvajalnika, kar bi lahko pomenilo hkratno predvajanje večih metronomov. Zato smo se poslužili načina, ki je dobro poznan v svetu objektno usmerjenih jezikov - singleton (angl. edinec). To pomeni, da se v sistemu naenkrat vedno nahaja samo ena instanca razreda, torej nujno samo en objekt. Razred ima nujno privatni konstruktor, ne dovoljuje izdelovanja novih instanc od zunaj, ampak jih lahko samo znotraj sebe prek metod, s katerimi tudi preverja če že obstajajo instance (ponavadi `getInstance()`). Sam sebi prek statičnega polja nosi svojo instanco, do katere lahko povsod dostopamo. Ob zagonu aplikacije instanca še ne obstaja, zato se mora samodejno skreirati ob prvi uporabi. Primer:

```
public class MetronomePlayer {
    private MetronomePlayer instance;

    // privatni konstruktor
    private MetronomePlayer() {
```

```
    }

    public MetronomePlayer getInstance() {
        if (instance == null) {
            instance = new MetronomePlayer();
        }
        return instance;
    }
}
```

4.2.3 Glavna nit uporabniškega vmesnika ter nit predvajalnika

Razred, ki skrbi za predvajanje, mora skrbeti tudi za upravljanje z nitjo predvajalnika in s tem mora biti kot nek posrednik med obema. Povezovati mora vmesnik ter nit, ki se izvaja v ozadju. Nit predvajalnika se tako ali tako cikla in vmes pregleduje vrednosti spremenljivk, ki živijo znotraj niti. Nit uporabniškega vmesnika pa ima dostop do teh spremenljivk, zato lahko na tak način obe niti komunicirata med seboj oz. tako iz uporabniškega vmesnika nadziramo predvajanje metronoma.

Zanimivo je, da aplikacija, če je narejena v drugo smer, ne deluje; nit predvajalnika ne more dostopati do spremenljivk v niti uporabniškega vmesnika. Pri takšni strukturi pride do napak, zanimivo je, da zlahka pride do podvajanja niti predvajalnika. Najbolje je znotraj zunanje niti, ki je edinec (angl. singleton), nositi referenco notranje niti predvajalnika.

Poglavje 5

Primer uporabe

Uporaba aplikacije je kar se da preprosta. Ob zagonu aplikacije se uporabniku ponudi glavni zaslon z opcijami za nastavitev metronoma, zato ga lahko takoj uporabljamo. Ko želi uporabnik shraniti nastavitve, samo pritisne na gumb Shrani. Če zeli urejati seznam predvajanja, le s prstom povleče (angl. swipe) od desnega proti levem robu zaslona in tako se mu prikaze seznam shranjenih tempov. Tam jim lahko preuredi vrstni red ali jih pobriše.

5.1 Uporabniški vmesnik

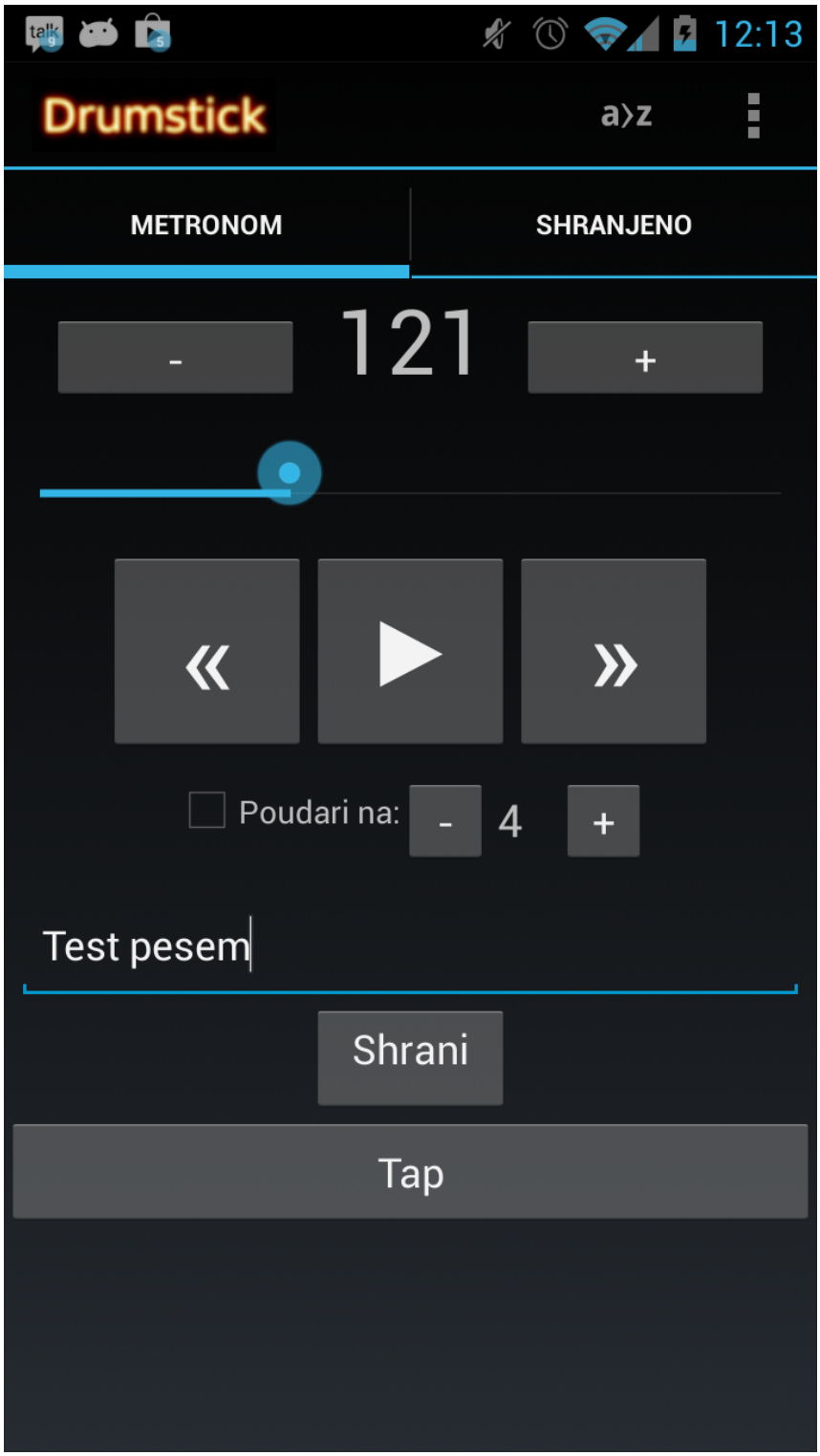
V tem poglavju so opisani ključni elementi in zunanje knjižnice uporabniškega vmesnika, ki smo jih uporabili.

5.1.1 Navigacija z zavihki

Preprosti navigaciji smo dodali še možnost premika s prstom (angl. swipe).

5.1.2 Deli uporabniškega vmesnika oziroma fragmenti

Obnašajo se kot aktivnosti z nekaj izjemami: imajo omejeno življenjsko dobo, med seboj lahko bolje komunicirajo ter so stalno prisotni v pomnilniku, ko smo v aktivnosti. Uporabljeni so kot glavni zaslon ter zaslon s seznamom.



Slika 5.1: Osnovni zaslon aplikacije.

5.1.3 Urejeni seznam z možnostjo preurejanja

Uporabili smo že narejenega, in sicer ga je ustvaril Eric Harlow [8]. Uporablja se v seznamu shranjenih tempov. S preurejanjem prek uporabniškega vmesnika dejansko dosežemo tudi preurejanje v bazi (posodablja se polje order, prek katerega se prikaže urejene elemente).

5.1.4 ActionBar

Gre za enovit način prikaza, ki ga je Google prvič privzeto vključil v Android različice 3.0 Honeycomb. Programerju se ni potrebno ukvarjati s tem, kako bo prikazal uporabniški vmesnik ter kako bodo razporejene nekatere kontrole na zaslonu. Takšna funkcionalnost je še posebej dobrodošla, ker operacijski sistem sam presodi, kako najbolje prerazporediti gumbe glede na različne velikosti zaslonov. Vse, kar je potrebno narediti, je le implementirati nekaj metod glede na naše potrebe. V diplomski nalogi smo vključili nekatere ključne gumbe v ActionBar: na primer gumb za preklop v način za upravljanje z zunanjo enoto. Prav tako smo uporabili prostor za logotip zgoraj levo. Lahko bi vključili še ostale pomembnejše gumbe (na primer predvajanje), vendar je mogoče bolje ključno funkcionalnost postaviti na sredino zaslona.

Za boljšo podporo na mobilnih napravah, ki nimajo vključene funkcionalnosti ActionBara - različice Androida nižje od 3.0 Honeycomb - smo namesto osnovnega ActionBara uporabili posebno razširitev, imenovano ActionBar Sherlock.[9] Gre za ogrodje, ki v primeru, da ima gostiteljski operacijski sistem podporo za ActionBar, uporabi vgrajeno, sicer pa naredi svojo implementacijo, ki se obnaša enako. Metode, ki jih lahko uporabljamo, so identične tistim, ki jih ponuja navaden ActionBar. Vse, kar moramo storiti je le, da namesto standardnega razreda za aktivnost Activity, v naših aktivnostih razširimo SherlockActivity. V našem primeru, ko uporabljamo fragmente, razširimo razred SherlockFragment namesto Fragment.

5.2 Zunanja enota

Ko uporabnik vstopi v način upravljanja prek zunanje enote, mu aplikacija zazna, če je bila zunanja enota priključena. Ko je priključena, se uporabniku prikaže obvestilo in aplikacijo lahko upravlja prek zunanje enote.

Na voljo ima 3 funkcije, vgrajene v zunanjo enoto, dostopne prek nožnih stikal:

- premik za eno pesem nazaj,
- ustavljanje (pavza),
- premik za eno pesem naprej.

Ob pritisku na katerega izmed nožnih stikal se tudi posveti lučka LED nad gumbom, kar sporoči uporabniku, da je aplikacija ukaz sprejela. Lučka sveti tako dolgo časa, dokler ni spet pripravljena na sprejem naslednjega ukaza (to je 500 milisekund). Tako je narejeno zato, da slučajno ne pride do nezaželjenega večkratnega zaporednega pritiska gumbov.

Poglavje 6

Možnosti izboljšav in razširitev

Aplikacija sicer odlično zadošča osnovnim potrebam, vendar bi, da bi zadostili večjemu krogu uporabnikov, lahko vstavili še kako izboljšavo ali dodatno funkcionalnost. V začetku načrtovanja imamo običajno bolj omejen pregled oz. nas zanima osnovna funkcionalnost aplikacije, kasneje pa ponavadi dobimo vse več idej za razvoj. Da vsaj okvirno zadostimo prvotno začrtanim rokom, žal ne moremo vseh idej razviti že takoj, ampak jih lahko shranimo za kakšno od naslednjih iteracij.

6.1 Izboljšave uporabniškega vmesnika

Tukaj so opisane različne izboljšave glede uporabniškega vmesnika, ki bi jih lahko v bodoče dodali aplikaciji.

6.1.1 Bolj standarizirano izbiranje taktovskega načina

V osnovi poudarjene note izbiramo na zelo preprost način - katero noto naj poudari, koliko nepoudarjenim naj sledi poudarjen udarec. Vendar se v glasbi uporablja drugačna notacija, namreč taktovski način. Kot osnovo uporabljamo osnovno dobo (četrtinka, osminka, šestnajstinka...), ki jo ponavljamo glede na število dob v taktu. Uporabniku bi lahko dali na voljo izbor taktovskih načinov (prednastavljenih), ali mu dali možnost, da si ga nastavi

sam. Uporabnik nato nastavi hitrost osnovne dobe (ravno tako v udarcih na minuto), prvo dobo v taktu ima pa možnost poudariti. Tako dosežemo sicer enako funkcionalnost od obstoječe, vendar je bolj standarizirana. Za uporabnika, ki je bolj vešč glasbene notacije, je takšen način mogoče bolj prijazen. Tudi prepisovanje tempa iz različne notne literature je tako malo bolj prijazno.

6.1.2 Osnovni izgled

Sam osnovni izgled je trenutno popolnoma osnoven oz. osnovan na privzetih slogih uporabniškega vmesnika. Vendar glede na to, da je to aplikacija, ki hoče upodobiti metronom, bi lahko rahlo spremenili barve ter mogoče celo dodali kakšno teksturo, ki bi bolj spominjala na to, kar hočemo prikazati. Da se ne bi preveč oddaljili od osnovnih principov načrtovanja uporabniškega vmesnika za platformo Android, bi lahko na primer zgolj na osnoven ActionBar dodali sloge prek datoteke `default_styles.xml`.

6.2 Zunanja enota prek povezave Bluetooth

Zunanja enota se trenutno poveže na mobilni telefon prek povezave USB. To ima lahko svoje omejitve glede uporabe, saj vedno potrebujemo kabel, na koncertih bi nam bilo pa mogoče bolj primerno pustiti telefon nekje bolj oddaljeno oz. na varnejših mestih (kakšen etui, morda kar v kovčku...). Za to bi lahko uporabili kakšno brezžično različico. K sreči povezava USB med dvema napravama temelji na istih specifikacijah kot Bluetooth. Tako lahko deluje protokol OpenAccessory tudi brezžično, posledično deluje tudi naša zunanja enota. Uporabili bi lahko Bluetooth vmesnik za IOIO, ki ga le priključimo na USB konektor naše zunanje enote[10].

Žal ima to tudi svoje slabosti. Starejši protokol ADB prek Bluetooth povezave ne deluje, zato lahko za komunikacijo uporabljamo le protokol OpenAccessory. Za brezžično povezavo Bluetooth torej nujno potrebujemo mobilno napravo z različico Androida 2.3.5 ali več.

6.3 Sinhronizacija

Ko si zapisujemo vrstne rede pesmi, ki jih bomo izvajali na koncertu, je včasih to prek mobilnega telefona rahlo zamudno. Lahko bi ustvarili spletni vmesnik za urejanje, ki bi se potem samodejno sinhroniziral z napravo. Dobra lastnost te funkcije bi bila tudi ta, da bi lahko seznam za igranje prek računalnika natisnili ter ga posredovali še drugim članom skupine. Sinhronizacijo bi najlažje implementirali prek spletnih storitev (angl. web service). Na eni strani bi razvili kodo za spletni strežnik, ki bi ponujal vsebino prek formata XML (angl. eXtensible Markup Language) ali JSON (angl. JavaScript Object Notation) ter avtentikacijo uporabnika. Na drugi strani pa primerno prilagodili klienta oziroma našo mobilno aplikacijo.

Poglavje 7

Sklepne ugotovitve

Diplomska naloga je odprla novo obzorje možnosti priključevanja zunanjih naprav in mobilnih naprav Android. Tukaj smo uporabili zgolj digitalni prenos (pritisk gumbov na plošči ter prižiganje lučk LED – tu obstajata samo 2 stanji, resnično oz. neresnično). V realnem svetu je tudi veliko slučajev, kjer lahko uporabimo analogne vhode ter izhode, na primer prebiranje različnih količin kot so temperatura, moč pritiska, itd. kot vhod. Prav tako lahko uporabimo tudi različne izhode, za nadziranje raznih manjših elektromotorjev, robotov, itd. Skupaj z mobilno napravo odlično sodelujejo ter vsak doda svoje zmožnosti, senzorje, nenazadnje tudi povezavo v internetno omrežje prek mobilnega ali brezžičnega omrežja. Dandanašnji mobilni telefoni so pa nasploh vse bolj zmogljivi, zato lahko tudi za takšne namene izkoristimo njihovo procesorsko moč.

Literatura

[1] Tempo. Dostopno na:
http://sl.wikipedia.org/wiki/Tempo

[2] Android (operacijski sistem). Dostopno na:
http://sl.wikipedia.org/wiki/Android_(operacijski_sistem)

[3] Uradna dokumentacija za Android. Dostopno na:
http://developer.android.com

[4] IntelliJ IDEA uradna stran. Dostopno na:
http://www.jetbrains.com/idea/

[5] Stran izdelovalca SparkFun z opisom vmesnika IOIO. Dostopno na:
https://www.sparkfun.com/products/10748

[6] Dokumentacija funkcije `setThreadPriority(int)`. Dostopno na:
http://developer.android.com/reference/android/os/Process.html#setThreadPriority(int)

[7] Pulse code modulation (PCM). Dostopno na:
http://en.wikipedia.org/wiki/Pulse-code_modulation

[8] Blog Erica Harlowa s kodo za preurejanje seznama. Dostopno na:
http://ericharlow.blogspot.com/2010/10/experience-android-drag-and-drop-list.html

[9] ActionBarSherlock domača stran. Dostopno na:
http://actionbarsherlock.com

[10] IOIO prek povezave Bluetooth. Dostopno na:
https://github.com/ytai/ioio/wiki/IOIO-Over-Bluetooth