Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Infračervený ovládač pre Android zariadenia

Bakalárska práca

2017
Ladislav Feldsam

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Infračervený ovládač pre Android zariadenia

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Peter Borovanský, PhD.
Konzultant:	RNDr. Michal Forišek, PhD.

Bratislava, 2017
Ladislav Feldsam

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ladislav Feldsam
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Infračervený ovládač pre Android zariadenia
Infrared remote control for Android devices

Cieľ: Cieľom práce je vytvoriť hardvérové zariadenie transformujúce analógový signál z audio-jacku na infračervený signál (IR) ovládajúci široké spektrum rôznych IR zariadení. Prvý návrh takéhoto zariadenia bol súčasťou ročníkového projektu študenta. Cieľom práce je tiež vytvoriť aplikáciu pre platformu OS Android, ktorá z mobilného zariadenia pomocou modulu spraví diaľkový ovládač zariadení. Okrem vysielania IR signálu, prídavný modul rozpoznáva a dekóduje signál zachytený iným IR ovládačom (napr. televízor, dataprojektor, ..). Po dekódovaní to následne vie aplikácia reprodukovať. Aplikácia preto bude fungovať nezávisle od databázy kódov rôznych ovládačov. Bude dekódovať signál z fyzických ovládačov nasnímaním ich IR signálov. Práca má hardvérovú a softvérovú časť. V prvej sa predpokladá doriešenie a overenie modulátora IR signálu pre správne fungovanie vysielania, a následne obvod pre snímanie a dekódovanie prichádzajúceho signálu. Súčasťou práce je aj jednoduchý hardvér pre testovanie na PC. Ďalším krokom je emitovanie IR signálu podľa zvolenej databázy kódov, ktorú vo finálnom riešení chceme eliminovať. Následne autor navrhne softvérové riešenie dekodéra IR signálu. Výsledkom práce z užívateľského hľadiska je aplikácia pre mobilné zariadenie. Preto aspekt užívateľského rozhrania musí byť v práci rovnako zohľadnený.

Vedúci: RNDr. Peter Borovanský, PhD.
Konzultant: RNDr. Michal Forišek, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 26.10.2016

Dátum schválenia: 31.10.2016 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Chcel by som sa poďakovať školiteľovi za spoluprácu pri tvorbe tejto práce. Tiež jeho rýchlym odpovediam na otázky, ktoré som práve potreboval.

Abstrakt

Cieľom bakalárskej práce je vytvoriť aplikáciu pre Android, ktorá eliminuje potrebu viacerých ovládačov na ovládanie viacerých zariadení s infračerveným signálom. Práca zahŕňa vlastný hardvérový modul, ktorý skenuje a generuje infračervený signál. Ten je generovaný aplikáciou a je zosilnený externým zdrojom, alebo batériou zariadenia kvôli väčšiemu dosahu ovládania. Každé zariadenie má iné hraničné tolerancie infračerveného signálu, ktoré sú v aplikácii možné nastaviť pre rôzne zariadenia (manuálne aj semi-automaticky). Každý naskenovaný/dekódovaný signál využíva jedno z týchto nastavení pre optimálne fungovanie. Aplikácia ponúka užívateľovi podobné rozhranie ku klasickým ovládačom, ktoré si môže prispôsobiť. Každému tlačidlu užívateľ môže nastaviť viacej signálov. To umožní, aby mohol ovládať jedným stlačením viacej funkcií jedného zariadenia, alebo viacerých rôznych zariadení sekvenčne naraz.

Kľúčové slová: Android, hardvér, infračervený signál, ovládač

Abstract

The goal of this bachelor thesis is to create an Android application that eliminates the need of multiple infrared remote controllers. The aim is to control multiple devices with infrared signal. A custom hardware module presented in this thesis scans and generates infrared signals. It is generated by the application and amplified by an external source or battery of the device in order to extend range control. Each device has different infrared tolerances, that can be set in the application for different devices (manual and semi-automatic). Each scanned/decoded signal uses one of these settings for optimal performance. The application offers a user interface similar to classic controller that can be customized. A user can set multiple signals for any button. This allows to control multiple features of one device or multiple different devices by pressing one button.

Keywords: Android, hardware, infrared signal, remote control

Obsah

Zoznam obrázkov

Úvod

Cieľom tejto práce je vytvoriť aplikáciu pre Android zariadenia, ktorá využíva v tejto práci vytvorený hardvérový modul. Tento modul umožní vysielať a prijímať infračervený (IR) signál pre zariadenia s audio jackom.

Modul prijíma špeciálne generovaný tón (resp. melódiu) zvukovou kartou Android zariadenia. Ten pretransformuje na modulované infračervené svetlo ovládaného zariadenia. Ním vieme zariadenie ovládať, ako stlačením jedného z tlačidiel na originálnom infračervenom ovládači ovládaného zariadenia.

Android zariadenie, ktoré úzko používa modul vytvorený v práci, vie tento signál naskenovať, a potom dekódovať pomocou naprogramovanej aplikácie. Tento naskenovaný a dekódovaný signál sa potom uloží vo vhodnom formáte do databázy pre ďalšie spracovanie. Odtiaľ ju môžeme získať a vygenerovať tón pre modul.

Aplikácia zjednodušuje ovládanie zariadení, používajúcich infračervené ovládače (televízor, dataprojektor, drony ovládajúce sa s IR ovládačmi, atď.). To znamená, že aplikácia rozozná a dekóduje signál z ovládača zariadenia a následne reprodukuje.

Výsledkom tohto procesu aplikácia umožní, naprogramovať makro tlačidlo, aby sme vedeli ovládať viacero zariadení naraz. Toto eliminuje potrebu viacerých ovládačov a tým zjednoduší ovládanie pre užívateľa.

Kapitola 1

Stav problematiky

V súčasnosti existujú podobné riešenia tejto problematiky, ako napríklad riešenie IR Droid [2]. Existujú aj kompaktné riešenia nášho hardvérového modulu, ktoré sú úplne pasívne a nevyužívajú žiadne externé napájanie. Využívajú len dve diódy, ktoré sú opačne zapojené. Tým sa umožni dvojnásobná frekvencia tak, že jedna dióda prepúšťa časti, kde je napätie kladné a druhá, kde záporné. Tento modul má výhodu, že je veľmi maličký a kompaktný, no nevýhodu, že jeho maximálny dosah aj pri zvukových kartách, ktoré dokážu dodať vyššie napätia, je rádovo desiatky centimetrov. Pri slabších ani len nefungujem lebo nedokáže spínať diódy vôbec.

Riešenie IR Droid [2], využíva baterku na napájanie, tak isto aj naše, ale cez USB rozhranie. Toto umožňuje aj napájanie priamo z USB rozhrania, ktoré dodáva 5V (pre nás ideálne). To nám umožňuje využiť batériu, ktorú je možné zapojiť cez toto rozhranie. Využívajú dve opačne zapojené diódy, čo nám neumožní presné nastavenie dĺžky impulzov a čas medzi nimi. V tejto práci si ukážeme riešenie, ktoré tento problém vyriešiť dokáže.

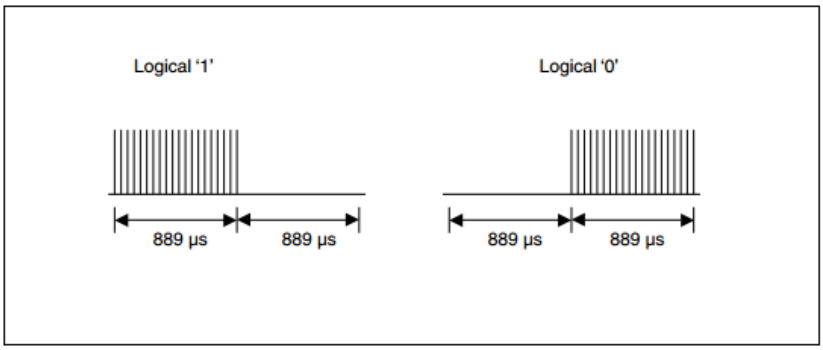
Najprv si ale priblížme ako infračervený signál v ovládačoch funguje.

1.1 Infračervený signál

Existuje viac protokolov pre infračervený signál, ale my sa budeme zaoberať hlavne s najviac rozšíreným protokolom RC5, ktorý je predchodcom protokolu RC6. Protokol RC6 sa veľmi nelíši od RC5, a tak pre jednoduchosť si to môžeme dovoliť, kvôli našej voľbe formátu ukladania signálu.

Popis protokolu v tejto kapitole je založený na popise od firmy STMicroelectronics [6]. Kód RC5 protokolu je 14-bitové slovo a používa Manchastrovú moduláciu s prenosovou frekvenciou 36-42kHz (záleží od výrobcu). Každý bit má rovnakú dĺžku 1,778ms, kde logická nula je kódovaná tak, že prvá polovica tohto času sa nevysiela žiadny sig-

nál, čo druhú polovicu sa vysiela signál s prenosovou frekvenciou 36-42kHz. Logická jednotka je presne naopak. Prvá polovica sa vysiela a druhá polovica je ticho. (Obr. 1.1)



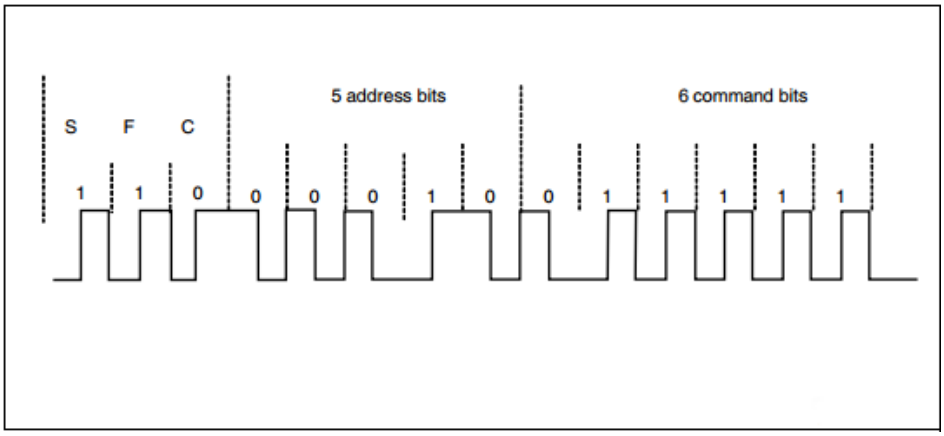
Obr. 1.1: Ukážka logickej jednotky a nuly v protokole RC5.

Zdroj: ST Microelectronics. AN3174 Application note [6]

RC5 dokáže generovať 2048 (32x64) rôznych príkazov organizovaných na 32 skupinách. Každá skupina ma 64 rôznych príkazov.

Každý RC5 signál obsahuje nasledujúce časti (Obr. 1.3):

- Štartovací bit (S): vždy logická 1 dĺžky jedného bitu.
- Bit oblasti (F): dĺžky 1 bitu. Ak 0, tak príkaz je na spodnej oblasti (0-63), ak 1, tak je príkaz na hornej oblasti (64-127). Tento bit bol pridaný neskôr, keď sa zistilo, že 64 bitové príkazy na jedno zariadenie boli nedostatočné. Predtým bit oblasti bol spoločný so štartovacím bitom.
- Kontrolný bit alebo spínajúci bit (C): dĺžky 1 bitu. Tento bit sa spína vždy, keď sa tlačidlo stlačí. Toto umožní zariadeniu rozhodnúť, že sa za sebou stlačilo to isté tlačidlo 2 alebo viac krát (napríklad "11"bude pre zariadenie znamenať "11").
- Adresa: dĺžky 5 bitov, ktorá rozhodne, do ktorej z 32 skupín patrí.
- Príkaz: dĺžky 6 bitov. Spoločne s bitom oblasti reprezentuje jedno zo 128 možností RC5 protokolu.



Obr. 1.3: Ukážka ako je rozdelený signál na logické časti

Zdroj: ST Microelectronics. AN3174 Application note [6]

Preto, aby sme sa vyhli kolíziám viacerých signálov medzi signálmi je vložený čas nečinnosti dĺžky 50 bitov. Znamená to, že periodicita medzi signálmi je 64 x 1 bit : 64 x 1,778 = 113,792 ms.

1.2 Riešenie IR Droid

IR Droid [2] má dve riešenia. Jedno riešenie je podobné nášmu. Využíva audio jack, kým druhé je bezdrôtové a funguje cez Bluetooth.
My sa budeme zaoberať len s riešením, ktoré využíva audio jack port, keďže je podobné k nášmu riešeniu. Porovnáme, v čom sa odlišujú, prípadne, v čom sú podobné.

Poznámka: Riešenia využívajúce audio jack, sú tiež schopné fungovania cez Bluetooth, ak sú pripojené do externého zariadenia podporujúce prehrávanie hudby cez Bluetooth a majú vývod s audio jack portom. O funkčnosti tejto možnosti sme nenašli žiadne informácie, a tiež sa nevyskúšala, kvôli chýbajúcemu modulu, ktorý by nám testovanie umožnil.

1.2.1 Nevýhody a porovnanie s IR Droid

Priblížíme si konkrétnejšie nevýhody riešenia IR Droid a povieme si, ktoré z nich budeme riešiť v práci, aby sme tieto problematiky resp. nepríjemnosti vylepšili.

Postupne si porovnáme riešenie modulu a aplikácie IR Droid k riešeniam tejto práce a povieme si, čo z nich budeme riešiť.

Modul

Obr. 1.5: Foto IR Droid modulu v1.0.

Zdroj: IR Droid [2]

Keďže ich (obrázok 1.7) a aj náš modul (opísaný v kapitole Hardvér 2) pracuje cez audio jack, tak, že zvuk sa generuje v zvukovej karte zariadenia. Toto prináša nejaké limitácie, ako je maximálna frekvencia, ktorú zariadenie dokáže generovať a, tak isto aj jeho maximálne napätie (maximálna hlasitosť).

V predchádzajúcej sekcii infračerveného signálu (sekcia 1.1) sme spomínali, že prenosová frekvencia jednotlivých impulzov sa pohybuje od 36-42kHz. Maximálna frekvencia už aj štúdiového aparátu je maximálne do 26kHz, čo nám z ďaľeka nestačí, aby sme dosiahli 42kHz. Preto riešenie IR Droidu, a tiež aj riešenie v modulu práce, má nejaký prístup zdvojnásobiť generovanú frekvenciu.

Jednou z najväčších nevýhod riešenia IR Droid je zdvojnásobenie frekvencie využívajúcej dve navzájom opačne prepojené IR diódy, ktoré fungujú nasledovne:

Jedna dióda emituje infračervené svetlo pri častiach, kde napätie signálu je kladné a druhá, kde je záporné. Táto metóda nám prakticky zdvojnásobí frekvenciu tým, že využije aj záporné časti signálu, ktoré by pri využití jednej diódy boli ignorované.

Týmto spôsobom, ale nevieme riadiť, ako dlho majú jednotlivé impulzy trvať, a aké medzere majú medzi nimi byť.

Toto nám nevadí, ak modul je nastavený presne podľa štandardu, ale v praxi každé zariadenie má inú zvukovú kartu, ktorá môže vytvárať o trochu inú zvukovú stopu. V takýchto prípadoch je dobré ak je modul flexibilnejší ohľadom týchto nastavení. Preto v našom riešení, tento problém riešime v kapitole 2.

Ďalšou problematikou je už spomínaná "maximálna hlasitosť". Ak máme zariadenie, ktoré nedokáže dodať dostatočné napätie na zvukovej karte, tak nám ani nespína diódu.

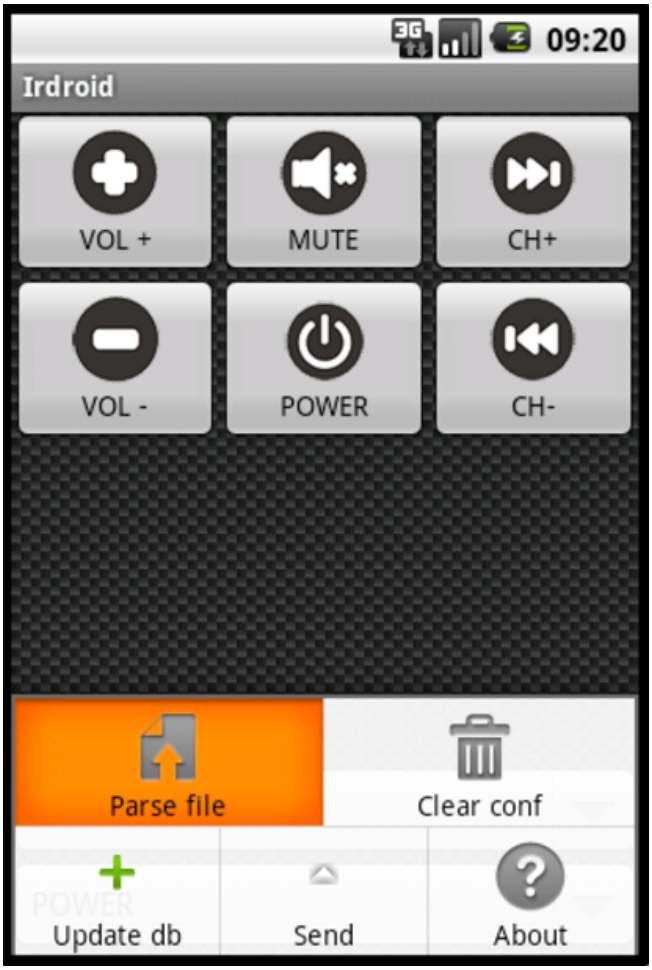
S týmto vznikajú nepríjemnosti použitia externého napájania pre zosilnenie signálu. Ak by sme tak neurobili, maximálny dosah ovládača by bol rádovo desiatky centimetrov, čo už prakticky nie je diaľkové ovládanie.

Modul v práci má v sebe vybudovanú tiež aj Infračervenú prijímajúcu diódu, ktorá umožní skenovanie infračervených vĺn. Modul IR Droidu toto nemá, lebo má iný softvérový prístup, ktorý si priblížíme v nasledujúcej podsekcii.

Softvér

Softvér IR Droidu funguje nad databázou kódov jednotlivých zariadení. Táto databáza môže mať veľa nevýhod. Napríklad aj to, že zariadenie, ktoré chceme ovládať, v nej nebude, lebo je nové, alebo v nej vôbec nie je zaznamenané.

Softvér sme nevedeli poriadne otestovať, lebo modul pre IR droid nebol zakúpený. Každopádne po nainštalovaní, softvér nebol vôbec intuitívny a pravdepodobne sa v ňom dá naraz používať len jedno zariadenie a presne to, ktoré je práve nastavené.



Obr. 1.7: Foto IR Droid aplikácie pre v1.0 modul

Zdroj: IR Droid [2]

Softvér, ktorý bol vytvorený v práci nepoužíva žiadnu databázu zariadení. Namiesto toho používa prijímajucu diódu modulu, ktorou sa signál naskenuje, dekóduje a potom uloží v rozumnom formáte pre nasledujúce použitie softvéru.

Taktiež GUI pre užívateľa bolo vytvorené, aby bolo intuitívne a veľmi podobné originál infračerveným ovládačom (mriežkovité rozloženie). Každé tlačidlo na tomto rozložení môže patriť pre rôzne zariadenie, alebo aj pre viac zariadení naraz, ak v danom tlačidle je nastavené viac signálov. Táto možnosť je povolená v softvéri práce a budeme ho volať makro tlačidlo.

IR Droid tiež podporuje nejakú formu makro tlačidla stiahnutím úplne inej, druhej aplikácie ktorá funguje cez LIRC (Linux infrared remote control) alebo WinLIRC (Windows ekvivalent), ktorá potrebuje zas iný WIFI modul IR Droidu, ktorým sa nebudeme v tejto práci zaoberať, lebo je to úplne iný prístup pre ovládanie ako sa v tejto práci používa.

Kapitola 2

Hardvér

V tejto časti práce sa budeme zaoberáme s hardvérovým modulom, ktorý umožní nášmu Android zariadeniu prijímať a vysielať IR signál.

Niektoré mobilné zariadenia majú v sebe zabudovaný IR vysielač a prijímač. Je to veľmi zriedkavé, a preto sa v tejto práci rozhodlo, aby to nebol rozhodujúci faktor, či zariadenie bude možné využívať aplikáciu alebo nie.

Modul v tejto práci na prijímanie a vysielanie využíva audio jack port, ktorý väčšina zariadení má.

2.1 Generovanie signálu

IR modul vytvára signál zo zvukovej stopy, ktorú dostane z audio jack portu zariadenia. Na to aby modul v práci vytvoril požadovaný infračervený signál pre riadené zariadenie musíme vygenerovať presnú zvukovú stopu.

Keďže zvuk sa tvorí v zvukovej karte zariadia, kde narazíme na drobné problémy, ktoré budeme musieť vyriešiť.

1. Napriek tomu, že generujeme štvorcový (digitálny) signál musíme počítať s tým, že zvuková karta ho skreslí na analógový, ale ak chceme ideálny infračervený signál potrebujeme digitálny.

2. Zvuková karta bežných mobilných zariadení dokáže generovať v najlepšom prípade maximálne 23kHz, no signál IR signálu má prenosovú frekvenciu 36kHz.

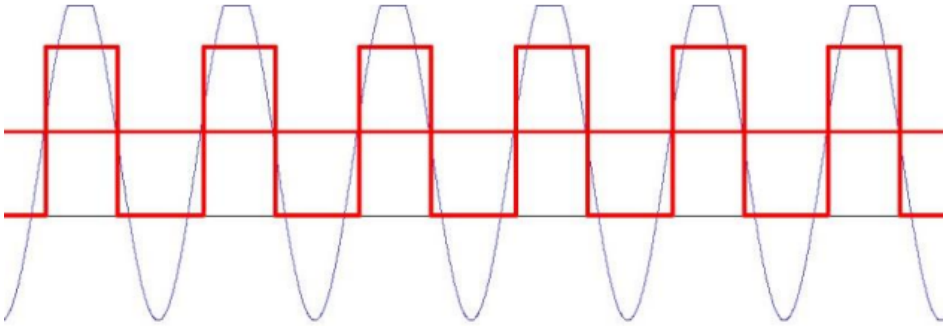
3. Signál zo samotnej zvukovej karty nie je dostatočne silný na to, aby spínal IR diódu tak, aby sme vedeli ovládať zariadenie z rozumnej vzdialenosti a preto bude potrebné nejaké externé napájanie.

Prvý a tretí problém vieme vyriešiť so zosilňovačom, ktorý nám zosilní signál. Ak ho zosilníme dostatočne silno, tak nám v jednom kroku vyhladí signál na štvorcový. Tento princíp digitalizácie je používaný aj v Shmittovom preklápacom obvode. Principiálne funguje tak, že ak na napájaní máme napríklad 5V, tak toto bude naša maximálna hranica zosilnenia. Ak aj najslabší signál budeme zosilňovať na túto hranicu, tak v praxi analógový signál sa premení na digitálny.

Druhý problém je menej triviálny. Tento problém je riešený spojením pravého a ľavého kanálu, ktoré sú od seba navzájom fázovo posunuté a spojené v XOR hradle. Túto vlastnosť môžeme využiť, keďže zvukové karty dokážu generovať dva nezávisle rôzne signály na pravom aj ľavom kanáli.

Budeme používať 3 zosilňovače. Prvé dva budú zosilňovať vstupný signál pred XOR hradlom, keďže XOR potrebuje čo najčistejší štvorcový signál pre správne fungovanie.

Aj keby naša zvuková karta dokázala dodať čistý štvorcový signál nebol by dosť silný pre XOR hradlo a preto ho zosilňujeme. Tým, že ho zosilňujeme a meníme na štvorcový signál, je pre nás výhodné z toho hľadiska, že pri veľkých frekvenciách zvuková karta signál masívne skresľuje. Tretí zosilňovač bude finálne zosilňovať samotný signál z XOR hradla pred IR diódou, aby dostala dostatočne silný signál. Posledný OP AMP zosilňovač by nebol potrebný ale použijeme ho pre využitie maximálneho potenciálu vysielajúcej IR diódy.



Obr. 2.1: Ukážka ako je vyriešený problém neskorého spínania

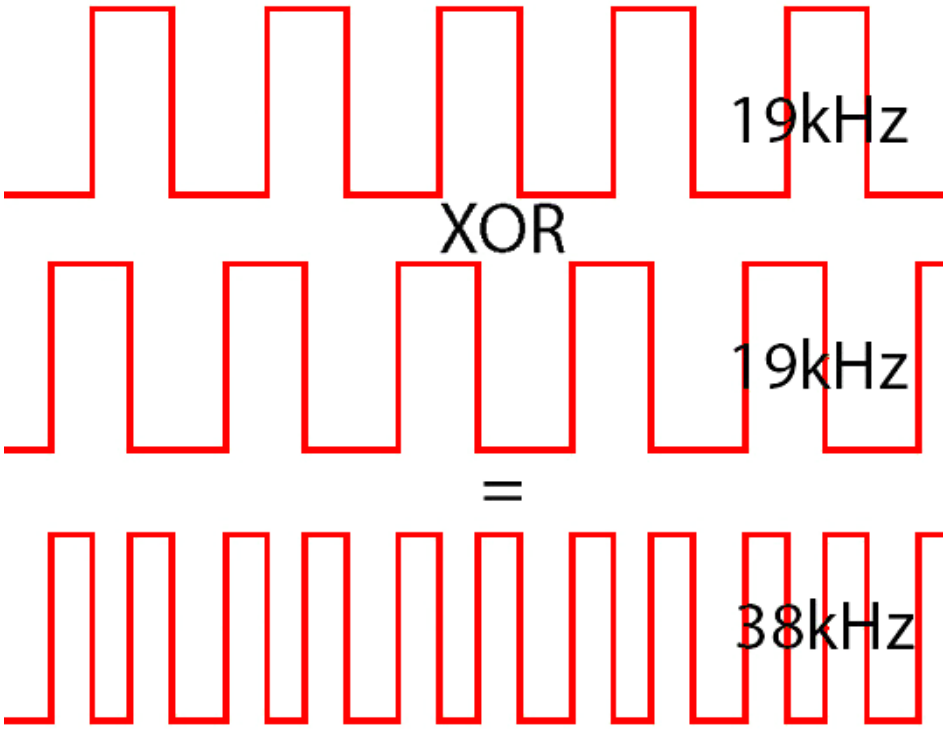
Na obrázku 2.1 je znázornený sínusový signál zo zvukovej karty a tiež zosilnený signál. Problémom zosilňovača OP-AMP, ktorý používame je, že nedokáže spínať signál okamžite, ale potrebuje nejaké minimálne napätie na to aby začalo zosilňovať. Tento problém je možný riešiť malým predĺžením kladných častí signálu aby OP-AMP skôr spínal (vertikálny posun).

funkcia XOR

Na riešenie problému, že zvuková karta nám nevie generovať dostatočne rýchly signál využijeme XOR hradlo. Zmiešame dva výstupy, ktoré sú od seba posunuté s frekvenciou 18kHz až 20kHz (záleží na danom výrobcovi akú prenosovú frekvenciu používa).

XOR potrebuje 2 vstupy. V našom prípade ľavý a pravý kanál, ktoré nemôžu byť rovnaké lebo na výstupe XOR-u by sme nič nemali. Preto budeme potrebovať pojem fázového posunu, ktorý hovorí o tom, o koľko stupňov je druhý signál horizontálne posunutý, kde 360° posun zobrazí signál na pôvodne miesto.

Ukážeme si, ako bude vyzerať vstup a výstup na OP-AMP-och pred vstupom do XOR-u na obrázku 2.2, na ktorom je znázornený 90° posun vertikálny posun.



Obr. 2.2: Ukážka dvoch horizontálne posunutých signálov o 90°a ich zjednotenie

Ak tieto dva vstupy pustíme na XOR hradlo, tak sa nám frekvencia zdvojnásobí. Kde sa signál prekrýva, signál vysielaný nebude a kde sa neprekrýva, tak vysielaný bude. XOR je veľmi vďačná logická funkcia, môžeme s ňou presne nastaviť, aké dlhé budú jednotlivé impulzy a to tým, že budeme meniť fázový (horizontálny) alebo vertikálny posun.

2.2 Skenovanie signálu

Skenovanie Signálu je umožnené v module zabudovanej prijímajúcej dióde infračerveného svetla.

Ide o jednoduché priame zapojenie (znázorenené v schéme zapojenia 2.3) tejto diódy s rezistorom do mikrofónového vstupu zariadenia.

Toto jednoduché optimistické zapojenie v zariadeniach podporujúce zvyšovanie a znižovanie hlasitosti, a taktiež prijímania hovorov, alebo iných funkcií pomocou gombíkov TRRS audio jacku, nachádzajúce sa napríklad na niektorých slúchadlách robí menšie problémy.

Aj napriek pridanej impedancie pomocou rezistora pred touto diódou, sa pri niektorých zariadeniach po zapojení nášho modulu stane, že sa zníži hlas alebo zapne prehrávanie hudby. Pri testovaných zariadeniach sa toto prestavenie po zapojení dalo nastaviť naspäť.
Tento fenómén sa deje kvôli infračervenému žiareniu z externých zdrojov ako napr. slnko alebo lampa, ktoré v prijímajúcej dióde tvori mikro napätie. Zariadenie následne tento šum zaregistruje ako keby sa stlačilo jedno z gombíkov.

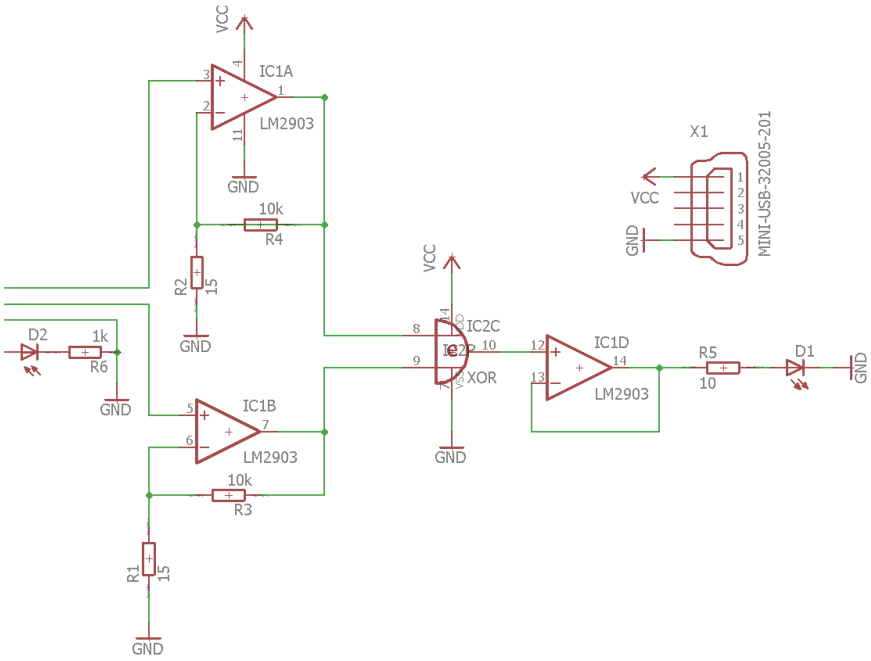
Riešenie tohto problému, by sa dalo riešiť iným komplexnejším zapojením tejto diódy, alebo inými alternatívnymi riešeniami, ktoré si rozoberieme v závere tejto práce ako návrh vylepšenia.

2.3 Schéma IR modulu

Funkciu samotného modulu sme si vysvetlili vyššie v sekcii 2.1 a 2.2.

Využíva 3 OP AMP zosilňovače, jedno XOR hradlo, vysielajúcu a príjimajúcu IR diódu, rezistory pre kontrolu napätí na jednotlivých komponentoch.

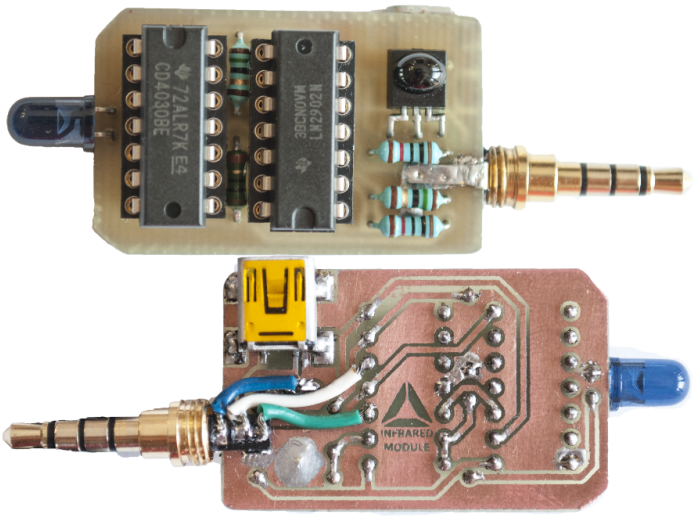
Schéma modulu je zobrazená na nasledovnom obrázku 2.3:



Obr. 2.3: Elektrická schéma zapojenia modulu práce

Modul na napájanie pre zosilňovač a XOR hradlo, využíva elektrické napätie z rozhrania USB zariadenia. Napätie z USB eliminuje potrebu bateriek, ktoré treba meniť. Rozhranie ale stále umožňuje vytvoriť malý držiak na batériu s mini USB káblom, ak by batéria bola žiadaná.

Samotný modul 2.4 bol vytvorený v domácom prostredí, tým ukazujeme aj jednoduchosť celého modulu, ktorý sa dá vyrobiť aj bez pomoci firmy vyrábajúce plošné spoje. Tento modul, by sa dal strojovou výrobou zminiaturizovať na veľmi malú veľkosť, ktorá by až tak zo zariadenia nevytŕčala.



Obr. 2.4: fyzicky vyrobený modul

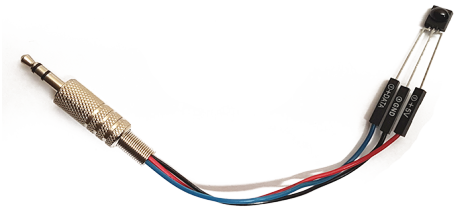
2.4 Testovací hardvér

V práci sa tiež vytvoril jednoduchý hardvér pre testovanie funkčnosti IR modulu na počítači.

Hardvér na testovanie využíva jeden fototranzistor, ktorý je pripojený do priletovaných portov zo starého počítača v 3,5mm audio jacku.

Problém zapnutia hudby a sťahovania hlasitosti, ktorý sa deje pri module do Android zariadenia sa pri tomto testovacom hardvéri netvorí.

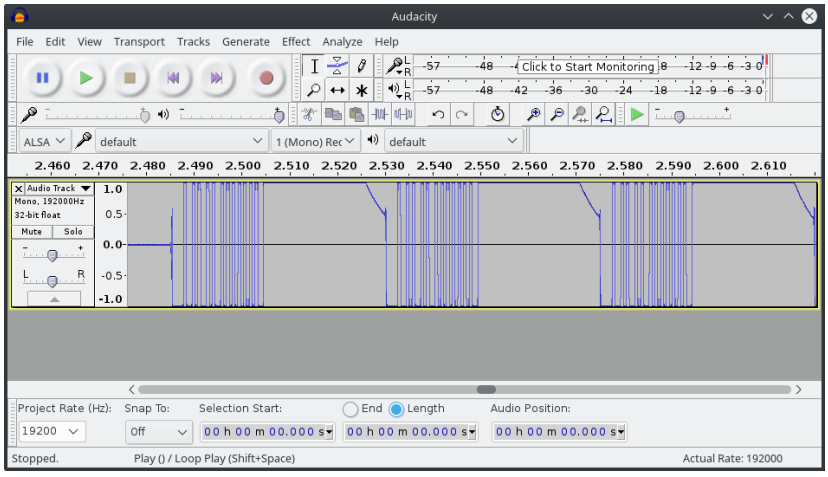
Je to kvôli tomu, že je zapojený v obyčajnom TRS audio jacku, ktorý túto funkcionalitu nepodporuje. Narozdiel od TRRS, ktorý je spoločný pre zvuk a aj mikrofón a funkcionalitu prepínania podporuje.



Obr. 2.5: Fotka testovacieho hardvéru

Tento hardvér žije v porte mikrofónu počítača a testovanie je umožnené nahrávaním zvuku a prezretím vizualizácie zvukovej stopy. Pri testovaní sa používal softvér Audacity, ale môže sa využiť hociaký iný softvér, ktorý vie nahrávať a vizualizovať zvukovú stopu.

Pri nahrávaní zvuku sa ovládačom namieri na fototranzistor testovacieho hardvéru, a stlačí sa tlačidlo, ktorého infračervené svetlo chceme vizualizovať.



Obr. 2.6: Vizualizácia zvukovej stopy infračerveného signálu v Audacity

Kapitola 3

Softvér

Kapitola softvér popisuje aplikáciu pre Android, ktorá úzko spolupracuje s IR modulom. Funkčnosť a konštrukcia modulu bola vysvetlená a ukázaná v predošlej kapitole 2.

Aplikácia bola vytvorená tak, aby jej grafické prostredie (GUI) bolo príjemné pre užívateľa, a tiež intuitívna z každého hľadiska jej používania funkcií.

V nasledujúcich sekciách si ukážeme ako jednotlivé funkcie aplikácie fungujú. Napríklad ako sa signál, ktorý je nahratý IR modulom dekóduje a potom generuje. Taktiež si povieme o tom ako je vyriešené to, že rôzne zariadenia môžu mať vlastné nastavenie pre generovanie signálu. Ako ich intuitívne vieme nastaviť a aké ďalšie možnosti nám aplikácia poskytuje. Každú grafickú časť užívateľa programu budeme ďalej menovať ako aktivitu s nejakým pomenovaním.

3.1 Hlavné funkcie

Medzi nahlavnejšie funkcie aplikácie patrí samotná aktivita ovládača, kde toto GUI je aj naším prvým privítaním do aplikácie.

Aktivita ovládača je najprv prázdna. Vidíme len tlačidlo nastavení v pravom dolnom rohu. Po kliknutí naň sa zobrazí menu, cez ktoré vieme zapnúť manažovanie tlačidiel ovládača, alebo sa dostať do druhých aktivít.

Týmito aktivitami sú napríklad pridanie signálov (Signals, sekcia 3.2), zariadení (Devices, sekcia 3.3), a tiež nastavenie niektorých globálnych nastavení (Global settings, spomenuté v podsekcii 3.2.2).

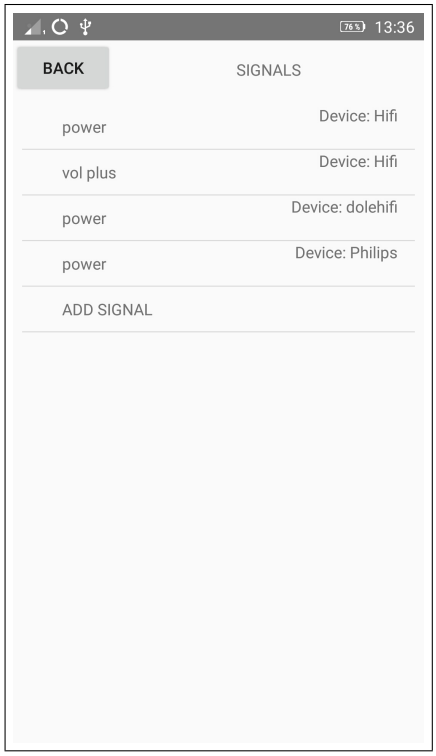
Ďalej si v práci tieto aktivity bližšie popíšeme a vysvetlíme ich podrobné funkcie.

3.2 Signál

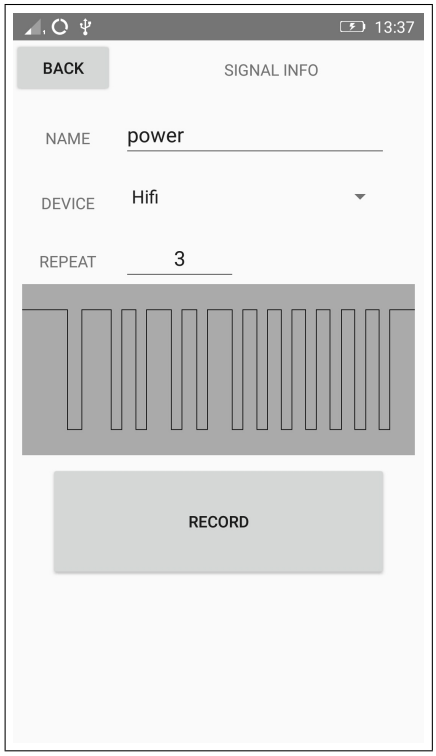
V tejto sekcii si bližšie povieme o tom ako sa signál v aplikácii nahráva, dekóduje a následne generuje. Tiež si popíšeme všetky funkcie, ktoré s jednotlivými signálmi môžeme robiť.

Do aktivity signálov sa dostaneme cez menu, ktoré sa zobrazilo po stlačení na nastavenia v hlavnej aktivite ovládača.

Tu sa zobrazí zoznam všetkých signálov (obrázok 3.1), ktoré už máme uložené (zo začiatku je prázdny). Tiež na konci tohto zoznamu je možnosť vytvorenia nového signálu, ktorý keď stlačíme sa dostaneme do ďalšej aktivity.



Obr. 3.1: Aktivita zoznamu signálov



Obr. 3.2: Aktivity samotného signálu

V tejto aktivite (obr 3.2) máme možnosť nastaviť meno signálu, zariadenie, pri ktorom sa tento signál bude používať (viac o zariadeniach si povieme v sekcii 3.3), a tiež koľkokrát sa má tento signál opakovať pri generovaní.

Pod všetkými nastaveniami je vizualizácia signálu, ktorá sa vyplní po úspešnom dekódovaní zvukovej stopy nastávajúceho po jeho nahratí. Nahrávanie je umožnené tlačidlom "RECORD" a proces, ktorý prebehne po stlačení tohto tlačidla si popíšeme podrobnejšie v podsekcii 3.2.1 a 3.2.2.

Ak si nevyberieme možnosť nového signálu, ale klikneme na signál v zozname, ktorý je už uložený, zobrazí sa rovnaká aktivita signálu, ale už vyplnená so všetkými para-

metrami, a tiež vizualizáciou nahraného signálu. Ďalej sa v tejto aktivite dá editovať signál.

3.2.1 Nahrávanie

Nahrávanie infračerveného svetla je umožnené modulom tejto práce. Tým, že modul je zapojený do audio jacku nášho zariadenia tak, ako aj generovanie, tiež nahrávanie je umožnené nahrávaním zvuku.

Každá zvuková karta má nejakú maximálnu vzorkovaciu frekvenciu (číslo v Hz), ktorá hovorí o tom, koľko bodov za sekundu dokáže zachytiť pri nahrávaní alebo použiť pri generovaní. Na to, aby sme získali maximálne vzorkovanie zariadenia, ktoré je pre nás potrebné z hľadiska, že pracujeme s veľmi vysokými frekvenciami, používame osobitne vytvorenú triedu v Jave.

Postupne skúša vzorkovacie frekvencie od najvyššej po najmenšiu. Ak vzorkovacia frekvencia nie je podporovaná zvukovou kartou, funkcia `getMinBufferSize` triedy `AudioRecord` vráti nulu. Naopak, ak podporovaná je, vráti minimálnu veľkosť vyrovnávacej pamäte pre to vzorkovanie, ktoré akurát skúšalo.

```
private int get_max_sample_rate(){
  int MAX_SAMPLE_RATE = 8000;
  for (int rate : new int[]
      {192000,176400,96000,88200,48000,44100,32000,22050,16000}) { // add the
      rates you wish to check against
    int bufferSize = AudioRecord.getMinBufferSize(rate,
        AudioFormat.CHANNEL_CONFIGURATION_DEFAULT,
        AudioFormat.ENCODING_PCM_16BIT);
    if (bufferSize > 0) {
      MAX_SAMPLE_RATE = rate;
      break;
    }
  }
  return MAX_SAMPLE_RATE;
}
```

Touto triedou bude rozšírená trieda nahrávania, a tiež generovania, aby používali rovnaké vzorkovanie. Je to dôležité kvôli formátu, ktorý sme si zvolili pre ukladanie signálu do databázy.

Pri samotnom nahrávaní signálu aplikácia využíva `AudioRecord` triedu Androidu,

ktorá umožňuje nahrávať zvuk. Metóda `startRecording` tejto triedy sa vykoná v osobitnom vlákne, aby nahrávanie bežalo mimo hlavného vlákna. Zastavenie nahrávania je vyriešené ukončením tohto vlákna a zastavením zapisovania do štruktúry `ArrayList<Short>`, ktorá slúži na ukladanie ešte nespracovaných dát, ktoré sme nahrali.

3.2.2 Dekódovanie

Nahraté nespracované dáta sa musia dekódovať. Pred tým, ako vysvetlíme ako dekódujeme signál, si povieme o formáte, v ktorom ho budeme ukladať. Tento formát nie je vytvorený podľa žiadneho štandardu, no je veľmi efektívny pre jednoduchosť následného spracovania.

Signál sa dekóduje a ukladá do poľa celých čísel. Absolútna hodnota čísel hovorí o dĺžke impulzov relatívne k maximálnej vzorkovacej frekvencii zariadenia. Ak by sme si chceli preložiť toto číslo do trvania impulzu v mikrosekundách, použili by sme nasledovný vzorec:

$$d = (|p|/MaxS) \cdot 10^5$$

Kde d je výsledná hodnota v mikrosekundách, p je číslo, ktoré chceme prekonvertovať do časového formátu a $MaxS$ je maximálna vzorkovacia frekvencia zvukovej karty zariadenia, ktorá bola zistená funkciou vysvetlenej v predchádzajúcej podsekcii 3.2.1.

Následne, ak toto číslo je kladné, znamená to, že v týchto častiach signálu bola generovaná prenosová frekvencia. Naopak, ak záporné, prenosová frekvencia generovaná nie je a na vstupe, bolo ticho.

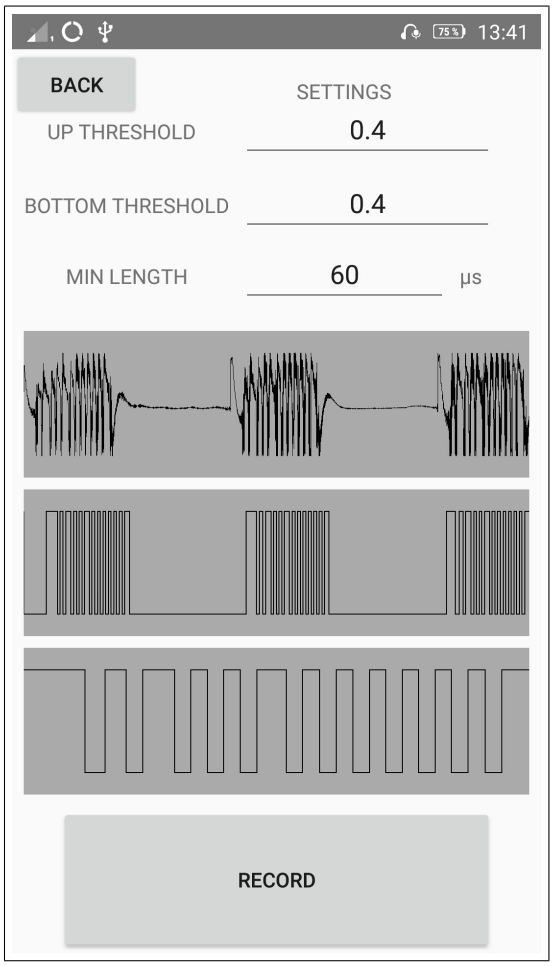
Samotné dekódovanie do tohto formátu z nespracovaných dát je uskutočnené sekvenčným prejdením cez nahrané dáta, kde sa sleduje ich hodnota (hustota týchto hodnôt je relatívna od použitej vzorkovacej frekvencie). Na nahrávke pri každej zmene toho, či sa signál generoval alebo nie, je možné odsledovať výkyv do kladnej alebo zápornej hodnoty. Ak tieto výkyvy presiahnu danú hodnotu, ktorú máme nastavenú ako prah citlivosti (threshold), zaznamenáme zmenu či sa prenosová frekvencia generovala alebo nie.

Taktiež aby sme filtrovali šum, ktorý je zachytený zvukovou kartou z modulu, použijeme podľa štandardu minimálnu dĺžku, pri ktorej tento prah citlivosti môžeme zaznamenať.

Citlivosť nahrávania (threshold), a tiež minimálnu dĺžku, pri ktorej má aplikácia reagovať na zmenu vieme nastaviť v globálnych nastaveniach. Vieme sa k nim dostať

z aktivity ovládača, stlačením na tlačidlo nastavení a následnom vybratí globálnych nastavení (Global settings).

V tejto aktivite, si tiež vieme vyskúšať či naše nastavenia so zariadením fungujú optimálne, prezretím vizualizácie dekódovaných dát.



Obr. 3.3: Ukážka globálnych nastavení signálu a vykreslenie dekódovania signálu

Ako aj na obrázku 3.3 vidno, jedno nahratie nám zachytilo ten istý signál viackrát. Toto je kvôli tomu, že jedno stlačenie na ovládači niekedy vygeneruje signál viackrát. Preto aj pri nastaveniach jednotlivého signálu, je možnosť nastavenie počtu opakovania signálu.

Toto využijeme, ako nahratie viacerých vzoriek toho istého signálu. Dekódovanie signálu akceptujeme ako úspešné len vtedy, ak tieto vzorky sú dekódované do rovnakej dĺžky. Teda zachytili sme všetky zmeny toho, či sa prenosová frekvencia generovala alebo bolo ticho rovnako.

Ak tieto dĺžky jednotlivých vzoriek sú rovnaké, jednotlivé hodnoty spriemerujeme a uložíme ako jeden signál (viď posledná vizualizácia na obr. 3.3).

3.2.3 Generovanie

Aký by mal byť signál na výstupe zariadenia, resp. vstupe modulu, sme si vysvetlili v kapitole hardvéru, sekcia 2.1. Každopádne, povieme si ako z dekódovaného signálu vygenerujeme signál, ktorým náš IR modul bude vedieť ovládať zariadenia.

Metóda `Generate` triedy `SignalGenerator` dostane ako vstup dekódovaný signál, ktorý je uložený v poli za sebou striedajúcich sa kladných a záporných čísel. Každé číslo reprezentuje ako dlho impulzy trvajú.

Trieda generátora v parametri konštruktora dostáva frekvenciu, horizontálne a vertikálne posunutie, ktoré následne berie do úvahy pri generovaní. Tieto hodnoty sa dajú tejto triede meniť. Bude to potrebné pri generovaní viacerých druhov signálov, podľa toho aké zariadenie budeme chcieť ovládať. Tu prichádza odpoveď na možnú otázku, odkiaľ dostaneme tieto hodnoty. Vysvetlíme si to v nasledujúcej podsekcii zariadení 3.3, ale zatiaľ sa budeme zaoberať s týmito hodnotami tak, že frekvencia je číslo prenosovej frekvencie v hertzoch (Hz), horizontálne posunutie je v číslo od 0 až po 360 v stupňoch a vertikálne posunutie bude rozšírenie kladných častí prenosnej frekvencie v percentách.

Na to aby sme mohli pustiť na výstup signál potrebujeme vygenerovať pole Bytov, ktorú potom trieda `AudioTrack` Androidu dokáže pochopiť a následne prehrať. Aby sme ju vedeli vytvoriť potrebujeme vedieť dĺžku celého signálu, čo vieme spraviť spočítaním absolútnej hodnoty celého pola dokopy.

```
numSamples = 0;
for(int i = 0; i < SignalinSampleLength.length; i++){
    numSamples += Math.abs(SignalinSampleLength[i]);
}
GeneratedSnd = new double[numSamples];
MonoGeneratedSnd = new byte[2 * numSamples];
StereoGeneratedSnd = new byte[4 * numSamples];
```

Keďže programátorsky nechceme priamo pracovať s Bytami vytvoríme si najprv jednoduché pole čísel, kde 1 pre nás bude znamenať maximálnu hranica zvuku, -1 minimálnu hranicu a nula bude reprezentovať generovanie ticha.

Postupne prechádzame polom, ktoré sme dostali na vstupe a sledujeme, akú dobu máme generovať prenosovú frekvenciu (kladné číslo v poli), a akú dobu ticho (záporné číslo v poli). Generujeme digitálny signál, aby skreslenie zvukovej karty bolo čo najmenšie. V praxi toto skreslenie bude vždy existovať, a to hlavne ak používame vysokú

frekvenciu. To, že zosilňovač nespína dostatočne rýchlo vieme kompenzovať dostatočným rozšírením kladných častí prenosovej frekvencie (v kapitole hardvéru sekcia 2.1 ako vertikálny posun).

Následne toto pole prerobíme na pole Bytov v jedno-kanálovom formáte (Mono) čo znamená, že ľavá aj pravá zvuková stopa je rovnaká.

Dvojkanálový (stereo) zvuk je len dvakrát jednokanálová (mono) zvuková stopa, kde dva Byty sú pravý kanál a nasledovné dva byty sú ľavý kanál. Takto vytvoríme náš horizontálny posun. Zoberieme si našu vygenerovanú mono stopu a jednoducho ju len posunieme v druhom kanáli, o taký fázový posun, aký potrebujeme.
Nasledovný Java kód zobrazuje proces konvertovania:

```
int idx = 0;
for (double dVal : GeneratedSnd) {
    short val = (short) (dVal * 32767);
    MonoGeneratedSnd[idx++] = (byte) (val & 0x00ff);
    MonoGeneratedSnd[idx++] = (byte) ((val & 0xff00) >>> 8);
}

int HorizontalOffset = (int)((this.HorizontalOffset /360.0) * (sampleRate/
    Frequency) * 2.0);

int j = 0;
for(int i = HorizontalOffset; i < MonoGeneratedSnd.length-2; i+=2){
    StereoGeneratedSnd[j++] = MonoGeneratedSnd[i];
    StereoGeneratedSnd[j++] = MonoGeneratedSnd[i+1];
    StereoGeneratedSnd[j++] = MonoGeneratedSnd[(i - HorizontalOffset)];
    StereoGeneratedSnd[j++] = MonoGeneratedSnd[(i+1) - HorizontalOffset];
}
```

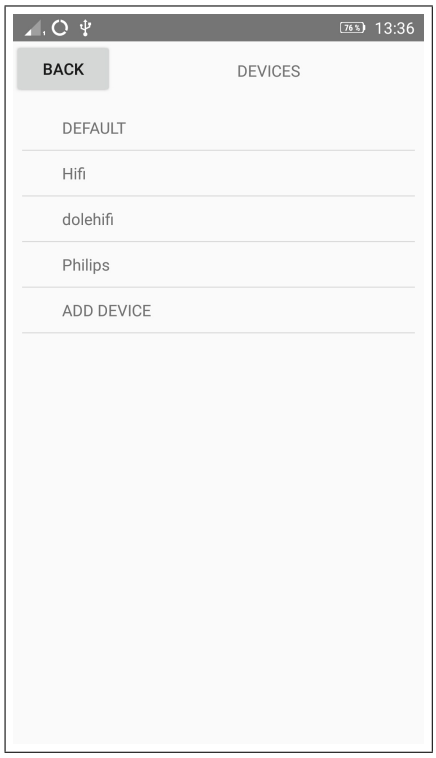
Vytvorená dvojkanálová (stereo) zvuková stopa sa následne zapíše metódou `Write` a prehrá metódou `Play` triedy `AudioTrack` Androidu.

3.3 Zariadenie

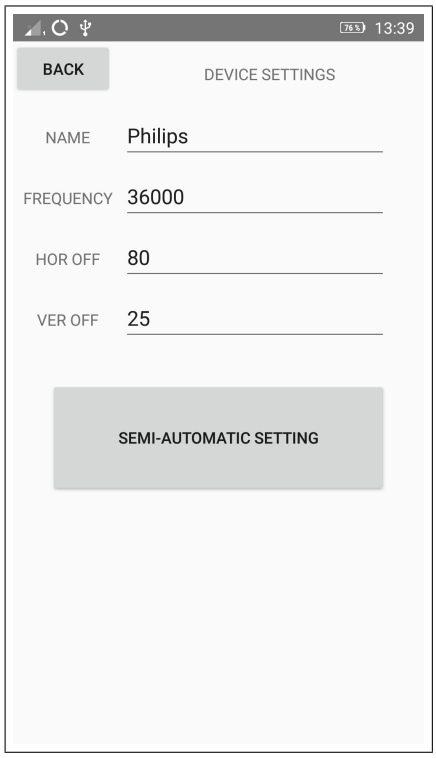
Ako sme si už v sekcii signálu 3.2 spomínali, každý signál môže patriť jednému zariadeniu. Na to, aby sme si vedeli vytvoriť nejaké zariadenie a nastaviť jeho vlastnosti musíme kliknúť na tlačidlo nastavení v aktivite ovládača a vybrať si zariadenia (Devices).

Zobrazí si nám podobný zoznam zariadení ako pri signáloch s tým rozdielom, že

tam nájdeme základné (“DEFAULT”) nastavenie, ktoré má v databáze ID 1. Dá sa meniť, no nedá sa nijako vymazať. Každý nový signál dostáva toto zariadenie ako základ nastavení pri generovaní.



Obr. 3.4: Aktivita zoznamu zariadení



Obr. 3.5: Aktivita samotného zariadenia

Ak si vytvoríme nové zariadenie alebo otvoríme už existujúce, vieme mu nastaviť meno, a tiež už tri známe nastavenia, ktoré sú potrebné nastaviť pre optimálne diaľkové ovládanie zariadení. Týmito nastaveniami sú frekvencia, horizontálny a vertikálny posun (vertical offset).

Tieto parametre si vieme nastaviť aj manuálne, ale nijako nevieme zistiť či sme ich nastavili správne. Jedine odskúšaním vygenerovania nejakého signálu, ktoré tomuto zariadeniu patrí. Tento proces môže byť zdĺhavý a nepríjemný. Preto sa vytvorilo semi-automatické nastavenie zariadenia, ktoré si rozoberieme v nasledujúcej podsekcii.

Poznámka: Tieto parametre zo skenovaného signálu nevieme zistiť lebo už spomínané limitácie zvukovej karty nám to nedovolia.

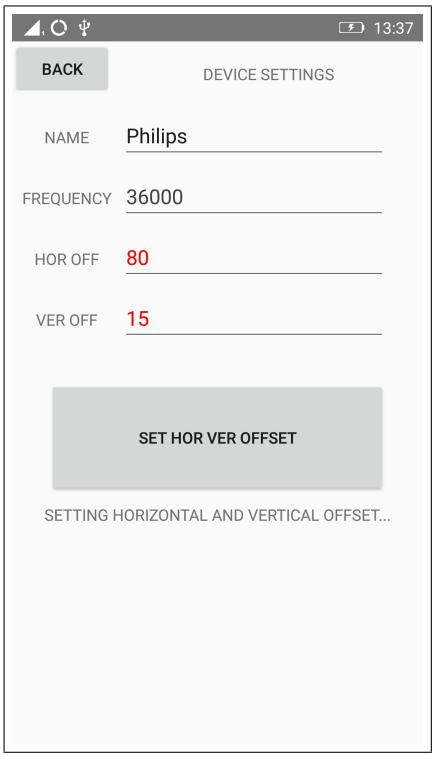
3.3.1 Semi-automatické nastavenie

Každé zariadenie, ktoré už má priradený nejaký signál (v nastaveniach signálu vybratie tohto zariadenia), má možnosť si semi-automaticky nastaviť tieto parametre.

Semi-automatické z toho hľadiska, že naša aplikácia nemá ako rozoznať či toto nastavenie je správne. Na to je potrebný človek, ktorý to aplikácii naznačí.

Tento proces automatizácie nastavení je pre zariadenie možné pustiť len vtedy, ak už má nejaký signál priradený. Ak má, po stlačení tlačidla "SEMI-AUTOMATIC SETTING" sa objaví zoznam signálov, ktoré sú tomuto zariadeniu priradené. Po vybratí jedného zo signálov sa začne proces nastavovania, a to práve tým signálom čo sme vybrali.

Začne sa najprv nastavovanie frekvencie. Aplikácia emituje signál pod rôznymi horizontálnymi a vertikálnymi nastaveniami pre skúšanú frekvenciu. Ak užívateľ uvidí nejakú reakciu na ovládanom zariadení, stlačí tlačidlo. To pre aplikáciu znamená, že táto frekvencia, približne pod týmito nastaveniami horizontálneho a vertikálneho posunu je validná. Ďalej začne proces podrobnejšieho skúšania týchto posunov a to tak, že sa vráti do toho okolia kde to fungovalo. Následné skúša tieto nastavenia podrobnejšie a to v sekundových intervaloch, aby užívateľ mal viac času reagovať na to, aby stlačil tlačidlo, že zariadenie na signál zareagovalo.



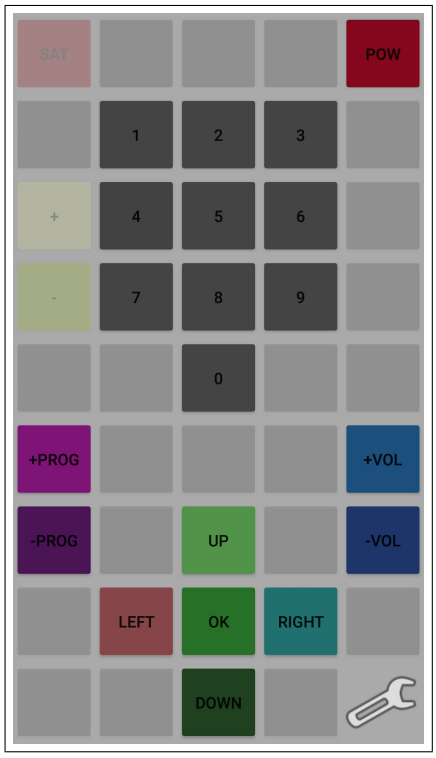
Obr. 3.6: Ukážka semi-automatického nastavenia v procese nastavovania posunov

Po skončení alebo aj prerušení semi-automatického nastavovania sa tieto parametre zapíšu do databázy.

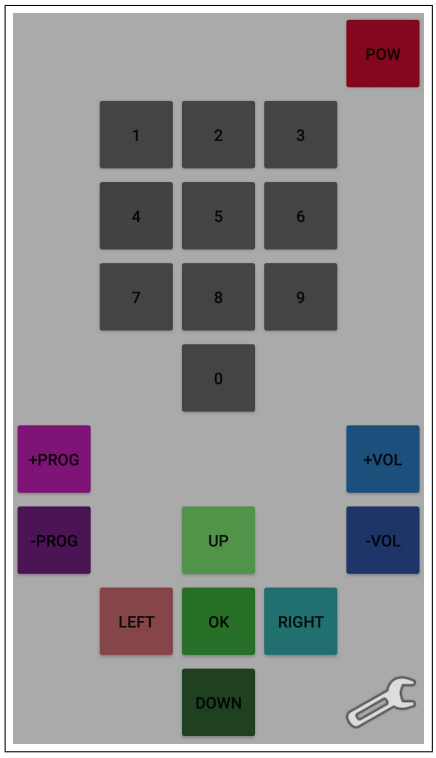
Pri testovaní už nebolo potrebné žiadne ďalšie vylepšovanie týchto parametrov, ale je možné ich stále manuálne editovať alebo aj spustiť semi-automatické nastavenie znova.

3.4 Ovládač

Táto sekcia sa bude zaoberať s hlavnou aktivitou ovládača, ktorá spája všetky hore uvedené funkcie do užívateľského rozhrania, z ktorého je možné ovládať zariadenia.



Obr. 3.7: Konfiguračný mód ovládača



Obr. 3.8: Uzamknutá aktivita ovládača

Pri novej inštalácii aplikácie je táto aktivita prázdna ale veľmi intuitívne do nej vieme pridať nové tlačidlá. Po stlačení na tlačidlo nastavení a vybratí možnosti "Manage buttons" sa na celej obrazovke zobrazí mriežka tlačidiel, ktoré si môžeme nastaviť (obrázok 3.7). Ak skončíme s nastavovaním tlačidiel, rovnakým spôsobom ako sme ich odomkli, tlačidlá uzamkneme. Tým sa premenia na tlačidlo ovládača, ktorým vieme ovládať naše požadované zariadenia (obrázok 3.8).

Kým ale máme manažovanie tlačidiel zapnuté, po kliknutí jedného z tlačidiel sa dostaneme do rozhrania, kde si vieme tlačidlo nastaviť.

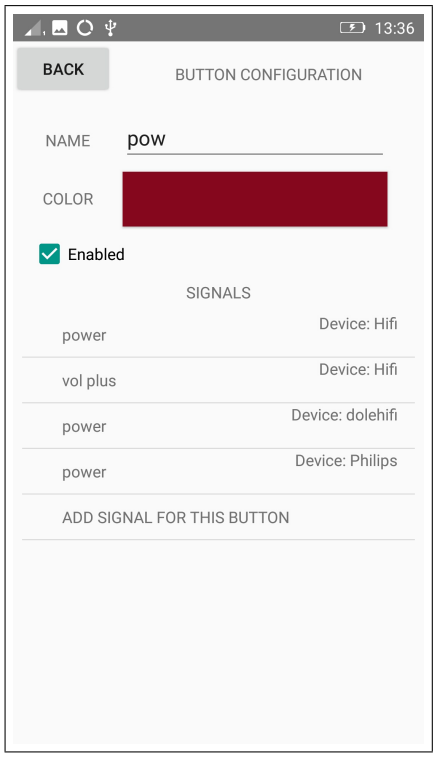
3.4.1 Nastavenie tlačidla

Každé tlačidlo má nasledovné vlastnosti: meno tlačidla, aktivovanosť, farba a zoznam signálov.

Postupne si tieto vlastnosti rozoberieme.

- Meno tlačidla: text, ktorý sa zobrazuje na aktivite ovládača kvôli rozoznaniu od ostatných tlačidiel, ak ich je viac.

- Aktivovanosť: vlastnosť, ktorá môže mať dva stavy. Tlačidlo je aktivované, to znamená, že po uzamknutí editovania tlačidiel na hlavnej aktivite ovládača tlačidlo bude viditeľné a klikateľné. Opačne, ak aktivované nie je, všetky nastavenia ostávajú uložené ale tlačidlo po uzamknutí nebude viditeľné a nebude reagovať na interakcie s užívateľom.
- Farba tlačidla: parameter, ktorý sa tiež zobrazí ako farba tlačidla v aktivite ovládača. Táto vlastnosť existuje kvôli väčšej možnosti prispôsobenia. Dialóg, v ktorom sa farba vyberá si rozoberieme podrobnejšie v tejto podsekcii neskôr.
- Zoznám signálov: umožňuje pridávať nové a sledovať priradené signály pre toto tlačidlo.



Obr. 3.9: Aktivita nastavenia tlačidiel

Obr. 3.10: HSV farebný dialóg

HSV farebný dialóg

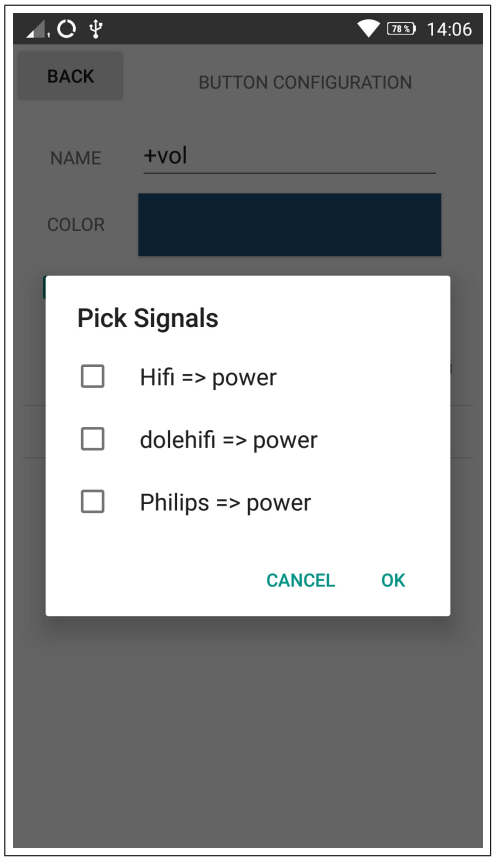
Keďže Android nemá v sebe vybudovaný dialóg na vyberanie farieb, použil sa v práci dialóg zo stránky Buzzing Android [1], ktorý je znázornený na obrázku 3.10. Tento dialóg sa volá HSV kvôli jeho možnosti výberu farby. H je odtieň (hue), S sýtosť (saturation) a V hodnota (value) farby (čím nižšia o to je farba tmavšia).

Do dialógu sa pridalo jedine zobrazenie hexadecimálneho kódu vybranej farby, ak by si niekto chcel o niečo presnejšie nastaviť vyberanú farbu.

Macro signál

To, že aké funkcie ktorého zariadenia budeme ovládať týmto tlačidlom, vidíme v zozname, ktorý je v aktivite posledný z hora. Tento zoznam môže obsahovať viac signálov ako len jeden. To umožňuje, aby sme s jedným tlačidlom dokázali ovládať viac funkcii jedného zariadenia alebo viac zariadení sekvenčne naraz. Pod pojmom sekvenčne naraz sa myslí, že sa vygenerujú všetky signály tohto zoznamu jeden za druhým, v tom poradí ako ich v zozname vidíme.

Na to aby sme pridali signál pre tlačidlo, si v zozname vyberieme poslednú možnosť "ADD SIGNAL FOR THIS BUTTON". Ak naň klikneme, objaví sa dialóg signálov (obrázok 3.11), ktoré pre toto tlačidlo nie sú priradené. To eliminuje možnosť generovať signál viackrát pre jedno tlačidlo. Na to máme možnosť opakovania v nastaveniach signálu.



Obr. 3.11: Dialóg vyberania signálov pre tlačidlo

3.4.2 Ovládanie

Samotné ovládanie zariadení ako už vieme, je umožnené cez uzamknutú aktivitu ovládača (obrázok 3.8).

Keď stlačíme tlačidlo raz, vygeneruje sa sekvencia signálov, ktoré máme nastavené v zozname signálov stlačeného tlačidla. Rovnako, ak tlačidlo držíme, ako by sme očakávali, táto postupnosť signálov sa bude generovať v cykle až kým ho budeme držať (v aplikácii je limitácia 50 opakovaní aby sme sa vyhli nekonečným cyklom).

O to aby sa signál opakoval toľko krát ako to máme nastavené, a v prípade makro tlačidla signále pospájali, sa nám stará naprogramovaná trieda `SignalComposer`. Metóda `Compose` tejto triedy dostane ako parameter pole signálov, kde každý signál môže patriť aj inému zariadeniu. Vytvorí jednu súvislú zvukovú stopu týchto signálov, a tým, že priamo využíva triedu `SignalGenerator` ju metódou `Play` vie aj prehrať.

3.5 Databáza

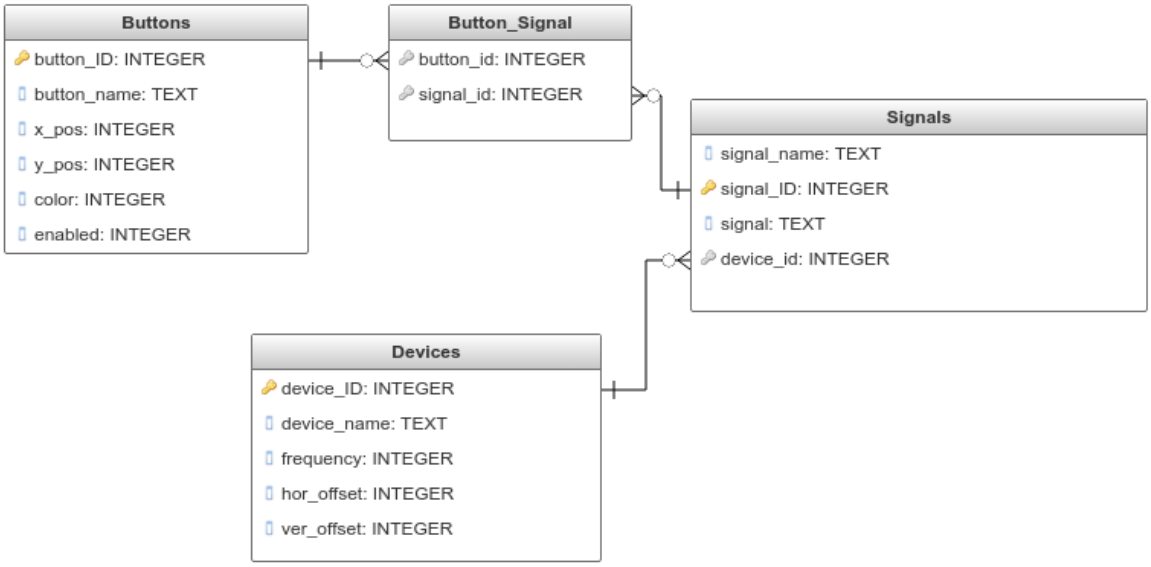
Aplikácia využíva SQLite databázu na ukladanie všetkých nastavení, čo sa týka tlačidiel, signálov a zariadení.

Aplikácia, tiež využíva triedu `SharedPreferences` Androidu, kde ukladáme len niektoré jednoduché premenné, ktoré si potrebujeme pamätať.

3.5.1 SQLite

Android podporuje pre svoje aplikácie SQLite databázu, ktorú sme aj pri našej aplikácii využili. Obsahuje štyri tabuľky:

- Tlačidlá: tu sa ukladajú všetky parametre tlačidla ako je ID, meno, ktoré sa zobrazuje na aktivite ovládača, jeho (x,y) pozícia v mriežke, farba a či je aktívny.
- Signál: Všetky signály sú uložené v tejto tabuľke databázy. ID, meno signálu, samotný signál, ktorý je uložený ako reťazec znakov kvôli limitácii SQLite, koľkokrát sa má tento signál pri generovaní opakovať, a tiež ku ktorému zariadeniu patrí.
- Zariadenia: V tejto tabuľke sú uložené parametre nastavení pre jednotlivé zariadenia, tak isto ako aj jeho ID a Meno.
- Tlačidlo-Signál: Táto tabuľka má uložené údaje o tom, ktoré signále má tlačidlo v aktivite ovládača pridelené.



Obr. 3.12: ERM diagram SQLite databázy

Pri prvej inštalácii sa do databázy vytvoria všetky tlačidlá dynamicky od toho aké proporcie má naše zariadenie. To, že máme päť tlačidiel cez šírku aktivity je fixne dané a od toho sa ďalej vypočíta koľko tlačidiel sa na výšku zmestí.

Tiež sa vytvára aj "DEFAULT" zariadenie, ktoré sme spomínali v sekcii 3.3.

Celá komunikácia s databázou je naprogramovaná v triede `DBHandler`. Aktivity z nej vedia vyberať informácie len cez MVC kontrolér (Model-View Controller), ktorý je návrhovým vzorom odporúčaným pre Android aplikácie.

Kontrolér našej aplikácie je naprogramovaný v triede `MVC_Controller`

3.5.2 Shared Preferences

V Aplikácii používame tiež triedu `SharedPreferences` Androidu. Slúži nám len na ukladanie jednoduchých premenných ako reťazec znakov alebo celé číslo.

Táto trieda Androidu ukladá údaje do XML súboru pod kľúčom, aký si zvolíme. Následne túto hodnotu vieme dostať pomocou tohto kľúča.

Naša aplikácia do `SharedPreferences` ukladá len tri údaje, ktoré sa dajú meniť v globálnych nastaveniach. Presnejšie nastavenia dekódovania signálu.

Komunikácia je naprogramovaná v triede `SPHandler` našej aplikácie.

Záver

Infračervené ovládače sa dobou pomaly míňajú do straty. Stále sa ale využívajú a budú využívať na ovládanie lacnejších zariadení, pri ktorých je vybudovanie komplexnejšieho bezdrôtového ovládania drahé.

Aplikácia v práci pri testovaní fungovala s viacerým ovládačmi. Našli sa aj ovládače, pri ktorých modul nevedel poriadne naskenovať signál na to, aby ho aplikácia vedela dekódovať.

Tento problém spôsobil prijímajúca dióda nášho modulu. Nie každý infračervený ovládač funguje s rovnakou dĺžkou infračerveného spektra. Preto aj náš modul, ktorý má vybudovanú len jednu prijímajúcu diódu, vedel skenovať infračervené spektrum, na ktoré je dióda vyrobená. Aj keď prijímajúca dióda vedela naskenovať nejakú stopu, kvôli inkonzistencie zvukových stôp jednotlivých ovládačov, nebolo možné vytvoriť lepší algoritmus na dekódovanie.

Tento problém, by možno vedela vyriešiť nová generácia modulu, ktorá by fungovala cez USB port zariadenia. Takýto druh modulu, by vedel eliminovať viac problémov, ktoré sme objavili počas práce. Napríklad generovanie nedostatočne vysokej frekvencie, zoslabeného signálu z audio jack portu, externého napájania, artefaktu skenovania atď.

Dôvod, prečo sa nevytvoril hardvér, ktorý by pracoval cez USB, je nutnosť ovládať programovanie ovládača (driver) pre tento modul, tiež aj amatérskych hardvérových schopností, a tiež nedostatku času pri tvorbe tejto práce.

Návrhom vylepšenia takéhoto modulu, by možno bolo vytvorenie ovládača, ktorý by fungoval s triedou `ConsumerIrManager` (Android, API level 19), ktorý umožňuje generovanie, a tiež skenovanie infračerveného signálu.

Ďalším vylepšením, by mohla byť možnosť integrovania databázy kódov ovládača, ktorú sme vynechali kvôli možnosti skenovania signálu.

Literatúra

[1] Buzzing Android, HSV Color Picker Dialog. `https://www.buzzingandroid.com/2012/11/hsv-color-picker-dialog/`. HSV farebný dialóg.

[2] IR Droid. `http://www.irdroid.com/`. Existujúce riešenie práce.

[3] Eearl Boysen Gordon McComb. Electronics for DUMMIES. 2005.

[4] Simon Kendal. Object Oriented Programing using Java, 2009.

[5] Reto Meier. Professional Android 4 Application Development. 2012.

[6] ST Microelectronics. *AN3174 Application note*. 2012. Implementing receivers for infrared remote control protocols using STM32F10xxx microcontrollers.

[7] Bob Zulinski Associate Professor of Electrical Engineering Michigan Technological University. *Introduction to Electronics*.