UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dušan Strgar

Mobilna aplikacija za daljinsko upravljanje fotoaparata

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dušan Strgar

Mobilna aplikacija za daljinsko upravljanje fotoaparata

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: izr. prof. dr. Patricio Bulić

Ljubljana, 2014

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: 00566 / 2013
Datum: 6.11.2013

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

Kandidat: **DUŠAN STRGAR**

Naslov: **MOBILNA APLIKACIJA ZA DALJINSKO UPRAVLJANJE FOTOAPARATA**
MOBILE APPLICATION FOR REMOTE CONTROL OF A CAMERA

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Izdelajte aplikacijo za platformo Android, ki bo omogočala daljinsko upravljanje prožilca na digitalnem fotoaparatu znamke Canon. Aplikacija naj omogoča tri načine krmiljenja prožilca: preko infra-rdeče povezave, preko serijske povezave USB ter preko avdio-izhoda na mobilnem telefonu. Za zadnji način krmiljenja naj aplikacija tvori pravokotne zvočne impulze, ki preko optospojnika vklapljajo in izklapljajo prožilec fotoaparata.

Mentor:
izr. prof. dr. Patricio Bulić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Dušan Strgar, z vpisno številko 63040399, sem avtor diplomskega dela z naslovom:

Mobilna aplikacija za daljinsko upravljanje fotoaparata

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- 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 12. februar 2014 Podpis avtorja: _____

ZAHVALA

Rad bi se zahvalil vsem, ki so mi kakorkoli skozi leta pomagali na poti mojega študija in izdelave diplomskega dela. Še posebej bi se rad zahvalil svoji mami Mihaeli in svojemu očetu Francu za njuno dolgoletno podporo pri izobraževanju in njuno moralno podporo. Za vso strokovno pomoč in nasvete pri izdelavi diplomskega dela se zahvaljujem svojemu mentorju izr. prof. dr. Patriciu Buliću. Posebej bi se rad zahvalil tudi Nataliji Dvoršak za njeno podporo, razumevanje in spodbude pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic in simbolov

ADT	(angl. Android Development Tools) razvojna orodja Android
API	(angl. Application Programming Interface) programski vmesnik
APS-C	(angl. Advanced Photo System type-C) format slikovnega senzorja
CMOS	(angl. Complementary Metal-Oxide Semiconductor) tehnologija izdelave slikovnega senzorja
DIP	(angl. Dual In-line Package) tip ohišja elektronskih komponent
DSLR	(angl. Digital Single Lens Reflex) digitalni zrcalnorefleksni fotoaparat
EOS	(angl. Electro-Optical System) sistem avtomatskega zaznavanja izostritve slike
IDE	(angl. Integrated Development Environment) integrirano razvojno okolje
IR	infrardeča svetloba
LED	(angl. Light Emitting Diode) svetleča dioda
PTP	(angl. Picture Transfer Protocol) protokol za prenos slik med napravami
SDK	(angl. Software Development Kit) orodje za razvoj programske opreme
TRS	(angl. Tip Ring Sleeve) oblika 3-polnega avdiopriključka
USB	(angl. Universal Serial Bus) univerzalno serijsko vodilo
Wi-Fi	(angl. Wireless Fidelity) tehnologija za prenos podatkov s pomočjo radijskih valov
XML	(angl. Extensible Markup Language) razširljiv označevalni jezik

Povzetek

Cilj diplomskega dela je bila izdelava mobilne aplikacije za operacijski sistem Android, ki bi uporabnikom na podprtih pametnih mobilnih napravah omogočala daljinsko upravljanje digitalnega zrcalnorefleksnega fotoaparata proizvajalca Canon. Aplikacijo smo poimenovali Upravljalnik fotoaparata. Primarni cilj, kateremu smo sledili v diplomskem delu, je bila izdelava aplikacije z osnovno funkcionalnostjo zajema posnetka oziroma sprožitve zaslonke. Obravnavali smo infrardeče daljinsko upravljanje za naprave z vgrajenim infrardečim oddajnikom, ožičeno kabelsko daljinsko upravljanje za uporabo v kombinaciji z napravinim avdio-izhodom in ustrezno generiranim zvočnim signalom ter tudi upravljanje prek USB-povezave. V prvem delu diplomskega dela smo predstavili fotoaparat, ki ga želimo daljinsko upravljati, ter opisali operacijski sistem Android in njegovo razvojno okolje. V drugem delu pa smo opisali izdelavo mobilne aplikacije. Za vsakega od načinov upravljanja smo najprej analizirali problematiko in poiskali možne rešitve, ki smo jih nato v okviru aplikacije praktično izdelali. Pri tem smo ponekod tudi razširili osnovno delovanje z dodatnimi funkcionalnostmi, kot so zajemanje posnetka s časovno zakasnitvijo ali slikanje v načinu bulb. V okviru razširitev delovanja smo prikazali tudi izdelavo funkcionalnosti intervalometra oziroma podpore za intervalno slikanje.

Ključne besede: Android, mobilna aplikacija, daljinsko upravljanje, intervalometer.

Abstract

The aim of this thesis was the creation of a mobile applications for the Android operating system, which would allow the users using supported smart mobile devices to remotely control a digital single-lens reflex camera from the camera manufacturer Canon. The application was named Upravljalnik fotoaparata. The primary objective, which we followed in the thesis, was to produce a application with a basic functionality of taking a photo or the triggering of the camera aperture. We discussed both the infrared remote controlling for devices with a built-in infrared transmitter, wired cable remote control for use in combination with a device's audio-output and a corresponding generated sound signals as well as control via the USB connection. In the first part of the thesis, we briefly introduce the camera that we want to remotely operate and we also describe the Android operating system and the development environment. In the second part, we describe the making of the mobile applications. For each of the controlling modes, we first analyzed the issue and searched for its possible solutions, which were then also practically built within the application. In doing so, we also sometimes expanded the basic operation with additional functionalities such as capturing a photo with a timed delay or shooting in Bulb mode. As an extension of the operation we also presented the intervalometer functionality and support for interval imaging .

Keywords: Android, mobile application, remote control, intervalometer.

1 Uvod

Dandanes digitalni zrcalnorefleksni (DSLR) fotoaparati in tudi nekateri naprednejši kompaktni oziroma brezzrcalni fotoaparati podpirajo možnost svojega daljinskega upravljanja oziroma krmiljenja s pomočjo različnih zunanjih vmesnikov in naprav. Primer take naprave je lahko bodisi infrardeči (IR) daljinski upravljalnik, kabelski upravljalnik, brezžični upravljalnik ali pa naprava, kot je prenosni računalnik z brezžično povezavo Wi-Fi (angl. *Wireless Fidelity*) ali kabelsko povezavo USB (angl. *Universal Serial Bus*). Proizvajalci fotoaparatov običajno tržijo lastne izdelke in rešitve za njihovo daljinsko upravljanje. Principi, po katerih modeli aparatov delujejo, so običajno zasnovani na odprt način. To pomeni, da v večini primerov delovanje in način komunikacije med fotoaparatom in zunanjo napravo nista zaščitena s kakšno posebno obliko šifriranja podatkov oziroma lastniškim protokolom prenosa podatkov, ki bi onemogočal zunanje posege v delovanje upravljanja. Proizvajalci na tak način omogočajo razvoj daljinskih upravljalnikov tretjim osebam, če to želijo storiti.

V okviru diplomskega dela smo se zato odločili izkoristiti omenjeno odprtost in velike zmogljivosti najnovejših pametnih mobilnih naprav ter izdelati mobilno aplikacijo za operacijski sistem Android. Cilj aplikacije je bila implementacija različnih načinov daljinskega upravljanja fotoaparata. Operacijski sistem Android je v zadnjih nekaj letih dosegel že tako visoko raven razvoja, da ob ustrezni strojni podpori omogoča več različnih načinov komunikacije z zunanjimi napravami. V našem primeru takšno napravo predstavlja fotoaparat. Primer nekaterih možnih načinov, ki jih operacijski sistem kot tak podpira, so komunikacija prek USB-vodila, povezave Bluetooth, povezave Wi-Fi, IR oddajnika in priključka za slušalke. Na tak način smo imeli pri izdelavi aplikacije več odprtih možnosti, kako dejansko izdelati želeno funkcionalnost daljinskega upravljanja. Po opravljeni analizi smo se odločili, da bomo v sklopu razvoja aplikacije poskusili implementirati komunikacijo prek USB-povezave, IR oddajnika in izhoda za slušalke. Omenjeni načini se najbolje navezujejo na obstoječe načine daljinskega upravljanja fotoaparatov, zato smo se odločili, da jih poskusimo implementirati še v sklopu mobilne aplikacije.

Na tržišču je dandanes več priznanih proizvajalcev fotoaparatov. Če naštejemo le nekatere, so to na primer Canon, Nikon, Sony, Pentax in Olympus. Glede daljinskega upravljanja njihovih lastnih modelov fotoaparatov imajo vsi ti proizvajalci zelo podobne pristope. Za namene diplomskega dela smo se omejili na modele fotoaparatov DSLR proizvajalca Canon, konkretneje na model Canon EOS (angl. *Electro-Optical System*) 550D. Razlog za omenjeno izbiro je bil dostop do naprave za izvajanje testiranja delovanja aplikacije, ki smo jo razvili. Želimo pripomniti, da se lahko, z ustreznimi prilagoditvami, večina ugotovitev in rešitev iz diplomskega dela uporabi tudi za izdelavo daljinskega upravljanja fotoaparatov nekaterih drugih proizvajalcev.

2 Predstavitev testnega fotoaparata

Za namen izdelave in testiranja mobilne aplikacije v sklopu diplomskega dela smo izbrali digitalni fotoaparat Canon EOS 550D. Oznaka EOS v nazivu predstavlja tehnologijo Canonovega sistem avtomatskega zaznavanja izostritve slike, ki je bil prvič predstavljen že leta 1987, torej še v času 35 mm filmskih fotoaparatov [1].

Fotoaparat EOS 550D (slika 1) je osnovni model zrcalnorefleksnega fotoaparata, predstavljenega leta 2010, ki ima slikovno tipalo CMOS (angl. *Complementary Metal-Oxide Semiconductor*) velikosti APS-C (angl. *Advanced Photo System type-C*) z 18 milijoni slikovnih pik [2]. Čeprav omenjeni fotoaparat spada v nekoliko nižji cenovni razred fotoaparatov DLSR proizvajalca Canon, vseeno nudi veliko funkcionalnosti, ki jih omogočajo dražji modeli. Po segmentaciji tržišča je namenjen bolj amaterskim fotografom, ki se mogoče šele začenjajo ukvarjati z resnejšo fotografijo, ali uporabnikom, ki mogoče želijo nekaj zmogljivejšega v primerjavi z različnimi kompaktnimi fotoaparati. Kot smo že omenili, ta fotoaparat vseeno ponuja veliko večino funkcij fotografiranja in upravljanja kot dražji modeli. Glavna razlika med izbranim fotoaparatom in dražjimi modeli je predvsem v kakovosti izdelave in trpežnosti pri uporabi ter v slikovnem tipalu, njegovi velikosti in delovnih karakteristikah. Ker je v prvi vrsti namenjen bolj nezahtevnim uporabnikom, je aparat nekoliko manjši in izdelan iz nekoliko cenejših materialov. Notranje ogrodje je narejeno iz nerjavečega jekla, zunanje ohišje pa iz polikarbonatne smole s steklenimi vlakni. Za izdelavo dražjih modelov se po navadi uporabljajo različne magnezijeve zlitine. Prednost polprofesionalnih in profesionalnih modelov je tudi večja odpornost na prah in vremenske vplive, kot sta na primer sneg ali dež pri uporabi v naravi.

Slika 1: Fotoaparat Canon EOS 550D [3]

2.1. Načini daljinskega upravljanja fotoaparata

Kot smo že omenili, fotoaparati EOS omogočajo več različnih načinov daljinskega upravljanja [4]. Nekateri od teh kot na primer kabelsko upravljanje so podprti že vse od prvih fotoaparatov naprej in so ostali skoraj nespremenjeni skozi leta obstoja serije, medtem ko so drugi načini upravljanja relativno novi. Primer je recimo brezžično upravljanje prek povezave Wi-Fi. Nekateri načini so namenjeni tudi povsem profesionalni uporabi in jih v cenejših potrošniških modelih še ni moč uporabljati. Tu mislimo predvsem na upravljanje prek povezave Wi-Fi ali pa mrežne povezave Ethernet.

V nadaljevanju je podan pregled posameznih načinov daljinskega upravljanja.

- Kabelsko upravljanje: ta način upravljanja (slika 2) je izmed vseh najstarejši, saj je v različnih oblikah v fotografstvu obstajal skoraj od samih začetkov pred več kot 100 leti. V primeru fotoaparatov EOS se uporablja tudi že od samega začetka serije in je po zasnovi najbolj preprosto v svojem delovanju. V poenostavljeni obliki je sestavljeno le iz konektorja, ki ga priključimo v fotoaparat, kabla z vsaj dvema vodnikoma ter po možnosti enim stikalom. Stikalo niti ni obvezno, saj bi ga lahko simulirali z ročno sklenitvijo obeh vodnikov. V primeru, da fotoaparat v fazi delovanja zazna, da sta dva vodnika sklenjena, sproži zaslonko in zajame sliko. Podprta funkcija upravljanja je zajem posnetka z opcijo predhodne izostritve slike.

Slika 2: Primera kabelskih daljinskih upravljalnikov [4]

- IR upravljanje: tudi ta način upravljanja podpira večina sodobnih fotoaparatov. Poteka na način, da fotoaparat na daljavo zaznava signale, poslane v obliki IR svetlobe. Pri uporabi je treba biti pozoren, da je upravljalnik usmerjen proti IR sprejemniku na sprednji strani fotoaparata. V nasprotnem primeru fotoaparat ne zazna signala in ne izvede želene operacije. Razdalja oziroma doseg možnega upravljanja je pogojen tudi z dobrimi pogoji za uporabo. V primeru motenj, povezanih z močno sončno svetlobo, obstaja možnost, da fotoaparat posameznih IR signalov ne sprejme pravilno. Domet upravljanja je običajno 5 m. Ker omenjeni način ni povsem 100-odstotno zanesljiv, je priporočen predvsem za nekritične aplikacije. Podprte funkcije

upravljanja so zajem posnetka, zajem posnetka z 2-sekundnim zamikom ali začetek in ustavitev snemanja videa. V diplomskem delu smo uporabili model daljinca RC-6 (slika 3).

Slika 3: IR daljinski upravljalnik RC-6 [4]

- Upravljanje prek USB-povezave: pri uporabi te povezave je mogoče fotoaparat priključiti na napravo, ki deluje v načinu gostitelja USB-povezave (angl. host) in podpira protokol PTP (angl. *Picture Transfer Protocol*) za komunikacijo s slikovnimi napravami. Podjetje Canon za operacijski sistem Windows ponuja brezplačno aplikacijo za upravljanje na omenjeni način. Aplikacija (slika 4) uporabniku omogoča, da na daljavo nastavlja praktično vse nastavitve fotoaparata, hkrati pa omogoča zajem posnetkov, izostritev slike, pregled v živem pogledu in tudi pregled že zajetih posnetkov kar iz pomnilnika naprave.

Slika 4: Prikaz vmesnika za USB-upravljanje

- Upravljanje prek povezave Wi-Fi: v letu 2013 je podjetje Canon predstavilo prva dva modela fotoaparatov (EOS 6D in EOS 70D [5]) z vgrajeno funkcionalnostjo Wi-Fi za prenos podatkov. Hkrati je predstavilo tudi brezplačni mobilni aplikaciji na platformah Android in iOS za daljinsko upravljanje omenjenih naprav. Na tak način je prvič omogočilo neposredno upravljanje svojih fotoaparatov prek radijskih valov. Brezžična komunikacija poteka po standardih 802.11b/g/n in ima doseg do približno 30 m oddaljenosti od naprave. Možnosti upravljanja so večinoma primerljive z USB-upravljanjem.

- Upravljanje prek mrežne povezave Ethernet: za uporabo v profesionalne namene je na temu namenjenih fotoaparatih na voljo priključek za mrežno povezavo Ethernet (slika 5). Primer uporabnosti tega načina, ki ga lahko podamo, je na primer športna priredba, kjer mogoče ni dovolj prostora za fotografa oziroma zanj ni varne lokacije, je pa dovolj prostora za postavitev fotoaparata. V tem primeru lahko takšen fotoaparat priključimo na mrežno povezavo Ethernet in ga z ustrezno programsko opremo ter računalnikom upravljamo tudi s 100 m oddaljene lokacije. Ta način je od vseh predstavljenih najhitrejši z vidika hitrosti prenosa podatkov iz fotoaparata.

Slika 5: Fotoaparat s povezavo Ethernet [6]

2.2. Obravnavane funkcije fotoaparata

Pri izdelavi mobilne aplikacije smo pozornost posvetili obravnavi funkcionalnosti, ki se navezujejo na temo našega diplomskega dela.

Pri možnostih daljinskega upravljanja nam model fotoaparata Canon 550D tovarniško omogoča upravljanje z daljinskim kabelskim sprožilcem, z IR daljinskim upravljalnikom ali pa upravljanje prek USB-vodila z ustrezno napravo. Slednja je lahko tako osebni računalnik kot pametna mobilna naprava, potrebujemo le ustrezno programsko opremo za komuniciranje in upravljanje. Fotoaparat ima za povezljivost z zunanjimi napravami na strani ohišja

gumirana vratca z vsemi vhodi in izhodi (slika 6). Vsa ožičena komunikacija in prenos podatkov med njim in zunanjimi napravami poteka prek teh priključkov.

Slika 6: Priključki na strani ohišja fotoaparata

V naši mobilni aplikaciji smo tako poskušali izdelati upravljanje za vse tri zgoraj naštete načine upravljanja. Poleg tega pa smo v aplikacijo vključili tudi možnost fotografiranja v načinu bulb. To je funkcija fotografiranja, pri kateri je lahko čas zajema posnetka (slikanja) poljuben in odvisen od želje fotografa [7]. Običajno imajo fotoaparati preddefinirane korake veljavnih časov oziroma hitrosti zaslonke. Ti se najpogosteje raztezajo po korakih v razponu približno od 1/8000 sekunde pa vse do 30 sekund. S funkcijo bulb pa je mogoče zaslonko fotoaparata držati odprto poljubno količino časa. Na tak način je omogočeno na primer fotografiranje zvezdnatega neba, saj lahko slikamo daljše časovno obdobje. Omejitev 30 sekund v tem primeru običajno ni dovolj za zajem zadostne količine svetlobe. Podrobnejši opis delovanja funkcije bulb v povezavi z aplikacijo Android bomo podali v poglavjih, ki se nanašajo na izdelavo mobilne aplikacije.

3 Operacijski sistem Android in minimalne zahteve

3.1. Splošno

Naša izbira za razvoj aplikacije na platformi Android je bila posledica več vzrokov. Android je trenutno najbolj priljubljen operacijski sistem, ki je namenjen mobilnim napravam, kot so telefoni in tablični računalniki. Posledično je nabor možnih potencialnih uporabnikov aplikacije zelo velik. Sam operacijski sistem je tudi že zelo razvit in preizkušen ter hkrati tudi precej enostaven za uporabo. Z vidika razvijalca je pomembno dejstvo, da je celotni sistem dobro dokumentiran in da ima dobra razvojna orodja za izdelavo aplikacij. Platforma Android ima na spletu zelo aktivno skupnost, na katero se lahko kot razvijalec aplikacije v primeru težav pri razvoju obrne po morebitno pomoč.

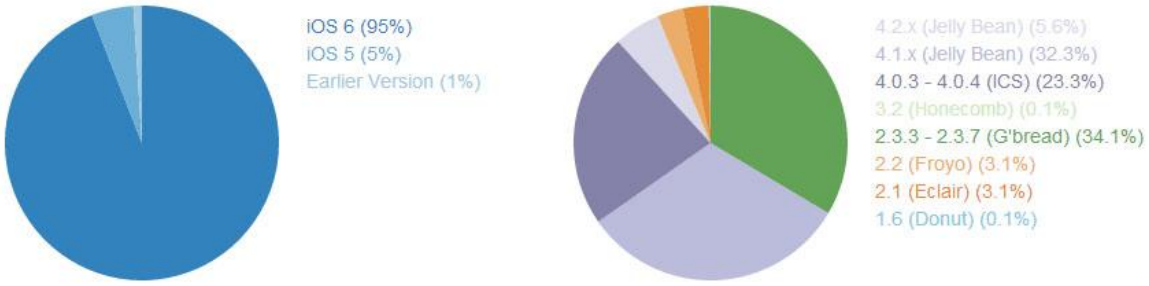
3.2. Predstavitev operacijskega sistema

Operacijski sistem Android je zasnovan okoli jedra Linux in je primarno namenjen delovanju na mobilnih napravah, še posebej na takih z zaslonom, občutljivim na dotik [8]. Prednost pred drugimi operacijskimi sistemi mu daje njegova vsestranskost, zmogljivost in prilagodljivost, saj lahko teče na skoraj poljubni kombinaciji strojne opreme. Podpira mikroprocesorsko arhitekturo ARM, MIPS in x86. Sistem je tudi povsem odprtokoden. Izvorna koda je objavljena pod licenco Apache in prosto dostopna z dovoljenji za prosto distribucijo ter modifikacijo.

Korenine razvoja segajo v leto 2003, vendar je pravi zagon v razvoju operacijski sistem dobil šele leta 2007, ko ga je kupilo in prevzelo podjetje Google. Prva mobilna naprava, ki jo je poganjal operacijski sistem Android, je bila javnosti predstavljena meseca oktobra leta 2008. Z vložkom, ki ga je Google po nakupu namenil razvoju, je Android že v nekaj letih prevzel primat v svetu po priljubljenosti ter številu mobilnih naprav, ki jih poganja.

Zadnjih nekaj let prevladuje mnenje, da je največja težava platforme Android njegova tako imenovana fregmentacija. S tem mislimo predvsem na veliko število različic operacijskega sistema, ki hkrati obstajajo na tržišču, ter na ogromno število različno zmogljivih naprav z različnimi velikostmi zaslonov. Po podatkih [9] je bilo pri analizah v letošnjem letu zaznanih že 11.868 različnih modelov naprav z operacijskim sistemom Android, ki med seboj poganjajo 8 različic omenjenega sistema. Skratka, del tega, kar je platformo Android uvrstilo na vrh, je danes tudi največja ovira za njegov nadaljnji razvoj. Z vidika programerja mobilnih aplikacij predstavlja ta fregmentacija veliko oviro, saj otežuje razvoj aplikacij. Programer mora imeti veliko znanja, da lahko razvoj aplikacije prilagodi na način, da doseže dobro delovanje na vseh želenih napravah in pri tem ohrani dobro konsistentno uporabniško izkušnjo. Če tu vzamemo za primerjavo operacijski sistem iOS [10], ki je v svetu po

priljubljenosti trenutno drugi, je razlika opazna. Okolje tega operacijskega sistema je namreč zasnovano na precej drugačen način. Naprave, ki ga poganjajo (telefoni iPhone in tablice iPad), so veliko bolj standardizirane z vidika zmogljivosti in velikosti zaslonov. Hkrati pa je pri tem tudi veliko bolje poskrbljeno za nadgrajevanje mobilnih naprav z najnovejšimi različicami operacijskega sistema. Slika 7 prikazuje primerjavo med številom aktivnih različic obeh operacijskih sistemov v mesecu juliju 2013. Podobno stane je tudi pri primerjavi uporabljenih velikosti zaslonov na obeh sistemih. Medtem ko sistem iOS uporablja le 4 različne velikosti zaslonov, jih je na sistemu Android na ducate [9]. Pri tem ne želimo trditi, da je en operacijski sistem boljši od drugega, želimo le predstaviti razlike in opozoriti na morebitne težave.



Slika 7: Primerjava aktivnih različic med sistemoma iOS in Android (julij 2013) [9]

Del uspeha operacijskega sistema Android so tudi oziroma predvsem aplikacije zanj. Poleg osnovnih sistemskih aplikacij namreč obstaja več sto tisoč različnih aplikacij, ki jih lahko uporabniki namestijo na svoje mobilne naprave in tako razširijo njihovo funkcionalnost. Nabor aplikacij sega v vse sfere življenja, lahko pa so plačljive ali brezplačne. Največja distribucija aplikacij poteka prek Googlove spletne trgovine Google Play, ki je sicer največja in najbolj priljubljena, vendar ni edina. Uporabnikova pot do aplikacije tudi ni nujno vezana na katero od spletnih trgovin. Če uporabnik to želi, si lahko aplikacijo na mobilno napravo namesti tudi ročno.

3.3. Predstavitev razvojnega okolja

Aplikacije za operacijski sistem Android je mogoče programirati v programskih jezikih C, C++ ali Java [11], vendar se za razvoj najpogosteje uporablja ravno slednji. Tudi mi smo pri izdelavi naše mobilne aplikacije uporabili programski jezik Java, zato se bomo v nadaljevanju osredotočili samo na ta programski jezik. Pomembno je omeniti, da je razvojno okolje Android podprto na operacijskih sistemih Mac OS, Linux in Windows. Pri izdelavi diplomskega dela smo aplikacijo razvijali na platformi Windows, zato se bomo v nadaljevanju sklicevali na to razvojno okolje.

Za razvoj aplikacij Java in Android je treba predhodno ustrezno konfigurirati razvojno okolje oziroma računalnik. Nameščen mora imeti tako SDK *(Software Development Kit*) Java kakor tudi SDK Android. SDK je skupek različnih orodij in knjižic za razvoj aplikacij na določeni platformi. Poleg tega za lažji razvoj potrebujemo tudi ustrezno razvojno orodje, kot je na primer Eclipse IDE (*Integrated Development Environment*) v povezavi z vtičnikom ADT (*Android Development Tools*). ADT je, kot rečeno, vtičnik za razvojno okolje Eclipse IDE, ki slednjemu doda vsa potrebna orodja ter funkcionalnosti, potrebne za razvoj mobilnih aplikacij Android [12]. Orodje Eclipse na ta način podpira in omogoča tako pisanje izvirne kode kot tudi vizualno izdelavo uporabniških vmesnikov aplikacije ter zagon aplikacije v emulatorju in tudi razhroščevanje aplikacije [13]. Omenili smo emulator Android. To je orodje, ki razvijalcu omogoča enostavno testiranje aplikacije kar znotraj razvojnega okolja, kot je Windows [14]. Emulator namreč kreira virtualno mobilno napravo, ki se obnaša skoraj identično kot resnična fizična naprava. Na tako virtualno napravo lahko nato tekom razvoja enostavno namestimo aplikacijo in jo poljubno testiramo. Največja prednost uporabe emulatorja je, da lahko na zelo enostaven in hiter način kreiramo in zaženemo virtualno napravo s poljubnimi specifikacijami za namene testiranja in razhroščevanja. Tako se tudi izognemo velikim stroškom, povezanih z nakupom naprav, če bi dejansko morali preizkusiti delovanje na različnih fizičnih napravah. Hkrati lahko ob emulatorju na razvojno okolje še vedno priključimo tudi fizično napravo Android in delovanje aplikacije preizkusimo na tak način. Pri tem je treba biti pozoren, da moramo v primeru, če želimo takšno fizično napravo uporabiti za razhroščevanje aplikacije, v nastavitvah naprave to tudi ročno omogočiti. Privzeto je namreč ta možnost izklopljena.

Kreirana aplikacija Android se v razvojnem okolju zgradi in združi v datoteko s končnico ».apk«. V njej je zbrana vsa prevedena koda Java ter potrebni viri za delovanje aplikacije. Omenjeno datoteko lahko potem tudi fizično prenesemo na napravo Android in aplikacijo ročno namestimo. Pred tem je treba na vsaki mobilni napravi v nastavitvah nastaviti, da je dovoljeno nameščanje aplikacije iz neznanih virov, saj je v nasprotnem primeru namestitev onemogočena.

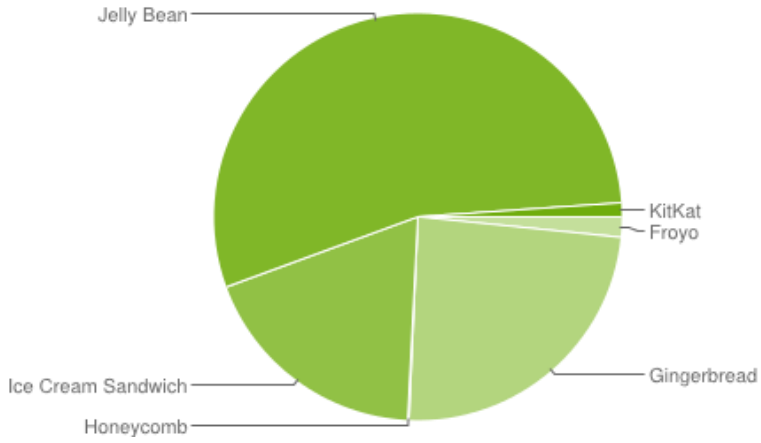
3.4. Minimalne programske zahteve

Pred začetkom razvoja naše mobilne aplikacije smo se morali seznaniti z različicami operacijskega sistema in analizirati, katere zahteve bo imela naša aplikacija pri delovanju. K temu smo pristopili na dva načina. Najprej smo morali raziskati, katera je minimalna različica operacijskega sistema, s katero lahko zadostimo vsem funkcionalnim potrebam delovanja aplikacije, nato pa še, katero različico bi bilo najbolje uporabiti za vizualni del aplikacije. Z vsako novo različico operacijskega sistema Android se namreč razširi nabor funkcionalnosti sistema, vendar te niso kompatibilne za nazaj. V primeru, da je aplikacija izdelana za določeno različico sistema, bo delovala na njej in vseh novejših, ne pa tudi na starejših. Če je namen razvijalca izdelati aplikacijo, ki jo lahko uporablja kar največ morebitnih uporabnikov, potem je treba analizirati statistike razširjenosti posameznih različic operacijskega sistema

Android. Razvijalec mora vedno uravnavati ravnovesje med željo po razvoju za novejše različice in med dejansko podporo delovanja aplikacije na mobilnih napravah. Z izdajo vsake nove različice operacijskega sistema namreč preteče tudi več mesecev, preden prve posamezne mobilne naprave začenjajo dobivati nadgradnje na to različico. V večini primerov proizvajalci napravi nudijo podporo za nadgradnje v obdobju dveh let od vstopa naprave na tržišče. Vse to privede do situacije, ko je lahko hkrati v obtoku večje število različic operacijskega sistema, vsaka s svojim naborom funkcionalnosti. Slika 8 in tabela 1 prikazujeta trenutno razširjenost posameznih različic operacijskega sistema Android.

Tabela 1: Delež posameznih različic operacijskega sistema Android (november 2013) [15]

Različica	Kodno ime	API	Odstotek
2.2	Froyo	8	1,6 %
2.3	Gingerbread	10	24,1 %
3	Honeycomb	13	0,1 %
4.0	Ice Cream Sandwich	15	18,6 %
4.1	Jelly Bean	16	37,4 %
4.2		17	12,9 %
4.3		18	4,2 %
4.4	KitKat	19	1,1 %



Slika 8: Delež posameznih različic operacijskega sistema Android (november 2013) [15]

Za zagotovitev želenih funkcionalnosti naše mobilne aplikacije mora ta omogočati izrabo IR oddajnika, generiranja in predvajanja zvočnih posnetkov, obenem pa želimo imeti še podporo za gostiteljsko USB-komunikacijo. Prva funkcionalnost je lahko neodvisna od operacijskega sistema Android oziroma je bila uradno dodana z različico 4.4. Zaradi tega za nas ni predstavljala minimalne zahteve. Drugo funkcionalnost operacijski sistem Android podpira že

od prvih različic, tako da jo podpirajo praktično vse današnje mobilne naprave. Za zagotovitev tretje funkcionalnosti pa smo ugotovili zahtevo po različici 3.1 operacijskega sistema, saj je bila z njo dodana podpora za USB-gostovanje [16].

Po opravljenih analizah smo tako prišli do ugotovitve, da moramo za zagotovitev predvidenih funkcionalnosti mobilne aplikacije zagotoviti podporo za različico operacijskega sistema 3.1 s kodnim imenom Honeycomb [17]. Na tak način bi pokrili 74,2 % vseh danes aktivnih naprav. Zatem pa smo opravili še analizo želene vizualne podobe aplikacije in prišli do spoznanja, da je bila z izdajo različice 4.0 operacijskega sistema dodana dodatna podpora za izdelavo naprednejših uporabniških vmesnikov [18]. Razlika med različico 3 in 4.0 je le 0,1 % (slika 8), tako da smo se na koncu odločili, da postavimo zahtevo po minimalni različici operacijskega sistema na vrednost 4.0 oziroma API (angl. *Application Programming Interface*) različice 15. Na tak način bi z mobilno aplikacijo še vedno podprli 74,1 % vseh danes aktivnih naprav Android.

3.5. Minimalne zahteve s strojno opremo

Kot smo že omenili, je operacijski sistem Android zelo fragmentiran. Na tržišču lahko zasledimo veliko število različnih naprav različnih proizvajalcev z različnimi strojnimi specifikacijami. To dejstvo privede to ugotovitve, da je dandanes zelo težko oziroma praktično nemogoče razviti aplikacijo, ki bi podpirala vse te različne naprave. Nabor možnih naprav, ki lahko poganjajo določeno aplikacijo, se lahko še dodatno skrči, če je delovanje aplikacije vezano na katero od strojnih lastnosti naprave. V našem primeru smo za pravilno delovanje načrtovane mobilne aplikacije poleg programskih zahtev morali zadostiti tudi določenim zahtevam po strojni opremi. Za razvoj in namene testiranja aplikacije smo imeli dostop do treh naprav, in sicer LG Nexus 5, Samsung Galaxy S2 ter Samsung Galaxy S4 Mini.

Ker smo imeli v prvem sklopu diplomskega dela namen izdelati daljinsko upravljanje fotoaparata z uporabo IR komunikacije, smo za to potrebovali mobilno napravo z IR oddajnikom. Dandanes je na tržišču kar nekaj takšnih naprav. Težava je v tem, da med različnimi proizvajalci ni implementirane standardizirane rešitve. Podpora za IR oddajnike je bila v sistem Android dodana šele z najnovejšo različico 4.4 KitKat [19], na vseh starejših različicah pa sta podpora in posledično programerski dostop do oddajnika odvisna od implementacije posameznega proizvajalca mobilne naprave. Ker smo pri izdelavi naše aplikacije imeli dostop le do ene naprave z IR oddajnikom (Samsung Galaxy S4 Mini), smo razvoj prilagodili njej. Posledično smo podprli še preostale naprave proizvajalca Samsung, ki uporabljajo isti mehanizem upravljanja IR oddajnika. Naprave preostalih proizvajalcev zaenkrat tako še niso podprte, bodo pa z nadgradnjo na različico 4.4 operacijskega sistema. V tej različici se je namreč upravljanje z IR oddajnikom implementiralo na nivoju operacijskega sistema in je neodvisno od implementacij proizvajalca naprave.

V drugem sklopu razvoja aplikacije smo obravnavali kabelsko upravljanje z uporabo izhoda za slušalke. Zahteva tukaj je bila ustrezna naprava, ki ima omenjeni izhod. Dandanes mobilno napravo, ki takšnega izhoda ne bi imela, že težko najdemo. Drugi pogoj, ki smo ga opazili šele med razvojem aplikacije, je bil, da mora naprava za pravilno delovanje upravljanja zagotoviti dovolj veliko moč signala na izhodnem priključku. Povedano drugače, izhodna glasnost mora biti dovolj velika. Od naših testnih naprav je imel le Samsung Galaxy S2 dovolj močan izhodni signal. Z nadaljnjim razvojem smo poskušali to zahtevo izničiti z opcijo, da bi bilo možno kabelsko upravljanje izvesti v kombinaciji z mikrokontrolerjem Arduino oziroma z uporabo namenskega elektronskega vezja z operacijskim ojačevalnikom.

V tretjem sklopu, ki smo ga obravnavali v diplomskem delu, smo zasledili zahtevo po strojni in programski podpori USB-gostovanja (angl. host). Slednja je bila pogoj, da smo lahko mobilno napravo uporabili za upravljanje fotoaparata prek USB-povezave. Težava je v tem, da tega načina povezovanja USB-naprav ne podpira vsaka mobilna naprava, saj zahteva, da proizvajalec nudi tako strojno kot tudi programsko implementacijo. Med napravami, ki smo jih imeli za namene razvoja in testiranja USB, gostovanje podpirata le napravi LG Nexus 5 in Samsung Galaxy S2.

4 Razvoj aplikacije Upravljalnik fotoaparata

V tem poglavju bomo predstavili celoten razvoj aplikacije Upravljalnik fotoaparata. Za uvod bomo opisali izdelavo osnovne strukture aplikacije, v nadaljevanju pa tudi njene glavne funkcionalne dele.

4.1. Izdelava osnovne strukture aplikacije

Izdelavo mobilne aplikacije smo začeli z razmišljanjem o tem, kakšno aplikacijo želimo izdelati, katere elemente želimo vgraditi vanjo in kako želimo strukturirati uporabniški vmesnik aplikacije. Od samega začetka smo imeli željo, da bo izdelana aplikacija enostavna za uporabo vsakemu povprečnemu uporabniku. V ta namen smo želeli funkcije čim bolj poenostaviti. Za njihovo lažje upravljanje smo si, kjer je bilo to smiselno, zamislili uporabo velikih gumbov.

Prvi korak, ki smo ga pri razvoju naredili, je bil kreiranje osnovne aplikacije. To smo storili z uporabo programskega čarovnika, ki je vgrajen v razvojno orodje Eclipse. Dobljena aplikacija je imela obliko programa Hello world [20]. Pripravljene je imela že vse osnovne mape za izvorno kodo in tudi za preostale datoteke, ki so potrebne za prvo namestitev aplikacije. V čarovniku smo med drugim določili ime za našo aplikacijo, ikono in temo videza aplikacije. Za slednjo smo privzeto izbrali temo Holo Dark [21]. Razlog to našo odločitev je bil, ker je ta tema temna. Na ta način smo nameravali zmanjšati vpliv svetlobe zaslona mobilne naprave na fotografiranje v primeru, da se napravo uporablja v bližini fotoaparata v temnem okolju oziroma v nočnem času.

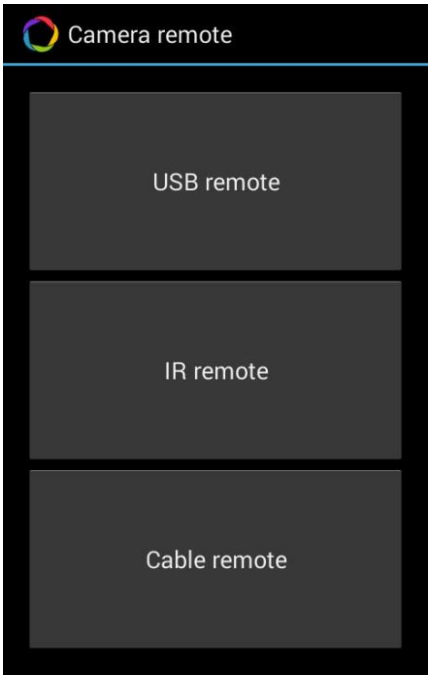
Na tem mestu želimo najprej predstaviti nekaj osnovnih gradnikov na platformi Android, katerih poznavanje je potrebno za lažje razumevanje nadaljnjega opisa izdelave aplikacije. Najosnovnejši in najpomembnejši od teh je *Dejavnost* (angl. *Activity*). Dejavnost je tisto, kar uporabnik vidi, ko upravlja z aplikacijo, tisto, s čimer ima interakcijo in kar dejansko običajno izvaja določeno funkcionalnost aplikacije. Vsak dejavnost je običajno sestavljen iz dveh delov. Prvi del je datoteka XML (angl. *Extensible Markup Language*) z izdelanim uporabniškim vmesnikom, drugi del pa je dejanski razred Java, ki implementira sistemski razred *Activity* in ki predstavlja dejanski program, s katerim upravlja uporabnik. V nadaljevanju bomo za pojem Activity uporabljali izraz okno.

Po pripravi osnovne strukture aplikacije smo razvoj nadaljevali z izdelavo posameznih elementov naše aplikacije. Najprej smo izdelali uvodno okno (slika 9), ki se uporabniku prikaže takoj ob odprtju aplikacije. Zamišljena funkcionalnost tega okna je bila omogočiti uporabniku hitro navigacijo na posamezni način upravljanja, ki ga želi uporabiti. V ta namen smo izdelali enostavno okno s tremi velikimi gumbi, ki skupaj zavzamejo celoten zaslon.

Predvideni gumbi imajo nazive USB-upravljanje, IR-upravljanje in kabelsko upravljanje. Ob kliku na posamezni gumb se uporabniku odprle okno za izbrani način upravljanja.

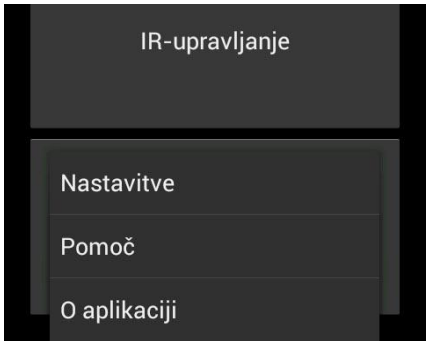
Slika 9: Uvodno okno aplikacije

Aplikacija, ki smo jo izdelali, je zasnovana na način, da je privzeti izbrani jezik uporabniškega vmesnika angleščina (slika 10), slovenščina pa je podprta kot možni lokalni jezik. To smo storili zaradi statusa angleščine kot svetovnega jezika in zaradi načina delovanja lokalizacije v operacijskem sistemu Android. Slednji je zasnovan na način, da je jezik uporabniškega vmesnika posamezne aplikacije vezan na nastavljeni jezik celotnega operacijskega sistema [22]. V primeru, da je izbrani jezik naprave slovenščina, bo aplikacija delovala v slovenščini. V primeru, da pa je na mobilni napravi nastavljen katerikoli drug jezik, bo pa aplikacija vedno uporabila privzete angleške prevode. S tem smo dosegli, da bo v primeru, če aplikacijo namesti uporabnik iz tujine, imel uporabniški vmesnik preveden v angleščino, ki mu je verjetneje bliže kakor slovenščina. S programerskega vidika je za dosego takšnega delovanja treba biti dosleden pri pisanju izvorne kode. Paziti je treba, da v kodo ne vključimo fiksnih besedil, ampak da namesto besedil vedno uporabljamo spremenljivke, ki se definirajo izven izvorne kode. Android v ta namen predvideva uporabo preddefiniranih map za posamezne jezike, iz katerih nato bere vrednosti spremenljivk in jih uporabi kot besedilo v izvorni kodi. Z nadaljnjim razvojem bi bilo v aplikacijo mogoče dodati še prevode za druge svetovne jezike, vendar v sklopu tega diplomskega dela tega nismo obravnavali.



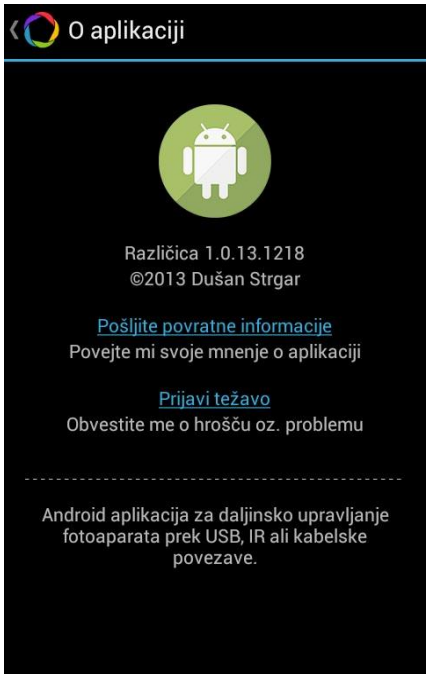
Slika 10: Angleški prevod uvodnega okna

Za dosego ustreznega delovanja aplikacije smo morali v aplikacijo vključiti tudi nekatera pomožna okna. To so nastavitve aplikacije, pomoč uporabnikom in okno z informacijami o aplikaciji. Vse te funkcije so mobilni napravi dostopne prek tipke za dostop do menija, ob pritisku na katero se na zaslonu odpre manjše okno z omenjenimi gumbi (slika 11). S klikom na posamezen gumb se odpre ustrezno novo okno.



Slika 11: Meni aplikacije

V oknu *O aplikaciji* so morebitnemu uporabniku prikazane dodatne informacije. Navedeni so različica aplikacije, ime razvijalca in povezave za oddajo elektronskega sporočila o zaznanih hroščih ali težavah pri uporabi. Sporočiti je možno tudi morebitne želje in predloge za izboljšave.

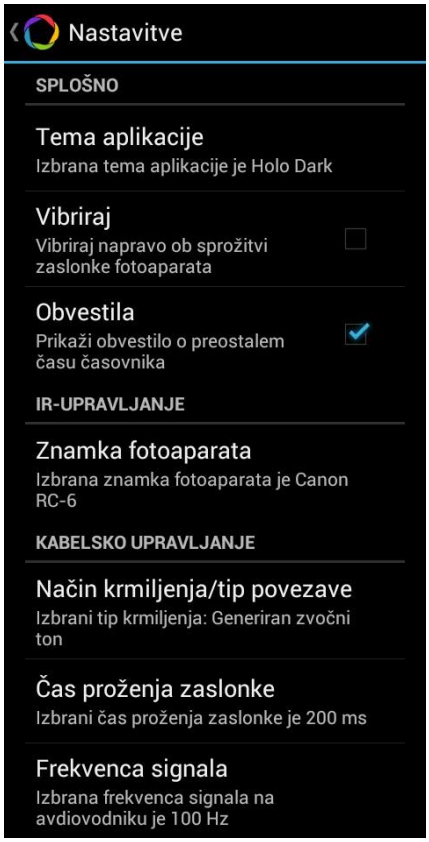


Slika 12: Okno *O aplikaciji*

Okno *Pomoč* je v aplikaciji predvideno za prikaz navodil za uporabo aplikacije, za pojasnilo različnih opcij v nastavitvah in za pojasnitev uporabe posameznih različnih načinov upravljanja fotoaparata. V sklopu tega diplomskega dela smo izdelali le prikazno okno brez besedil pomoči. Dodati bi jih bilo mogoče z nadaljnjim razvojem mobilne aplikacije.

Zadnje okno v sklopu pomožnega menija je okno z nastavitvami aplikacije (slika 13). V njem so na enem mestu zbrane vse pomembnejše nastavitve, ki se uporabljajo pri delovanju aplikacije. Posamezne nastavitve so ločene glede na sklop, ki ga obravnavajo. Splošne nastavitve so nastavitve teme aplikacije, vibriranja naprave ob fotogafiranju in prikaza obvestila ob izvajanju časovnika. Poleg tega pa imamo tudi nastavitve za sklop IR upravljanja in za kabelsko upravljanje. V okviru prvih smo uporabniku omogočili možnost izbire znamke fotoaparata, ki ga želi z aplikacijo upravljati. Ta nastavitev je zaenkrat fiksna in ni nastavljiva. Dodali smo jo zato, da lahko uporabniku podamo informacijo o podprtih fotoaparatih, predvideli pa smo tudi, da bi bilo aplikacijo v prihodnje možno z nadaljnjim razvojem razširiti in ji dodati podporo tudi za druge znamke fotoaparatov. Pri nastavitvah za kabelsko upravljanje je glavna nastavitev izbira načina krmiljenja. Uporabnik ima na voljo tri možnosti, in sicer generiran zvočni signal, predposnet zvočni signal in mikrokontroler Arduino. V primeru, da je za omenjeno nastavitev izbrano generiranje zvočnega signala, lahko uporabnik definira tudi frekvenco in trajanje generiranega signala. V nasprotnem primeru sta ti dve nastavitvi onemogočeni. Možne izbire frekvence so 80 Hz, 100 Hz, 150 Hz in 200 Hz, možni časi za proženje zaslonke pa so 200 ms, 500 ms in 1000 ms.

S podanimi nastavitvami si lahko uporabnik nekoliko prilagodi delovanje aplikacije, če mu privzete nastavitve iz kateregakoli razloga ne ustrezajo.



Slika 13: Okno *Nastavitve*

Poleg zgoraj opisanih oken, ki predstavljajo ogrodje aplikacije, bomo v nadaljevanju predstavili še tri osnovna okna, ki predstavljajo glavno funkcionalnost aplikacije. To so okna za IR-upravljanje, kabelsko upravljanje in USB-upravljanje.

4.2. Izdelava IR daljinskega upravljanja

Eden izmed ciljev, ki smo si jih zastavili pri izdelavi diplomskega dela, je bil, da bi v okviru mobilne aplikacije Android simulirali delovanje tovarniškega IR daljinskega upravljalnika. To smo poskušali storiti z metodo obratnega inženirstva. Naša zamisel je bila, da bi z uporabo mikrokontrolerja Arduino, ustreznega vezja in programa dešifrirali IR signale, ki jih za posamezne operacije generira tovarniški daljinec. Pridobljene podatke o signalu bi nato uporabili za generiranje signalov v okviru naše aplikacije.

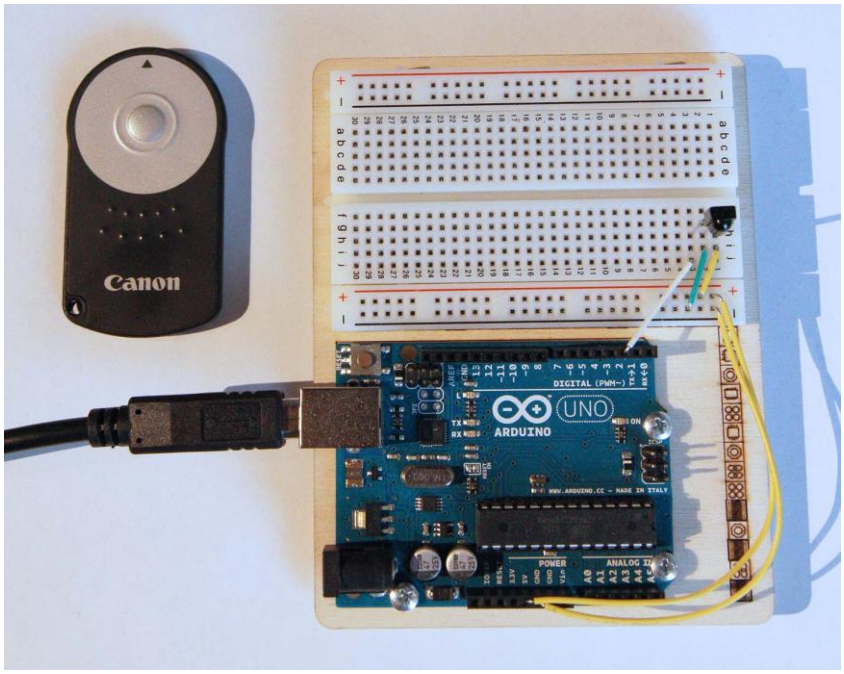
4.2.1. Analiza tovarniškega IR daljinskega upravljalnika

Omenili smo že, da modeli fotoaparatov Canon EOS dandanes v večjem številu podpirajo daljinsko upravljanje s pomočjo IR komunikacije. Za fotoaparat Canon EOS 550D se lahko v ta namen uporabijo modeli tovarniških upravljalnikov RC-1, RC-5 ali RC-6 (slika 3).

Slednjega smo uporabili v sklopu izdelave tega diplomskega dela. V prejšnjih poglavjih smo omenili, da model RC-6 podpira operaciji takojšnjega slikanja kot tudi slikanja z 2-sekundnim zamikom po pritisku na gumb za sprožitev. Ker za omenjeni tovarniški model upravljalnika nismo našli vseh potrebnih specifikacij, smo po krajšem iskanju na svetovnem spletu našli funkcionalno podobno napravo [23] s podanimi specifikacijskimi podatki. Ti so valovna dolžina IR svetlobe in nosilna frekvenca komunikacijskega signala. Te podatke smo tudi uporabili v nadaljevanju izdelave. Analiziranje delovanja smo začeli z izdelavo vezja za branje in zajem IR signalov. V ta namen smo uporabili mikrokontroler Arduino Uno (slika 14) [24]. To je razvojna plošča, zasnovana okoli procesorja ATmega328, ki ima med drugim 14 digitalnih vhodov/izhodov, 6 analognih vhodov in USB-povezavo. Bistvena lastnost kontrolerja Arduino je, da ga je mogoče poljubno programirati. Programiranje poteka v lastnem razvojnem okolju z imenom Arduino Software IDE, operira pa lahko s programskima jezikoma C in C++. Kot orodje je zelo priljubljeno, saj omogoča razvoj enostavnih in tudi kompleksnih elektronskih vezij, ki lahko na podlagi vhodnih in izhodnih signalov ter krmilnega programa interaktivno operira s fizičnim svetom.

Slika 14: Mikrokontroler Arduino Uno [24]

Vezje (slika 15), ki smo ga izdelali za analizo IR signalov, je vsebovalo mikrokontroler Arduino in IR sprejemnik TSOP38238 [25]. Za ta model sprejemnika smo se odločili zaradi ustrezne podpore valovni dolžini svetlobe, in sicer 940 nm. Ta podatek smo pridobili iz specifikacije upravljalnika. Mikrokontroler je bil prek USB-povezave povezan z osebnim računalnikom. Prek te USB-povezave je potekalo napajanje celotnega vezja ter tudi komunikacija med računalnikom in vezjem. IR sprejemnik smo napajali neposredno iz mikrokontrolerja Arduino. To smo storili tako, da smo nogo 2 vezali na maso, nogo 3 pa na 5-voltni izhod mikrokontrolerja. Podatkovni izhod (noga 1) iz sprejemnika smo vezali na digitalni vhod 2 kontrolerja. IR sprejemnik deluje na način, da sprejeti signal pretvori v digitalne impulze, ki jih potem preberemo z mikrokontrolerjem.



Slika 15: Vezje za zajem IR signalov

Snovanju in izdelali vezja je sledila izdelava programa za branje IR signalov. Za pomoč in osnovo pri izdelavi programa Arduino smo vzeli primer že razvitega programa [26], ki smo ga nato še nekoliko prilagodili našim potrebam.

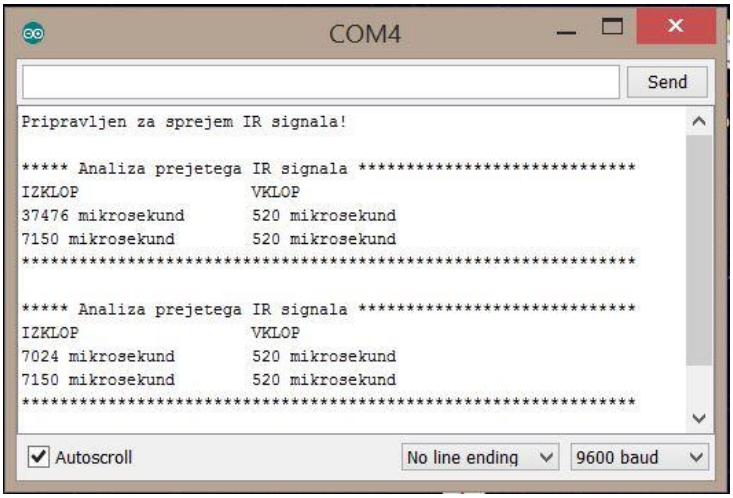
Spodaj je navedena programska koda programa Arduino za detekcijo in branje časov impulzov sprejetega IR signala.

```
/* Arduino program za analizo časov prejetega IR signala */
#define vhodnaNoga_PIN  PIND
#define vhodnaNoga      2
#define maxdolzinapulza 65000 // maksimalna dolzina IR pulza - 65 milisekund
#define Resolucija 10
unsigned int pulzi[200][2];  // matrika pulzov signala; shranimo do 200 parov pulzov
unsigned int trenutnipulz = 0; // indeks obravnavanega pulza

// zacetek programa
void setup(void) {
  Serial.begin(9600); // odpri serijska vrata, nastavi hitrost prenos podatkov na 9600 bps
  Serial.println("Pripravljen za sprejem IR signala!"); // izpisi tekst
}
// glavna loop zanka
void loop(void) {
  unsigned int visokipulz, nizkipulz;  // začasne variable
  visokipulz = nizkipulz = 0; // na začetnu setiramo dolžino na 0
  while (vhodnaNoga_PIN & (1 << vhodnaNoga)) { // visoka vrednost na nogi; posegamo
direktno v register nogic za hitrejšo obdelavo pulzov
    visokipulz++; // stejemo mikrosekunde
    delayMicroseconds(Resolucija);
    if ((visokipulz >= maxdolzinapulza) && (trenutnipulz != 0)) { // pulz je predolgi oz.
se je izvajanje končalo
      printpulses();
      trenutnipulz=0;
      return;
```

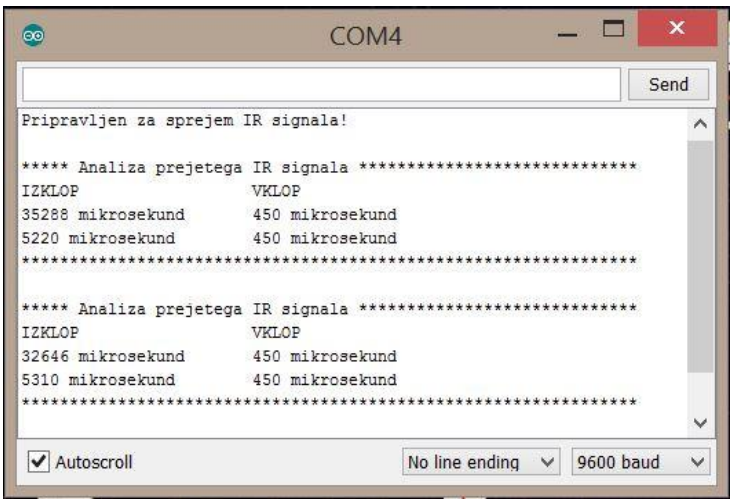
```
    }
  }
  pulzi[trenutnipulz][0] = visokipulz;
  while (! (vhodnaNoga_PIN & _BV(vhodnaNoga))) {// nizka vrednost na nogi
    nizkipulz++; // stejemo mikrosekunde
    delayMicroseconds(Resolucija);
    if ((nizkipulz >= maxdolzinapulza)  && (trenutnipulz != 0)) {
      printpulses();
      trenutnipulz=0;
      return;
    }
  }
  pulzi[trenutnipulz][1] = nizkipulz;
  trenutnipulz++;
}

void printpulses(void) {
  Serial.println("\n\r***** Analiza prejetega IR signala ***************************");
  Serial.println("IZKLOP\t\t\tVKLOP");
  for (unsigned int i = 0; i < trenutnipulz; i++) {
    Serial.print(pulzi[i][0] * Resolucija, DEC);
    Serial.print(" mikrosekund\t");
    Serial.print(pulzi[i][1] * Resolucija, DEC);
    Serial.println(" mikrosekund");
  }
  Serial.println("*****************************************************************");
}
```

Program je zasnovan na način, da ob zagonu nadzira sprejemnik za morebitni IR signal. V trenutku, ko ga zazna, prične z beleženjem in izpisom posameznih časovnih parov, ko je bil signal na logični vrednosti 0 in 1. Glede delovanja programa bi lahko posebej izpostavili način detekcije signala. Običajno se za branje podatkov na vhodih v mikrokontroler uporabi ukaz *digitalRead()*. Je relativno počasen, zato je bil za detekcijo zelo kratkih časov med posameznimi impulzi IR signala neprimeren. Na tak način smo dobivali nezanesljive čase IR impulzov. Z nadaljnjim razvojem smo prešli na branje podatkov neposredno iz registra digitalnih vhodov, kar je časovno veliko hitreje kot prek namenske funkcije za branje. Z uporabo tega načina smo nato dobivali in beležili konsistentne čase zaznanih signalov.

Slika 16 in slika 17 prikazujeta čase posameznih impulzov IR signala, kot ga generira tovarniški daljinski upravljalnik RC-6 za posamezno od obeh možnih operacij upravljanja. Na obeh slikah sta prikazana dva zajeta sprejema signalov. Iz slik je razvidno tudi, da so generirani signali časovno zelo kratki in obsegajo le dva para identičnih paketov impulzov. Signala za posamezni operaciji *Slikaj* in *Slikaj z zakasnitvijo* se med sabo razlikujeta le po nekoliko spremenjenih zakasnitvah med impulzi. V primeru operacije *Slikaj* imamo dva identična pulza, pri čemer IR oddajnik približno 520 mikrosekund oddaja signal in nato 7,15 milisekunde miruje, preden se postopek še enkrat ponovi. V primeru operacije *Slikaj z zakasnitvijo* je čas oddajanja nekoliko krajši kot v prvem primeru (450 mikrosekund), čas mirovanja pa je tudi krajši in znaša približno 5,3 milisekunde. Tako smo dobili približne podatke o strukturi IR signalov, ki pa smo jih morali preizkusiti še v praksi. V tem trenutku namreč še nismo bili povsem prepričani, ali je kontroler Arduino pravilno zabeležil čase signalov in če so pridobljeni časi signalov sploh pravilni oziroma v kolikšni meri odstopajo od resničnih vrednosti.



Slika 16: Časi IR signala za operacijo *Slikaj*



Slika 17: Časi IR signala za operacijo *Slikaj z zakasnitvijo*

Nekaj, česar iz omenjenih podatkov ni bilo moč razbrati, pa je bila nosilna frekvenca oddajnega IR signala. Komunikacija med napravami namreč poteka na način, da se podatki vedno pošiljajo z določeno frekvenco. V primeru našega oddajnika je ta frekvenca približno 33 KHz, tako da je čas posamezne periode oziroma impulza približno 30,3 mikrosekunde. Na ta način smo izvedeli, koliko period frekvence traja oddajanja in mirovanje signala. Vrednost omenjene frekvence smo pridobili iz specifikacije upravljalnika.

Poleg predstavljenega načina analiziranja IR komunikacije z mikrokontrolerjem Arduino smo na svetovnem spletu našli tudi analizo IR signalov z uporabo elektronskega osciloskopa [27]. Avtorjeva odkritja se z manjšimi odstopanji ujemajo z našimi ugotovitvami glede oblike in

trajanja posameznih IR signalov. Tako smo dobili potrditev, da je bila naša analiza z uporabo mikrokontrolerja Arduino pravilna in da smo na takšen način pridobili pravilne podatke.

4.2.2. Šestnajstiški format ProntoEdit

Potem ko smo pridobili potrebne podatke o strukturi signala, smo delo nadaljevali z izdelavo kode signalov ProntoEdit [28]. To je strukturiran šestnajstiški format zapisa, ki se pogosto uporablja za zapis IR signalov. Njegova prednost je, da predstavi signal na človeku laže berljiv način. Zapis je strukturiran na način, da je sestavljen iz niza šestnajstiških besed. Niz prvih štirih besed definira strukturo signala, vse nadaljnje besede pa čase impulznih sekvenc. Prva beseda je vedno 0000 in pove, da je signal naučen. Naslednja beseda predstavlja frekvenco nosilnega signala, s katerim se naj signal oddaja. Pridobljena je po enačbi *Frekvenca v KHz = 1000000/(N * 0,241246)*, kjer *N* predstavlja decimalno vrednost šestnajstiškega zapisa. Tretja in četrta beseda v zaporedju predstavljata število ponovitev posameznih parov sekvenc. Za uvodnimi štirimi šestnajstiškimi besedami se namreč nahajajo posamezni pari sekvenc vklopa in izklopa IR signala. Vrednosti so podane v odvisnosti od časa oziroma trajanja periode nosilnega signala. Če za primer vzamemo naš podatek, da oddajanje signala traja povprečno 490 mikrosekund, lahko glede na podatek nosilne frekvence signala izračunamo, da to znaša približno 16 period signala. Dobljeno vrednost nato pretvorimo v šestnajstiški zapis in dobimo prvo polovico para sekvence, čas oddajanja signala. Za drugo polovico sekvence moramo na identičen način pretvoriti vrednost časa o mirovanju signala za posamezno operacijo. Ker se sekvenca v primeru našega IR signala ponovi dvakrat, moramo to storiti tudi pri zapisu šestnajstiške oblike signala. Sestavljeni zapisi v obliki ProntoEdit so zapisani na naslednji način:

Decimalni zapis:

- operacija *Slikaj*: 0 138 2 0 16 232 16 232
- operacija *Slikaj z zakasnitvijo*: 0 138 2 0 16 170 16 170

Šestnajstiški zapis:

- operacija *Slikaj*: 0000 008a 0002 0000 000f 00e8 000f 00e8
- operacija *Slikaj z zakasnitvijo*: 0000 008a 0002 0000 000f 00aa 000f 00aa

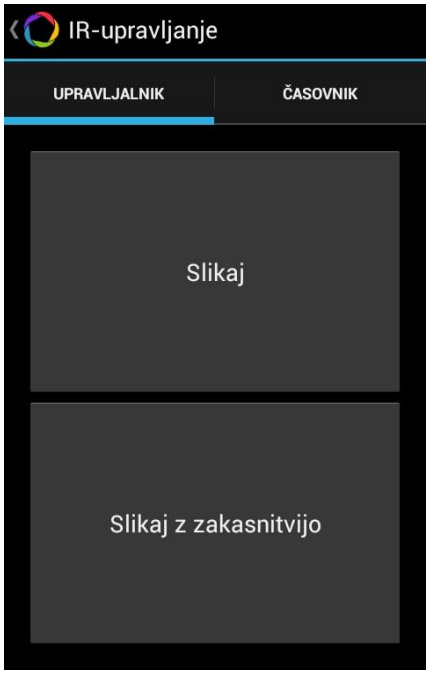
Oba zapisa smo v nadaljevanju uporabili kot osnovo za izdelavo mobilne aplikacije.

4.2.3. Izdelava rešitve v mobilni aplikaciji

Po opravljeni analizi IR komunikacije in iskanja ustreznih IR ukaznih signalov smo začeli z izdelavo funkcionalnosti v naši mobilni aplikaciji.

Prvi korak pri izdelavi je bila izdelava ustreznega uporabniškega vmesnika. Cilj pri tem je bil, da bi kar se da dobro odražal funkcionalnost tovarniškega daljinskega upravljalnika RC-6. Iz

tega razloga smo vmesnik izdelali na način, da z velikimi gumbi omogočamo tako izbiro operacije takojšnega slikanja kot tudi operacije slikanja z zakasnitvijo. Na sliki 18 je prikazan videz uporabniškega vmesnika za ta način upravljanja.



Slika 18: Uporabniški vmesnik za IR daljinsko upravljanje

Ko smo zaključili z izdelavo uporabniškega vmesnika, smo začeli z razvojem funkcionalnosti generiranja IR signalov na mobilnih napravah. Pri tem smo imeli več ovir, ki smo jih morali rešiti. Kot smo že omenili, v strojnih zahtevah mobilne aplikacije za obravnavani način upravljanja potrebujemo napravo z vgrajenim IR oddajnikom. Pri razvoju smo imeli dostop do ustrezne naprave v obliki mobilnega telefona Samsung Galaxy S4 Mini, zato smo razvoj v prvi fazi implementirali za to mobilno napravo in posledično tudi za druge podobne naprave proizvajalca Samsung. IR komunikacija na platformi Android pred izidom operacijskega sistema različice 4.4 KitKat namreč še ni bila standardizirana. Zaradi tega ima vsak proizvajalec izdelan svoj način upravljanja z IR oddajnikom. Ti načini običajno niso javno dokumentirani, zato tudi ni neposrednega javno dostopnega programskega vmesnika za njihovo uporabo. Za našo implementacijo komunikacije smo za osnovo uporabili objavljeno odprtokodno rešitev [29] z že izdelanim načinom dostopa do sistemskega razreda, ki implementira upravljanje z oddajnikom. Tega smo na spodaj prikazani način inicializirali tudi v naši mobilni aplikaciji.

Funkcija *irInit()*:

```
public void irInit() {
	try {
		irdaService = getSystemService("irda");
		Class c = irdaService.getClass();
		Class p[] = { String.class };

		irWrite = c.getMethod("write_irsend", p);
	} catch (NoSuchMethodException e) {
		e.printStackTrace();
	} catch (NullPointerException e) {
		e.printStackTrace();
	}
}
```

Sestavni del implementacije komunikacije je tudi funkcija *hex2dec()*. Njena funkcija je, da podani signal, zapisan v šestnajstiški obliki (oblika zapisa ProntoEdit), pretvori v decimalno obliko in jo vrne kot niz, da se lahko ta nato uporabi za generiranje IR signala.

Funkcija *hex2dec()*:

```
public String hex2dec(String irData) {
	List<String> list = new ArrayList<String>(Arrays.asList(irData.split(" ")));
	list.remove(0); // prva beseda, nima uporabnega podatka
	int frequency = Integer.parseInt(list.remove(0), 16); // frekvenca nosilnega sig.
	list.remove(0); // število sekvenc 1
	list.remove(0); // število sekvenc 2

	for (int i = 0; i < list.size(); i++) {
		list.set(i, Integer.toString(Integer.parseInt(list.get(i), 16)));
	}
	frequency = (int) (1000000 / (frequency * 0.241246));
	list.add(0, Integer.toString(frequency));

	irData = "";
	for (String s : list) {
		irData += s + ",";
	}
	return irData;
}
```

V spodaj prikazani kodi lahko vidimo klic funkcije *btn_irCable_shoot_onClick* ob pritisku na gumb *Slikaj*. Omenjena funkcija generira in dodaja IR signal. V tem trenutku se uporabita tudi predhodno inicializirani razred in funkcija *hex2dec()*.

Funkcija *btn_irCable_shoot_onClick()*:

```
public OnClickListener btn_irCable_shoot_onClick = new OnClickListener() {
	@Override
	public void onClick(View v) {
	String data = hex2dec(getResources().getString(R.string.ircode_ shoot));
		if (data != null) {
			irWrite.invoke(irdaService, data);
		}
	}
};
```

V času testiranja smo ugotovili, da ukazi ProntoEdit, ki smo jih kreirali, niso delovali po pričakovanjih. Operacija *Slikaj* sprva sploh ni delovala, *Slikanje z zakasnitvijo* pa sicer je, vendar zelo nekonsistentno. Z nadaljnjim testiranjem in razhroščevanjem smo preizkusili več različnih nosilnih frekvenc, nekoliko pa smo tudi spremenili čase oddajanja in mirovanja signala. Naš šestnajstiški ukazni niz za operacijo *Slikaj* smo tako spremenili iz *0000 008a 0002 0000 000f 00e8 000f 00e8* v *0000 007e 0002 0000 0010 00f0 0010 00f0*. Ugotovili smo tudi, da je generirani izhodni signal prešibak in ga fotoaparat običajno ne zazna. Iz tega razloga smo tudi modificirali naš zapis signala na način, da smo povečali število ponovitev sekvenc oddajanja in mirovanja oddajnika. Iz oblike *0000 007e 0002 0000 0010 00f0 0010 00f0* z dvema ponovitvama sekvenc smo tako prišli do niza *0000 007e 0014 0000 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0 0010 00f0* s štirinajstimi ponovitvami sekvence oddajanja in mirovanja. Po vseh omenjenih modifikacijah zapisa signala smo na koncu prišli do pravilno delujočih ukaznih nizov. Ti so v kombinaciji z mobilno aplikacijo in našim testnim mobilnikom delovali primerljivo dobro kakor tovarniški daljinski upravljalnik Canon RC-6.

Kot smo že omenili, je zgoraj opisani način upravljanja specifičen za določene podprte naprave Samsung. Omenili pa smo tudi že, da je operacijski sistem Android z različico 4.4 KitKat dobil standardiziran način upravljanja z vgrajenim IR oddajnikom. Omenjena različica je sicer izšla šele tekom naše izdelave diplomskega dela, vendar smo se vseeno odločili, da poskusimo implementirati IR komunikacijo tudi na ta način. Razlog za našo odločitev je bil predvsem v dodatni podpori za večje število naprav, ki bodo lahko izrabile to funkcionalnost naše aplikacije. Po pregledu dokumentacije [30] smo ugotovili, da je novi način upravljanja zelo podoben načinu, ki ga je za lastne naprave implementiralo podjetje Samsung in katerega smo že uporabili pri izdelavi aplikacije. Z nekaj dodatnega vloženega dela smo tako obstoječo rešitev dopolniti tudi za novi način upravljanja. Delovanje po novi in tudi stari implementaciji smo ločili s preverjanjem različice operacijskega sistema. Če je trenutna različica, pod katero teče aplikacija, različica 4.4 Kitkat, potem aplikacija deluje po novem načinu. Sicer pa deluje po starem.

V različici Android 4.4 je za implementacijo IR komunikacije predviden novi programski vmesnik, razred *ConsumerIrManager*. Ta skrbi za inicializacijo oddajnika ter za generiranje in oddajanje IR signala. Po pregledu dokumentacije smo zasledili, da novi način delovanja predvideva nekoliko spremenjene vhodne podatke kakor naša obstoječa implementacija za naprave Samsung. Razred *ConsumerIrManager* namreč za generiranje IR signala potrebuje podatek o nosilni frekvenci signala ter matriko celoštevilskih števil s podatki o izmeničnih vzorcih časov oddajanja in mirovanja. Največja sprememba med starim in novim načinom delovanja je tako bila dopolnitev že omenjene funkcije *hex2dec()*. Spomnimo, da slednja pretvori podatke iz oblike zapisa ProntoEdit v obliko, primerno za IR oddajnik. Funkcijo smo predelali na način, da sedaj vrne matriko celoštevilskih števil. V tej matriki so navedeni časi vklopa in izklopa signala v mikrosekundah. Novo funkcijo smo poimenovali *hex2dec_KitKat()*.

Funkcija *hex2dec_KitKat()*:

```
public int[] hex2dec_KitKat(String irData) {
	List<String> list = new ArrayList<String>(Arrays.asList(irData.split(" ")));
	list.remove(0); // prva beseda, nima uporabnega podatka
	int frekvenca = Integer.parseInt(list.remove(0), 16); // frekvenca nosilnega sig.
	list.remove(0); // število sekvenc 1
	list.remove(0); // število sekvenc 2

	frekvenca = (int) (1000000 / (frekvenca * 0.241246));
	double perioda = 0.0;
	perioda = (1 / (double) frekvenca) * 1000000;

	List<Integer> vzorec = new ArrayList<Integer>();
	for (int i = 0; i < list.size(); i++) {
		vzorec.add((int) ((double) (Integer.parseInt(list.get(i), 16) * perioda)));
	}
	vzorec.add(0, frekvenca);

	int[] vrni = new int[vzorec.size()];
	for (int i = 0; i < vrni.length; i++)
		vrni[i] = vzorec.get(i);
	return vrni;
}
```

Nekoliko smo predelali tudi preostale funkcije za operiranje z IR oddajnikom. Te sedaj vsebujejo tudi kontrolo za različico operacijskega sistema, pod katero se aplikacija izvaja.

Funkcija *irInit()*:

```
public void irInit() {
	if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.KITKAT) {
		// implementacija za OS verzija >= 4.4 KitKat
		try {
			irdaService = (ConsumerIrManager)
context.getSystemService(CONSUMER_IR_SERVICE);
		} catch (NullPointerException e) {
			e.printStackTrace();
		}
	} else {// implementacija za OS verzija < 4.4 KitKat
		try {
			irdaService = getSystemService("irda");
			Class c = irdaService.getClass();
			Class p[] = { String.class };
			irWrite = c.getMethod("write_irsend", p);
		} catch (NoSuchMethodException e) {
			e.printStackTrace();
		} catch (NullPointerException e) {
			e.printStackTrace();
		}
	}
}
```

Funkcija *transmitIr()*:

```
public void transmitIr(int type) {
	if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.KITKAT) {
		// implementacija za OS verzija >= 4.4 KitKat
		int[] data = null;
		if (type == 0) {
data = hex2dec_KitKat(getResources().getString(R.string.ircode_canon_shoot));
```

```
		} else if (type == 1) {
			data =
hex2dec_KitKat(getResources().getString(R.string.ircode_canon_shootWithDelay));
		}
		if (data != null) {
			int frequency = data[0];
			int[] podatki = new int[data.length - 1];
			for (int i = 0; i < podatki.length; i++)
				podatki[i] = data[i + 1];
			((ConsumerIrManager) irdaService).transmit(frequency, data);
		}
	} else {// implementacija za OS verzija < 4.4 KitKat
		String data = null;
		if (type == 0) {
			data = hex2dec(getResources().getString(R.string.ircode_canon_shoot));
		} else if (type == 1) {
			data =
hex2dec(getResources().getString(R.string.ircode_canon_shootWithDelay));
		}
		if (data != null) {
			irWrite.invoke(irdaService, data);
		}
	}}
```

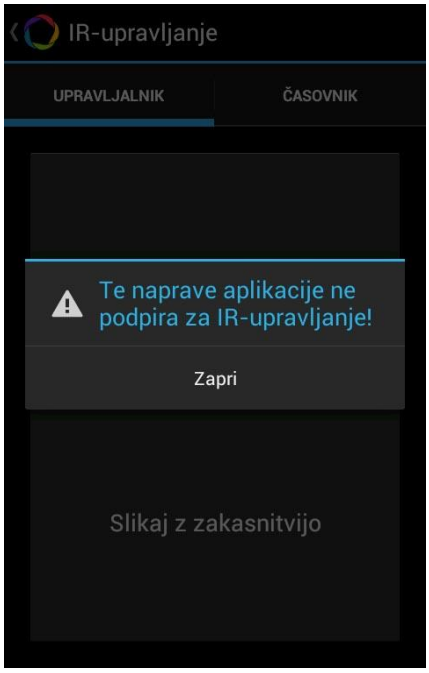
Zaradi novosti različice operacijskega sistema Android 4.4 KitKat v tem trenutku praktično še ni naprave, ki ima IR oddajnik in tudi že poganja najnovejšo različico sistema, zato je naša implementacija IR komunikacije izven emulatorja Android še nepreizkušena.

Pri razvoju smo se zavedali, da vse mobilne naprave ne bodo podprte z našo implementacijo IR upravljanja, zato smo zanje ob odprtju okna za omenjeni način upravljanja fotoaparata dodali obvestilo (slika 19). Z njim opozarjamo morebitne uporabnike, da njihove naprave mobilna aplikacija ne podpira in da ta funkcionalnost ne bo delovala. Funkcija, s katero to kontroliramo, je *checkIrdaSupport()*. Izdelali smo jo na način, da preverjamo podporo za IR komunikacijo tako v primeru različice 4.4 KitKat kot tudi v primeru uporabe starejše različice operacijskega sistema.

Funkcija *checkIrdaSupport()*:

```
private boolean checkIrdaSupport() {
	if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.KITKAT) {
		// implementacija za OS verzija >= 4.4 KitKat
		try {
			irdaService = (ConsumerIrManager)
context.getSystemService(CONSUMER_IR_SERVICE);
			if (((ConsumerIrManager) irdaService).hasIrEmitter()) {
				return true;
			}
		} catch (NullPointerException e) {
			return false;
		}
		return false;
	} else {// implementacija za OS verzija < 4.4 KitKat
		try {
			Object irdaService = this.getSystemService("irda");
			if (irdaService.getClass() != null) {
				return true;
			}
		} catch (NullPointerException e) {
```

```
                    return false;
                }
            }
            return false;
        }
```



Slika 19: Obvestilo o nepodprti mobilni napravi

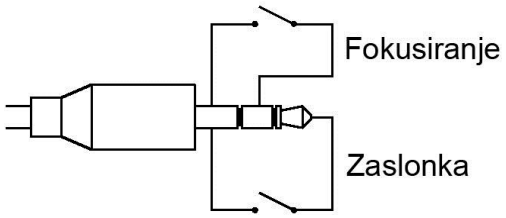
4.3. Izdelava kabelskega daljinskega upravljanja

4.3.1. Opis problema

V poglavju, kjer je predstavljen fotoaparat, smo že omenili, da je ena od možnosti daljinskega upravljanja fotoaparata upravljanje prek žične kabelske povezave. Fotoaparati znamke Canon EOS imajo vgrajena vrata za priključke (slika 6), kjer se nahaja priključek za priklop zunanjega kabelskega sprožilca. Omenjeni priključek se lahko od modela do modela razlikuje. Cenejši, bolj amaterski fotoaparati po navadi uporabljajo enostaven 2,5 mm stereo avdiopriključek TRS (angl. *Tip Ring Sleeve*) (slika 20), medtem ko dražji profesionalni modeli uporabljajo Canonov posebni priključek N3. Oba priključka po zasnovi vsebujeta 3 pole, razlikujeta pa se po obliki in dimenzijah. Ker je naš testni fotoaparat uporabljal 2,5 mm priključek TRS, smo v diplomskem delu obravnavali uporabo tega tipa priključka.

Slika 20: 2,5 mm priključek TRS s kablom [31]

Način delovanja daljinskega upravljalnika je za oba omenjena tipa priključkov identičen. Deluje po zelo enostavnem principu, da fotoaparat ves čas preverja stanje na priključku. V primeru, da je katera od dveh povezav električno sklenjena, izvede temu ustrezno operacijo. Na ta način lahko na primer na eni strani v fotoaparat priključimo kabel z dvema ali s tremi vodniki, na drugi strani pa kabel razvlečemo na oddaljenost tudi 10 m ali več. S stikalom lahko nato sklenemo ustrezno povezavo, ki jo bo fotoaparat zaznal, sprožil zaslonko in zajel posnetek. Pripomniti moramo, da pri tem načinu upravljanja lahko operiramo le z maksimalno tremi vodniki (od katerih je eden skupen). Zaradi tega smo omejeni tudi s številom funkcij, ki jih lahko na ta način upravljamo. Fotoaparat namreč prek tega tipa povezave podpira le dva tipa operacij, in sicer sprožitev zaslonke in izostritev posnetka. Shemo elektronske vezave za 2,5 mm priključek TRS in omenjeni operaciji vidimo na sliki 21.



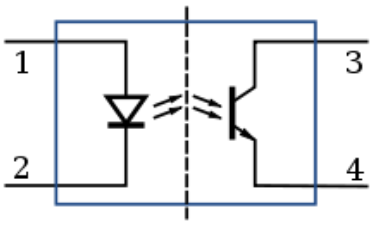
Slika 21: Shema vezave fotografskih operacij za 2,5 mm avdiopriključek TRS [32]

4.3.2. Opis rešitve

Cilj pri izdelavi mobilne aplikacije je bil implementirati zgoraj omenjeni princip delovanja. Največja težava, s katero smo se pri tem soočili, je bila, kako uporabiti izhod za slušalke na mobilni napravi in programskih vmesnikih Android API na način, da simuliramo delovanje stikala. Po analizi in raziskovanju delovanja avdioizhodov smo ugotovili, da ta omogoča predvajanje signala v analogni sinusni obliki, nismo pa našli rešitve, kako implementirati električno stikalo. Ob tem smo postali pozorni tudi na potencialni problem, da lahko z neposredno električno povezavo obeh naprav poškodujemo eno oziroma obe napravi. Tako smo morali opustiti prvotno zamisel in poiskati boljšo rešitev za opisani problem.

Rešitev, ki smo jo na koncu našli, je pomenila, da moramo pri izvedbi kabelskega upravljanja poleg mobilne naprave in fotoaparata uporabiti še eno vmesno elektronsko vezje. Predvideli smo uporabo elektronske komponente z imenom optospojnik [33], ki bi ga vključili v vezje med mobilno napravo in fotoaparat in bi v vezju prevzel funkcijo stikala.

Optospojnik je naprava, katere primarna funkcija je, da povezuje dve izolirani elektronski vezji. V našem primeru eno vezje predstavlja fotoaparat, drugo pa mobilna naprava. Kot nakazuje že ime, optospojnik deluje na način, da za prenos informacij iz enega vezja v drugo uporablja svetlobo in ne neposredne električne povezave. Elektronsko ponazoritev optospojnika lahko vidimo na sliki 22.

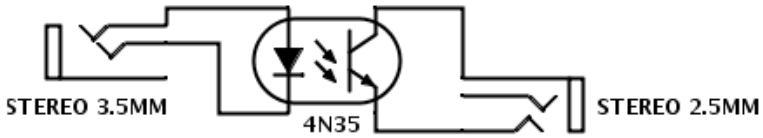


Slika 22: Shematski diagram optospojnika [33]

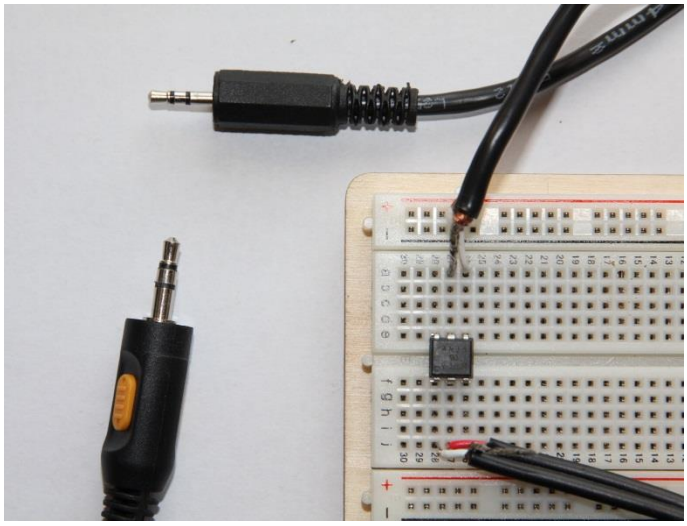
Ko na eni strani komponente med vhodnima poloma 1 in 2 priključimo dovolj veliko električno napetost, se znotraj komponente na eni strani vključi svetleča dioda (LED), kar na drugi strani zazna na svetlobo občutljiv tranzistor. V tem trenutku se tranzistor, ki v tem primeru deluje kot stikalo, vklopi in povezava med izhodnima poloma 3 in 4 je sklenjena.

Obe strani vezja sta v tem primeru tudi galvansko ločeni [34]. To pomeni, da med njima ni neposredne električne povezave, tako da ena naprava morebiti ne more poškodovati druge. Optospojnik na omenjeni način rešuje oba problema, ki smo ju imeli pred tem, zato smo razvoj nadaljevali z njegovo uporabo. Naslednji problem, ki smo ga morali rešiti, pa je bil, kako optospojnik krmiliti z uporabo avdiosignala.

Izdelavo kabelskega daljinskega upravljanja smo nadaljevali z izdelava testnega kabla in vezja z optospojnikom. Na sliki 23 vidimo električno shemo vezja na sliki 24 pa izdelano vezje, ki smo ga uporabili za testne namene pri izdelavi diplomskega dela.



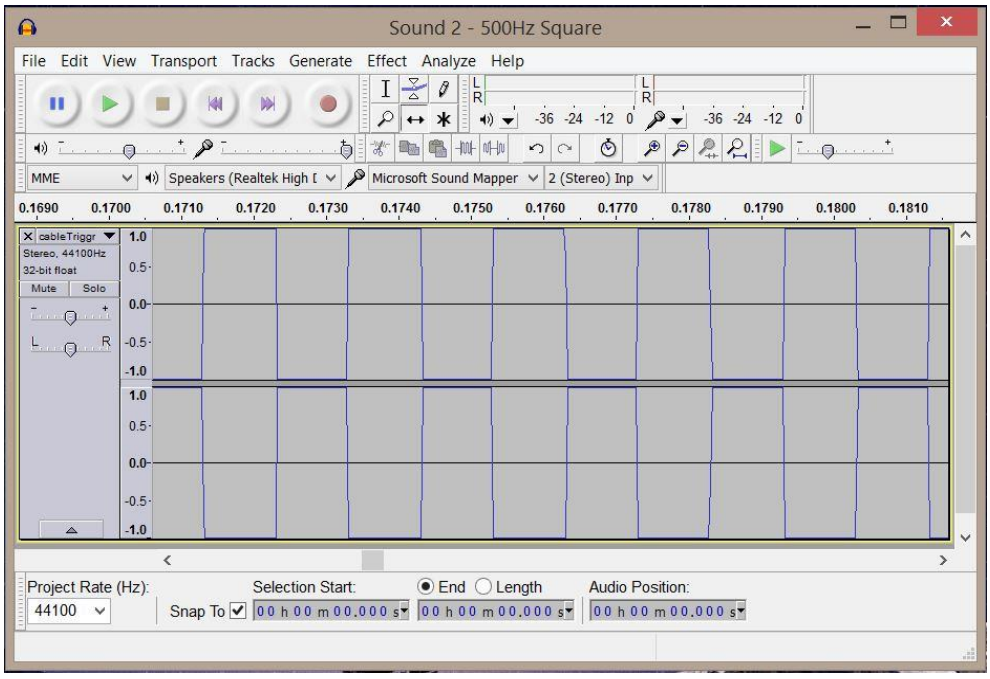
Slika 23 - Elektronska shema testnega vezja za kabelsko daljinsko upravljanje



Slika 24: Testno vezje in kabel za kabelsko daljinsko upravljanje

Sestavljeno je bilo iz 2,5 mm avdiopriključka TRS, daljšega dvokanalnega avdiokabla, vezja z optospojnikom 4N35 [35] ter kratkega avdiokabla, ki je bil zaključen s standardnim 3,5 mm stereo avdiopriključkom TRS. Optospojnik smo tudi ročno sprožili, da smo se prepričali, ali naše vezje in kabel dejansko delujeta. To smo storili na način, da smo na vhodno nogo priključili 3-voltno napetost in nato z digitalnim multimetrom preverjali stik na izhodnih nogah optospojnika. Ko smo imeli izdelan kabel, smo začeli z delom na proženju optospojnika z avdiosignalom. Postopek razvoja je bil takšen, da smo pred pričetkom razvoja funkcije generiranja avdiosignalov v mobilni aplikacije poizkusili generirati ustrezen signal na namiznem računalniku z uporabo programa Audacity [36]. To je brezplačen zmogljiv program za urejanje avdioposnetkov, ki med drugim omogoča tako vizualizacijo kot tudi

generiranje zvočnih signalov. Naša testna postavitev je vsebovala zgoraj omenjeni kabel, ki je bil na eni strani priključen na testni fotoaparat, na drugi pa na osebni računalnik. V programu Audacity smo nato začeli z generiranjem različnih oblik signalov. Po večjem številu preizkušenih signalov, različnih frekvenc, oblik in amplitud smo le našli tip signala, na katerega je optospojnik reagiral s sprožitvijo. Ugotovili smo, da je v našem primeru za optospojnik 4N35 najoptimalnejši zvočni signal kvadratne stereo oblike s frekvenco približno 200 Hz do 500 Hz (slika 25). Ugotovili smo tudi, da je frekvenco moč spreminjati v območju od 100 Hz do 1000 Hz in bo optospojnik vseeno v večjem odstotku časa deloval. Glede oblike signala smo ugotovili, da mora biti obvezno kvadratne oblike in da mora biti ob tem eden od stereo kanalov fazno zamaknjen za 180 stopinj. Optospojnik po specifikaciji namreč za delovanje potrebuje enosmerno napetost, avdiosignal pa je po naravi izmenična napetost. Električna napetost se v tem primeru skozi čas spreminja. Omenjena napetost sicer ni najprimernejša za krmiljenje optospojnika, vendar smo ugotovili, da je pod določenimi pogoji vseeno ustrezna. Rešitev tega problema je bila na koncu kombinacija pravilno strukturiranega avdiosignala in dovolj občutljivega optospojnika. S faznim zamikom enega od dveh kanalov stereo signala dosežemo, da je amplituda in posledično električna napetost med obema kanaloma v vsaki periodi maksimalna možna. Na sliki 25 vidimo, da se amplituda razteza od vrednosti +1 v enem kanalu do –1 v drugem.



Slika 25: Avdiosignal, generiran v programu Audacity

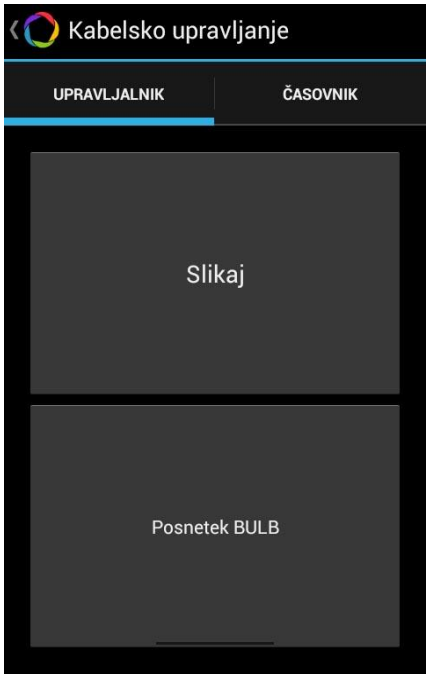
Če na optospojnik priključimo le vodnika s signalom enega in drugega kanala (skupno maso pustimo prosto), dosežemo, da na vsako drugo periodo optospojnik zazna napetost in sproži stikalo. Omenili smo vsako drugo periodo. Če znova pogledamo sliko 25, vidimo, da se iz ene periode v drugo amplituda obrne. Ko je na primer v prvi periodi električna napetost pozitivne

vrednosti, je v naslednji periodi napetost negativne vrednosti. Zaradi načina, kako delujejo diode LED (kot na primer tista znotraj optospojnika), je možno, da ta sveti le ob priključeni pozitivni napetosti. Z namenom, da čas trajanja maksimalne napetosti kar se da povečamo, je treba obliko signala iz običajne sinusne pretvoriti v kvadratno obliko. Tako dosežemo, da je maksimalna napetost znotraj posamezne periode vzdrževana čim dlje časa, s čimer jo ima optospojnik priložnost zaznati in se vključiti. Teoretično bi lahko frekvenco generiranega signala nastavili na primer na 1 Hz. V praksi smo ugotovili, da ta pristop ne deluje. Točnega vzroka za to nismo ugotovili, domnevamo pa, da je vzrok v nezmožnosti različnih naprav, da bi na zvočnem izhodu generirale ustrezen želeni signal tako nizkih frekvenc.

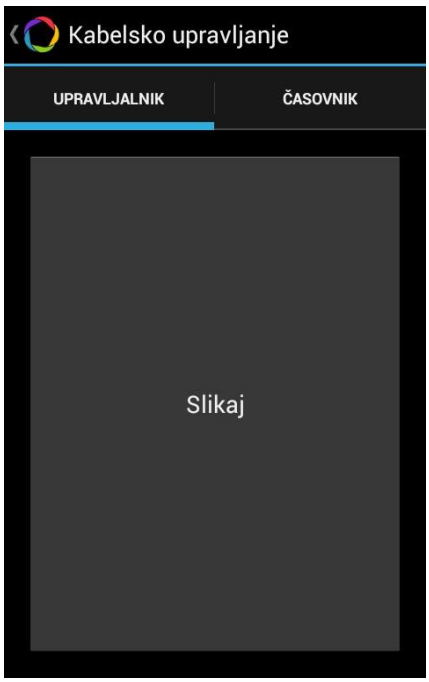
4.3.3. Izdelava rešitve v mobilni aplikaciji

Zatem, ko smo imeli izdelan delujoč praktični primer krmiljenja optospojnika z zvočnim signalom, je sledil naslednji korak – izdelava podobne funkcionalnosti v sklopu mobilne aplikacije.

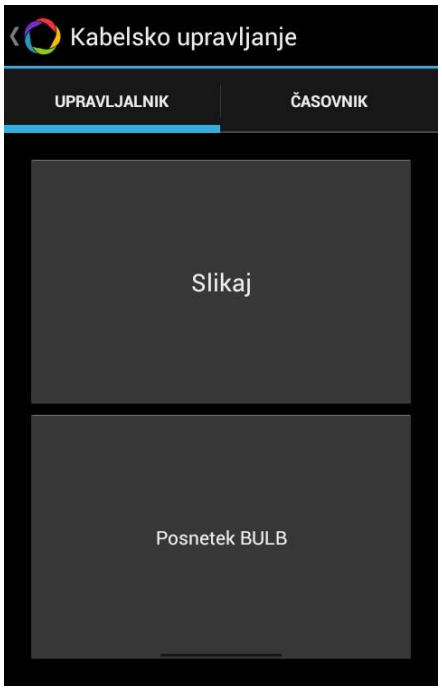
Pričeli smo z zastavitvijo in izdelavo uporabniškega vmesnika. Ker smo v sklopu tega načina upravljanja predvideli več možnosti krmiljenja, smo za vsakega od njih najprej izdelali prilagojen uporabniški vmesnik. Tako smo za predviden privzeti način delovanja, kjer za krmiljenje optospojnika uporabljamo generirani zvočni signal, izdelali vmesnik oziroma okno z dvema velikima gumboma. Za enega smo predvideli funkcionalnost *Slikaj*, na drugem pa funkcionalnost *Bulb posnetek* (slika 26). Za primerjavo smo za vmesnik v načinu delovanja s predposnetim signalom predvideli le en gumb, in sicer za operacijo *Slikaj* (slika 27). Vzrok za izvzeto bulb funkcionalnost v primeru slednjega je bila fiksna dolžina trajanja predposnetega signala, s katerim nam ni uspelo izdelati ustrezne bulb funkcionalnosti za razliko od poljubne dolžine trajanja pri generiranem signalu. Tretji način, ki smo ga izdelali, je bil predviden za uporabo v kombinaciji z mikrokontolerjem Arduino. V tem primeru smo uporabili identičen uporabniški vmesnik kot za primer generiranega zvočnega signala, in sicer dva velika gumba za operaciji *Slikaj* in *Bulb posnetek* (slika 28).



Slika 26: Uporabniški vmesnik za kabelsko upravljanje z uporabo generiranega zvočnega signala



Slika 27: Uporabniški vmesnik za kabelsko upravljanje z uporabo predposnetega zvočnega signala



Slika 28: Uporabniški vmesnik za kabelsko upravljanje z uporabo mikrokontrolerja Arduino

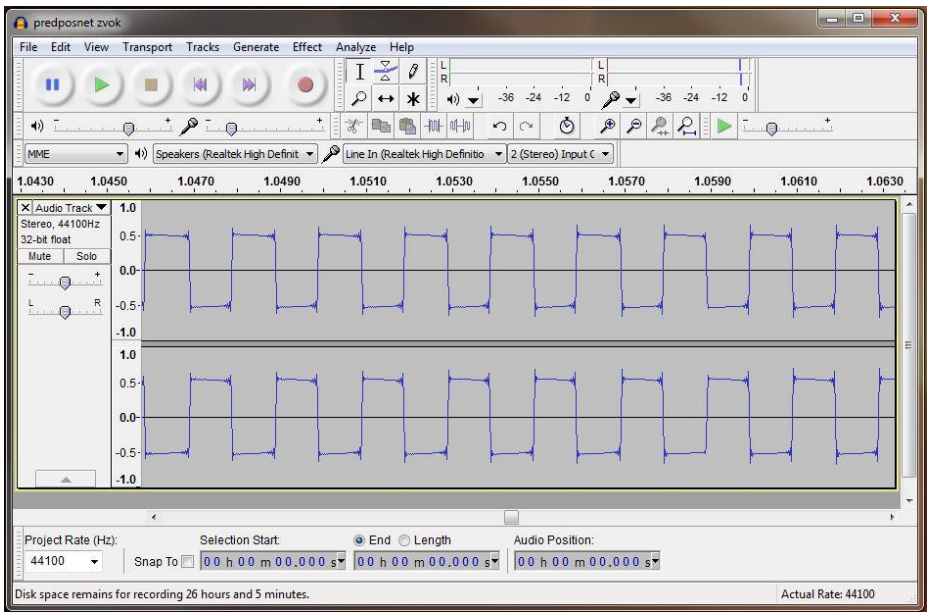
Z izdelanimi uporabniškimi vmesniki smo nadaljevali z izdelavo programske logike. Začeli smo s poskusom implementacije predvajanja že posnetega zvočnega signala, ki smo ga pred tem v razvojnem okolju Windows generirali s pomočjo programa Audacity. Najprej smo namreč želeli preveriti, ali je tudi mobilna naprava zmožna generirati ustrezen izhodni signal iz podanega posnetka. Naša ugotovitev je bila, da je na eni od naših testnih naprav (mobilniku Samsung Galaxy S2) predvajani zvočni posnetek optospojnik pravilno krmilil, medtem ko ga na drugih (mobilniku Samsung Galaxy S4 Mini in LG Nexus 5) ni. Po opravljeni analizi smo prišli do ugotovitve, da drugi mobilnik tudi pri polni glasnosti ne generira dovolj velike izhodne napetosti, da bi lahko z njo krmilili delovanje optospojnika 4N35. Domnevamo, da bi za kabelsko upravljanje lahko vseeno uporabili tudi naprave s takšnim šibkejšim zvočnim izhodom, vendar bi morali v vezju uporabiti optospojnik z drugačnimi karakteristikami. Potrebovali bi namreč takšnega, ki bi imel nižji prag zahtevane električne napetosti za krmiljenje.

Kot smo že omenili, smo različne načine krmiljenja kabelskega upravljanja ločili z nastavitvijo v nastavitvah aplikacije. Istočasno je lahko izbran samo eden od treh možnih načinov in glede na izbiro se prilagaja tudi uporabniški vmesnik ter delovanje aplikacije. V nadaljevanju bomo posamezno opisali delovanje vseh treh omenjenih načinov krmiljenja. Začeli bomo z načinom, kjer krmiljenje izvajamo s predposnetim glasbenim signalom. Omenili smo že, da smo za ta način izdelali uporabniški vmesnik s samo enim gumbom za operacijo *Slikaj*. Ob kliku omenjenega gumba se aplikacija odzove na način, da kliče funkcijo za predvajanje posnetka *playPrerecordedTone()*.

Funkcija *playPrerecordedTone()*:

```
public void playPrerecordedTone() {
	try {
		MediaPlayer mp = new MediaPlayer();
		if (mp.isPlaying()) {
			mp.stop();
			mp.release();
			mp = new MediaPlayer();
		}
		AssetFileDescriptor descriptor = getAssets().openFd(
				"sound/cableTriggrTone_500Hz_square.wav");
		mp.setDataSource(descriptor.getFileDescriptor(),
				descriptor.getStartOffset(), descriptor.getLength());
		descriptor.close();
		mp.prepare();
		mp.setVolume(1f, 1f);
		mp.setLooping(false);
		mp.start();
	} catch (Exception e) {
	}
}
```

Iz programske kode je razvidno, da se v funkciji kliče programski razred MediaPlayer [37], ki je sestavni del operacijskega sistema Android. Z njegovo uporabo kreiramo objekt *mp*, kateremu nato določimo datoteko wav s predposnetim signalom *cableTriggrTone_ 500Hz_square.wav*. Omenjeno datoteko smo pred tem izvozili iz programa Audacitiy in jo uvozili v mobilno aplikacijo. V nadaljevanju funkcije določimo še, da naj se glasnost obeh avdiokanalov nastavi na maksimalno vrednost in da naj se predvajanje izvede le enkrat brez ponovitev. Na koncu funkcije zaženemo še predvajalnik in naloženi posnetek se v tem trenutku prične predvajati. Predstavljena koda je precej enostavna in ob testiranju na mobilnih napravah smo na ta način uspešno krmilili fotoaparat. Na sliki 29 lahko vidimo zajeti izhodni signal predvajanega posnetka iz naprave Samsung Galaxy S2.



Slika 29: Posneti izhodni signal ob predvajanju predposnetega zvočnega signala

Slabost zgoraj opisanega načina upravljanja je predvsem v fiksni obliki podanega posnetka, s čimer mislimo tako na njegovo dolžino kot tudi obliko. Iz tega razloga smo se nato odločili izdelati še način krmiljenja z generiranim zvočnim signalom. Težava, na katero smo tukaj naleteli, je bila, kako zagotoviti generiranje ustrezne oblike avdiosignala.

Pri izdelavi generiranja avdiosignala smo se najprej obrnili na dokumentacijo operacijskega sistema Android. Pri tem smo ugotovili, da bi lahko za naše potrebe uporabili namenski programski razred za predvajanje generiranih avdiosignalov, razred AudioTrack [38]. Nato je bila naša naloga kreiranje ustreznega nabora podatkov, ki bi jih nato že omenjeni razred predvajal kot avdiosignal z želeno obliko. Na svetovnem spletu smo pri tem zasledili izvedbo odprtokodnega generatorja sinusnega signala [39]. Uporabili smo ga kot osnovo za izdelavo naše implementacije glede na naše funkcijske zahteve. Omenjeni primer generira sinusni signal z določeno frekvenco in dolžino. Naša zahteva pa je bila kreiranje signala kvadratne oblike, ki je na enem od obeh stereo kanalov fazno zamaknjen za 180 stopinj. Signal ima lahko tudi poljubno frekvenco in možnost neomejenega trajanja dolžine predvajanja.

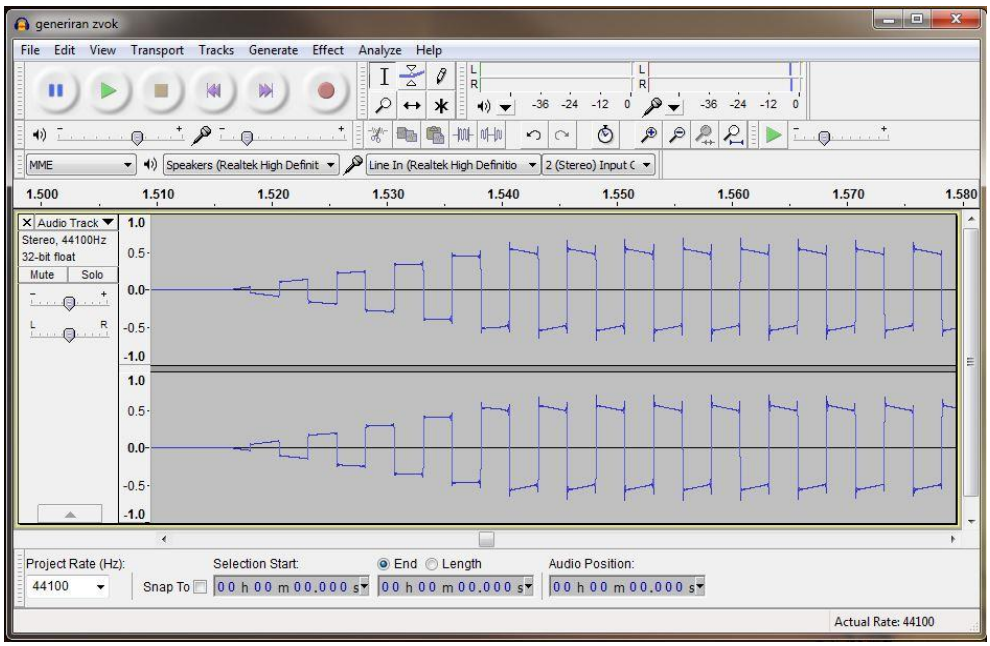
Funkcija *generateTone()*:

```
private AudioTrack generateTone(float frequency, int duration, boolean isLoop) {
	// tip: 1 – samo CH1; 2 – samo CH2; 0 - normalno
	int sampleRate = 44100;
	float leftVolume = 1.0f;
	float rightVolume = 1.0f;

	int count = (int) (sampleRate * 2.0 * (duration / 1000.0)) & ~1;
	short[] samples = new short[count];
	for (int i = 0; i < count; i += 2) {
		short sample = (short) (Math.signum(Math.sin(2 * Math.PI * i
				/ (sampleRate / frequency))) * 0x7FFF);
		samples[i + 0] = sample; // kanal 1
		samples[i + 1] = (short) -sample; // kanal 2; fazni zamik kanala za 180°
	}
	AudioTrack track = new AudioTrack(AudioManager.STREAM_MUSIC,
			sampleRate, AudioFormat.CHANNEL_OUT_STEREO,
			AudioFormat.ENCODING_PCM_16BIT, count * (Short.SIZE / 8),
			AudioTrack.MODE_STATIC);
	track.setStereoVolume(leftVolume, rightVolume);
	track.write(samples, 0, count);
	if (isLoop) {
		track.reloadStaticData();
		track.setLoopPoints(0, samples.length / 2, -1);
	}
	return track;
}
```

Zgoraj opisana koda Java predstavlja glavno funkcijo za generiranje avdiosignala za potrebe naše aplikacije. Deluje na način, da najprej v prvi polovici kode pripravi nabor podatkov v obliki matrike bajtov. Te podatke o signalu zatem v drugem delu kode prenesemo v kreirani programski objekt AudioTrack in ga pripravimo za predvajanje. Glavna operacija, ki se izvaja v omenjeni funkciji, je uporaba enačbe sinusne krivulje [40] za izračun podatkov o posamezni točki na sinusni krivulji. Rezultat tega izračuna se nato z uporabo funkcije Java *Math.signum* [41] pretvori v podatek o kvadratni obliki signala. Signum funkcija enostavno pretvori vse

vrednosti, ki so večje od 0, v vrednost 1, ter vse vrednosti, ki so manjše od 0, v vrednost –1. Tako smo kreirali generiranje signala kvadratne oblike neposredno iz sinusne krivulje. Naslednji korak je bil izdelava faznega zamika na enem od stereo kanalov. Po proučevanju dokumentacije smo ugotovili, da so podatki v stereo avdioformatu strukturirani tako, da imajo izmenični zapisi v matriki bajtov, ki jo generiramo, vrednost posameznega kanala. Povedano drugače, vsak zapis na lihi poziciji v matriki predstavlja vrednost enega kanala, vsak sodi zapis v matriki pa predstavlja podatke za drugi kanal. Če vzamemo za primer niz bajtov {11, 44, 25, 45, 38, 31, 50, 26}, potem levi kanal predstavljajo vrednosti 11, 25, 38, 50, medtem ko so podatki za desni kanal 44, 45, 31, 26. V funkciji *generateTone()* smo fazni zamik izvedli na način, da vsak drugi zapis v matriki bajtov pomnožimo z –1. Na ta način dobimo vrednosti, ki so inverzne vrednostim drugega kanala, kar pa pomeni, da sta sedaj kanala v obratnih orientacijah. Tako smo prišli do signala, ki zadosti vsem našim zahtevam.

Z uporabo zgoraj predstavljene funkcije lahko generiramo signal s poljubno frekvenco in dolžino. Na sliki 30 lahko vidimo zajeti izhodni signal mobilne naprave ob generiranju zvočnega signala.



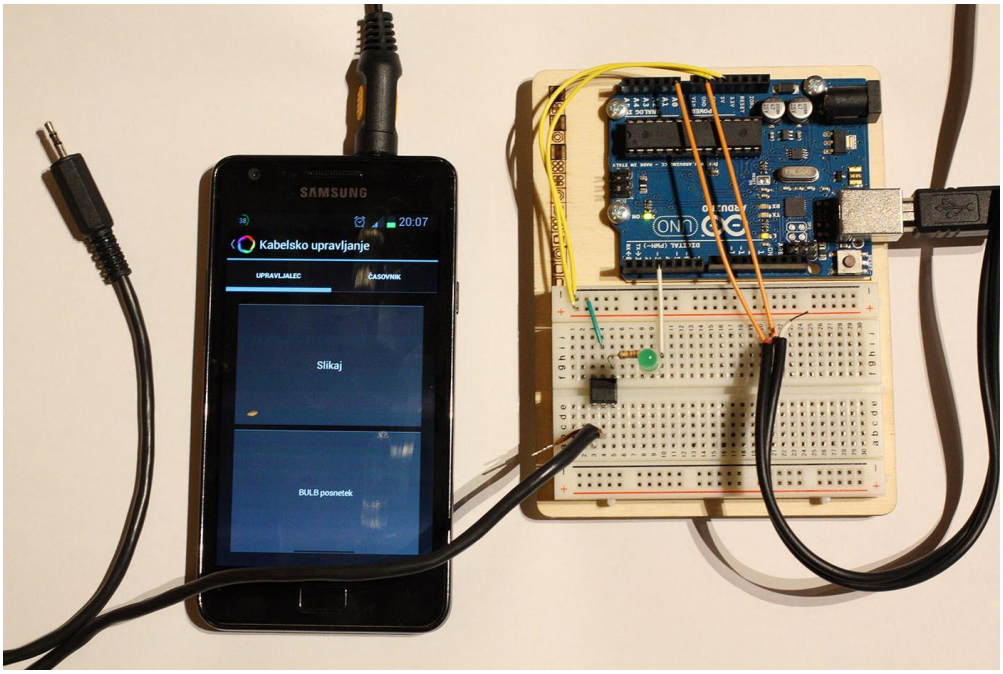
Slika 30: Posneti izhodni signal ob predvajanju generiranega zvočnega signala

Prednost generiranega signala je njegova parametrizacija. Nastavljiva je tako frekvenca kot trajanje posnetka. Slednje je lahko teoretično tudi brez omejitve, kar smo izrabili pri izvedbi funkcionalnosti bulb slikanja. V tem primeru smo na uporabniškem vmesniku poleg gumba *Slikaj* kreirali še preklopni gumb *Bulb posnetek*. Vzrok, zakaj smo uporabili preklopni gumb, je v tem, da na ta način laže definiramo, kdaj se bulb slikanje prične in zaključi. Preklopni gumb ima namreč dve možni stanji, vključen ali izključen. Na ta dva dogodka je vezana dolžina predvajanja krmilnega signala. Signal se namreč generira in predvaja ves čas, ko je gumb aktiviran, prekine pa se ob izklopu gumba.

4.3.4. Zagotavljanje kompatibilnosti z večjim številom mobilnih naprav

V prejšnjem odstavku smo opisali in izdelali okvirno rešitev za želeno funkcionalnost kabelskega upravljanja fotoaparata vendar je ta imela slabost oziroma pomanjkljivost. Pravilno delovanje ni bilo vedno zagotovljeno z različnimi mobilnimi napravami. Iz tega razloga smo v nadaljevanju izdelave diplomske naloge poskušali poiskati še dodatne alternativne oblike implementacije naše oblike upravljanja.

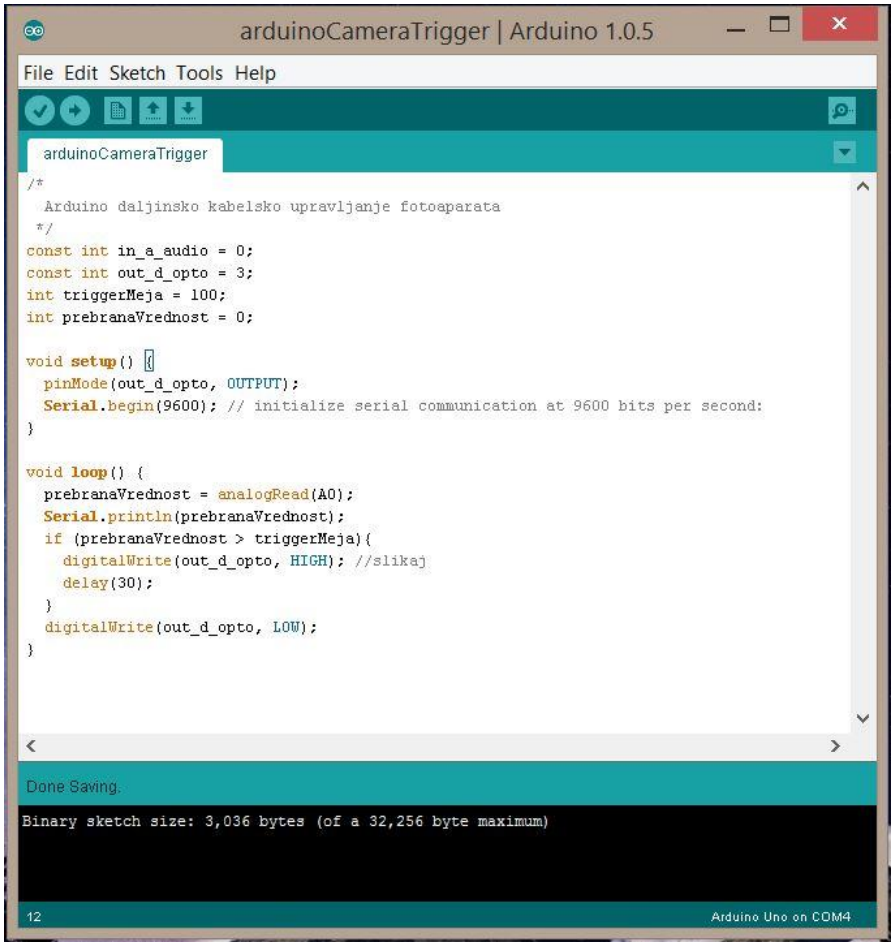
Po premisleku smo prišli do ideje, da bi bilo mogoče izdelati krmiljenje fotoaparata tudi s pomočjo vezja Arduino. Z njegovo uporabo bi sicer povečali kompleksnost uporabljenega krmilnega vezja, dosegli bi pa večjo združljivost z večjim številom mobilnih naprav. Zamisel je bila, da bi pri tem načinu upravljanja v mobilni napravi prek izhoda za slušalke še vedno generirali ustrezen signal. Razlika bi bila, da bi signal neposredno kontroliral mikrokontroler, ga ustrezno interpretiral in nato izvedel krmiljenje optospojnika. Implementacija je bila zasnovana na osnovi že izdelanega krmiljenja z generiranim zvočnim signalom. Za razliko od testne postavitve za prejšnja dva opisana načina krmiljenja (slika 24) smo v tem primeru v vezje med optospojnik in mobilno napravo vključili še mikrokontroler. Njegova naloga je, da kontrolira vhodni avdiosignal, in ko ga zazna, ustrezno proži optospojnik. V mobilni aplikaciji nam za ta način krmiljenja ni bilo treba naredili veliko sprememb. Kot smo že omenili, je uporabniški vmesnik v tem primeru isti kot v načinu z generiranim zvočnim signalom. Imamo dva velika gumba za operaciji *Slikaj* in *Bulb posnetek*. Tudi za generiranje zvočnega signala smo uporabili isti mehanizem. Edina sprememba, ki smo jo pri tem naredili, je bila, da smo avdiosignal fiksirali na frekvenco 600 Hz.



Slika 31: Testno vezje za kabelsko daljinsko upravljanje z mikrokontrolerjem Arduino

Programsko kodo, ki smo jo izdelali za krmiljenje mikrokontrolerja Arduino, lahko vidimo navedeno spodaj oziroma na sliki 32, kjer je prikazana v okviru programa Arduino Software IDE.

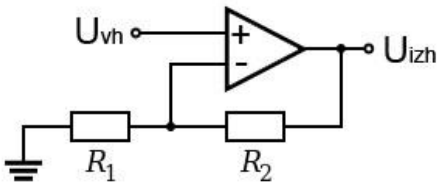
```
/* Arduino daljinsko kabelsko upravljanje fotoaparata */
const int in_a_audio = 0;
const int out_d_opto = 3;
int triggerMeja = 100;
int prebranaVrednost = 0;

void setup() {
  pinMode(out_d_opto, OUTPUT);
  Serial.begin(9600); // initialize serial communication at 9600 bits per second:
}

void loop() {
  prebranaVrednost = analogRead(A0);
  Serial.println(prebranaVrednost);
  if (prebranaVrednost > triggerMeja){
    digitalWrite(out_d_opto, HIGH); //slikaj
    delay(30);
  }
  digitalWrite(out_d_opto, LOW);
}
```



Slika 32: Program Arduino za kabelsko upravljanje fotoaparata v razvojnem orodju Arduino Software IDE

Program deluje na način, da ob zagonu v neskončni zanki ves čas s funkcijo *analogRead()* preverja vrednost na analognem vhodu številka 0. Ta vhod je prek kabla povezan na mobilno napravo, ki oddaja zvočni signal. Ko mikrokontroler na vhodu zazna vrednost, večjo od mejne vrednosti, se odzove na način, da digitalni izhod številka 3 nastavi na visoko vrednost oziroma ga vključi. Na ta vhod sta nato prek zaščitnega upora priključena optospojnik in dioda LED. Optospojnik se ob visoki vrednosti na digitalnem izhodu vklopi in sklene izhodno stran, ki je prek kabla povezana s fotoaparatom. Dioda LED sicer za delovanje vezja ni potrebna, je pa zaželena zaradi lažje vizualizacije tega, kdaj je optospojnik aktiviran. Omeniti želimo tudi, da je mejna vrednost konstanta. Njeno vrednost smo definirali skozi testiranje. V našem primeru znaša 100. V nadaljevanju programa lahko vidimo še, da potem ko se izhod številka 3 aktivira, počakamo 30 milisekund in ga nato znova nastavimo na nizko vrednost oziroma ga izklopimo. Zatem se zanka ponovi. Za pravilno delovanje krmiljenja je pomembno tudi, da je vhodni priključek 0 ves čas vezan na referenčno maso. Na ta način v primeru, ko na vhodni strani ni priključene naprave, mikrokontroler prebere pravilno vrednost 0. V nasprotnem primeru bi prebral naključno vrednost, ki bi bila po vsej verjetnosti večja od mejne vrednosti, kar bi napačno sprožilo optospojnik.

Drugi dodatni način implementacije krmiljenja fotoaparata z uporabo kabelske povezave, ki smo si ga zamislili, pa je predvideval uporabo elektronske komponente imenovane operacijski ojačevalnik [42]. Z uporabo slednje smo nameravali izdelati vezje za ojačanje avdio signal mobilne naprave na raven, ki bi omogočala zanesljivo krmiljenje vezja optospojnika iz slike 23. Na sliki 33 lahko vidimo elektronsko shemo vezja (neinvertirajočega) ojačevalnika, ki smo ga uporabili v ta namen. Predstavljeno vezje deluje na način, da operacijski ojačevalnik ojača vhodno napetost za faktor, ki je definiran z razmerjem vrednosti uporov R1 in R2. Odvisnost ojačanja izhodne napetosti od razmerja vrednosti uporov opisujeta enačbi 1 in 2.



Slika 33 – Elektronska shema neinvertirajočega ojačevalnika [42]

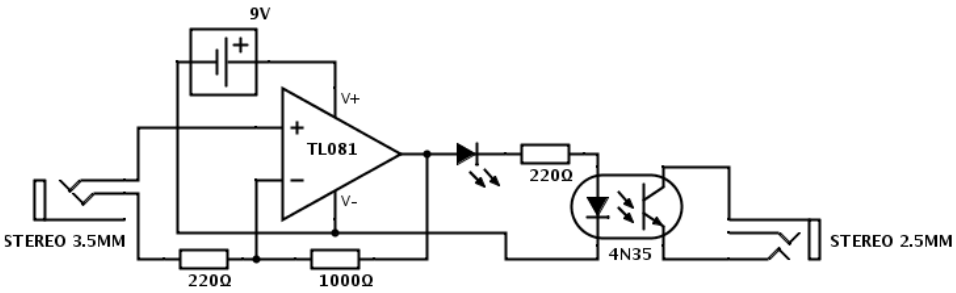
$$U_{izh} = U_{vh} * \left(1 + \frac{R2}{R1}\right) (1)$$

$$A = \left(1 + \frac{R2}{R1}\right) (2)$$

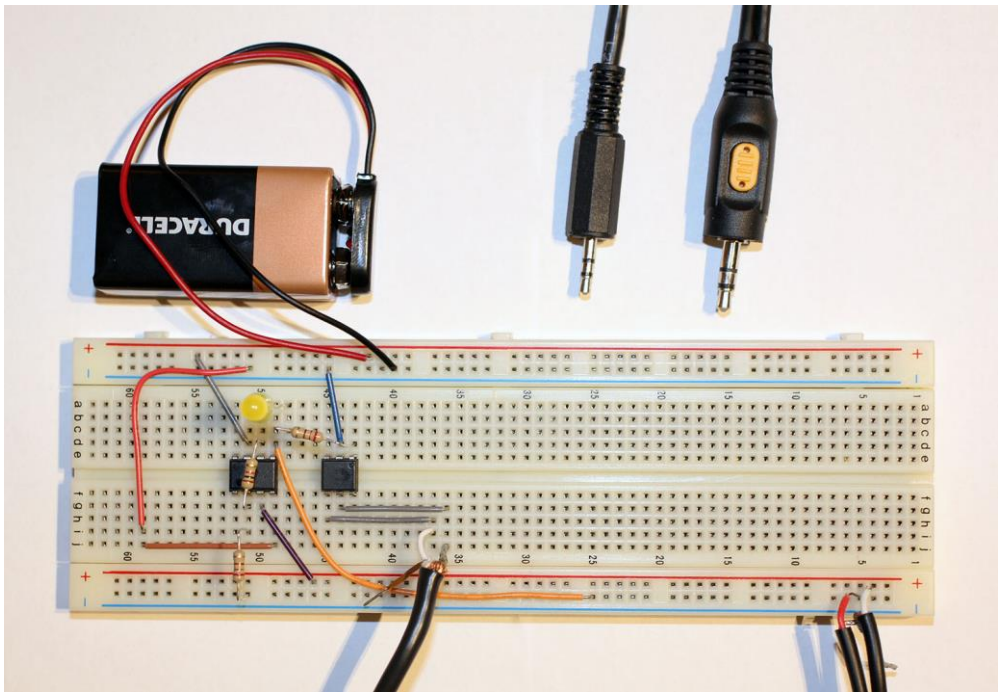
Kot smo že omenili je bil namen izdelave ojačevalnega vezja, nadgradnja obstoječega vezja z optospojnikom (slika 23). Pri izdelavi neinvertirajočega ojačevalnika smo uporabili model operacijski ojačevalnik TL081 [43] v 8-pinskem plastičnem DIP (*dual in-line package*)

ohišju. Glede na opravljene meritve in analize generiranega avdio signala mobilne naprave, ki smo jih pred tem v sklopu izdelave kabelskega upravljanja že opravili, smo spoznali, da bi bilo potrebno v novem vezju avdio signal ojačati za večkratni faktor. Po razmisleku smo se odločili, da bo ta faktor ojačanja znašal 5. Tako bi na izhodu ojačevalnika namreč dobili napetost, ki bi zadostila specifikacijam za krmiljenja optospojnika 4N35. Omenjeno ojačanje smo dosegli na način, da smo ob upoštevanju enačbe 2 ustrezno izbrali vrednosti za upora R1 in R2. Za upor R1 smo tako izbrali vrednost 220 ohmov za upor R2 pa vrednost 1000 ohmov.

Shemo krmilnega vezja, ki smo ga izdelali lahko vidimo na sliki 34, na sliki 35 pa lahko vidimo izdelano testno vezje na prototipni ploščici. Ker je operacijski ojačevalnik aktivna elektronska komponenta smo potrebovali v vezje vključiti tudi zunanje napajanje v obliki 9-voltne električne baterije.



Slika 34 - Elektronska shema testnega vezja za kabelsko daljinsko upravljanje z uporabo operacijskega ojačevalnika



Slika 35 - Testno vezje za kabelsko daljinsko upravljanje z uporabo operacijskega ojačevalnika

Način kako deluje omenjeno vezje ojačevalnika je, da na eni strani preko 3,5 mm TRS priključka nanj priključimo mobilno napravo. Ko v mobilni aplikaciji izvedemo operacijo slikanja se na izhodu za slušalke generira izhodni signal v obliki električne napetosti. Slednjo operacijski ojačevalnik v vezju ojača in kot taka se le-ta nato uporabi za krmiljenje optospojnika. Kot smo v diplomskem delu že omenili se slednji ob dovolj veliki vhodni napetosti vključi in sklene električni tokokrog na svoji izhodni strani, ki je preko vodnika priključena na fotoaparat.

Pripomniti želimo še, da smo za namene testiranja in razvoja mobilne aplikacije testna vezja sestavili na prototipni ploščici (slike 24, 31 in 35) in da se v primeru drugega opisanega vezja to napaja neposredno iz USB-vodila. V primeru, da bi se vezja uporabljala v vsakdanje namene (v naravi), bi bilo treba opisana vezja izdelati v obliki tiskanega vezja. Njegove dimenzije bi bile glede na število uporabljenih komponent relativno majhne. V primeru vezja Arduino bi lahko v končni različici vezja namesto mikrokontrolerja Arduino uporabili tudi Arduino Mini oziroma Arduino Nano, ki sta po dimenzijah precej manjša. V tem primeru bi ga tudi baterijsko napajali. Na ta način bi dosegli večjo kompaktnost in zanesljivost vezja.

4.4. Izdelava funkcije intervalometra

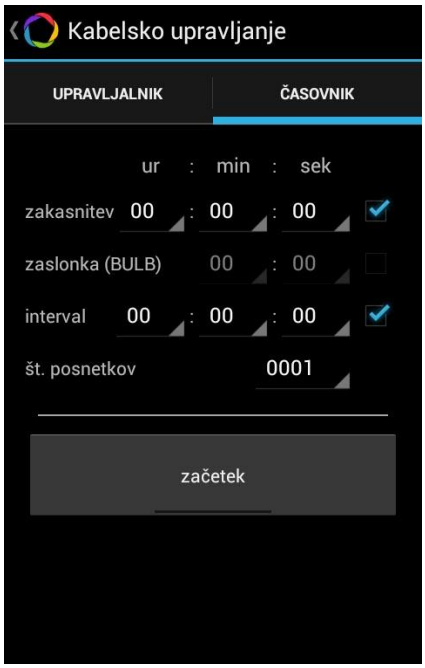
Potem ko smo uspeli izdelati funkcije daljinskega upravljanja z uporabo IR in kabelske komunikacije, smo dobili zamisel, da bi mogoče lahko ti dve funkciji razširili in implementirali intervalometer oziroma časovnik. Intervalometer [44] je naprava, ki šteje časovne intervale in na vsako iteracijo izvede preddoločeno operacijo. Funkcionalnost takšne naprave s fotografskega vidika bi bila, da uporabniku omogoča nastavitev nekaterih časovnih parametrov in število želenih ponovitev. Naprava bi ob zagonu ob upoštevanju podanih časovnih parametrov te odštevala in na vsako novo ponovitev na fotoaparatu sprožila zajem posnetka. Časovni parametri, ki jih je moč nastaviti, so na primer uvodna zakasnitev pred pričetkom odštevanja števila ponovitev ali pa čas med ponovitvama. V fotografstvu so intervalometri v glavnem uporabljeni za izdelavo tako imenovanih videoposnetkov Time-Lapse [45] ali pa za zajem posnetkov s časovno zakasnitvijo.

Funkcijo delovanja kot intervalometer imajo dandanes nekateri dražji modeli fotoaparatov že vgrajeno, drugi fotoaparati pa lahko za to izrabijo zunanje vire upravljanja. Na svetovnem spletu lahko z uporabo orodja Google Iskanje zelo hitro najdemo veliko število tako komercialnih izdelkov kot tudi doma narejenih izvedb intervalometrov. Na sliki 36 lahko vidimo tipičen primerek naprave, ki deluje kot intervalometer in se uporablja v fotografstvu. Glavni elementi takšne naprave so na primer majhen zaslon, na katerem je mogoče nastavljati parametre, gumb za sprožitev delovanja in kabel za povezavo naprave s fotoaparatom.

Slika 36: Intervalometer Canon TC-80N3 [4]

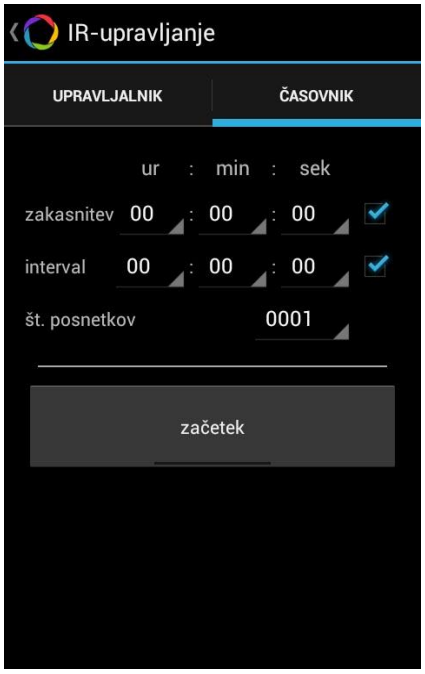
Zamisel, ki se nam je tako porodila, je bila, da bi to funkcionalnost vgradili v mobilno aplikacijo, ki smo jo že izdelali. Glede na to, da smo že uspeli implementirati zajem posnetkov z uporabo tako IR komunikacije kot tudi prek kabelske povezave, smo ugotovili, da bi bilo treba izdelati samo še dodatni uporabniški vmesnik in nato funkcionalnost časovnika. Uporabniški vmesnik bi nudil možnost vnosa časovnih parametrov in števila želenih ponovitev. Celotno implementacijo smo izdelali dvakrat, po enkrat za obe že obravnavani obliki daljinskega upravljanja.

V primeru časovnika na kabelskem upravljanju (slika 37) smo izdelali tudi zajem posnetka v bulb načinu. V tem primeru smo za časovnik dodali še en dodaten parameter, ki smo ga poimenovali *zaslonka (BULB)*. Način, na katerega smo implementirali delovanje časovnika, je bil, da za vsak interval oziroma periodo predvajamo generiran zvočni posnetek za količino časa, definiranega z omenjenim dodatnim parametrom. Na ta način je na fotoaparatu zaslonka odprta toliko časa, kolikor dolgo se predvaja zvočni posnetek.



Slika 37: uporabniški vmesnik časovnika za kabelsko upravljanje

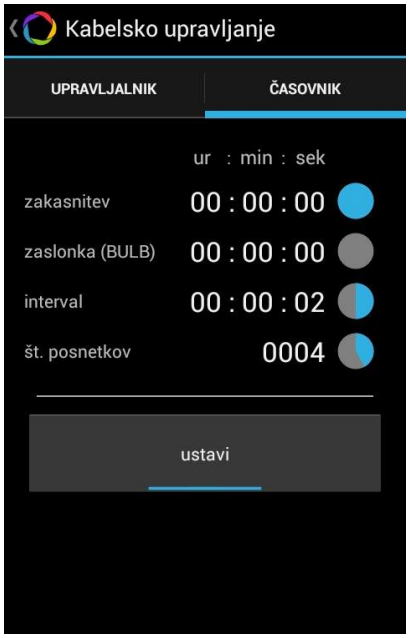
Videz uporabniškega vmesnika za implementacijo časovnika pri IR upravljanju je prikazan na sliki 38. Iz omenjene slike je razvidno, da za ta način upravljanja nismo dodali opcije za določanje časa zaslonke bulb, čeprav bulb slikanje fotoaparat podpira za ta način upravljanja. Vzrok, zakaj funkcije nismo implementirali, leži v morebitnih možnih napakah, ki lahko nastanejo v komunikaciji med IR oddajnikom in fotoaparatom. Upravljanje bulb funkcije namreč zahteva tako IR signal, ki aktivira slikanje, kot tudi signal, ki slikanje zaključi. Morebitna težava zna nastati, ko se eden od teh dveh signalov izgubi ali pa ga fotoaparat ne zazna pravilno. V tem primeru se na primer slikanje ne zaključi ob predvidenem času, temveč šele z morebitnim naslednjim sprejetim IR signalom. V primeru časovnika zna to povzročiti, da se poruši vse predvideno zaporedje nadaljnjih ukazov, zaradi česar ta izgubi svojo uporabnost. Za rešitev problema bi morali iz fotoaparata dobiti povratno informacijo, ali je bil poslani signal tudi prejet. Zaradi narave komunikacije tega v primeru IR upravljanja ni mogoče doseči.



Slika 38: Uporabniški vmesnik časovnika za IR-upravljanje

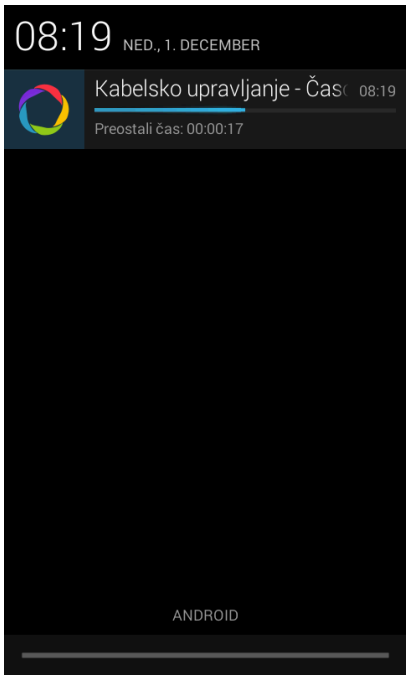
Na tem mestu bomo na kratko opisali delovanje in uporabniški vmesnik časovnika. Kot je razvidno iz slik (slika 37 in slika 38), uporabniški vmesnik obsega določanje različnih časov v razponu želenih ur, minut in sekund, možnost nastavitve števila želenih posnetkov, ki jih želimo zajeti, ter gumb za začetek delovanja. Poleg vnosa posameznih časovnih parametrov imamo tudi možnost izbrati oziroma obkljukati, katere od parametrov želimo, da jih časovnik uporabi pri delovanju. Delovanje časovnika je precej enostavno. Ob kliku na gumb *Začetek* se časovnik aktivira. Preračunajo se časi posameznih operacij in število ponovitev, potek izvajanja pa je prikazan v uporabniškem vmesniku (slika 39). V primeru, da je nastavljen čas zakasnitve, se v prvem koraku izvajanja ta čas odšteva. Ko omenjeni čas preteče, se prvič izvede funkcija, ki komunicira s fotoaparatom. V tem trenutku se tudi prvič zmanjša parameter števila posnetkov za vrednost 1. V naslednjem koraku izvajanja se prične odštevanje podanega časovnega intervala med posnetki. Ko ta čas preteče, se znova izvede funkcija za komunikacijo in spet se zmanjša parameter števila posnetkov. Celoten postopek se nato ponovi še tolikokrat, kolikor je bilo definirano število želenih posnetkov. Pri definiranem parametru bulb zaslonke se ta upošteva v funkciji, ki izvaja komunikacijo s fotoaparatom. Ob zaključku oziroma ročni ustavitvi izvajanja se časovnik ponastavi na vrednosti parametrov pred pričetkom izvajanja.

Razlika v programski implementaciji med slikanjem na gumb in časovnikom je v tem, da pri prvem izvajamo kodo za krmiljenje fotoaparata samo enkrat, in sicer ob interakciji uporabnika z gumbom. V primeru časovnika pa se kodo za sprožitev slikanja izvaja glede na intervale časovnika in števila njegovih ponovitev. V primeru IR načina upravljanja fotoaparata se v trenutku klicanja funkcije za komunikacijo generira in odda IR signal. V primeru kabelskega upravljanja pa se v tem trenutku generira avdiosignal, ki se predvaja na izhodu za slušalke.



Slika 39: Prikaz časovnika v izvajanju

Kot opcijsko funkcionalnost v delovanju časovnika smo dodali tudi prikaz obvestila o poteku časovnika oziroma o preostalem času izvajanja. Slednje je prikazano v obvestilni vrstici operacijskega sistema Android (slika 40). Na ta način je uporabniku vedno na dosegu informacija o poteku izvajanja časovnika, tudi če v vmesnem času odpre katero drugo aplikacijo. Ob kliku na prikazano obvestilo se znova odpre aplikacija. Prikaz omenjenega obvestila je možno nastaviti v nastavitvah aplikacije.



Slika 40: Prikaz obvestila o poteku časovnika

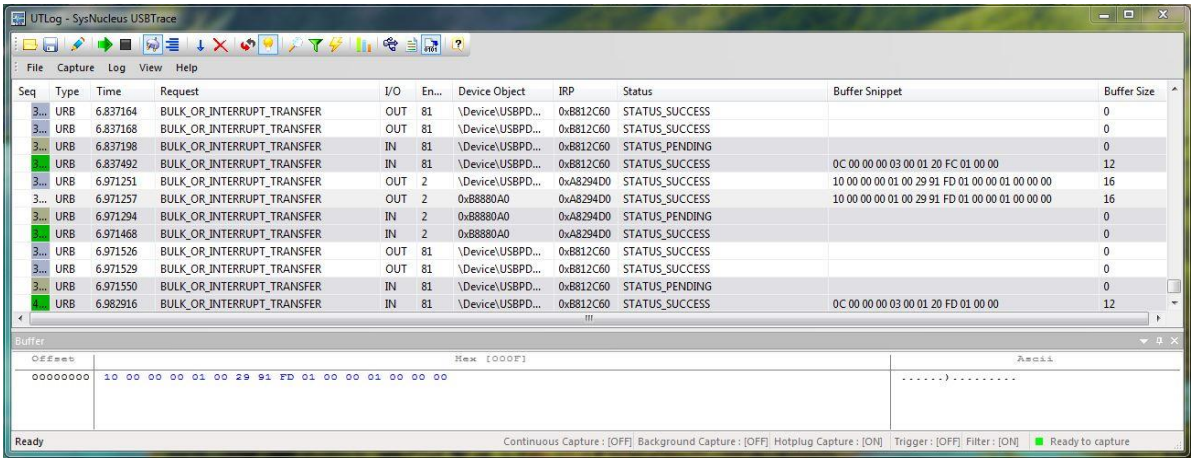
4.5. Izdelava USB-daljinskega upravljanja

Z izdelanim daljinskim upravljanjem fotoaparata z uporabo IR komunikacije in kabelske povezave smo se lotili še izdelave upravljanja fotoaparata prek USB-povezave. Naše mnenje je, da je ta način upravljanja od vseh treh obravnavanih najbolj kompleksen in najtežji za implementacijo. Največja težava, ki smo jo imeli pri izdelavi, je bila, kako pridobiti pravilne podatke in ukaze za komuniciranje s fotoaparatom. Nikjer namreč nismo našli nobene javno dostopne dokumentacije, iz katere bi lahko to razbrali. Zato smo te podatke poskušali pridobiti z analizo obstoječe aplikacije za osebne računalnike, ki omogoča upravljanje s fotoaparatom preko USB-povezave.

4.5.1. Proučitev namizne aplikacije EOS Utility

Podjetje Canon za upravljanje svojih fotoaparatov uporabnikom nudi brezplačno aplikacijo za operacijski sistem Windows EOS Utility (slika 4), ki med drugim omogoča upravljanje fotoaparata tudi prek USB-povezave.

V sklopu izdelave diplomskega dela smo omenjeno aplikacijo uporabili na način, da smo na računalnik priključili fotoaparat in nato izbrali možnost njegovega daljinskega upravljanja. Poleg aplikacije smo na računalniku imeli nameščen tudi program za beleženje zgodovine prenešenih podatkov USB-komunikacije. S njim smo nato zajeli vse podatke, ki so se pošiljali med računalnikom in fotoaparatom v trenutku, ko smo v aplikaciji EOS Utility pritisnili na gumb *Slikaj*. Podatke, ki smo jih tako pridobili, smo nato ročno analizirali in iz njih poskušali razbrati potrebne ukaze za proženje operacije *Slikaj* na fotoaparatu.



Slika 41: Prikaz zajetih podatkov USB-komunikacije pri upravljanju fotoaparata z aplikacijo EOS Utility

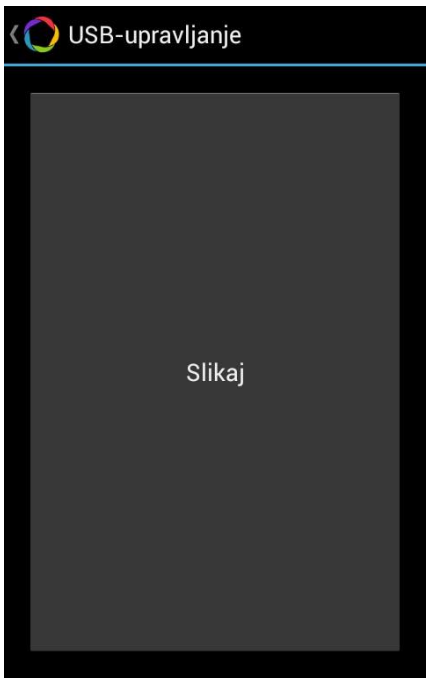
Po daljšem analiziranju zajetih podatkov smo ugotovili, kako so podatki strukturirani. Opazili smo namreč določene vzorce v podatkih. Zapisani so v šestnajstiški obliki, sestavljeni pa iz

niza besed, pri čemer ima vsaka svoj pomen. Opazili smo, da prve štiri besede navajajo dolžino posameznega ukaza v številu šestnajstiških besed, ki jih ta vsebuje. Nato sledita dve besedi, ki se navidezno nikoli ne spremenita in imata vrednost 01 00. Naslednji dve besedi sta vedno odvisni od operacije, ki smo jo kliknili, zato smo domnevali, da predstavljata dejanski ukaz fotoaparatu. Vsemu temu pa sledu še niz štirih besed, ki najverjetneje predstavljajo identifikator seje, saj se z vsakim ukazom povečuje za vrednost 1.

Ker iz pridobljenih zajetih podatkov nismo pridobili vseh želenih podatkov o komunikaciji, smo jih poskušali poiskati še kje drugje. Na svetovnem spletu smo našli podoben projekt [46], kot ga obravnava naša diplomska naloga, kjer so ustvarjalci izdelali upravljanje fotoaparatov Canon prek USB-povezave za mikrokontroler Arduino. Z analizo njihove dokumentacije smo pridobili še dodatne informaciji o strukturi podatkov ter o ukazih za posamezne operacije upravljanja. Sedaj, ko smo približno razumeli delovanje komunikacije med aplikacijo in fotoaparatom, smo začeli z izdelavo podobne implementacije v naši mobilni aplikaciji.

4.5.2. Izdelava rešitve v mobilni aplikaciji

Podobno kot pri ostalih načinih daljinskega upravljanja, ki smo jih že predstavili v sklopu diplomskega dela, smo tudi tu razvoj mobilne aplikacije začeli z izdelavo uporabniškega vmesnika. Ker smo nameravali za ta načina upravljanja izdelati le operacijo *Slikaj*, smo temu primerno zastavili tudi videz uporabniškega vmesnika (slika 42). Vseboval je le en velik gumb, ki zaseda celotno površino zaslona in ima naziv Slikaj.



Slika 42: Uporabniški vmesnik za USB-upravljanje

Z izdelanim uporabniškim vmesnikom pa smo napore vložili v izdelavo programske logike za vzpostavitev USB-povezave in komunikacije med aplikacijo in fotoaparatom. V ta namen smo začeli prebirati objavljeno dokumentacijo vmesnikov Android API za USB-komunikacijo [47], pregledali pa smo tudi dokumentaciji priložene primere aplikacij [48]. Kot smo že omenili, smo v našem primeru obravnavali le USB-komunikacijo v načinu gostovanja (angl. USB Host). Zanimalo nas je namreč, na kakšen način se v mobilni aplikaciji vzpostavi ta oblika USB-komunikacije. Ugotovili smo, da je implementacija vzpostavitve komunikacije relativno enostavna, saj je možno v večji meri v ta namen uporabiti kar primere iz dokumentacije operacijskega sistema Android. Implementacija vseeno zahteva večje razumevanje kode, kakor jo je moč v sklopu tega diplomskega dela predstaviti. Kljub temu želimo predstaviti določene pomembnejše elemente.

Vzpostavitev komunikacije med napravo Android in priključeno napravo poteka v več korakih. Najprej je treba priključeno napravo v aplikaciji zaznati. Nato je treba od uporabnika pridobiti ustrezna dovoljenja, da aplikacija lahko komunicira z omenjeno napravo. S pridobljenimi dovoljenji lahko v naslednjem koraku pridobimo ustrezne podatke o USB-povezavi, o vmesniku povezave in o zaključnih točkah (angl. *Endpoints*) komunikacije. Na slednje se nato sklicujemo pri pošiljanju in branju podatkov. Postopek vzpostavitve povezave med aplikacijo in priključeno napravo je podrobneje opisan v dokumentaciji Android [47]. Z vzpostavljeno povezavo pa smo lahko pričeli s pošiljanjem podatkov med napravama.

V primeru naše aplikacije se komunikacija vrši samo ob pritisku na gumb *Slikaj* v funkciji *btn_remoteUsb_shoot_onClick()*. Znotraj le-te se dvakrat kreira matrika bajtov. V prvi matriki je shranjen niz podatkov, ki fotoaparatu sporoči, naj ustvari novo sejo za operacijo. V drugi pa je shranjen ukaz za izvedbo operacije *Slikaj*. Oba ukaza se v zaporedju pošljeta fotoaparatu, kjer ta nato izvede ustrezno operacijo. Ob tem je pomembno tudi, da po pošiljanju ukaza preberemo tudi povratne informacije. Ugotovili smo namreč, da brez tega koraka funkcionalnost ne deluje pravilno. Samo pošiljanje in branje podatkov se izvaja v dodatnih funkcijah *write()* in *read()*.

Funkcija *btn_remoteUsb_shoot_onClick()*:

```
public OnClickListener btn_remoteUsb_shoot_onClick = new OnClickListener() {
	@Override
	public void onClick(View v) {
		try {
			byte[] ukaz1 = new byte[] { (byte) 0x10, 0x00, 0x00, 0x00, 0x01, 0x00,
0x02, 0x10, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00 };
			write(ukaz1, ukaz1.length, 1000); // odpiranje seje
			byte buf1[] = read(1000); // preberemo povratne informacije
			byte[] ukaz2= new byte[] { (byte) 0x0c, 0x00, 0x00, 0x00, 0x01, 0x00,
0x0f, (byte) 0x91, 0x01, 0x00, 0x00, 0x00 };
			write(ukaz2, ukaz2.length, 1000); // ukaz zajami posnetek
			byte buf2[] = read(1000); // preberemo povratne informacije
		} catch (Exception e) {
			Log.e(TAG, "Napaka v USB komunikaciji!");
		}
	}
};
```

Funkcija *write()*:

```
public void write(byte[] data, int length, int timeout) {
	mDeviceConnection.bulkTransfer(mEndpointBulkOut, data, length, timeout);
}
```

Funkcija *read()*:

```
public byte[] read(int timeout) {
	byte data[] = new byte[inMaxPS];
	mDeviceConnection.bulkTransfer(mEndpointBulkIn, data, inMaxPS, timeout);
	return data;
}
```

Največ težav pri izdelavi USB-načina upravljanja smo imeli ravno s kreiranjem zgoraj opisanih ukazov za odpiranje seje in za zajem posnetka. Naša prvotna zamisel je bila, da bomo v naši izvedbi komunikacije uporabili le ukaze, ki jih pri delovanju uporablja namizna aplikacija EOS Utility. Izkazalo se je, da nam z uporabljenimi identičnimi ukazi ni uspelo izvesti operacije slikanja. Točnega vzroka za to nismo ugotovili, domnevamo pa, da je vzrok v drugačnem protokolu ukazov, ki uporablja namizna aplikacija. Ne glede na to smo delo na naši aplikaciji nadaljevali. Rešitev smo na koncu našli v dokumentaciji predhodno omenjenega projekta Arduino za USB-upravljanje fotoaparatov [49]. V njej smo namreč zasledili vrednosti ukazov za začetek nove seje (šestnajstiška vrednost 1002) in za operacijo *Slikaj* (šestnajstiška vrednost 910f). Te vrednosti smo nato vključili v funkcijo *btn_remoteUsb_shoot_onClick()*, ki smo jo predstavili zgoraj. Na tak način smo dobili delujočo različico upravljanja fotoaparata prek USB-povezave.

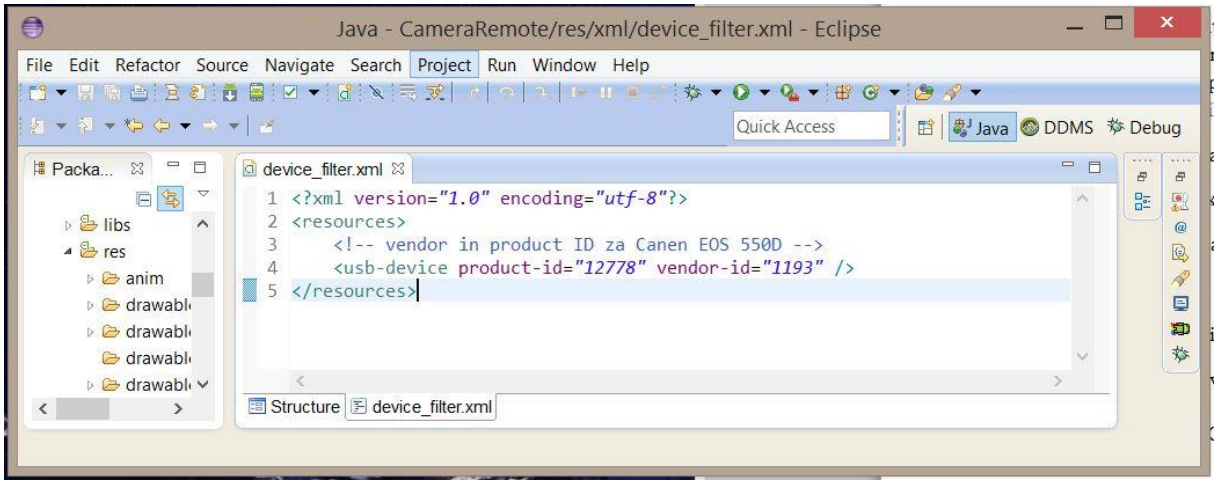
Na kratko bi radi predstavili še nekatere pomembnejše elemente pri izdelavi aplikacije za ta sklop upravljanja.

Ena od možnosti, ki jih platforma Android v primeru USB-komunikacije nudi razvijalcu, je namreč tudi to, da lahko definira, s katerimi napravami lahko aplikacija komunicira. Možno je namreč kreirati filter po proizvajalcu, po modelu naprave, po tipih naprav (fotoaparat). Natančnejši seznam možnih atributov je naslednji:

- *vendor-id*: identifikacijska koda proizvajalca;
- *product-id*: identifikacijska koda naprave;
- *class*: razred USB naprav;
- *subclass*: podrazred USB naprav;
- *protocol*: protokol naprave ali vmesnika.

Podatek z nobenim ali z vsemi od zgoraj naštetih atributov se zapiše v posebno XML-datoteko. Nanjo se ob filtriranju sklicuje aplikacija. V našem primeru smo v aplikacijo dodali filter samo za fotoaparat Canon 550D (slika 43), z nadaljnjim razvojem pa bi nabor podprtih naprav lahko razširili. V prvi vrsti še za ostale modele fotoaparatov Canon EOS, kasneje pa mogoče tudi za modele naprav drugih proizvajalcev. Podatek o atributu *vendor-id* in *product-id* smo v našem primeru pridobili kar iz aplikacije, ki smo jo uporabili za zajem podatkov

USB-komunikacije. Omenjene podatke je za posamezno USB-napravo mogoče najti tudi na svetovnem spletu.



Slika 43: Datoteka za filtriranje naprav za USB-povezavo

Pripomniti želimo tudi, da je pogoj za uporabo daljinskega upravljanja prek USB-povezave ustrezna mobilna naprava, ki podpira gostovalni USB-način, ter da je treba v tem primeru za povezovanje obeh naprav uporabiti poseben tip USB-kabla. Kabel (slika 44) mora biti izdelan po specifikaciji USB On-The-Go oziroma USB OTG [50].

Slika 44: Kabel USB On-The-Go [51]

USB-naprave za medsebojno komunikacijo uporabljajo arhitekturo gospodar/suženj. Naprava v vlogi gospodarja oziroma gostitelja vzpostavi in izvaja vso komunikacijo prek USB-vodila, naprava v vlogi sužnja pa ima posredno vlogo v komunikaciji. Različne naprave so zasnovane, da se obnašajo na en ali drug omenjeni način. Osebni računalniki denimo imajo vedno vlogo gospodarja. Naprave, kot so fotoaprati, tiskalniki in mobilni telefoni, pa so izdelani na način, da se v USB-povezavi obnašajo kot sužnji. Na tak način so lahko omenjene naprave tudi manjše in manj kompleksne za izdelavo. Da se lahko takšna naprava obnaša kot gospodar v USB-povezavi, mora imeti v ta namen tudi specifično strojno in programsko

podporo. Tudi v tem primeru se naprava vseeno primarno obnaša kot suženj. Še vedno lahko na primer mobilni telefon priključimo na osebni računalnik in bo deloval kot suženj. Prav iz tega vzroka se uporablja kabel On-The-Go, saj omogoči, da se naprava obnaša kot gostitelj in gospodar povezave, če nanjo nato prek omenjenega kabla priključimo drugo suženjsko napravo (na primer fotoaparat).

Omenjeni kabel se sicer vizualno ne razlikuje od ostalih USB-kablov.

5 Sklepne ugotovitve

V diplomskem delu je bila predstavljena izdelava aplikacije za operacijski sistem Android s funkcionalnostjo upravljanja fotoaparata DSLR Canon EOS 550D. Opisali smo izdelavo tako osnovne strukture aplikacije Android kot tudi izdelavo posameznih funkcionalnosti, specifičnih za temo diplomskega dela. Razdelili smo jih na tri osnovne sklope, in sicer na daljinsko upravljanje z uporabo vgrajenega IR oddajnika, na kabelsko upravljanje in na upravljanje prek USB-povezave. Pri izdelavi posameznih sklopov smo na posameznih področjih naleteli na specifične probleme, ki smo jih nato morali rešiti. Pri izdelavi upravljanja z IR komunikacijo nam je tako predstavljalo problem dejstvo, da med proizvajalci različnih mobilnih naprav ni standardiziranega načina dela z IR oddajnikom. Primorani smo bili izdelati funkcionalnost, ki jo podpira le določen del mobilnih naprav določenega proizvajalca oziroma naprav, ki poganjajo najnovejšo različico operacijskega sistema Androd 4.4 KitKat. Pri izdelavi upravljanja prek kabelske povezave smo večjo pozornost morali nameniti načinu, kako dejansko povezati obe napravi. Rešitve, ki smo jih na koncu našli, sicer niso optimalna, saj zahtevajo dodatna vmesna elektronska vezja, vendar smo na ta način problem vendarle uspešno rešili. Pri izdelavi upravljanja prek USB-povezave lahko rečemo, da smo imeli na voljo zelo dobro in jasno dokumentacijo o poteku komunikacije prek USB-vodila na platformi Android. Na problem smo tukaj naleteli zaradi strojnih zahtev, ki jih mora izpolnjevati mobilna naprava, če želi izrabiti ta način upravljanja. Poleg tega pa je največjo težavo v tem sklopu izdelave aplikacije predstavljala pomanjkljiva javno dostopna dokumentacija o implementaciji USB-komunikacijskih protokolov med napravo in fotoaparatom.

Kljub vsem opisanim zapletom, s katerimi smo se tekom razvoja soočili, nam je uspelo izdelati mobilno aplikacijo, ki izpolnjuje zastavljene cilje. Z nadaljnjim razvojem in raziskovanjem bi jo bilo možno še na različne načine izboljšati. Zamisli, ki se porajajo, so, da na primer razširimo upravljane prek USB-povezave z možnostjo nastavljanja različnih parametrov fotoaparata. Izboljšali bi lahko tudi upravljanje prek IR povezave na način, da bi z nadaljnjim razvojem aplikaciji dodali tudi podporo za upravljanje drugih znamk in modelov fotoaparatov.

Izdelava aplikacije je bila zanimiva praktična izkušnja. Pridobili smo mnogo novega znanja glede razvoja na platformi Android in za mikrokontroler Arduino. Ker se tudi sami amatersko ukvarjamo s fotografstvom, nam bo aplikacija v praksi služila ob potrebi po daljinskem upravljanju lastnega fotoaparata.

Viri

[1] (2013) Canon EOS. Dostopno na: http://en.wikipedia.org/wiki/Canon_EOS

[2] (2013) Canon EOS 550D. Dostopno na: http://en.wikipedia.org/wiki/Canon_EOS_550D

[3] (2010) Canon EOS T2i (550D) Review. Dostopno na: http://digital-photography-school.com/canon-eos-t2i-550d-review

[4] (2013) EOS Remote Controllers. Dostopno na: http://shop.usa.canon.com/shop/en/catalog/product-accessories/eos-digital-slr-camera-accessories/eos-remote-controllers

[5] (2013) EOS for Enthusiasts. Dostopno na: http://www.canon-europe.com/For_Home/Product_Finder/Cameras/Digital_SLR/enthusiasts/index.aspx

[6] (2012) Canon EOS-1D X First Test. Dostopno na: http://www.shutterbug.com/content/canon-eos-1d-x-first-test

[7] (2013) Bulb (photography). Dostopno na: http://en.wikipedia.org/wiki/Bulb_(photography)

[8] (2013) Android (operating system). Dostopno na: http://en.wikipedia.org/wiki/Android_(operating_system)

[9] (2013) Android Fragmentation Visualized. Dostopno na: http://opensignal.com/reports/fragmentation-2013/

[10] (2013) iOS. Dostopno na: http://en.wikipedia.org/wiki/IOS

[11] (2013) Android software development. Dostopno na: http://en.wikipedia.org/wiki/Android_software_development

[12] (2013) Android Developer Tools. Dostopno na: http://developer.android.com/tools/help/adt.html

[13] (2013) Eclipse (software). Dostopno na: http://en.wikipedia.org/wiki/Eclipse_(software)

[14] (2013) Android Emulator. Dostopno na: http://developer.android.com/tools/help/emulator.html

[15] (2013) Dashboards. Dostopno na: http://developer.android.com/about/dashboards/index.html

[16] (2013) USB Host. Dostopno na: http://developer.android.com/guide/topics/connectivity/usb/host.html

[17] (2013) Honeycomb. Dostopno na: http://developer.android.com/about/versions/android-3.0-highlights.html

[18] (2013) Ice Cream Sandwich. Dostopno na: http://developer.android.com/about/versions/android-4.0-highlights.html

[19] (2013) Android KitKat. Dostopno na: http://developer.android.com/about/versions/kitkat.html

[20] (2013) Hello world program. Dostopno na: http://en.wikipedia.org/wiki/Hello_world_program

[21] (2013) Themes. Dostopno na: http://developer.android.com/design/style/themes.html

[22] (2013) Localizing with Resources. Dostopno na: http://developer.android.com/guide/topics/resources/localization.html

[23] (2013) Rocketfish™ - Remote Wireless Shutter Control for Canon. Dostopno na: http://www.rocketfishproducts.com/products/cameras-camcorders/RF-RSCWLC12.html

[24] (2013) Arduino Uno. Dostopno na: http://arduino.cc/en/Main/arduinoBoardUno

[25] (2013) IR Receiver Modules for Remote Control Systems. Dostopno na: https://www.sparkfun.com/datasheets/Sensors/Infrared/tsop382.pdf

[26] (2013) Using an IR Sensor. Dostopno na: http://learn.adafruit.com/ir-sensor/using-an-ir-sensor

[27] (2013) Synchronize multiple Canon Rebel T4i video shooting with reverse-engineered RC-6 protocol. Dostopno na: http://zavax.wordpress.com/2013/01/01/synchronize-multiple-canon-rebel-t4i-video-shooting-with-customized-remote-control/

[28] (2013) Infra Red Signals - Pulse Width Method. Dostopno na: http://www.hifi-remote.com/infrared/IR-PWM.shtml

[29] (2013) IrDude. Dostopno na: https://github.com/rngtng/IrDude

[30] (2013) ConsumerIrManager. Dostopno na:

http://developer.android.com/reference/android/hardware/ConsumerIrManager.html

[31] (2014) AUDIO CABLE 2.5MM 8". Dostopno na: http://www.cooking-hacks.com/shop/on-demand-products/audio-cable-2-5mm-8

[32] (2013) Camera remote release pinout list. Dostopno na: http://www.doc-diy.net/photo/remote_pinout/

[33] (2013) Opto-isolator. Dostopno na: http://en.wikipedia.org/wiki/Opto-isolator

[34] (2013) Galvanic isolation. Dostopno na: http://en.wikipedia.org/wiki/Galvanic_isolation

[35] (2013) IC 4N35 Optocoupler. Dostopno na: http://www.engineersgarage.com/electronic-components/4n32-optcoupler

[36] (2013) Audacity (audio editor). Dostopno na: http://en.wikipedia.org/wiki/Audacity_(audio_editor)

[37] (2013) MediaPlayer. Dostopno na: http://developer.android.com/reference/android/media/MediaPlayer.html

[38] (2013) AudioTrack. Dostopno na: http://developer.android.com/reference/android/media/AudioTrack.html

[39] (2013) Android Tone Generator. Dostopno na: https://gist.github.com/slightfoot/6330866

[40] (2013) Sine. Dostopno na: http://en.wikipedia.org/wiki/Sine

[41] (2013) Sign function. Dostopno na: http://www.answers.com/topic/sign-function

[42] (2014) Operational amplifier applications. Dostopno na: http://en.wikipedia.org/wiki/Operational_amplifier_applications

[43] (2014) TL081. Dostopno na: http://www.ti.com/lit/ds/symlink/tl081-n.pdf

[44] (2013) Intervalometer. Dostopno na: http://en.wikipedia.org/wiki/Intervalometer

[45] (2013) Time-lapse photography. Dostopno na: http://en.wikipedia.org/wiki/Time-lapse_photography

[46] (2013) Canon eos cameras principles of interfacing and library description. Dostopno na: http://www.circuitsathome.com/canon-eos-cameras-principles-of-interfacing-and-library-description

[47] (2013) USB Host. Dostopno na:
http://developer.android.com/guide/topics/connectivity/usb/host.html

[48] (2013) Samples. Dostopno na: http://developer.android.com/tools/samples/index.html

[49] (2013) PTP USB control camera data. Dostopno na:
http://www.circuitsathome.com/ptpusb-control-camera-data#EOS550D

[50] (2013) USB On-The-Go. Dostopno na: http://en.wikipedia.org/wiki/USB_On-The-Go

[51] (2014) Micro USB Host Cable (OTG Cable). Dostopno na:
http://www.diygadget.com/micro-usb-host-cable-otg-cable-xoom-galaxy-s2-galaxy-note-galaxy-nexus-acer-iconia-a510-toshiba-tg01-archos-g9-and-more.html