UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Knižnica pre mobilné zariadenia zabezpečujúca dátový prenos cez audio port

Bakalárska práca

2013

Ján Tančibok

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Knižnica pre mobilné zariadenia zabezpečujúca dátový prenos cez audio port

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 2508 informatika
Školiace pracovisko: Katedra Informatiky
Školiteľ: RNDr. Michal Forišek, PhD.

Bratislava, 2013
Ján Tančibok

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ján Tančibok
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Knižnica pre mobilné zariadenia zabezpečujúca dátový prenos cez audio port

Cieľ: Prvým cieľom práce je preskúmať existujúce riešenia implementácie obojsmernej komunikácie medzi mobilným zariadením a externým príslušenstvom cez audio port. Výstupom práce by mala byť nová vlastná implementácia takejto knižnice pre platformu Android. Súčasťou práce by mala byť analýza vhodnosti navrhovaného riešenia a tiež dôraz na univerzálnosť, prípadne ľahkú rozšíriteľnosť výslednej knižnice.

Vedúci: RNDr. Michal Forišek, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 29.10.2012

Dátum schválenia: 30.10.2012

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poď akovanie

Rád by som týmto poď akoval môjmu školiteľ ovi RNDr. Michalovi Foriškovi, PhD. za jeho pomoc a cenné rady. Autor myšlienky a tiež konštruktér HART modemu je Ing. Maxim Mizov, zo SAV Košice. Ďakujem vám obom.

Abstract

The aim of this work is to implement a library providing two-way data communication over audio port. The benefits of this solution is possibility of powering device and simple hardware construction.

We communicate with a modem supporting the HART protocol, which can be built into a variety of sensors and other devices. In this work, we are focused to the Android platform for its diffusion and wide functionality.

Klíčové slova: serial port, audio port, jack, Android, library, HART, UART, FSK, modulator, demodulator, software radio

Abstrakt

Cieľom práce je implementovať knižnicu poskytujúcu obojsmernú dátovú komunikáciu cez audio port. Prínosy tohoto riešenia sú zabezpečenie napájania prístroja a jednoduchá výroba hardvéru.

V tejto práci sa zmeriavame na platformu Android, pre jej rozšírenosť a bohatú funkcionalitu. Komunikovať budeme s modemom podporujúcim HART protokol, ktorý sa dá zabudovať do rôznych senzorov a zariadení.

Kľúčové slová: serial port, audio port, jack, Android, knižnica, HART, UART, FSK, modulácia, demodulácia, softvérové rádio

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Komunikácia je a vždy bola veľmi dôležitá. Existuje veľa možných spôsobov ako komunikovať. Pred začatím výmeny informácií si treba vybrať vhodné komunikačné médium. Pri dobrej reprezentácií dát, môžeme vytvoriť spojenie takmer po hociktorom médiu aj poštovými holubmi. Médiá poznáme káblové, bezdrôtové, internet, dajú sa deliť aj podľa signálu na digitálne a audio.

Jedným z medií sú aj audio vlny, toto médium je už staršie, je vytláčané postupnou digitalizáciou, ale stále existuje množstvo zariadení podporujúcich tento druh komunikácie, ide najmä o hlasové a video prenosy. Výhody oproti digitálnemu prenosu sú jednoduchosť a cena zriadenia, netreba sa zapodievať synchronizáciou systému a chybové bity sa neakumulujú v celom prenosovom systéme. [1, p. 269] Počítače a aj mobilné zariadenia však pracujú s digitálnymi dátami, prevod z digitálnych dát na analógové sa nazýva modulácia a opačný postup je demodulácia. Na túto konverziu nám slúži zariadenie všeobecne nazývané modem, názov je kombinácia **mod**ulácie a **dem**odulácie.[1, p. 392] Keďže ľudské uši aj oči fungujú na analógovom princípe, analógový druh komunikácie bude ešte dlho podporovaný.

Na rozdiel od hardvéru, kde je množstvo rôznych modemov, v Android zariadeniach nie je žiaden natívne zabudovaný modem, cieľom práce je softvérovo suplovať modem, a to implementáciou knižnice podporujúcej moduláciu aj demoduláciu digitálneho a analógového signálu. Táto knižnica by mala byť univerzálna, ľahko implementovateľná, prípadne rozšíriteľná. Tým pádom bude zabezpečená obojstranná komunikácia s externými zariadeniami a zariadeniami s platformou Android. Praktické využitie je najmä v zapájaní rôzneho hardvéru na mobilné zariadenia, predpokladá sa využitie pre rôznu meraciu techniku, senzory.

Rozhodol som sa prispieť do tejto oblasti vývoja na Android, lebo podobná komplexná a open-source práca ešte neexistuje a pritom je potrebná a užitočná. Doterajšie existujúce riešenia neposkytujú základnú funkcionalitu na implementovanie dátovej komunikácie, napríklad komunikácia s meracími zariadeniami musí byť obojsmerná, zariadenia musia namerané dáta aj posielať, nie len prijímať kontrolné signály. Niektoré riešenia sa nedajú použiť v

projektoch pre ich uzavretosť, alebo špecifikáciu len na jednu platformu a určitý typ hardvéru, napríklad špecifikácia len na Andrulino. Táto tematika je bližšie rozobraná v prvej kapitole.

V prvej kapitole sa venujem prehľadu medzi existujúcimi riešeniami obojsmernej dátovej komunikácie cez audio port. Sú tu zhrnuté základné vlastnosti existujúcich riešení a opísané nedostatky, ktoré ma motivovali začať túto prácu a ktoré by táto práca mala vyriešiť. Cieľom kapitoly je ukázať prečo je treba túto knižnicu a jej miesto vo vývoji aplikácií pre Android.

V druhej kapitole sa nachádza dokumentácia knižnice, každý projekt by mal mať podrobnú a prehľadnú dokumentáciu, najmä, ak ide o knižnicu, kde sa očakáva používanie viacerými programátormi.

Tretia kapitola je venovaná metodike riešenia problému, je tu krátky úvod do prostredia Android, pre ktoré je knižnica implementovaná. Väčší dôraz je kladený na teóriu softvérového rádia a základné princípy práce z audio-signálom, modulácia3.3, demodulácia3.4 a základné filtrovanie, aj keď v príslušných podkapitolách uvádzame aj naše implementované riešenia.

Verím, že táto práca pomôže jednoducho implementovať takýto typ komunikácie, širšiemu poľu vývojárov, a uľahčí im prácu s viacerými I/O zariadeniami.

Kapitola 1

Prehľad existujúcich riešení

Kapitola ukazuje súčasný stav riešení dátového prenosu cez audio jack. Vysvetľuje výber tohto spôsobu, uvádza príklady z praxe a motiváciu použitia audio sériového portu. V tejto kapitole ukazujeme čiastočné, ale aj plne funkčné príklady dátových prenosov realizovaných cez audio jack.

Aj napriek rozmachu android platformy, sme nenašli žiadne univerzálne a ľahko použiteľné riešenie audio-digitálnej komunikácie. Pár čiastočných riešení nemalo zrozumiteľnú dokumentáciu, ani žiadne príklady použitia kódu. Vyhovujúca a komplexná implementácia, napríklad aj vo forme knižnice, zatiaľ chýba. Preto sme sa rozhodli takúto knižnicu naprogramovať.

1.1 Robots everywhere

Na stránke[2] je voľne k stiahnutiu aplikácia, aj zdrojové kódy. Aplikácia je však len na audio out, zaujímavejšie bude urobiť práve audio in. Po hlbšom preskúmaní a otestovaní s modemom DS8500 HART, sme zistili, že stiahnutý program nefunguje, ani pri rôznych nastaveniach modulácie. Z kódu sa dá pochopiť serializácia dát aj vytvorenie packetov, ale audio vlna nie je vytváraná typickým spôsobom. Vlna vytvorená programom nemá tvar sínusovky, možno špeciálny hardvér ktorého schémy sú tiež na stránke by tieto dáta vedel demodulovať, DS8500 HART modem to ani po úprave parametrov na HART protokol nezvládol, z čoho je jasné, že toto riešenie nie je použiteľné univerzálne.

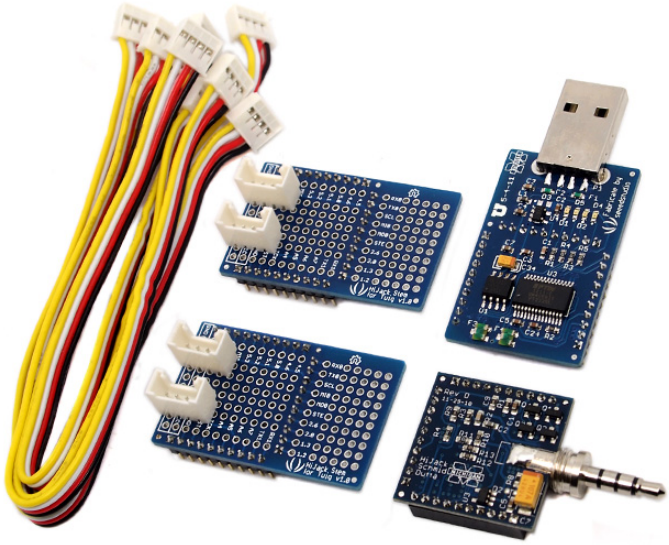
Obr. 1.1: Porovnanie Robots everywhere modulácie hore a klasickej FSK modulácie dole.

1.2 HiJack

HiJack je projekt z Michiganskej univerzity[3]. Cieľom je vytvoriť unifikované senzory komunikujúce s mobilnými zariadeniami. Zatiaľ je riešenie dostupné len na iOS, v budúcnosti plánujú rozšírenie aj na Android a Windows. Na moduláciu a demoduláciu používajú mikrokontroler, čo má viac nevýhod, treba ho naprogramovať a je drahší ako modemový čip, predaj hardvéru je zabezpečený cez portál seedstudio[4] kde je možné si kúpiť celý kit za 79$. Hlavná nevýhoda je drahšia cena, veľa ľudí je schopných si senzory vyrobiť svojpomocne, schémy sa dajú nájsť na internete, nepotrebujú k tomu drahý kit. Za tie peniaze by už mohli mať pekné komerčné riešenie. Kto má však rád skladačky a iOS, toto pre neho môže byť dobrá cesta .

Napájanie senzorov je zabezpečené cez, HiJack energy harvester, ktorý poskytuje zátaž 7.4 mW s 47% efektivitou prevodu na výkon. A to tak, že je pustený 22 kHz tón z jedného audio kanála na mobilnom telefóne, cenu elektronických komponentov na harvester uvádzajú na 2.34$, ale až pri 10 000 kusoch. Skúmajú aj iné prístupy vedúce k vyššej prevodovej efektivite, na stránke majú uvedené maximálne napätie 3,3V. Ak by sa podarilo zlepšiť získavanie výkonu, maximálne napätie by mohlo byť aj viac ako 5V.

Obr. 1.2: Unifikovaný tvar HiJack senzoru.[3]

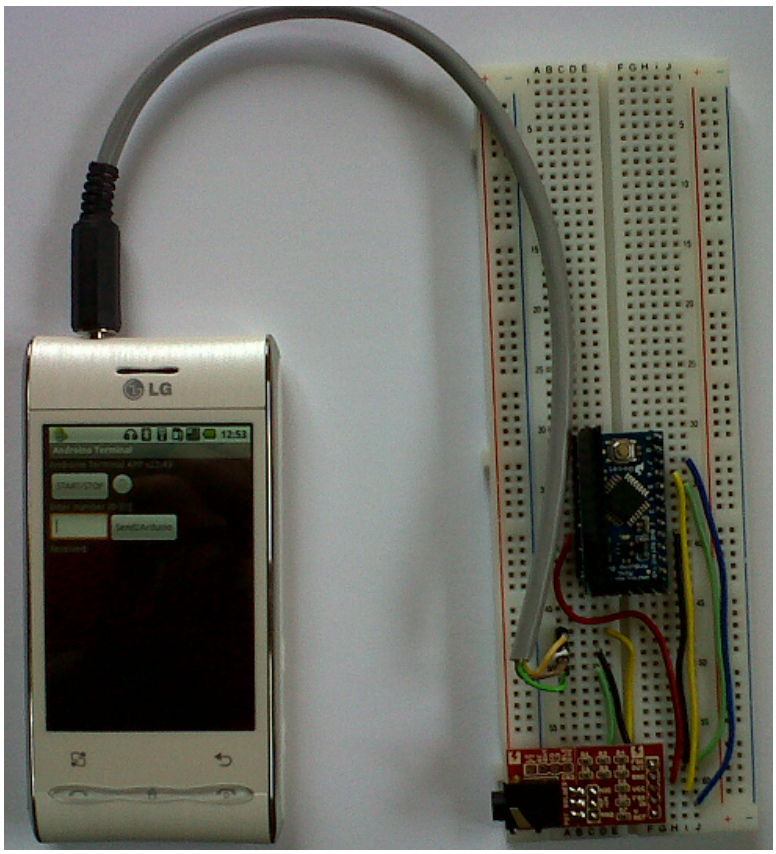


Obr. 1.3: HiJack sa predáva ako sada, HiJack doska, programátor a dve dosky na budovanie senzorov.[4]

1.3 Androino

Androino je aj podľa názvu zameraný na platformu Arduino a operačný systém Android. Arduino je celosvetovo rozšírená skladačka, existuje množstvo návodov, bez problémov sa dá kúpiť aj na Slovensku, je možné ju použiť v rôznych aplikáciach. Keby sme chceli spraviť projekt používajúci iba sériový port, potrebovali by sme Arduino dosku v cene od 20€ do 60€, Arduino modem na iOS a Android za 10€ plus káble a samotný projekt/senzor. Z toho vyplýva, že je zbytočné používať Arduino len kvôli audio sériovému portu. Ak však už máte Arduino, tak tento projekt je jednoduchým riešením ako získať audio sériový port. Na

stránke projektu[5] je návod na zapojenie hardvéru, aj stiahnuteľná aplikácia na Android. Aplikácia implementuje obojsmernú komunikáciu a podporuje aj jednoduchú detekciu chýb s automatickými dopytmi na opakovanie chybnej správy. Aplikácia však posiela a prijíma len integer v intervale od 0 do 31. Skladačky sa pre vysokú jednotkovú cenu vôbec neoplatia pri komerčnej sériovej výrobe, môžu súžiť len ako prototypy.



Obr. 1.4: Fungujúce zapojenie Android aplikácia, Arduino doska a modem.[5]

1.4 HMB-TEC a DO-RA

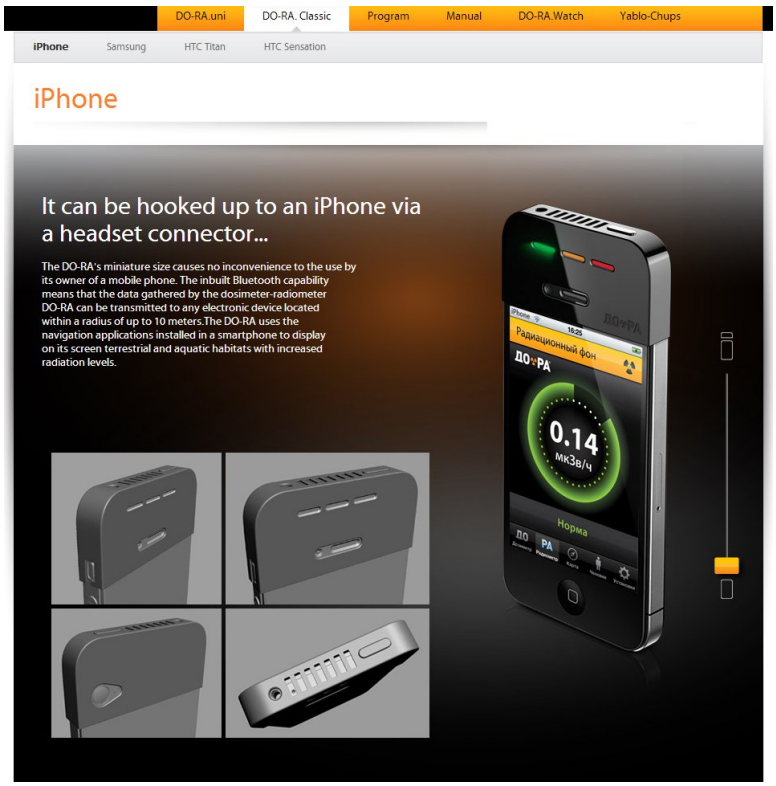
Komerčne zamerané projekty, predávajú rôzne senzory s pripojením priamo na mobil, HMB-TEC[6] je nemecká firma podporujúca len iOS, majú rôzne senzory, ktoré sú robustné, iPhone sa do nich vkladá, celkovo to vyzerá ako dva iPhony na sebe. Rádiometer ponúkajú za 148€.

DO-RA[7] je ruská firma. Jej senzory merajú radiáciu a majú rôzne modely aj na Android zariadenia, dizajnovo sú veľmi pekne spracované senzor po pripojení splynie s telom telefónu, pôsobí ako prirodzená súčasť telefónu. iPhone model obsahuje aj stavové led svetielka, táto firma si naozaj dala záležať na designe senzorov, uviedla na trh aj špeciálne senzory pre

iOS v tvare jabĺk. Aplikácia spolupracujúca s gps modulom telefónu, je voľne stiahnuteľná z Google Play alebo iTunes, avšak senzory sú dostupné len na pred-objednávku.

Ďalší úspešný projekt je čítačka kariet na mobilné zariadenia SquareUp[8], bohužiaľ služba je dostupná len v USA.

Úspešnosť komerčných riešení dokazuje, že komunikácia cez audio port je vhodná na senzorové typy projektov, tu je dôležitý demodulačný aspekt a jednoduchosť napájania. Modulácia sa viac využije v posielaní ovládacích sekvencií do rôznych zariadení.



Obr. 1.5: Dozimeter DO-RA na iPhone.[7]

Obr. 1.6: Yablo-chups dizajnové merače rádioaktivity. [7]

Obr. 1.7: Rádiometer od firmy HMB-TEC.[6]

1.5 Iné riešenia

Komunikácia s vonkajším svetom sa na mobilných platformách dá realizovať rôznymi metódami. WiFi, Bluetooth, NFC, USB a audio port. WiFi, NFC a Bluetooth majú výhodu, že prenos sa deje bezdrôtovo, ale cez USB a audio port sa dá napájať pripojené zariadenie. Každá z uvedených metód má rôzne výhody a nevýhody.

Z bezdrôtových technológií je na komunikáciu medzi externým hardvérom a mobilom najpoužívanejší Bluetooth, Bluetooth vysielače sú už celkom lacné, ceny sa pohybujú okolo 10$. Implementácia pre Android platformu je veľmi dobre zdokumentovaná. Na strane hardvéru, na príklad senzoru, sa musí použiť mikrokontroler, čo je nevýhoda pri malých a lacných senzoroch. Zaujímavá je aj technológia NFC (Near field communication), ktorú ale zatiaľ podporuje len pár drahých zariadení.

Na pripojenia hardvéru k mobilu cez USB, treba aby bol hardvér externe napájaný, tzn. mód príslušenstva. Ak by android zariadenie podporovalo USB host mód, môže mobilné zariadenie napájať pripojený hardvér. USB host mód, ale v súčastnosti podporujú len high-end tablety a high-end mobily. Oba módy sú dobre zdokumentované aj na oficiálnych stránkach.[9]

Cez audio jack sa dá napájať pripojený hardvér, nemá implementačné obmedzenia. Pripojením špeciálneho hardvéru sa dá urobiť root alebo jailbreak, získať root práva. Dôležitá výhoda je aj rozšírenosť audio jackov, audio port má každé android zariadenie, kompatibilné sú aj staršie verzie Androidu, čo je dôležité, lebo aj dnes sa ešte predávajú. Hlavné nevýhody sú pomalé modemové spojenie a na niektoré projekty nie je vhodné používať káblové riešenia. Aj napriek rýchlosti, môže tento druh komunikácie konkurovať aj moderným technológiám.

Kapitola 2

Dokumentácia

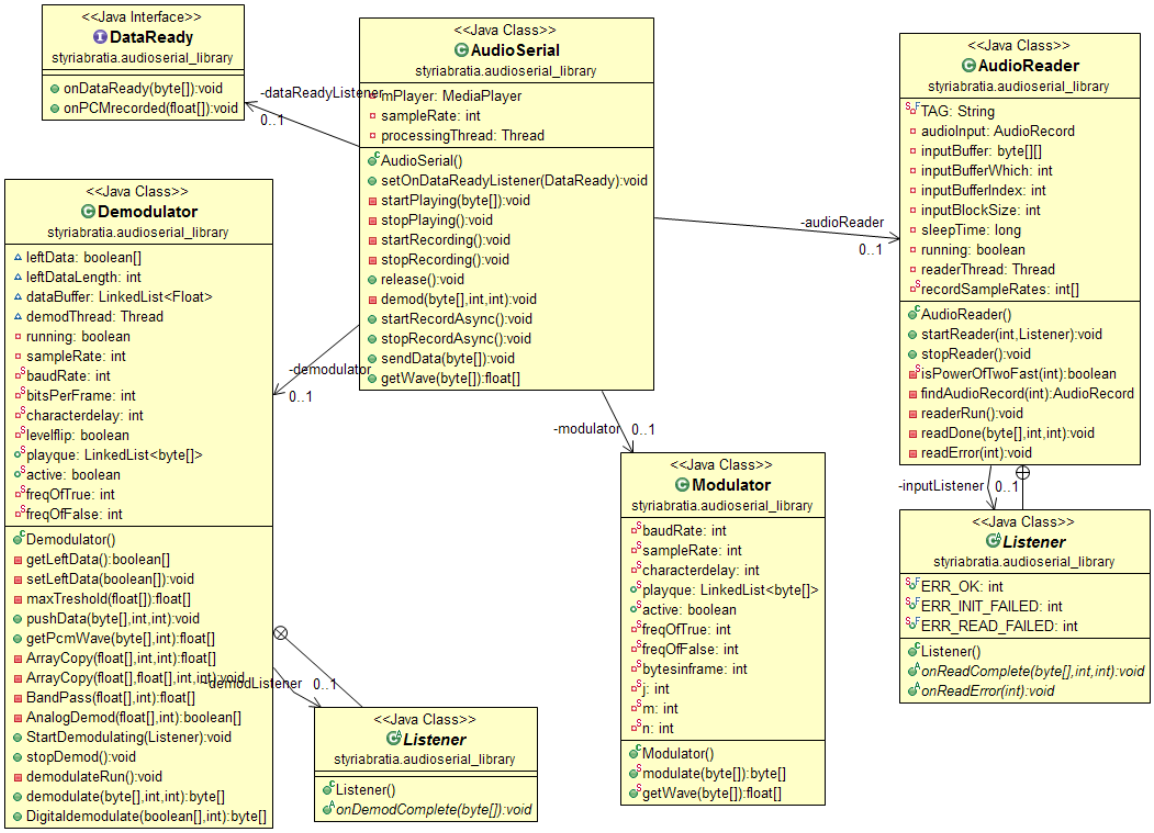
Táto kapitola nie je len strohou dokumentáciou implementovaných metód knižnice, ale ukazuje tiež globálny pohľad na naše riešenie, a obsahuje aj všetky potrebné informácie na prácu s touto knižnicou.

2.1 Knižnica

Knižnica je navrhnutá tak, aby práca s ňou bola čo najjednoduchšia, používateľovi stačí trieda AudioSerial. Uml schéma sa nachádza na obrázku 2.1. Knižnica rieši aj paralelné spracovanie dát - nahrávanie ako aj prehrávanie, bežia v samostatných vláknach, modulácia a demodulácia majú tiež vlastné vlákna. Odovzdávanie dát medzi vláknami je riešené cez listenery, používateľovi knižnice stačí implementovať DataReady interface, ktorý spúšťa onDataReady metódu, ktorá vracia demodulované dáta.

Aktuálna verzia knižnice spracováva dáta off-line, čo znamená, že najprv nahrávku nahrá, potom ju spracuje. Nahrávať sa dá s buffrom dĺžky minimálne 4096 bajtov, toto číslo závisí od konkrétneho zariadenia, ale čím väčší buffer sa zvolí tým kvalitnejšie môže prebehnúť jeho analyzovanie. Prehrávanie tiež najprv vytvorí audio stopu a následne ju trieda AudioSerial prehrá ako Androidový objekt AudioTrack pomocou Android triedy MediaPlayer, na nahrávanie používame Android triedu AudioRecord.

Vo väčšine projektov malé oneskorenie, spôsobené nahrávaním, nevadí. Ako sme spomínali, buffre sa dajú skoro ľubovolne zmenšiť, modulácia funguje aj po bytoch. V ďalších verziách knižnice sa plánujeme zaoberať aj online spracovávaním dát, takéto riešenie ale bude vyžadovať externú knižnicu.



Obr. 2.1: UML graf knižnice

UML graf implementovanej knižnice.

2.1.1 AudioSerial

Používateľovi stačí objekt Hlavná trieda knižnice pre pracovanie s celou knižnicou. Je jednoducho navrhnutá má len pár verejných tried, pre lepší prehľad. Hlavná trieda knižnice spracováva moduláciu a demoduláciu v príslušných inštanciách, a má na starosti len prehrávanie a správu nahrávania audio záznamu na android zariadení.

AudioSerial

public AudioSerial()

Empty constructor.

setOnDataReadyListener

public void setOnDataReadyListener(DataReady listener)

Sets DataReady Listener to this class. Listener in method onDataReady(byte data) returns demodulated data.

startRecordAsync

public void startRecordAsync()
Starts recording thread, and demodulation process, demodulated data are returned through DataReady interface.

stopRecordAsync
public void stopRecordAsync()
Stops recording thread.

sendData
public void sendData(byte[] data)
Send data out of device, data are modulated and then played.
@param data Data to modulate and send.

getWave
public float[] getWave(byte[] data)
Returns modulated data in raw format, normalized, sinus wave, used for control, or drawing plot.
@param data Data to be modulated and returned in floats

demod
public void demod(byte[] data)
Perform demodulation without threading and recording.

release
public void release()
Call to release resources and stop recording, and playing.

2.1.2 Demodulator

Trieda Demodulator ako z názvu vyplýva má na starosti demoduláciu signálu. Demodulačný proces beží v samostatnom vlákne, dáta sa mu posielajú metódou pushData a demodulované dáta sa vracajú cez listener, ktorý sa inicializuje pri spúštaní vlákna.

Public methods
Demodulator
public Demodulator()
Empty constructor.

pushData

public void pushData(byte[] data, int sRate, int recordFormat)

Pass data to be demodulated. Data are store in LinkedList structure.

@param data recorded audio

@param sRate sample rate of data

@param recordFormat PCM16 or PCM8, from AudioRecord properties

getPcmWave

public float[] getPcmWave(byte[] data, int recordFormat)

Returns discrete float representation of PCM wave, Little endian encoding is used.

@param data Byte representation of PCM audio wave.

@param recordFormat Format of recording, PCM 16bit and PCM 8bit are supported.

StartDemodulating

public void StartDemodulating(Listener listener)

Start demodulating thread.

@param listener Listener from this class, returns bytes in onDemodComplete method.

stopDemod

public void stopDemod()

Stops demodulating thread.

Digitaldemodulate

public byte[] Digitaldemodulate(boolean[] result, int sampleRate)

Demodulation of digital signal, finding packets and retrieve bytes from them. Called after analog demodulation.

@param result array of bits in boolean format, 1 represent True, 0 False

@param sampleRate sample rate of recording in waveform

Private methods

maxTreshold

private float[] maxTreshold(float[] data)

Procedure which increases differences in signal.

demodulateRun

private void demodulateRun()

Calling in thread, by start demodulation, control of demodulation process is here.

BandPass

private float[] BandPass(float[] waveform, int sampleRate)

BandPass filter implemented with FFT and inverse FFT.

@param waveform Audio wave which will be demodulated, length needs to be power of two, because BandPass and FFT.

@param sampleRate sample rate of recording in waveform

AnalogDemod

private boolean[] AnalogDemod(float[] waveform, int sampleRate)

Procedure performs demodulation from analog raw audio wave, returns bits in boolean format.

@param waveform Audio wave which will be demodulated, length needs to be power of two, because BandPass and FFT.

@param sampleRate sample rate of recording in waveform

2.1.3 Modulator

Trieda zastrešujúca moduláciu. Funkcionálne je rozdelená na dve časti, v prvej prebieha modulácia, vytvorenie postupnosti bitov a vytváranie audio vlny v druhej prebieha prekódovanie na PCM formát.

Modulator

public Modulator()

Empty constructor

modulate

public static byte[] modulate(byte[] data)

Calls getWave to get modulated audio wave, then conversion to 16bit PCM is made.

@param data Buffer obsahujúci dáta.

getWave

public static float[] getWave(byte[] data)

Creates stream of bits from bytes, then modulation is performed, by copying prepared samples in specific frequency.

@param data Buffer obsahujúci dáta.

returns floats representing modulated signal

2.1.4 AudioRader

Trieda vo vlastnom vlákne zaznamenáva vstup z mikrofónu, keď má plný buffer posiela dáta triede implementujúcej listener. AudioRader je modifikovaná trieda implementovaná Ianom Cameronom Smithom[10]

Trieda Listener
Listener si inner class of AudioReader, could be created in onStartMethod. Returns recorded data or error.
onReadComplete
public abstract void onReadComplete(byte[] buffer, int sample, int format)
Metóda je vyvolaná keď sú načítané dáta v bufferi.

@param buffer Buffer obsahujúci dáta.
@param sample Samplerate potrebné pre demoduláciu.
@param format Formát dát, PCM8 PCM16.

startReader
public void startReader(int block, Listener listener)
Start this reader.

@param rate The audio sampling rate, in samples / sec.
@param block Number of samples of input to read at a time. This is different from the system audio buffer size.
@param listener Listener to be notified on each completed read.

stopReader
public void stopReader()
Stop reading microphone input. Releases resources.

2.1.5 DataReady

Interface DataReady treba implementovať kvôli asynchrónnej demodulácií dát. Pomocou tohto interface sa v metóde onDataReady vracajú demodulované dáta.

onDataReady

public abstract void onDataReady(byte[] demodulatedByte)

In this method are demodulated data returned.

@param demodulatedByte demodulated data

onPCMrecorded

public abstract void onPCMrecorded(float[] rawWave)

Implement to get raw PCM floats.

@param rawWave

2.2 Demo aplikácia

Ku knižnici je implementovaný aj jednoduchý testovací projekt, ktorý prijíma a posiela dáta a tiež vizualizuje vytvorenú a aj prijímanú audio vlnu. Dáta sú jednoduchého textového typu. Na simuláciu hardvérového senzora používame freeware program AccessPort [11], ktorý spolupracuje s rôznymi modemami.

Na vykreslenie audio vlny používame custom View, ktorý používame v Aktivite Show a tiež vo fragmente v MainActivity, kvôli tomuto fragmentu je vyžadované minimálne SDK verzie až 11, ale ak sa fragment odstráni aplikácia a samozrejme aj knižnica sú podporované už SDK 8.

2.2.1 MainActivity

MainActivity spúšťa a zastavuje demoduláciu, do textového pola zobrazuje prijaté správy. Pre ukážku a testovacie účely nahrávanú audio stopu zobrazuje vo fragmente. Text z TextEditu posiela na modulovanie.

Serializácia reťazcov

V nasledujúcej ukážke, serializujeme textové dáta a následne ich modulujeme a posielame pomocou triedy AudioSerial.

```
AudioSerial as = new AudioSerial();
EditText edit = (EditText) findViewById(R.id.editTextSend);

as.sendData(edit.getText().toString().getBytes());
```

Takto správne získať z ASCI kódov v byte, char.

```
byte[] demodData;
StringBuffer sb = new StringBuffer(16);

for(int i=0;i<demodData.length;i++){
    sb.append(Character.toString((char) demodData[i]));
}
```

2.2.2 Show

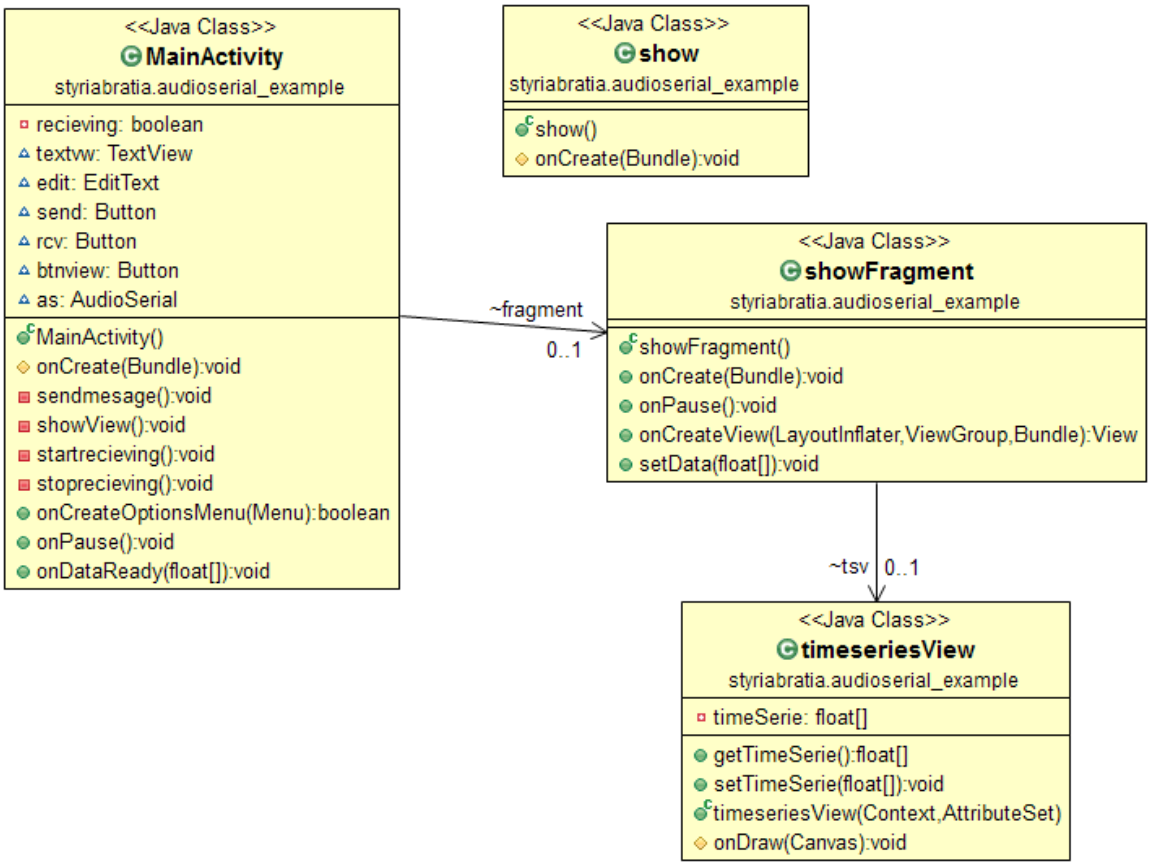
Základná aktivita kde sa zobrazuje modulovaný signál, na vykreslenie sa používa TimeseriesView, ktorý dostane parametre cez bundle.

2.2.3 ShowFragment

Fragment s custom TimeseriesViewom, slúži na zobrazenie audio vlny v MainActivity.

2.2.4 TimeseriesView

Custom view zobrazuje dáta vo float formáte, na Canvas. Dokáže ukladať bitmapy na SD kartu.



Obr. 2.2: UML graf testovacej aplikácie

UML graf jednoduchej vzorovej aplikácie, používajúcej knižnicu.

Kapitola 3

Implementácia

V tejto kapitole sa venujeme úvodu do prostredia, použitých princípov a teoretickému základu modulácie a demodulácie, v príslušných podkapitolách uvádzame vlastné riešenie aj s ukážkami dôležitých častí kódu.

3.1 Android knižnica

Na úvod kapitoly uvádzame základné charakteristiky prostredia Android a tiež rozdiely medzi vývojom pre desktopovú a mobilnú platformu. Android je rozšírená open source platforma nielen pre mobilné zariadenia, zastúpenie na trhu v štvrtom kvartály 2012, bolo 70%, pre porovnanie druhý najväčší prienik má iOs s 21%. V tabuľke je zhrnutie analýzy trhu operačných systémov od Gartner[12].

OS	Android	iOS	Microsoft Phone	BlackBerry	Samsung Bada	Symbian
podliel v %	69,7	20,9	3	3,5	1,3	1,2

Tabuľka 3.1: zastúpenie na trhu OS, Q4, 2012

Zastúpenie na trhu nie je jediným kritériom na výber platformy. Aj keď má iOS menšie zastúpenie ma väčšiu kúpyschopnosť aplikácií, môže to byť tým, že používatelia sú zvyknutý a ochotný si priplatiť. Sériový port cez audio port implementovaný v open source knižnici, by mal byť dostupný čo najviac developerom, rozhodol som sa pre operačný systém Android.

Vývojové nástroje Android nevyžadujú aby boli všetky súbory súčasťou projektu. Použitím knižnice sa dá vytvoriť a ďalej spravovať kód a aj ostatné prostriedky, ktoré môžu byť použité ostatnými Android projektami, preto by knižnica mala byť univerzálna. Projekt označený ako knižnica sa nedá priamo skompilovať ako klasická Android aplikácia.

Používanie knižnice pomáha sprehľadniť kód, pridáva ďalšiu štruktúru projektu, oddeľuje všeobecnejší a univerzálny kód od aplikácie. Pridaním knižnice do projektu, projekt získava dodatočnú funkcionalitu, v súčastnosti je dostupných čoraz viac Open Source knižníc pre platformu Android. Preto je pre každého vývojára dôležité porozumieť knižniciam a naučiť sa pracovať s nimi. Pri kompilovaní projektu, ktorý používa knižnice, sa skompilujú aj komponenty knižnice a pridajú sa do APK súboru aplikácie. Knižnica môže obsahovať Java triedy, Android komponenty aj zdroje. Iba assety nie sú podporované.

Keďže Androidový vývoj prebieha v Jave a zvyčajne sa používa Eclipse, knižnice sa používajú podobným spôsobom, ukážeme krátky návod ako pridať knižnicu do projektu. V programovacom prostredí Eclipse sa knižnice vytvárajú, vo vlastnostiach projektu, na karte Android treba zaškrknúť políčko „je knižnica“. Vo vlastnostiach projektu, ktorý má používať vytvorenú knižnicu, sa na karte Android dá pridať knižnica, otvorí sa ponuka s knižnicami v aktuálnom pracovnom priečinku. Knižnica musí deklarovať všetky svoje súčasti, napríklad činnosti, služby, atď. v súbore AndroidManifest.xml tak isto ako bežná aplikácia. Každý ďalší projekt, môže tiež používať túto knižnicu.[13]

3.2 Vysvetlenie pojmov

Pre porozumenie problematiky spracovania audio-signálu je dôležité osvojiť si základné pojmy. Niektoré pojmy sú ľahko zameniteľné preto ich uvádzame prehľadne a spolu.

Baud rate - vyjadruje počet zmien stavu prenosového média za sekundu. Jednotkou je baud.

Bit rate - je počet bitov spracovaných alebo prenesených za jednotku času. Jednotkou je bit per second (bit/s) bit za sekundu.

Sample rate - vzorková frekvencia, je počet vzoriek za jednotku času, zvyčajne za sekundu, jednotkou je hertz.

Sample / vzorka - vzorkou je myslená práve jedna hodnota, nesúca údaj o výške amplitúdy signálu.

Parity bit - Je hodnota označujúca či je počet jednotkových bitov v dátovej sekvencii párny alebo nepárny, používa sa ako najjednoduchšia forma zistenia chyby v prenose.

PCM - Pulzná kódová modulácia je metóda reprezentovania analógového zvukového signálu ako signál digitálny.

UART - Universal Asynchronous Receiver/Transmitter, je zariadenie pre sériovú komunikáciu, zapája sa na Rx pin pre príjem, Tx pin pre posielanie, ak sa žiadne dáta neposielajú vysiela logickú jednotku.

Android AudioTrack Trieda slúžiaca na prehrávanie súborov. Má statický a stream mód, stream na väčšie dáta ktoré sa nevmestia do operačnej pamäte, statický mód posiela dáta na prehrávanie práve raz pred začatím prehrávania. Je rýchlejší na malé dáta, lepšia odozva, používa sa napríklad pri hrách.

3.3 Modulácia

Prenos bajtov pomocou audio signálu sa rieši moduláciou a demoduláciou. Všetky dáta vieme serializovať do postupnosti bajtov, preto sa môžme zamerať len na prenos bajtov. Proces ovplyvňovania nosného signálu, typicky sínusového, za účelom prenesenia informácie sa nazýva modulácia.

Základné typy analógovej modulácie sú:

- Amplitúdová modulácia (AM, amplitude modulation)
- Fázová modulácia (PM, phase modulation)
- Frekvenčná modulácia (FM, frequency modulation)

Typy číslicovej modulácie sú:

- Amplitúdové kľúčovanie (ASK, amplitude-shift keying).
- Frekvenčné kľúčovanie (FSK, frequency-shift keying).
- Fázové kľúčovanie (PSK, phase-shift keying).

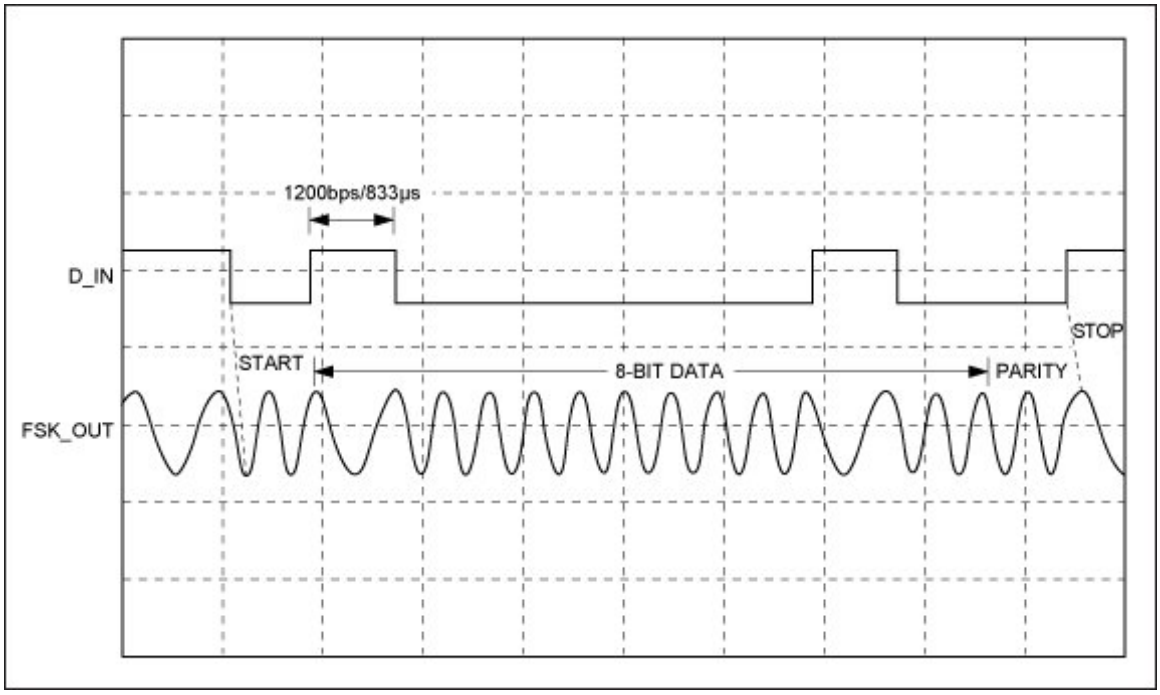
V digitálnej modulácií je analógový nosný signál modulovaný, diskrétnym signálom. Dá sa povedať, že diskrétny digitálny signál sa kľúčuje pomocou analógového signálu. Každý typ číslicovej modulácie má konečný počet rôznych signálov na reprezentáciu číslicových údajov.

3.3.1 FSK modulácia

Táto podkapitola približuje FSK moduláciu, na príklade konkrétneho modemu.[14]

Testovací modem DS8500 HART Modem, podporuje HART protokol (Highway Addressable Remote Transducer). HART protokol je dnes jeden z najpopulárnejších priemyselných protokolov, bol vyvinutý na začiatku osemdesiatych rokov spoločnosťou Rosemount. Bol to prvý obojsmerný digitálny komunikačný protokol pre inteligentné prevádzkové prístroje, ktorý nerušil analógový signál. Znamená to, že protokol umožňuje prenos jednej analógovej veličiny, zvyčajne hlavnej a digitálnej informácie, ktorá sa môže využívať na konfiguráciu, kalibráciu, diagnostiku a monitorovanie prístroja, alebo môže ísť o hlavnú procesnú veličinu v digitálnej podobe. Obidva typy prenosov môžu prebiehať paralelne na tom istom káblovom spojení.

My sa budeme zaoberať práve analógovou časťou, ktorá je založená na FSK klúčovaní. Tento druh číslicovej modulácie má konečný počet frekvencií, konkrétne budeme používať 1200Hz a 2200Hz, každá z týchto frekvencií bude kódovať jednu číslicu, teda 0 alebo 1.



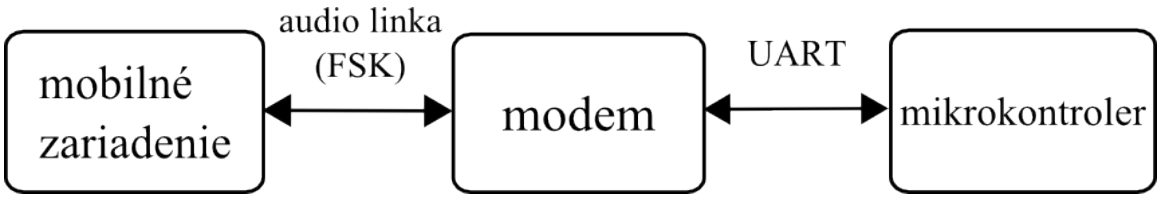
Obr. 3.1: FSK modulačná vlna
Schéma FSK modulácie.[14]

Naša FSK modulácia musí dáta posielať podľa UART formátu, ktorým modem komunikuje na digitálnej úrovni. UART formát vyžaduje jeden štartovací bit, 8 dátových bitov,

parity bit a stop bit. Celkový balast pridaný k dátam sú 3 bity z 8 dátových bitov, čo je 37,5%. Schému modulácie môžme vidieť na obrázku 3.1, dĺžku alebo počet vzoriek v jednom zakódovanom bite vypočítame ako

$$n = \frac{samplerate}{baudrate}$$

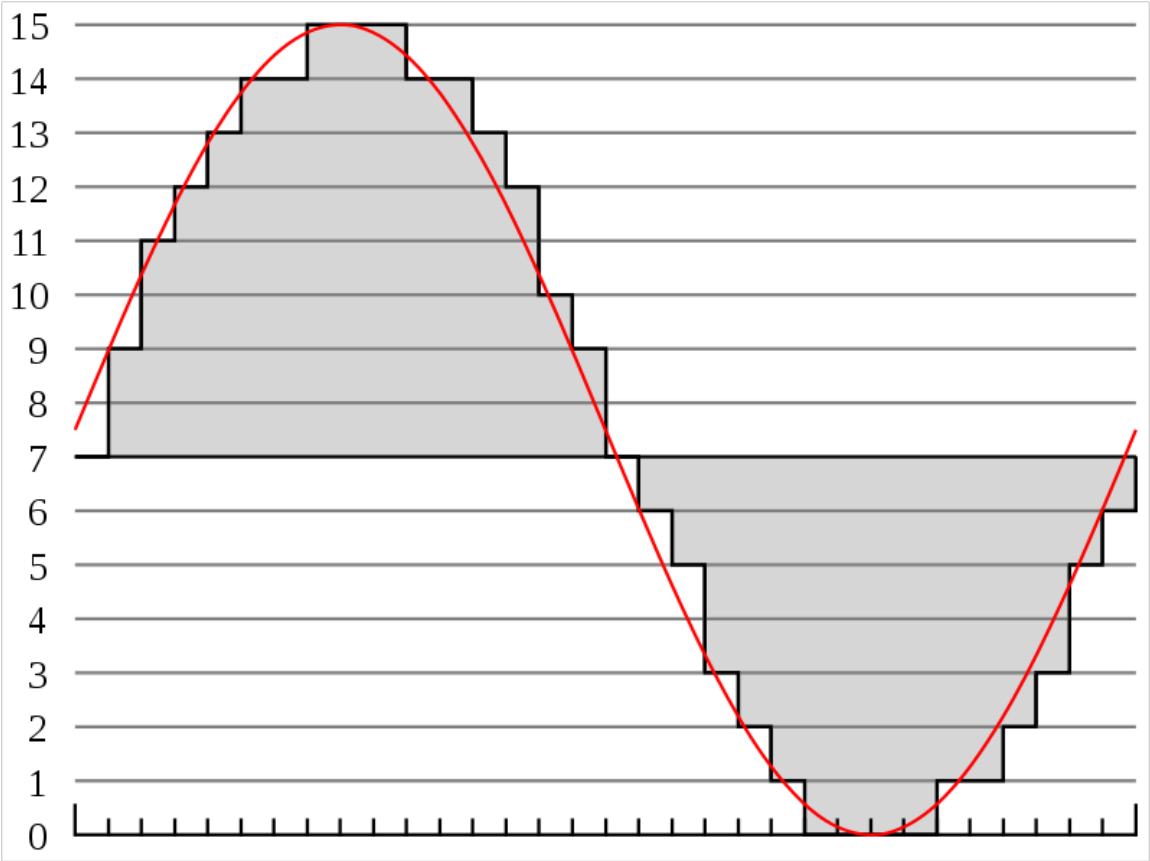
. Na obrázku ďalej vidíme, že štart bit je zakódovaný vyššou frekvenciou, 2200Hz teda bitová hodnota je 0 a stop bit má frekvenciu 1200Hz, bitová 1.



Obr. 3.2: Schéma zapojenia
Schéma zapojenia mobilného zariadenia a modemu s mikrokontrolerom.

3.3.2 PCM kódovanie

PCM kódovanie je jeden zo spôsobov akým sa analógový signál zaznamenáva do digitálnej podoby. V roku 1926, Paul M. Rainey z Western Electric patentoval fax, ktorý posielal signál v 5-bit PCM formáte. Princíp ukladania sa dá popísať ako diskrétne odčítanie hodnôt v pravidelných intervaloch. Jedna zaznamenaná hodnota, výška amplitúdy sa nazýva vzorka a proces vzorkovanie. Počet vzoriek udáva *sampling rate* alebo vzorkovacia frekvencia. Čím vyššia je vzorková frekvencia a čím väčší je rozsah hodnôt ktorý môže jedna vzorka nadobúdať tým viac sa autentickosť digitálnej reprezentácie blíži k spojitej audio vlne. Obrázok 3.3 ukazuje vzorkovanie červenej sínusovej vlny, graf je rozdelený na pravidelné intervaly a pre každý bod je určená najbližšia hodnota sínusovej vlny, presnosť je tu obmedzená na 4 bity. Takže sa jedná o 4-bitové PCM.



Obr. 3.3: ukážka PCM

PCM vzorkovanie audio vlny.[15]

Nyquistov-Shannonov teorém

„Presná rekonštrukcia spojitého, frekvenčne obmedzeného, signálu z jeho vzoriek je možná len vtedy, ak bola vzorkovacia frekvencia vyššia ako dvojnásobok najvyššej harmonickej zložky vzorkovaného signálu.“

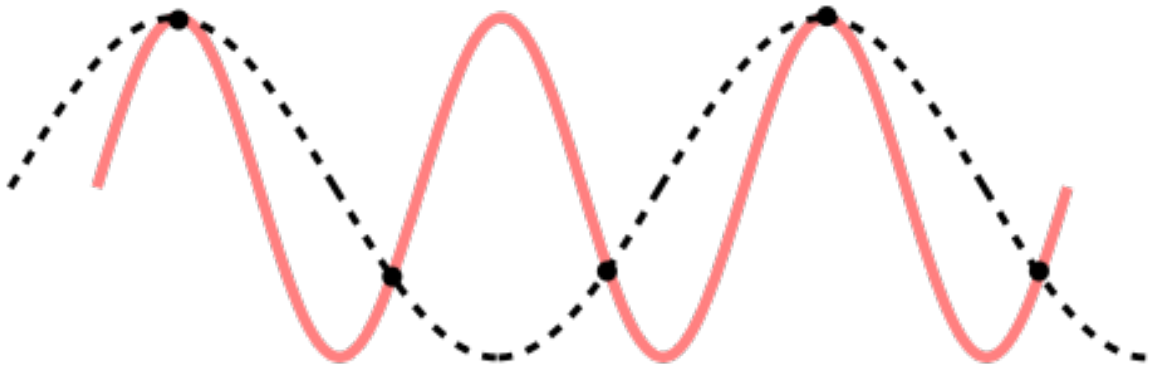
Teorém sa dá vyjadriť aj touto pozmenenou rovnicou, kde:

Δt - je interval medzi dvoma vzorkami

f_{max} - je hodnota najvyššej zložky signálu

$$\Delta t \leq \frac{1}{2 f_{max}}$$

Nyquistov-Shannonov teorém sa dodržiava pri spracovávaní digitálnych dát dokonca aj pri vzorkovaní obrazových dát. Dôležitý je aj pri Fourierovej transformácií. Vzniknutý aliasing, pri zle zvolenej vzorkovacej frekvencií, je vidno na obrázku 3.4.



Obr. 3.4: Nyquistov-Shannonov teorém
Pri zvolení vzorkovacej frekvencie len 1.5 násobku maximálnej zaznamenávanej frekvencie, dochádza ku skresleniu, vzniká nejednoznačnosť pri reprezentácií. Čierne body sú jednotlivé vzorky. [16]

3.3.3 Implementácia modulácie

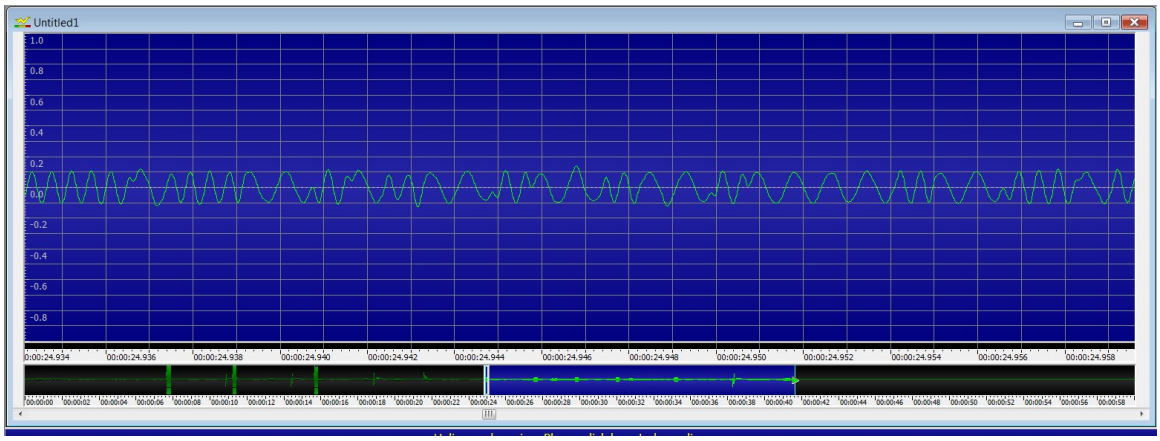
Moduláciu sme implementovali na základe [17] a [2]. Je implementovaná do triedy Modulator, podľa obsahu sa dá rozdeliť na tri časti, prípravu dát, samotnú moduláciu a úpravu formátu.

Teoreticky sa dajú posielať všetky typy dát, ktoré sa dajú serializovať na postupnosť bajtov, serializácia všetkých základných dátových typov je v Jave natívne implementovaná. Budeme používať UART formát, ktorého rámce obsahujú osem dátových bitov teda jeden bajt. Bajt je v Jave najmenší číslicový typ, preto budeme bity reprezentovať pomocou typu boolean, ktorý je vymenovaný typ a nadobúda hodnoty *true* alebo *false*, preto sa musíme spoľahnúť, že Java virtual machine bude *true* vždy reprezentovať ako 1 a *false* ako 0. Prevod Bajtu do dvojkovej sústavy, je dobré urobiť klasickým vyhľadávaním dvojkových mocnín, spravíme bitový and bajtu s príslušnou mocninou dvojky a porovnaním s mocninou dvojky získame bit v boolean zápise. Na začiatok rámca pridáme štart bit s hodnotou 0, za dátové bity pridáme parity bit, ktorý vypočítame z dátových bitov a stop bit s hodnotou 1. Z takto pred-pripravených dát vytvoríme audio vlnu.

Dáta by sme mohli modulovať už pri vytváraní jednotlivých bitov, keď to oddelíme síce stúpne zložitosť, musíme znova prejsť cez všetky prvky pola, ale kód je ľahšie čitateľný. Čiže keď máme vytvorené pole bitov, môžme začať moduláciu. Jednoduchý spôsob generovania vlny, zväčšujúcim sa j generuje sínusoidu, toto je však náchylné na chyby v časovaní.

```
j = 0;
```

```
for (int i = 0; i < bits.length; ++i) {
  for (int k=0;k<n;k++){
    if(bits[i]){
      waveform[j] = (float)
        Math.sin(2*Math.PI*j*freqOfTrue/sampleRate);
    }else{
      waveform[j] = (float)
        Math.sin(2*Math.PI*j*freqOfFalse/sampleRate);
  }
  j++;
  }
}
```



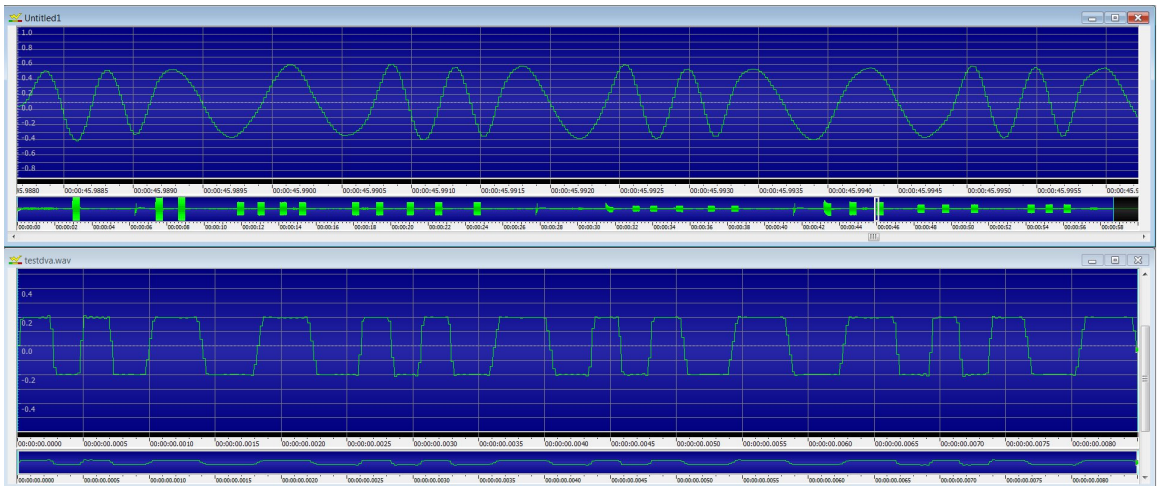
Obr. 3.5: Prvý pokus modulácie
Implementácia modulácie s chybným fázovým posunom.

Druhý krajší spôsob tvorenia vlny, vyžaduje predprípravu dát. Dopredu vypočítame reprezentácie 0 a 1, a potom z nich vyskladáme vlnu pre všetky bity. Jedna táto reprezentácia, pattern, má veľkosť n, pričom $n = samplerate/baudrate$. Podiel $freqOfTrue/sampleRate$ určuje periódu a tým aj frekvenciu sínusoidy. Tento spôsob trochu zjednodušil aj zložitosť výpočtu amplitúdy, keďže výpočet prebehne len dvakrát, raz pre 0 raz pre 1, a potom ak je bit v poli *true* tak sa na koniec tvoriacej sa vlny nakopíruje predpripravený pattern pre true bit, ak je false tak pattern so zmenenou frekvenciou.

```
float[] TrueVzorka = new float[n];
for (int k=0;k<n;k++){
  TrueVzorka[k] = (float)
    Math.sin(2*Math.PI*k*freqOfTrue/sampleRate);
}
```

```
float[] FalseVzorka = new float[n];
for (int k=0;k<n;k++){
  FalseVzorka[k] = (float)
    Math.sin(2*Math.PI*k*freqOfFalse/sampleRate);
}
j = 0;
for (int i = 0; i < bits.length; ++i) {
  if(bits[i]){
    for (int k=0;k<n;k++){
          waveform[j+k] = TrueVzorka[k];
  }else{
        for (int k=0;k<n;k++){
          waveform[j+k] = FalseVzorka[k];
        }
  }
  j = j + n;
}
```

Dôležité je zjednotenie nastavení jednotlivých tried, keďže modulácia a prehrávanie sú implementované v rôznych triedach, vznikla potreba zjednotenia lokálnych premenných. Riešením je špeciálna trieda pre nastavenia napríklad Androidové shared preferences. Rozdielne nastavenia vedú k rôznym chybám, ak sa pri generovaní bit rate nastaví na 48000Hz a pri prehrávaní v inej triede sa nezmení táto hodnota a ostane nastavená na 44100Hz, tak chyba sa prejaví ako zrýchlenie vytvorenej vlny, vlna bude mať kratšie trvanie vzoriek, vznikne posunutie a tento signál nebude správne demodulovaný.



Obr. 3.6: Finálna modulácia

Porovnanie, implementovanej modulácie vo finálnej podobe s výstupom z HART modemu.

Na prehrávanie na Android zariadení sme vybrali 16 bitový PCM formát. Čo znamená, že moduláciou získaná sínusoida sa v treťom kroku musí skonvertovať z typu float na byte. Z floatu najprv prenásobením maximálnou hodnotou získame short, ten rozdelíme na dva bajty ktoré uložíme za seba. Zachováme little endian poradie bajtov, ktoré je vďaka procesorom Intel najrozšírenejšie, least significant byte je uložený na menšej adrese, v našom prípade menšom indexe poľa. Tiež zachováme znamienko.

Na obrázku 3.6 je finálny výstup fsk modulácie, pri porovnaní s výstupom z hardvérového modemu vidíme, že až na rozdielnu amplitúdu sa jedná o ten istý UART rámec. Takto vytvorená audio vlna sa dá prehrať pomocou Android.AudioTrack. Používame statický mód, čo znamená, že dáta sú poslané na prehrávanie pred začatím prehrávania a to práve raz.

3.4 Demodulácia

Demodulácia je opačný proces ako modulácia, implementácia demodulácie je zložitejšia, ale aj zaujímavejšia. Na začiatku podkapitoly ukážeme teóriu demodulácie, potom uvedieme našu vlastnú implementáciu.

Demoduláciu by sme vedeli rozdeliť do rovnakých kategórií ako moduláciu v kapitole 3.3. Preto sa budeme rovno zaoberať všeobecným pohľadom a potom uvedieme príklady FSK demodulácie.

3.4.1 Typy demodulácií

Na implementáciu knižnice sme si vybrali FM moduláciu, nie len pre to, že na tento typ modulácie existuje množstvo hardvéru ale hlavne preto, že šumenie sa najčastejšie prejavuje amplitúdovo čiže neovplyvňuje frekvenciu signálu. Čiže aj keď implementácia FM demodulácie je zložitejšia ako AM modulácia, pre lepšie vlastnosti sa oplatí používať FM spôsob. Hlavná podstata všetkých typov FM demodulácie je previesť frekvenčne modulovaný signál na AM, z ktorého sa dajú dáta jednoducho odčítať (pozeráme len veľkosť amplitúdy).

Jednoduchá demodulácia detekciou prechodu cez nulu

Metóda (angl. zero crossing [18]) je založená na vlastnosti sínusu, ktorý sa škáluje v závislosti od frekvencie, pri väčšej frekvencií prejde sínusovka nulou viac krát ako pri menšej frekvencií. Tento princíp sa využíva tak, že vždy keď sa detekuje prechod nulou vygeneruje sa impulz fixnej dĺžky. Integrovaním impulzov, na tejto dĺžke, získame demodulovaný signál. V digitálnej demodulácii, čo je aj prípad našej knižnice stačí rátať počet zmien znamienka.

Princíp demodulácie prechodom cez nulu vyjadrený rovnicou:

$$f = \frac{1}{\lambda} \tag{3.1}$$

f značí frekvenciu. [Hz]
λ je vlnová dĺžka. [s]

Vylepšená demodulácia detekciou prechodu cez nulu

Vylepšená demodulácia detekciou prechodu cez nulu (angl. improoved zero crossing [19]), do klasického hľadania prechodov cez nulu, pridáva aj algoritmus na odhad frekvencie. Vždy, keď sa nájdu dve susedné vzorky s rôznym znamienkom, vyhodnotí sa aktuálna frekvencia ako podiel vzorkovacej frekvencie a odhadovaným časom medzi dvoma prechodmi. Frekvencia FM signálu je úmerná amplitúde modulačného signálu a teda robíme FM demoduláciu.,

Odhad frekvencie ja založený na 3.1 tento vzťah môžme prepísať:

$$f\Delta = \frac{f_s}{t_n - t_p} - f_c$$

$f\Delta$ značí okamžitú frekvenčnú odchýlku. [Hz]
f_s vzorkovacia frekvencia [Hz]
f_c nosná frekvencia [Hz]

t_n čas nasledujúcej vzorky [id vzorky]
t_p čas predošlej vzorky [id vzorky]

Digitálne rátanie prechodov cez nulu sa dá vylepšiť bandpass filtrom. Ktorý odstráni šum, ktorý môže mať nízku amplitúdu ale okolo nuly, bude sa pripočítavať čo nechceme. Tento typ demodulácie sme sa rozhodli aj implementovať.

Fázová slučka

Je klasická metóda používaná v analógových hardvérových prevodníkoch, ale dá sa implementovať aj v digitálnej podobe. Podstatou je skonvertovať FM modulovaný signál do AM modulácie, to sa dá urobiť diferenciáciou signálu a následnej AM demodulácie pomocou bandpass filtra.

Demodulácia pomocou arkustangens kvadrantov

Demodulácia pomocou arkustangens kvadrantov využíva kvadrantovú reprezentáciu FM modulovaného signálu.[20] Tento signál sa dá reprezentovať ako:

$$I(t) = Acos(\phi(t))$$

$$Q(t) = Asin(\phi(t))$$

I je reálna časť signálu, (**i**n-phase)
Q je imaginárna časť signálu, (**q**uadrature-phase)
ϕ značí momentálnu fázu prijatého signálu.
Z týchto vzťahov sa dá momentálna fáza získať odvodením cez uhol komplexného čísla $I + jQ$, odtiaľ sa dá vyjadriť arkus tangensom. Pri vyjadrovaní arkus tangensu sa nám vykráti A, ktoré značí amplitúdu nosného signálu. Derivácia okamžitej fázy prijatého signálu v čase, priamo úmerná s modulovaným signálom, čiže sme demodulovali FM signál.

Táto metóda má dobré výsledky aj na zašumenej nahrávke, keďže šum vplýva hlavne na amplitúdu signálu a tá sa pri tejto metóde vykráti.

Z tejto metódy sú odvodené aj ďalšie typy FM demodulácie ako napríklad oneskorenie zakladaného pásma,(baseband delay) kde sa základné pásmo získava práve kvadrantovým rozložením signálu. Podobnou metódou je aj diferenciácia základného pásma.

3.4.2 Filtre

Na digitálne spracovanie signálu sa dajú použiť rôzne filtre a algoritmy. Uvedieme z nich pár základných, niektoré sme využili aj v našej knižnici.

FFT

Fast fourier transformation, rýchla furierová transformácia slúži na spektrálnu analýzu dát. Najjednoduchší príklad použitia z bežného života je práve ukazovateľ spektra na prehrávačoch alebo ekvalizér. Teoreticky je FFT ideálna na FM demoduláciu lebo konvertuje frekvenčne kódovaný signál na amplitúdovo kódovaný, ktorý sa omnoho ľahšie analyzuje.

BandPass

Band-pass filter (pásmový priepust) je lineárny filter ktorý prepúšťa signál len určitých frekvencií. Dá sa vytvoriť kombináciou **low-pass** a **high-pass** filtrov. Ideálny band-pass filter má úzke priepustné pásmo, prepustí len určité frekvencie, bez vedľajšieho šumu, a úplne zmaže všetky frekvencie mimo pásmovej priepustnosti. V skutočnosti žiaden band-pass filter nie je dokonalý. Oblasť kde sú frekvencie len potlačené a nie vymazané sa nazýva pásmo tlmenia (roll-off). Ani digitálna implementácia nemá pásmo tlmenia nulové. Tá zvyčajne používa FFT ktorej nepresnosti sme spomenuli vyššie. Na prepustenie len vybraných frekvencií sa môže použiť aj butterworthov filter.

Maximalizácia tresholdu

Je jednoduchý filter ktorý zväčší amplitúdy na maximálnu úroveň, pri frekvenčnej modulácií, amplitúdy nie sú nosičom signálu, ale je dôležité aby boli vzorky dobre rozlíšiteľné pre ďalšie spracovanie. Používa sa až po spracovaní záznamu inými rozumnejšími filtrami.

Kontrola parity

Filtre pomáhajú predísť chybám v demodulácií, ak sa predsa len nejaké chyby vyskytnú je možné ich do určitej miery opraviť, alebo ak používame synchrónny prenos, vypýtať si chybné údaje znova. Na základnú detekciu jednoduchých chýb sa dá použiť kontrola parity. HART protokol vyžaduje prítomný parity bit v každom posielanom rámci.

Existujú rôzne typy parity bitov, základné sú párny a nepárny. Párny paritný bit nastavíme na logickú 1 ak je počet 1 v danom reťazci, rámci, nepárny. Ak je počet jednotiek párny nastavíme ho na 0. Čiže ak je výsledok logickej XOR operácie jedna, túto jednotku ako keby zaznačíme do parity bitu, čiže celková parita reťazca je párna. Paritu totiž rátame len z dátovej časti. Pri nepárnom bite je to naopak.

Vďaka detekcií chýb parity bitom, vieme odhaliť chybný bit v dátovom rámci. Nevieme tento bit opraviť lebo nemáme informáciu, ktorý z prijatých bitov má invertovanú hodnotu. Kvôli jednoduchosti je kontrola parity implementovaná v mnohých zariadeniach.

3.4.3 Demodulácia v našej knižnici

Práca ktorú treba urobiť sa dá rozdeliť na analógovú a digitálnu časť demodulácie, analógová časť musí z postupnosti floatov, teda zo sínusoidy, získať postupnosť bitov, v digitálnej časti z postupnosti bitov vytvárame jednotlivé bajty, úlohou je rozpoznať štart a stop bit, prípadne overiť paritu.

Demodulácia analógového signálu

Naša implementácia demodulácie, opačne ako modulácia najprv, dekóduje PCM kódovanie a hodnoty prevedie do typu float. Používame ByteBuffer, konkrétne jeho metódu getShort. Je dôležité zmeniť predvolené nastavenie kódovania z big endian na použitý little endian. V prípade, že zariadenie nepodporuje 16 bitové PCM, ale len 8 bitové, ako napríklad Androidový emulátor je toto bezpredmetné a dáta len prevedieme na float a normalizujeme.

Pred samotnou demoduláciou audio signál upravíme BandPass filtrom, kvôli tomuto filtru spracovávame dáta s veľkosťou mocniny dvojky. Tento filter nie je dokonalý záleží na sample rate nahrávky a tiež veľkosti práve spracovávanej sekvencie. Treba tu nájsť správny pomer medzi veľkosťou nahrávky a dlhším, oneskorenejším spracovávaním. FFT rozdelí frekvencie do niektorých okien, veľkosť okna sa vypočíta ako *sample rate / dĺžka záznamu*. Pričom kvôli Nyquist–Shannonovmu teorému sú použiteľné okná len do frekvencie polovičnej hodnoty ako sample rate. Čiže ak je sample rate 48000 a spracovávame nahrávku o veľkosti len 64 bitov, dostaneme okná 0-350Hz 350-1100Hz ... -24000Hz. Pre frekvencie v rozmedzí jedného okna dostávame len jeden údaj o ich zastúpení v buffri.

Na demodulovanie sme si kvôli jednoduchosti implementácie vybrali demoduláciu detekciou prechodu cez nulu. Implementácia tejto techniky je celkom jednoduchá, stačí, že sledujeme zmenu znamienka v signále pri každej detekovanej zmene, zvýšime počítadlo. Podľa počtu vzoriek medzi dvoma zmenami znamienka sme schopný určiť frekvenciu.

Rýchlu Furierovu transformáciu zabezpečuje open source knižnica *Minim*. Táto knižnica je vytvorené pre špeciálne prostredie Processing, niektoré triedy a našťastie aj FFT sa bez problémov dajú použiť aj na Androide. Autor už vydal novú verziu ktorá nie je závislá na Processingových triedach ale používateľ si musí doimplementovať, pár metód pre prácu s dátami ktoré boli poskytované prostredím. Preto napriek komplexnosti používame len FFT.

BandPass filter sme implementovali pomocou Minim FFT triedy, s ktorou sa pracuje intuitívne, vie vypísať hodnoty frekvencií nie je nutné ich zvlášť rátať. Keďže k FFT existuje aj inverzná operácia, princíp BandPass filtra je rozložiť spracovávanú nahrávku na frekvencie, všetkým nežiadúcim frekvenciám nastaviť hodnotu na 0 a naspäť použiť inverzné FFT.

Demodulácia digitálneho signálu

Demodulácia digitálneho signálu musí v postupnosti bitov nájsť jednotlivé bitové rámce. Používame UART formát, čo znamená, že dĺžka rámca je 11bitov. Rámec začína štart bitom s hodnotou 0, pred štart bitom musel byť bit s hodnotou 1, stop bit, alebo v prípade, že neposielame dáta je dobré posielať jednotky, ustálenú frekvenciu. Čiže každý rámec začína zmenou 1 na 0, potom overíme stop bit, keďže signál spracovávame z nahrávky, overiť stop bit je jednoduché. Jednoduché je overiť či po stop bite znova nasleduje štart bit, alebo rámec jednotiek v prípade chýbajúcich dát. Ak je všetko toto správne môžme overiť parity bit, ak je parity bit nesprávny možno sme našli zlý začiatok rámca alebo sa v prenose vyskytla chyba. V prípade chyby môžme požiadať o retransmisiu. V prípade zlý začiatok odhalíme tak, že nám nasledujúce rámce nebudú spĺnať podmienky štart a stop bitov, vtedy môžme znovu nájsť iný štart bit. Spätná kontrola ani kontrola parity ešte v knižnici nie je implementovaná.

Po identifikovaní správneho rámca, môžme dátové bity skonvertovať do bajtu. Keďže Java nepozná neznamienkové premenné, použijeme dvojkový doplnkový kód na získanie hodnoty. Snažíme sa byť kompatibilný s Javou, chceme aby prenos fungoval aj medzi dvoma Android zariadeniami s našou knižnicou.

Keďže veľkosť buffera nemusí byť deliteľná s počtom bitov v rámci, môže sa stať, že nám nejaké dáta zvýšia. Ak nám ostali nejaké bity, z ktorých nevieme zložiť rámec, musíme ich odložiť, aby nám nechýbali keď prídu ďalšie dáta. Keďže celú demoduláciu má na starosti trieda demodulátor môžme si dáta uložiť do globálnej premennej. Tieto dáta potom spracujeme s novými dátami.

Na demodulácií je najťažšia synchronizácia. Či v analógovej alebo digitálnej časti je dôležité správne nájsť začiatok rámca. Frekvenčné skreslenia vznikajú aj chybami, nepresnosťou ADC, DAC prevodníkoch. Časový odstup medzi jednotlivými pulzmi vzorkovacej frekvencie nie je konštantný (závisí od kvality kryštálového oscilátora v prevodníku).Táto chyba sa nazýva časové chvenie, nestálosť (jitter).

3.4.4 Pripojenie zariadení

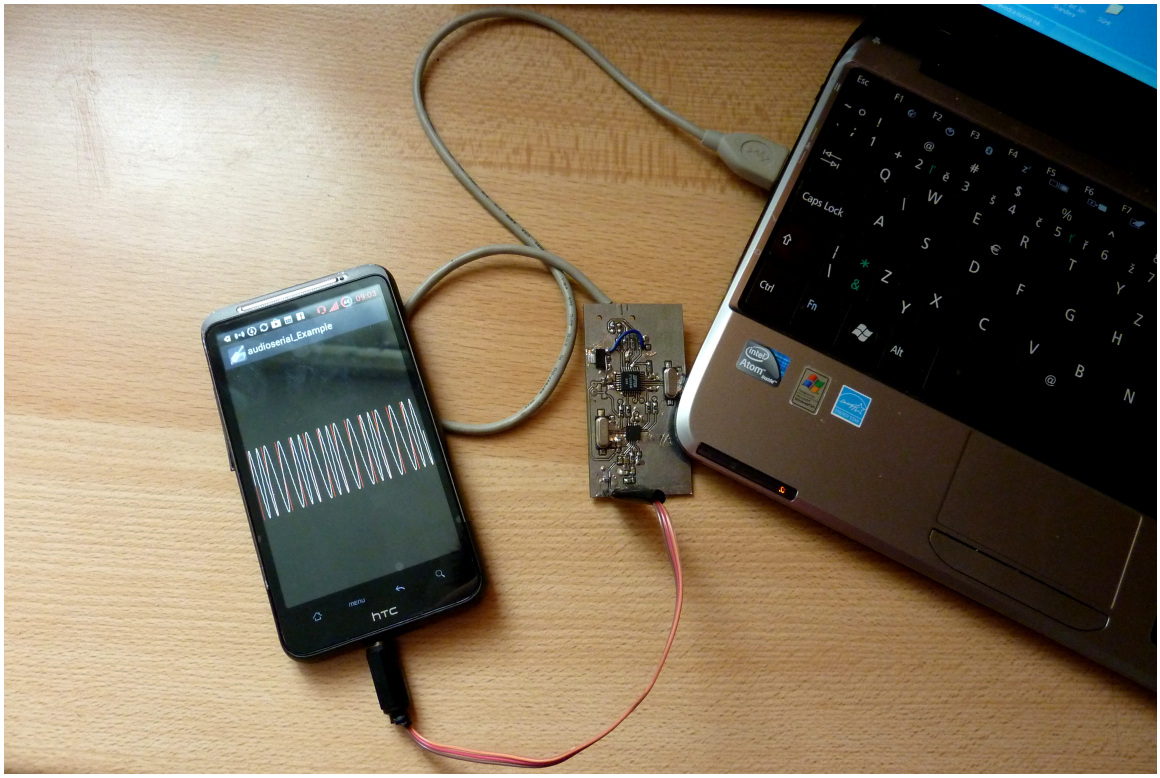
Pripojenie zariadení je jednoduché a dá sa realizovať obyčajným audio káblom, treba si však uvedomiť, že mobilné zariadenia majú audio konektor štvor-pólový. Mikrofón aj stereo výstup na slúchadlá sú na jednom konektore. Kábel na obrázku 3.7 sa dá zakúpiť napríklad tu [21] slúži na pripojenie mobilu k počítaču.

Na obrázku 3.8 je ukážka zapojenia HART modemu, miesto USB portu a počítača môže byť zapojené iné zariadenie, senzor, robot. Schémy na poskladanie modemu budú zverejnené na stránke knižnice, náklady na kúpu súčiastok by nemali presiahnuť 10€. Schéma zapojenia je na obrázku 3.2.

Jednou z výhod používania audio komunikácie oproti USB je ten, že mobilné zariadenie môže byť počas trvania komunikácie napájané. Bohužiaľ na testovanom mobile HTC Desire HD, sú tieto konektory blízko pri sebe a napájanie zariadenia značne rušilo audio signál, debugovanie prenosu bolo preto náročnejšie. Pri pripojení cez audio kábel vznikal len menší šum, zatiaľ, čo pri pripojení modemu bola komunikácia absolútne zašumená. Môžno pri iných mobilných zariadeniach bude interleaving signálov riešený lepšou izoláciou, alebo oddelením do iných častí zariadenia, a bude možné simultánne prenášať dáta obidvoma sériovými portami.

Obr. 3.7: Audio kábel

Redukcia štvorpólového audio jack kábla na dva troj pólové ktoré sa používajú na pripojenie do počítača.



Obr. 3.8: HART modem

Ukážka zapojenia modemu.

Záver

V tejto bakalárskej práci sme sa snažili priblížiť spracovávanie audio signálu. Ukázali sme analógový a z väčšej časti digitálny pohlaď na problematiku. Teória je dôležitý základ pre začatie akéhokoľvek projektu, preto sme sa venovali aj všeobecnej teórií, a širšiemu úvodu do problematiky, vďaka čomu sme si vybrali spôsob a metódy ktoré sme následne implementovali. Teóriou sme sa snažili vysvetliť prečo a hlavne ako funguje digitálne spracovanie audio signálu. Developerovi android aplikácie môže byť teória nápomocná pri práci s knižnicou, a môže pochopiť ako knižnica funguje.

Potencionálnych používateľov knižnice sme sa v kapitole 1 snažili, zaujať ukážkou existujúcich komerčných riešení, ktoré reálne používajú komunikáciu cez audio jack. Tiež sme ukázali výhody tejto formy komunikácie.

Súčasťou práce je aj dokumentácia implementovanej knižnice. Dokumentácia je napísaná v anglickom jazyku pre jednoduchšiu ďalšiu distribúciu knižnice. V budúcnosti by sme chceli dokumentáciu uverejniť aj na samostatnej webstránke knižnice, ktorá ešte momentálne nie je súčasťou práce.

Cieľom práce je uľahčiť a čo najviac zjednodušiť komunikáciu mobilných zariadení a príslušenstva, priniesť zaujímavý spôsob z pohľadu jeho vlastností a pomerne nízkych výrobných nákladov. Tento cieľ sa podarilo čiastočne splniť. Odovzdaná knižnica má plne funkčné odosielanie dát, moduláciu. Tento smer funguje a knižnica cez HART modem komunikuje s počítačom.

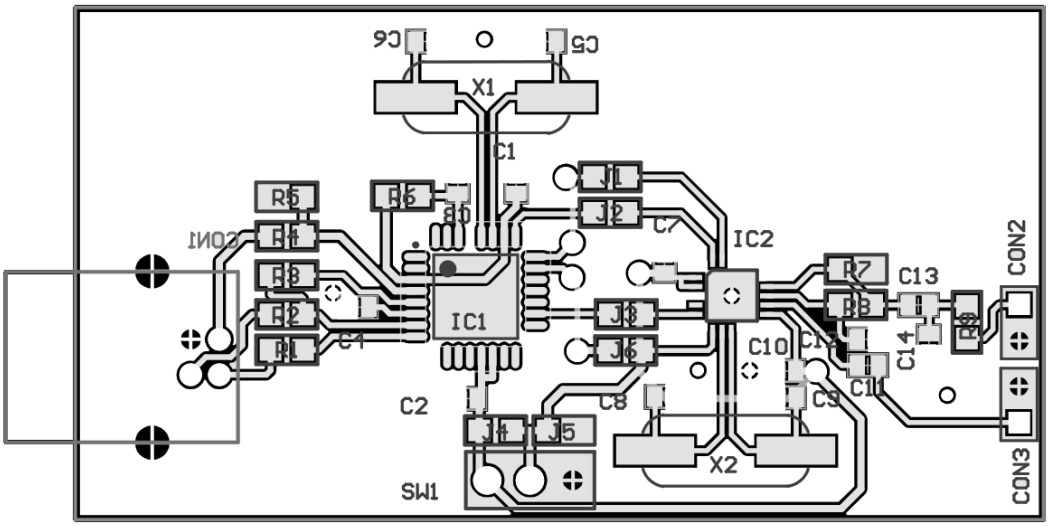
Aj napriek odovzdaniu práce plánujeme ďalší vývoj knižnice, v knižnici momentálne nie sú implementované všetky spomínané funkcie, nevyriešený je problém so synchronizáciou audio signálu pri demodulácií. Budeme sa snažiť tieto nedostatky odstrániť, a implementáciu knižnice dokončiť aby mohla byť profesionálne použitá. Tiež plánujeme uverejnenie demo aplikácie na Android Play.

Výstupom práce je implementovaná knižnica, ktorá je dostupná na priloženom CD aktuálna verzia bude na GIT depozitári [22]. Tento depozitár plánujeme v budúcnosti aktualizovať. V prílohe je tiež dostupná schéma FSK modemu ako tiež návrh plošného spoja.

Pri tejto práci som sa veľa naučil. Od písania pomocou LaTeX až po teóriu skrývajúcu sa za fourierovými transformáciami alebo fungovaním hardvéru ako HART modem. Práca na tejto práci ma bavila a som rád, že mi bolo umožnené sa jej venovať.

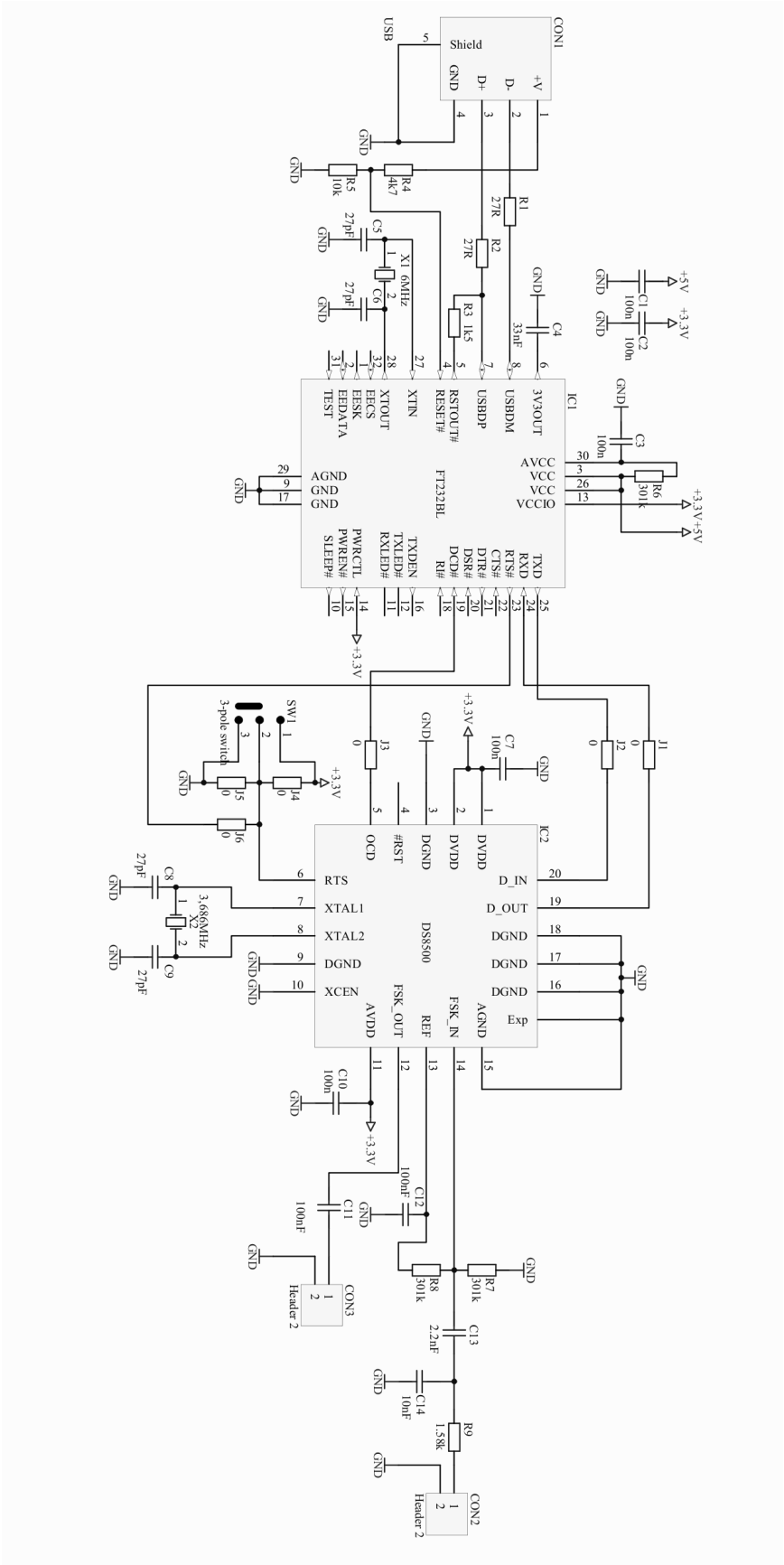
Verím, že táto práca, môže pomôcť jednoduchšie implementovať softvérový modem na Android zariadenia, tiež môže pomôcť s prácou s I/O zariadeniami.

Prílohy



Obr. 3.9: FSK modem čip

Návrh plošného spoja FSK modemu



Obr. 3.10: Schéma FSK modemu

CD so zdrojovými kódmi

Literatúra

[1] Roger L. Freeman. *Telecommunication System Engineering*. John Wiley and Sons, Inc., Hoboken, New Jersey, 2004.

[2] Robots Everywhere. Serial on android using audio port. http://robots-everywhere.com/re_wiki/index.php?title=Serial_on_Android_using_the_audio_port, November 2012.

[3] University of Michigan. Hijack. http://web.eecs.umich.edu/~prabal/projects/hijack/, Jún 2012.

[4] Seedstudio. Seedstudio e-shop. http://www.seeedstudio.com/depot/hijack-development-pack-p-865.html, Máj 2013.

[5] Androino team. Androino blog. http://androino.blogspot.com/, Marec 2012.

[6] HMB-TEC. Radiometer. http://www.hmb-tec.de/HMB-TEC, Jún 2012.

[7] Inter soft. Dosimeter dora. intersofteurasia.ru/eng/, Apríl 2013.

[8] SquareUp. Squareup cardreader. https://squareup.com/, Máj 2012.

[9] Google. Android usb connectivity. http://developer.android.com/guide/topics/connectivity/usb/index.html.

[10] Ian Cameron Smith. Hermit audio analyzer. http://www.androidadb.com/source/moonblink-read-only/HermitAndroid/src/org/hermit/android/io/AudioReader.java.html.

[11] SUDT inc. Accessport program. http://www.sudt.com/en/ap/index.html, Január 2012.

[12] Duncan McLeod. Android takes 70 market share. http://www.techcentral.co.za/android-takes-70-market-share/38179/, Február 2013.

[13] Lars Vogel. Android library projects. http://www.vogella.com/articles/AndroidLibraryProjects/article.html, Marec 2012.

[14] Inc. Maxim Integrated. Tutorial 4676, hart. http://www.maximintegrated.com/app-notes/index.mvp/id/4676, Jún 2010.

[15] Wikipédia. Pcm graphic. http://en.wikipedia.org/wiki/File:Pcm.svg, Máj 2006.

[16] Wikipedia.org. Sampling theorem. http://en.wikipedia.org/wiki/Sampling_theorem, Február 2012.

[17] Paul Reeves. Generate and play tone in android. http://marblemice.blogspot.sk/2010/04/generate-and-play-tone-in-android.html, Apríl 2010.

[18] Aalborg University and group 506. Pc fm-radio receiver. http://kom.aau.dk/group/05gr506/report/node10.html#SECTION04615000000000000000, 2005.

[19] Aalborg University and group 506. Pc fm-radio receiver. http://kom.aau.dk/group/05gr506/report/node27.html, 2005.

[20] James Michael Shima. Fm demodulation using a digital radio and digital signal processing. http://www.hyperdynelabs.com/dspdude/papers/DigRadio_w_mathcad.pdf, Máj 1995.

[21] Alza. Audio kábel. http://www.alza.sk/audio-4polovy-3-5mm-jack-2x-3-5mm-jack-d341926.htm, Máj 2013.

[22] Ján Tančibok. Zdrojové kódy audioserial knižnice. https://sourceforge.net/projects/audioserialport/files/.

[23] Wikipédia. Parity bit. http://en.wikipedia.org/wiki/Parity_bit, Marec 2013.

[24] Inc. Analog Services. About hart – prolog & table of contents. http://www.analogservices.com/about_part0, September 1999.

[25] Bob Watson. Fsk: Signals and demodulation. http://www.xn--sten-cpa.se/share/text/tektext/digital-modulation/FSK_signals_demod.pdf, Január 2001.

[26] Vladimír Trojanovič. Spracovanie fm a am pomocou softvéroveho rádia. http://www.kemt.fei.tuke.sk/personal/drutarovsky/students/pdfs/trojanovic2006.pdf, Máj 2006.